

# **Altair Knowledge Studio® 2022.3 for Apache Spark™**

## User Guide

---

---

## Table of Contents

|                                                          |     |
|----------------------------------------------------------|-----|
| Introduction .....                                       | 7   |
| Summary of New Features .....                            | 9   |
| Technical Information and Client Support.....            | 10  |
| Hardware and Operating System Requirements.....          | 10  |
| Technical Support.....                                   | 14  |
| Copyright Information .....                              | 15  |
| License Administration .....                             | 21  |
| License Types.....                                       | 21  |
| Altair Units Licensing .....                             | 22  |
| Configuring Altair Units Licensing .....                 | 22  |
| Authorize Machine for Managed Altair Units License ..... | 31  |
| Registration-Key Licensing.....                          | 36  |
| License Manager for Registration-Key Licenses .....      | 38  |
| Connecting to Data Mining Engine and Spark Server .....  | 51  |
| Connecting to the Data Mining Engine.....                | 51  |
| Setting the Working Directory .....                      | 53  |
| Spark Server Connection .....                            | 55  |
| Projects and Workflows.....                              | 63  |
| Projects.....                                            | 63  |
| Workflows .....                                          | 67  |
| Data Source Connections, Data Import and Export.....     | 86  |
| Accessing Data in Spark Storage.....                     | 86  |
| Spark Dataframe Link .....                               | 86  |
| Text Import .....                                        | 87  |
| Parquet Import .....                                     | 93  |
| Hive Table Import.....                                   | 97  |
| Data Export .....                                        | 101 |
| Spark Model Link.....                                    | 106 |
| Managing Spark Objects.....                              | 108 |
| Scheduling Analytic Tasks on Spark.....                  | 111 |
| Data Manipulation .....                                  | 118 |
| Aggregate .....                                          | 118 |
| Append .....                                             | 122 |
| Deduplicate.....                                         | 127 |

---

|                                       |     |
|---------------------------------------|-----|
| Filter .....                          | 130 |
| Filtering Join .....                  | 133 |
| Join .....                            | 138 |
| Modify Schema .....                   | 144 |
| Partition.....                        | 146 |
| Reshape .....                         | 149 |
| Sampling.....                         | 158 |
| Missing Value Substitution .....      | 162 |
| Variable Transformations .....        | 166 |
| Expression Editor .....               | 172 |
| Specifying Format.....                | 177 |
| Function Syntax.....                  | 181 |
| Transforming Multiple Variables ..... | 226 |
| Data Profiling.....                   | 228 |
| Weight.....                           | 229 |
| Overview Report.....                  | 232 |
| Dataset Chart.....                    | 239 |
| Data View .....                       | 242 |
| Segment Viewer .....                  | 244 |
| Segment Viewer Options .....          | 245 |
| Segment Viewer Charts .....           | 247 |
| Cross Tabs .....                      | 250 |
| Cross Tabs Chart.....                 | 257 |
| Cross Tabs: Report.....               | 258 |
| Characteristic Analysis .....         | 259 |
| Correlations .....                    | 264 |
| Saved Charts .....                    | 268 |
| Variable Selection.....               | 278 |
| Modeling.....                         | 284 |
| Model Instances.....                  | 284 |
| Decision Trees.....                   | 287 |
| Decision Trees and Splits .....       | 287 |
| Inserting a Decision Tree .....       | 290 |
| Supported Measures .....              | 304 |
| Creating and Modifying Splits .....   | 307 |
| Find Split.....                       | 307 |

|                                                    |     |
|----------------------------------------------------|-----|
| Automatic Grow .....                               | 311 |
| Force Split .....                                  | 313 |
| Ignore Variable .....                              | 320 |
| Erase Child Nodes.....                             | 321 |
| Editing Splits.....                                | 322 |
| Force/Edit Split with Requested DV Percentage..... | 325 |
| Saving Changes .....                               | 329 |
| Navigating between Splits.....                     | 330 |
| Viewing a Decision Tree .....                      | 332 |
| Tree View.....                                     | 332 |
| Tree Map .....                                     | 345 |
| Node Data .....                                    | 346 |
| Split Report.....                                  | 347 |
| Node Report .....                                  | 348 |
| Node Chart .....                                   | 351 |
| Profile Chart.....                                 | 354 |
| Parameters and Attributes.....                     | 358 |
| Copying Splits and Trees.....                      | 368 |
| Modifying Tree Attributes .....                    | 376 |
| Using the Attribute Editor.....                    | 376 |
| Weight in Decision Trees.....                      | 381 |
| Variable Usage .....                               | 384 |
| Intervals/Bins .....                               | 385 |
| Node Datasets .....                                | 386 |
| Accuracy Estimate: Resubstitution .....            | 388 |
| Creating Calculation from Split .....              | 390 |
| Strategy Trees .....                               | 395 |
| Inserting a Strategy Tree .....                    | 397 |
| Viewing a Strategy Tree.....                       | 399 |
| Adding Node Calculations.....                      | 413 |
| Expression Editor .....                            | 416 |
| Expression Helpers.....                            | 421 |
| Average and Sum Aggregate Helpers.....             | 423 |
| Conditional Aggregate.....                         | 427 |
| Computed Conditional Treatment .....               | 431 |
| Index Helper .....                                 | 433 |



---

|                                                           |     |
|-----------------------------------------------------------|-----|
| Discrete Distribution Helper.....                         | 436 |
| Node Cardinality Helper .....                             | 439 |
| Solution Constraint Helper .....                          | 439 |
| Treatments .....                                          | 442 |
| Managing Treatments.....                                  | 443 |
| Generating Treatments From Calculations .....             | 446 |
| Treatment Report .....                                    | 448 |
| Creating Splits.....                                      | 449 |
| Find Split.....                                           | 449 |
| Force Split .....                                         | 450 |
| Navigating between Splits.....                            | 452 |
| Modifying Splits .....                                    | 453 |
| Modifying Tree Attributes .....                           | 458 |
| Using the Attribute Editor.....                           | 458 |
| Variable Usage Types .....                                | 463 |
| Weight.....                                               | 464 |
| Validating and Scoring a Strategy.....                    | 468 |
| Strategy Validation .....                                 | 468 |
| Strategy Scoring .....                                    | 481 |
| Boosted Classification .....                              | 490 |
| Boosted Regression .....                                  | 495 |
| Clustering.....                                           | 500 |
| Linear Regression .....                                   | 504 |
| Logistic Regression .....                                 | 509 |
| Random Forest Classification.....                         | 513 |
| Random Forest Regression.....                             | 518 |
| Spark Model Link .....                                    | 523 |
| Model Results View .....                                  | 525 |
| Model Evaluation and Deployment .....                     | 527 |
| Model Validation .....                                    | 527 |
| Model Analyzer .....                                      | 540 |
| Models Analyzer Views.....                                | 544 |
| Scoring.....                                              | 549 |
| Code Generation .....                                     | 560 |
| Spark Generic Code Node.....                              | 564 |
| Selecting and Reusing Variables with Variable Lists ..... | 568 |

---

---

|                                           |     |
|-------------------------------------------|-----|
| Variable List.....                        | 568 |
| Field Finder.....                         | 576 |
| Exporting Variable Lists.....             | 579 |
| Options and Preferences .....             | 581 |
| General application preferences.....      | 582 |
| Logging and Command Line Tools.....       | 615 |
| Activity Log .....                        | 615 |
| Query Log .....                           | 615 |
| Spark Tasks.....                          | 617 |
| ODBC/Code Log.....                        | 618 |
| Find and Replace Text .....               | 622 |
| Printing .....                            | 624 |
| Printing a Tree .....                     | 624 |
| Printing Charts, Grids, and Reports ..... | 626 |
| Printing to PDF .....                     | 627 |
| Mathematical Formulations .....           | 630 |
| Decision Tree.....                        | 630 |
| References .....                          | 643 |
| Index.....                                | 645 |

All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means without prior written permission of Altair Engineering, Inc. (Altair). KnowledgeSEEKER, Knowledge Seeker, KnowledgeSTUDIO, Knowledge Studio, KnowledgeREADER, KnowledgeMANAGER, and InsightOPTIMIZER, are service marks, trademarks, or registered trademarks of Altair. All other products, logos, or brand names are either service marks, trademarks, or registered trademarks of their respective owners. Altair data mining technology is protected under Canadian, US and international copyright laws and applications have been filed under Canadian, US and international patent laws.

---

# Introduction

Welcome to version 2022.3 of Knowledge Studio for Apache Spark™!

**Knowledge Studio for Apache Spark™** is a comprehensive data science platform integrated with Apache Spark technology to provide advanced data mining and predictive analytics on large-scale distributed data structures such as Hadoop HDFS, Amazon S3, and other storage types supported by Spark. Data can be loaded directly from Hadoop HDFS and ViewFs, Hadoop Archive, Amazon S3, FTP and Network Shares. Hive tables, Text (CSV) and Parquet formats are supported. Data manipulation, profiling, modeling (including interactive decision trees and strategy trees), model evaluation and deployment are supported on Spark. Python code is automatically generated for all operations. Custom programs for data science tasks using any available algorithm libraries in Python can be written within Knowledge Studio workflows. Jupyter Notebook can be launched from within Knowledge Studio workflow for more advanced coding environment. All workflow tasks on Spark can be scheduled for automatic execution.

Page | 7

## Main Differentiators

- intuitive graphical user interface and superior visualization of results
- superior usability and short learning curve that ensure high productivity
- repeatable workflows with automatic data refresh
- best-in-class decision trees
- strategy trees for developing and deploying strategies
- creating and running custom and pre-built programs in Python PySpark

## The Welcome Page

The Welcome page displayed upon starting the application provides links to the following features:

- **Recent Working Directories**  
Connect to any of the recently used Working Directories. The Working Directory locations were saved from your previous sessions.
- **Create a New Project**  
A [project](#) consists of one or more workflows and the results of their execution: a collection of datasets and analysis results such as decision trees and other models, strategy trees, model analyzers, generated code, and other objects. Creating a new project is the first step in your work with the application. The **Create New Project** link in the Welcome screen is a shortcut to the **File | Create Project** menu command.
- **Getting Started**

- **Help:** Click the Help link to access the built-in user guide, also available from the Help menu.
- **User Community:** Join our online community of experts and users to share insights, collaborate with colleagues and find more ways to take full advantage of your product. This link takes you directly to the Knowledge Studio and Knowledge Seeker user forum, where you can participate in online discussions, network with users like yourself, and get answers to your questions with 24/7 support from the community members. The User Community link is also available under the Help menu: **Help | User Community**.

### Learn About Altair Software

To learn more about the products, services, and training we offer, visit our website <https://www.altair.com/data-analytics/> or select "About Altair Data Analytics" from the Help menu. For customer support information, see <https://www.altair.com/data-analytics-support/> and the [Technical Support](#) Help topic.

---

## Summary of New Features

This topic contains a summary of new features introduced in Knowledge Studio for Apache Spark, versions 2021.3 - 2022.3.

Page | 9

### **Versions 2021.1 - 2022.3**

Versions 2021.1 - 2022.3 of Knowledge Studio for Apache Spark are maintenance releases with a number of fixes and no new features.

### **Versions 2021.3 and 2022.0**

Versions 2021.3 and 2022.3 of Knowledge Studio for Apache Spark are maintenance releases. They support Apache Spark™ versions 3.1.x and 2.4.x.

# Technical Information and Client Support

## Hardware and Operating System Requirements

### Overview: Supported Operating Systems

Knowledge Studio for Apache Spark is available in the standalone and client/server configurations for the following operating systems:

| KS Workstation (Standalone or Client)                                                         | KS Server                                                                                              |
|-----------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| Windows 10<br>Windows 11<br>Windows Server 2016<br>Windows Server 2019<br>Windows Server 2022 | Windows Server on x86-64 platform<br>Windows Server 2016<br>Windows Server 2019<br>Windows Server 2022 |
|                                                                                               | <u>Linux on x86-64 platform:</u><br>Red Hat Enterprise Linux 7 and 8<br>CentOS 7 and 8                 |

The product is supported only on 64-bit systems.

Knowledge Studio for Apache Spark requires Apache Spark server. System requirements for Knowledge Studio for Apache Spark are described below.

### System Requirements

This section describes the minimum and recommended system requirements for KS Workstation in the desktop configuration and KS Server and Workstation in the client/server configuration.

**Microsoft .NET Framework 4.6.1** (or higher) and **.NET Framework 3.5 Service Pack 1** are prerequisites for KS Workstation. The KS Workstation installation package includes web installers for these components and installs them automatically if they are not found on the system.

**Microsoft Visual C++ 2015-2022 Redistributable** package is required for KS Workstation and KS Server. Both KS Workstation and KS Server installers include the web installer for Microsoft Visual

C++ 2015-2022 Redistributable and installs it unless the same or a higher version is already present.

KS Workstation includes KS PDF Printer, which enables the Copy to PDF functionality for exporting Altair trees, grids, charts and reports to PDF. KS PDF Printer uses Amyuni PDF Converter by [Amyuni Technologies, Inc.](#) Note: In case KS PDF Printer is removed from the system, it can be restored by running AngossPDFInstall.cmd from the folder C:\Program Files\Altair Data Analytics\KS Workstation 2022.3.0\bin.

The following are the minimum and recommended system requirements for all configurations:

## 1. Standalone Configuration

| KS Workstation 2022.3 – Standalone                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Minimum                                                                                                                                                                                                                                                                                                          | Recommended                                                                                                                                                                                                                                                                                                                                      |
| <ul style="list-style-type: none"><li>Windows 10 or 11 (64-bit)</li><li>x86-64-based CPU, 1.5 GHz</li><li>4 GB RAM</li><li>Disk space:<ul style="list-style-type: none"><li>3 GB for application program files</li><li>1+ GB for user's Altair projects</li></ul></li><li>Monitor resolution: 1152x864</li></ul> | <ul style="list-style-type: none"><li>Windows 10 (64-bit)</li><li>Quad-core x86-64-based CPU, 2 GHz, or better</li><li>8 GB RAM or more</li><li>Disk space:<ul style="list-style-type: none"><li>3 GB for application program files</li><li>1+ GB for user's Altair projects</li></ul></li><li>Monitor resolution: 1680x1050 or better</li></ul> |

## 2. Client/Server Configuration

| KS Workstation 2022.3 – Client                                                                                                                                                                                           |                                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Minimum                                                                                                                                                                                                                  | Recommended                                                                                                                                                                                                                                    |
| <ul style="list-style-type: none"><li>Windows 10 or 11 (64-bit)</li><li>x86-64-based CPU, 1.5 GHz</li><li>4 GB RAM</li><li>Disk space: 3 GB for application program files</li><li>Monitor resolution: 1152x864</li></ul> | <ul style="list-style-type: none"><li>Windows 10 (64-bit)</li><li>x86-64-based CPU, 2 GHz, or better</li><li>8 GB RAM or more</li><li>Disk space: 3 GB for application program files</li><li>Monitor resolution: 1680x1050 or better</li></ul> |

| KS Server 2022.3                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Minimum                                                                                                                                                                                                                                                                                         | Recommended                                                                                                                                                                                                                                                                                                                                           |
| <ul style="list-style-type: none"> <li>Windows Server 2016, 2019, 2022 or Red Hat Enterprise Linux 7 or 8; CentOS 7 or 8</li> <li>An x86-64-based CPU, 1.5 GHz</li> <li>8 GB RAM</li> <li>Disk space: 3 GB for application program files</li> <li>1+ GB per user for Altair projects</li> </ul> | <ul style="list-style-type: none"> <li>Windows Server 2016 or 2019; Red Hat Enterprise Linux 7 or 8; CentOS 7 or 8</li> <li>Multi-core x86-64-based CPU(s) with at least 4 cores in total, 2 GHz, or better</li> <li>16 GB RAM or more</li> <li>Disk space: 3 GB for application program files</li> <li>1+ GB per user for Altair projects</li> </ul> |

**\*\* Note:** System requirements for the server depend on the maximum number of users expected to be using KS Server concurrently. To ensure that the software is compatible with your environment and will deliver acceptable performance, please fill in the Client Installation Questionnaire. If you need our help in determining the server size sufficient for the desired number of concurrent users of KS Server, please contact [Altair Support](#).

Please refer to the Deployment Guide for installation guidelines.

### 3. Apache Spark Server and Hadoop Cluster Requirements

The types of distributed storage supported by Knowledge Studio for Apache Spark are the same as those supported by Apache Spark. The types of distributed storage supported by Knowledge Studio for Apache Spark are the same as those supported by Apache Spark. The product is supported for any Hadoop distribution satisfying the Apache Spark server requirements listed below.

Examples of supported Hadoop distributions:

- **Cloudera CDH.** Supported versions: All CDH versions compatible with Apache Spark 2.4 and 3.1: (CDH 6.2 and higher; See “CDS Powered By Apache Spark Requirements” [https://www.cloudera.com/documentation/spark2/latest/topics/spark2\\_requirements.html](https://www.cloudera.com/documentation/spark2/latest/topics/spark2_requirements.html))
- **MapR Converged Data Platform.** Includes support for Spark Standalone, Spark on YARN, and Spark on Mesos. Supported versions of MapR Ecosystem Packs (MEPs): All versions that support Spark 2.4 and 3.1 (MEP 6.2.0 and higher). See <https://docs.datafabric.hpe.com/62/EcosystemRN/SparkRN.html>.

Note: Using Spark cluster configurations with dynamic allocation is not recommended for Knowledge Studio for Apache Spark.



System requirements for Knowledge Studio software components are described in items 1 and 2 above. The following are the requirements for the **Apache Spark server**.

### Apache Spark server requirements

Page | 13

- Apache Spark versions 2.4 and 3.1
  - Cluster management system options: Standalone (native Spark cluster) or [Hadoop YARN](#) or [Apache Mesos](#)
- [Python](#) version **3.6 or higher**, 64-bit (Python programming language)
- Required Python packages:
  - Pandas: the [Python Data Analysis Library](#) provides data structures and analysis tools for Python.
  - [NumPy](#): the package for scientific computing with Python
  - [SciPy](#): the fundamental library for scientific computing (version 0.19.1 or higher)
  - [Scikit-learn](#): tools for data mining and data analysis (version 0.19.0 or higher)
  - [PyArrow](#): Python API of Apache Arrow (version 0.9.0 or higher)
- [Jupyter Notebook](#) version **4 or higher**
- [JupyterHub](#) version **1.0 or higher**

For **Jupyter Notebook** prerequisites and installation instructions, see <https://jupyter.readthedocs.io/en/latest/install.html>

For **JupyterHub** prerequisites and installation instructions, see <https://jupyterhub.readthedocs.io/en/latest/>

Note: The use of JupyterHub as a multi-user Jupyter notebook server is recommended. If you prefer not to use JupyterHub, you can use basic authentication implemented by other mechanisms and configure them according to your security, authentication, and file system access requirements.

Examples of distributed storage clusters supported by Altair:

- [Amazon Web Services Cloud \(AWS\)](#) (note that Spark environment on Amazon AWS includes Jupyter Notebook and JupyterHub)

- [Microsoft Azure HDInsight](#) (note that Spark environment on Azure has Jupyter, but no JupyterHub).
- Custom clusters satisfying the above requirements for Apache Spark.

For your convenience, Altair provides a script for installing the necessary prerequisites, including the required Python packages. The script is provided upon request. You will need to modify it as you see fit to install the components that you need. Please contact Customer Support at [dasupport@altair.com](mailto:dasupport@altair.com).

Page | 14

For the client/server configuration of Knowledge Studio for Apache Spark, the range of TCP ports 6060 – 6090 on the server must be open for inbound connections from Knowledge Studio client machines. In the standalone (desktop) configuration, Knowledge Studio for Apache Spark also uses ports in this range, but this is confined to localhost.

Users of Knowledge Studio for Apache Spark can launch Jupyter notebook from within the Knowledge Studio application. For this integration feature to work, the URL for the Jupyter notebook server must be accessible from the local machine where KS Workstation is installed. In the client/server mode, it must be accessible from both the client and server machines.

Make sure that the time zone is set correctly on the Spark server. This is important for the functions that deal with timestamp data type.

Please refer to the Deployment Guide for installation guidelines.

## Technical Support

Licensed users seeking technical support for the application should email questions to [dasupport@altair.com](mailto:dasupport@altair.com) or contact us at 1-888-715-7391 (toll-free in North America) or +1-416-593-2417.

We welcome your suggestions on how to make our application easier to understand and use. Please forward your comments and suggestions to [dasupport@altair.com](mailto:dasupport@altair.com).

## New Releases and Updates

We plan future updates, upgrades, and new releases which will be available to licensed users. Visit our web site <https://www.altair.com/knowledge-studio/> for the latest news about our products. Contact Customer Support if you have questions about upgrading your software.

## Documentation

The following additional resources are available for customers and evaluating clients on the Altair software download site:

- **Release notes** (PDF)

- **Deployment Guide** – The installation and configuration guide for all configurations of Knowledge Studio for Apache Spark (PDF).
- **User Guide** – PDF version of the Help system content.
- **KS Server Administration Utilities.** – A user guide to simple utilities for usage monitoring and user process administration for the client/server configuration of Altair Data Analytics applications (PDF).

## Copyright Information

Copyright © 1994-2022 Altair Engineering Canada, Ltd. All rights reserved.

Copyright © 2018-2022 Altair Engineering Inc. All rights reserved.

This computer program and related documentation are protected under national and international copyright and intellectual property laws. Use hereof is subject to the end user license agreement between the licensed named user and Altair Engineering, Inc. Unauthorized reproduction, distribution or other use of this computer program, including use hereof by any person other than the authorized licensee hereof, is expressly prohibited under such laws, may result in serious civil and criminal penalties, and in the event of violations will be prosecuted to the fullest extent of the law.

Also, portions of the software utilize the following copyrighted material, the use of which is hereby acknowledged:

Copyright © 2008-2022 Software FX, Inc. All Rights Reserved. **Chart FX** is a registered trademark of Software FX, Inc.

Copyright © 1998-2011 Crownwood Software, Inc. All Rights Reserved. (Registered number 03592253 / VAT No. 711126780). **DotNet Magic** is a registered trademark of Crownwood Software, Ltd.

Copyright ©1992-2022 Infragistics, Inc., 2 Commerce Drive, Cranbury, NJ 08512. All rights reserved. Infragistics, **NetAdvantage**, and the Infragistics logo are registered trademarks of Infragistics, Inc.

Copyright ©2014-2022 Northwoods Software Corp. (**GoDiagram for WinForms**) All rights reserved. Northwoods Software® is a Registered Trademark of Northwoods Software Corporation.

Copyright ©2001-2022 Amyuni Technologies Inc. (**Amyuni PDF Converter**) All Rights Reserved. Amyuni and the Amyuni logo are trademarks of Amyuni Technologies Inc.

Copyright ©1986-2022 Circle Systems, Inc. (**Stat/Transfer** drivers for data transfer) All rights reserved.

Copyright © 2003-2022 Tableau Software. All Rights Reserved. **Tableau®** and **Tableau Software®** are registered trademarks of Tableau Software Inc. License agreements and terms: <http://www.tableau.com/legal>

Page | 16

Copyright © 2022 ActiveState Software. All rights reserved. **ActivePython** 3.8 is used under an enterprise license with ActiveState. The ActivePython distribution includes the following Python packages subject to their respective copyrights and end-user license agreements:

absl\_py 0.14.1, alembic 1.4.1, astunparse 1.6.3, autonormalize 1.0.1, azure\_core 1.23.0, azure\_storage\_blob 12.11.0, boto3 1.17.22, botocore 1.20.112, cachetools 4.2.4, certifi 2021.10.8, cffi 1.15.0, charset\_normalizer 2.0.7, click 8.0.3, cloudpickle 2.0.0, colorama 0.4.4, cryptography 36.0.2, Cython 0.29.17, dask 2021.9.1, databricks\_cli 0.16.4, distributed 2021.9.1, docker 5.0.3, entrypoints 0.4, featuretools 0.23.0, Flask 2.1.0, fsspec 2021.10.0, gast 0.3.3, gitdb 4.0.9, GitPython 3.1.27, google\_auth\_oauthlib 0.4.6, google\_auth 2.3.0, google\_pasta 0.2.0, gorilla 0.4.0, grpcio 1.41.0, h5py 2.10.0, HeapDict 1.0.1, idna 3.3, imbalanced\_learn 0.8.0, importlib\_metadata 4.11.3, isodate 0.6.1, itsdangerous 2.1.2, Jinja2 3.0.2, jmespath 0.10.0, joblib 1.1.0, Keras\_Preprocessing 1.1.2, Keras 2.4.3, locket 0.2.1, Mako 1.2.0, Markdown 3.3.4, MarkupSafe 2.0.1, mlflow 1.11.0, msgpack 1.0.2, msrest 0.6.21, numpy 1.18.5, oauthlib 3.1.1, opt\_einsum 3.3.0, packaging 21.0, pandas 1.2.3, partd 1.2.0, patsy 0.5.2, pip 21.3, pip 22.0.4, pmdarima 1.7.1, prometheus\_client 0.13.1, prometheus\_flask\_exporter 0.19.0, protobuf 3.18.1, psutil 5.8.0, pyasn1\_modules 0.2.8, pyasn1 0.4.8, pycparser 2.21, pyparsing 2.4.7, python\_dateutil 2.8.2, python\_editor 1.0.4, pytz 2021.3, pywin32 227, PyYAML 6.0, querystring\_parser 1.2.4, requests\_oauthlib 1.3.0, requests 2.26.0, rsa 4.7.2, s3transfer 0.3.7, scikit\_learn 0.24.2, scipy 1.7.1, setuptools 49.6.0, six 1.16.0, smmap 5.0.0, sortedcontainers 2.4.0, SQLAlchemy 1.3.13, sqlparse 0.4.2, statsmodels 0.11.1, tabulate 0.8.9, tblib 1.7.0, tensorboard\_data\_server 0.6.1, tensorboard\_plugin\_wit 1.8.0, tensorboard 2.7.0, tensorflow\_gpu\_estimator 2.3.0, tensorflow\_gpu 2.3.1, termcolor 1.1.0, threadpoolctl 3.0.0, toolz 0.11.1, tornado 6.1, tqdm 4.62.3, typing\_extensions 4.1.1, urllib3 1.26.7, waitress 2.1.1, websocket\_client 1.3.2, Werkzeug 2.0.2, wheel 0.37.0, wrapt 1.13.2, xgboost 1.4.2, zict 2.0.0, zipp 3.7.0.

Copyright © 2018-2022 Intel Corporation. **Intel® Math Kernel Library (Intel® MKL)** is used under the terms of the Intel Simplified Software License: <https://software.intel.com/en-us/license/intel-simplified-software-license>

**Bonmin** (Basic Open-source Nonlinear Mixed INteger programming) is a numerical analysis library for solving general MINLP (Mixed Integer NonLinear Programming) problems. Bonmin version 1.5 is used under the terms of the Eclipse Public License 1.0 (EPL-1.0) <https://www.eclipse.org/legal/epl-v10.html>. Bonmin project description: <https://projects.coin-or.org/Bonmin>. Source code:

<https://projects.coin-or.org/Bonmin/browser> and <https://github.com/coin-or/Bonmin>.

Copyright © Microsoft Corporation. **Microsoft Automatic Graph Layout, MSAGL**. All rights reserved. MSAGL version 2.1 is used under the terms and conditions of the MIT License: <https://github.com/Microsoft/automatic-graph-layout/blob/master/LICENSE>. Source code: <https://github.com/Microsoft/automatic-graph-layout>.

Page | 17

Copyright © 2002-2012 Davide Icardi, Darius Damalakas. **SourceGrid** version 2.0.9.0 is used under the terms of the MIT License: <https://github.com/siemens/sourcegrid/blob/master/LICENSE>. Source code: <https://github.com/siemens/sourcegrid>. See also <https://www.codeproject.com/Articles/3531/SourceGrid-Open-Source-C-Grid-Control>.

Copyright © 1998-2022 The OpenSSL Project. All rights reserved. Software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>) is used under the terms and conditions of the OpenSSL license <http://www.openssl.org/source/license.html>. Source code: <https://github.com/openssl/openssl>. OpenSSL is a registered trademark owned by OpenSSL Software Foundation.

Copyright © 2014-2022 The OpenBSD Project. All rights reserved. **LibreSSL** version 2.5.5 is used under the terms and conditions of the Apache License version 1.0 <https://www.apache.org/licenses/LICENSE-1.0>, 4-clause BSD License <https://directory.fsf.org/wiki/License:BSD-4-Clause>, and ISC license <https://www.isc.org/downloads/software-support-policy/isc-license/>. Source code: <https://github.com/libressl-portable/>.

Copyright © 1998 John E. Bossom. Copyright © 1999, 2006 Pthreads-win32 contributors (<http://sources.redhat.com/pthreads-win32/contributors.html>). **Pthreads-win32** version 2.9.0.0 - POSIX Threads Library for Win32 - is used under the terms of the GNU Lesser General Public License (LGPL) version 2.1: <https://www.sourceware.org/pthreads-win32/copying.lib.html>. Source code: <ftp://sourceware.org/pub/pthreads-win32>.

Copyright © 2022, Applied Informatics Software Engineering GmbH and Contributors. **POCO C++ Libraries** version 1.11.0 are used under the terms and conditions of the Boost Software License 1.0 <https://pocoproject.org/about.html#license>. Portions of the POCO C++ Libraries utilize copyrighted material acknowledged at

<https://pocoproject.org/docs/90100-Acknowledgements.html>. POCO

Project description and source: <https://pocoproject.org/>.

Copyright © 2013 John Dickinson. The **C# wrapper for SHBrowseForFolder** is used under the terms and conditions of the MIT License <http://opensource.org/licenses/mit-license.php>. Source code and description: <http://unafaltadecompression.blogspot.ca/2013/04/browsing-for-files-and-folders-c.html>.

Page | 18

The **libiconv** library version 1.14 for text encoding conversion is used under the terms of the GNU Lesser General Public License (LGPL):

<http://www.gnu.org/licenses/lgpl.html>. Description:

<https://www.gnu.org/software/libiconv/>. Source code:

<https://savannah.gnu.org/projects/libiconv/>

The .NET Gold Parser Engine **bsn-goldparser** version 5.2.0 - is used under the terms of the GNU Lesser General Public License (LGPL)

<http://www.gnu.org/licenses/lgpl.html>. Description:

<http://www.goldparser.org/doc/index.htm>. Source code:

<https://github.com/siriusch/bsn.GoldParser>.

Copyright © 2011, 2013 Vincent Tan Wai Lip. The **SpreadsheetLight** library version 3.4 (27 July 2013) is used under the terms of the MIT License

<http://spreadsheetlight.com/license>. Source code:

<http://spreadsheetlight.com/download>.

The linear programming solver **CLP (COIN-OR LP)** version 1.14.0 is used under the terms of the Eclipse Public License 1.0 (EPL-

1.0): <https://github.com/coin-or/Clp/blob/master/LICENSE>. Source code:

<https://github.com/coin-or/Clp>.

Copyright © 2018-2022 Newtonsoft. Copyright © 2007 James Newton-King. A JSON framework for .NET, **JamesNK/Newtonsoft.Json** version 6.0 is used under the terms of the MIT License

<https://github.com/JamesNK/Newtonsoft.Json/blob/master/LICENSE.md>.

Description: <http://www.newtonsoft.com/json>. Source code:

<https://github.com/JamesNK/Newtonsoft.Json>.

**DocX** version 1.0 is a .NET library for manipulating Microsoft Word files used under the terms of the Microsoft Public License (Ms-PL):

<https://opensource.org/licenses/MS-PL>. Description and source code:

<https://github.com/xceedsoftware/DocX>.

Copyright © 1998-2022 Neil Hodgson ([neilh@scintilla.org](mailto:neilh@scintilla.org)). All rights reserved. **Scintilla** is a free source code editing component used under the terms of the license for Lexilla, Scintilla, and SciTE

<https://www.scintilla.org/License.txt>. Source code:  
<https://sourceforge.net/projects/scintilla>.

Copyright © 2006, 2007 Martin R. Gagné ([martingagne@gmail.com](mailto:martingagne@gmail.com)). All rights reserved. The loading circle animation component v1.0 for .NET is used under the terms of the Code Project Open License (CPOL):  
<https://www.codeproject.com/info/cpol10.aspx>. Description and source code: <https://www.codeproject.com/Articles/14841/How-to-write-a-loading-circle-animation-in-NET>.

Page | 19

Copyright © 2015 Elad Zelingher, CodeSharp. All rights reserved. A C# implementation of **WAMP** (The Web Application Messaging Protocol) version 18.11.1 is used under the terms of The FreeBSD License:  
<https://github.com/Code-Sharp/WampSharp/blob/master/LICENSE>.  
Description and source code: <https://github.com/Code-Sharp/WampSharp>.

A Windows SSPI SSL implementation for CUPS version 2.2.9 is used under the terms and conditions of the Apache License Version 2.0  
<https://www.cups.org/doc/license.html>. Source code:  
<https://github.com/apple/cups>.

Icons made by Freepik (<http://www.freepik.com>) from Flaticon (<http://www.flaticon.com>) are used under the terms of the Creative Commons BY 3.0 license: <http://creativecommons.org/licenses/by/3.0/>

Icons of The Noun Project (<http://www.thenounproject.com>) are used under the Terms of Use: <https://thenounproject.com/legal/>, Creative Commons Attribution (CC BY) and Public Domain (PD) licenses  
<https://creativecommons.org/licenses/by/3.0/>,  
<https://creativecommons.org/publicdomain/mark/1.0/>.

Apache, Apache Spark, Spark and the Spark logo are trademarks of the Apache Software Foundation. Knowledge Studio for Apache Spark provides predictive analytics on distributed storage types supported by Apache Spark.

Microsoft, MS, Visual C++, C#, Excel, Access, Windows, and Windows NT are registered trademarks, and ActiveX, Developer Studio, DirectX, and Visual J++ are trademarks of Microsoft Corporation.

All other products, logos, or brand names are either service marks, trademarks, or registered trademarks of their respective owners.

Altair would like to thank the UCI Repository of Machine Learning Databases for the "Census" database distributed by us and used throughout the tutorial. Reference: Merz, C.J., & Murphy, P.M. (1998).

---

UCI Repository of machine learning databases  
[\[http://www.ics.uci.edu/~mlearn/MLRepository.html\]](http://www.ics.uci.edu/~mlearn/MLRepository.html). Irvine, CA:  
University of California, Department of Information and Computer  
Science.

The above third-party components that are licensed under, and subject to the permission of the MIT License or the Lesser General Public License (LGPL) are provided "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THESE COMPONENTS OR THE USE OF OTHER DEALINGS IN THESE COMPONENTS.

Page | 20

See also the [End-User License Agreement for Altair Data Analytics Software](#).



---

# License Administration

## License Types

Two licensing systems are supported for Altair Data Analytics applications:

- Altair Units licensing
- Registration-key licensing

The licensing system that your product is using depends on the terms of your contract.

### Altair Units Licensing

Altair Units is a value-based license management system enabling metered usage of an entire suite of products. All Altair products have a capability to use a single pool of recyclable Altair Units.

If you purchased Knowledge Studio with an Altair Units license, you will receive a license file and Altair Licensing Quick Installation Guide, which describes the installation and configuration of Altair License Manager with the License Server. The process of configuring the end users' machines with Knowledge Studio or Knowledge Seeker to connect to Altair License Server is described in the topic [Configuring Altair Units Licensing](#).

Note: The Text Analytics Add-on is only available with registration-key licensing. If your product includes this feature, the add-on license is installed using the instructions in the topic [License Manager for Registration-Key Licenses](#).

### Registration-Key Licensing

[Registration keys](#) issued by Altair License Administrator are used by the end users to acquire licenses online from the [Altair License Fulfillment Server \(ALFS\)](#) – a secure self-service license generation website.

The licenses are acquired from ALFS using the [License Manager](#) module available in the Help menu of Knowledge Studio. Registration-key licenses can also be acquired by email when network access to ALFS is not available.

Registration-key licenses can be subscription-based and evaluation licenses.

#### Named User Licenses

Named-User licenses are user-based node-locked licenses available for both the standalone and client/server configurations. They will only operate on a specific machine for a specific user ID.

### Concurrent User Licenses (Legacy)

Page | 22

Concurrent user licenses are available only for the client/server configuration of Altair software or the standalone configuration on a server accessed via Remote Desktop Connection or Citrix. They are node-locked licenses that will only operate on a specific server for no more than a preset number of concurrent user connections. The total number of potential users is not limited. Concurrent licenses are suitable for customers who will have a high turnover of end users of the application.

A concurrent license has a restriction on the number of users that can simultaneously work with the software in client/server mode, but has no restriction on the identity of the user. Anyone having the client component (KS Workstation software) installed on their machine can connect to the server component and work with it, as long as the number of users connected at the same time does not exceed the limit imposed by the license.

See also:

[Configuring Altair Units Licensing](#)

[Altair License Fulfillment Server \(ALFS\)](#)

[License Manager for Registration-Key Licenses](#)

## Altair Units Licensing

### Configuring Altair Units Licensing

If you purchased Knowledge Studio with an Altair Units license, the application must be configured to connect to an Altair License Server (unless you received a Standalone license). This topic describes the process of configuring the Knowledge Studio host machine to connect to an Altair License Server.

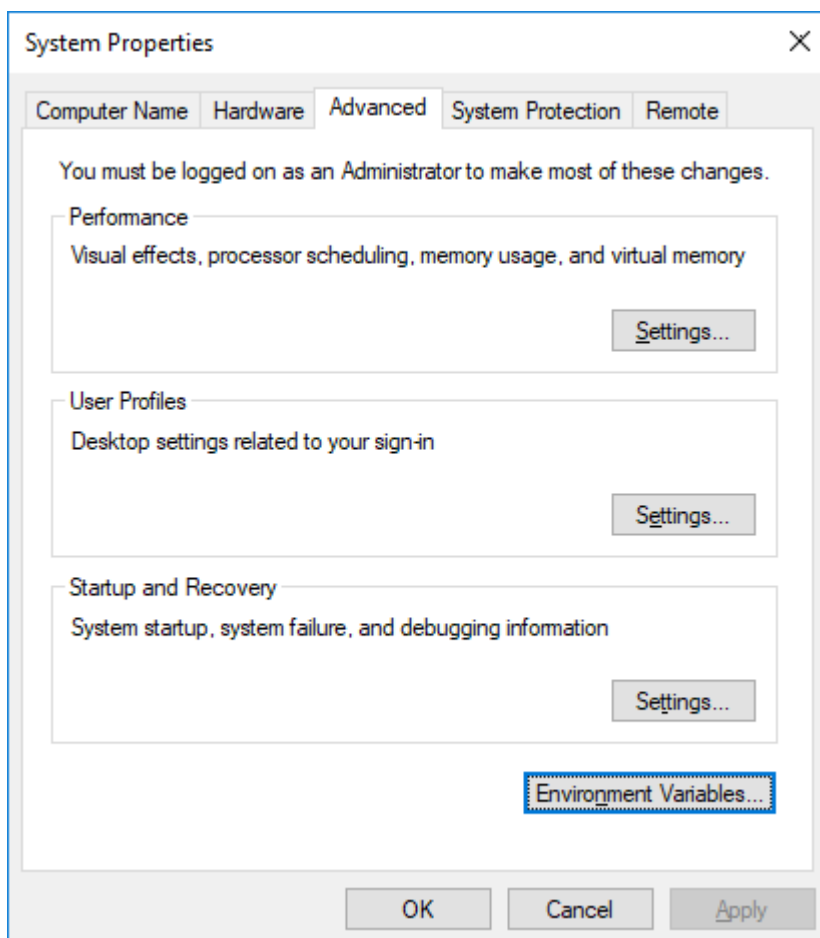
- If you are using Altair License Server on premises, the installation of the Altair License Manager software on the server is described in the Altair License Quick Installation Guide that your organization received from Altair. In what follows, it is assumed that Altair License Manager is already installed and configured.
- If you are using a Managed Altair Units license, where the License Server is hosted by Altair, follow the steps in the topic [Authorize Machine for Managed Altair Units License](#) before performing the configuration steps below.

There are three Knowledge Studio configurations:

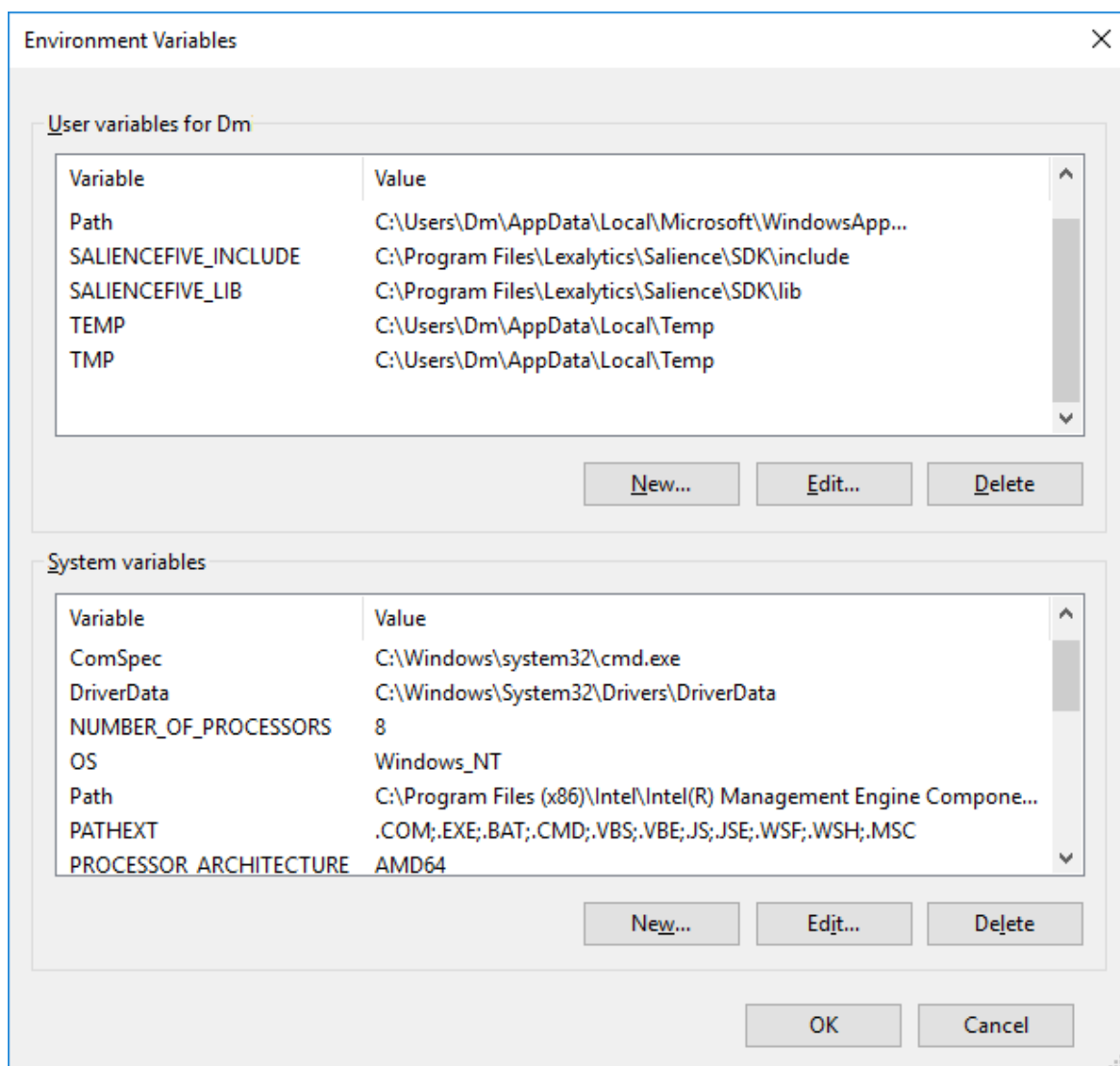
- Standalone
- Client/Server for Windows Server
- Client/Server for Linux

## Standalone

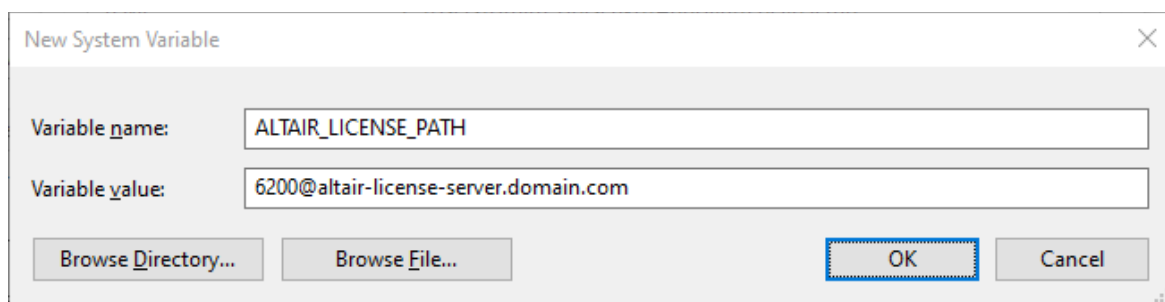
From the Windows Control Panel, select **System** and click **Advanced System Properties**.



In the Advanced tab of the System Properties dialog, click the **Environment Variables** button.

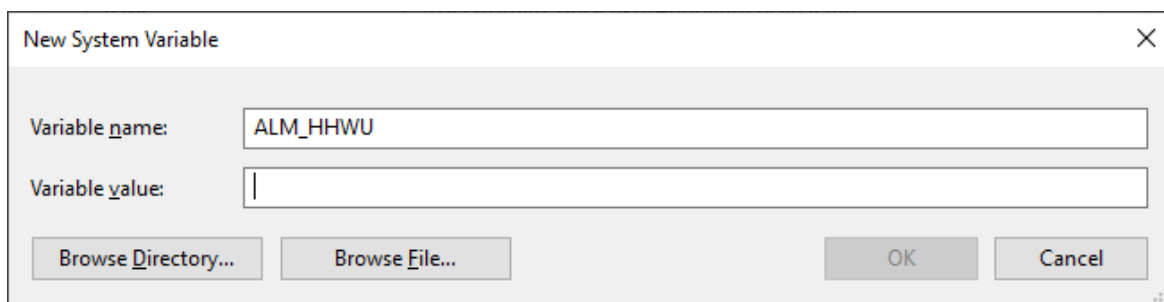


Click **New** in the *System variables* section or in the *User variables* section, depending on whether the Altair license server connection is required for all users on this machine or only the user ID currently logged in.



- Set *Variable name* to **ALTAIR\_LICENSE\_PATH**.
- For the *Variable value* field:
  - If you are using Altair License Server on premises, enter its address in the format **NNNN@<server-name>**, where NNNN is the port number and <server-name> is the Altair License Server host name. Example: **6200@altair-license-server.domain.com**
  - If you are using Managed Altair Units, set the value to **False**

Click **New** in the same section where you defined ALTAIR\_LICENSE\_PATH.



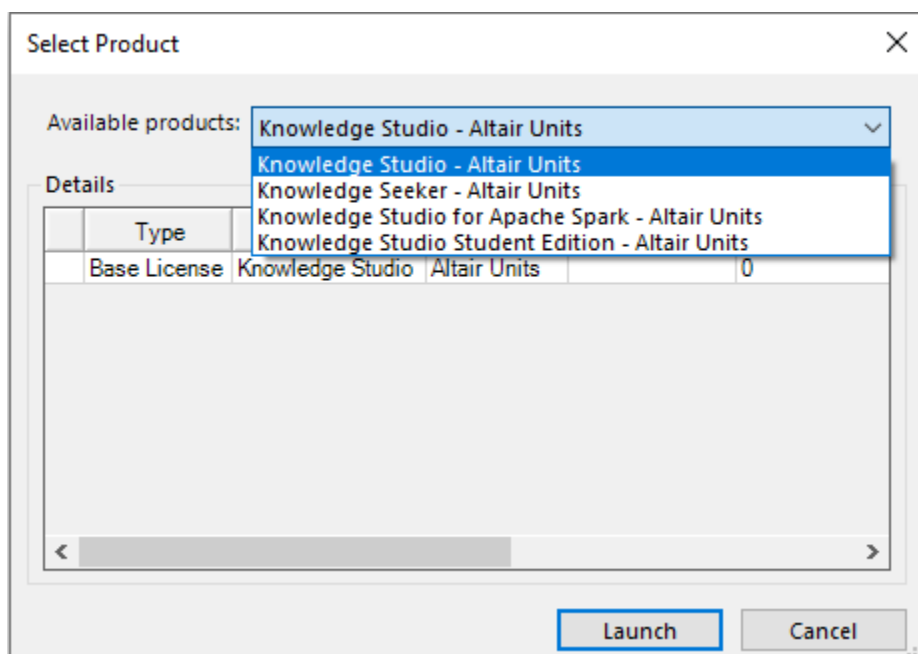
The image shows a 'New System Variable' dialog box. It has a title bar with a close button (X). Inside, there are two text input fields. The first is labeled 'Variable name:' and contains the text 'ALM\_HHWU'. The second is labeled 'Variable value:' and is empty. Below the input fields are four buttons: 'Browse Directory...', 'Browse File...', 'OK', and 'Cancel'.

- Set *Variable name* to **ALM\_HHWU**
- To use Managed Altair Units, set the value of ALM\_HHWU to **True**. Otherwise, set it to **False**

Click **OK** in all dialogs to save the changes and exit the Advanced System Properties dialog.

Start the KS Workstation application.

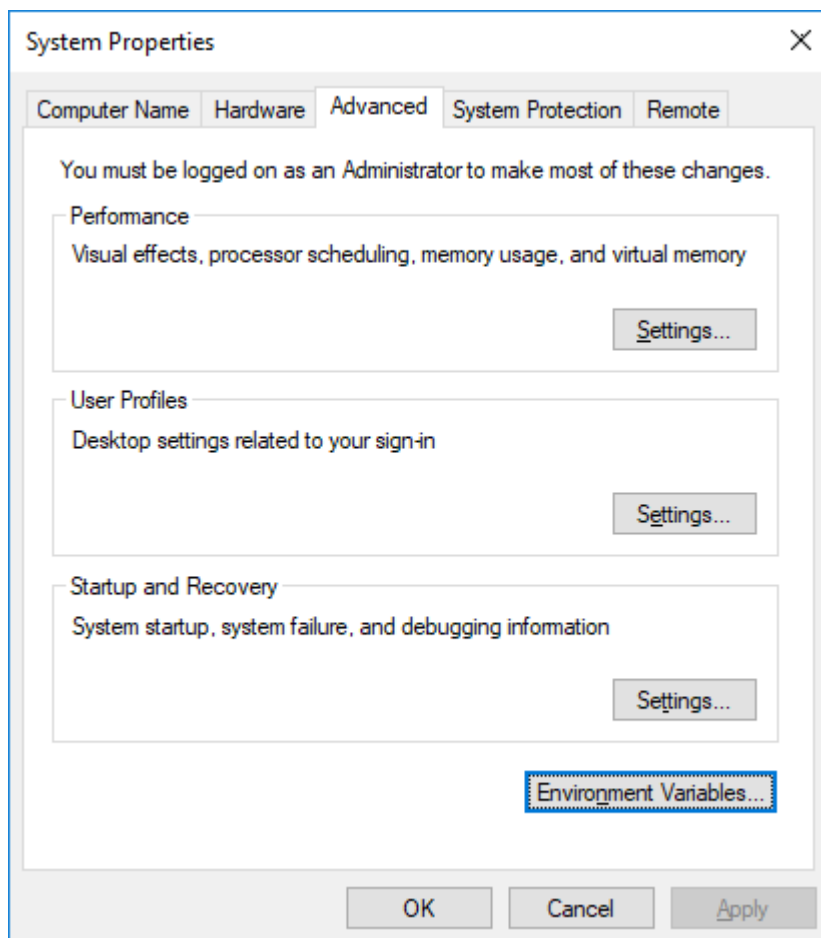
The Product Selection window will prompt you to select one of the three products: Knowledge Seeker, Knowledge Studio, or Knowledge Studio for Apache Spark.



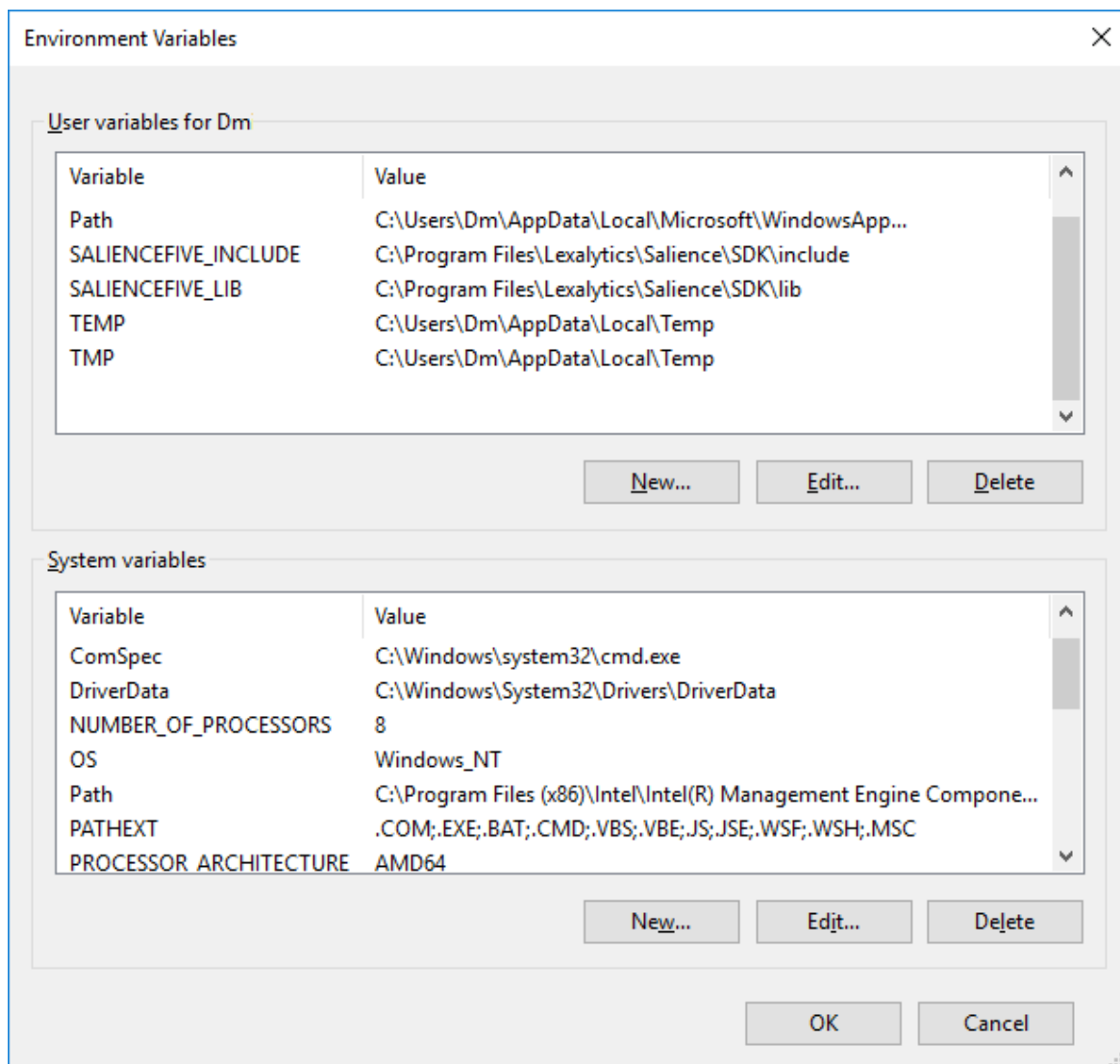
Select Knowledge Studio for Apache Spark and click **Launch**.

## Client/Server for Windows

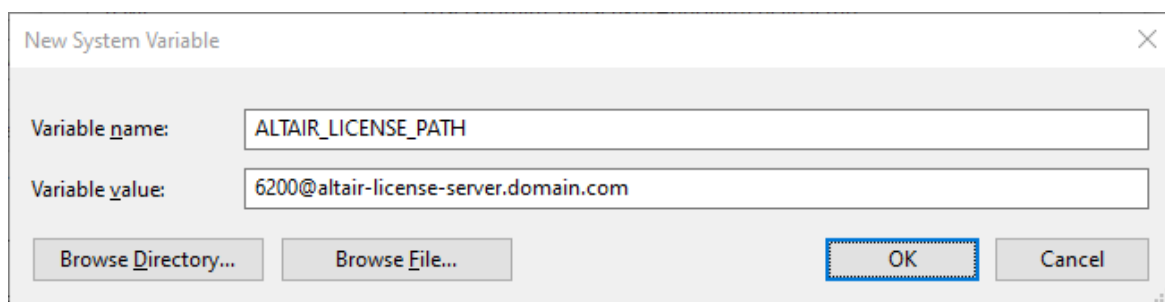
From the Windows Server Control Panel, select **System** and click **Advanced System Properties**.



In the Advanced tab of the System Properties dialog, click the **Environment Variables** button.



Click the **New** button in the *System variables* section.

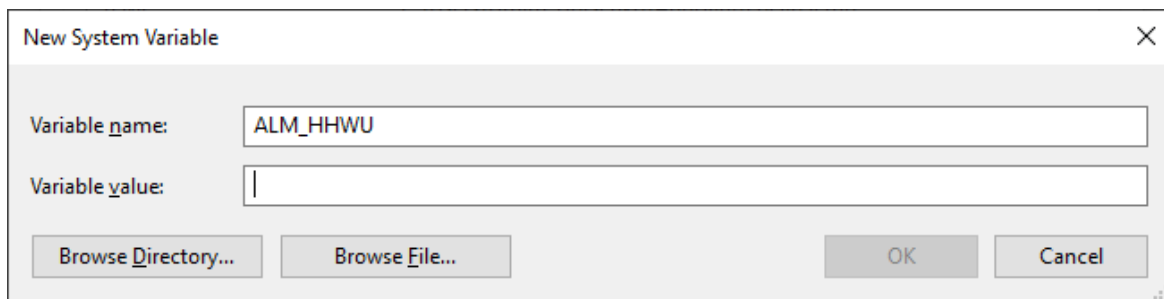


- Set *Variable name* to **ALTAIR\_LICENSE\_PATH**.



- For the *Variable value* field:
  - If you are using Altair License Server on premises, enter its address in the format **NNNN@<server-name>**, where NNNN is the port number and <server-name> is the Altair License Server host name. Example: **6200@altair-license-server.domain.com**
  - If you are using Managed Altair Units, set the value to **False**

Click **New** in the same section where you defined ALTAIR\_LICENSE\_PATH.



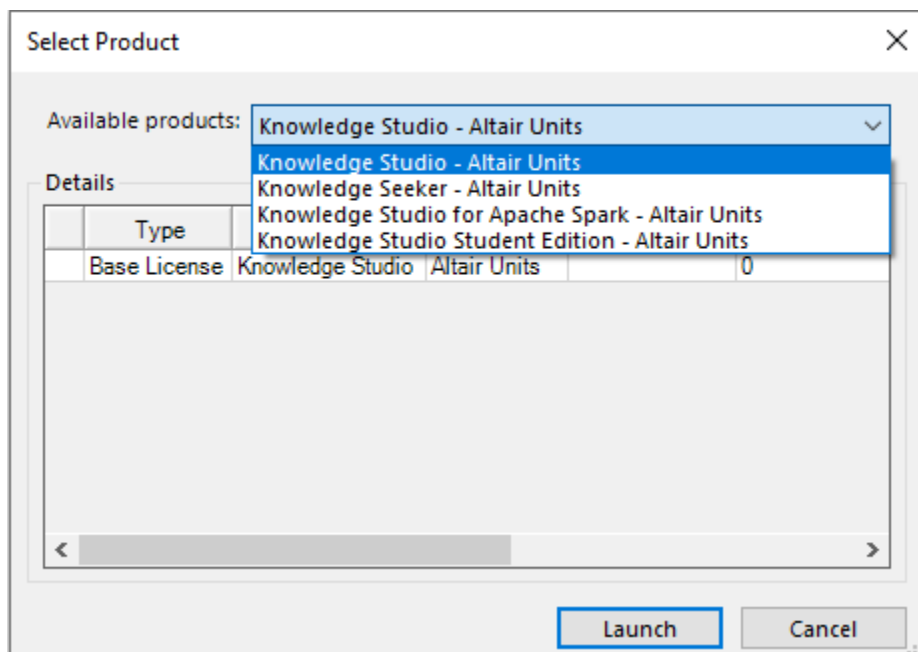
The image shows a 'New System Variable' dialog box. It has a title bar with a close button (X). Inside, there are two text input fields. The first is labeled 'Variable name:' and contains the text 'ALM\_HHWU'. The second is labeled 'Variable value:' and is empty. Below these fields are two buttons: 'Browse Directory...' and 'Browse File...'. At the bottom right are 'OK' and 'Cancel' buttons.

- Set *Variable name* to **ALM\_HHWU**
- To use Managed Altair Units, set the value of ALM\_HHWU to **True**. Otherwise, set it to **False**:

Click **OK** in all dialogs to save the changes and exit the Advanced System Properties dialog.

Start the KS Workstation application on any client machine.

Connect to KS Server. The Product Selection window will prompt you to select one of the three products: Knowledge Seeker, Knowledge Studio, or Knowledge Studio for Apache Spark.



Select Knowledge Studio for Apache Spark and click **Launch**.

## Client/Server for Linux

Define the environment variables `ALTAIR_LICENSE_PATH` and `ALM_HHWU` in the KS Server startup script `ks_start` as described below. For information about `ks_start`, see the section “Configuring KS Server on Linux” in the Deployment Guide.

If you are using Altair License Server on premises, set the value `ALTAIR_LICENSE_PATH` to the address of the Altair License Server in the format `NNNN@<server-name>`, where `NNNN` is the port number and `<server-name>` is the Altair License Server host name. Example:

```
export ALTAIR_LICENSE_PATH=6200@altair-license-server.domain.com
```

Set the value of `ALM_HHWU` to **false**:

```
export ALM_HHWU=false
```

If you are using Managed Altair Units, set `ALM_HHWU` to true and `ALTAIR_LICENSE_PATH` to false:

```
export ALM_HHWU=true
```

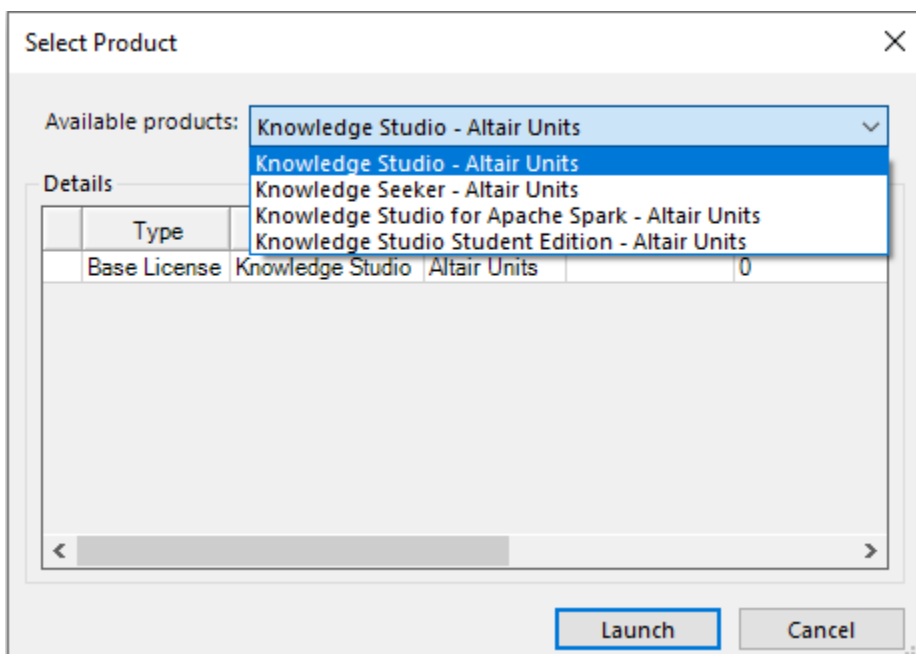
```
export ALTAIR_LICENSE_PATH=false
```

Save the changes in `ks_start` and restart KS Server as described in the section “Starting and Stopping the KS Service” in the Deployment Guide.

Define `ALTAIR_LICENSE_PATH` the same way in the script `ksenv.sh` used for troubleshooting purposes. This script is in the same directory as `ks_start`.

Page | 31

Start KS Workstation on any client machine and connect to KS Server. The Product Selection window will prompt you to select one of the three products: Knowledge Seeker, Knowledge Studio, or Knowledge Studio for Apache Spark.



Select Knowledge Studio for Apache Spark and click **Launch**.

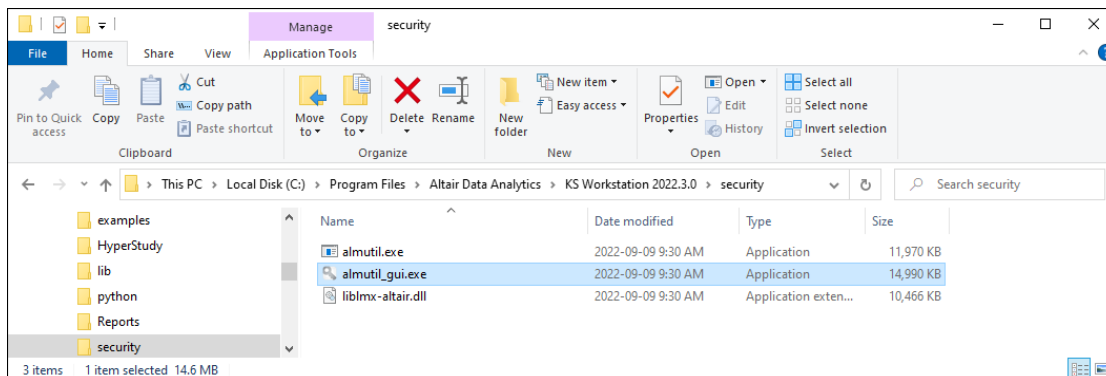
## Authorize Machine for Managed Altair Units License

This topic describes how to authorize your machine for a Managed Altair Units license, where the License Server is hosted by Altair. This step is not required if you are using Altair License Server on premises.

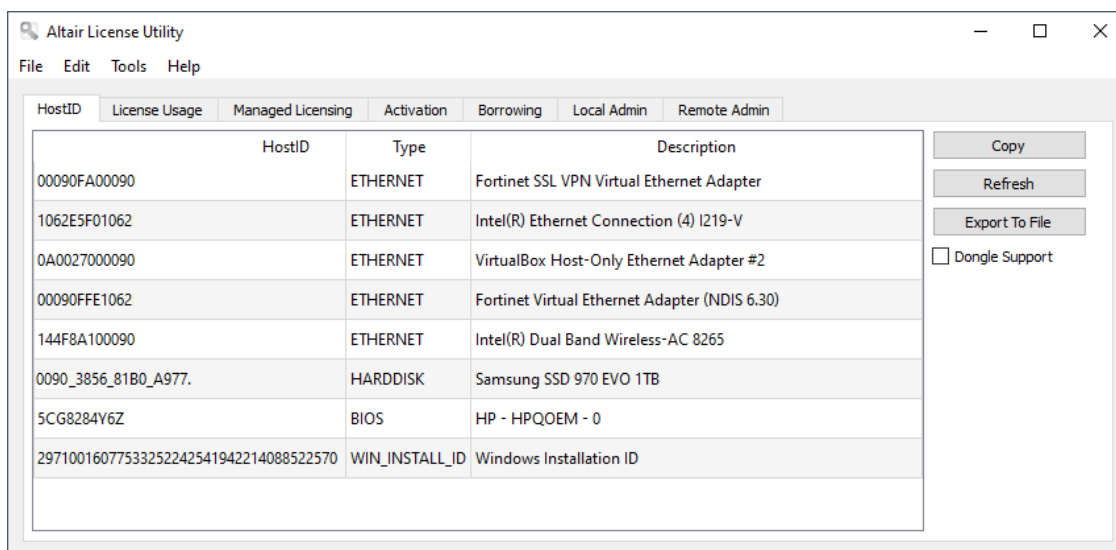
### Standalone configuration for a single user:

1. If you have not registered on the AltairOne portal yet, ask a user with Altair license admin role at your company to add you as a user at <https://AltairOne.com>. You will receive an email from Altair with a link to activate your AltairOne account. Click the activation link in your email and register on the AltairOne portal.

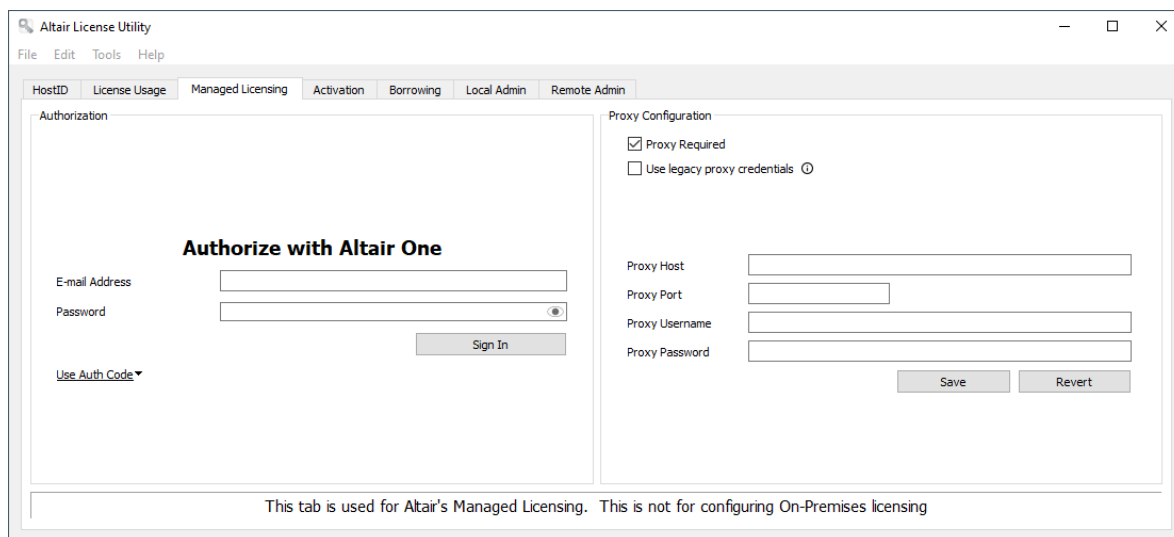
- Open the file browser and navigate to the folder `C:\Program Files\Altair Data Analytics\KS Workstation 2022.3.0\security`



- Double-click on the **almutil\_gui**. The Altair License Utility window opens:



- Go to the **Managed Licensing** tab of Altair License Utility window:



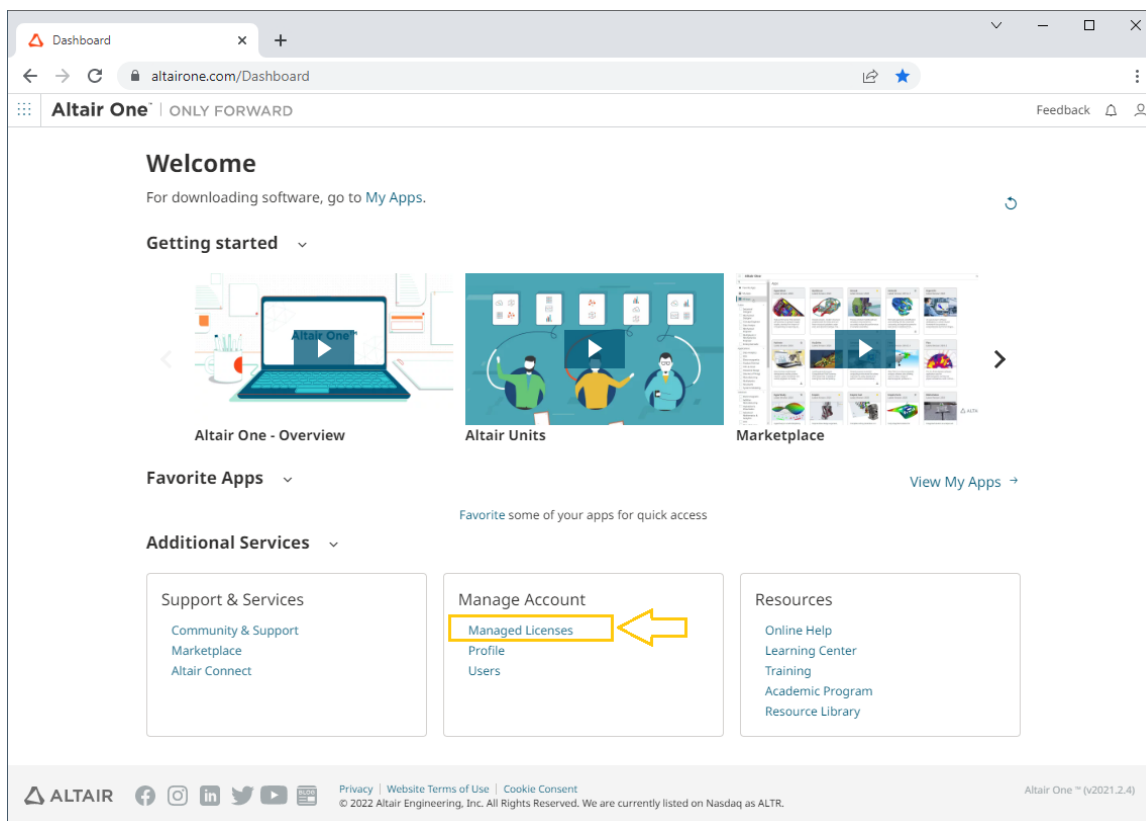
5. Specify your AltairOne credentials: your email address and password that you used to register at the [AltairOne](#) portal. Click **Sign In**.

Alternatively, if you are not registered at the [AltairOne](#) portal and received an authorization code an Altair license admin user at your company, click Use Auth Code, specify the authorization code received from this user and click **Authorize**.

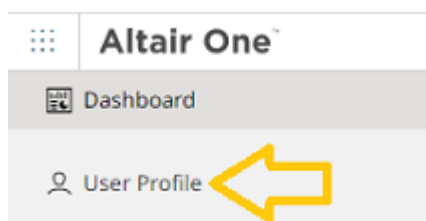
You will get the message: “Successfully authorized this machine with the Hosted HWU system”. Click **OK**.

#### **Client/Server configuration or Standalone configuration for multiuser access:**

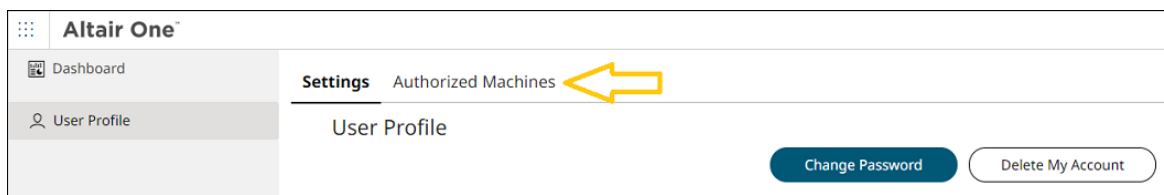
1. If you have not registered yet on the AltairOne portal, ask a user with Altair license admin role at your company to add you as a user at <https://AltairOne.com>. You will receive an email from Altair with a link to activate your Altair One account. Click the activation link in your email and register on the Altair One portal.
2. Log on to <https://altairone.com/Dashboard>. In the Manage Account section of the Altair One Dashboard page, click **Managed Licenses**, as shown below.



3. In the left pane, click **User Profile**:



4. Click **Authorized Machines** in the User Profile page:

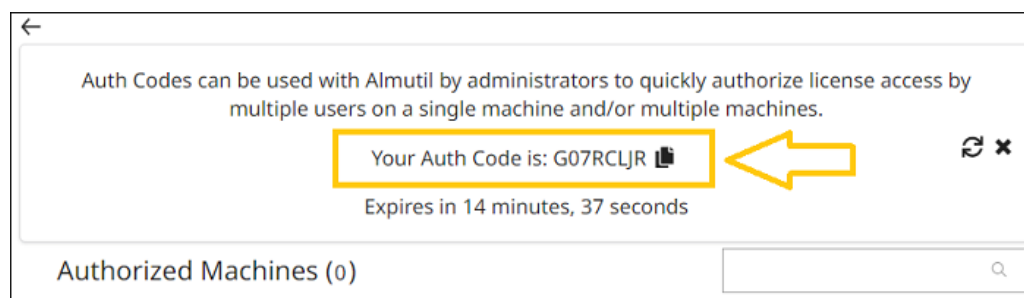


5. In the Authorized Machines page, click **Generate Auth Code** at the top right corner:



6. This will generate an authorization code to authorize a machine for the Altair license. Copy the code to the clipboard. Note: The code is valid for 15 minutes, so you must perform the next 7 steps within 15 minutes of generating the code.

Page | 35



7. After copying the code:
  - For the Standalone configuration on Windows: run Command Prompt as administrator and change to the folder `C:\Program Files\Altair Data Analytics\KS Workstation 2022.3.0\security`
  - For the Client/Server configuration on Windows: run Command Prompt as administrator and change to the folder `C:\Program Files\Altair Data Analytics\KS Server 2022.3.0\security`
  - For the client/server configuration on Linux: Change to `/usr/local/altair/licensing15.1/bin/` or to a directory that contains **almutil**. If you cannot find almutil on your system, please contact Altair Customer Support.
8. Run the following command as administrator, where *NNNN* is the authorization code generated at the previous step:
 

```
almutil -alauth -system -code NNNNN
```

The almutil tool can also be used to test the connectivity with AltairOne servers and configure proxy settings for use with AltairOne. Run `almutil` to view the detailed help information on the usage of the utility.
9. Once you get the message that the machine was successfully authorized, restart the KS Server service (in the client/server configuration) or KS Workstation (in the Standalone configuration).

Make sure you set the environment variables for Managed Altair Units as described above in the section [Configuring Altair Units Licensing](#).

---

## Registration-Key Licensing

### Altair License Fulfillment Server (ALFS)

ALFS is a secure self-service website used to acquire licenses and view license usage information. A [Registration Key](#) is required to log in to ALFS. Knowledge Studio accesses ALFS through the License Manager feature available from the Help menu. ALFS can also be accessed directly from a Web browser at <https://license.angoss.com/>.

ALFS keeps a copy of all acquired licenses. In the event of a system failure or accidental deletion of the license on your computer, the license can be retrieved from ALFS and reinstalled.

See also:

[Registration Keys](#)

[License Manager for Registration-Key Licenses](#)

[License Types](#)

### Registration Keys

After you purchase, renew, or request an evaluation of Knowledge Studio with a registration-key license, Altair sends you a registration key that provides access to your license(s). A registration key is a string about 30 characters long.

### Types of Registration-Key Licenses

#### Named User Licenses

Named-User licenses are user-based node-locked licenses available for both the standalone and client/server configurations. They will only operate on a specific machine for a specific user ID.

#### CPU & RAM-based Licenses

CPU & RAM-based licenses are available only for Windows Server and Linux platforms. These node-locked licenses tied to the machine ID of the host server activate the software only if the number of logical CPUs and the amount of RAM do not exceed the limits specified in the license. The total number of users may or may not be limited, depending on the contract terms.

See also:

[Altair License Fulfillment Server \(ALFS\)](#)

[License Manager for Registration-Key Licenses](#)

[License Types](#)



---

## Prerequisites for Installing Registration-Key Licenses

Administrative rights are not required to install licenses. The end users must have Read & Write permissions for the license directory.

If an IT administrator assists the intended user in activating the license, the administrator should log in with the user ID of the intended user.

The path to the license directory is displayed in the [License Manager](#) dialog which can be accessed from the Help menu.

Page | 37

## Client/Server Configuration

In the client/server configuration, the license resides on the server. Before installing the license, make sure that KS Workstation is configured to connect to the server.

1. Start KS Workstation using the desktop icon or the Altair Data Analytics program folder in the Windows Start menu.
2. You will be prompted for your password to log in to KS Server. Enter your password (your Windows or Linux user password, depending on the server OS).

If you are not prompted for your password:

- Select the **Connect** command from the **File** menu.
- In the Establish Connection dialog:
  - Set the Mode to Client/Server Login.
  - Enter the server name or IP address and the port number. The default port is 5470.
  - Specify the working directory if known; otherwise, leave the field blank. When the connection is established, the file browser will let you browse the server and select a directory if it was not specified correctly.
  - Enter the user ID and password (your Windows or Linux user credentials, depending on the server OS).

3. Click OK to connect. After the connection has been established, the License Manager will open. The License Manager can also be accessed from the Help menu.

Proceed to the topic [License Manager for Registration-Key Licenses](#).

---

## License Manager for Registration-Key Licenses

The License Manager for registration-key licenses can be accessed from the Help menu of the application.

*Note: This License Manager dialog is not to be confused with the Altair License Manager software, which is a part of the [Altair Units Licensing](#) system.*

Page | 38

To prepare for license installation via the License Manager interface, see the topic [Prerequisites for Installing Registration-Key Licenses](#).

The License Manager for registration-key licenses can be used to perform the following tasks:

- Install a new license using one of the three available options
- Remove license(s)
- View currently installed licenses
- Obtain information required for generating license files when disconnected from the Internet
- Access the [Altair License Fulfillment Server \(ALFS\)](#) to view the license usage and history for a given registration key.
- Change the license location (in the standalone mode only)

**License Manager**

| ID                | VALUE                    |
|-------------------|--------------------------|
| User ID           | jsmith                   |
| Unique User ID    | 5R4IWG6HC2               |
| Machine ID        | LAP105                   |
| Unique Machine ID | IOBI6Y6ZBWASD            |
| Host OS           | Windows 10.0 Build 17763 |
| Working Directory | "C:\Altair KS Projects\" |
| License Location  | "C:\Altair KS License\"  |
| License File      | A0710D0D.lic             |
| Port              | 5470                     |
| Mode              | Local                    |
| Secure            | True                     |
| License Addons    |                          |

**License in Use**

User:

License Type:

Product:

Expiry:

Grace:

**Default Registration Key**

| Remove?                  | VALID | REASON | ACTIVE              | EXPIRE              | GRACEEVALUATIONPERIOD |       |
|--------------------------|-------|--------|---------------------|---------------------|-----------------------|-------|
| <input type="checkbox"/> | true  |        | 2018-10-17 12:00:00 | 2020-01-31 12:00:00 | 0                     | Knowl |

## Installing Licenses

**NOTE: For the Client/Server configuration only:** Connect from KS Workstation (client) to KS Server BEFORE installing the license(s). If you are an administrator activating a named-user license for a user, connect to the server using the credentials of the intended user. To make sure you are in the client/server mode, select the **Connect** command from the **File** menu. If the Mode is Standalone, change it to Client/Server Login, specify the connection information and click OK.

The connection dialog can be launched by the **File | Connect** command from the main menu.

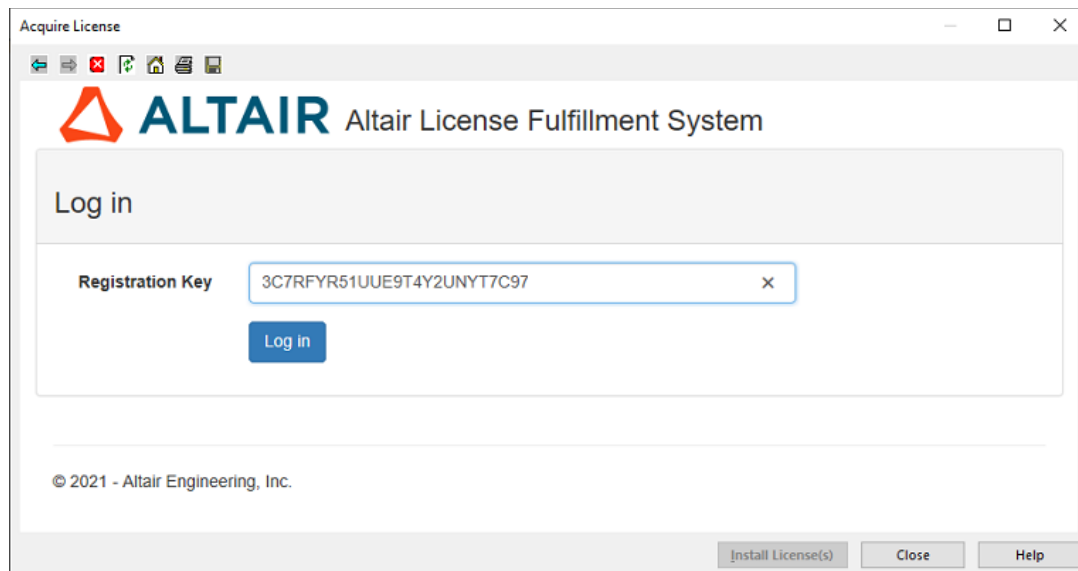
## Acquiring Licenses Online

Conditions:

- The Altair License Fulfillment Server (ALFS) website <https://license.angoss.com> must be accessible from the end user's machine.

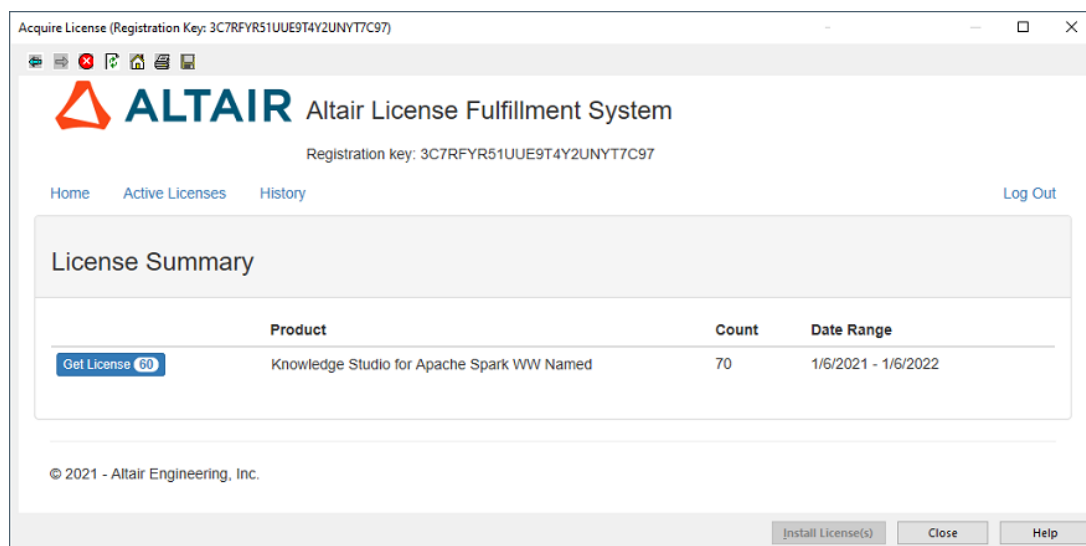
Procedure:

- Click **Acquire License Online** in the License Manager window to navigate to the ALFS web page.



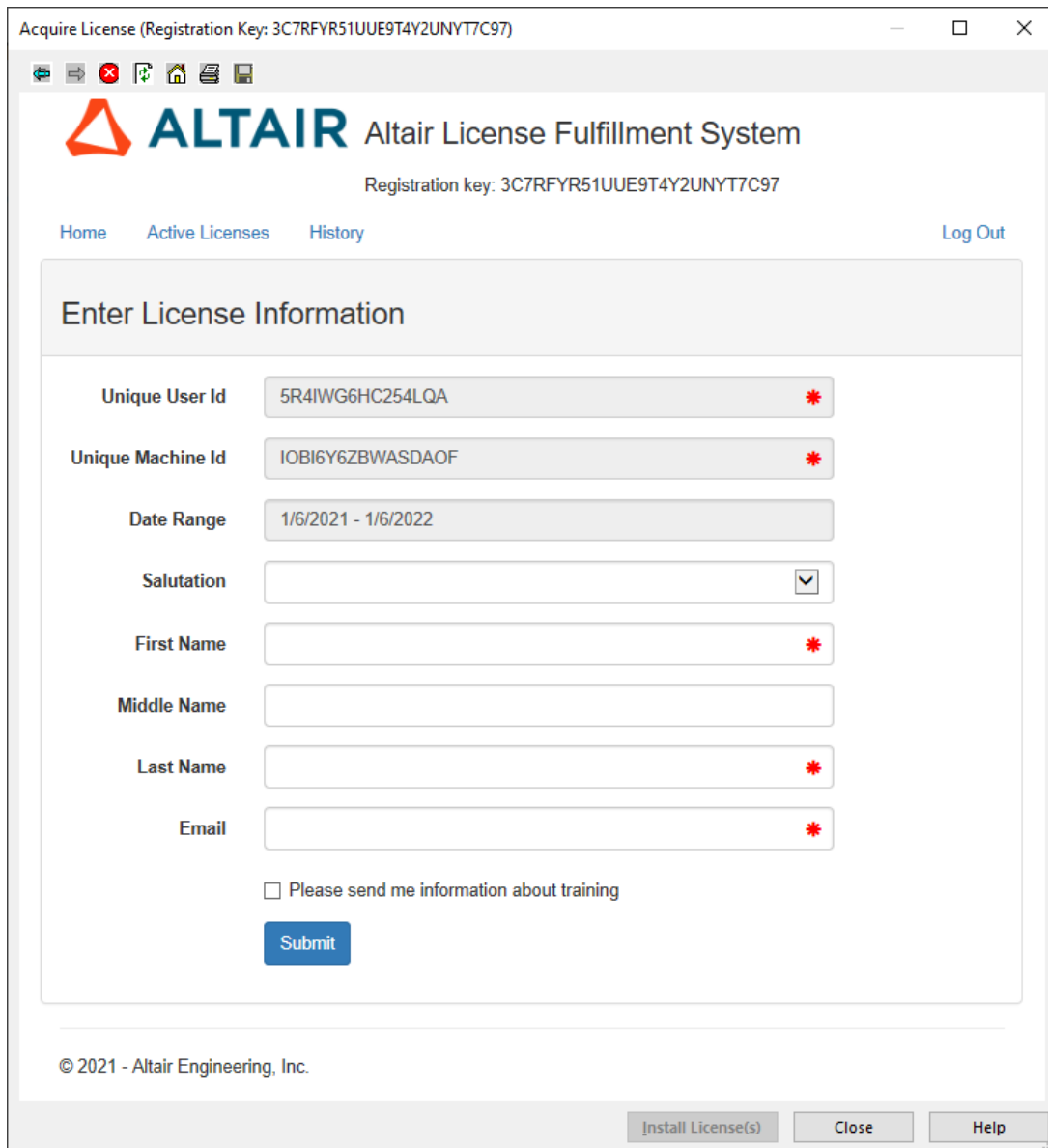
Page | 40

- Log on to ALFS using the Registration Key provided by Altair. The next screen displays the name of the licensed product, the number of licenses available, and the validity date range.



- Click the **Active Licenses** link to view the licenses that have been installed. To reinstall a license, click **View** for the license you want to reinstall, then click Yes when prompted to install it.
- Click **History** to view the license installation history, if desired.

- To generate a new license, click the **Get License** button and enter the required information.



Acquire License (Registration Key: 3C7RFYR51UUE9T4Y2UNYT7C97)

 **ALTAIR** Altair License Fulfillment System

Registration key: 3C7RFYR51UUE9T4Y2UNYT7C97

[Home](#) [Active Licenses](#) [History](#) [Log Out](#)

### Enter License Information

Unique User Id: 5R4IWG6HC254LQA \*

Unique Machine Id: IOBI6Y6ZBWASDAOF \*

Date Range: 1/6/2021 - 1/6/2022

Salutation:

First Name:  \*

Middle Name:

Last Name:  \*

Email:  \*

☐ Please send me information about training

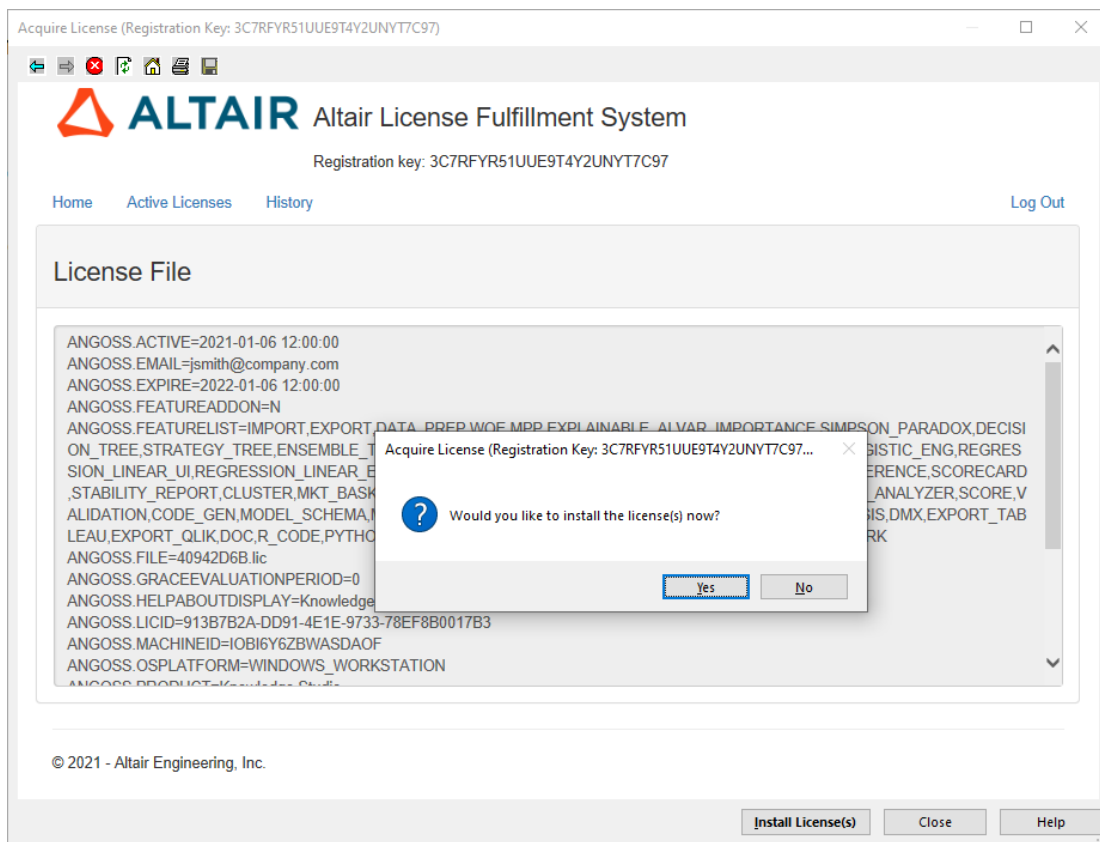
© 2021 - Altair Engineering, Inc.

- The Unique Machine ID and Unique User ID fields are auto-populated by the application and cannot be edited. These are encrypted versions of the machine ID and user ID.
- Enter your first name, last name, and Email address.
  - For concurrent-user licenses, make sure the KS Server license directory is specified in the *Concurrency Lock File Location* field. Do not change this value. If this field is blank, find out the license location on the server and enter it here. On Windows server, it is usually C:\Program Files\Altair Data Analytics\License,

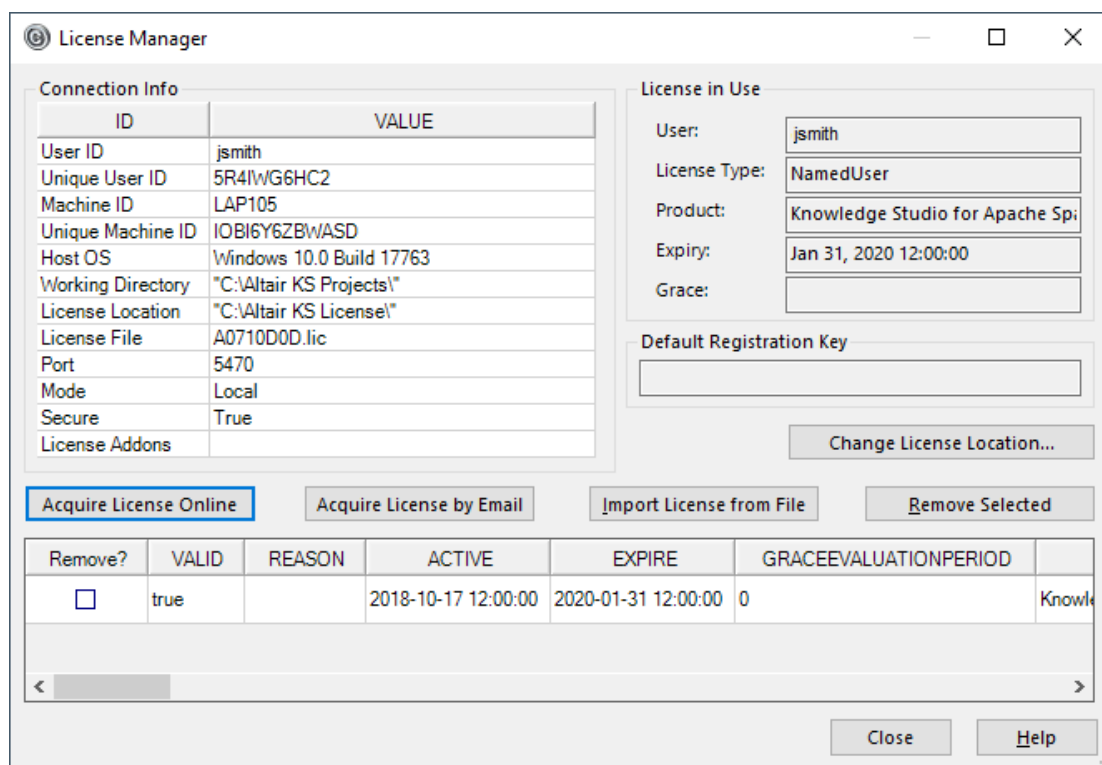
while on Linux it is `/opt/Altair-Data-Analytics/license`. This directory on the server must have read and write permissions for all users of the application.

- Click **Submit**.
- The next screen displays the license content and prompts you to install the license.

Page | 42



- Click **Yes** to install the license. If you click **No**, you can still install the license by clicking **Install License(s)** at the bottom-right corner of the screen or close the dialog.
- The user will be returned to the License Manager window with the newly installed license(s) displayed in the lower frame of the dialog as a white row with the columns Remove?, VALID, REASON, etc.



- Close the License Manager dialog.

The menus File, Edit, View, Insert, Tools, Window, and Help should now be enabled.

### Using the “Acquire License by Email” button:

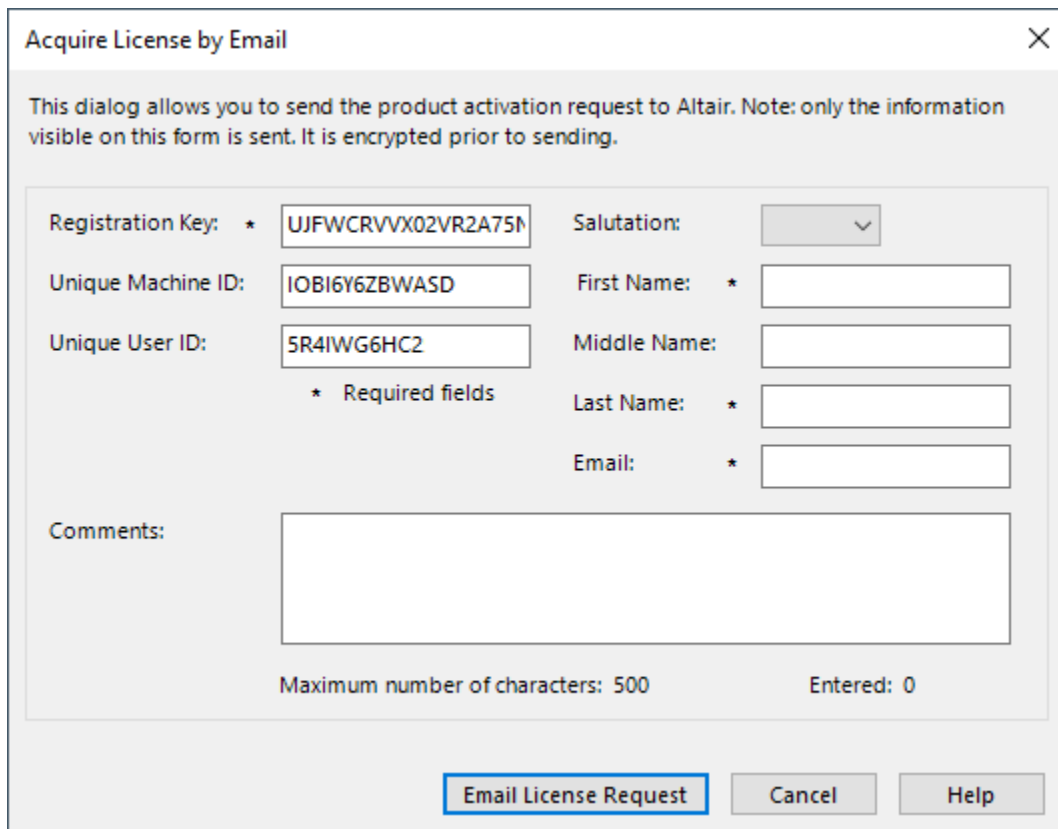
Conditions:

- Email service is required on the end user’s machine.
- Internet access is not required on the end user’s machine.

Clicking the Acquire License button will generate an email message to be sent to Altair with the information needed to create a license.

**Note:** By default, User based licenses will become locked to the user profile that was logged on at the time of license generation. This does not concern Concurrent licenses.

**Warning:** If the licenses to be installed are for the client/server configuration, please ensure to connect from your instance of KS Workstation to the server component BEFORE generating or installing any license. To connect to the server, select **File | Connect** from the application menu.



The dialog box is titled "Acquire License by Email" and includes a close button (X) in the top right corner. Below the title bar, a note states: "This dialog allows you to send the product activation request to Altair. Note: only the information visible on this form is sent. It is encrypted prior to sending." The form contains several input fields: "Registration Key:" with a text box containing "UJFWCRVVX02VR2A75N" and an asterisk; "Unique Machine ID:" with a text box containing "IOBI6Y6ZBWASD"; "Unique User ID:" with a text box containing "5R4IWG6HC2" and an asterisk; "Salutation:" with a dropdown menu; "First Name:" with a text box and an asterisk; "Middle Name:" with a text box; "Last Name:" with a text box and an asterisk; "Email:" with a text box and an asterisk. A "Comments:" section with a large text area is located below the "Unique User ID" field. A status bar at the bottom of the form indicates "Maximum number of characters: 500" and "Entered: 0". At the bottom of the dialog, there are three buttons: "Email License Request" (highlighted with a blue border), "Cancel", and "Help".

The Registration Key, Unique Machine ID, and Unique User ID fields should already be populated. These fields should not be changed.

The Registration password is blank by default, so this field should be left blank.

1. Enter your First and Last name and email address.
2. Make sure the Type of Activation Request is "Get License".

For concurrent-user licenses:

Specify the folder for the concurrency lock file in the Concurrency Lock File Location field. In most cases we recommend specifying the Altair license folder. On Windows Server, it is usually C:\Program Files\Altair Data Analytics\License, while on Linux it is the Altair-Data-Analytics/license subdirectory of the installation folder (for example, /opt/Altair-Data-Analytics/license). The concurrency lock file is required for controlling concurrent connections. This location needs to have read and write permissions for all users of the application.

3. Click the **Email License Request** button. The license request will be sent to the email address kw-productactivation@altair.com.



4. When you receive an email with the license from the License Administrator or Customer Support, open the License Manager window and copy the license content from the email to the clipboard.

6. Click on the **Add License** button in the License Manager window. The Specify License Content dialog opens.

7. Paste the license content from the clipboard into this dialog by pressing CTRL C. Make sure you include only the license text and don't add any other characters. An example of the license text can be found in the [License Types](#) topic.

8. Click OK. The license should now be installed and visible in the lower frame of the License Manager.

If the Registration Key has licenses for more than one product associated with it, a list of all available products and licenses will be returned in the response email from Altair Customer Support. Open the received email and review the entries in the list for the product whose license needs to be generated. You will need the following details for that product: Product ID, Version #, and Order #.

1. Open License Manager again by selecting **Help | License Manager** from the menu.

2. Click on the Acquire License by Email button and do the following:

3. Enter the user's First and Last name and email address.

4. Make sure the Type of Activation Request is "Get License".

5. Enter the Product ID, Version #, and Order # exactly as they appear in the email sent from kw-productactivation@altair.com.

*Additional step for Concurrent License installation only:* Specify the location where the concurrency lock file will be stored. (The concurrency lock file is required for the concurrent license to work.) This location needs to have read and write permissions for all users of the application.

6. If this is not a concurrent license installation, please leave the Concurrency lock file location field blank and proceed to the steps below.

7. Click the Email License Request button. The license request will be sent to Altair and the response email with the license will be sent back to the user.

8. Once you receive the email from kw-productactivation@altair.com in response to your request, copy the license content from the email. Click on the Add License dialog of the License Manager. The Specify License Content dialog opens. Paste the license content into this dialog and click OK.

**Note:** When copying and pasting the license ensure not to accidentally include any extra spaces or carriage returns.

The license should now be installed and visible in the lower frame of the License Manager.

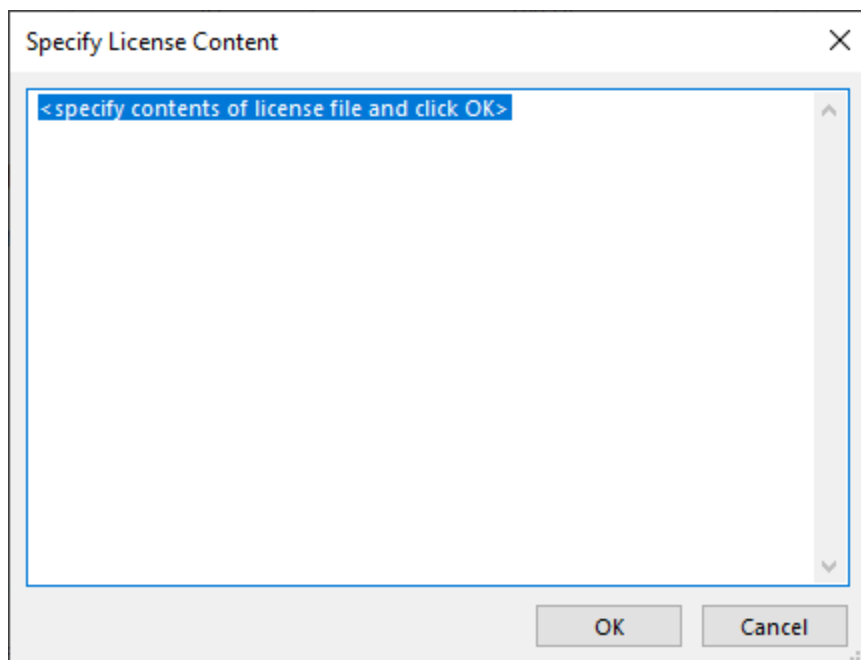
### Using the “Add License” button:

Page | 46

Conditions:

- Internet access is not required on the end user’s machine.
- Email service is not required on the end user’s machine.

Clicking on the Add License button in License Manager opens the Specify License Content dialog.



The license text copied from an email or from ALFS should be pasted into this dialog. Clicking OK will install the license and return the user to the License Manager with the newly installed license visible.

### Using the “Remove Selected” button to remove installed license(s):

Conditions:

- Internet access is not required on the end user’s machine.

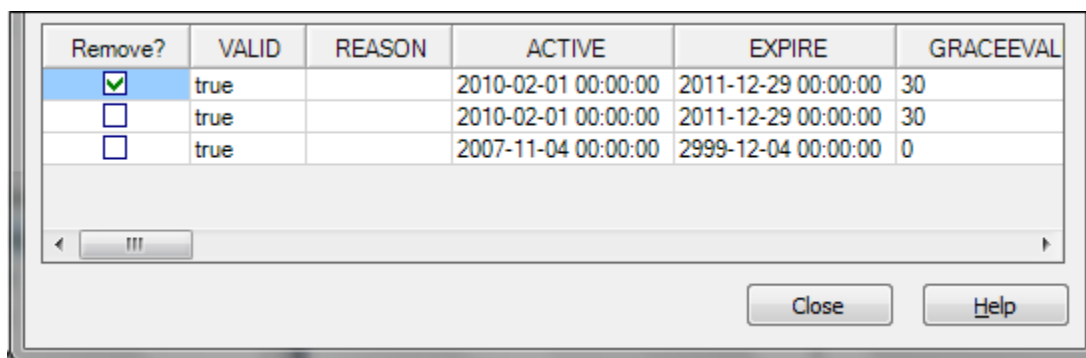
- Email service is required on the end user's machine if proof of license deletion is required from the user by Altair.

Licenses can be uninstalled and removed from the application by using the 'Remove Selected' button of the License Manager.

Page | 47

A confirmation email will be sent to Altair Client Support, [dasupport@altair.com](mailto:dasupport@altair.com), indicating the license that is being removed.

- Click on the checkbox in the 'Remove?' column of the license to be removed.



| Remove?                             | VALID | REASON | ACTIVE              | EXPIRE              | GRACEEVAL |
|-------------------------------------|-------|--------|---------------------|---------------------|-----------|
| <input checked="" type="checkbox"/> | true  |        | 2010-02-01 00:00:00 | 2011-12-29 00:00:00 | 30        |
| <input type="checkbox"/>            | true  |        | 2010-02-01 00:00:00 | 2011-12-29 00:00:00 | 30        |
| <input type="checkbox"/>            | true  |        | 2007-11-04 00:00:00 | 2999-12-04 00:00:00 | 0         |

- Multiple licenses can be removed at the same time if needed by selecting and checking their respective check boxes.
- After all licenses to be removed have been selected, click the 'Remove Selected' button.
- The Confirm License Deletion dialog will appear giving the user a chance to review the license(s) that will be removed.
- Users have the option to send a confirmation email to Customer Support. By default, the checkbox at the bottom of the Confirm License Deletion dialog is checked and an email will be sent to Customer Support.
- Clicking OK in the Confirm License Deletion dialog will remove the selected license(s) and send a confirmation email to Customer Support if the checkbox had not been unchecked.
- The user will be returned to the License Manager with the list of licenses installed, no longer containing the license(s) that were removed.

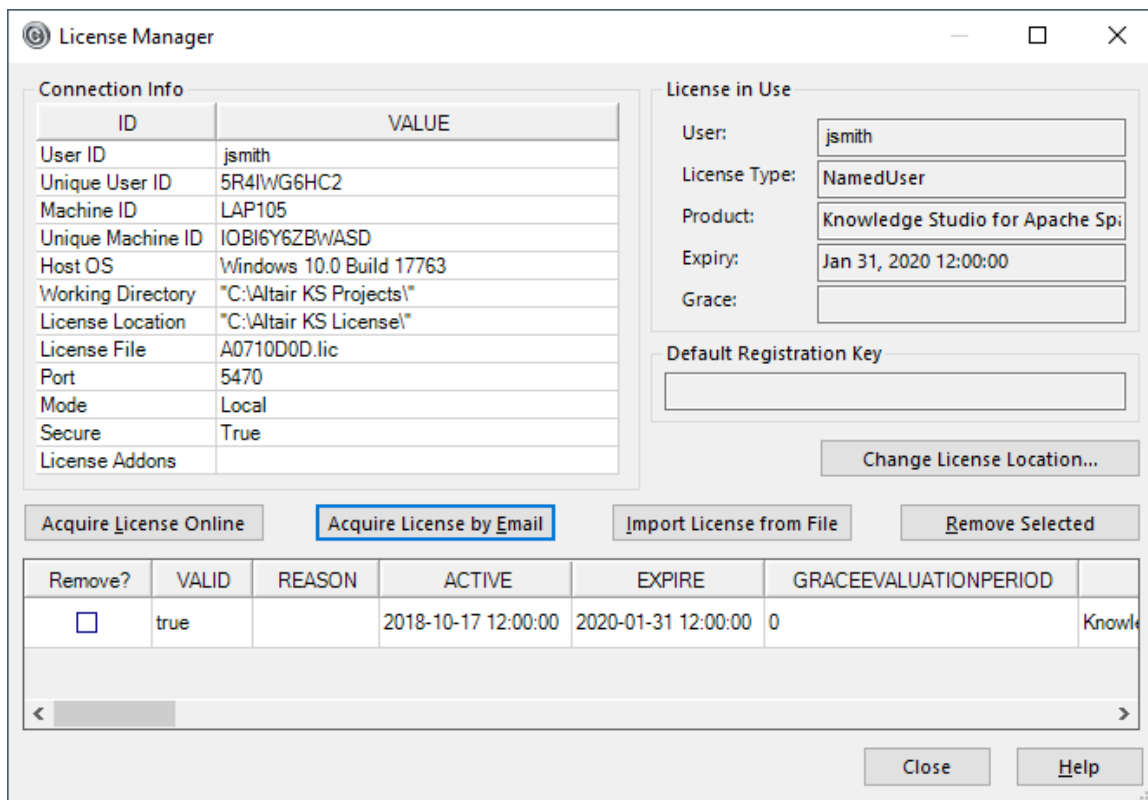
## Acquiring Licenses by Email

Use the function **Acquire License by Email** in the License Manager if there is no Internet access on your machine. This will generate an email message to be sent to Altair Customer Support with the information required to generate a license.

**Note:** If you are an administrator acquiring a named-user license for another user, make sure that you log in to Windows with the intended user's ID.

If you are using the application in the client/server configuration, connect to KS Server first: Start KS Workstation, select the **Connect** command from the **File** menu, choose the **Server Login** mode, and specify the connection properties.

The License Manager window opens when you start the application. It is also available under the **Help** menu. Click the button **Acquire License by Email** in the License Manager.



| ID                | VALUE                    |
|-------------------|--------------------------|
| User ID           | jsmith                   |
| Unique User ID    | 5R4IWG6HC2               |
| Machine ID        | LAP105                   |
| Unique Machine ID | IOBI6Y6ZBWASD            |
| Host OS           | Windows 10.0 Build 17763 |
| Working Directory | "C:\Altair KS Projects\" |
| License Location  | "C:\Altair KS License\"  |
| License File      | A0710D0D.lic             |
| Port              | 5470                     |
| Mode              | Local                    |
| Secure            | True                     |
| License Addons    |                          |

|               |                                |
|---------------|--------------------------------|
| User:         | jsmith                         |
| License Type: | NamedUser                      |
| Product:      | Knowledge Studio for Apache Sp |
| Expiry:       | Jan 31, 2020 12:00:00          |
| Grace:        |                                |

Default Registration Key

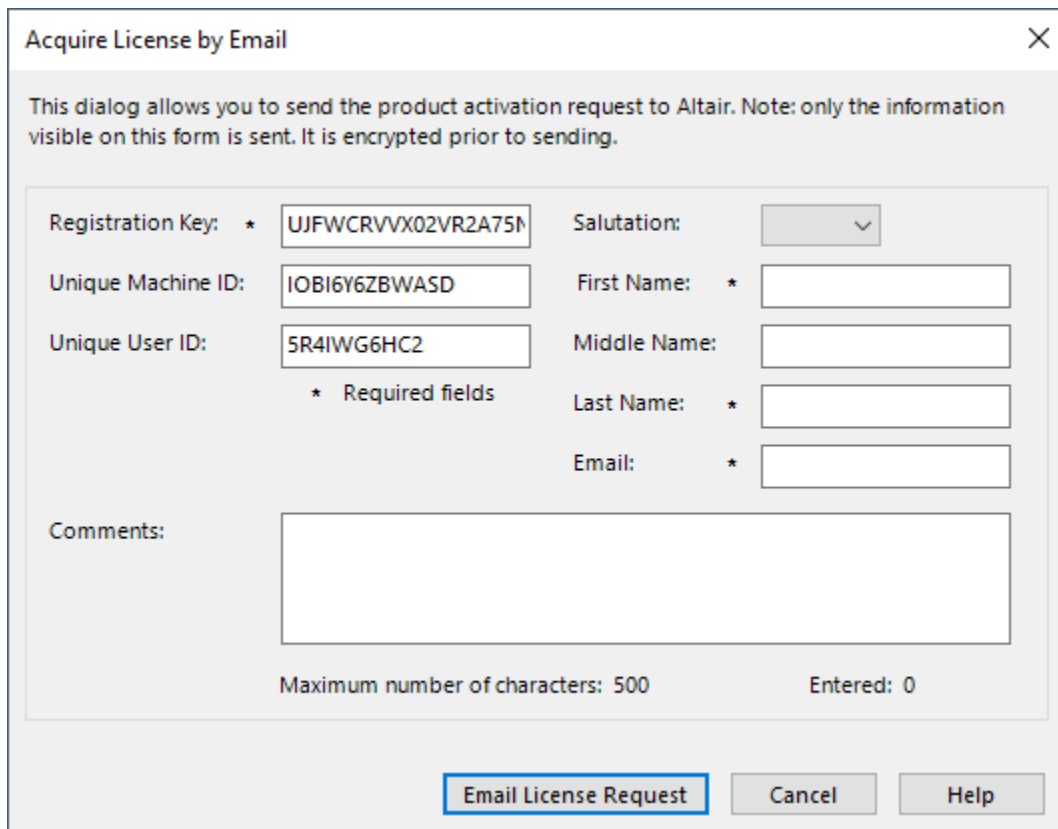
Change License Location...

Acquire License Online   **Acquire License by Email**   Import License from File   Remove Selected

| Remove?                  | VALID | REASON | ACTIVE              | EXPIRE              | GRACEEVALUATIONPERIOD |       |
|--------------------------|-------|--------|---------------------|---------------------|-----------------------|-------|
| <input type="checkbox"/> | true  |        | 2018-10-17 12:00:00 | 2020-01-31 12:00:00 | 0                     | Knowl |

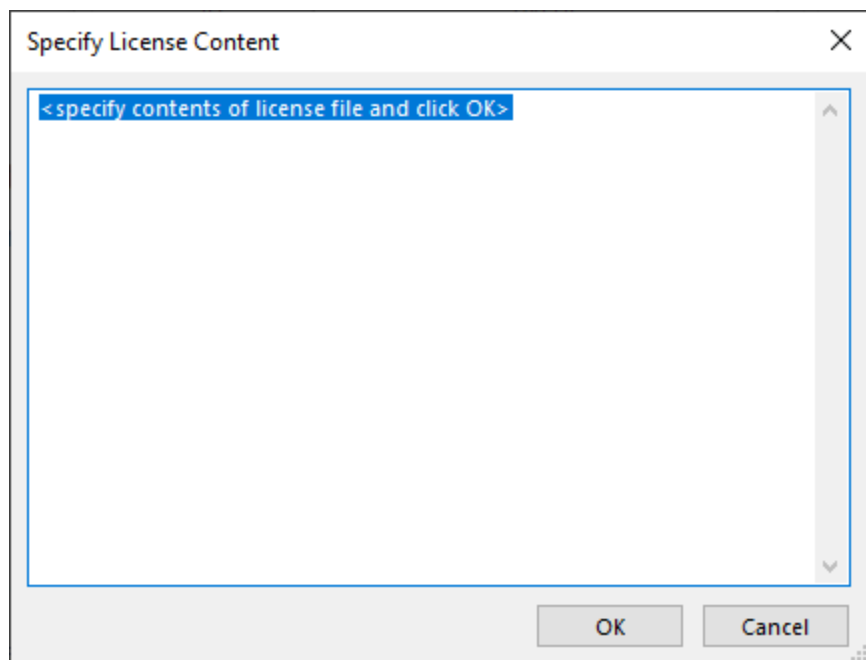
Close   Help

The Acquire License by Email dialog opens. The Unique Machine ID and Unique User ID fields are populated automatically, and these values should not be changed.



The dialog box is titled "Acquire License by Email" and includes a close button (X) in the top right corner. Below the title bar, a message states: "This dialog allows you to send the product activation request to Altair. Note: only the information visible on this form is sent. It is encrypted prior to sending." The form contains several input fields: "Registration Key:" with a text box containing "UJFWCRVVX02VR2A75M" and an asterisk; "Unique Machine ID:" with a text box containing "IOBI6Y6ZBWASD"; "Unique User ID:" with a text box containing "5R4IWG6HC2" and an asterisk; "Salutation:" with a dropdown menu; "First Name:" with a text box and an asterisk; "Middle Name:" with a text box; "Last Name:" with a text box and an asterisk; and "Email:" with a text box and an asterisk. A "Comments:" section with a large text area is located below the "Unique User ID" field. A note "\* Required fields" is placed between the "Unique User ID" and "First Name" fields. At the bottom of the text area, it says "Maximum number of characters: 500" and "Entered: 0". At the bottom of the dialog, there are three buttons: "Email License Request" (highlighted with a blue border), "Cancel", and "Help".

1. Enter your first name, last name, and email address.
2. Enter any comments if necessary.
3. Click the **Email License Request** button. The license request will be sent to Altair Customer Support at [productactivation@datawatch.com](mailto:productactivation@datawatch.com).
4. Once you receive an email from Altair Customer Support in response to your request, open the attached license file in Notepad or any other text editor and copy its contents to the clipboard.
9. Run KS Workstation and open License Manager. Click the button **Import License from File** in the License Manager window. Paste the license content into the Specify License Content dialog. Do not change the license text. Click OK.



The installed license will be shown in the lower frame of the License Manager. Close the License Manager.

---

# Connecting to Data Mining Engine and Spark Server

## Connecting to the Data Mining Engine

Page | 51

The Altair Data Mining Engine is the core component of Knowledge Studio that interprets user input, connects to Apache Spark and translates the instructions (predictive analysis operations such as calculating summary statistics, creating and scoring a predictive model, etc.) to the Spark instructions to be performed on distributed data. See [Spark Server Connection](#) for details.

In the Standalone configuration, both the Data Mining Engine and the user interface component reside on the same computer. In the Client/Server configuration, the Engine resides on a server. In all configurations, when you launch the application, it connects to the Data Mining Engine, checks the license, and opens the Working Directory with your projects in the Project Explorer (the left panel). The path to the Working Directory is displayed at the top of the Project Explorer panel.

In the standalone mode, the connection to the Data Mining Engine is automatic, unless the Working Directory specified in the current settings is not accessible or does not exist.

### Establishing a connection using the File | Connect menu command is necessary when:

- the Project Explorer panel is blank and the commands **Create Project** and **Set Working Directory** under the File menu are disabled, which may happen in the following cases:
  - the default Working Directory is invalid (does not exist or is not accessible)
  - the default client/server connection information was not specified at the installation stage
  - after a previous unsuccessful attempt to connect to the server (for example, if incorrect user credentials or host information was used)
  - after the connection was terminated abnormally or reset as a result of server maintenance
- connecting to KS Server on a different server
- changing from the standalone to the client/server mode or vice versa

To connect to the Data Mining Engine, select the menu command

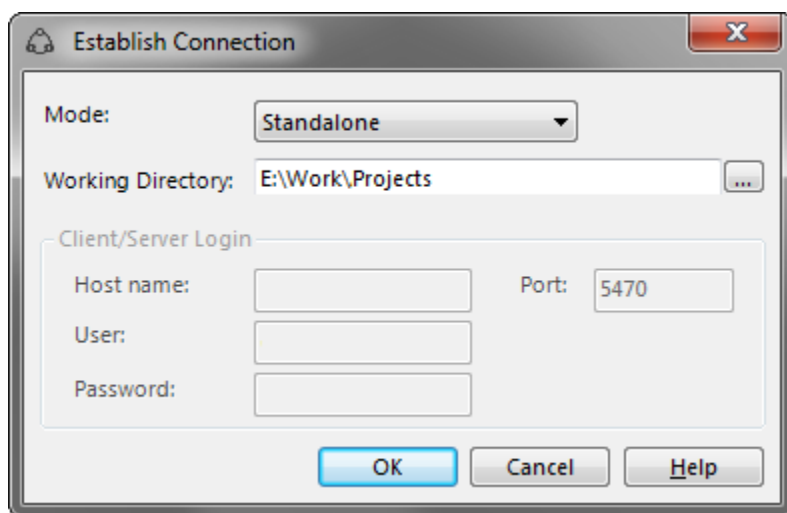
### **File | Connect**

or click the leftmost icon on the application toolbar. This brings up the Establish Connection dialog box.

## Standalone Mode

When working in standalone mode, the default working directory is the Altair folder under the My Documents folder, but it can be set to any other location at the installation stage. If the working directory is accessible, the connection is established automatically upon launching the application. Otherwise, select **File | Connect** from the menu and specify the correct name for the Working Directory:

Page | 52



|                          |                                                                                                                                                                     |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Mode</b>              | Standalone                                                                                                                                                          |
| <b>Working Directory</b> | The <a href="#">Working Directory</a> is the location where your projects and the related documents are stored. It is sometimes referred to as the Mining Location. |

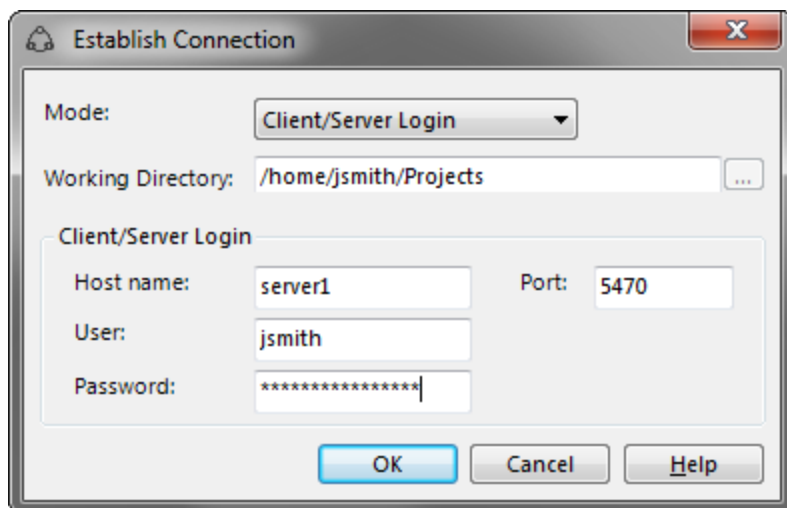
If already connected, you can change the current Working Directory at any time using the menu command **File | Set Working Directory** or by clicking the second icon from the left on the application toolbar. See the [Setting the Working Directory](#) topic for details.

## Client/Server Login Mode

In the client/server mode, the application engine is on the server machine that hosts KS Server. The user's workstation only contains the client component with graphical user interface to connect to the server and visualize the results. The connection properties can be set during the installation process. If all the properties are set during the installation, the connection is established automatically as you launch the client component and specify your password for the server.

If you need to establish the connection in one of the cases indicated above, you must select the command **File | Connect** and enter the connection information for the server that hosts KS Server. For example:





|                          |                                                                                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Mode</b>              | Client/Server Login                                                                                                                                   |
| <b>Working Directory</b> | The <a href="#">Working Directory</a> is the location where project related documents are stored. It is sometimes referred to as the mining location. |
| <b>Host name</b>         | The name or IP address of the machine hosting KS Server.                                                                                              |
| <b>Port</b>              | The number of the port to access KS Server.                                                                                                           |
| <b>User</b>              | User ID recognized by the KS Server host machine.                                                                                                     |
| <b>Password</b>          | User password.                                                                                                                                        |

If already connected, you can change the current Working Directory at any time using the menu command **File | Set Working Directory**, by clicking the second icon from the left on the application toolbar, or pressing CTRL W on your keyboard.

See also:

[Setting the Working Directory](#)

[About Projects](#)

[Creating a New Project](#)

[Spark Server Connection](#)

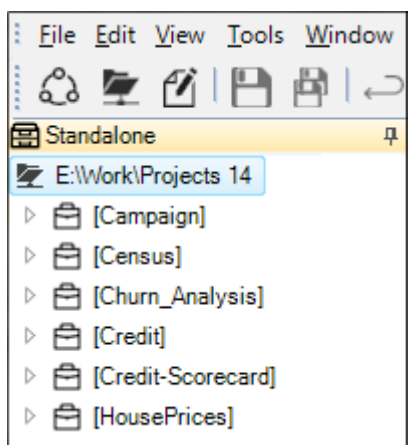
## Setting the Working Directory

The Working Directory is the location where your projects are stored. You can have more than one Working Directory if you have more than one set of projects. You can change your Working Directory at any time.

The Working Directory is usually the Altair folder under My Documents, which is the installation default for the Standalone mode.

In the Standalone configuration, the Working Directory is usually a local folder on the user's workstation, but it can also be on a network drive. In the Client/Server configuration, it is usually on the server machine that hosts KS Server.

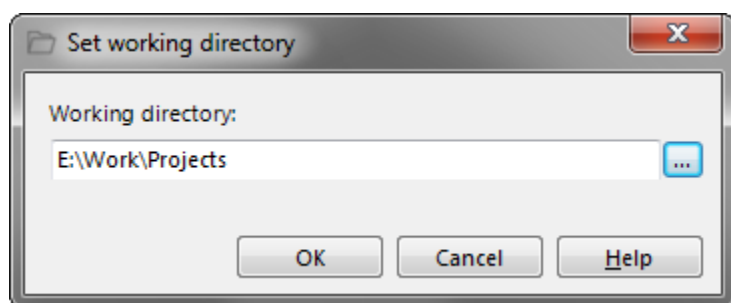
The Working Directory name is displayed at the top level of the Project Explorer panel. See the topic [About Projects](#) for details.



To change the Working Directory:

### From the menu:

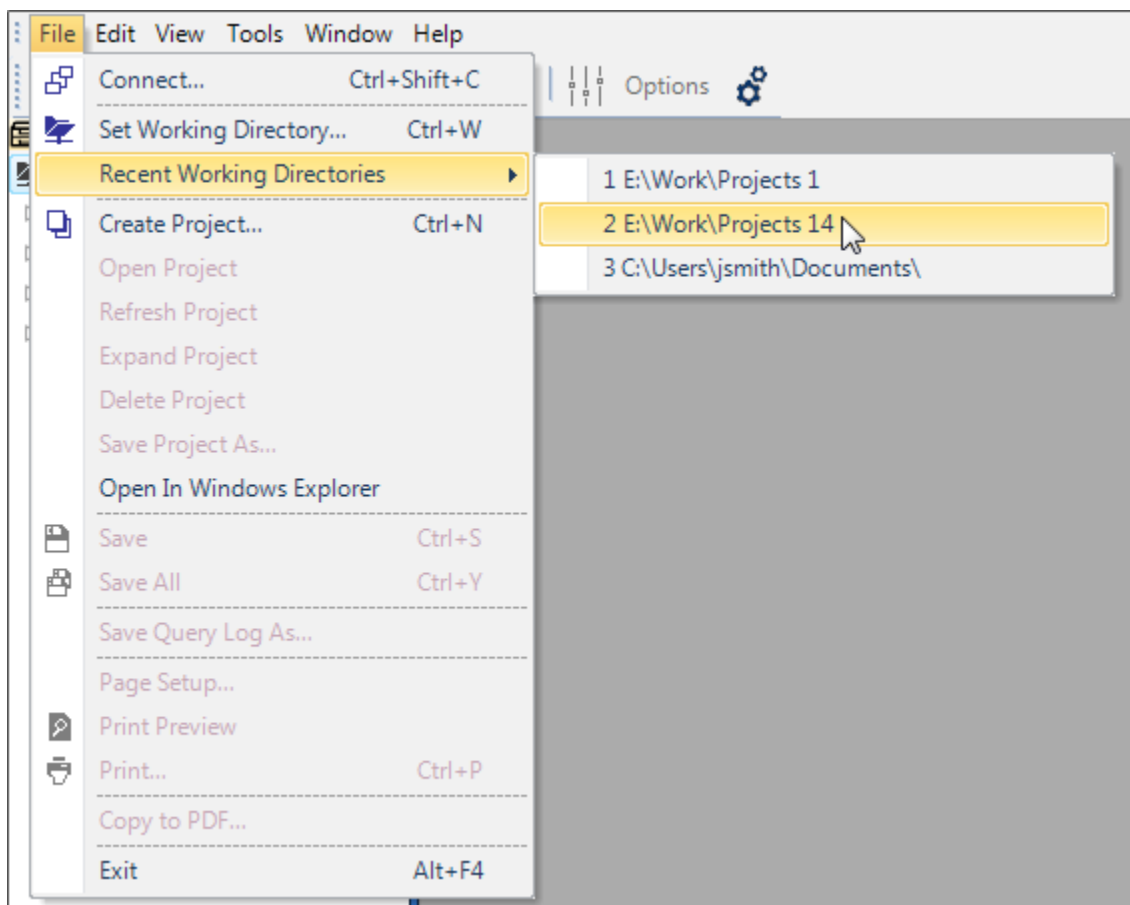
#### File | Set Working Directory



Type the name of the new Working Directory or use the browse button to find the desired directory.

You can open the Working Directory in Windows Explorer by right-clicking on its name in Project Outline and selecting *Open in Explorer* from the context menu. This may be useful if you wish to view error logs written to the working directory, back up your projects, share them with other users, and so on.

A list of recently used Working Directories is available from the File menu for easy switching between working directories.



## See Also:

[About Projects](#)

[Creating a New Project](#)

[Connecting to the Data Mining Engine](#)

[Spark Server Connection](#)

## Spark Server Connection

### Connecting to Spark

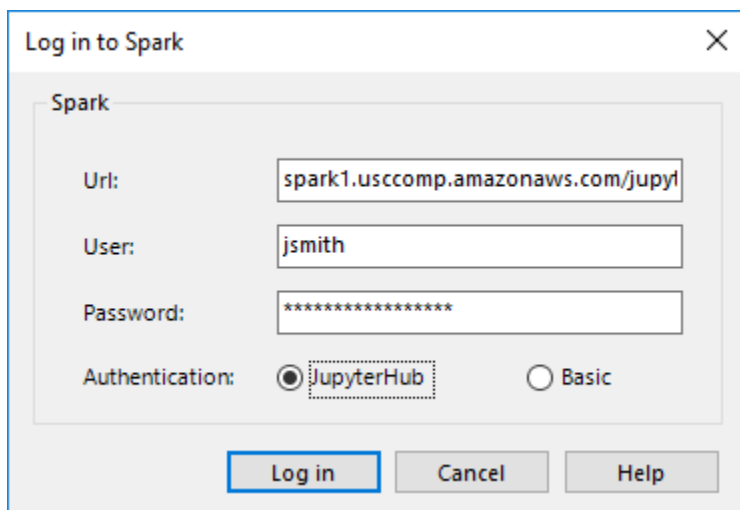
When opening a project with a workflow that uses Spark-integrated functions (nodes), you are automatically prompted to connect to the same Spark server. The connection dialog provides an option to force a new session or continue the previous one, if applicable.

Otherwise, you can connect to the Spark Server in two ways:

- Add a Spark-integrated node to the Workflow canvas and open its wizard. This will bring up the Spark server connection dialog.

OR

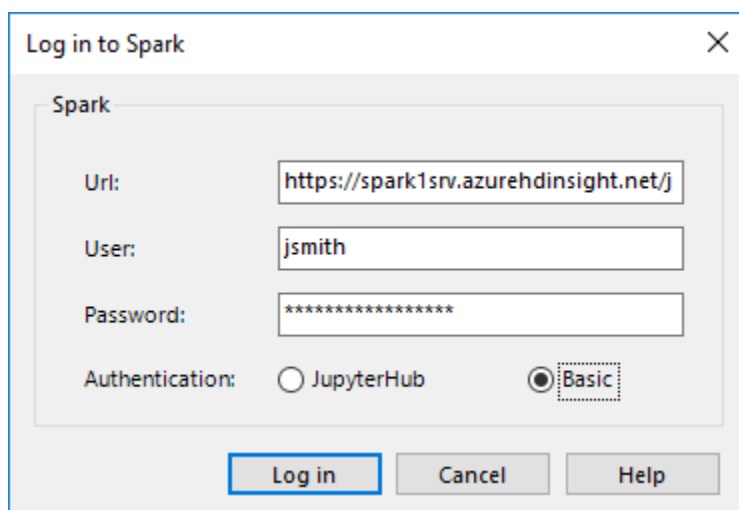
- Click on the Spark server connection icon in the Toolbar:

A screenshot of the 'Log in to Spark' dialog box. The dialog has a title bar with a close button. Inside, there's a section titled 'Spark' with four input fields: 'Url:' containing 'spark1.usccomp.amazonaws.com/jupyter', 'User:' containing 'jsmith', 'Password:' containing a masked password '\*\*\*\*\*', and 'Authentication:' with two radio buttons, 'JupyterHub' (selected) and 'Basic'. At the bottom are three buttons: 'Log in', 'Cancel', and 'Help'.

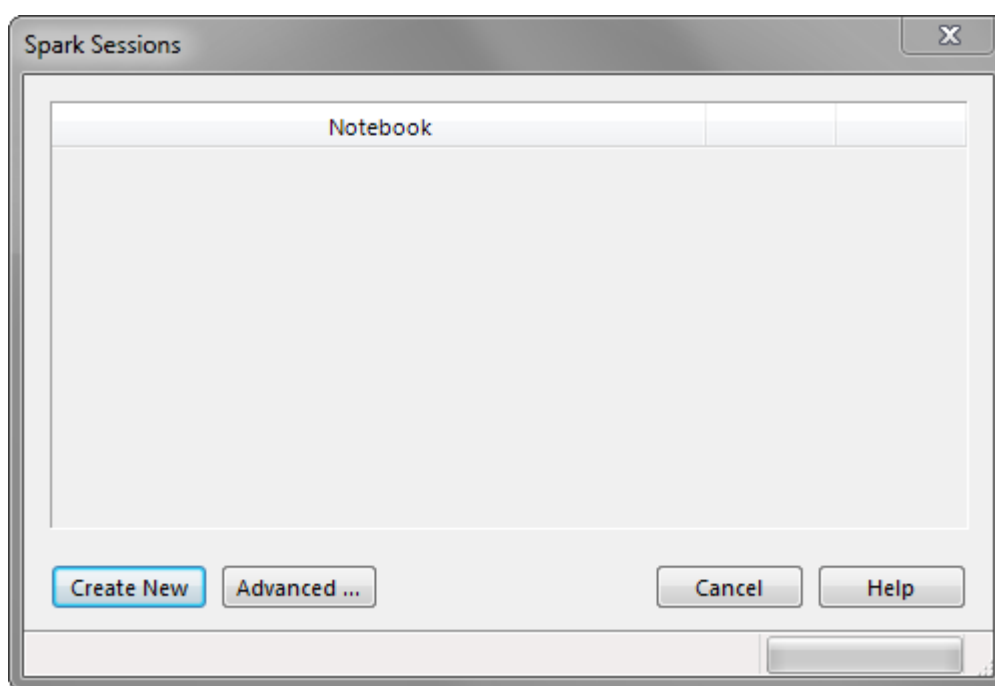
In the *Log in to Spark* dialog:

- Specify the URL for the Spark server you wish to connect to. The https:// prefix is optional.
- Select the **JupyterHub** option if JupyterHub is installed and configured on the Spark server (recommended). Otherwise, select the **Basic** authentication option.
- Enter your user credentials for the Spark server.

**Note:** Some Hadoop cluster hosting services have no Jupyter installed, so *Basic* authentication would be the only choice for such services. One example is Azure HDInsight. If you are connecting to Spark on Azure HDInsight, select *Basic* authentication.



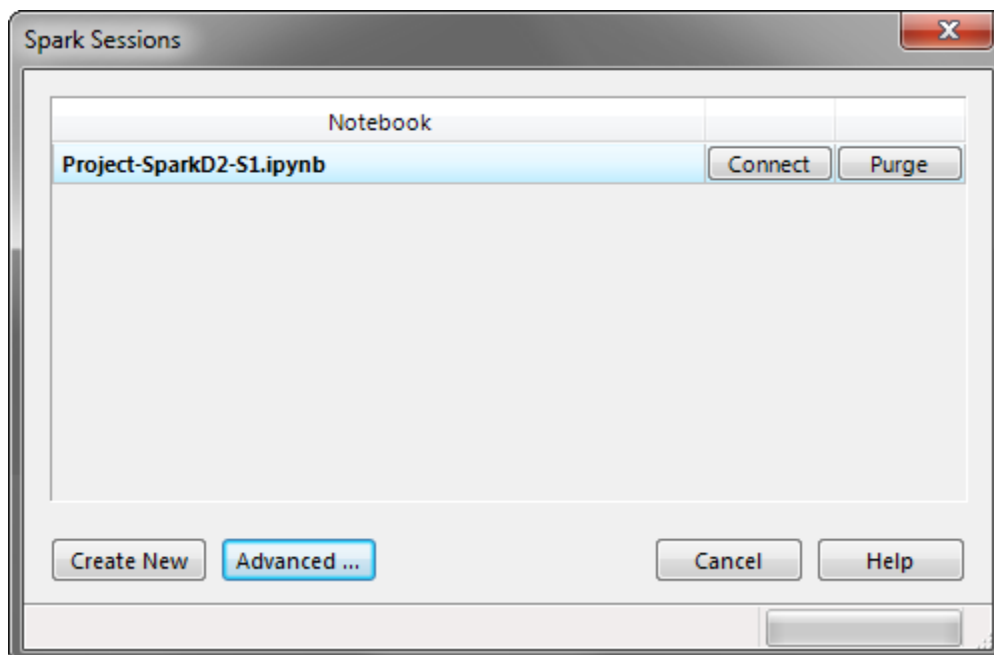
Upon successful login, the Spark Sessions dialog opens. It displays the list of Spark sessions of the specified user ID that currently exist on the Spark server (if any).



The Spark Sessions dialog allows you to:

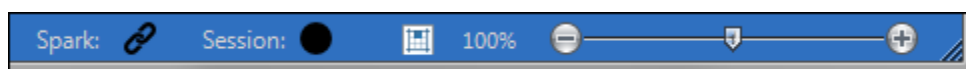
- connect to an existing session (click **Connect** for the desired session in the list)
- purge any of your existing sessions and destroy all its objects (click **Purge** for the desired session in the list)

- click **Advanced** if you want to specify any *SparkContext* parameters
- click **Create New** to create a new session

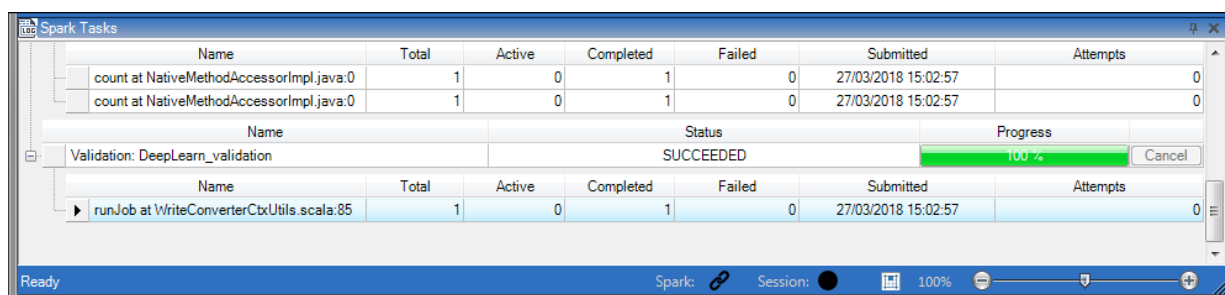


Page | 58

When the session has been created or you have connected to an existing session, the Status bar at the bottom of the application window shows the Spark connection status and the session status. The session status icon in the form of a solid black disk means the session is active. A blank circle means the session is idle. When the session is no longer alive, the Spark connection icon is a broken chain. The notation is similar to that in Jupyter Notebook.



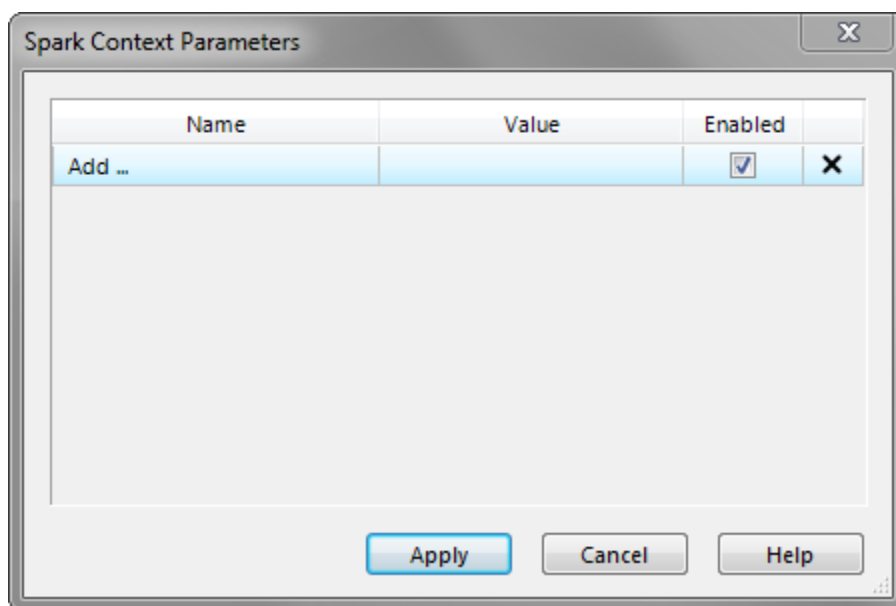
To check the task progress of the current Spark tasks, select the **Spark Tasks** command from the **View** menu. This opens the Spark Tasks frame in the lower part of the Knowledge Studio application window. It shows the name, status, and progress of your Spark tasks in this session. Here is an example of what the Spark Tasks frame may show during a model validation process:



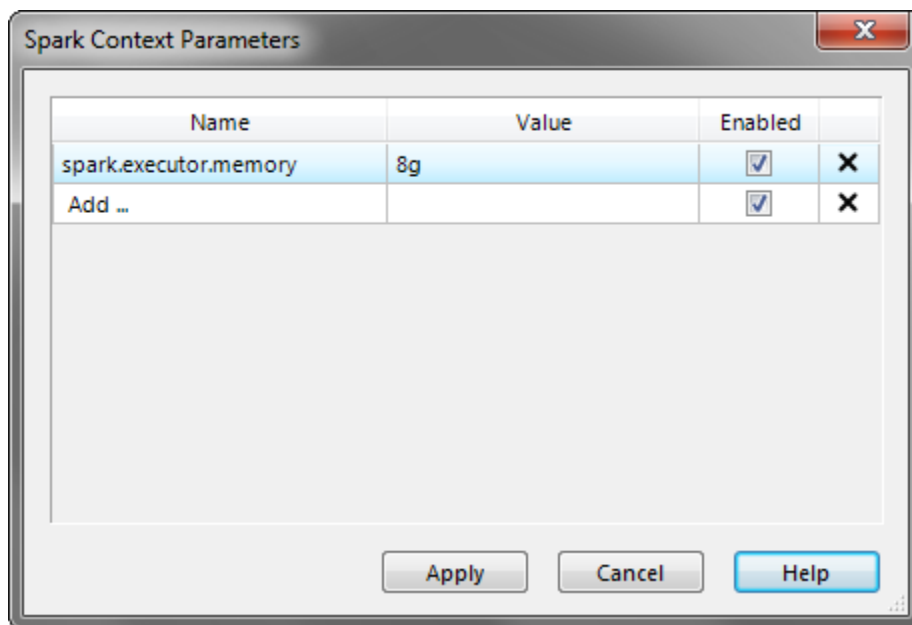
## SparkContext Parameters

If you click **Advanced**, the SparkContext Parameters dialog opens. Click Add... to add a new parameter and specify the desired value. Any previously defined parameters are also displayed in this list. The parameters you enter will be saved for future connections. If you no longer need a parameter, you can either disable it (deselect) or remove it by clicking the x button. The parameters apply to the session being created or connected to.

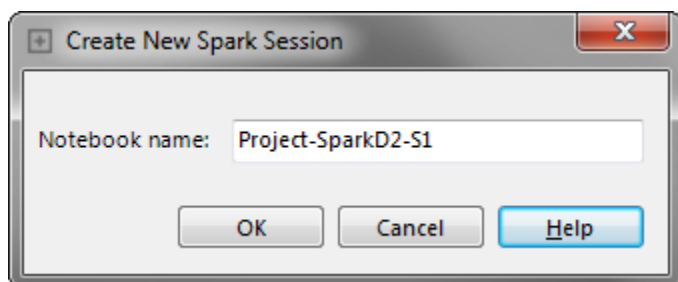
Page | 59



For example, to specify the amount of memory to use per executor process, enter **spark.executor.memory** in the Name field and enter the required amount in the Value field - for example, **8g** for 8 Gigabytes. See the [Spark Configuration documentation](#) for more information. You may want to consult with your cluster administrators to make sure the SparkContext parameter values you intend to set for your session are within the allowed limits.

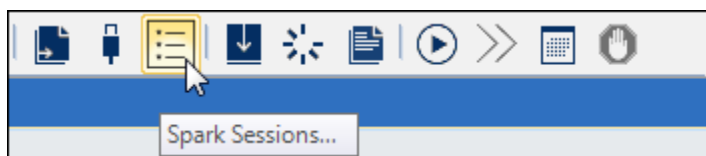


Once you click **Create New** to create a new session, specify the name for the new Jupyter Notebook session and click OK.



The progress bar at the bottom right corner shows the Spark context creation status.

At any point during your work, you can view, create new, or manage your Spark sessions by clicking the Spark Sessions icon in the toolbar :



## Disconnecting from the Spark Session



If you just close the current project with a live Spark connection without exiting the application, you will not be disconnected from the session.

You are automatically disconnected (logged out) from your current Spark session if you:

- close the Knowledge Studio application
- reconnect to the Data Mining Engine using the **File | Connect** command
- reconnect to the current Working Directory using the **Set Working Directory** or **Recent Working Directories** command under the **File** menu

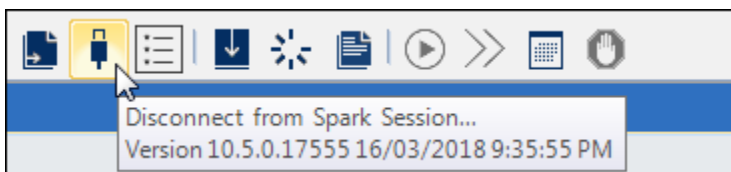
Page | 61

In all of the above cases, after the disconnection, the Spark session is **not** purged and the objects are **not** destroyed.

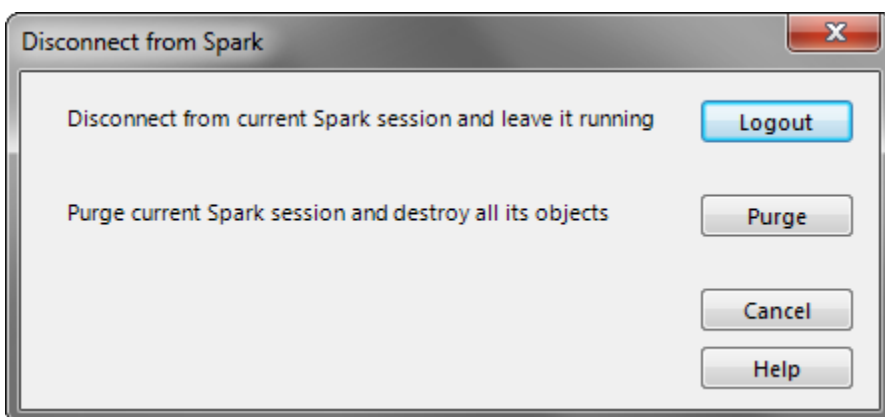
To explicitly disconnect from the current Spark session, click the Disconnect icon in the toolbar:



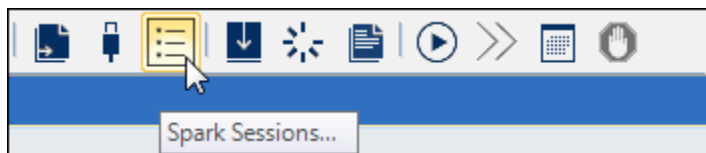
Note that when you hover your mouse over the Spark Session Disconnect button, the tooltip shows information about the current Spark session (version, date, and time in UTC - Coordinated Universal Time).



Upon clicking the Disconnect icon, the dialog "Disconnect from Spark" will open. Click "**Logout**" if you want to disconnect without purging the current Spark session and destroying all its objects. Otherwise, click **Purge**.



Alternatively, disconnect from your session using the Spark Sessions icon in the toolbar 



## Spark Settings

Before proceeding with analytic tasks on Spark, review the default configuration parameters for Apache Spark processing that may be essential to optimize performance. Adjust these parameters if necessary. These defaults can be set via the **Tools | Preferences** command. Please see the topic [Preferences: Spark Settings](#) for details.

Make sure that the time zone is set correctly on the Spark server. This is important for the functions that deal with timestamp data type.

**Note:** Users can launch Jupyter notebook from within the Knowledge Studio application. For this integration feature to work, the URL for the Jupyter notebook server must be accessible from the local machine where KS Workstation is installed. In the client/server mode, it must be accessible from both the client and server machines.

# Projects and Workflows

## Projects

Page | 63

A project contains [workflows](#) and the objects produced by executing operations in these workflows, i.e., datasets, models, reports, etc. Each project is represented by a folder in the file system.

The Project Explorer (the left panel) shows all projects in the current [Working Directory](#), which is set using the **File | Set Working Directory** command. You can view and work with multiple projects in the Project Explorer at the same time.

Each new project you create by selecting **File | New Project** from the menu is represented by a physical folder in your current Working Directory. Individual project components (models, model analyzers, etc.) are stored as files in the project folder.

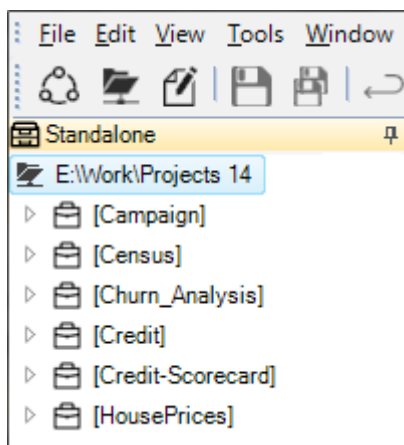
Conversely, every folder inside the current Working Directory is considered a project. In fact, you can copy files to a project folder and they will appear in the Project Explorer after the project is refreshed.

Any Altair project can be read by the Altair Data Mining Engine of the same version on any supported operating system.

## Project Explorer

The Project Explorer panel on the left displays the list of all projects in your Working Directory. Expand a project by clicking on the triangle on the left-hand side to view its contents.

The Working Directory name is displayed at the top level of the Project Explorer panel. All projects contained in the Working Directory are displayed beneath its name. In this example, there are seven projects in the Working Directory.



The hierarchy of objects within a project is displayed in a typical Windows Explorer-like fashion. At the top of the hierarchy is the project. Attached to the project are dataset objects, models, reports, generated code, model analyzers, and possibly some desktop documents. Decision trees and predictive models are normally child objects of datasets. Some types of objects are always automatically placed at the highest level (as a child object of the project itself), for example, Model Analyzer.

Page | 64

To open any object, such as a dataset, double-click on it in the Project Explorer or right-click on it and select **Open** from the context menu. Alternatively, select the item and press Enter. This will display the default view of the open object (for example, the Overview Report of a dataset).

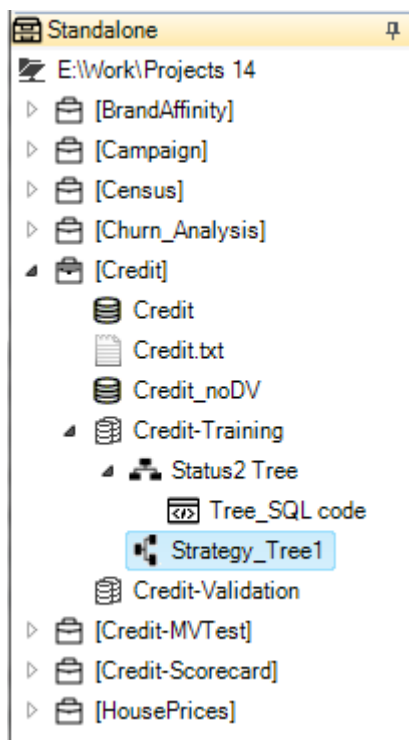
For easy navigation between the currently open objects in your project as well as between currently open projects, make sure you use the Workbook Tabs mode (menu: **Tools | Preferences**, go to the **General** tab and select Workbook Tabs). In this mode, the set of tabs at the top of the workspace window represents objects and projects that are currently open.

Use the view tabs along the bottom of the object window to navigate through its views.

Normally, Project Explorer appears on the left side in a *docked* position. You can choose to auto-hide the Project Explorer window or make it floating and move it anywhere within the application window. To this end, just right-click its title bar and select **Floating** or **Auto-Hide**, or click its title bar and drag it to the desired location.

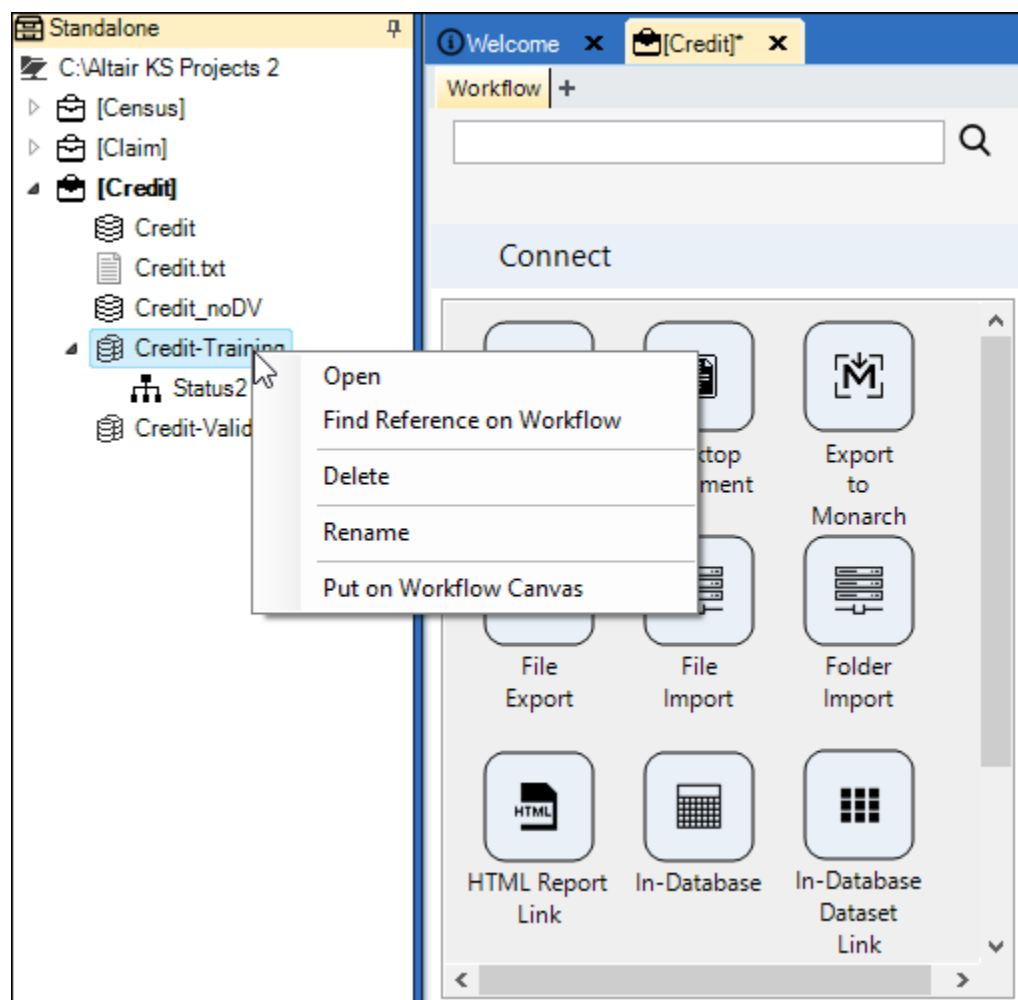
In the example below, the Census project has been expanded to show all the elements that belong to it. The Census project contains a **dataset** (Census) with two *partitions* (Census-Training and Census-Validation), a **Decision Tree** (Income), a *Strategy Tree* (Strategy1), *SQL code* generated for the Income Tree (SQL rules), and *Model Analyzer* for analyzing the decision tree performance.

The title bar of the Project Explorer indicates whether you are working with data mining engine installed on the local machine (Standalone) or a remote server (Client/Server).



You can rename any project or object within the project tree. The rename operation will be tracked in the activity log so you can see the history by selecting **View | Activity Log** from the menu. To rename an object select it in the Project Outline and choose the **Rename** command from the **Edit** menu. You can also right-click on the object and select **Rename** from the context menu, or press the F2 key.

All objects within the project explorer have the right-click context menu. Common commands such as **Rename**, **Delete** and **Open in Explorer** are shown in the context menu. You can use the context menu for dataset objects to insert Measure of Predictive Power or a desktop document.



It is possible to select more than one object at a time. This is done by holding down the Control (Ctrl) or Shift keys while clicking the mouse. In this case, the list of commands in the right-click menu shows only operations applicable to multiple datasets. Operations such as **Delete** can be performed on multiple objects at a time.

See also:

[Setting the Working Directory](#)

[Creating and Deleting Projects](#)

[Workflows](#)

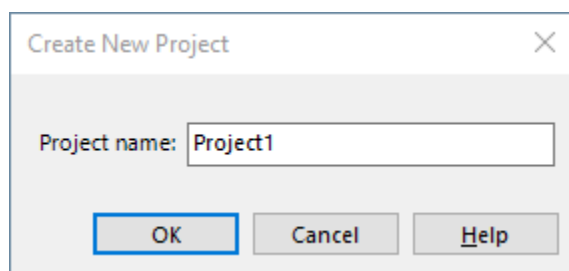
### **Creating Projects**

A project contains process workflows and the results of their execution, i.e., analysis objects such as datasets, models, reports, etc. The Project Explorer panel shows all projects in the current

[Working Directory](#), which is set using the **File | Set Working Directory** command. See [Projects](#) for details.

To create a new project in the current Working Directory, select the command **File | Create Project** from the main menu.

Page | 67



Enter the project name and click OK. A new project with this name will appear in the Project Explorer.

The project name:

- **cannot** contain the following special characters:
  - [ ] " \* + # & ! % ( ) { } ` ' @ \$ ^ , . / : ; < = > ? \ |
- **can** contain spaces, for example, "**Credit Risk Project**".

### **Deleting Projects**

To delete a project, right-click on the project name in the Project Explorer and select **Delete** from the context menu. Alternatively, click on it and choose **File | Delete Project** from the main menu. The project with all its contents will be deleted from the Project Explorer and from the disk.

### **See also:**

[Setting the Working Directory](#)

[Projects](#)

[Workflows](#)

## **Workflows**

A **workflow** defines a data mining process in terms of operational nodes and data flows. A typical process starts with extraction and preparation of the data, goes through modeling and ends with model deployment or outputting deployment code, and optionally exporting data.

---

Similar to a program, a workflow is a description of interconnected operations together with their inputs and outputs which can be re-run multiple times with different inputs.

The Knowledge Studio workflow automation feature allows you to easily construct such workflows in a visual, interactive way: by dragging, dropping and connecting process nodes on the visual canvas and specifying their inputs, outputs, and parameters.

Page | 68

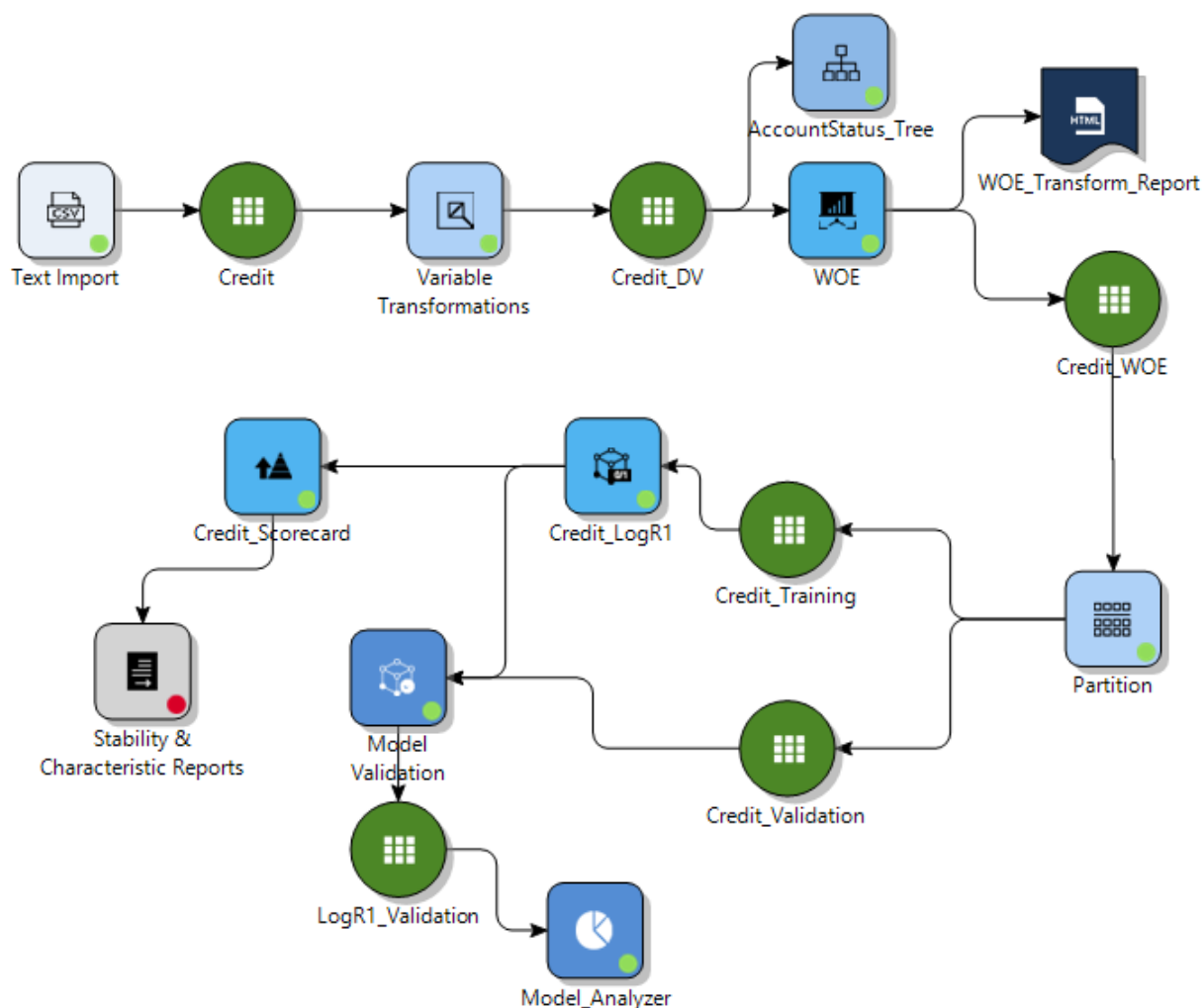
The process defined by a workflow constructed this way can easily be re-run on updated or refreshed data inputs. Particular data transformations, modeling, and other operations you defined in a workflow can be repeated as long as the structure of the new input data is the same.

Knowledge Studio workflow execution features allow you to re-run individual nodes and segments of the workflow as well as the entire process.

One of the most useful deployment features is automatic SAS Language code generation for all stages of the Scorecard building workflow. Code can be generated for the whole workflow, sub-processes, or individual nodes. Then it can be copied directly into SAS System and run there. Other types of code such as SQL Function can also be generated.

Here is an example of a workflow that builds a decision tree, a logistic regression, a scorecard, scores it and analyzes its performance:

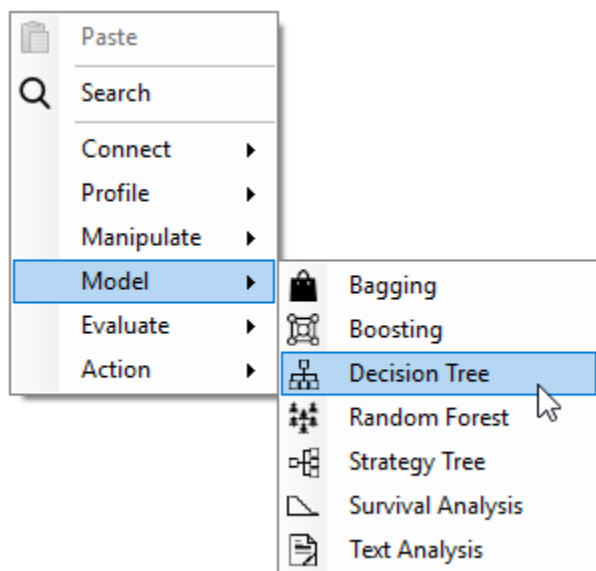




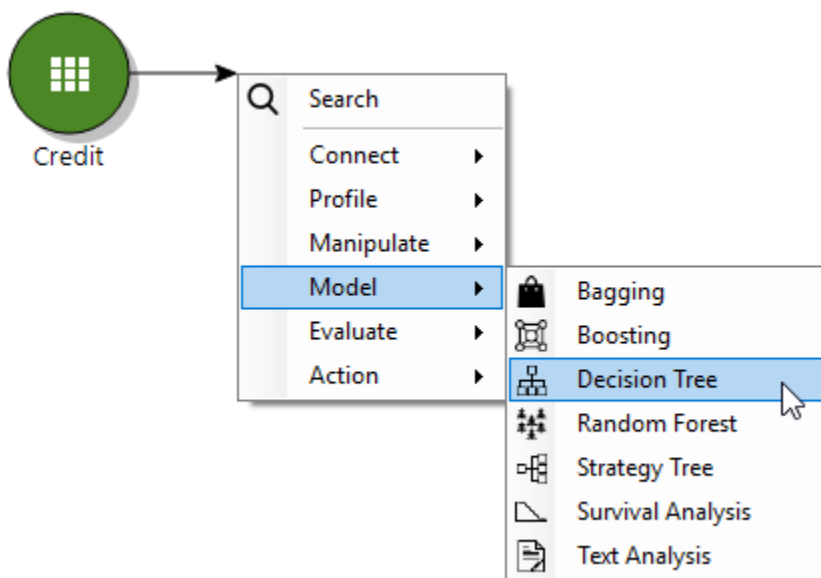
## Adding Nodes to Workflow

There are several ways to add nodes to the workflow canvas:

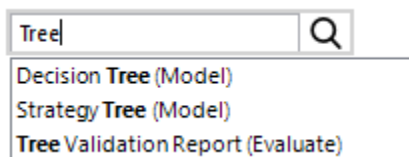
1. Drag-and-drop: Drag the desired node from a functional palette (Connect, Profile, Manipulate, Model, Evaluate, or Action) to the workflow canvas.
2. Right-click on the canvas and select the desired node name from the hierarchical context menu.



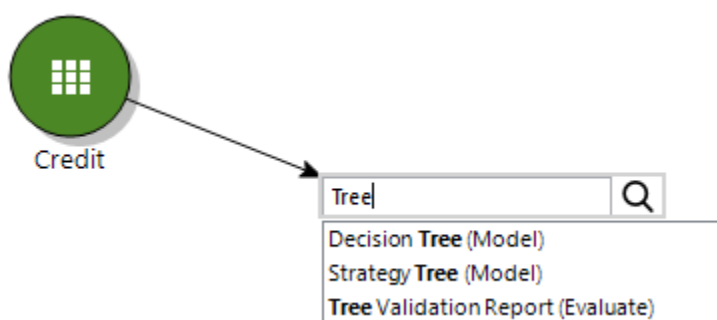
3. Drag a connection arrow from the input node to an area on the workflow canvas. Release the mouse and select the desired node category and then the node name from the context menu.



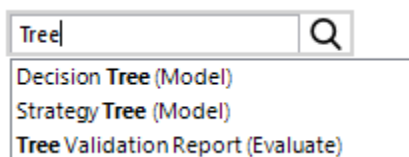
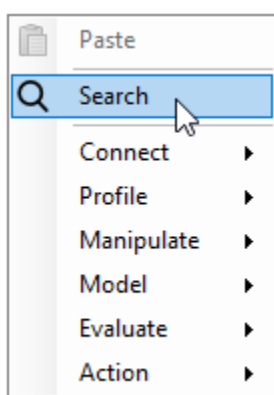
4. Using a keyboard shortcut: Click on the workflow canvas background and press **CTRL + I** on the keyboard. ("I" for "Insert"). The Search box will appear on the canvas near the mouse pointer. Start typing the name of the node you need to insert on the workflow. The auto-complete feature suggests the variants as you type. Select the desired node name from the drop-down list of suggested nodes.



If you select an existing node on a canvas (by clicking on it) right before pressing **CTRL + I**, the search box appears with the connection arrow from the assumed input node. Once you are finished specifying the name of the desired node in the search box, the new node will be added, already connected to the input node.



5. Right-click on the workflow canvas background and select the **Search** command from the context menu. The Search box will appear. Start typing the name of the node you wish to insert on the workflow. The auto-complete feature suggests the variants as you type. Select the desired node name from the drop-down list of suggested nodes.

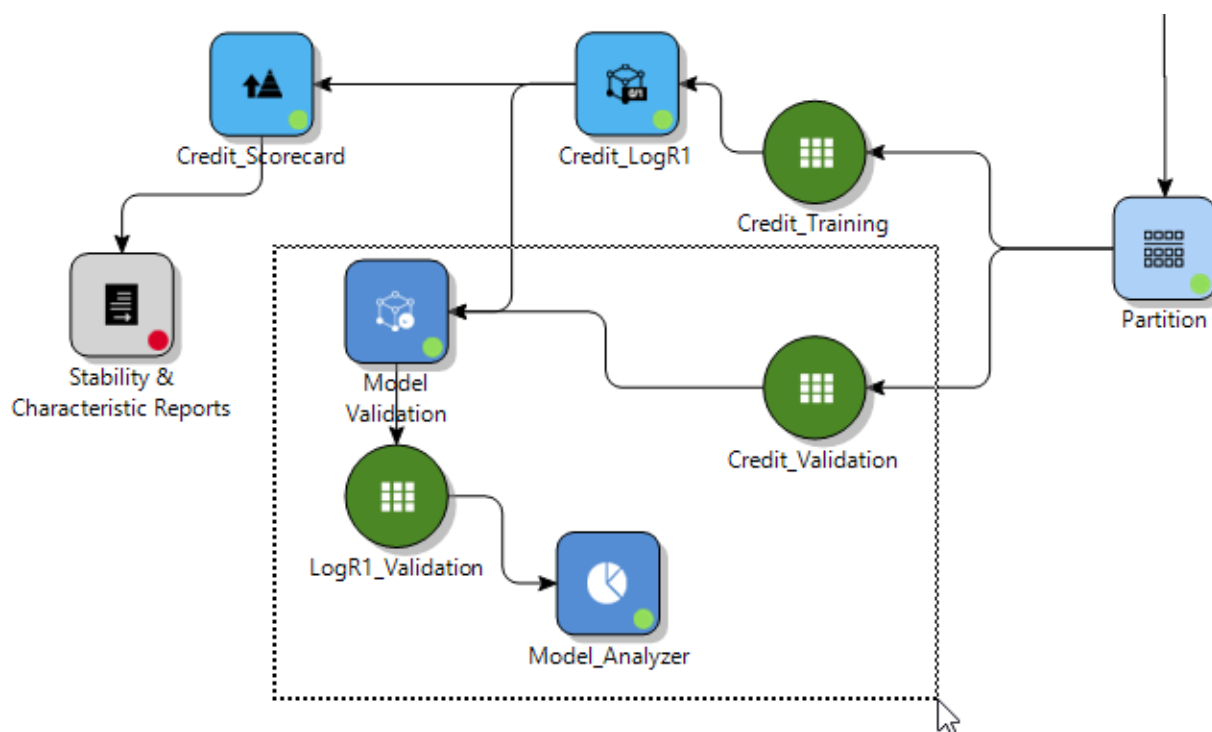


## Manipulating Nodes and Groups of Nodes

Nodes can be moved around the workflow canvas simply by dragging and dropping.

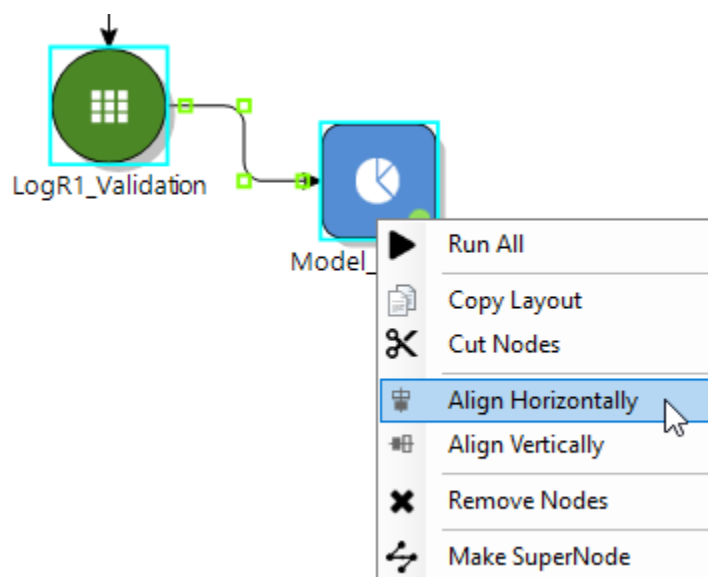
Page | 72

- To move a single node, click inside the node (closer to the centre rather than the edge) and drag it to the desired location.
- To move a group of nodes, select the desired group first by dragging a rectangle over the area enclosing these nodes as shown in the example below.



The nodes in the selected group will become highlighted with a light-gray frame around each node. Then click inside any node in the group (closer to the centre rather than the edge) and drag it to the desired location. This will move the whole group, provided that it is still highlighted.

- To align a group of nodes, select the desired group by dragging a rectangle over the area enclosing these nodes as shown above. The group becomes highlighted. Right-click on any node in the group and select "Align Horizontally" or "Align Vertically" from the context menu, as shown below.



## Supernodes

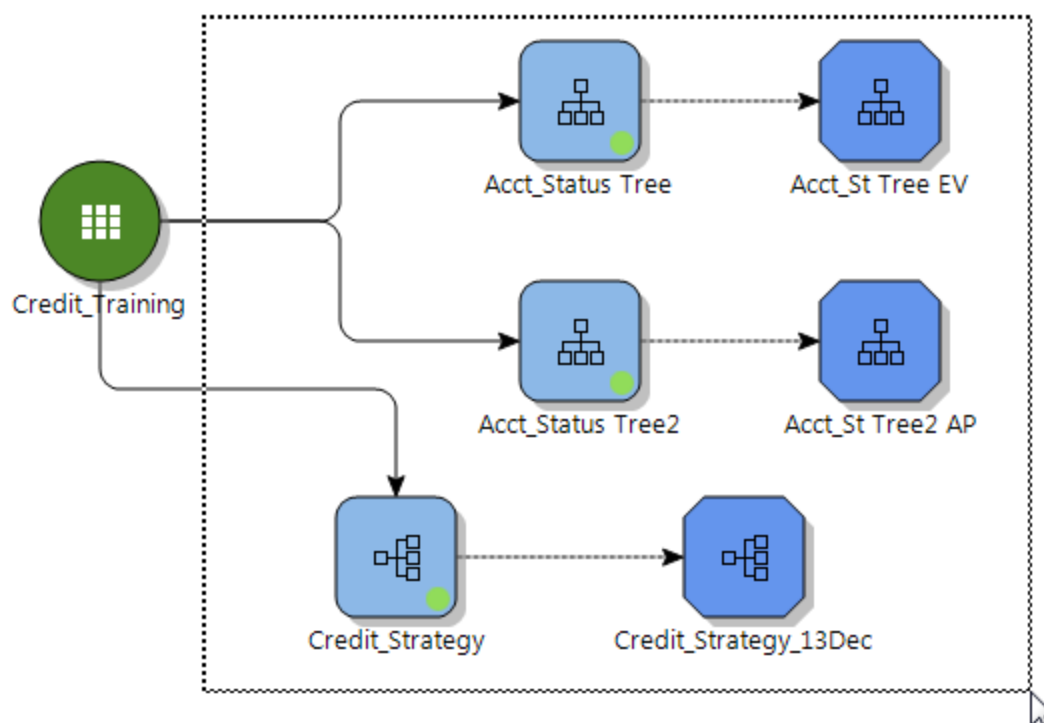
Supernodes represent a convenient way to organize workflows.

When building or viewing large and complex workflows with a lot of nodes and connections, you can merge any group of nodes into a single "**supernode**". A supernode is a cluster of nodes merged into a single node. This way the whole sections of a workflow can be contained inside a single node for operational and viewing convenience. The functional schema of the workflow is preserved as the supernode's input and output connections are the same as those of its members. A supernode can be ungrouped at any time to restore the original representation of the corresponding group of nodes on the canvas.

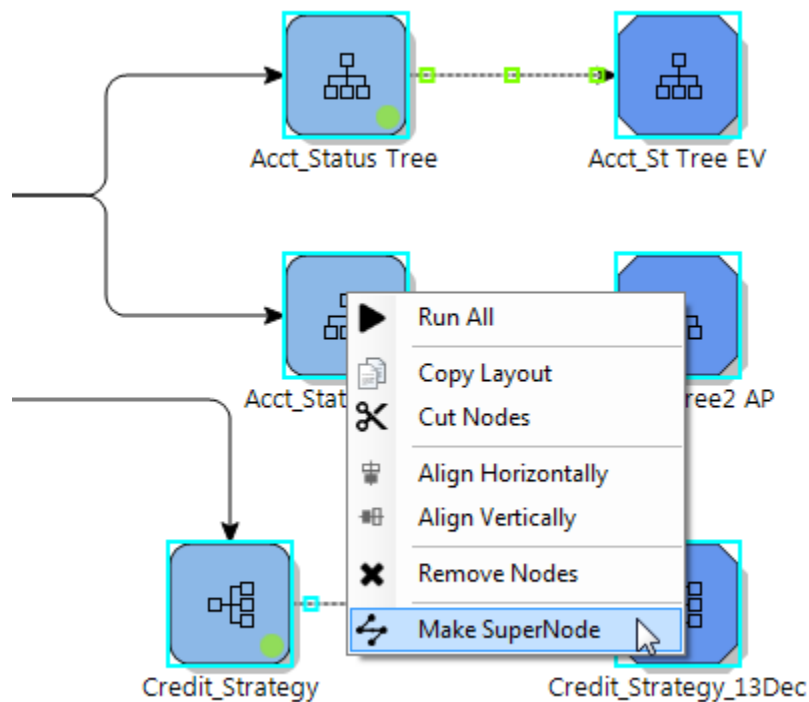
To create a supernode, select the desired group of nodes in any of the following ways:

- by dragging a rectangle over the area enclosing these nodes as shown in the example fragment below; or
- by clicking on the desired nodes while holding down the CTRL or SHIFT key.

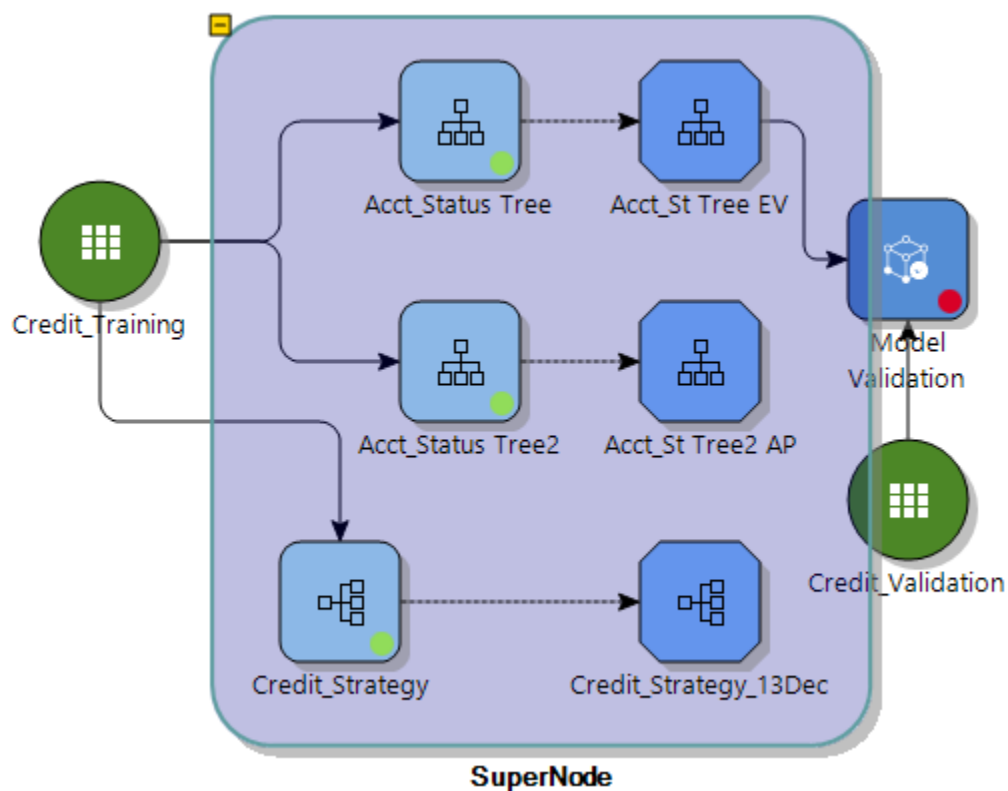
The nodes in the selected group will become highlighted with a light-gray frame around each node. You can add / exclude nodes to / from the current selection by clicking on the (un)desired nodes while holding down the CTRL key.



Right-click on any node in the highlighted group and select **Make SuperNode** from the context menu, as shown below:

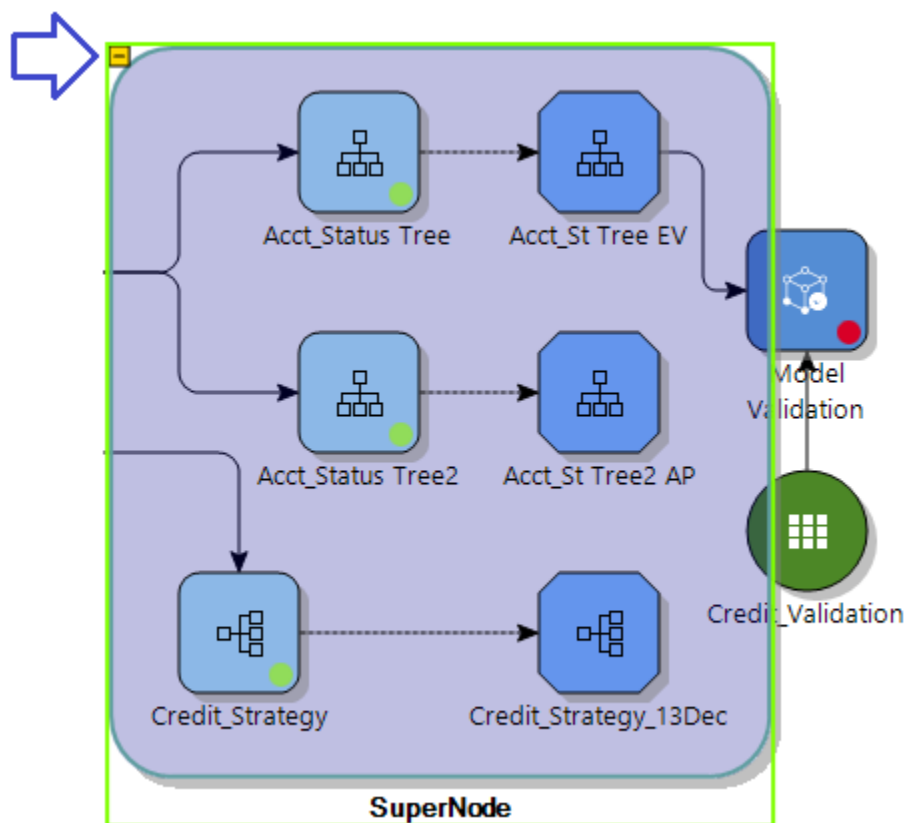


A light-blue frame enclosing the selected group of nodes indicates the supernode. Any nodes that were not in the original selection are shown as partially shaded. In the example below, only the 6 tree nodes comprise the supernode.

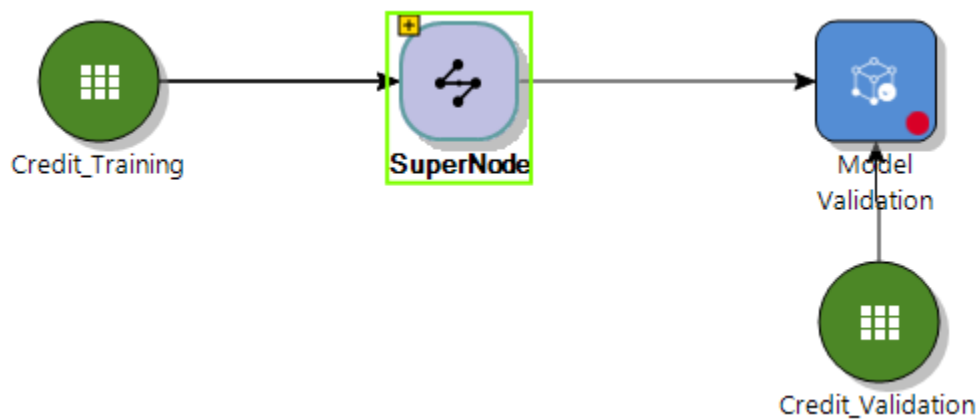


To merge (collapse) the group into a single node, click on the "minimize" control at the upper-left corner of the supernode frame.





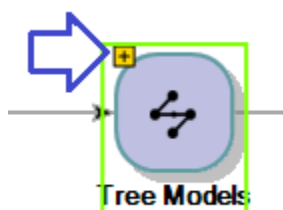
As a result, the highlighted group is merged into a single node labelled "SuperNode".



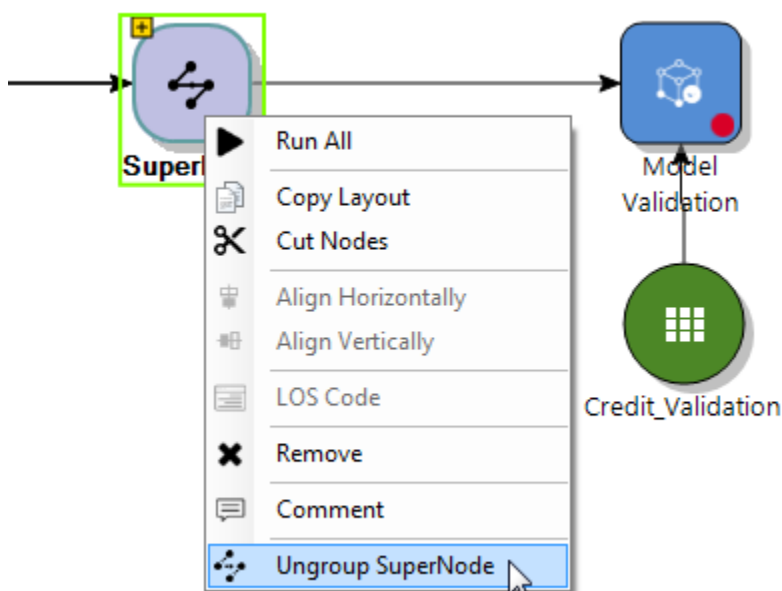
To rename a supernode, click on its label and edit the text as necessary.



To expand a supernode, click on the plus sign (the "expand" control) at the upper-left corner of the supernode. The supernode will be expanded (but not ungrouped), showing all nodes within it.



To ungroup a supernode, right-click on it and select **Ungroup SuperNode** from the context menu. This can be done in any state of the supernode (compact or expanded).



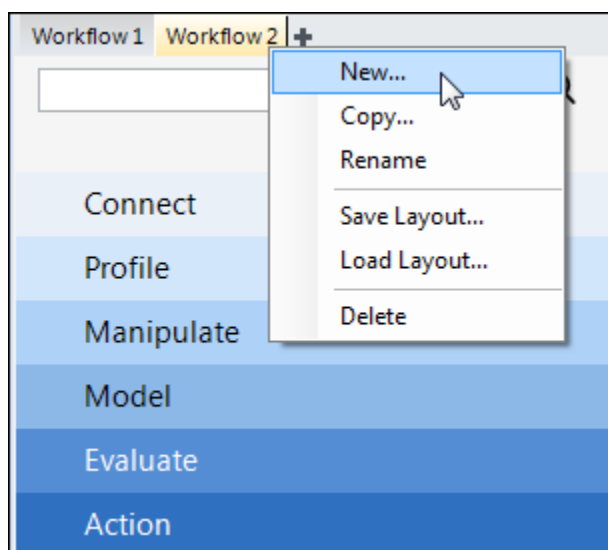
### Other operations with Supernodes:

- To add any node on the canvas to a Supernode click on it to select, then press the SHIFT key and drag the node to the Supernode while holding down the SHIFT key. You can also drag a node from a functional palettes directly to a Supernode in the same way.

- To remove a node from a Supernode, expand the Supernode if it is in the collapsed form, click on the node you wish to remove, then press the SHIFT key and drag the node out of the Supernode. The same can be done with multiple nodes at the same time.
- To copy or move a Supernode to another workflow, right-click on it and select "**Copy Layout**" or "**Cut Nodes**" from the context menu. Then switch to another Workflow tab and paste the supernode.
- Moving (dragging) nodes within a Supernode towards its borders will resize the Supernode.
- You can add a comment to a Supernode the same way as to any normal node (right-click and select the **Comment** command from the context menu).
- To run a Supernode right-click on it and select **Run All**. This will run all nodes within the Supernode.
- It is possible to create nested Supernodes (put one supernode inside another).

## Managing Workflows

You can create more than one process in a single workflow canvas, and also have multiple workflow canvasses in a single project. If one canvas gets too crowded, create a new one by either clicking the **plus** sign or by right-clicking on the Workflow tab at the top of the Node Group panel and selecting **New** from the context menu. Switch between multiple workflows by clicking on their tabs. To change the order of the Workflow tabs, drag one of the tabs to the desired position relative to other tabs. You can also use the right-click context menu of the Workflow tabs to rename or delete workflows. The same commands are available under the Workflow menu.



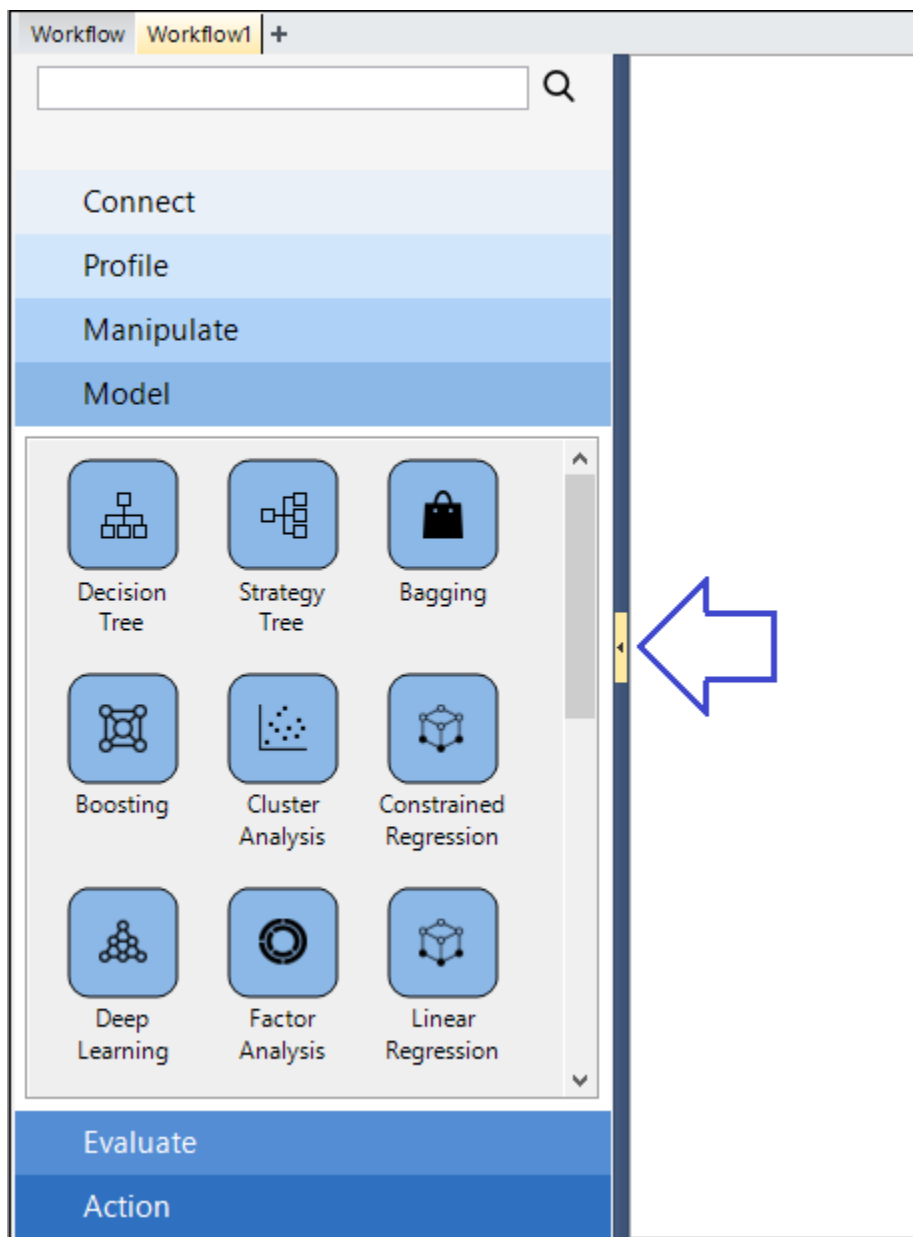
While creating and running workflows you can also easily keep track of the objects produced by running them. The project folders and objects created inside them are displayed in the **Project Explorer** panel on the left-hand side. Aside from showing all projects in your Working Directory, Project Explorer allows you to view outputs and delete the outputs that are no longer necessary.

## View Options

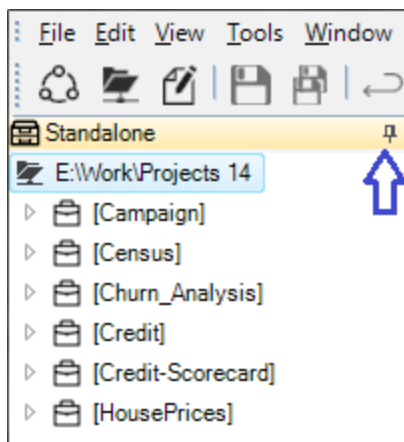
### Docking Panels

If you need to build or view large and complex processes in your Workflow canvas, you can save some space by docking the Node Groups panel to the left. Just click on the arrow in the middle of the separator between the Node Groups panel and the canvas:

Page | 80



You can also auto-hide the Project Explorer panel. Just click on the pin at the top right corner of the Project Explorer panel. To switch back from the auto-hide mode, open it and click on the pin again.



### **Switching to Workflow from Object Views**

When a specific object in a workflow is open (a tree, a dataset, etc.), you can easily switch from any object view back to the workflow in any of the following ways:

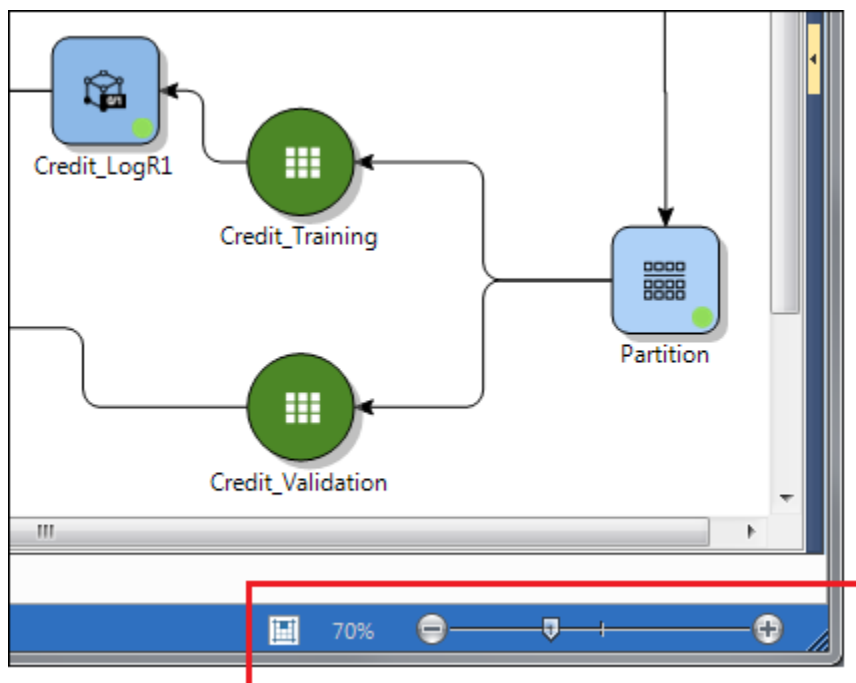
- using the toolbar button  ("Switch to Workflow")
- clicking on the Workflow tab representing the current project at the top of the object view
- double-clicking on the project name in the Project Explorer panel
- using the keyboard shortcut CTRL TAB (if the workflow was open immediately before the current object view)

### **Pan and Zoom**

#### **Zoom:**

You can easily navigate across a large workflow using pan and zoom features when the workflow is larger than the application window. Zoom in or out using the standard zoom control with a slider provided in the Status bar at the bottom right corner, as shown in the screenshot below. Zooming can be performed in several ways:

- moving the slider or clicking the +/- icons on the zoom control;
- scrolling the mouse wheel while pressing down the CTRL key;
- using the keyboard shortcuts CTRL - (zoom out), CTRL + (zoom in), and CTRL 0 (restore zoom to 100%).

**Pan:**

The pan control allows you to drag the workflow canvas around with your mouse instead of using the scroll bars. While holding down the CTRL key on your keyboard, click anywhere in the workflow canvas, hold the left mouse button down and start dragging to move the canvas.

**Mini-map**

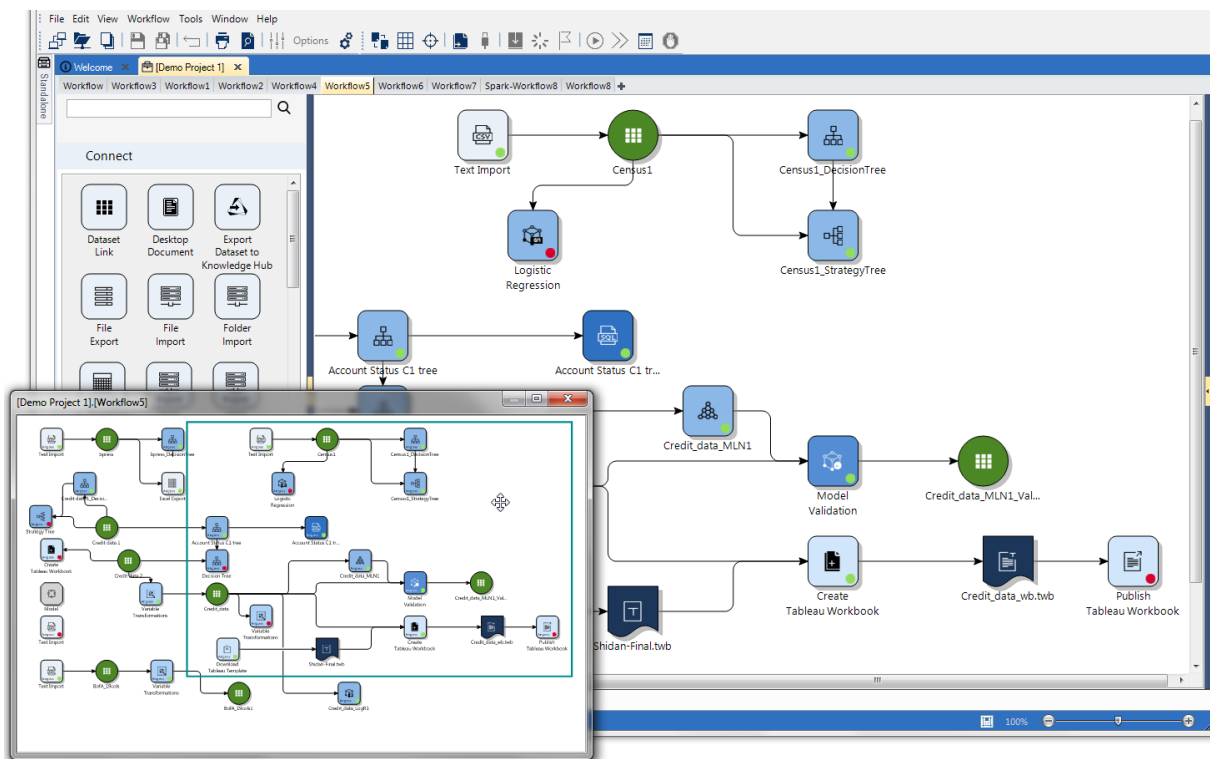
An interactive mini-map of the workflow can be opened by clicking the mini-map button in the

toolbar . It shows the current view of the workflow relative to the whole workflow and allows to easily navigate across large workflows that don't normally fit in a single view, as well as zoom in and out.

A transparent rectangular frame inside the mini-map serves as a movable lens or magnifying glass and indicates the part of the workflow shown in the current view. The fastest way to navigate to the desired part of the workflow is to click on that part in the mini-map. The frame will move and the Workflow view will be updated immediately to show the part you clicked on. Alternatively, click anywhere within the frame and drag it around the mini-map frame, dropping at the desired position. The view of the workflow will be updated dynamically depending on the position of the frame.

You can resize the mini-map, and you can also resize the "magnifying" frame by dragging on any of its corners. Changing the frame size will cause the workflow to zoom in or out.

Each project can have only one mini-map. If you switch to another workflow within the same project, the mini-map will change accordingly to show the current workflow.



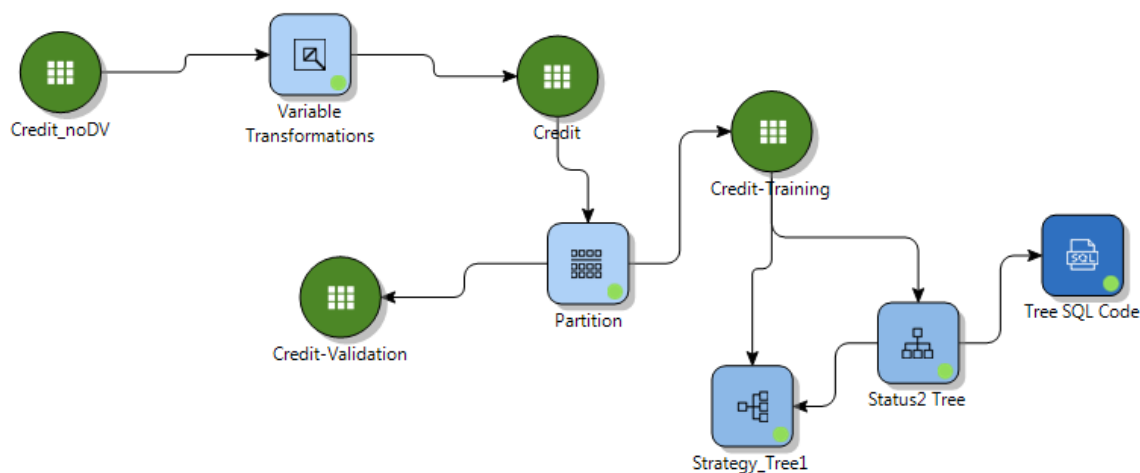
Page | 83

### **Auto-arrange All Nodes in a Workflow**

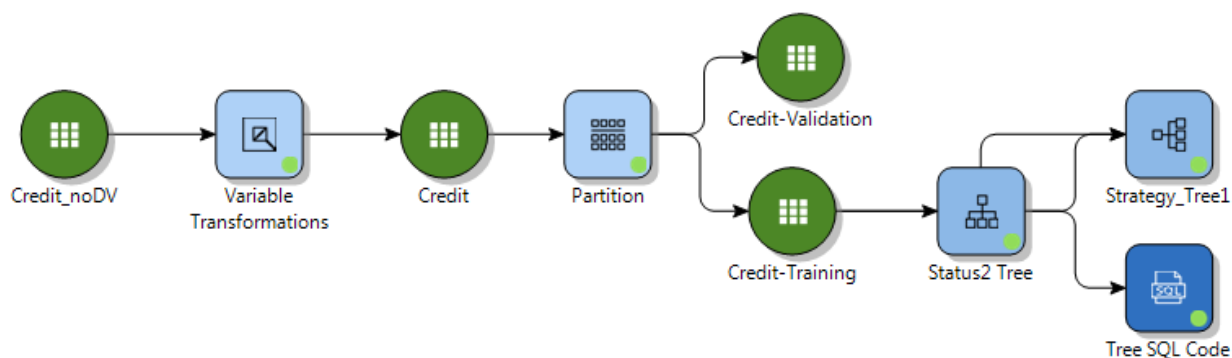
You can auto-arrange nodes in your workflow to align all connections in the horizontal direction by clicking the Auto-Arrange button  in the toolbar. To undo the change, click the button again.

Here is an example of a workflow before and after auto-arranging:

#### **1. Before Auto-Arrange:**



## 2. After Auto-Arrange:



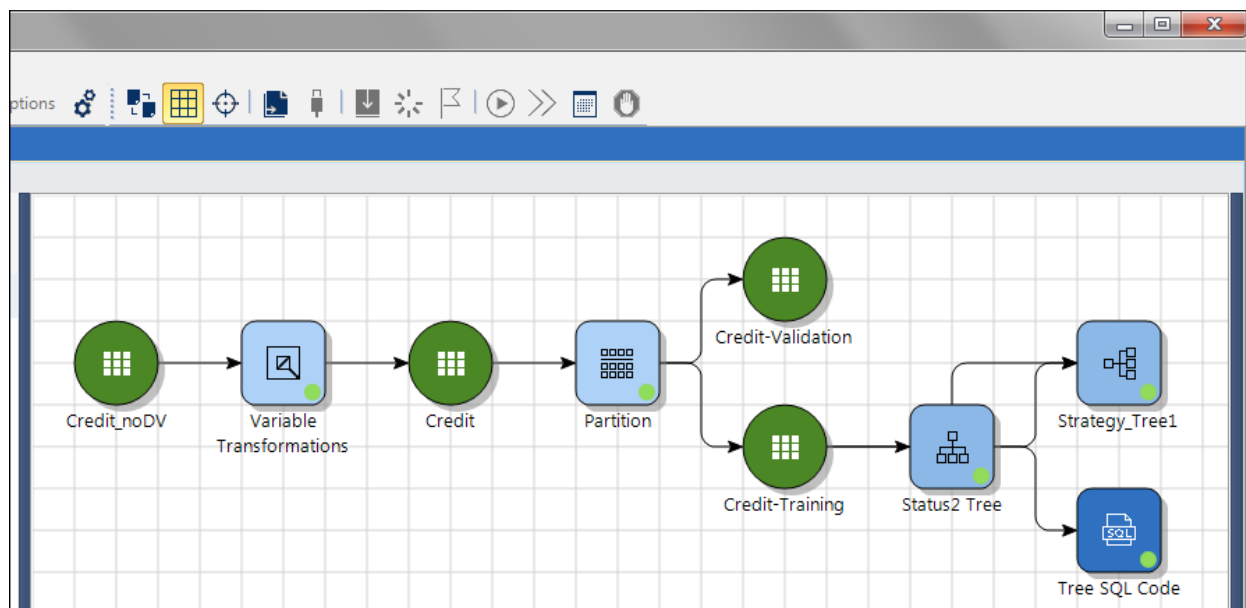
### Undo

To undo an Auto-arrange action or any other node manipulation operation in the workflow, click the Undo button  in the toolbar. Click it several times if you need to undo a series of operations.

### Show Grid

To show the grid in the background of the Workflow canvas, click the Show Grid button  in the toolbar. To hide the grid, click the same button again. When the grid is shown, nodes snap to the grid when being moved.





See also:

[Setting the Working Directory](#)

[Projects](#)

[Creating and Deleting Projects](#)

# Data Source Connections, Data Import and Export

## Accessing Data in Spark Storage

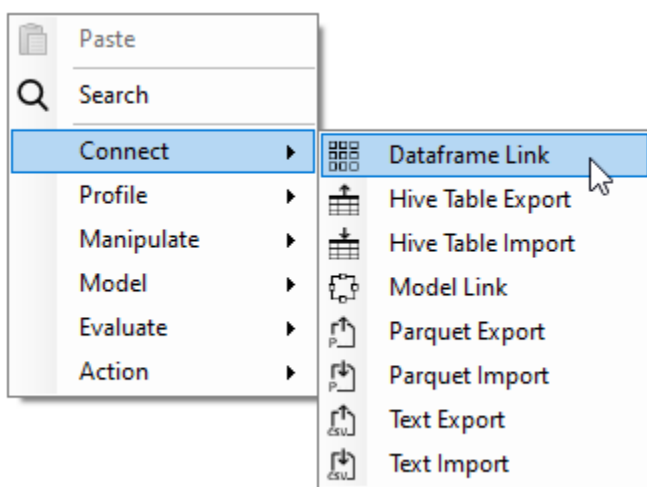
Page | 86

### Spark Dataframe Link

Any Spark dataframe that is not represented in your project can be put on the workflow canvas using the **Dataframe Link** node in the Connect palette.

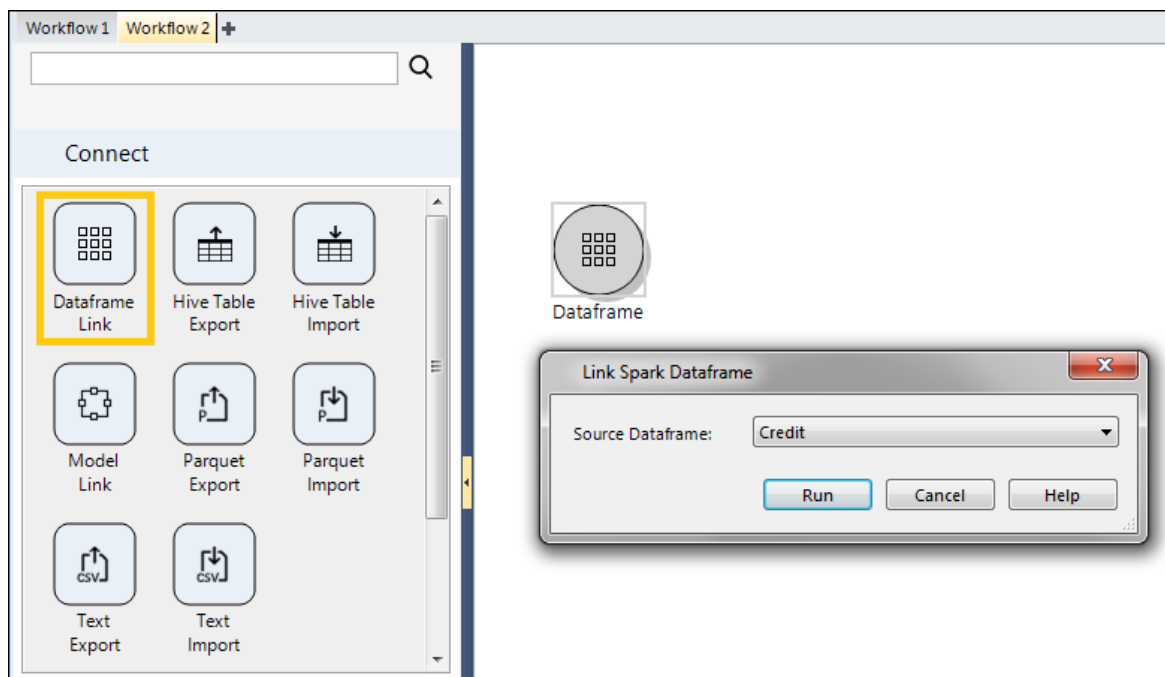
To link an existing Spark dataframe to a dataset node on the workflow canvas:

- Right-click anywhere on the Workflow canvas background. From the context menu, select **Connect --> Dataframe Link** as shown in the example below. A new **Dataframe** node will be created and the Link Spark Dataframe dialog will open.



OR:

- Drag the **Dataframe Link** node from the **Connect** palette onto the canvas. Then double-click the new Dataframe node (or right-click on it and select **Link** from the context menu).



In the Link Spark Dataframe dialog, select the desired dataframe from the Source Dataframe list and click **Run**.

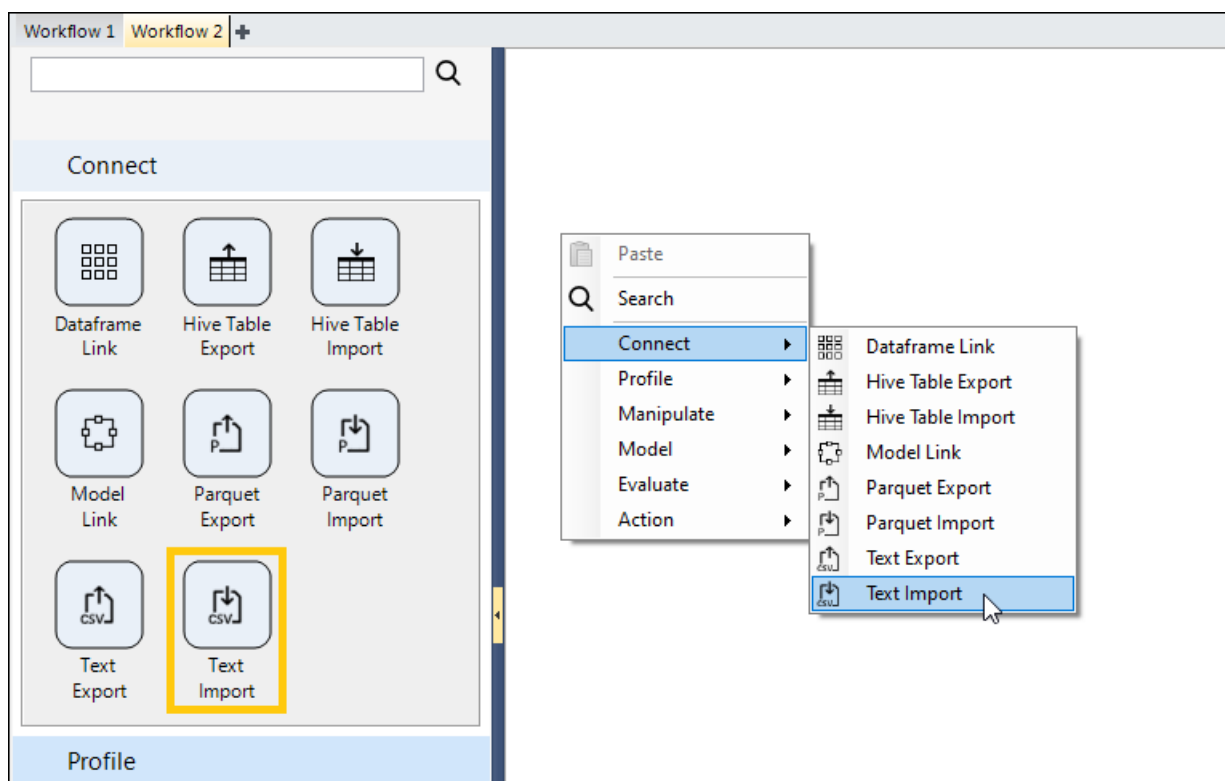
The new Dataframe node is now enabled and linked to the selected Spark dataframe.



## Text Import

To load data from a text file in your Spark storage system into a Spark dataframe in your project workflow:

- Right-click anywhere on the Workflow canvas background. From the context menu, select **Connect --> Text Import** as in the example screenshot below. A new **Text Import** node will be created and the wizard will open. (Note: in this example, the **Spark** filter has been applied for quick search and restricting the available choices to Spark-related functions only).



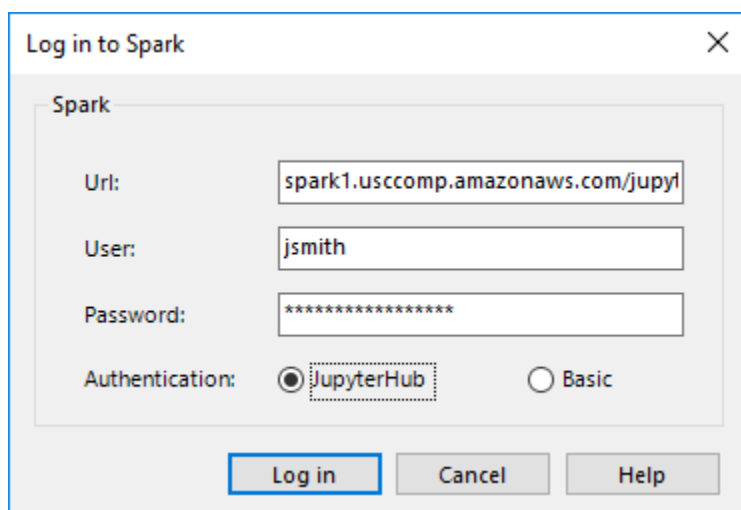
OR:

- Drag the **Text Import (Spark)** node from the **Connect** palette onto the canvas.



Double-click the **Text Import** node or right-click on it and select **Modify** to open the wizard.

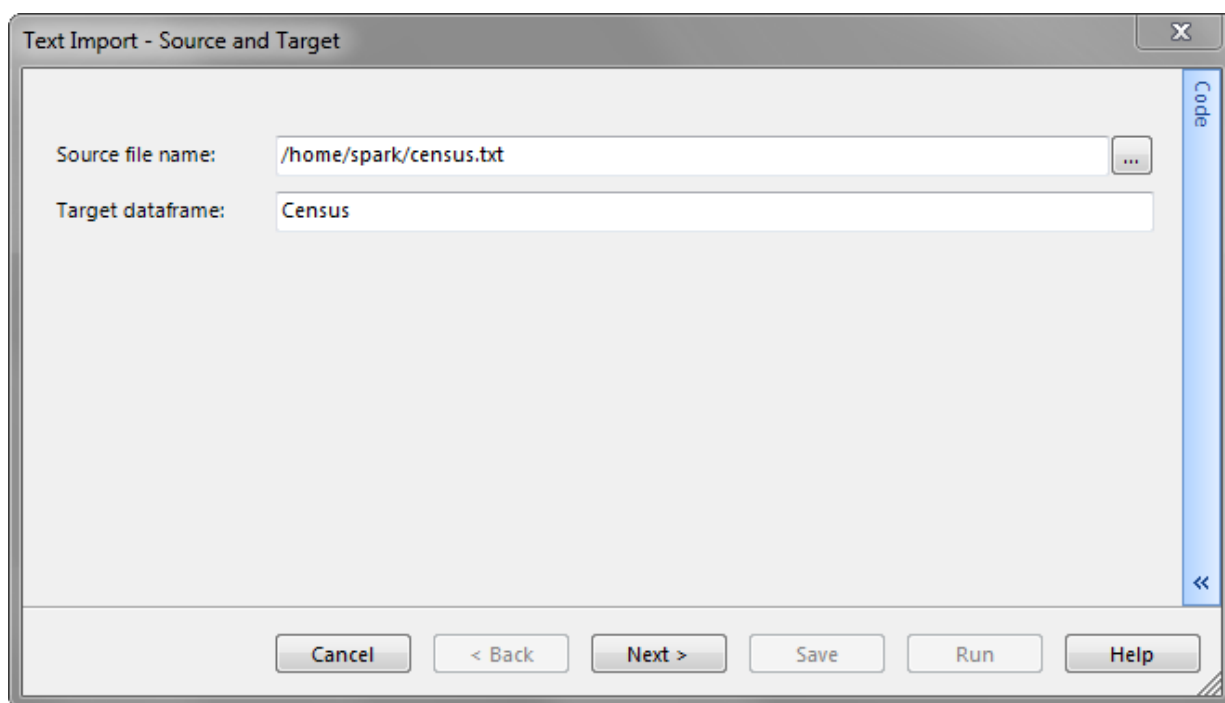
At this point, if you are not already connected to the Spark server, you will be prompted to connect. Specify the URL, user credentials, and the authentication type (**JupyterHub** or **Basic**).



A dialog box titled "Log in to Spark" with a close button (X) in the top right corner. It contains a "Spark" section with the following fields: "Url:" with the value "spark1.usccomp.amazonaws.com/jupyterl", "User:" with the value "jsmith", and "Password:" with a masked value "\*\*\*\*\*". Below these is an "Authentication:" section with two radio buttons: "JupyterHub" (selected) and "Basic". At the bottom are three buttons: "Log in", "Cancel", and "Help".

Once connected to the Spark server:

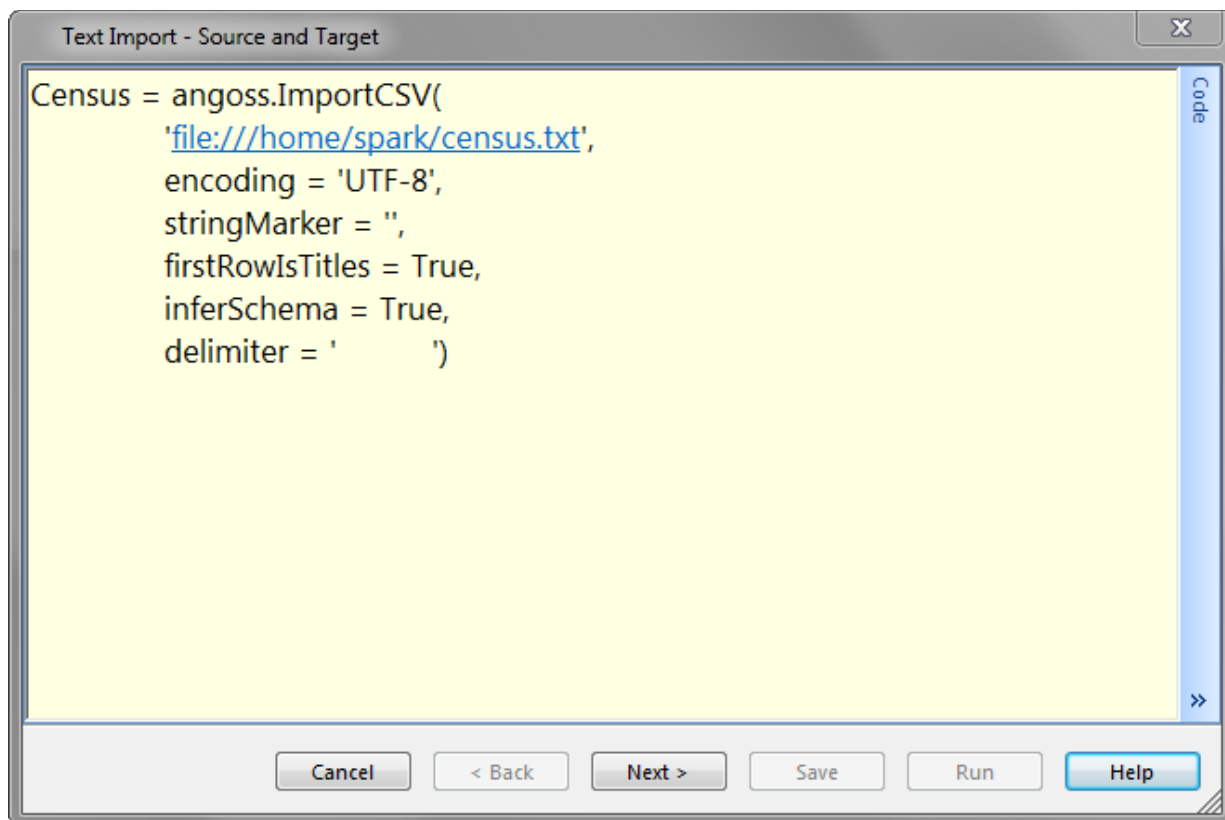
- Browse the server to select the source file to import.
- Specify the name for the new dataset in the *Target dataframe* field



A dialog box titled "Text Import - Source and Target" with a close button (X) in the top right corner. It contains two fields: "Source file name:" with the value "/home/spark/census.txt" and a browse button (...), and "Target dataframe:" with the value "Census". On the right side, there is a blue vertical bar labeled "Code" with a double arrow (<<) at the bottom. At the bottom of the dialog are five buttons: "Cancel", "< Back", "Next >", "Save", and "Run", followed by a "Help" button.

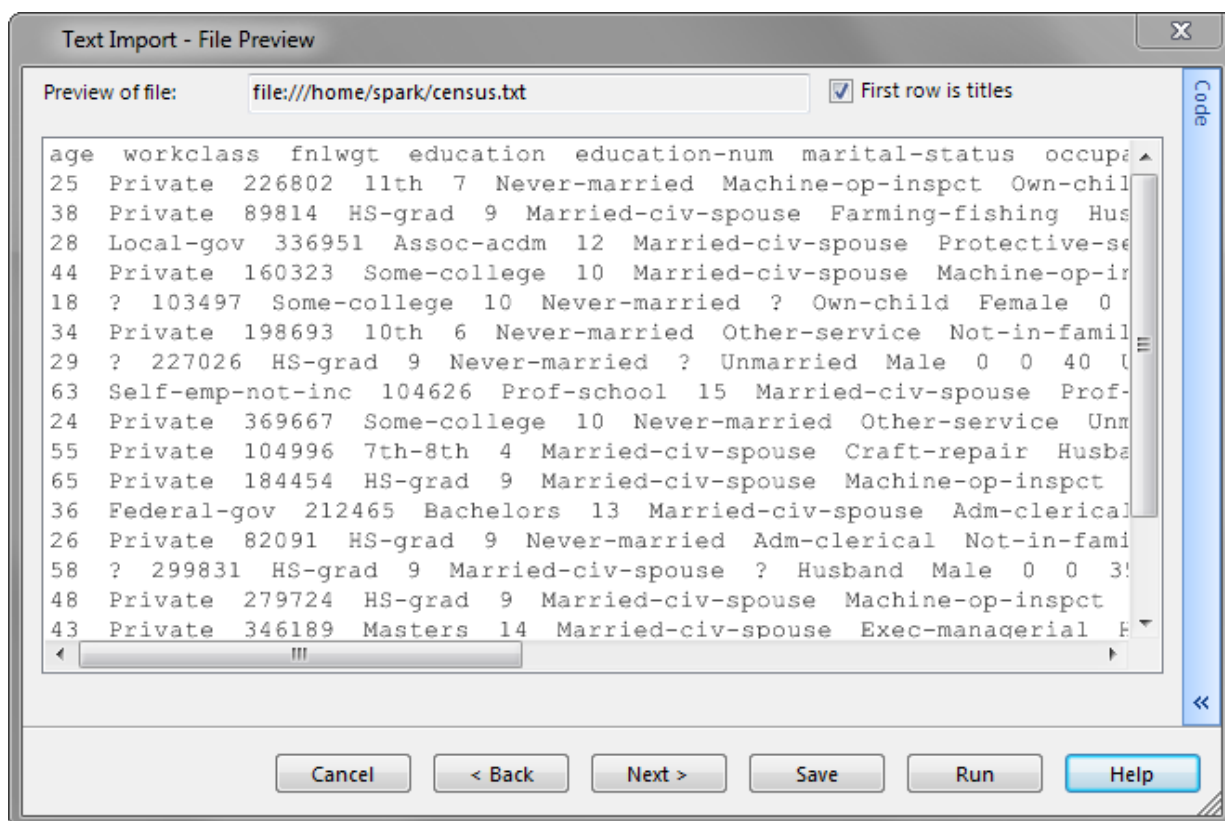
You can preview the corresponding Python code at any time by clicking the arrow at the bottom of the Code bar in the wizard - the blue vertical bar on the right side of the window. Click the arrow

again to close the preview. Note: In the code preview window you can easily increase the font size for better visibility using the mouse scroll wheel while holding down the CTRL key.

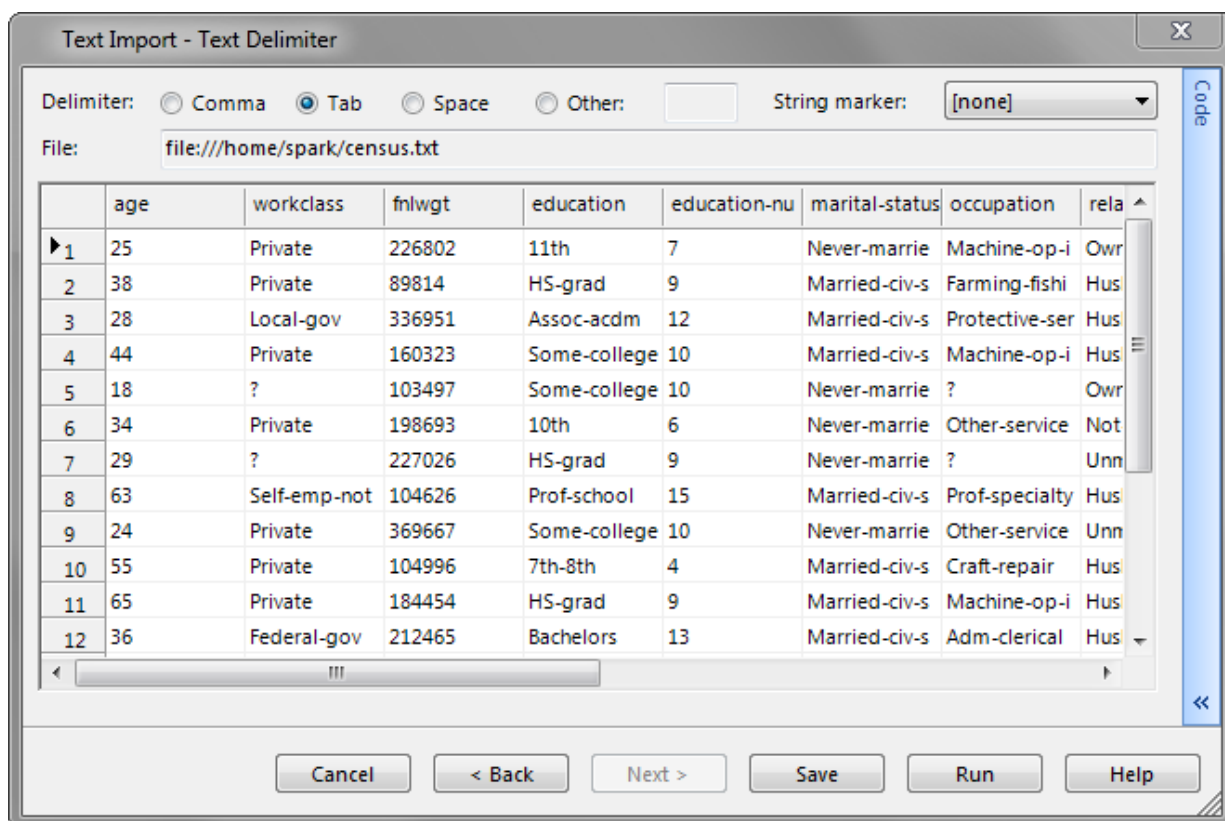


Click Next to view a sample of the data in the source file. The Preview window will open.

Check the box "**First row is titles**" if the first row of your file contains the column titles.



Both **Next** and **Run** buttons are enabled at this stage. You can click Run to start the import with the default delimiter and string marker settings, or click Next to continue specifying these options.



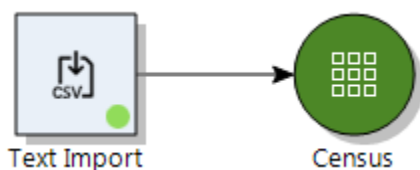
At this step, the wizard makes the best guess on the column delimiter in your source text file. If the auto-detected delimiter is correct, you should see your data separated into columns appropriately. Otherwise, specify the correct delimiter. Choose the String marker (none, double quote, or single quote).

The data types are inferred and assigned automatically.

Note: With Apache Spark v3.1.1, Date fields may be imported as String type. Please check the field types in the imported dataframe. If necessary, use the [Modify Schema](#) node to change the data types as desired.

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately.

The output dataset (dataframe) is created in the workflow after running the node. Double-click the resulting dataset node to open it.



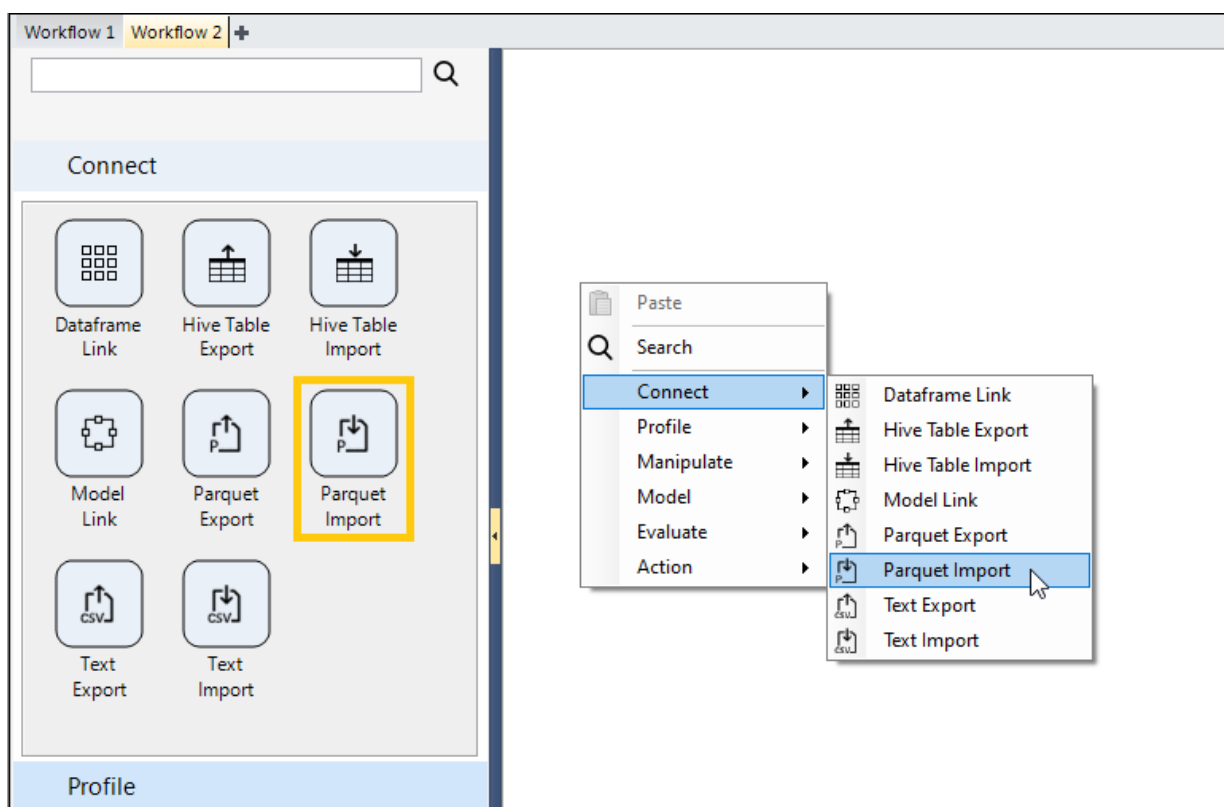


## Parquet Import

To import data from a Parquet file in your Spark storage into a Spark dataframe in your project workflow:

Page | 93

- Right-click anywhere on the Workflow canvas background. From the context menu, select **Connect --> Parquet Import** as in the example screenshot below. A new **Parquet Import** node will be created and the wizard will open. (Note: in this example, the **Spark** filter has been applied for quick search and restricting the available choices to Spark-related functions only).



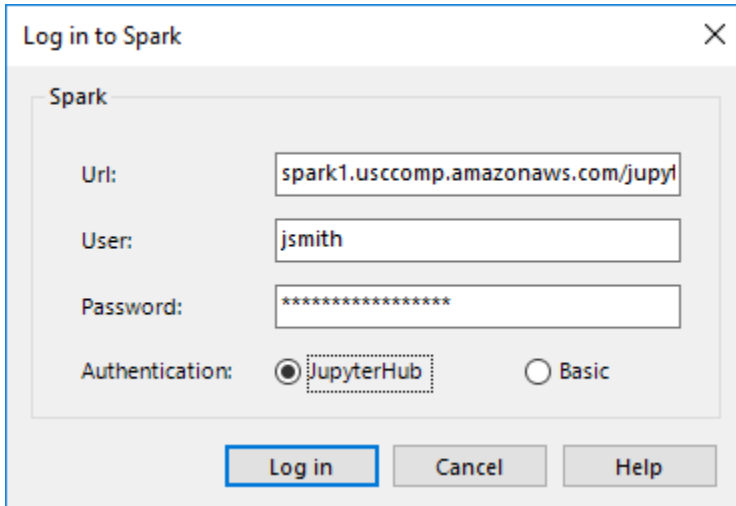
OR:

- Drag the **Parquet Import** node from the **Connect** palette onto the canvas.



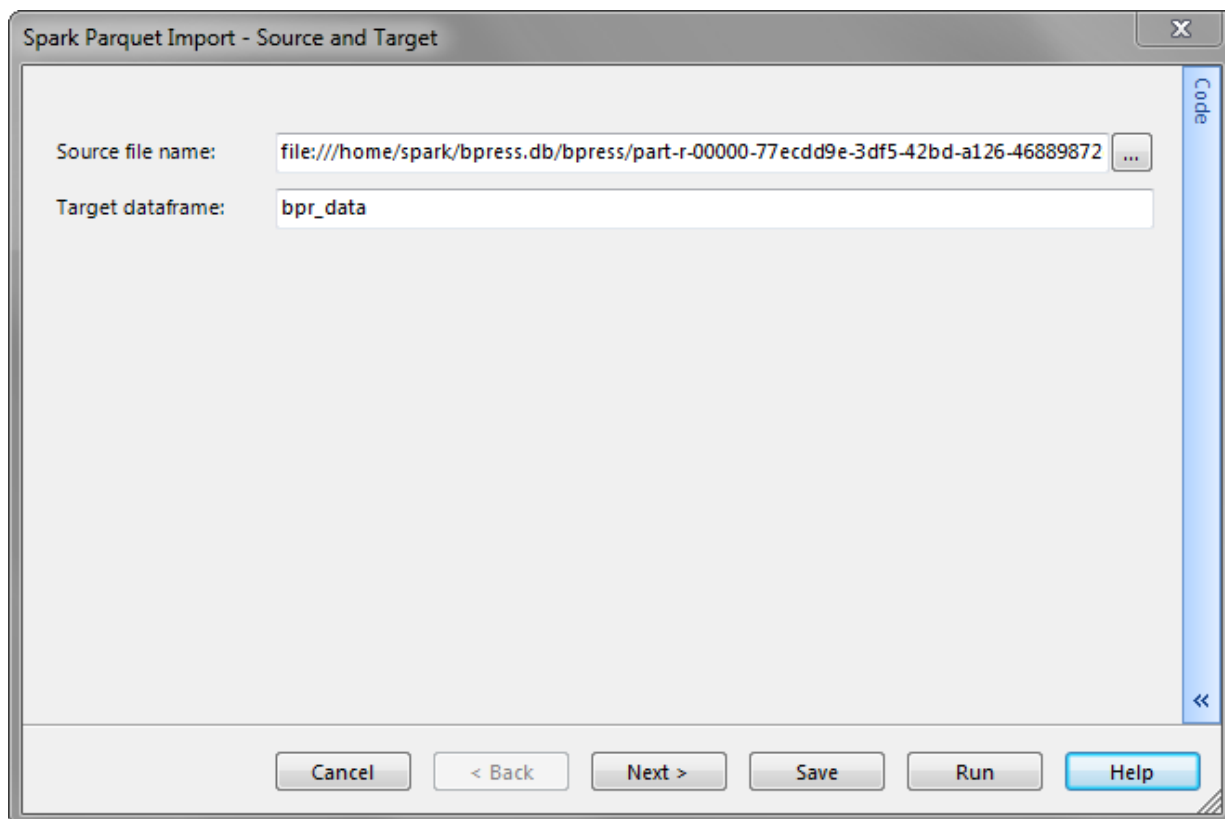
Double-click the **Parquet Import** node or right-click on it and select **Modify** to open the wizard.

At this point, if you are not already connected to the Spark server, you will be prompted to connect. Specify the URL, user credentials, and the authentication type (**JupyterHub** or **Basic**).

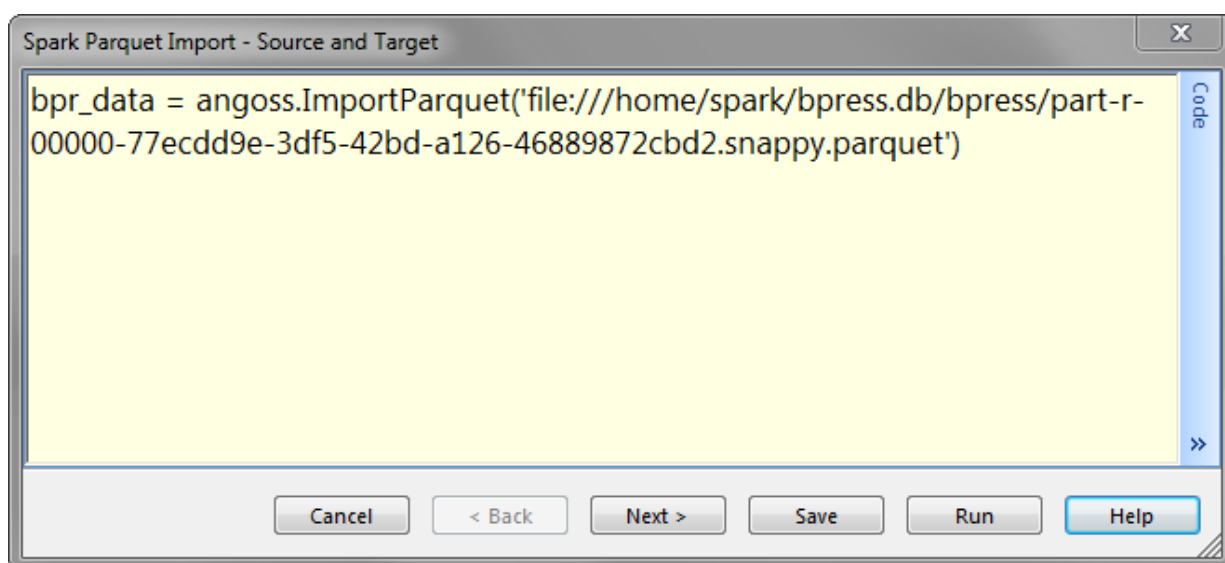
A screenshot of a 'Log in to Spark' dialog box. The dialog has a title bar with a close button (X). Inside, there's a section titled 'Spark' containing four input fields: 'Url:' with the value 'spark1.uscomp.amazonaws.com/jupyter', 'User:' with the value 'jsmith', 'Password:' with a masked password '\*\*\*\*\*', and 'Authentication:' with two radio buttons, 'JupyterHub' (selected) and 'Basic'. At the bottom, there are three buttons: 'Log in' (highlighted with a blue border), 'Cancel', and 'Help'.

Once connected to the Spark server:

- Browse the server to select the source file to import.
- Specify the name for the new dataset in the *Target dataframe* field



You can preview the corresponding Python code at any time by clicking the arrow at the bottom of the Code bar in the wizard - the blue vertical bar on the right side of the window. Click the arrow again to close the preview. Note: In the code preview window you can easily increase the font size for better visibility using the mouse scroll wheel while holding down the CTRL key.



Click Next to view a sample of the data in the source file. The Table Preview window will open.

Spark Parquet Import - Table Preview

Name: `/home/spark/bpress.db/bpress/part-r-00000-77ecdd9e-3df5-42bd-a126-46889872cbd2.snappy.parquet`

|    | yearseducatio | income | gender | weight | height | zscore | hypertension |
|----|---------------|--------|--------|--------|--------|--------|--------------|
| 1  | 2.0           | 16.0   | Male   | 141.0  | 626.0  | 4.53   | Normal       |
| 2  | 2.0           | 9.0    | Male   | 187.0  | 635.0  | 5.03   | High         |
| 3  | 2.0           | 11.0   | Male   | 191.0  | 707.0  | 3.84   | Normal       |
| 4  | 2.0           | 3.0    | Male   | 160.0  | 657.0  | 4.43   | Normal       |
| 5  | 2.0           | 15.0   | Male   | 216.0  | 709.0  | 4.43   | Normal       |
| 6  | 2.0           | 12.0   | Male   | 206.0  | 720.0  | 2.76   | Low          |
| 7  | 2.0           | 9.0    | Female | 215.0  | 632.0  | 4.9    | High         |
| 8  | 5.0           | 10.0   | Female | 89.0   | 641.0  | 4.51   | Normal       |
| 9  | 2.0           | 11.0   | Male   | 188.0  | 679.0  | 3.75   | Normal       |
| 10 | 2.0           | 16.0   | Female | 284.0  | 677.0  | 3.68   | Normal       |
| 11 | 5.0           | 10.0   | Female | 150.0  | 626.0  | 4.37   | Normal       |
| 12 | 1.0           | 13.0   | Male   | 180.0  | 649.0  | 4.15   | Normal       |
| 13 | 2.0           | 7.0    | Female | 167.0  | 619.0  | 4.32   | Normal       |

Buttons: Cancel < Back Next > Save Run Help

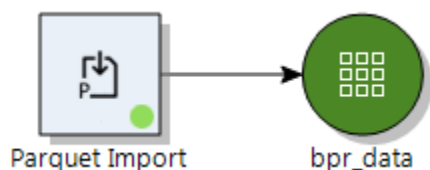
Page | 96

The data types are inferred and assigned automatically.

Note: With Apache Spark v3.1.1, Date fields may be imported as String type. Please check the field types in the imported dataframe. If necessary, use the [Modify Schema](#) node to change the data types as desired.

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately.

The output dataset is created in the workflow after running the node. Double-click the resulting dataset node to open it.

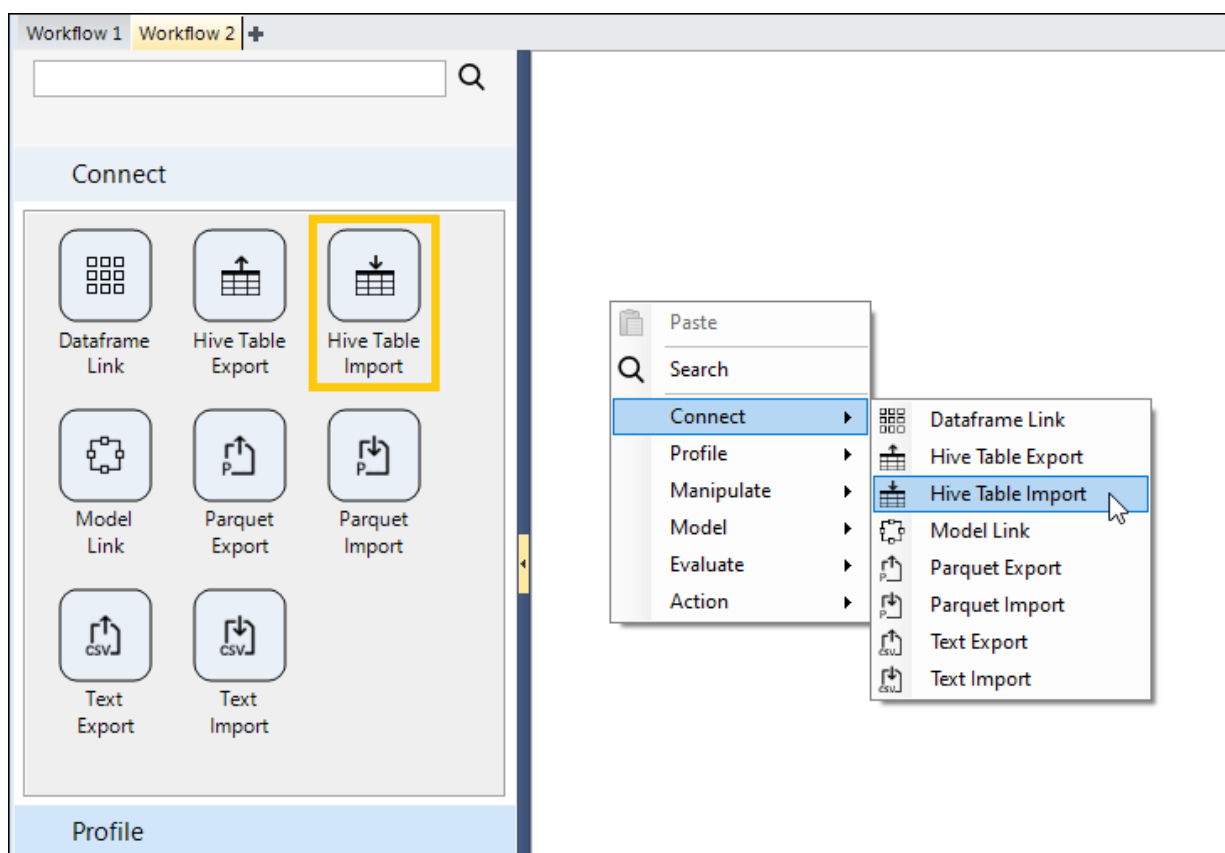


## Hive Table Import

To import data from a Hive table in your Spark storage into a Spark dataframe in your project workflow:

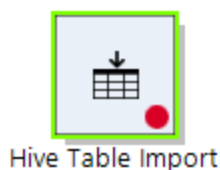
- Right-click anywhere on the Workflow canvas background. From the context menu, select **Connect --> Hive Table Import** as in the example screenshot below. A new **Hive Table Import** node will be created and the wizard will open. (Note: in this example, the **Spark** filter has been applied for quick search and restricting the available choices to Spark-related functions only).

Page | 97



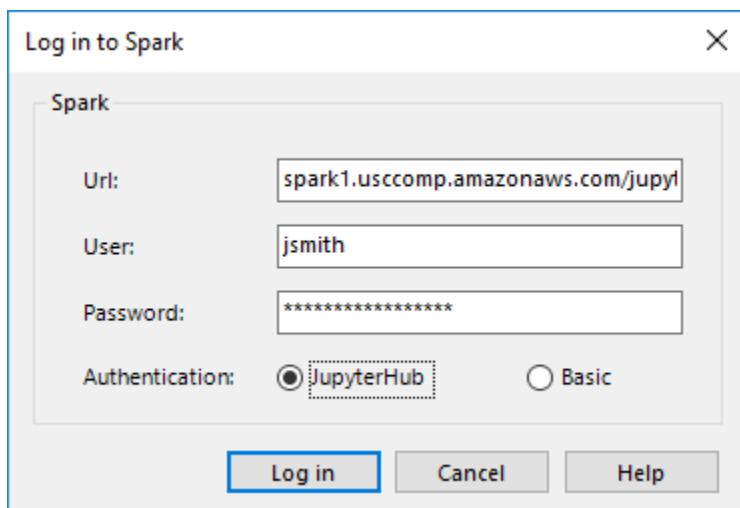
OR:

- Drag the **Hive Table Import** node from the **Connect** palette onto the canvas.



Double-click the **Hive Table Import** node or right-click on it and select **Modify** to open the wizard.

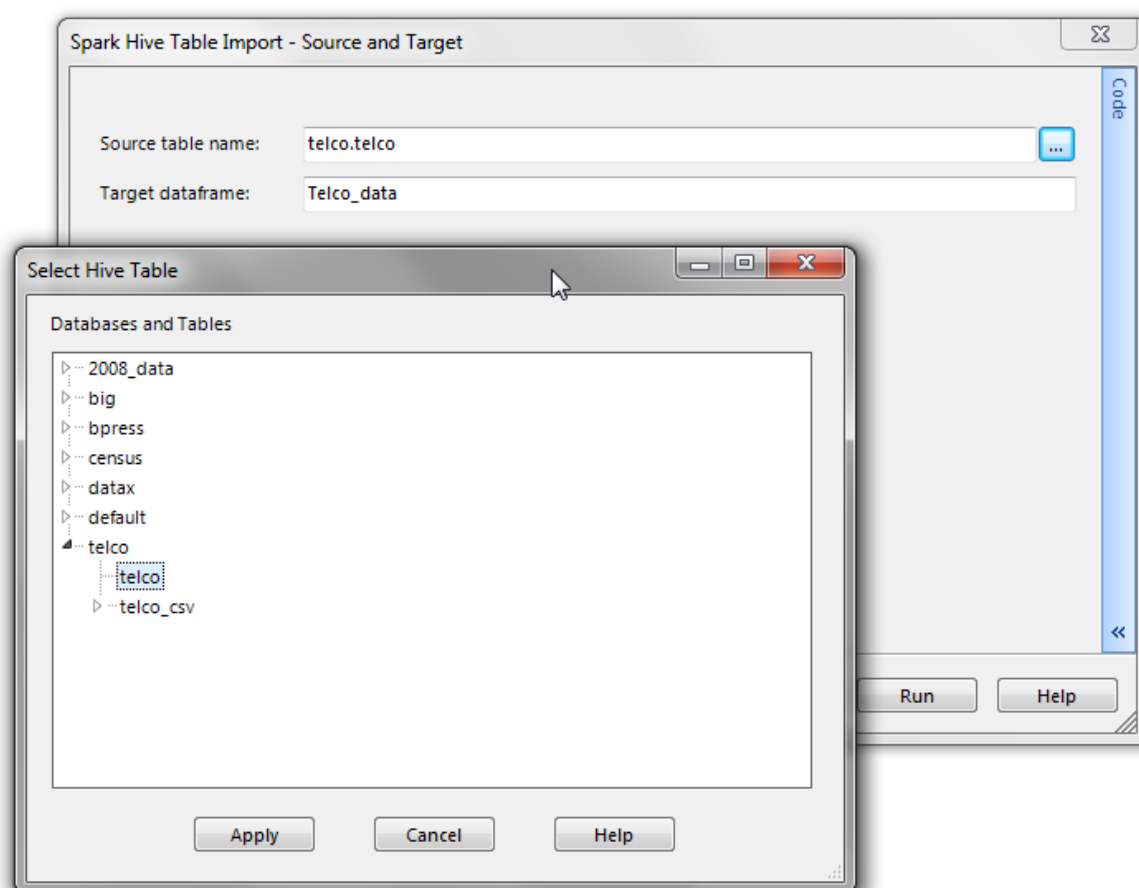
At this point, if you are not already connected to the Spark server, you will be prompted to connect. Specify the URL, user credentials, and the authentication type (**JupyterHub** or **Basic**).



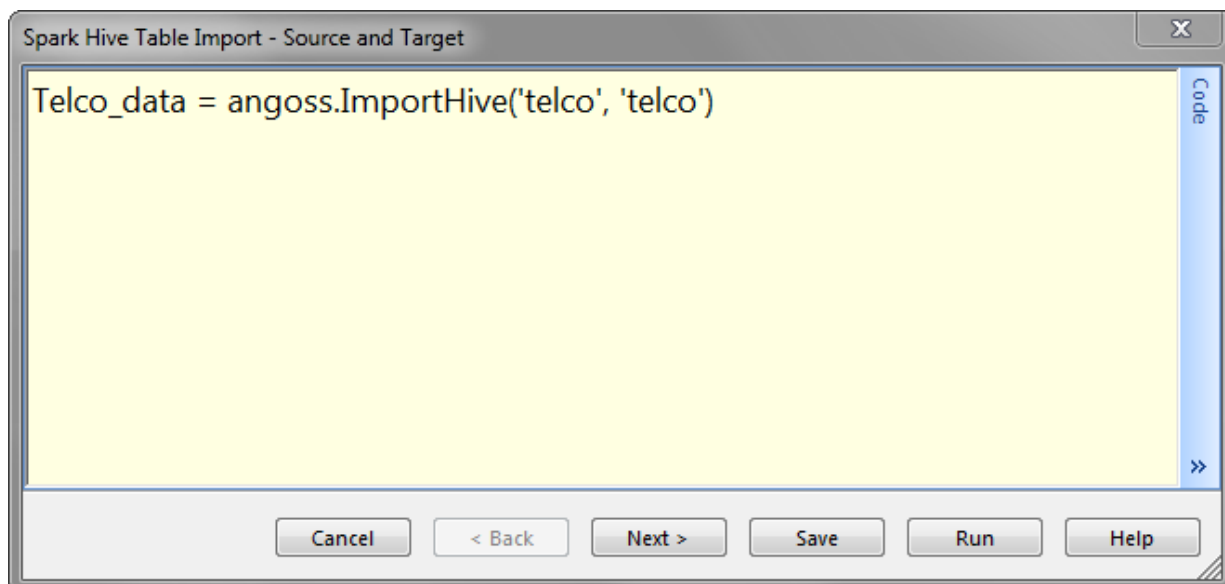
The image shows a 'Log in to Spark' dialog box. It has a title bar with a close button. Inside, there's a section titled 'Spark' containing four input fields: 'Url:' with the value 'spark1.usccomp.amazonaws.com/jupyter', 'User:' with the value 'jsmith', 'Password:' with a masked value '\*\*\*\*\*', and 'Authentication:' with two radio buttons, 'JupyterHub' (selected) and 'Basic'. At the bottom, there are three buttons: 'Log in', 'Cancel', and 'Help'.

Once connected to the Spark server:

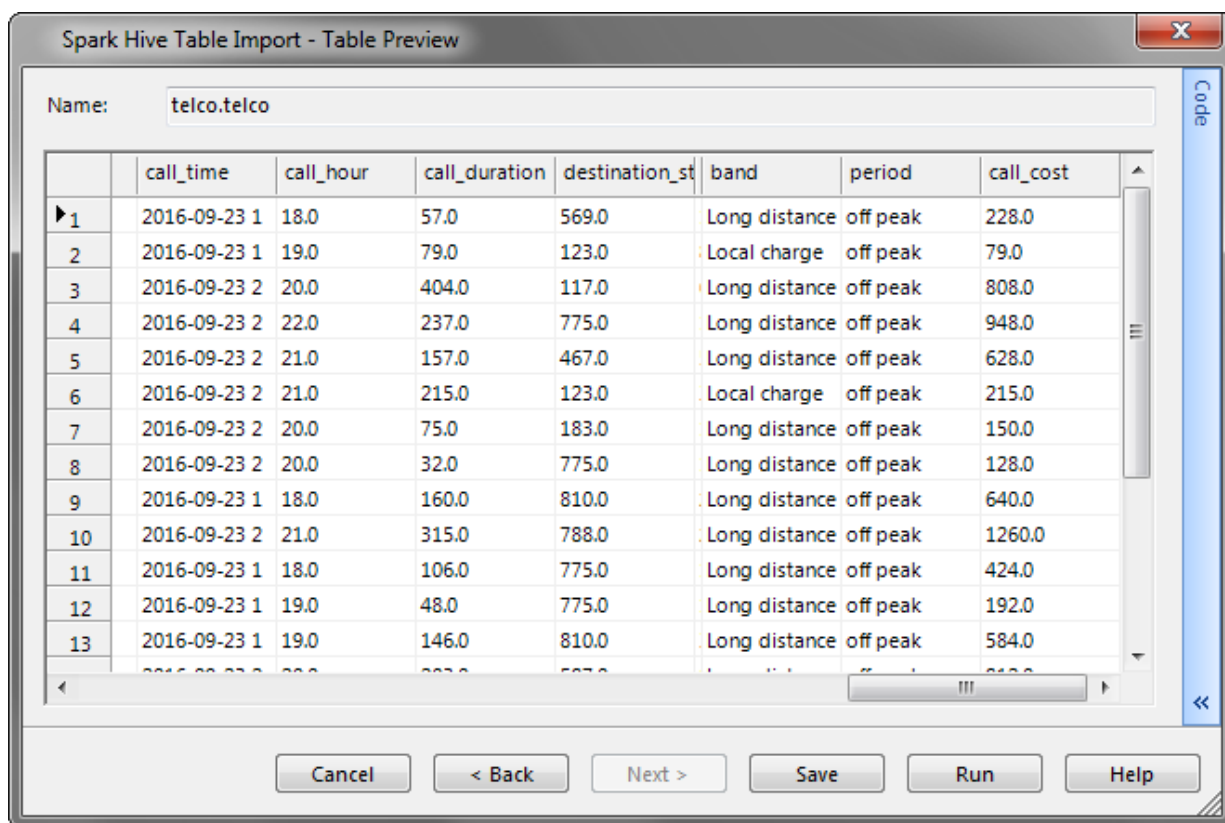
- Browse for the source table in the Hive storage and select it, or simply enter the name of the table to import
- Specify the name for the new dataset in the *Target dataframe* field



You can preview the corresponding Python code at any time by clicking the arrow at the bottom of the Code bar in the wizard - the blue vertical bar on the right side of the window. Click the arrow again to close the preview. Note: In the code preview window you can easily increase the font size for better visibility using the mouse scroll wheel while holding down the CTRL key.



Click Next to view a sample of the data in the source file. The Table Preview window will open.



The data types are inferred and assigned automatically.

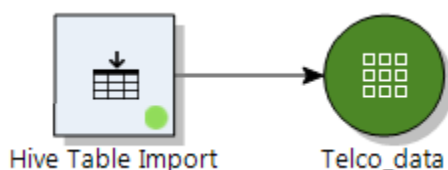


Note: With Apache Spark v3.1.1, Date fields may be imported as String type. Please check the field types in the imported dataframe. If necessary, use the [Modify Schema](#) node to change the data types as desired.

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately.

Page | 101

The output dataset is created in the workflow after running the node. Double-click the resulting dataset node to open it.



## Data Export

Data from Spark dataframes can be exported to files or tables in the Spark storage system.

The following destination formats are supported:

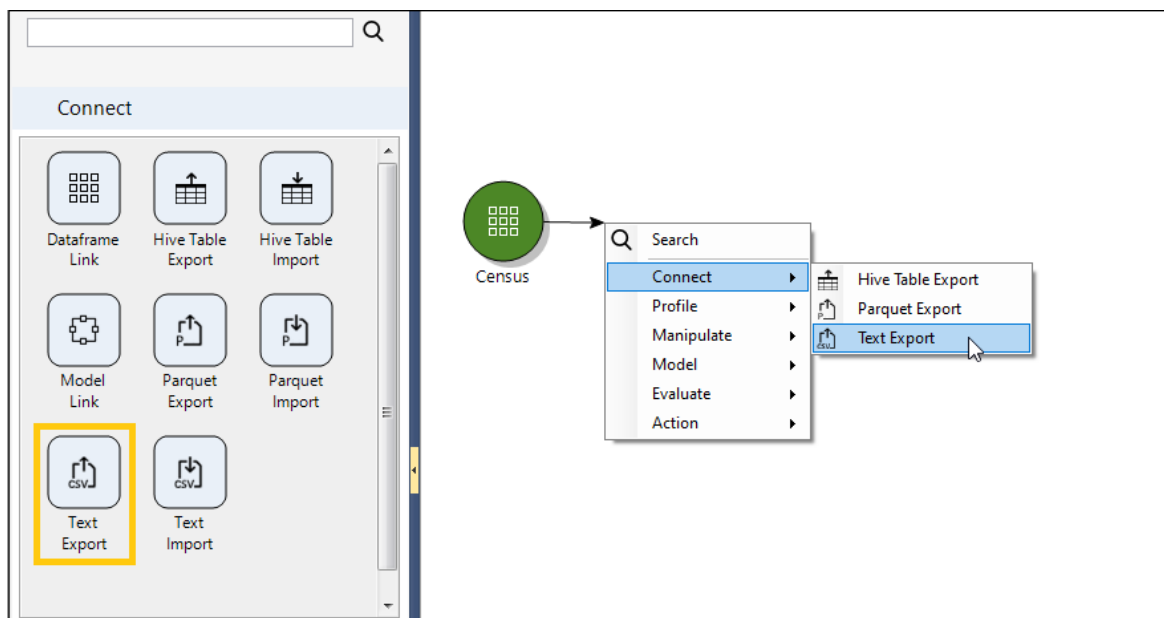
- Text
- Hive Table
- Parquet

To export data from a Spark dataframe:

- Drag a connection arrow from the dataframe node to any area on the Workflow canvas. When you release the mouse button, a context menu will appear. From the context menu, select **Connect → <Format> Export** (where <Format> is your desired output format). See the example screenshot below. A new **Export** process node will be created.

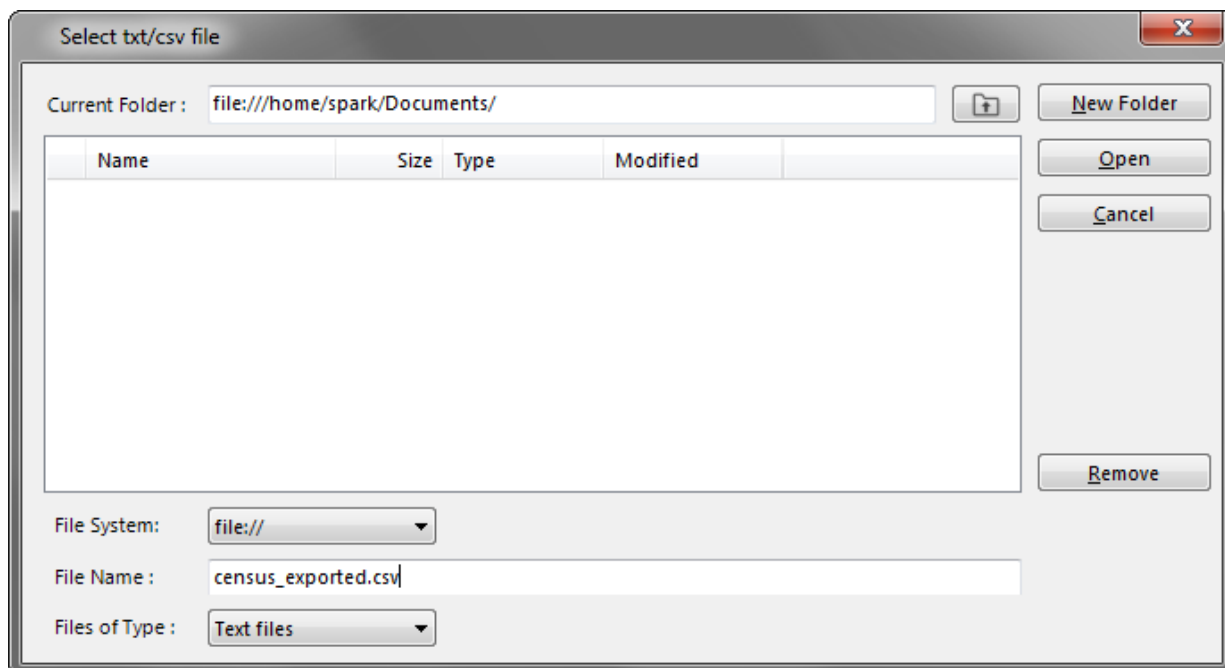
OR:

- Drag the **<Format> Export** node from the **Source** palette onto the canvas, where <Format> is the desired output format, and connect the input dataframe to the Export node. Then double-click on the new node or right-click and select **Modify**.

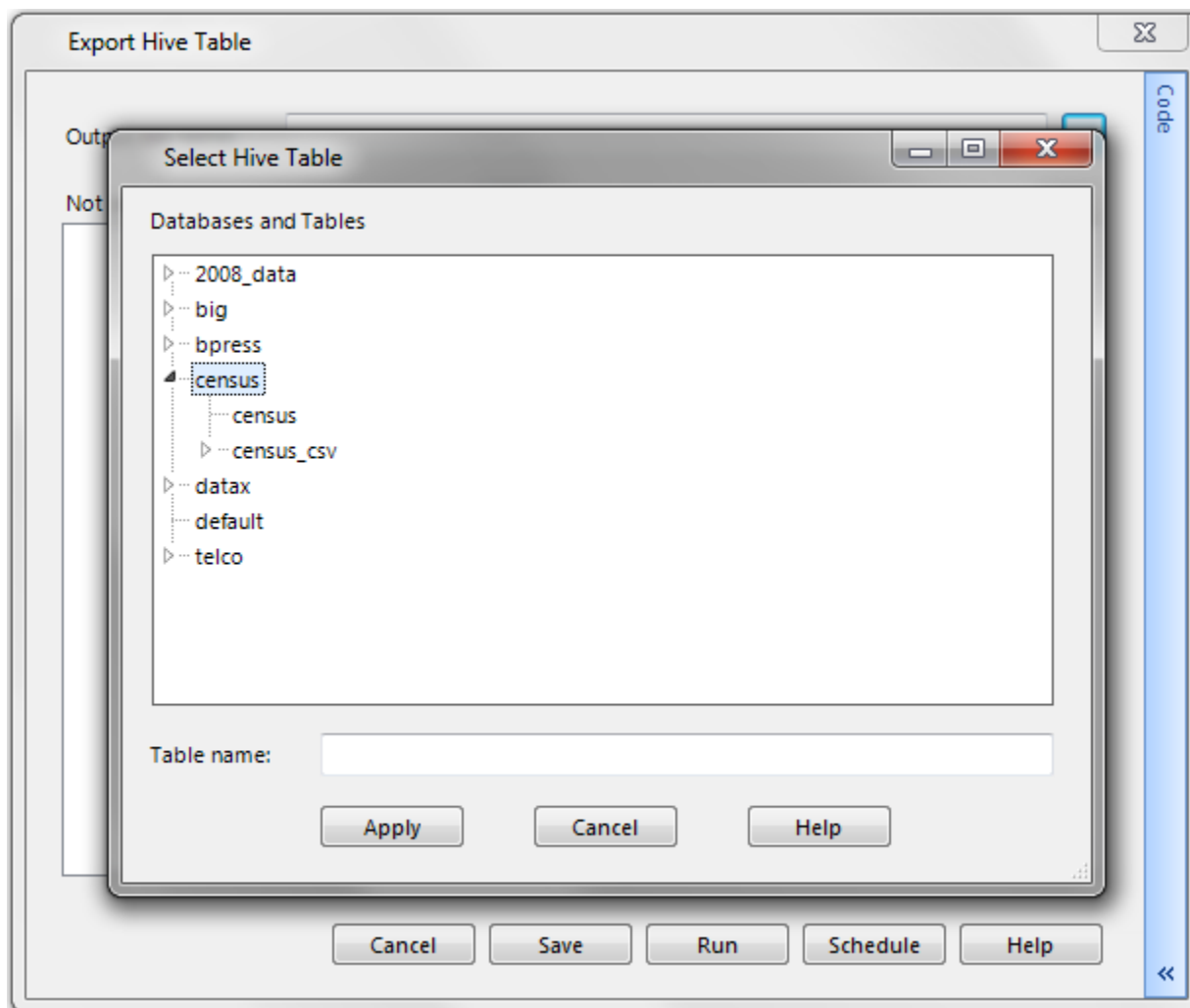


For each export format,

- browse the Spark storage for the destination folder
- select the storage type (file system type) from the "Files of Type" drop-down list at the bottom
- specify the name of the new file to be created in the File Name field at the bottom of the dialog



When exporting to a Hive table, browse for the desired database.

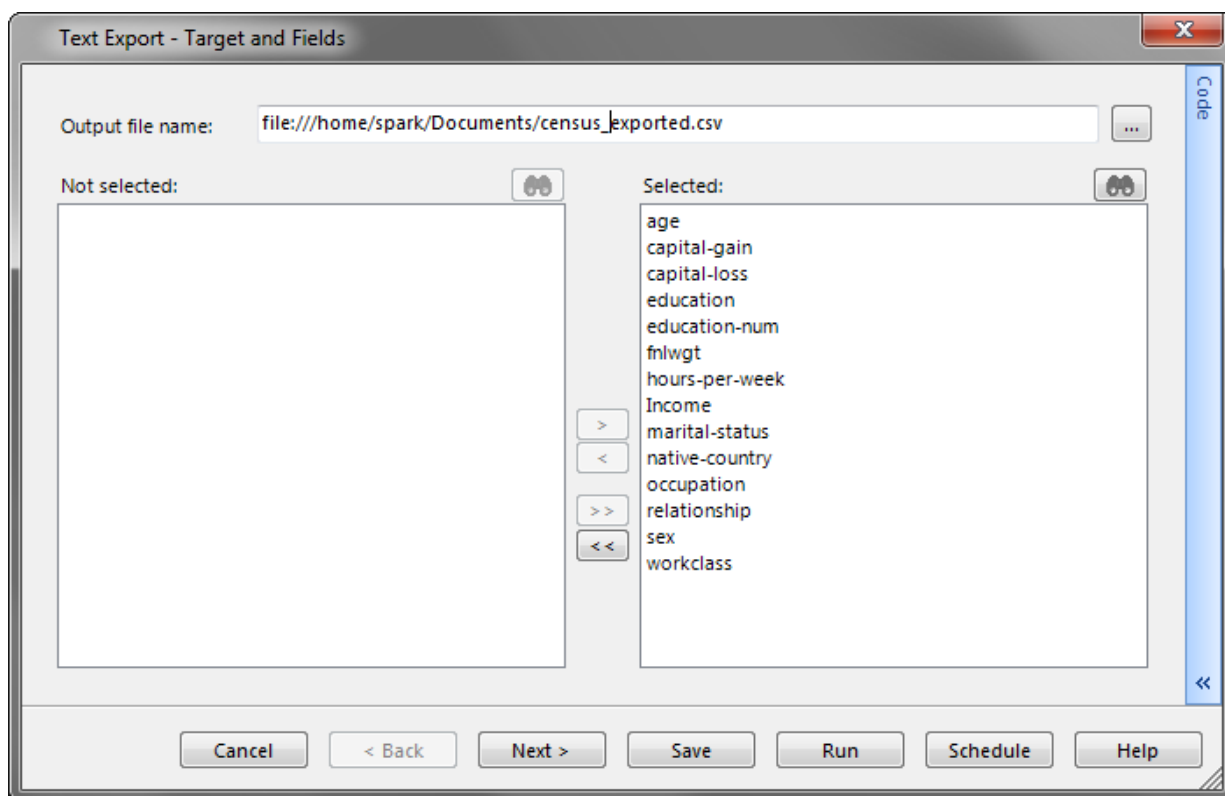


Once you have specified the destination folder, select the source fields you want to include in the exported file. By default, all fields are selected for exporting.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

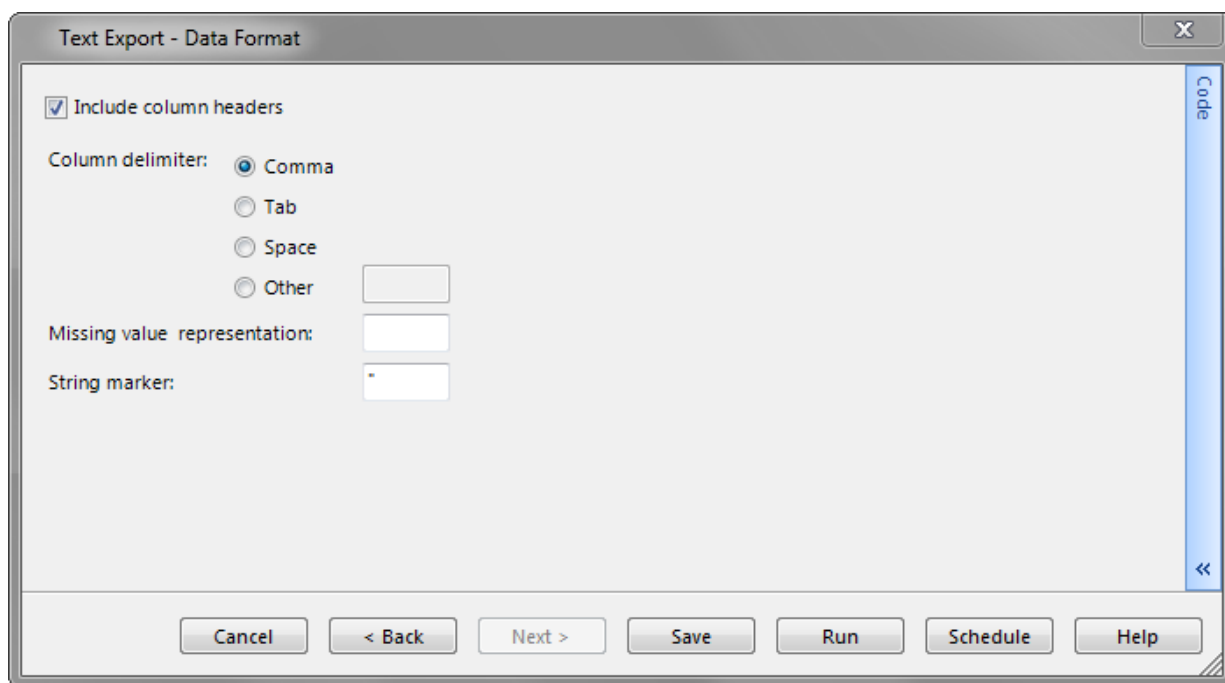
The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.



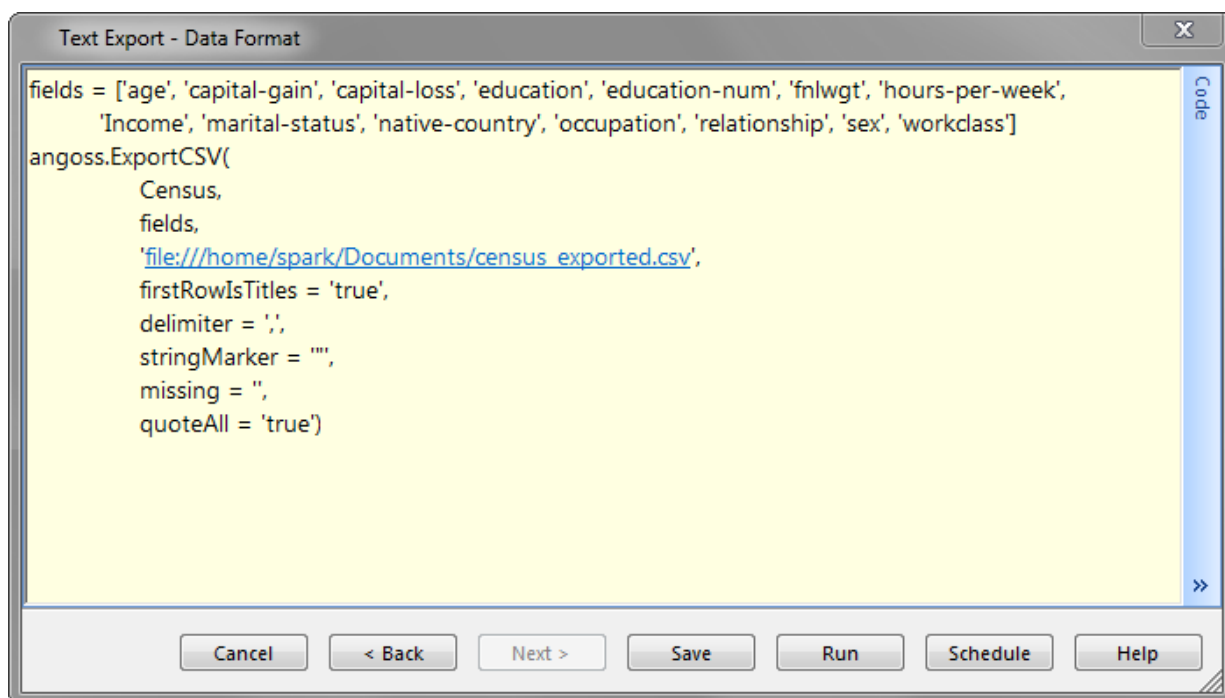
Exporting to all supported formats is straightforward. You only need to specify the output file name and the fields to be included.

When exporting to Text, in addition, you need to specify the following options:

- option to include column headers
- column delimiter
- missing value representation
- string marker (if necessary)



You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the Code bar - the blue vertical bar on the right side of the window. The code can be copied, but not edited. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

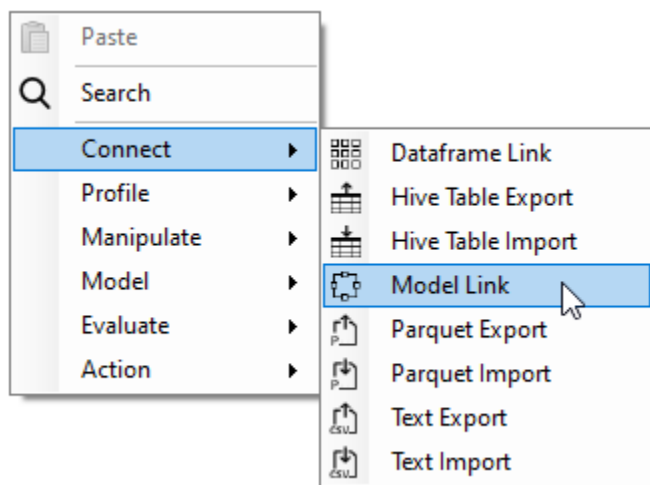


## Spark Model Link

Any Spark model available in memory during your current Spark session, whether it is present on your workflow canvas or not, can be put on the workflow canvas using the **Model Link** node in the **Connect** palette.

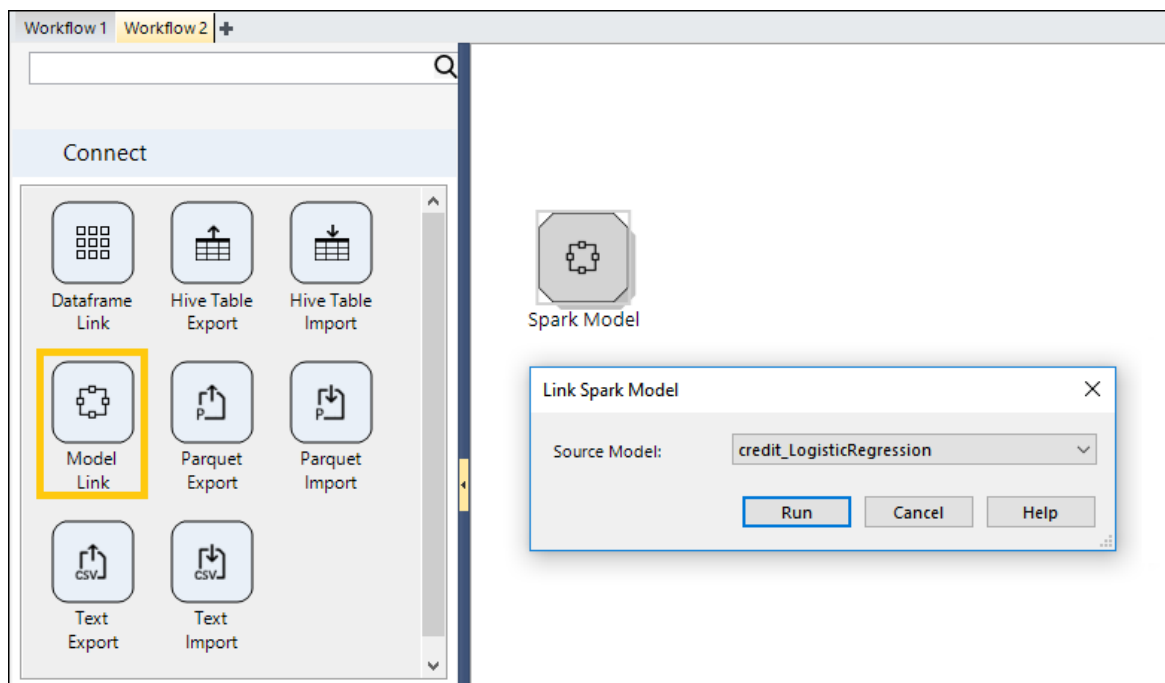
To link an existing Spark model to a new model node on the workflow canvas:

- Right-click anywhere on the Workflow canvas background. From the context menu, select **Connect --> Model Link** as shown in the example below. A new **Spark Model** node will be created and the Link Spark Model dialog will open.

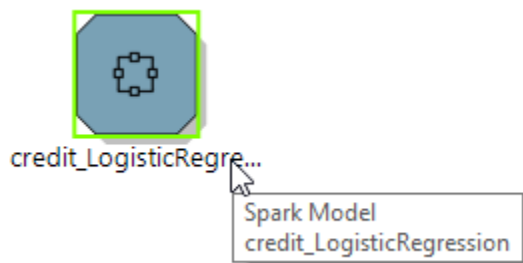


OR:

- Drag the Spark **Model Link** node from the **Connect** palette onto the canvas. Then double-click the new Spark Model node or right-click on it and select **Link** from the context menu.



Select the desired model from the Source Model list of the Link Spark Model dialog and click **Run**. The Spark Model node is now linked to the selected Spark model. Double-click on it to view the model.



The model node created this way is a model instance, which can be viewed and scored or validated. If it is a decision tree or a strategy tree, it can be modified in the interactive tree view, like any tree instance. If it is a regression model or any model other than a tree, it cannot be retrained, since it is not a process node.

Any model node placed on the canvas using the Model Link node can be re-linked by double-clicking on the node and selecting a different model.

## Managing Spark Objects

Spark objects created in the cluster memory by Knowledge Studio for Apache Spark (analytic dataframes, models, etc.) can be **saved** in the Spark storage, **purged**, and **restored**.

Page | 108

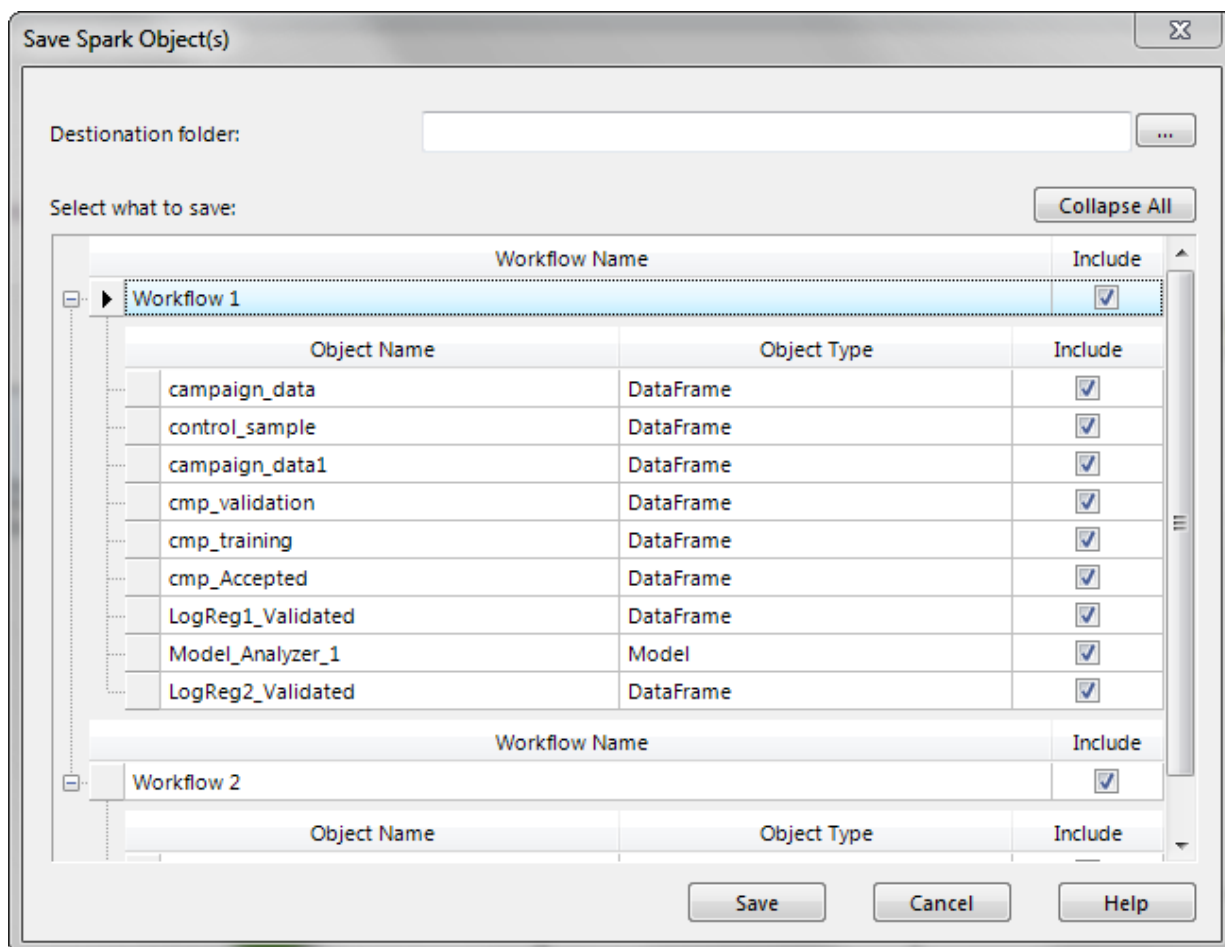
### Saving Spark Objects

To save a Spark object (dataframe, model) in the Spark storage system, click on the Save Spark

Objects icon in the toolbar:

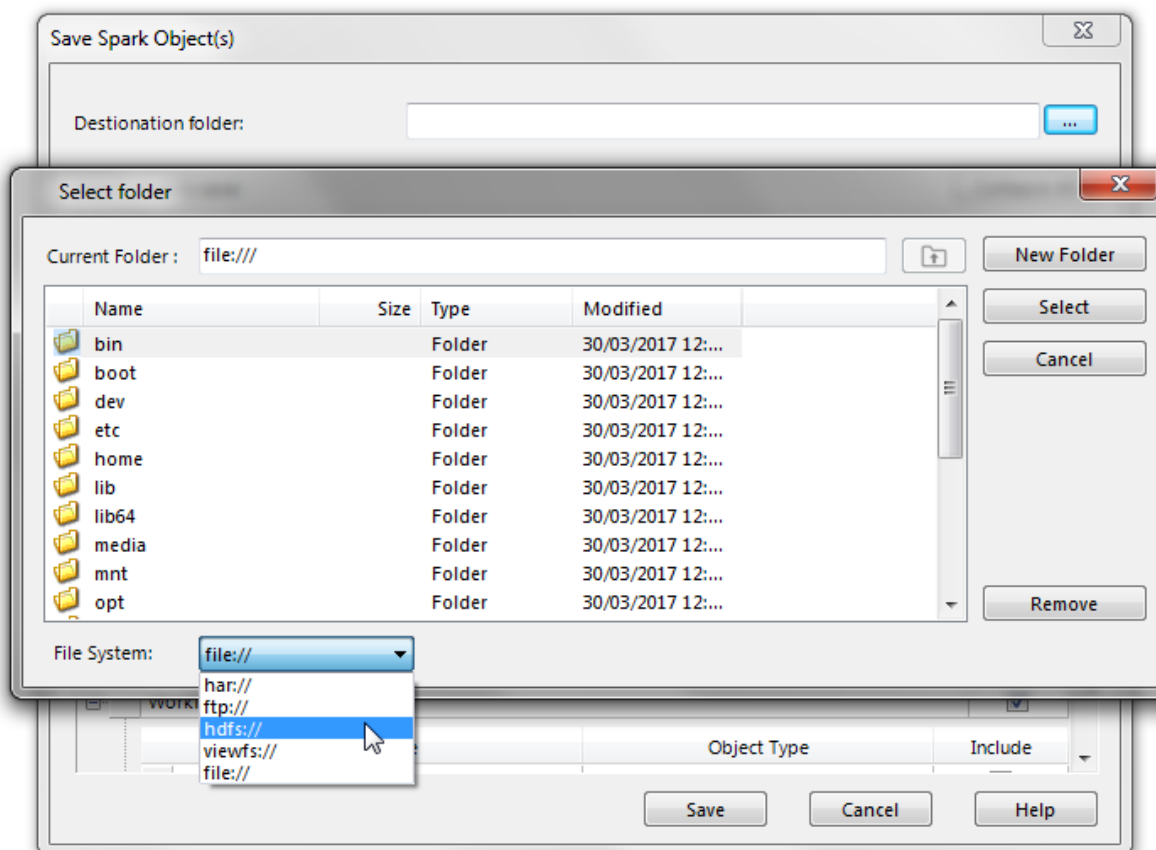


The Save Spark Object(s) dialog opens. It shows the names and types of objects in Spark memory for all of your current workflows. The list is grouped by workflows.





Specify the Destination folder in the Spark storage. Click on the browse button at the top right to browse your Spark storage system. In the Select Folder dialog, choose the desired File System from the drop-down list at the bottom, as shown below. If available, you can choose HDFS, Hadoop archive, Hadoop ViewFS, FTP, and other systems.



Double-click on the desired folder and click **Select**. Or click **New Folder**, specify the name for the new folder and click **Select**. Then click **Save**. The objects will be saved in the specified folder and you will be able to restore them later.

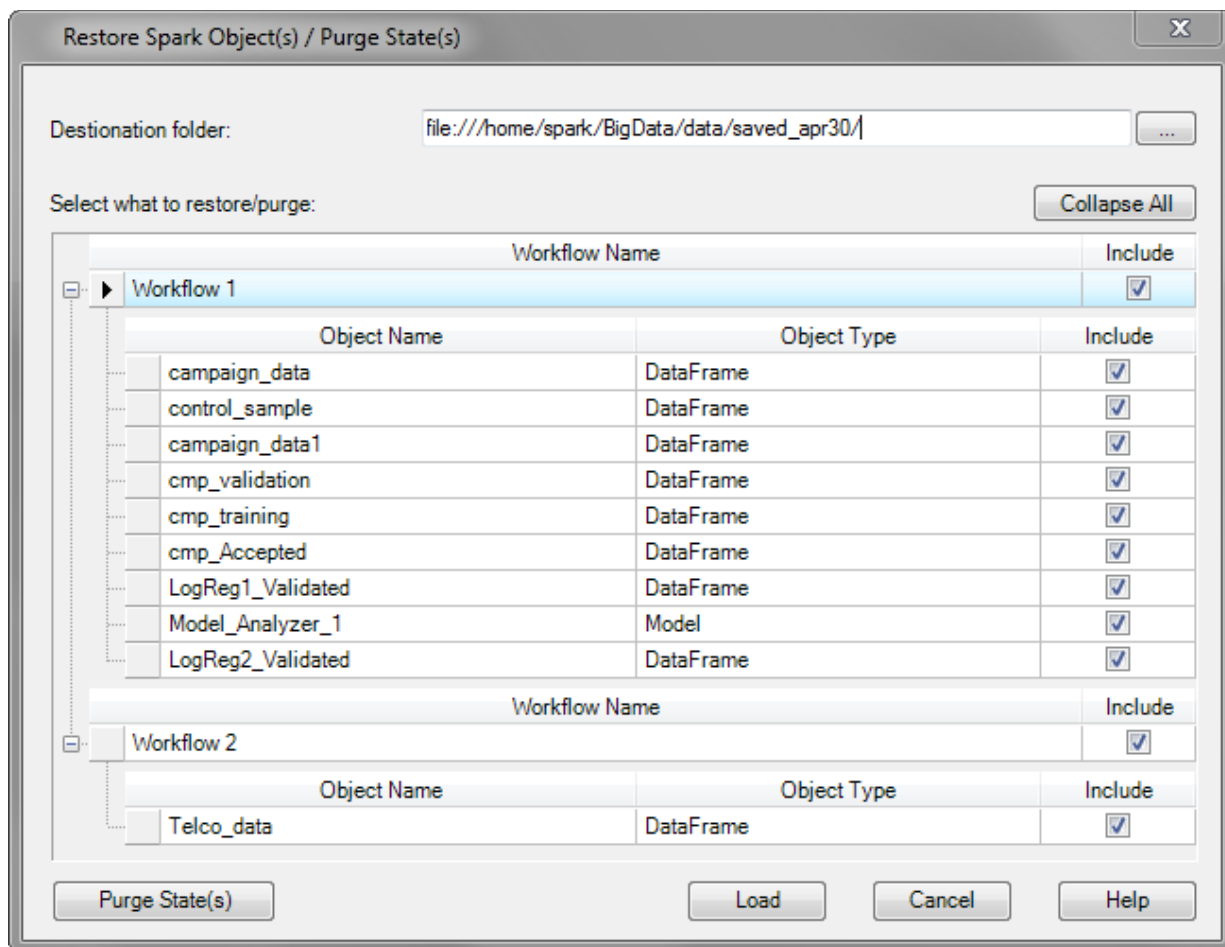
If you want to delete an existing folder from the storage, select it and click **Remove**.

## Restoring Spark Objects

To restore Spark objects from storage, click on the Restore/Purge Spark Objects button in the

toolbar: 

This brings up the dialog Restore Spark Object(s) / Purge State(s). Browse for the folder with previously saved objects, then click **Load**.



To purge all current Spark session objects from memory, click the **Purge State(s)** button. All objects in the memory of the current Spark session will be destroyed.

**Note:** A restored Spark dataframe or model cannot be saved again, since it has already been saved. If you try to save a restored dataframe or model, you may get a message that it failed to save. If you modify a Spark model and run it again, the model can be re-saved, and so can be a modified dataframe.

## Scheduling Analytic Tasks on Spark

The execution of any node or groups of nodes in the workflows of Knowledge Studio for Apache Spark can be scheduled using the **Task Scheduler** feature available for all node types. Scheduled tasks continue to run on Spark even in unattended mode when the application is closed and the user logs out and disconnects from the session. This can be especially beneficial for complex tasks or very large data volumes that take a lot of time to process.

The task scheduler can be invoked in three ways:

- From the main menu using the **Scheduled Tasks** button  in the main menu. (In this context it only shows the list of existing scheduled tasks and does not allow to schedule new tasks)
- Using the **Schedule** button in the wizard of any functional node
- Using the **Schedule** command in the right-click context menu of any functional node on the workflow

If scheduled tasks on your computer can only be created by users with administrative privileges, you may need to start Knowledge Studio **as administrator** before using the scheduling feature.

- On Windows 10, 11, Windows Server 2016, 2019, 2022, you will also need to edit the properties of the newly created task in Windows Task Scheduler. See the section *Modifying Scheduled Task Properties* below.
- On Red Hat Enterprise Linux 8 and CentOS 8, task scheduling requires the **at** utility and the **atd** job scheduler daemon. To install and enable them, use the commands

```
sudo yum install at
sudo systemctl start atd
sudo systemctl enable atd
```

### Scheduled Tasks and Licensing

A scheduled task launches a separate instance of the application executable in the background, and therefore it will perform a separate license check when it starts.

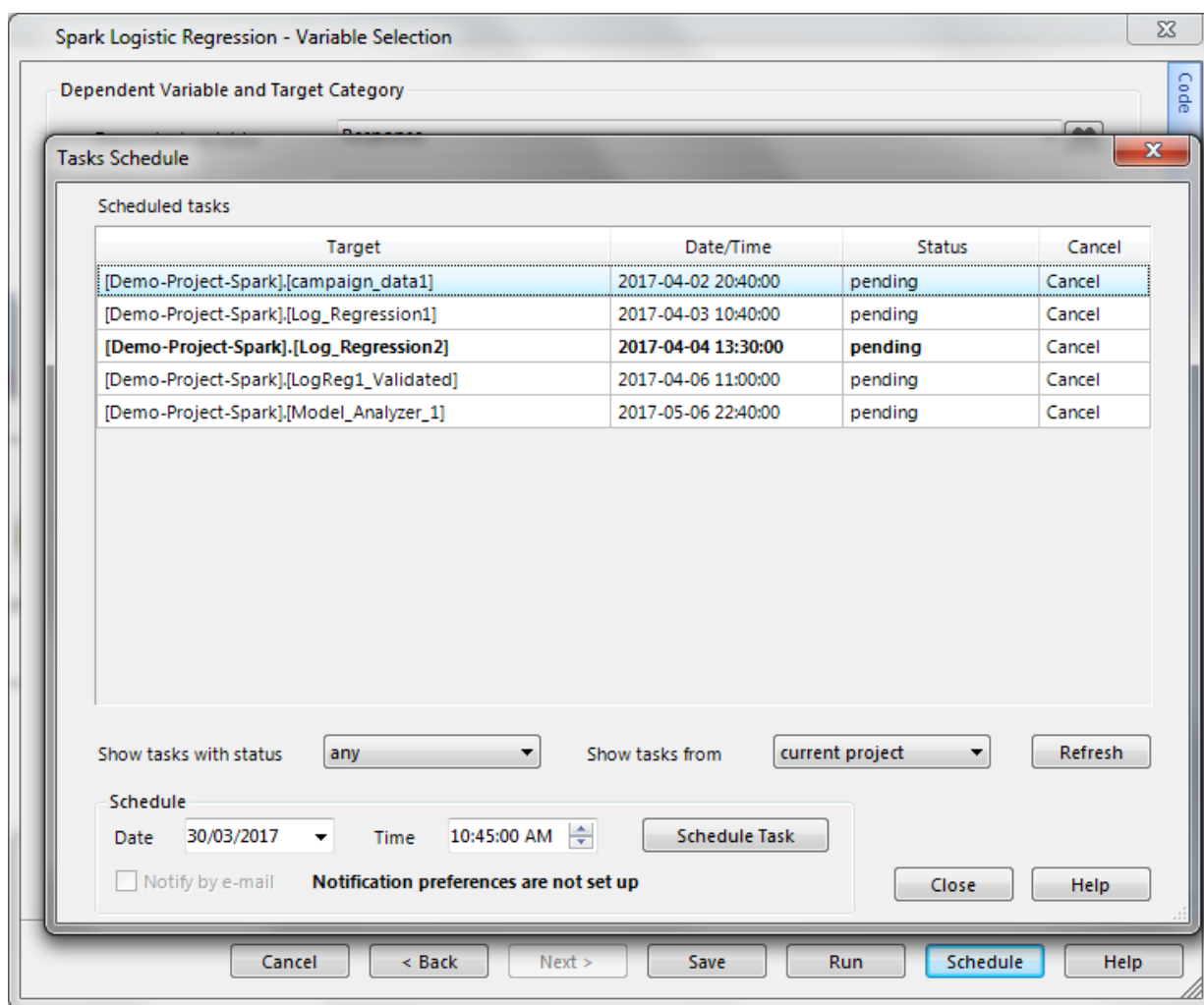
- In the case of a [named-user](#) license, if you modify the task properties in Task Scheduler or cron, make sure the task is configured to run under the account of a licensed user.
- In the case of a [concurrent](#) license, the instance launched by the scheduled task will use a new connection, so it will run only if connections are available at the scheduled start time. Since tasks are usually scheduled to run in unattended mode when the application is closed, one user ID would not normally take more than one connection. However, the concurrent connections limit should be taken into account when scheduling multiple tasks.

## Scheduled Tasks Dialog

In the example below, the scheduler is invoked by clicking the **Schedule** button in the Logistic Regression node wizard. Alternatively, right-click on the node and select the **Schedule** command from the context menu.

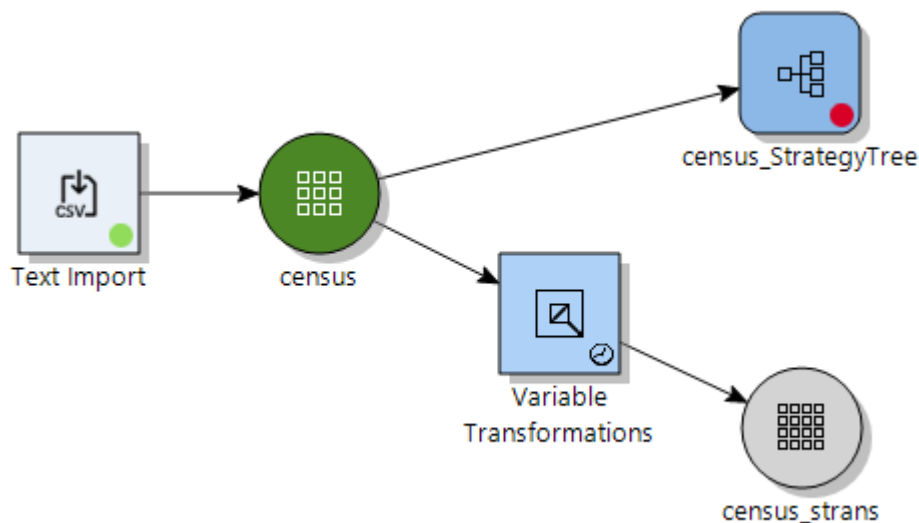
Page | 112

The **Task Schedule** window displays **all** scheduled tasks in all workflows of the current project (regardless of whether it is invoked from a particular node or the main menu).



- To schedule a new task, specify the desired date and time to run it, then click the **Schedule Task** button. A new line representing this task will appear in the list above.
- To cancel an existing task, click **Cancel** for this task in the Cancel column on the right.

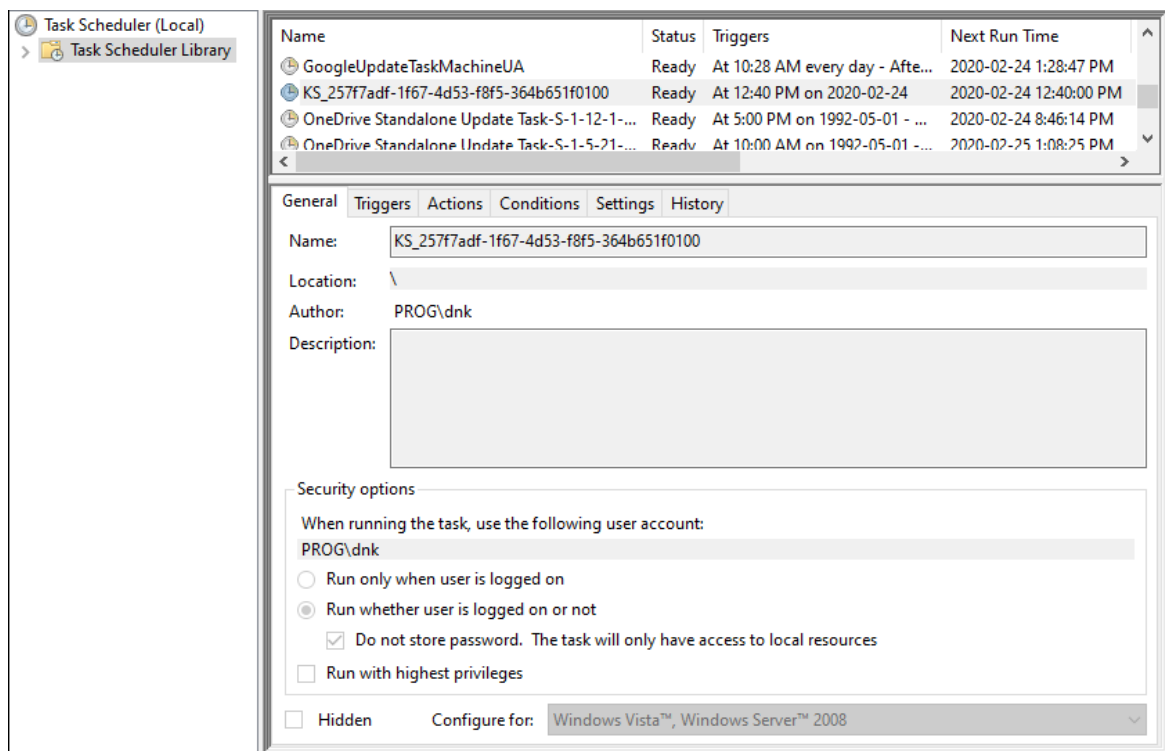
- Once the task has been scheduled, close the Tasks Schedule dialog. The node with a scheduled task will have a clock icon indicating that the associated task is scheduled to run. In the example below, the Variable Transformations node is scheduled to run.



## Modifying Scheduled Task Properties

The scheduled task properties can be modified if necessary using Windows Task Scheduler (on Windows) or *cron* (on Linux).

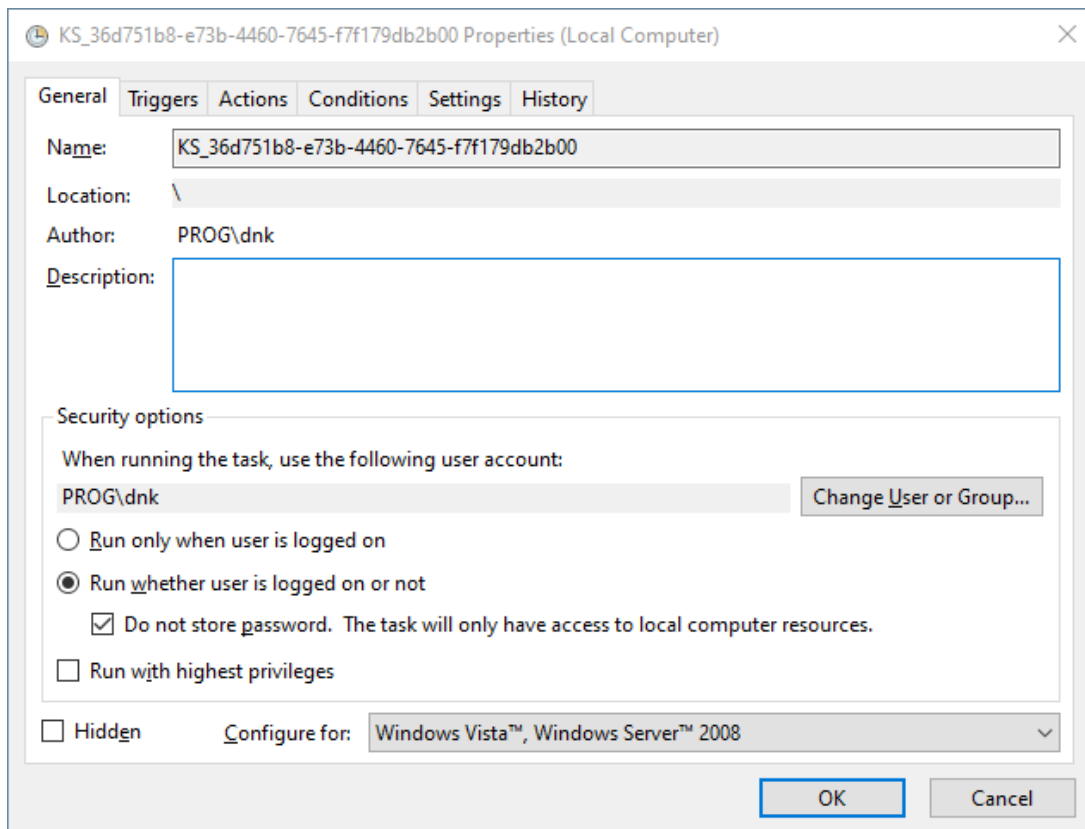
For example, to modify the task properties on Windows, open Task Scheduler from Windows Administrative Tools and select *Task Scheduler Library* in the leftmost panel. The name of the task created by Knowledge Studio starts with "KS\_". Double click on it and modify the settings in the *General*, *Triggers*, *Conditions*, and other tabs if necessary.



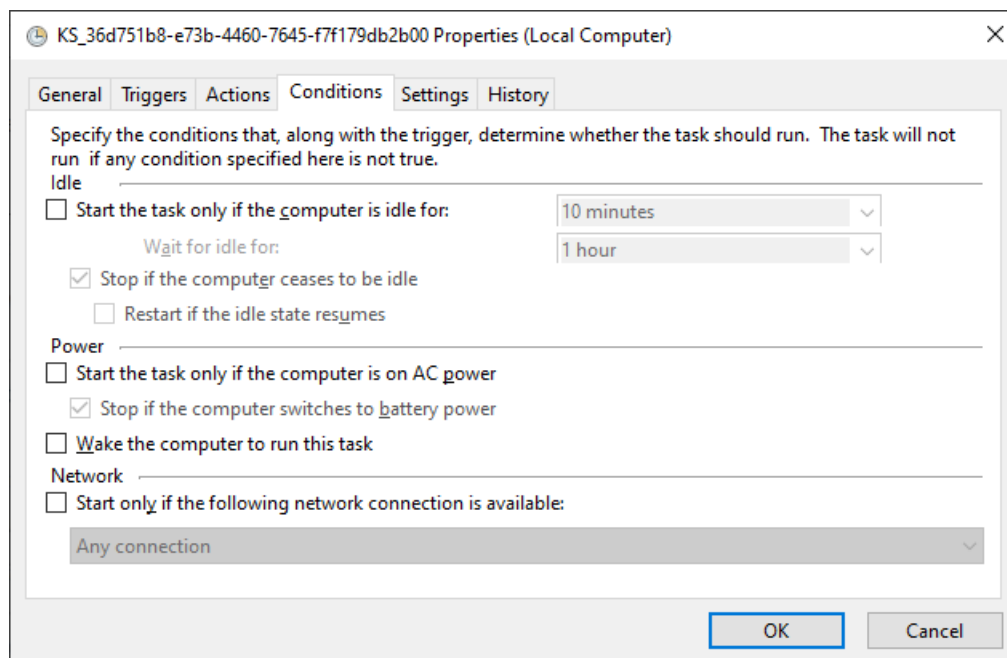
On Windows 10, 11, Windows Server 2016, 2019, 2022, the following modifications may be necessary:

- In the **General** tab:

- You may need to deselect the option "*Do not store password. The task will only have access to local resources*". When applying this change, you will need to specify the password for the account you chose for running the task.
- If necessary, click the button "*Change User or Group*" and specify a different user account for running the task. This must be an account of a licensed user. In the case of a [concurrent](#) license or [Altair Units](#) license, you can choose any user account as long as it has sufficient privileges to start the program as defined in the *Actions* tab. See the note on Scheduled Tasks and Licensing at the beginning of this topic.



- In the **Conditions** tab, specify the desired power and network availability options. Note: On laptops, the option "*Start the task only if the computer is on AC power*" is selected by default, so you may want to deselect it.



## Email Notification Options

- If you need to set up Email notifications in the scheduler to inform any user(s) of the task status changes, you need to set up Notification preferences in the application preferences under the **Tools** menu. After that, the checkbox **Notify by email** will be enabled at the bottom of the Tasks Schedule dialog, and you can select it if necessary.
- When setting the Email Notification options using **Tools | Preferences**, specify the SMTP host name or IP address, SMTP port, User ID and password (optional), and the *From/To* Email addresses. The option **Enable SSL** is selected by default, but you may need to disable it (depending on the security settings in your Email client). If you are not getting notifications after configuring these settings, deselect the Enable SSL option and try again.

The settings shown in the image below are given as an example template. Ask your IT team for assistance if you are not sure what parameters to specify.



Preferences

Tree Configuration Split Parameters Tree Training Tree Attributes Node Report Split Report Other Models  
General **Notifications** Overview Report Segment Viewer Cross Tabs Correlations Tree Colors Tree Node Tree Growth

SMTP

Host: casarray.company.com Port: 25

Credentials

User ID: jsmith Password: \*\*\*\*\*

Email

From: clusteradmin@company.com

To: deployment\_team@company.com ☒ Enable SSL

Reset  
Reset All  
OK  
Cancel  
Help

## Data Manipulation

### Aggregate

Data aggregation is used to group records on one or more attributes and replace the other attributes with summary statistics based on the resulting groups of records.

A common example is transforming transactional data from a point-of-sale system, which has one record per transaction, to a dataframe with a single record per customer, with attributes containing aggregate information about customer transactions. The aggregated data could contain attributes like the total number of transactions, total amount purchased, or average purchase size per customer.

In the example below, records with the same customer ID (loyalty card ID) need to be aggregated. For each customer, the following summary statistics needs to be calculated: the total number of purchased items, the total number of transactions, the preferred payment method (i.e., most frequent payment method).

|    | transaction_id | card_id   | payment_method | timestamp  | product_code | items_number | items_amount |
|----|----------------|-----------|----------------|------------|--------------|--------------|--------------|
| 1  | 145419         | 100000111 | Cash           | 19/05/2001 | 000000078    | 1            | 89           |
| 2  | 121302         | 100000111 | Cash           | 27/04/2001 | 000000284    | 1            | 980          |
| 3  | 51466          | 100000111 | Cash           | 20/02/2001 | 000000488    | 1            | 69           |
| 4  | 143150         | 100000111 | Cash           | 17/05/2001 | 000000595    | 1            | 299          |
| 5  | 51036          | 100000111 | Cash           | 20/02/2001 | 000000622    | 1            | 69           |
| 6  | 205287         | 100000111 | CreditCard     | 19/07/2001 | 000000808    | 1            | 3599         |
| 7  | 54688          | 101213614 | CreditCard     | 23/02/2001 | 000000049    | 1            | 69           |
| 8  | 327653         | 101213614 | CreditCard     | 07/12/2001 | 000000061    | 1            | 69           |
| 9  | 95799          | 101213614 | CreditCard     | 03/04/2001 | 000000064    | 1            | 79           |
| 10 | 91569          | 101213614 | CreditCard     | 30/03/2001 | 000000066    | 1            | 69           |
| 11 | 127498         | 101213614 | CreditCard     | 02/05/2001 | 000000079    | 1            | 89           |
| 12 | 161202         | 101213614 | CreditCard     | 03/06/2001 | 000000084    | 1            | 89           |
| 13 | 54689          | 101213614 | CreditCard     | 23/02/2001 | 000000084    | 2            | 178          |

The resulting dataframe will have one record per customer ID, and three fields for the required summary statistics:

|   | card_id   | Total items purchased | Total transactions | Preferred payment method |
|---|-----------|-----------------------|--------------------|--------------------------|
| 1 | 100000111 | 5105                  | 6                  | Cash                     |
| 2 | 100000199 | 3137                  | 3                  | CreditCard               |
| 3 | 100000343 | 726                   | 4                  | CreditCard               |
| 4 | 100000375 | 366                   | 4                  | CreditCard               |
| 5 | 100000392 | 8669                  | 31                 | DebitCard                |
| 6 | 100000444 | 3537                  | 22                 | DebitCard                |
| 7 | 100000462 | 1054                  | 3                  | DebitCard                |
| 8 | 100000480 | 1369                  | 4                  | CreditCard               |

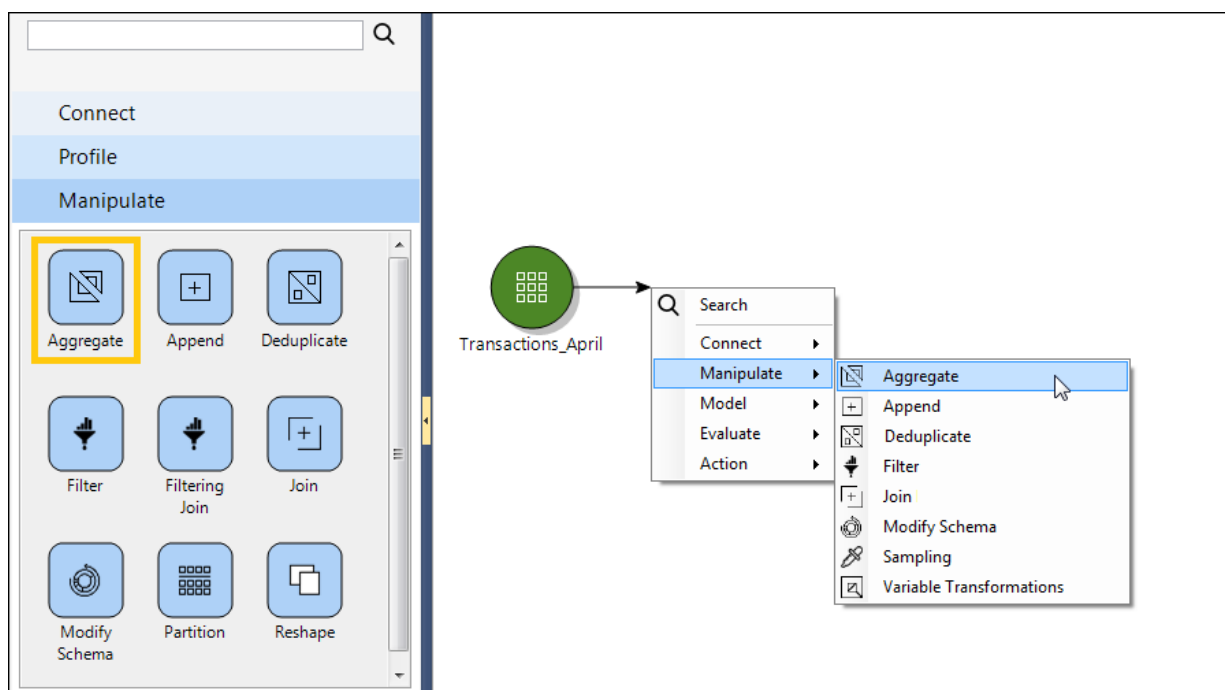
The **Aggregate** node in the **Manipulate** palette performs data aggregation for a Spark dataframe in the workflow, producing a new dataframe.

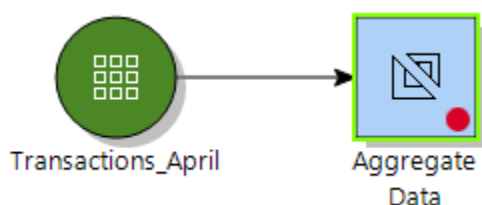
To perform data aggregation for a Spark dataframe in your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will appear. From the context menu, select **Manipulate --> Aggregate** as in the example screenshot below. A new **Aggregate** process node will be created.

OR:

- Drag the **Aggregate** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.





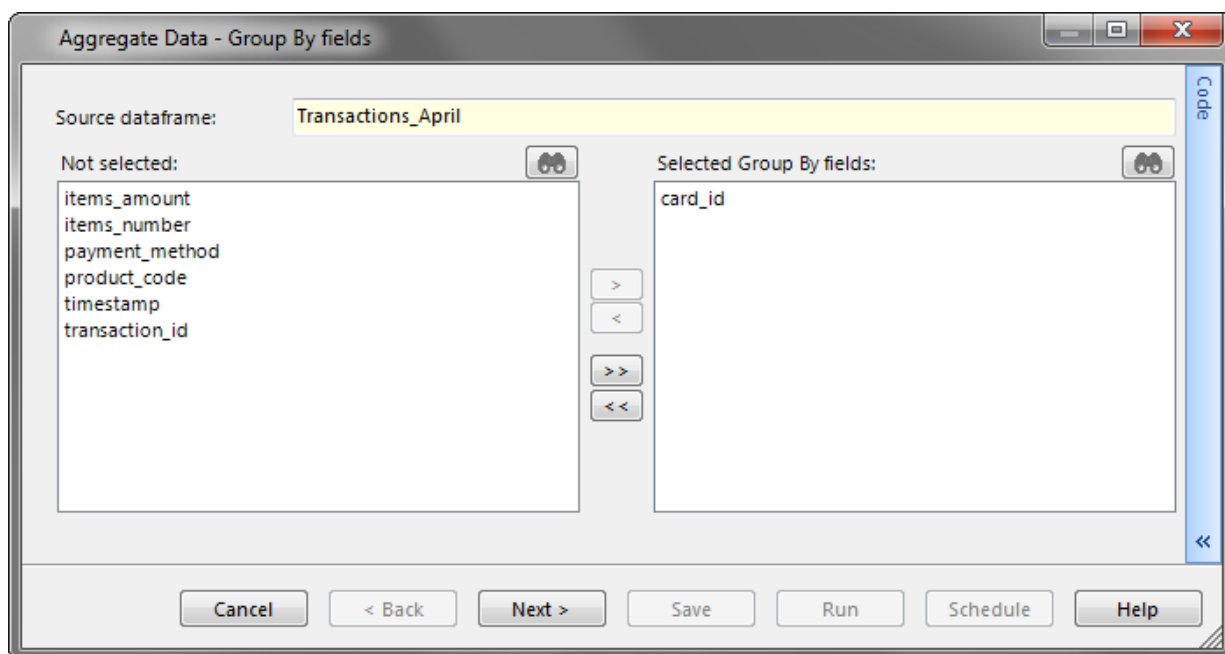
If the wizard does not open automatically, double-click the **Aggregate** node or right-click on it and select **Modify** to open the wizard.

In the first page of the wizard, select the field(s) to group by for the aggregate operation. These are the fields representing the attributes you need to generate summary statistics for.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.



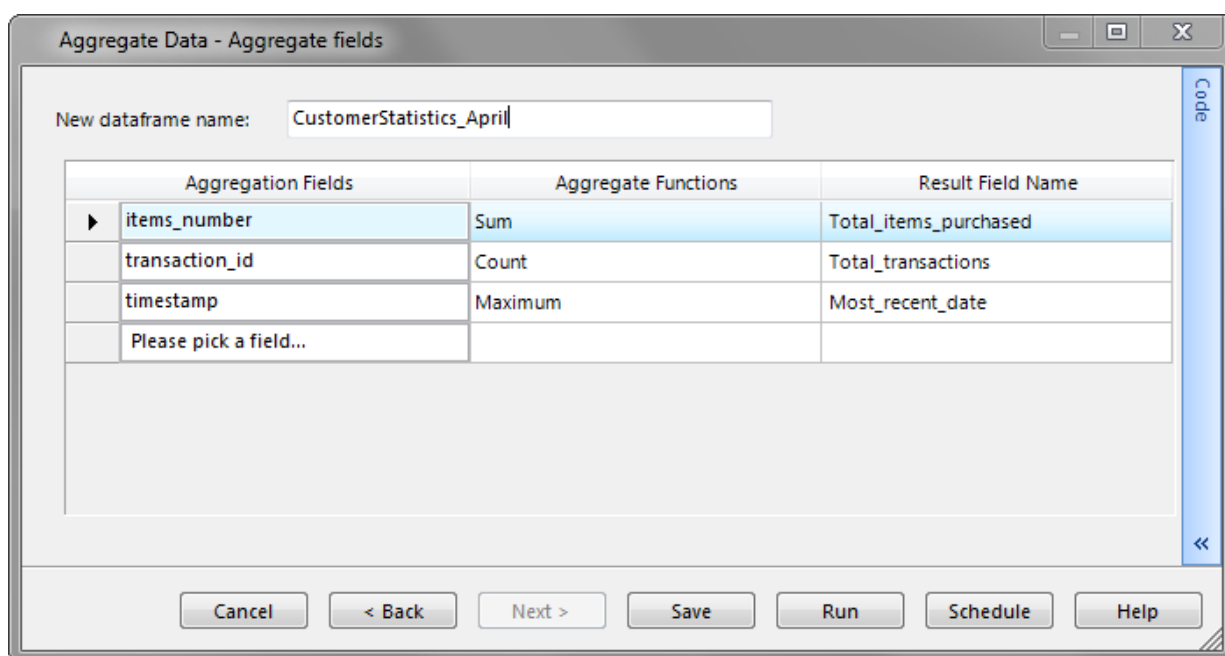
Click Next. In the *Aggregate fields* page:

- specify the name for the output (aggregated) dataframe
- select the aggregation fields, aggregate functions to be applied to each aggregation field, and specify the names for the resulting summary statistics fields.

The aggregate functions available for numeric fields are: *Sum*, *Average*, *Count*, *Maximum*, *Minimum*, *Variance*, and *Standard Deviation*. For other data types, available aggregate functions are: *Count*, *Maximum*, and *Minimum*.

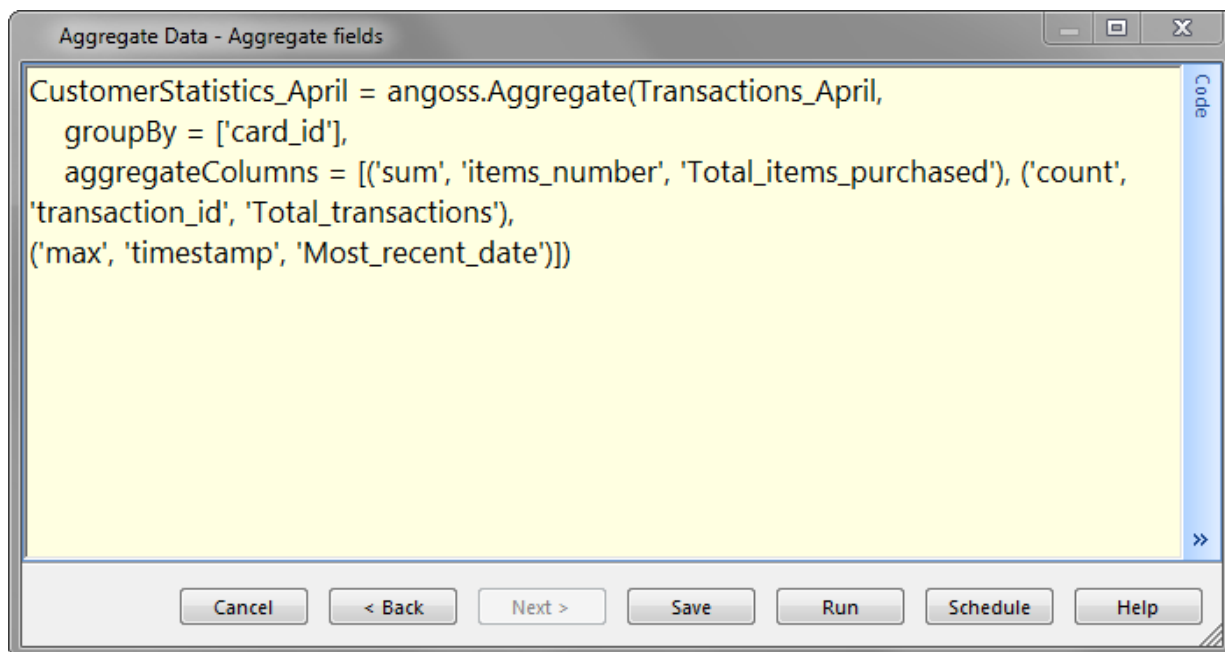
Page | 121

For example, the field *Total\_items\_purchased* in the screenshot below will represent the total number of items purchased - SUM([items\_number]) - for each individual Customer ID (the Group By field selected at the previous step).



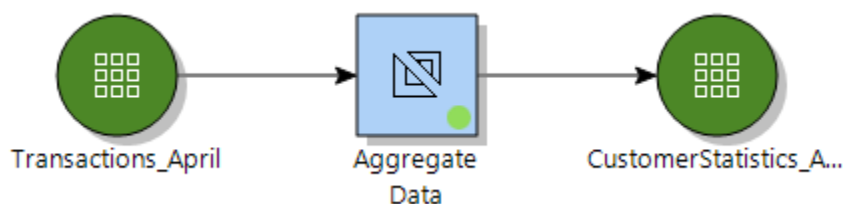
| Aggregation Fields     | Aggregate Functions | Result Field Name     |
|------------------------|---------------------|-----------------------|
| items_number           | Sum                 | Total_items_purchased |
| transaction_id         | Count               | Total_transactions    |
| timestamp              | Maximum             | Most_recent_date      |
| Please pick a field... |                     |                       |

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe is created in the workflow after running the node.



Double-click the output dataframe node to view the results.

## Append

The **Append** operation from the **Manipulate** palette merges two Spark dataframes by appending all records from one dataframe to the end of another.

One example is the task of merging customer transaction data that arrives in monthly batches. Since all the datasets have the same structure, records from one dataset must be appended to another dataset. This simple example shows retail transaction dataframes for January and February that need to be merged. The timestamp field indicates the transaction date:

|     | transaction_id | card_id   | payment_method | timestamp  | product_code | items_number |
|-----|----------------|-----------|----------------|------------|--------------|--------------|
| ▶ 1 | 3260           | 104711180 | Cash           | 04.01.2001 | 000000003    | 1            |
| 2   | 30405          | 102936649 | CreditCard     | 31.01.2001 | 000000003    | 1            |
| 3   | 28080          | 103696712 | CreditCard     | 28.01.2001 | 000000003    | 1            |
| 4   | 10637          | 100826685 | DebitCard      | 11.01.2001 | 000000003    | 1            |
| 5   | 18257          | 102633333 | DebitCard      | 19.01.2001 | 000000003    | 1            |
| 6   | 8202           | 100016812 | DebitCard      | 09.01.2001 | 000000003    | 1            |
| 7   | 6277           | 103940003 | Cash           | 07.01.2001 | 000000003    | 2            |

Page | 123

+

|     | transaction_id | card_id   | payment_method | timestamp  | product_code | items_number |
|-----|----------------|-----------|----------------|------------|--------------|--------------|
| ▶ 1 | 57280          | 100263804 | CreditCard     | 26.02.2001 | 000000003    | 1            |
| 2   | 55295          | 105112071 | DebitCard      | 24.02.2001 | 000000003    | 1            |
| 3   | 39738          | 105114408 | DebitCard      | 09.02.2001 | 000000003    | 1            |
| 4   | 49118          | 105201505 | CreditCard     | 18.02.2001 | 000000003    | 1            |
| 5   | 49749          | 106033341 | CreditCard     | 18.02.2001 | 000000003    | 1            |
| 6   | 51705          | 100770252 | CreditCard     | 20.02.2001 | 000000003    | 1            |

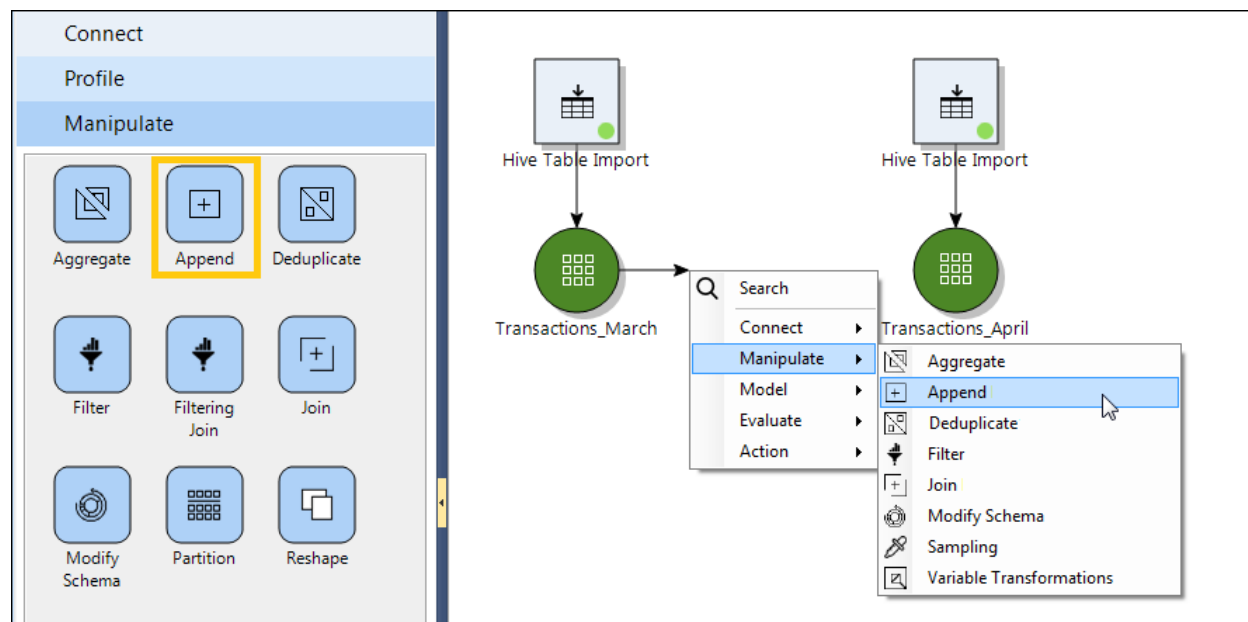
The resulting dataset will have the same columns, with the rows for February data appended at the end of the January data

The append operation normally requires that the two dataframes have the same structure, i.e., the same set of columns in the same order. The Append wizard provides additional flexibility by allowing the user to map the columns, so the names and order of the columns in the two dataframes need not be the same. However, the data type of the corresponding columns must match. Intelligent defaults for the mapping are provided to automatically map any columns that have the same name and data type in both dataframes.

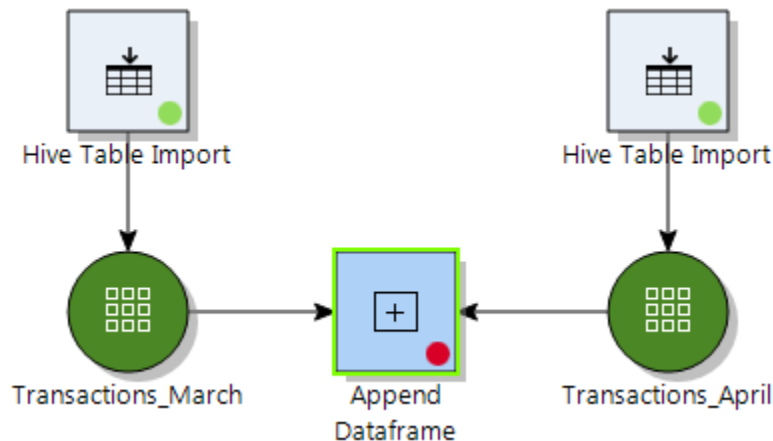
The first dataframe determines the output columns in the resulting dataframe. If Dataframe 1 has fewer columns than Dataframe 2, then the columns of Dataframe 1 not matched with any column in Dataframe 2 are appended with missing values (null).

## The Append Operation

To append one Spark dataframe to another, drag the **Append** node from the **Manipulate** palette onto the canvas and connect the two input dataframes to the node.



If the wizard does not open automatically, double-click the **Append** node or right-click on it and select **Modify**.



In the first page of the wizard, choose the order of the dataframes to append. Dataframe 2 will be appended at the end of Dataframe 1. Specify the name for the output dataframe. In the example below, the dataframe *Transactions\_April* will be appended to *Transactions\_March*, and the result will be named *Transactions\_March-April*. The default name for the output dataframe is <Name of Dataframe1>\_Appended.



Append Dataframe - Select Source Dataframes

Dataframe 1: Transactions\_March Dataframe 2: Transactions\_April

New dataframe name: Transactions\_MarchApril

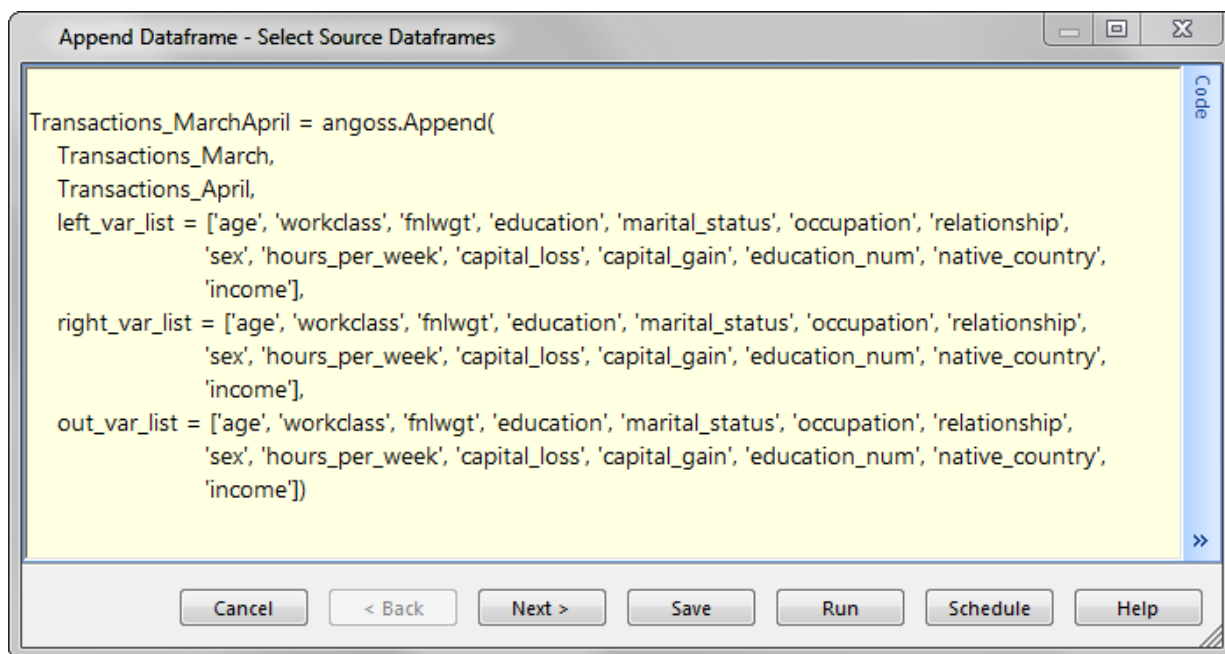
In the second page of the wizard, verify the mapping of the fields of Dataframe 1 to those of Dataframe 2 and change it if necessary. You will only be able to map the fields of the same data type (for example, Number to Number, String to String, etc.) To change the mapping for any field, click on the corresponding field in the Dataframe 2 Fields column and select the desired field from the list.

Append Dataframe - Field Mapping

Append [Transactions\_April] to [Transactions\_March]

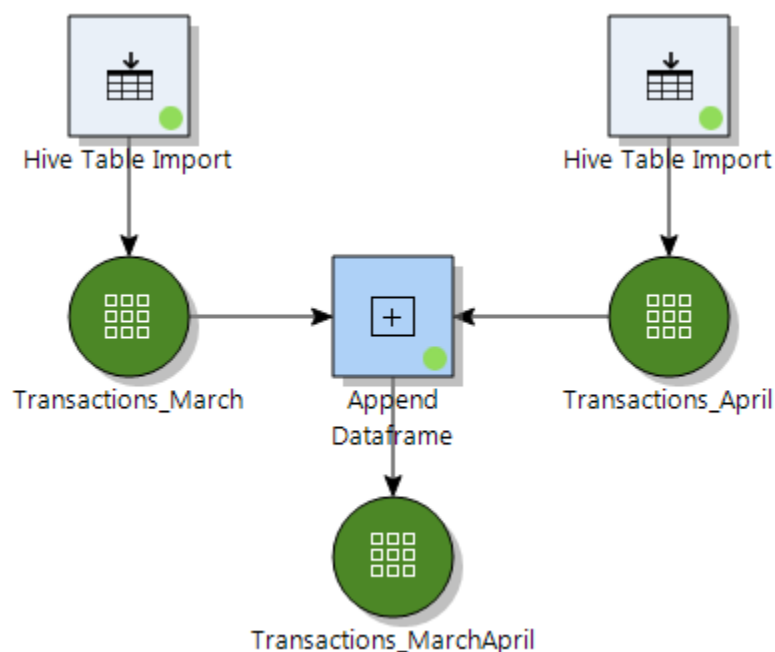
| Dataframe 1    | Dataframe 2    | Target         | Data Type |
|----------------|----------------|----------------|-----------|
| ▶ age          | age            | age            | Integer   |
| workclass      | workclass      | workclass      | String    |
| fnlwgt         | fnlwgt         | fnlwgt         | Number    |
| education      | education      | education      | String    |
| marital_status | marital_status | marital_status | String    |
| occupation     | occupation     | occupation     | String    |
| relationship   | relationship   | relationship   | String    |
| sex            | sex            | sex            | String    |
| hours_per_week | hours_per_week | hours_per_week | String    |
| capital_loss   | capital_loss   | capital_loss   | Number    |

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

As a result, the node representing the output dataframe appears on the workflow canvas. Double-click on the output dataframe to view the result.



## Deduplicate

The **Deduplicate** node in the **Manipulate** palette provides the capability to remove rows with duplicate values in all fields or in selected fields of a Spark dataframe in the workflow.

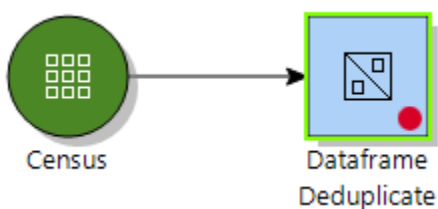
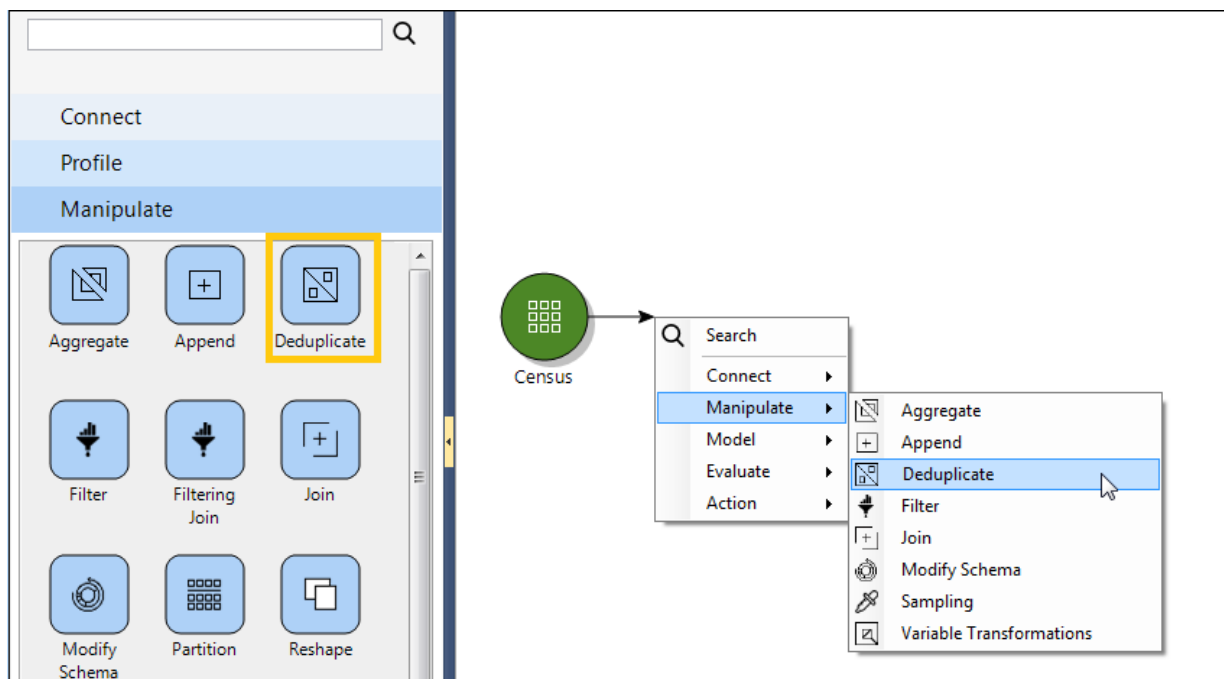
If specific fields are selected, then the row that replaces the removed rows is supposed to be the first row in the set of duplicates, but since the operation is applied to large-scale distributed storage, this remaining row may not necessarily be the first ("top") one in this set.

To deduplicate the rows in a dataframe:

- Drag a connection arrow from a Spark dataframe node to any area on the workflow canvas background. When you release the mouse button, a context menu will appear. From the context menu, select **Manipulate --> Deduplicate** as in the example screenshot below. A new **Deduplicate** process node will be created.

OR:

- Drag the **Deduplicate** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.



If the wizard does not open automatically, double-click the **Deduplicate** node or right-click on it and select **Modify** to open the wizard. Specify the options for the operation:

- Specify the name for the output (target) dataframe.
- Choose whether to deduplicate on entire row (remove only completely identical rows) or on specific columns (remove rows with duplicate values only in specific columns).
- If you selected the second option, choose the columns (the "Group By" variables) for your deduplication criterion, as shown in the example below.
- *Tip:* You can highlight multiple variables by clicking on a row and dragging the mouse down, or by clicking the rows while holding down the SHIFT or CTRL key. Then click in the checkbox for any of the highlighted rows to select all of them.

**Dataframe Deduplicate**

Source dataframe:

Target dataframe:

**Deduplicate Options**

☐ Deduplicate on entire row

☒ Deduplicate on column(s)

| #  | Variable Name  | <input checked="" type="checkbox"/> Select/Deselect All |
|----|----------------|---------------------------------------------------------|
| 10 | capital-gain   | <input type="checkbox"/>                                |
| 11 | capital-loss   | <input type="checkbox"/>                                |
| 12 | hours-per-week | <input type="checkbox"/>                                |
| 13 | native-country | <input type="checkbox"/>                                |
| 14 | Income         | <input type="checkbox"/>                                |
| 15 | ID             | <input checked="" type="checkbox"/>                     |
| 16 | First_Name     | <input checked="" type="checkbox"/>                     |
| 17 | Last_Name      | <input checked="" type="checkbox"/>                     |

Buttons: Cancel, Save, Run, Schedule, Help

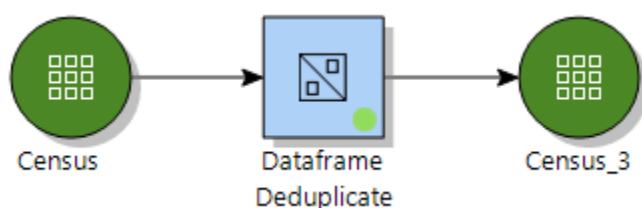
You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview. In the example below, the type of several numeric fields is changed from 'integer' to 'double'.

In the above example, the Python code is

```
Census_3 = angoss.Deduplicate(Census,  groupBy = ['ID',
'First Name', 'Last Name'])
```

Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe containing the format definitions is created in the workflow after running the node.



Double-click the output dataframe node to view the results.

## Filter

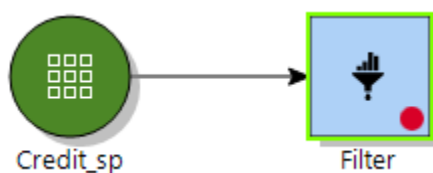
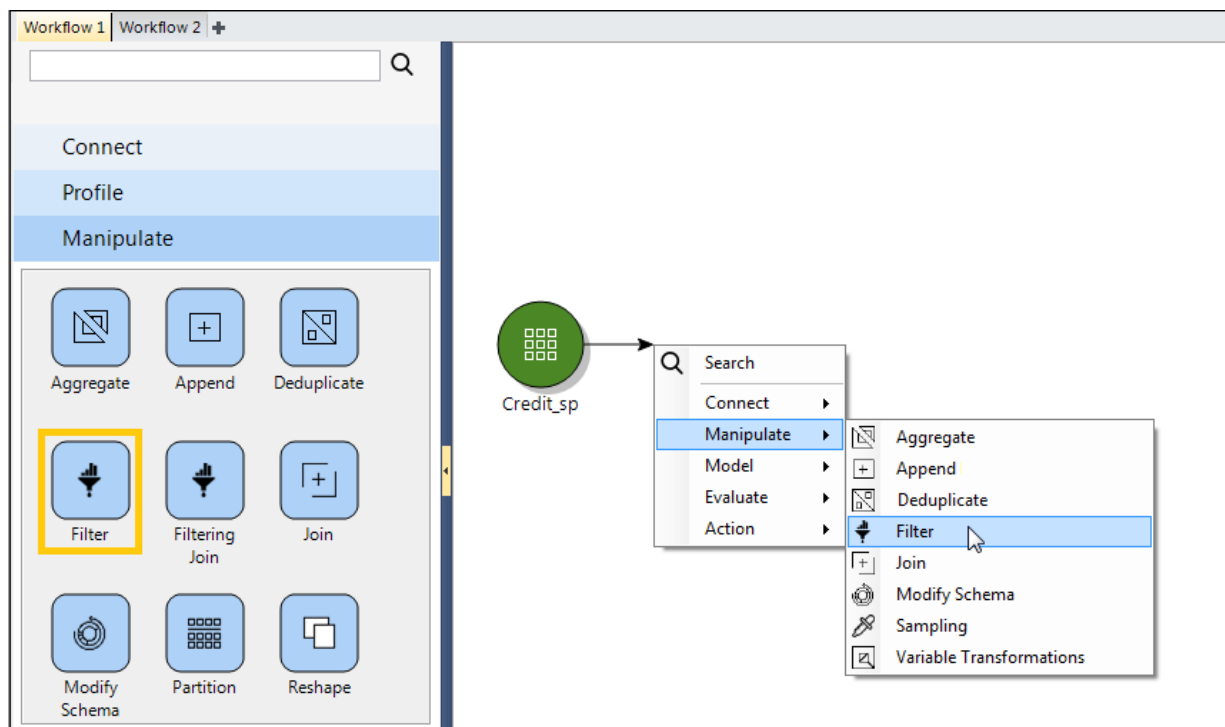
The **Filter** node in the **Manipulate** palette is used to select the records of a dataframe that satisfy certain conditions.

To apply a filter to a Spark dataframe in your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will appear. From the context menu, select **Manipulate** → **Filter** as in the example screenshot below. A new **Filter** process node will be created.

OR:

- Drag the **Filter** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.



If the wizard does not open automatically, double-click the **Filter** node or right-click on it and select **Modify** to open the wizard.

The wizard helps you construct an expression for the filter. Each row represents a condition on a certain field. Select the first condition field in the first column. Then specify the condition itself (usually some kind of relation, for example, `Profit` >=0). Click in the last column to select the operator (AND / OR) that applies to the next condition (if any).

Spark Filter

Source dataframe: Credit\_sp

Target dataframe: Credit\_sp\_P1

Expression builder:

| Field              | Condition        | Values | Operator |   |
|--------------------|------------------|--------|----------|---|
| Profit             | Greater or Equal | 0      | AND      | X |
| Initial Score      | Greater          | 550    | AND      | X |
| Residential Status | Is not null      |        |          | X |

Preview:

```
IF 'Profit' >= 0 AND 'Initial Score' > 550 AND ('Residential Status' IS NOT NULL)
```

Buttons: Cancel Save Run Schedule Help

Code

When specifying values, you can select the value or type a custom value. If you need to enter a value, click on the drop-down list in the right-hand side of the value cell and select Type, then enter the desired value and click Apply, as shown in the example below.

| Field              | Condition        | Values      | Operator |   |
|--------------------|------------------|-------------|----------|---|
| Profit             | Greater or Equal | 0           | AND      | X |
| Initial Score      | Greater          | 550 Type    | AND      | X |
| Residential Status | Is not null      | IS NOT NULL |          | X |

Type Value(s)

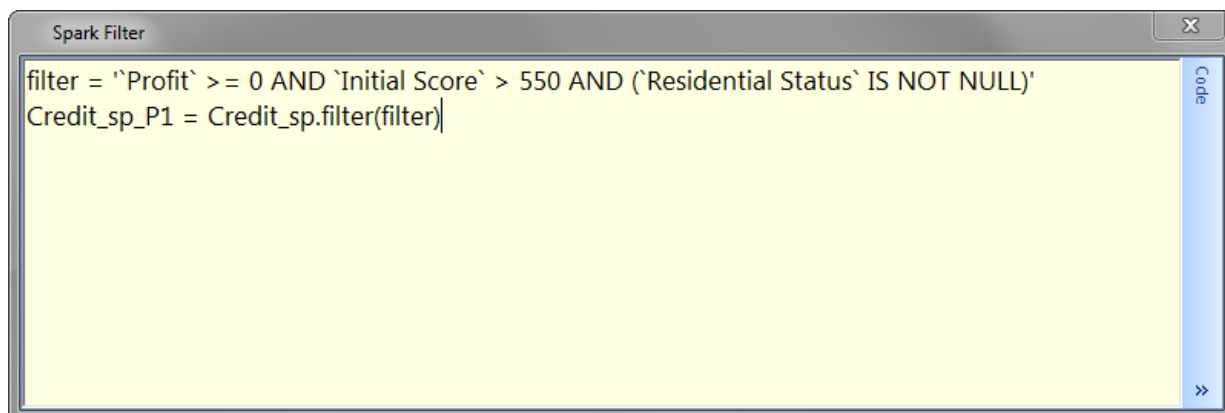
Single value:

550

Apply Cancel

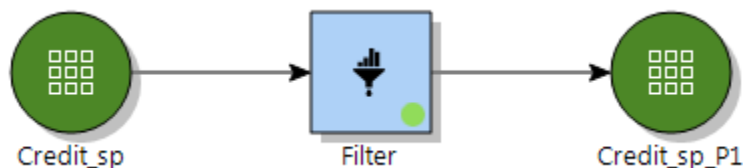
You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.





Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe is created in the workflow after running the node.



Double-click the output dataframe node to view the results.

## Filtering Join

The **Filtering Join** node in the **Manipulate** palette joins two Spark dataframes to produce a new one containing the records from the first input dataframe that either match (*semi-join*) or do not match (*anti-join*) in the second input dataframe.

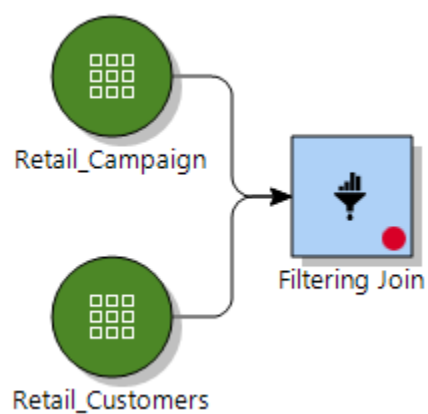
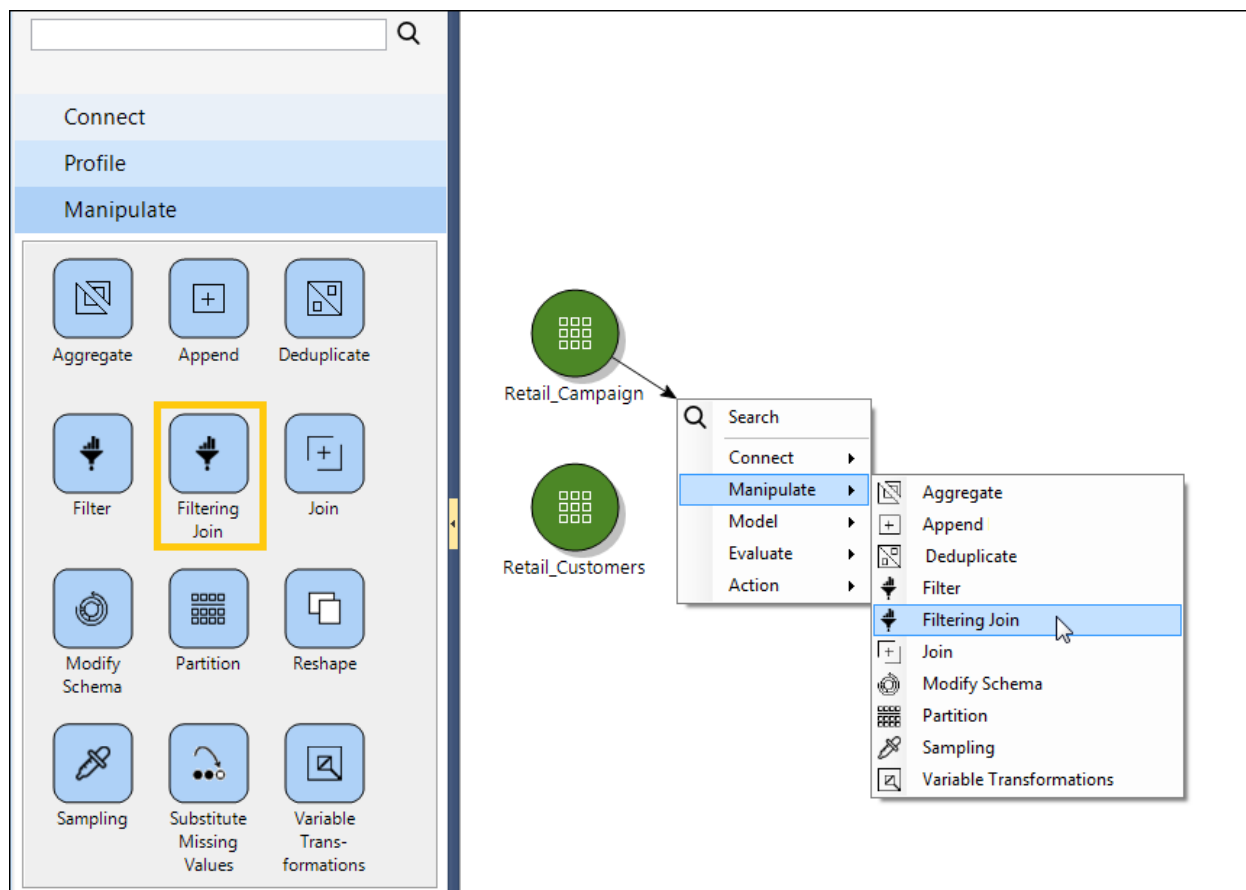
To join more than two dataframes, you need to consecutively join pairs of dataframes.

To perform a filtering join of two Spark dataframes:

- Drag a connection arrow from one of the dataframe nodes to any area on the Workflow canvas background. Releasing the mouse button brings up a context menu. From the context menu, select **Manipulate --> Filtering Join**, as in the example screenshot below. A new **Filtering Join** node will be created. Then connect the other input dataframe to this node.

OR:

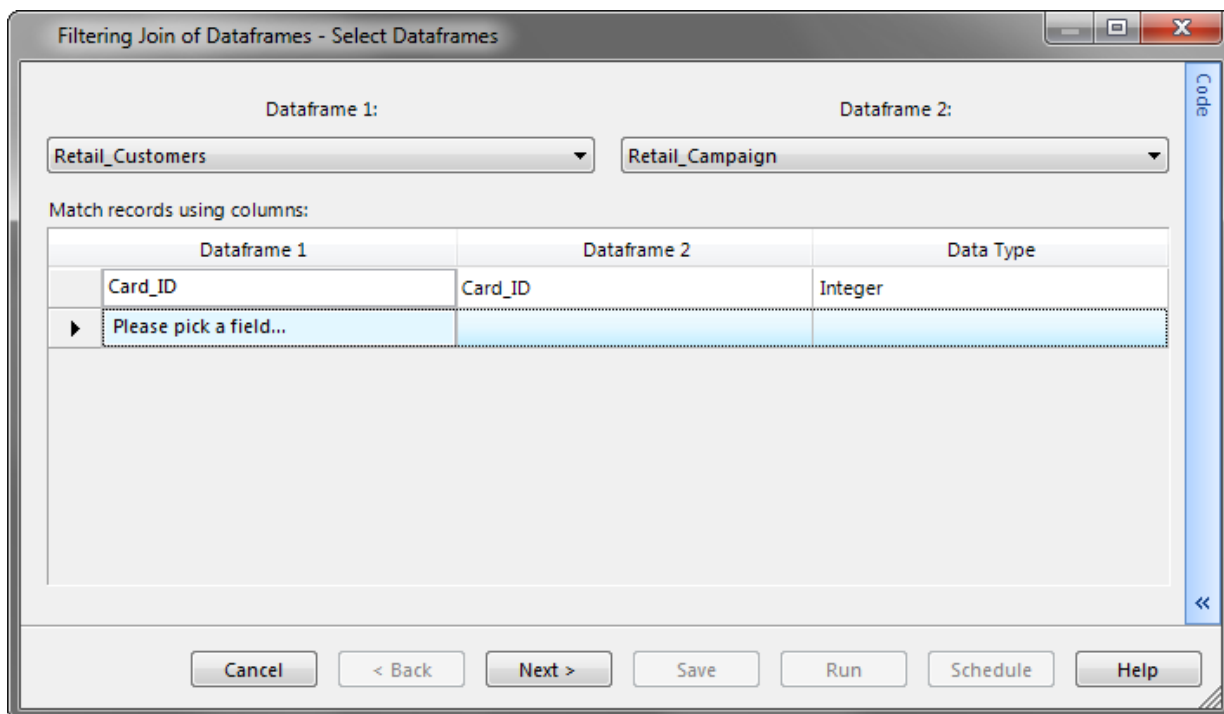
- Drag the **Filtering Join** node from the **Manipulate** palette onto the canvas and connect two input dataframes to it.



Double-click the **Filtering Join** node to open the wizard. Alternatively, right-click on it and select **Modify**.

In the first page of the wizard:

- select the order for the dataframes you wish to join (**Dataframe 1** and **Dataframe 2**). Dataframe 1 is the one whose records will be in the output, filtered by the match condition selected at the next step;
- select the join column(s) in each dataframe. The records in the input dataframes will be matched by the selected column(s). The corresponding columns should have compatible data types in both dataframes.



Filtering Join of Dataframes - Select Dataframes

Dataframe 1: Retail\_Customers

Dataframe 2: Retail\_Campaign

Match records using columns:

| Dataframe 1            | Dataframe 2 | Data Type |
|------------------------|-------------|-----------|
| Card_ID                | Card_ID     | Integer   |
| Please pick a field... |             |           |

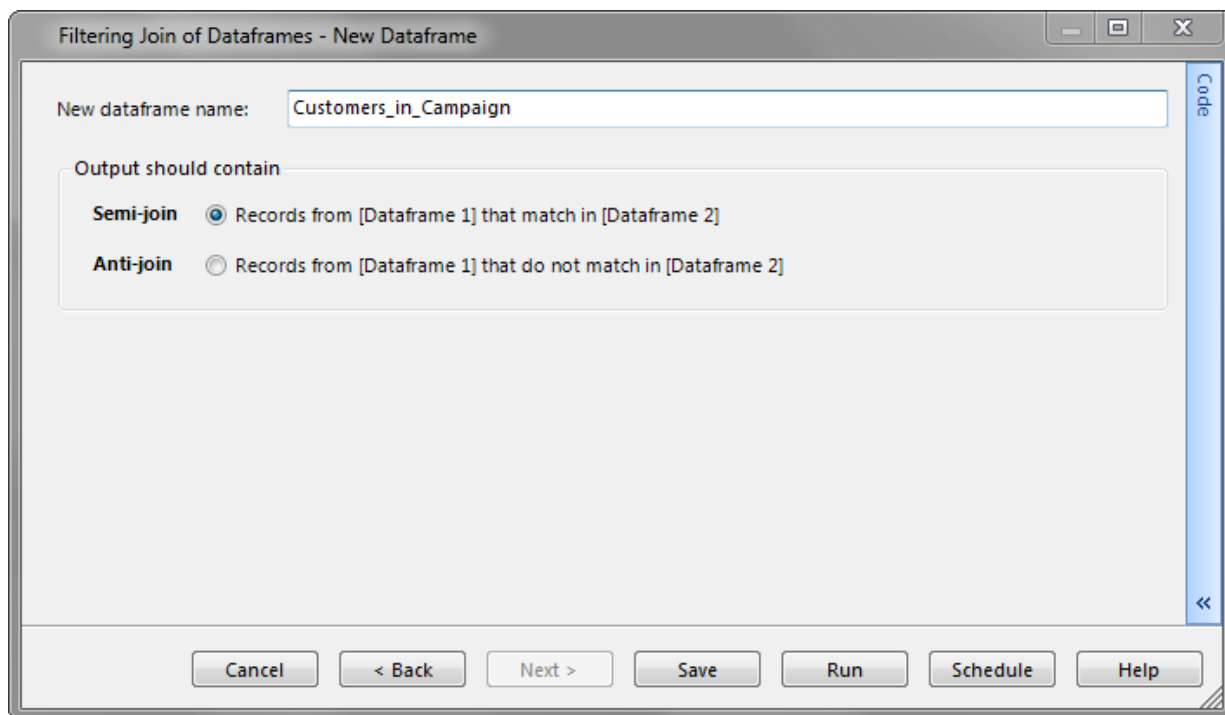
Buttons: Cancel, < Back, Next >, Save, Run, Schedule, Help

In the above example, a dataframe with customer demographics data is being joined with a dataframe containing campaign response data. Both dataframes have a Card ID field (loyalty card number).

Click **Next**. In the next page, specify the name for the output dataframe and select the join type depending on what records you want to include in the result:

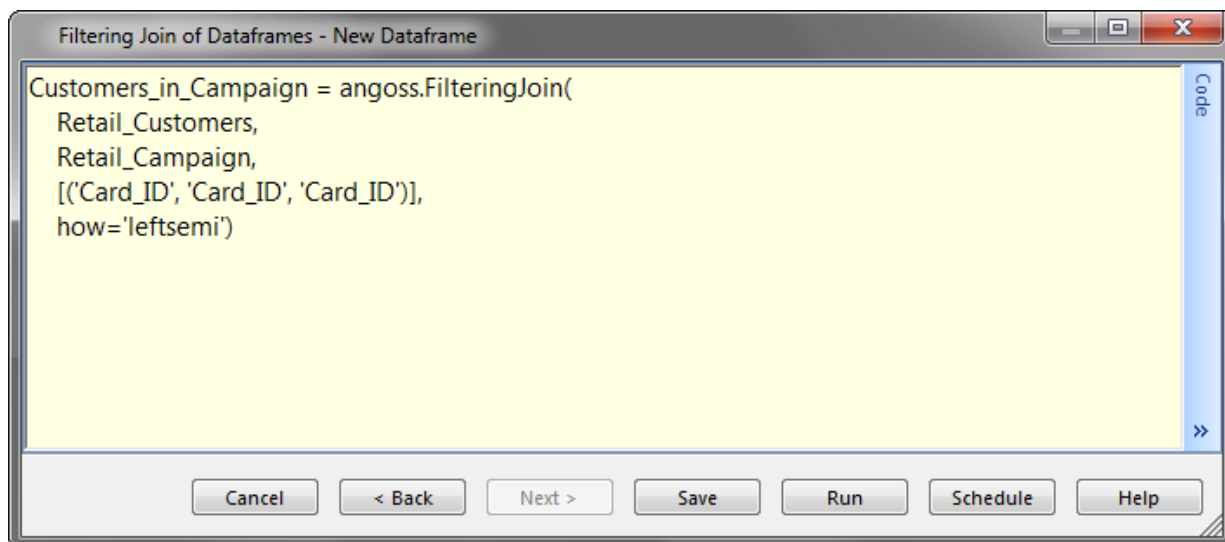
- **Semi-join**: Include the records from Dataframe 1 that match in Dataframe 2
- **Anti-join**: include the records from Dataframe 1 that do **not** match in Dataframe 2

In both cases, only columns from Dataframe 1 are included in the output.



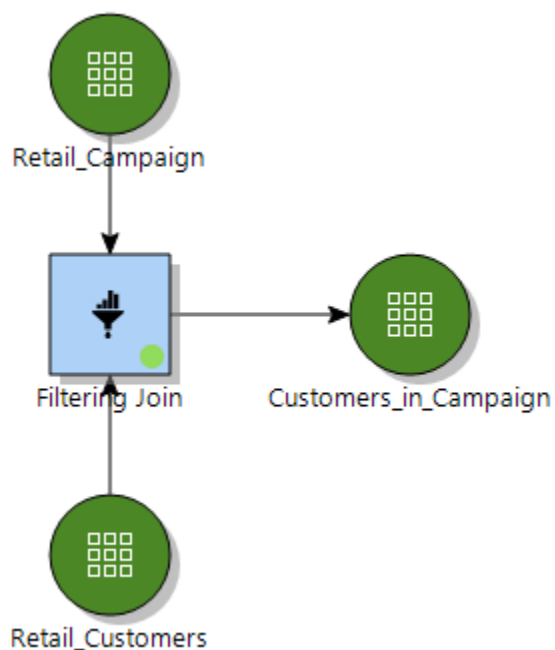
In the above example, only columns from the `Retail_Customers` dataframe will be present in the output dataframe because it was chosen as Dataframe 1. If *Semi-join* is selected, the output will contain the records from `Retail_Customers` that match in `Retail_Campaign` (customers who responded to the marketing campaign). If *Anti-join* is selected, the output will contain the records from `Retail_Customers` that do not match in `Retail_Campaign` (customers who were not included or did not respond to the marketing campaign).

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

Upon running the node double-click the output dataframe node to view the results.



## Join

The Spark dataframes to be joined must have at least one common attribute (field) to serve as a link. For general information about joins, see the first part of the topic [Joining dataframes](#).

Page | 138

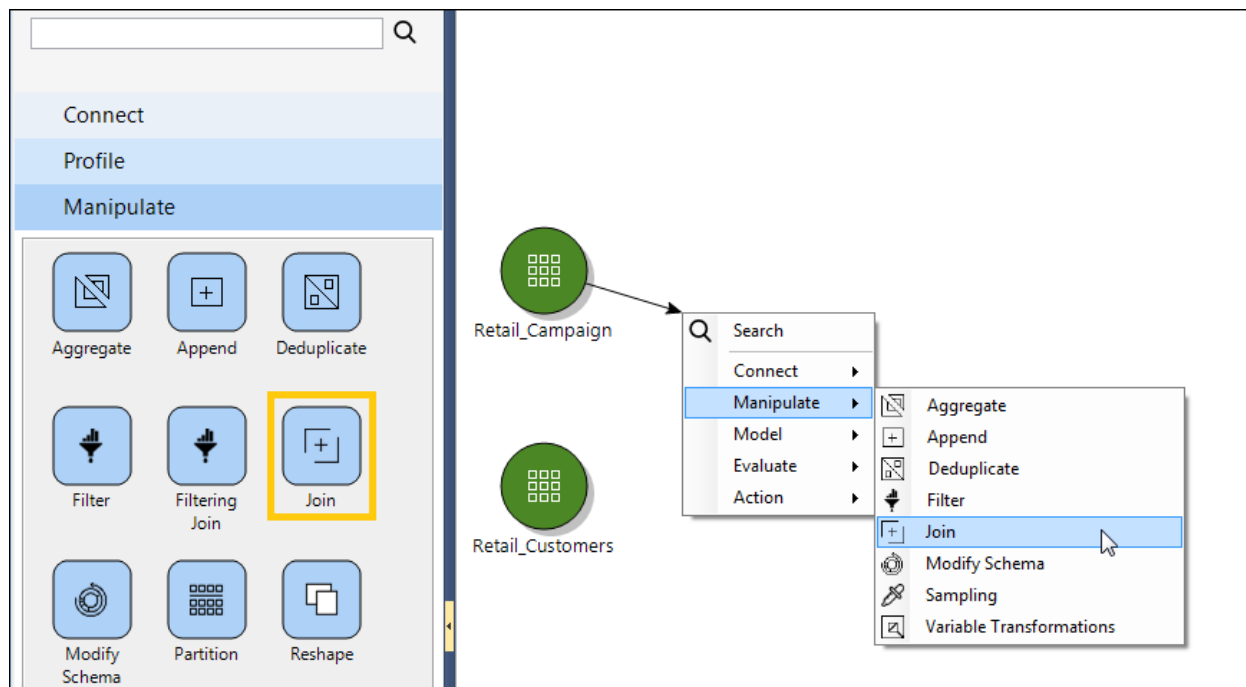
The **Join** node in the **Manipulate** palette creates a new dataframe that contains the selected fields from two source dataframes in your workflow, matched by the selected common field. If you need to join more than two dataframes, you need to consecutively join pairs of dataframes.

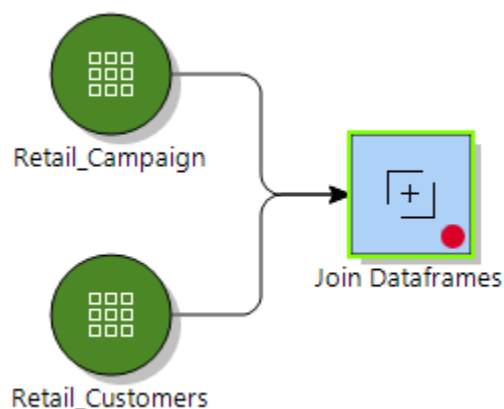
### To join two Spark dataframes:

- Drag a connection arrow from one of the dataframe nodes to any area on the Workflow canvas background. Releasing the mouse button brings up a context menu. From the context menu, select **Manipulate --> Join** as in the example screenshot below. A new **Join** process node will be created. Then connect the other input dataframe(s) to this node.

OR:

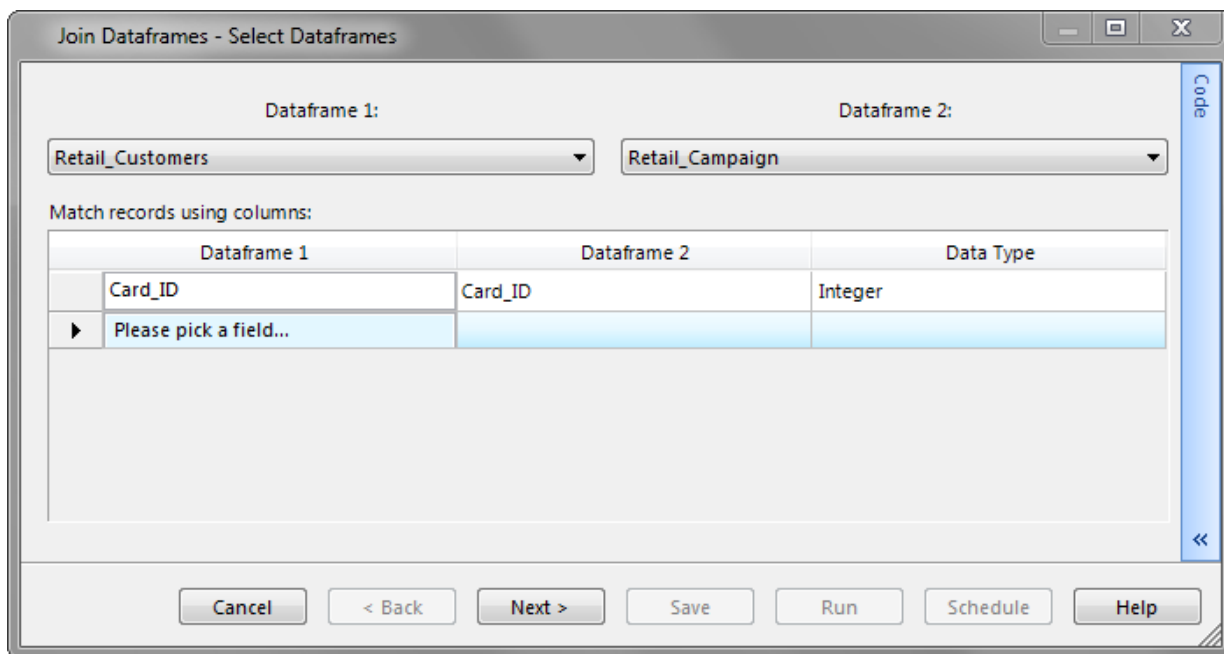
- Drag the **Join** node from the **Manipulate** palette onto the canvas and connect the input dataframes to it.





Double-click the **Join** node or right-click on it and select **Modify**.

In the first page of the wizard, select the order for the dataframes you wish to join (Dataframe 1 and Dataframe 2). Select the join column(s) in each dataframe. The records in the input datasets will be matched by the selected column(s). Make sure the corresponding columns have the same data type in both datasets.



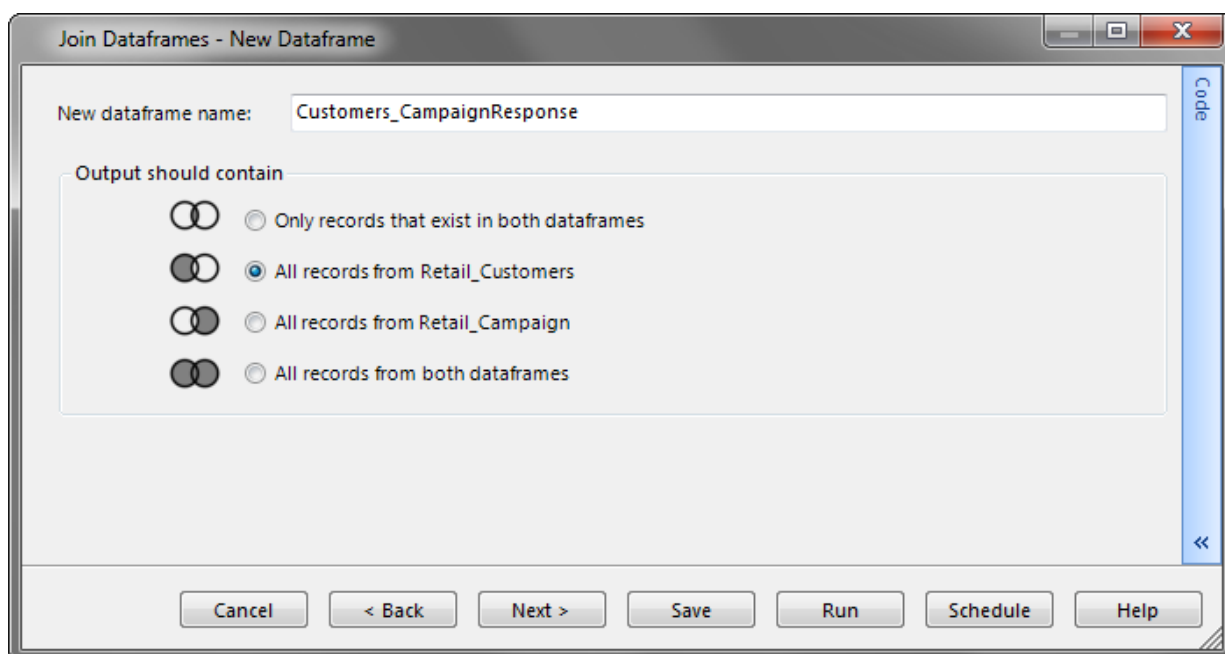
The screenshot shows the 'Join Dataframes - Select Dataframes' dialog box. It has two dropdown menus at the top: 'Dataframe 1:' set to 'Retail\_Customers' and 'Dataframe 2:' set to 'Retail\_Campaign'. Below these is a section titled 'Match records using columns:' containing a table with three columns: 'Dataframe 1', 'Dataframe 2', and 'Data Type'. The first row shows 'Card\_ID' for both dataframes with a data type of 'Integer'. The second row has a dropdown arrow and the text 'Please pick a field...'. At the bottom of the dialog are several buttons: 'Cancel', '< Back', 'Next >', 'Save', 'Run', 'Schedule', and 'Help'. A 'Code' button is visible on the right side of the dialog.

| Dataframe 1              | Dataframe 2 | Data Type |
|--------------------------|-------------|-----------|
| Card_ID                  | Card_ID     | Integer   |
| ▶ Please pick a field... |             |           |

In the above example, a dataframe with campaign response data is being joined with a dataframe with demographics data. Both dataframes have a Card ID field.

Click Next. In the *New dataframe* page, specify the name for the new dataframe and select the type of join depending on which records you wish to include in the result. The available options are:

- include only records that exist in both dataframes (*inner join*) - this is the default option
- include all records from Dataframe 1 (*left outer join*)
- include all records from Dataframe 2 (*right outer join*)
- include all records from both dataframes

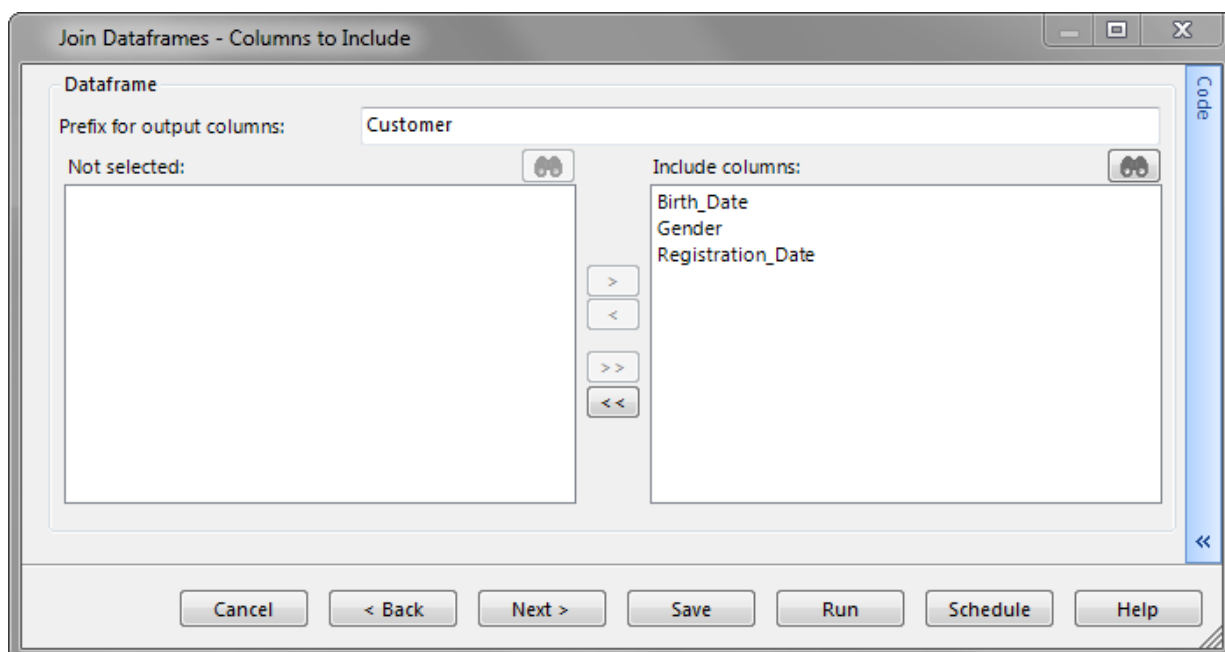


In the case where all records from one of the dataframes are included (left or right outer join), any records that do not match will have null values inserted for attributes from the unmatched side of the join.

The default output dataframe name is the concatenation of the two source dataframe names with an underscore in between. Modify it if necessary.

In the next page, select the fields to include in the output from the first dataframe and specify an optional prefix for these fields in the new dataframe. The prefix is followed by an underscore. You will be able to modify the field names at the last wizard step if necessary.

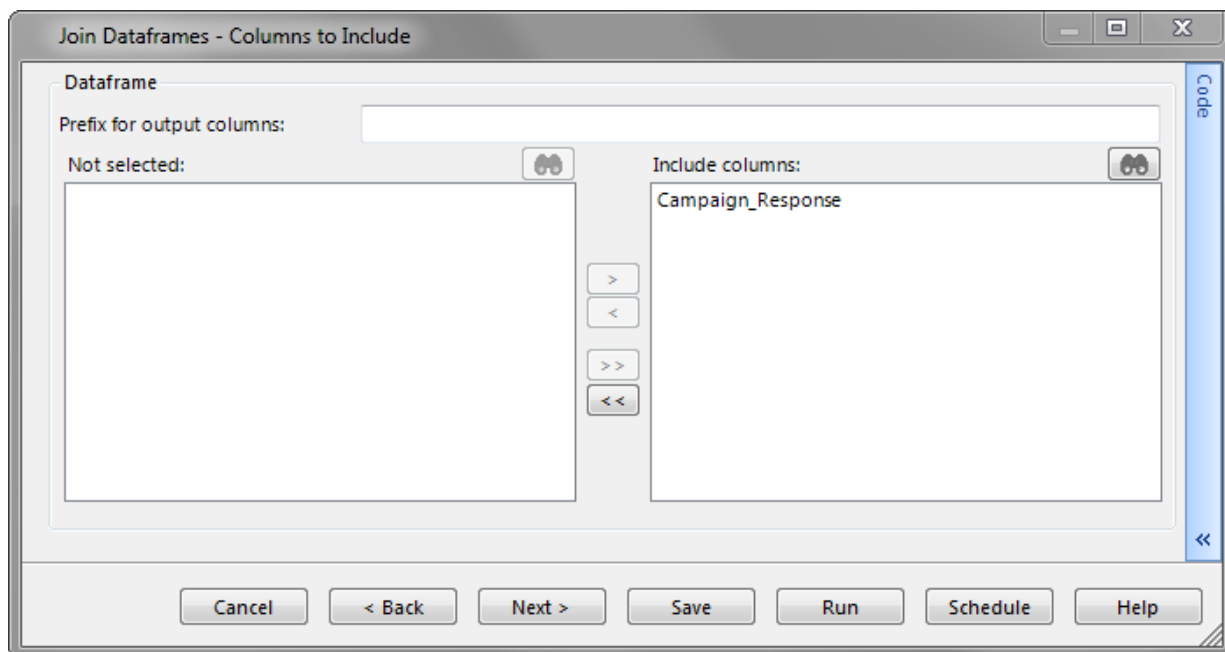


*Tip:*

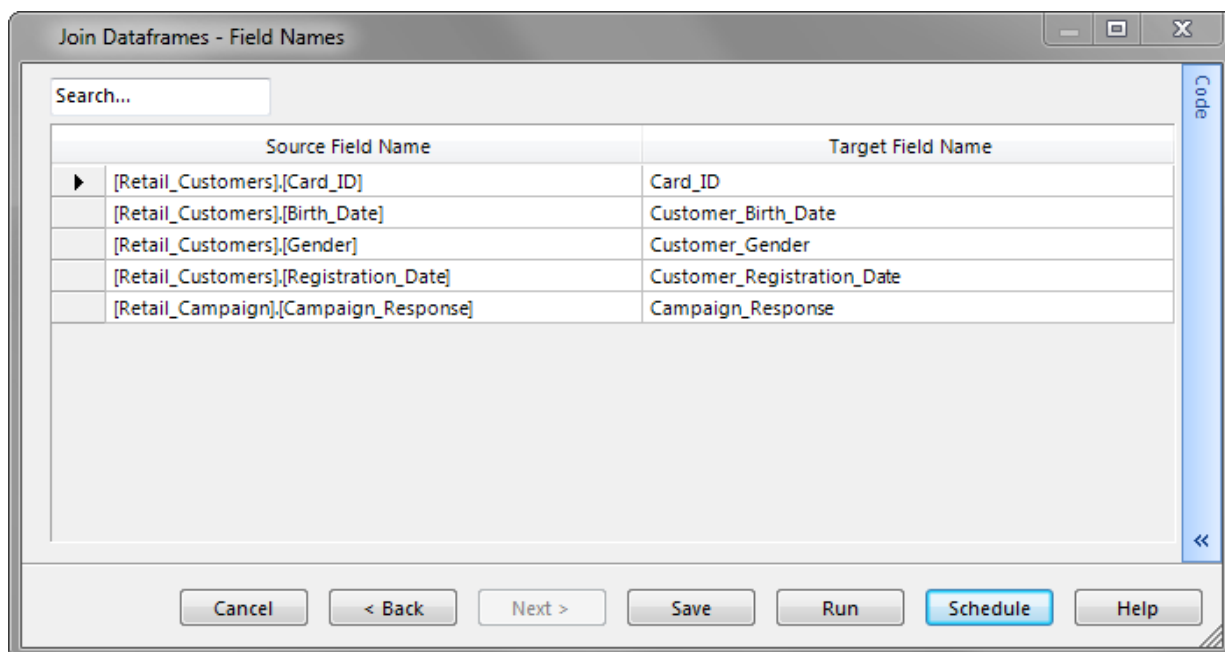
You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

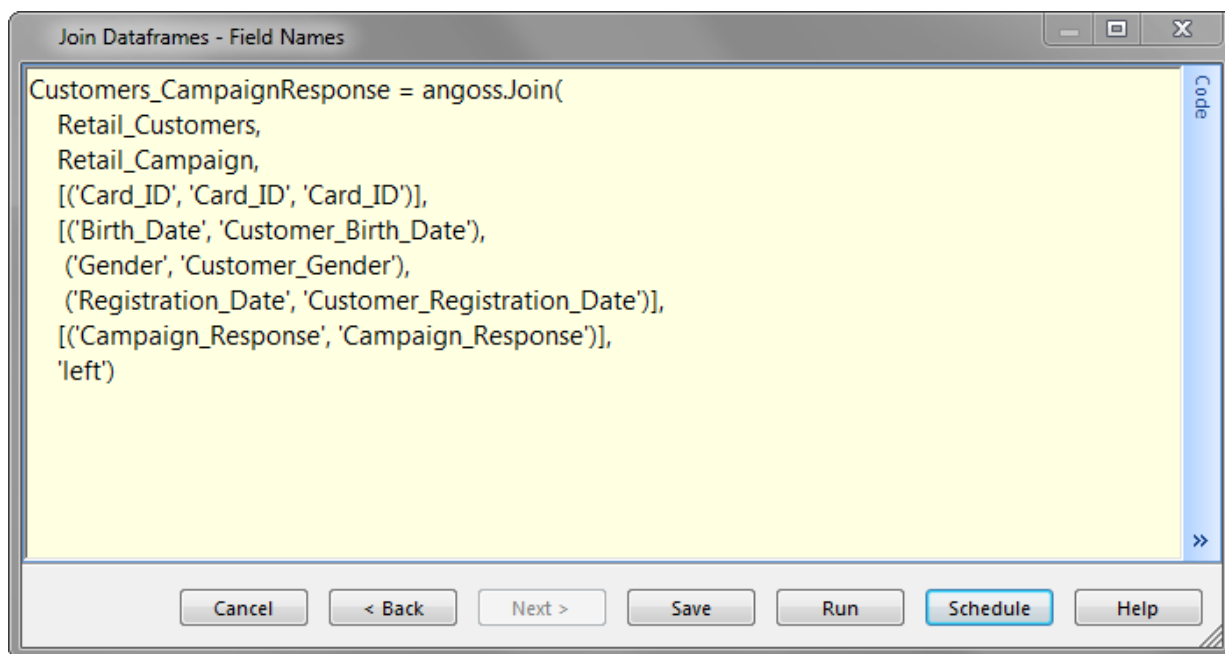
In the next page, select the fields to include in the output from the second dataframe and specify an optional prefix for these fields in the new dataframe. You will be able to modify the field names at the last wizard step if necessary.



In the next wizard page you have an option of specifying field names for the new dataframe if you want them to be different from the names generated automatically. If the resulting dataframe represents a table in the In-Database Analytics mode, make sure the new field names comply with the naming standards of your target database. For example, field names in different databases may have restrictions regarding their length, special characters, etc.



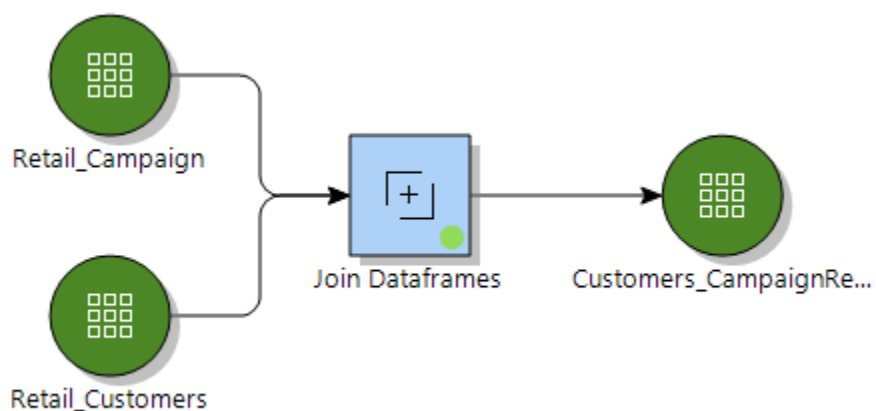
You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



```
Customers_CampaignResponse = angoss.Join(  
    Retail_Customers,  
    Retail_Campaign,  
    [('Card_ID', 'Card_ID', 'Card_ID')],  
    [('Birth_Date', 'Customer_Birth_Date'),  
     ('Gender', 'Customer_Gender'),  
     ('Registration_Date', 'Customer_Registration_Date')],  
    [('Campaign_Response', 'Campaign_Response')],  
    'left')
```

Page | 143

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.



Double-click the output dataframe node to view the results.

## Modify Schema

The **Modify Schema** node in the **Manipulate** palette provides the capability to modify names and data types of any fields in a Spark dataframe in the workflow.

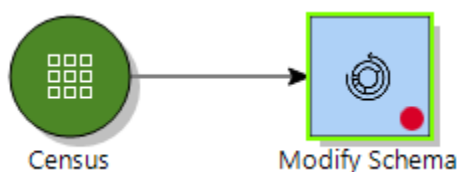
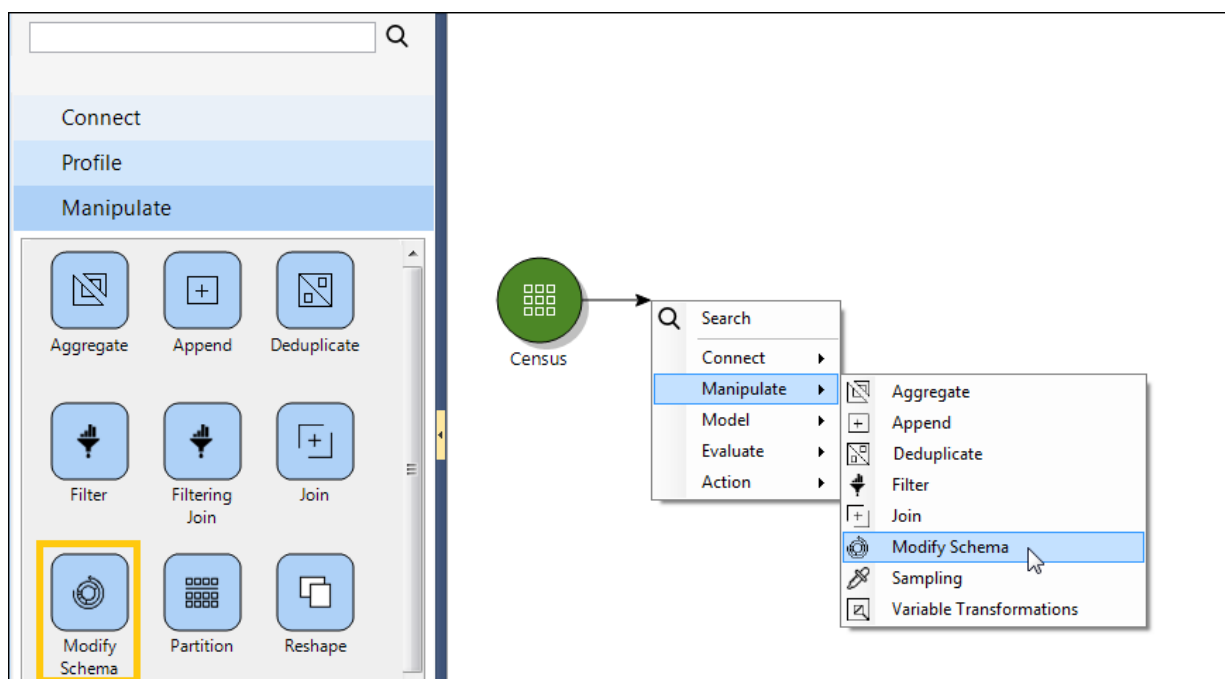
Page | 144

To change the names and / or data types of any fields in a dataframe:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will appear. From the context menu, select **Manipulate --> Modify Schema** as in the example screenshot below. A new **Modify Schema** process node will be created.

OR:

- Drag the **Modify Schema** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.

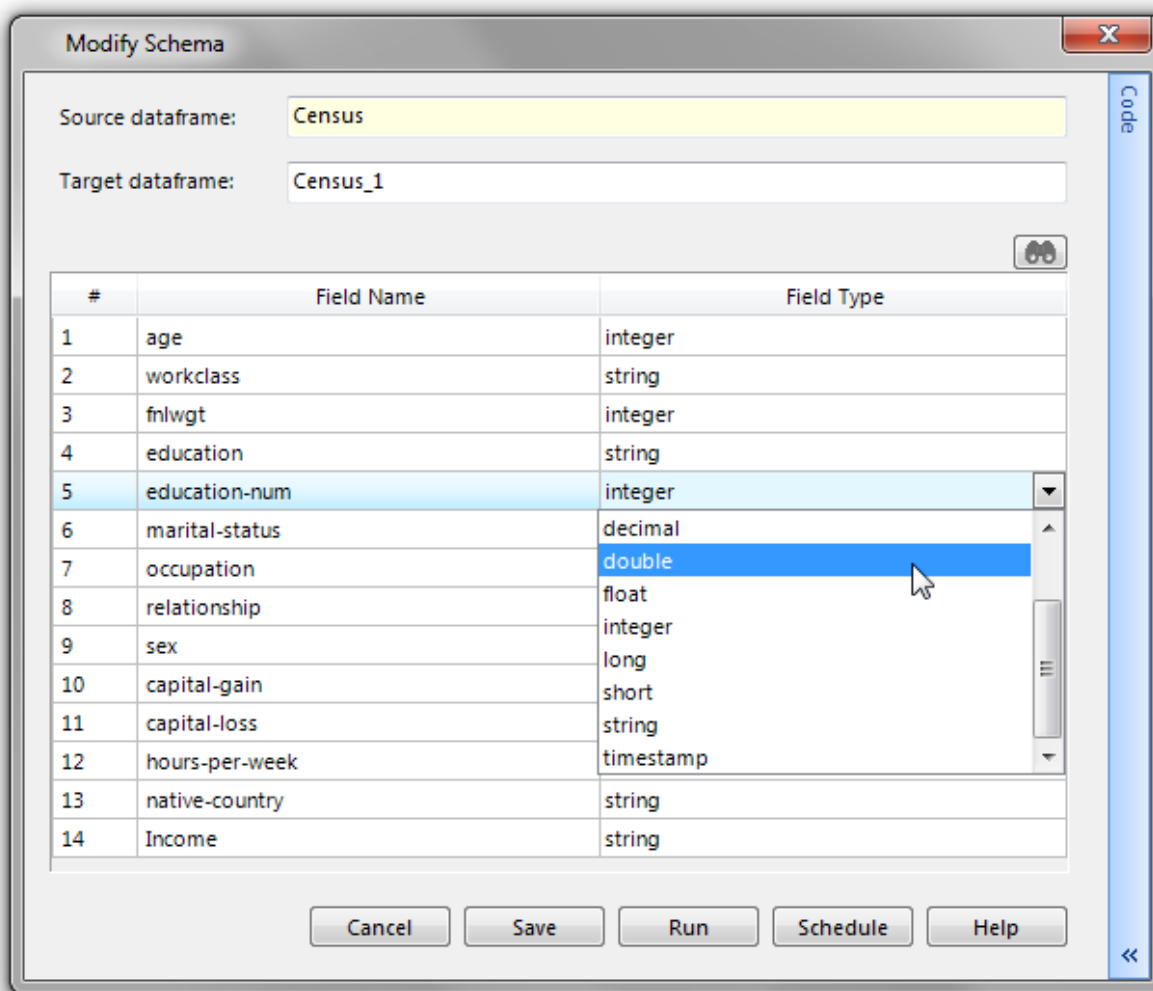


If the wizard does not open automatically, double-click the **Modify Schema** node or right-click on it and select **Modify** to open the wizard. Specify the options for the operation:

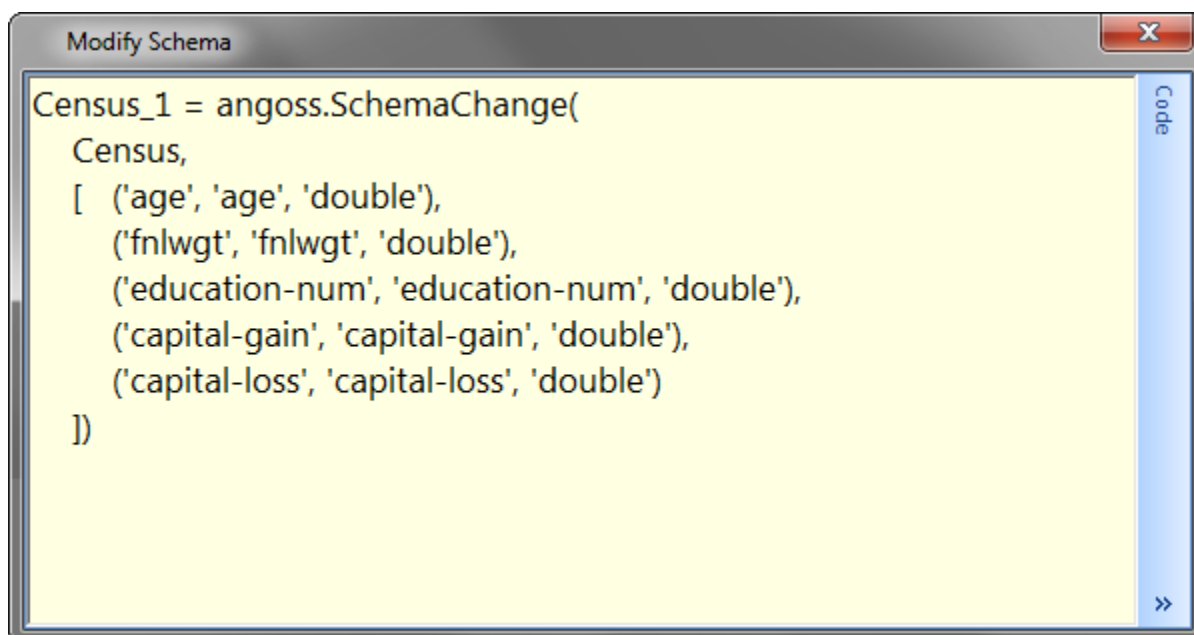
- Specify the name for the output (target) dataframe.
- To change the data type of a field, click in the Field Type column for the desired field and select the required type from the drop-down list as shown below.
- To modify the name of a field, just click and type in the Field Name column for the desired field.

Page | 145

To quickly find a field, click the [Field Finder](#) button , which is especially useful when there are a lot of variables (hundreds or thousands).



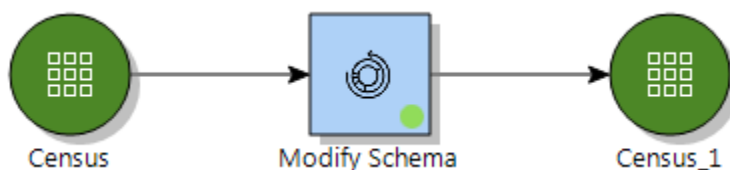
You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview. In the example below, the type of several numeric fields is changed from 'integer' to 'double'.



```
Census_1 = angoss.SchemaChange(  
  Census,  
  [ ('age', 'age', 'double'),  
    ('fnlwgt', 'fnlwgt', 'double'),  
    ('education-num', 'education-num', 'double'),  
    ('capital-gain', 'capital-gain', 'double'),  
    ('capital-loss', 'capital-loss', 'double')  
  ]  
)
```

Click **Save** to save the settings, or click **Run** to save the settings and run the node immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe is created in the workflow after running the node.



Double-click the output dataframe node to view the results.

## Partition

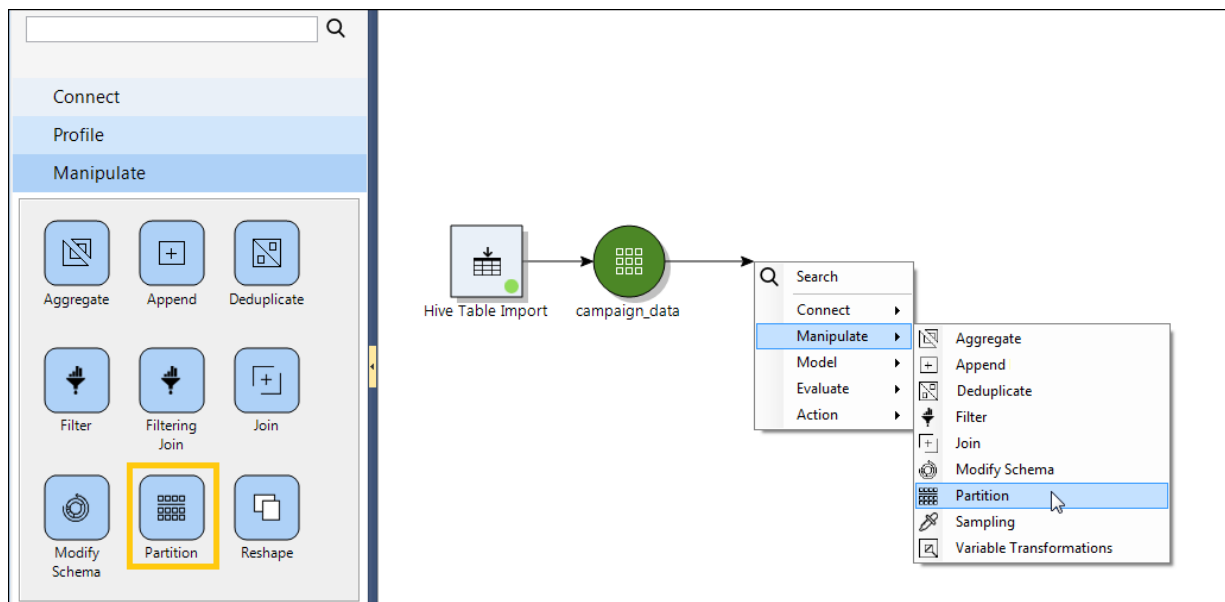
The **Partition** node in the **Manipulate** palette is used to partition a dataframe into disjoint subsets (partitions).

To partition a Spark dataframe in your workflow:

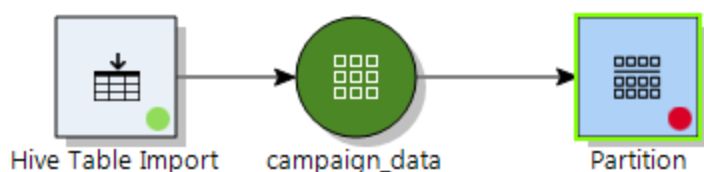
- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will appear. From the context menu, select **Manipulate → Partition** as in the example screenshot below. A new **Partition** process node will be created.

OR:

- Drag the **Partition** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.



If the wizard does not open automatically, double-click the **Partition** node or right-click on it and select **Modify** to open the wizard.



In the wizard, click "Add partition..." in the Child dataframe column to define a new partition. In the *Sampling weight* field, specify a number between 0 and 1 to indicate the portion of the parent dataset that your partition should represent. For example, enter 0.7 for a 70% partition.

**Dataframe Partition** [X]

Parent Dataframe:

Random Seed:

| Child dataframe     | Sampling weight | Delete |
|---------------------|-----------------|--------|
| cmp_data_training   | 0.7             | Delete |
| cmp_data_validation | 0.3             | Delete |
| Add partition ...   | 0               | Delete |

Cancel Save Run Schedule Help

Code

<<

The *Random seed* option is provided for the repeatability of results.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.

**Dataframe Partition** [X]

```
(cmp_data_training, cmp_data_validation) = \
    angoss.Partition(
        data = campaign_data,
        seed = 1,
        weights = [0.7, 0.3])
```

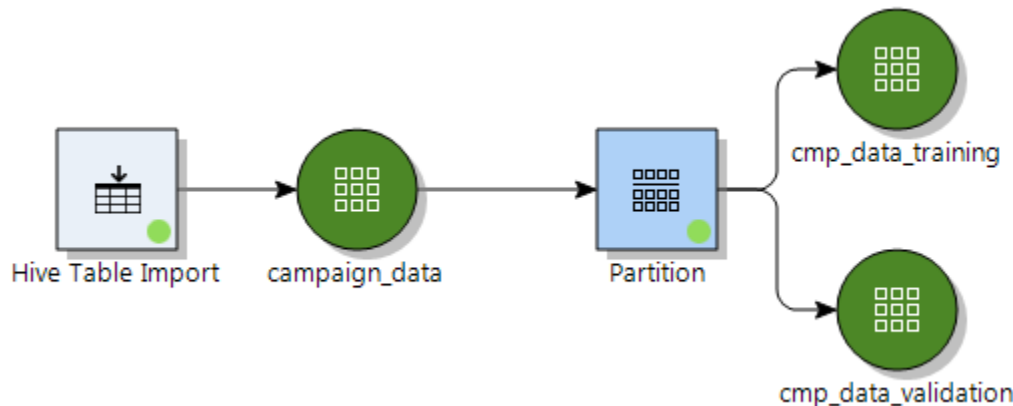
Code

>>



Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframes are created in the workflow after running the node.



Double-click the output dataframe nodes to view the results.

## Reshape

The **Reshape Dataframe** node in the **Manipulate** palette reshapes Spark dataframes using a pivot transformation of one of the following two types:

1. **"Wide-to-Long"** transformation: Several columns are converted into a single column by stacking the values from the original columns, while one or several other columns are used as key(s).

Example: The input dataframe contains the information about stock items shipped or received. Each row indicates the transaction date / time and the amount of each of the two item types. With the "Wide-to-Long" reshape, the amount values from two columns (*Item1* and *Item2*) are stacked into a new column called *Amount*, and the new column *Stock\_item* keeps the item names.



|    | Date       | Time     | Item1 | Item2 |
|----|------------|----------|-------|-------|
| 1  | 2001/1/01  | 18:05:09 | 228   | 100   |
| 2  | 2001/1/01  | 19:22:25 | 79    | 79    |
| 3  | 2001/1/01  | 20:22:17 | 808   | 554   |
| 4  | 2001/1/01  | 22:07:19 | 948   | 409   |
| 5  | 2001/3/01  | 21:27:20 | 628   | 596   |
| 6  | 2001/6/01  | 21:21:18 | 215   | 22    |
| 7  | 2001/7/01  | 20:21:10 | 150   | 74    |
| 8  | 2001/8/01  | 20:29:09 | 128   | 5     |
| 9  | 2001/9/01  | 18:19:20 | 640   | 225   |
| 10 | 2001/9/01  | 21:10:18 | 1260  | 1230  |
| 11 | 2001/10/01 | 18:01:27 | 424   | 193   |
| 12 | 2001/10/01 | 19:03:10 | 192   | 127   |
| 13 | 2001/10/01 | 19:10:03 | 584   | 243   |
| 14 | 2001/10/01 | 20:01:28 | 812   | 318   |
| 15 | 2001/10/01 | 20:16:29 | 1428  | 931   |
| 16 | 2001/12/01 | 20:25:19 | 1800  | 1247  |
| 17 | 2002/1/01  | 15:00:16 | 1602  | 1356  |
| 18 | 2002/3/02  | 20:06:09 | 700   | 492   |
| 19 | 2002/3/02  | 20:14:01 | 195   | 186   |

|    | Date      | Time     | Stock_item | Amount |
|----|-----------|----------|------------|--------|
| 1  | 2001/1/01 | 18:05:09 | Item1      | 228    |
| 2  | 2001/1/01 | 18:05:09 | Item2      | 100    |
| 3  | 2001/1/01 | 19:22:25 | Item1      | 79     |
| 4  | 2001/1/01 | 19:22:25 | Item2      | 79     |
| 5  | 2001/1/01 | 20:22:17 | Item1      | 808    |
| 6  | 2001/1/01 | 20:22:17 | Item2      | 554    |
| 7  | 2001/1/01 | 22:07:19 | Item1      | 948    |
| 8  | 2001/1/01 | 22:07:19 | Item2      | 409    |
| 9  | 2001/3/01 | 21:27:20 | Item1      | 628    |
| 10 | 2001/3/01 | 21:27:20 | Item2      | 596    |
| 11 | 2001/6/01 | 21:21:18 | Item1      | 215    |
| 12 | 2001/6/01 | 21:21:18 | Item2      | 22     |
| 13 | 2001/7/01 | 20:21:10 | Item1      | 150    |
| 14 | 2001/7/01 | 20:21:10 | Item2      | 74     |
| 15 | 2001/8/01 | 20:29:09 | Item1      | 128    |
| 16 | 2001/8/01 | 20:29:09 | Item2      | 5      |
| 17 | 2001/9/01 | 18:19:20 | Item1      | 640    |
| 18 | 2001/9/01 | 18:19:20 | Item2      | 225    |
| 19 | 2001/9/01 | 21:10:18 | Item1      | 1260   |

2. The "Long-to-Wide" transformation is the inverse of the above. A selected column is converted ("fanned out") into a set of new columns. The values from one column are converted to a set of new column titles, while the values from another column become their values, aggregated as required.

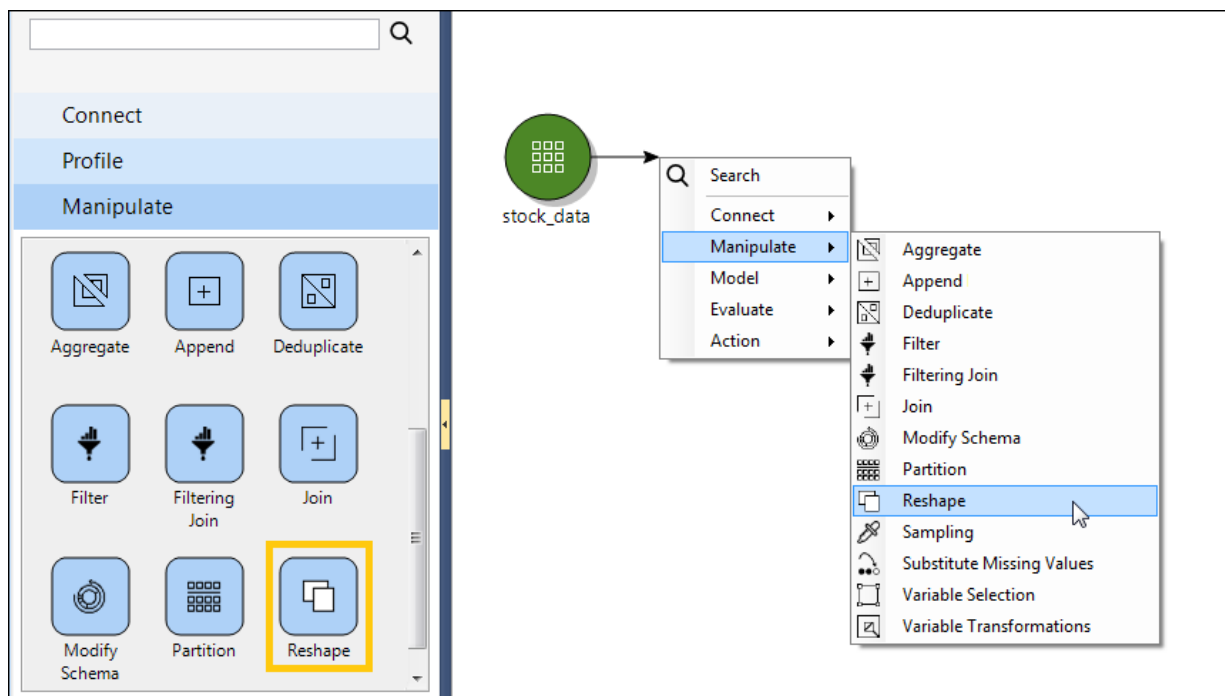
Example: The input dataframe contains the information about stock items (shipped or received), with a single stock item per row. Each row indicates the transaction date / time, the item type, and the amount of this item received or shipped in this transaction. With the "Long-to-Wide" reshape, the Stock\_item column is converted to as many columns as there are item names, and the amount for each item column is calculated by applying an aggregate calculation across the records grouped by the selected key(s).

To reshape a dataframe in any of the two ways listed above:

- Drag a connection arrow from a Spark dataframe node to any area on the workflow canvas background. Releasing the mouse button will bring up the context menu. From the context menu, select **Manipulate --> Reshape** as in the example screenshot below. A new **Reshape** process node will be created.

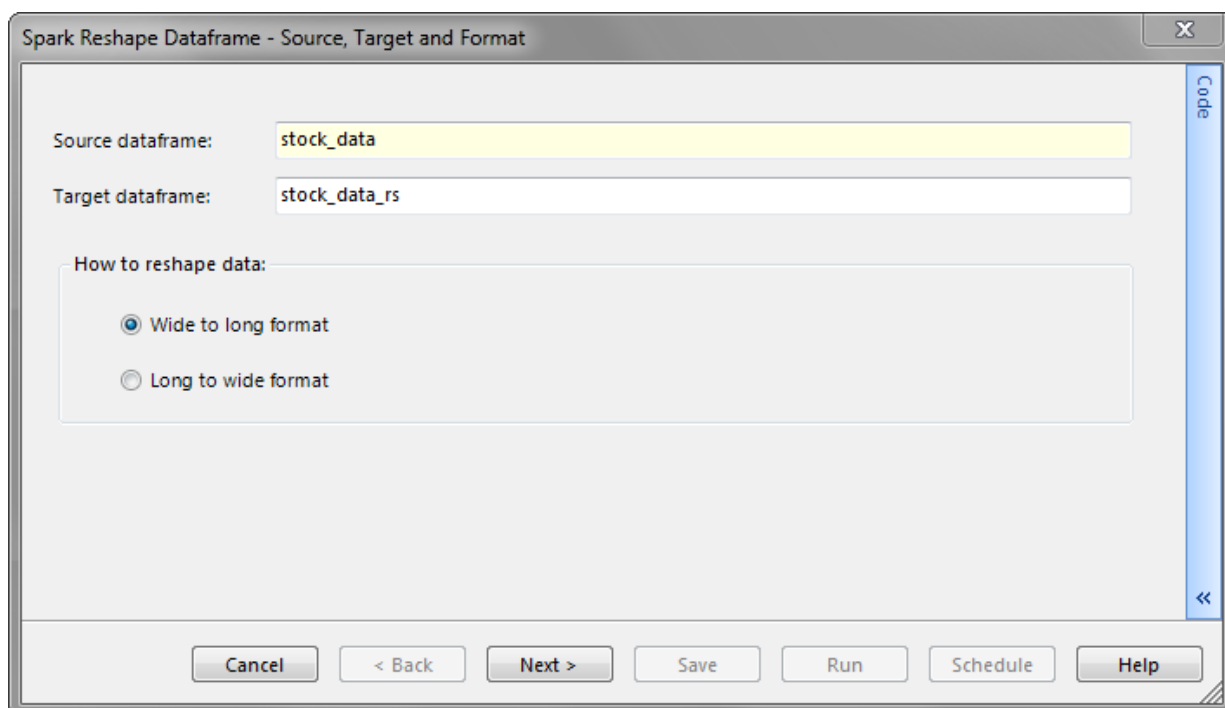
OR:

- Drag the **Reshape** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.



If the wizard does not open automatically, double-click the **Reshape** node or right-click on it and select **Modify** to open the wizard. Specify the options for the operation:

- Specify the name for the output (target) dataframe.
- Choose the transformation type: "Wide to long format" or "Long to wide format". (The former is selected by default.)



The image shows a dialog box titled "Spark Reshape Dataframe - Source, Target and Format". It has a close button (X) in the top right corner. The dialog is divided into three main sections. The first section, "Source dataframe:", has a text field containing "stock\_data". The second section, "Target dataframe:", has a text field containing "stock\_data\_rs". The third section, "How to reshape data:", contains two radio buttons. The first radio button is selected and is labeled "Wide to long format". The second radio button is labeled "Long to wide format". At the bottom of the dialog, there is a row of buttons: "Cancel", "< Back", "Next >", "Save", "Run", "Schedule", and "Help". On the right side of the dialog, there is a vertical blue bar with the word "Code" at the top and a double arrow icon at the bottom.

Click **Next**. The next steps of the wizard depend on the type of transformation selected.

## Option 1: Wide to Long Format

If the **Wide to long format** is selected, several columns are stacked to form a new column containing all values of the original columns. For illustration, we use the following simple example:

|    | Date       | Time     | Item1 | Item2 |
|----|------------|----------|-------|-------|
| 1  | 2001/1/01  | 18:05:09 | 228   | 100   |
| 2  | 2001/1/01  | 19:22:25 | 79    | 79    |
| 3  | 2001/1/01  | 20:22:17 | 808   | 554   |
| 4  | 2001/1/01  | 22:07:19 | 948   | 409   |
| 5  | 2001/3/01  | 21:27:20 | 628   | 596   |
| 6  | 2001/6/01  | 21:21:18 | 215   | 22    |
| 7  | 2001/7/01  | 20:21:10 | 150   | 74    |
| 8  | 2001/8/01  | 20:29:09 | 128   | 5     |
| 9  | 2001/9/01  | 18:19:20 | 640   | 225   |
| 10 | 2001/9/01  | 21:10:18 | 1260  | 1230  |
| 11 | 2001/10/01 | 18:01:27 | 424   | 193   |
| 12 | 2001/10/01 | 19:03:10 | 192   | 127   |
| 13 | 2001/10/01 | 19:10:03 | 584   | 243   |
| 14 | 2001/10/01 | 20:01:28 | 812   | 318   |
| 15 | 2001/10/01 | 20:16:29 | 1428  | 931   |
| 16 | 2001/12/01 | 20:25:19 | 1800  | 1247  |
| 17 | 2002/1/01  | 15:00:16 | 1602  | 1356  |
| 18 | 2002/3/02  | 20:06:09 | 700   | 492   |
| 19 | 2002/3/02  | 20:14:01 | 195   | 186   |

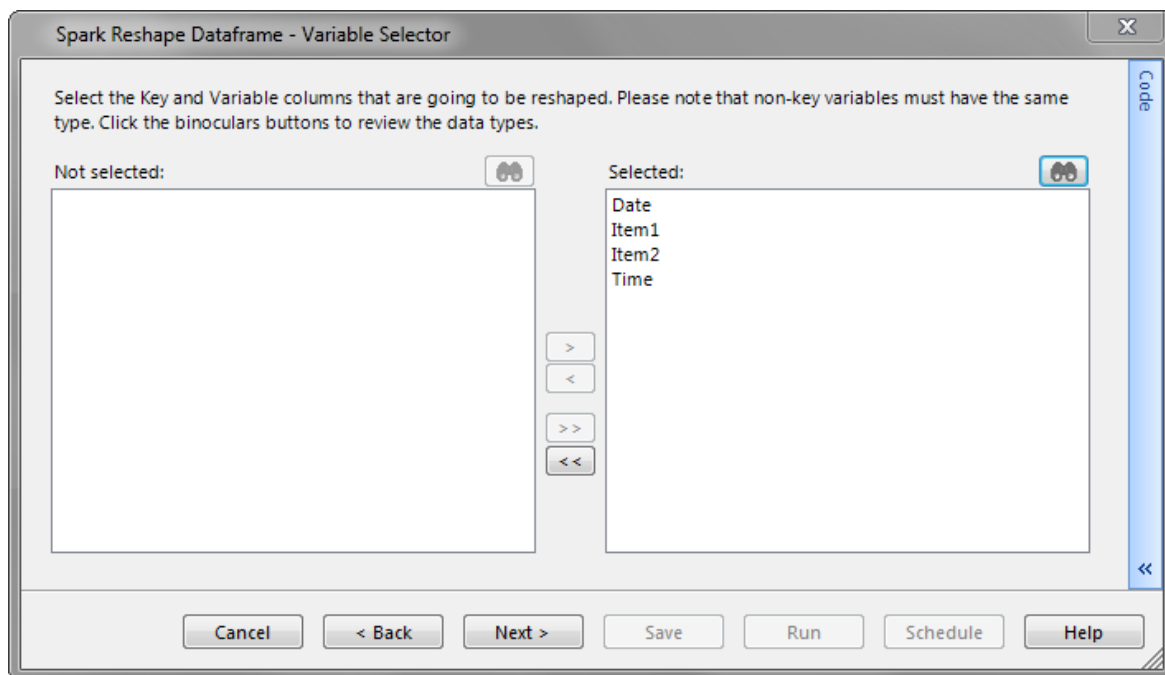
Overview | Dataframe Chart | **Data** | Segment Viewer | Cross Tabs | Correlations | Saved Charts

In the Variable Selector page, select all columns that need to be included in the output dataframe - both the **Key** columns AND the **Variable** columns. The Variable columns are the ones whose values have to be stacked and put in a new column. A minimum of three columns must be selected: at least one as a key column and at least two others to be stacked.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

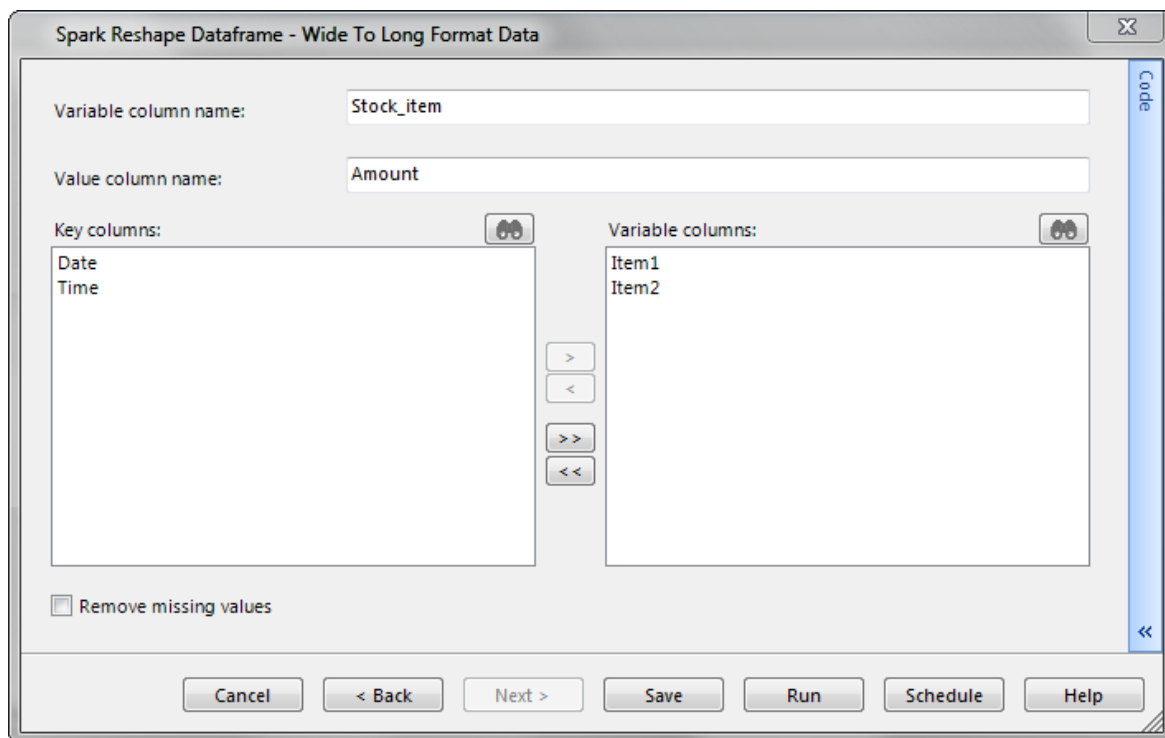
The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.



Click **Next**.

In the Wide to Long Format Data page, specify the following parameters:

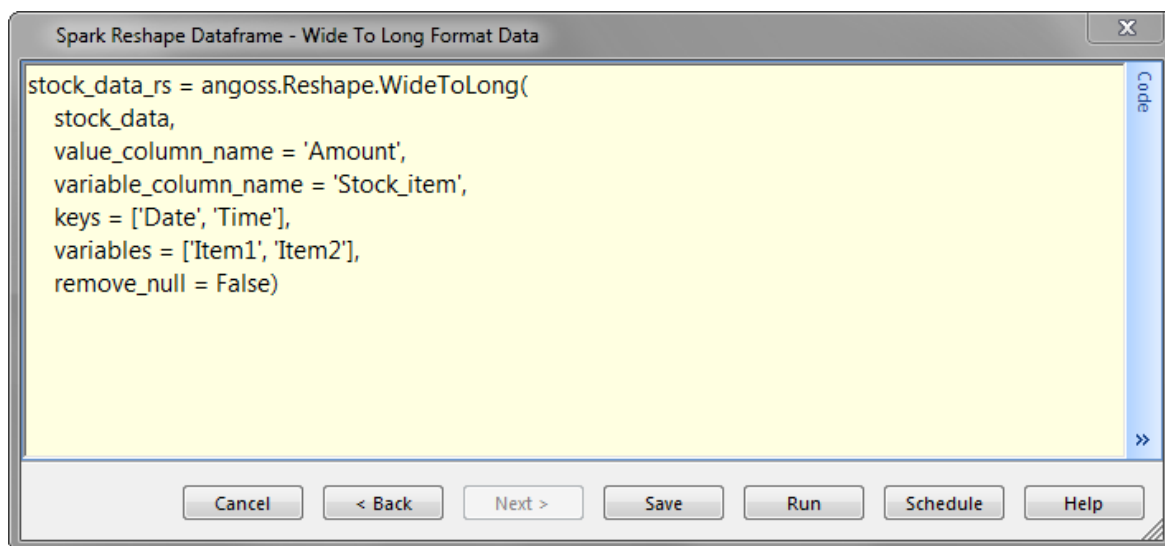
- *Variable column name*: Enter the name for the new column to contain the names of the variables being stacked.
- *Value column name*: Enter the name for the new column to contain the values of the variable columns being stacked.
- *Variable columns*: Select at least two Variable columns to be stacked. In the example below, the values from two columns "Item1" and "Item2" will be stacked in the new column called Amount.  
All Variable columns selected must have the same data type. You can check the data types by clicking the binoculars button. Note that the subtypes should match too (for example, if one column is long integer and another is short integer, the wizard will report a type mismatch).
- *Key columns*: Leave the other columns (to be used as keys) in the *Key columns* list. At least one key must be present.
- *Remove missing values*: If necessary, select this option to drop the records with missing values in the Value column from the output dataframe.



The dialog box is titled "Spark Reshape Dataframe - Wide To Long Format Data". It contains the following fields and controls:

- Variable column name:** A text field containing "Stock\_item".
- Value column name:** A text field containing "Amount".
- Key columns:** A list box containing "Date" and "Time".
- Variable columns:** A list box containing "Item1" and "Item2".
- Remove missing values:** A checkbox that is currently unchecked.
- Navigation buttons:** ">", "<", ">>", and "<<" buttons are located between the Key and Variable columns lists.
- Footer buttons:** "Cancel", "< Back", "Next >", "Save", "Run", "Schedule", and "Help".
- Code bar:** A blue vertical bar on the right side with a double arrow icon at the bottom.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



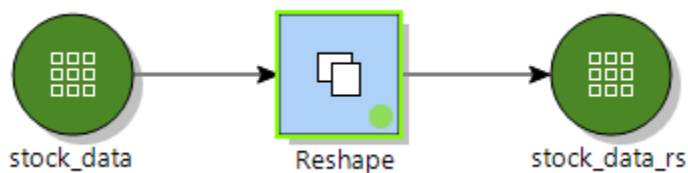
The dialog box is the same as the one above, but the Code bar is expanded to show the following Python code:

```
stock_data_rs = angoss.Reshape.WideToLong(  
    stock_data,  
    value_column_name = 'Amount',  
    variable_column_name = 'Stock_item',  
    keys = ['Date', 'Time'],  
    variables = ['Item1', 'Item2'],  
    remove_null = False)
```

The Code bar now has a double arrow icon at the bottom right.

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

As a result, the node representing the output dataframe appears on the workflow canvas.



Page | 156

Double-click on the output dataframe to view the result and go to the **Data** tab. In this example, for each date and time (key columns), the Stock\_item column shows the item name ("Item1" or "Item2"), and the amount for this item is shown in the Amount column.

|    | Date      | Time     | Stock_item | Amount |
|----|-----------|----------|------------|--------|
| 1  | 2001/1/01 | 18:05:09 | Item1      | 228    |
| 2  | 2001/1/01 | 18:05:09 | Item2      | 100    |
| 3  | 2001/1/01 | 19:22:25 | Item1      | 79     |
| 4  | 2001/1/01 | 19:22:25 | Item2      | 79     |
| 5  | 2001/1/01 | 20:22:17 | Item1      | 808    |
| 6  | 2001/1/01 | 20:22:17 | Item2      | 554    |
| 7  | 2001/1/01 | 22:07:19 | Item1      | 948    |
| 8  | 2001/1/01 | 22:07:19 | Item2      | 409    |
| 9  | 2001/3/01 | 21:27:20 | Item1      | 628    |
| 10 | 2001/3/01 | 21:27:20 | Item2      | 596    |
| 11 | 2001/6/01 | 21:21:18 | Item1      | 215    |
| 12 | 2001/6/01 | 21:21:18 | Item2      | 22     |
| 13 | 2001/7/01 | 20:21:10 | Item1      | 150    |
| 14 | 2001/7/01 | 20:21:10 | Item2      | 74     |
| 15 | 2001/8/01 | 20:29:09 | Item1      | 128    |
| 16 | 2001/8/01 | 20:29:09 | Item2      | 5      |
| 17 | 2001/9/01 | 18:19:20 | Item1      | 640    |
| 18 | 2001/9/01 | 18:19:20 | Item2      | 225    |
| 19 | 2001/9/01 | 21:10:18 | Item1      | 1260   |

Overview | Dataframe Ch... | **Data** | Segment Viewer | Cross Tabs | Correlations | Saved Charts

## Option 2: Long to Wide Format

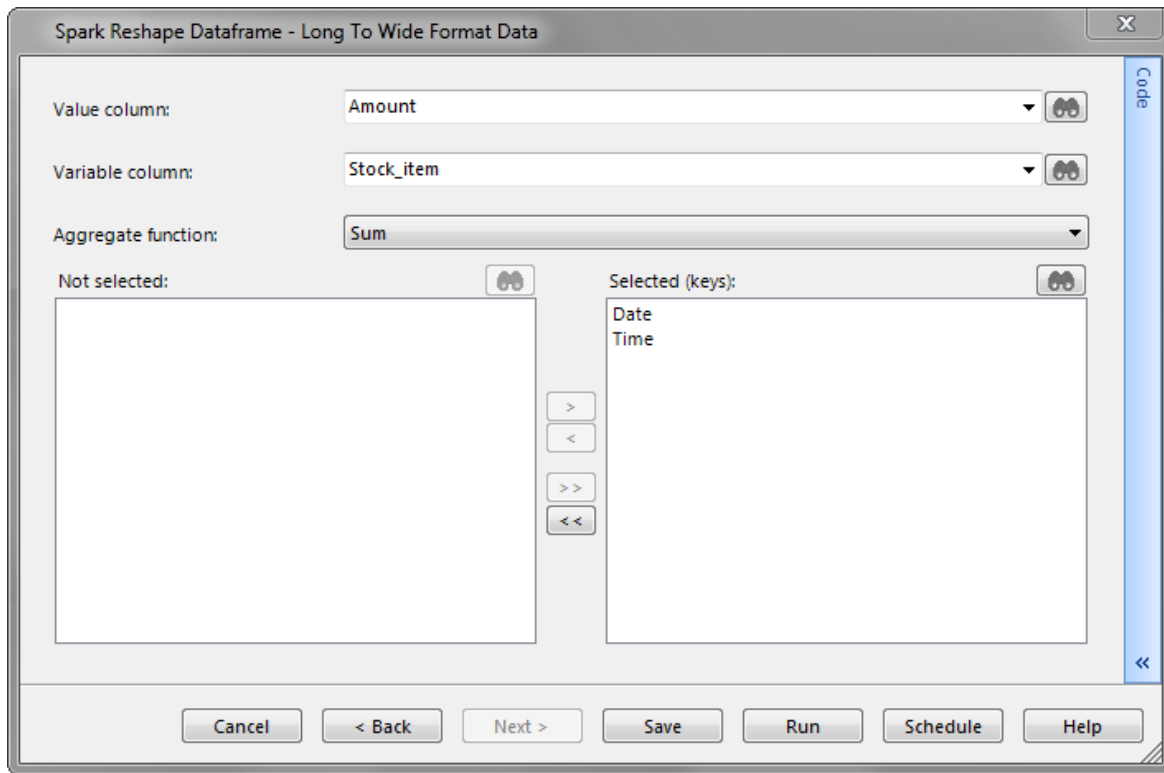


If the **Long to wide format** is selected, a pivot transformation is applied to convert one of the columns into a set of new columns, and the resulting dataframe is aggregated.

In the Long to Wide Format Data page, specify the following parameters:

- *Value column:* Select the column where the values of the new variables should come from.
- *Variable column:* Select the column that contains the names of the new variables to be created. This column should be of String type.
- *Aggregate function:* Select the aggregate function to apply to the values in the columns to be created. Aggregation will be performed over the records grouped by the key column(s).  
If the selected *Value* column is numeric, then the available functions are Count, Sum, Average, Minimum, and Maximum. Otherwise, the available functions are Count, Minimum, and Maximum.
- *Selected key(s):* Select the column(s) to be used as Keys. These will be the columns to group by when aggregating the records in the dataframe.

In this example, for each new Variable column, the total Amount (SUM ('Amount')) will be calculated for the records grouped by Date and Time.



Spark Reshape Dataframe - Long To Wide Format Data

Value column: Amount

Variable column: Stock\_item

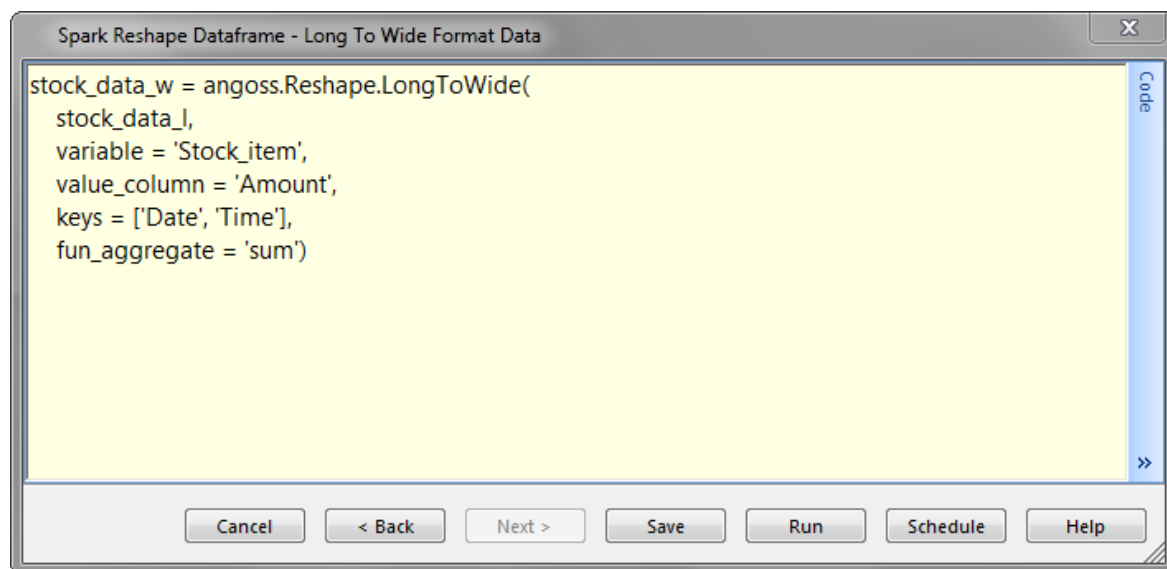
Aggregate function: Sum

Not selected:

Selected (keys):  
Date  
Time

Buttons: Cancel, < Back, Next >, Save, Run, Schedule, Help, Code

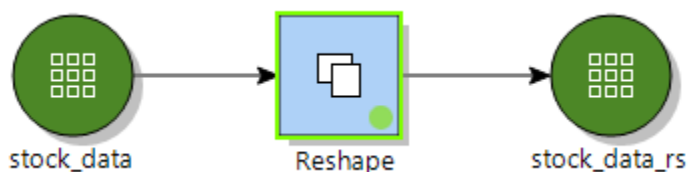
You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Page | 158

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

As a result, the node representing the output dataframe appears on the workflow canvas.



Double-click on the output dataframe and go to the **Data** tab to view the result.

## Sampling

The **Sampling** node in the **Manipulate** palette performs data sampling for a Spark dataframe in the workflow, producing a new dataframe.

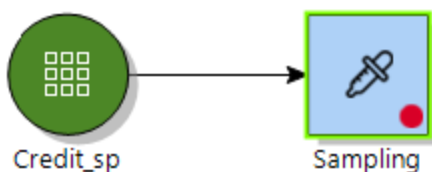
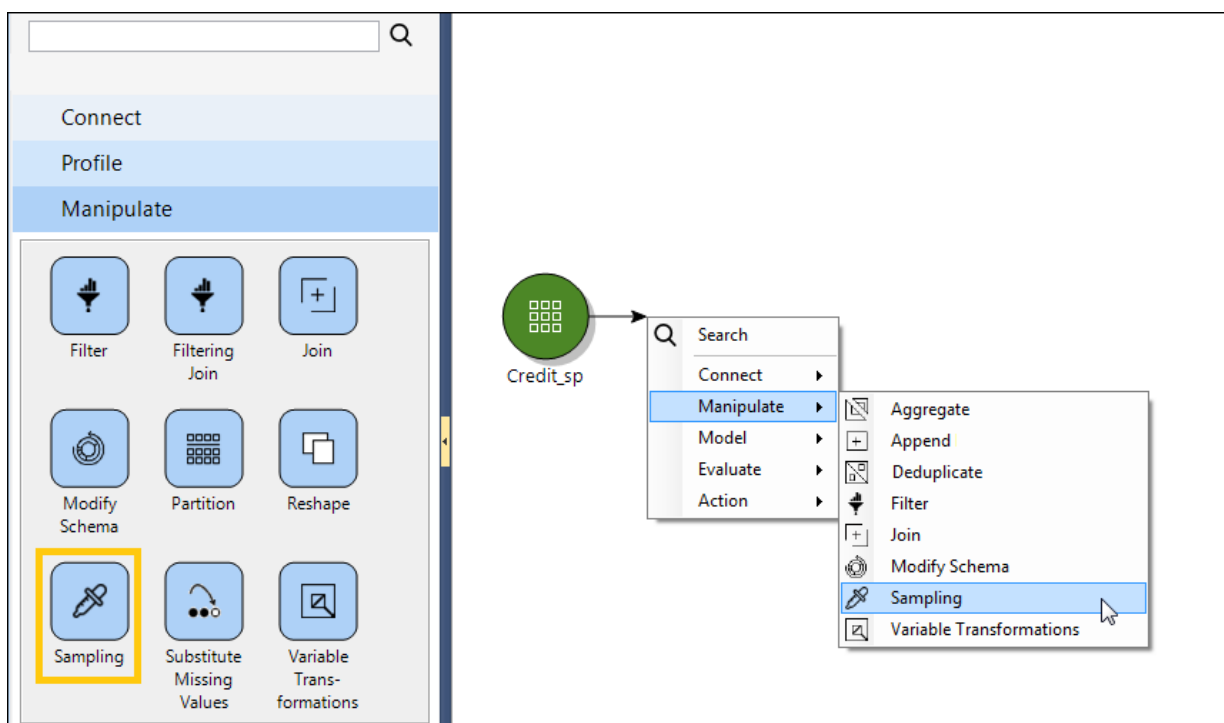
To perform data sampling for a Spark dataframe in your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will appear. From the context menu, select **Manipulate --> Sampling** as in the example screenshot below. A new **Sampling** process node will be created.

OR:

- Drag the **Sampling** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.

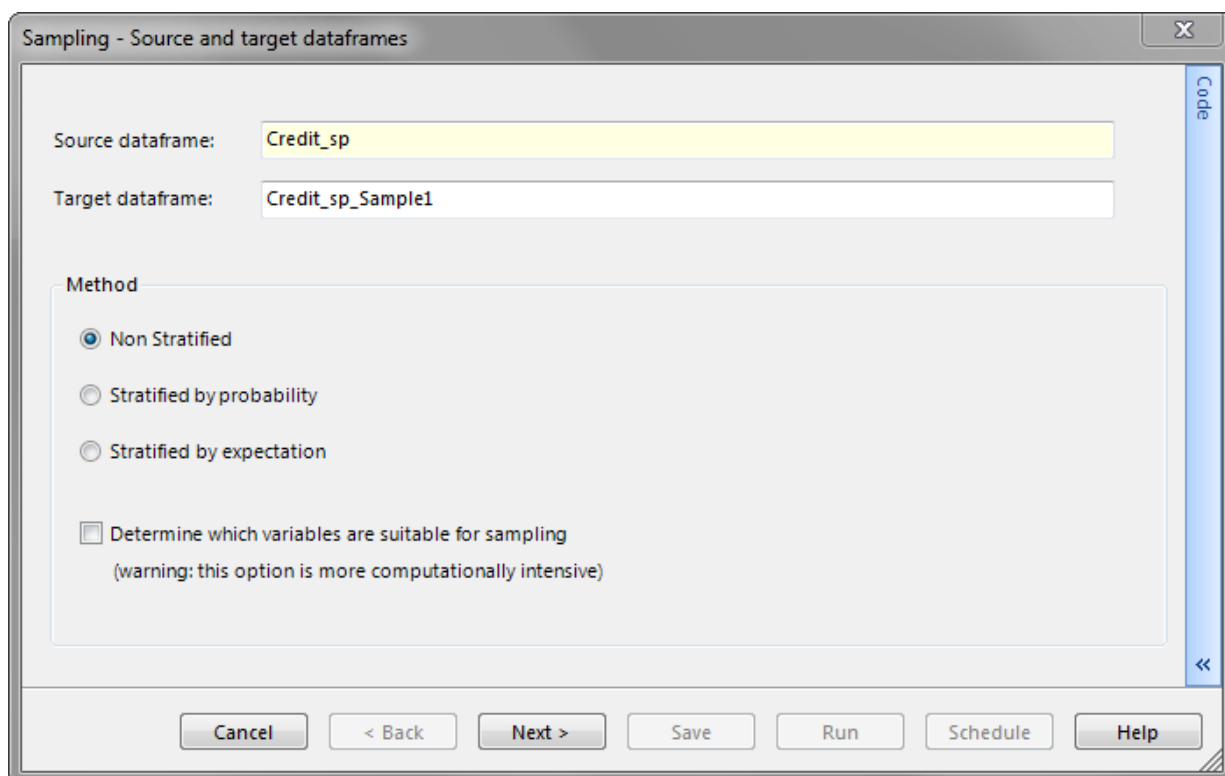
Page | 159



If the wizard does not open automatically, double-click the **Sampling** node or right-click on it and select **Modify** to open the wizard.

- specify the name for the output dataframe
- select the method:
  - Non-stratified (random)
  - Stratified by probability

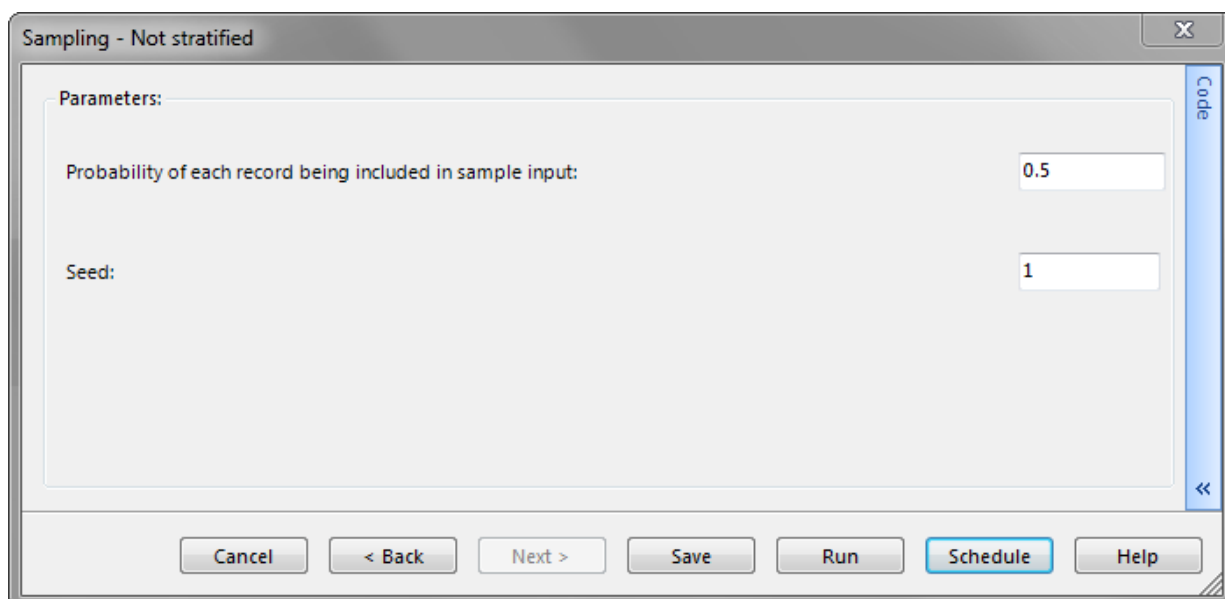
- Stratified by probability
- The stratified option requires selecting a field. Select the option “*Determine which variables are suitable for sampling*”. Note that this may take a long time.



The dialog box titled "Sampling - Source and target dataframes" contains the following elements:

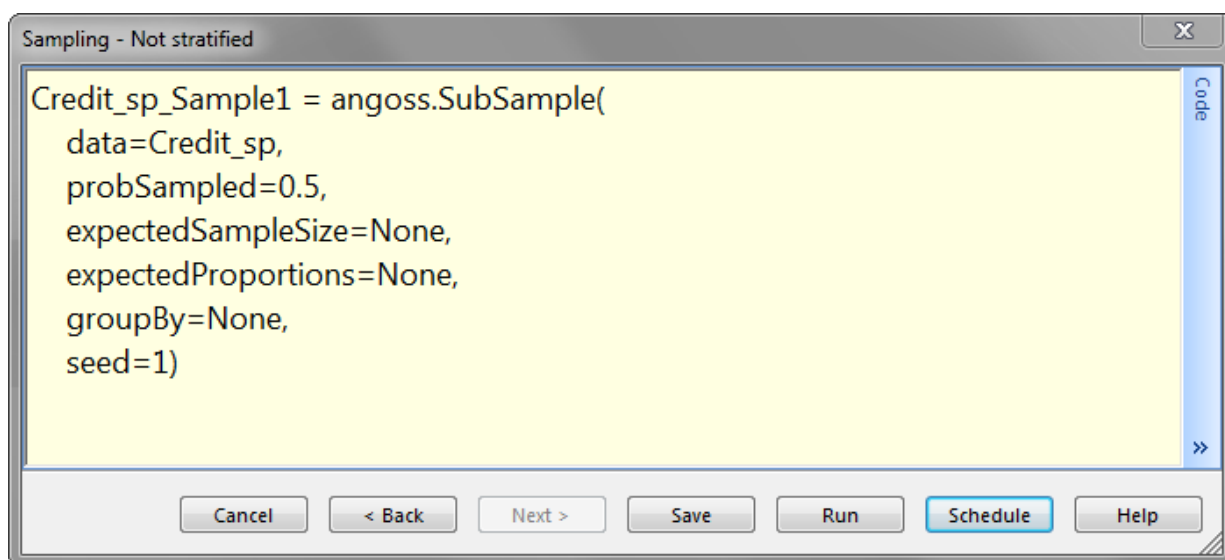
- Source dataframe:** A text field containing "Credit\_sp".
- Target dataframe:** A text field containing "Credit\_sp\_Sample1".
- Method:** A section containing four radio buttons and one checkbox:
  - ☒ Non Stratified
  - ☐ Stratified by probability
  - ☐ Stratified by expectation
  - ☐ Determine which variables are suitable for sampling  
(warning: this option is more computationally intensive)

At the bottom of the dialog are several buttons: "Cancel", "< Back", "Next >", "Save", "Run", "Schedule", and "Help". A vertical "Code" button is on the right side.



The dialog box titled "Sampling - Not stratified" contains a "Parameters:" section with two input fields: "Probability of each record being included in sample input:" with a value of 0.5, and "Seed:" with a value of 1. A blue vertical bar on the right side is labeled "Code" and has double arrows at the bottom. At the bottom of the dialog are buttons for "Cancel", "< Back", "Next >", "Save", "Run", "Schedule" (highlighted in blue), and "Help".

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



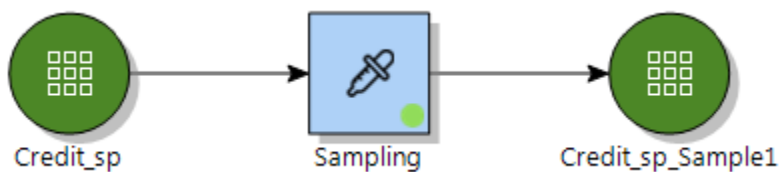
The dialog box titled "Sampling - Not stratified" shows the Python code generated from the parameters. The code is displayed in a yellow-highlighted text area. The code is: 

```
Credit_sp_Sample1 = angoss.SubSample(  
    data=Credit_sp,  
    probSampled=0.5,  
    expectedSampleSize=None,  
    expectedProportions=None,  
    groupBy=None,  
    seed=1)
```

 The blue vertical bar on the right is labeled "Code" and has double arrows at the bottom. The bottom buttons are the same as in the first image, with "Schedule" highlighted.

Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe is created in the workflow after running the node.



## Missing Value Substitution

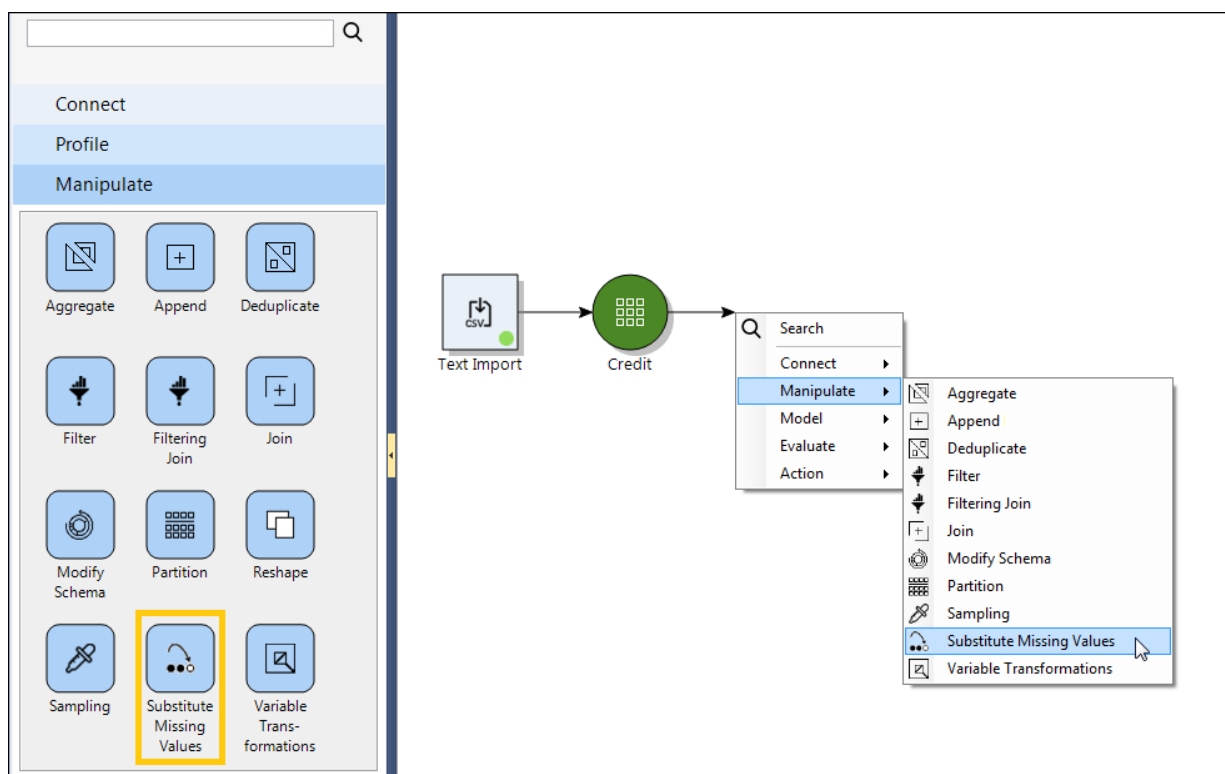
The **Substitute Missing Values** node allows you to replace missing values (nulls) in one or more fields of a Spark dataframe. Missing values in a numeric field are substituted with the mean value of that field. Missing values in String fields can be substituted with arbitrary user-defined string values.

To substitute missing values in one or more fields of a Spark dataframe:

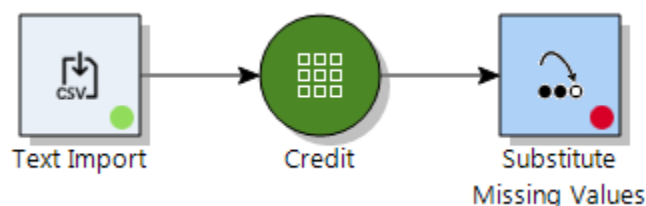
- Drag a connection arrow from the dataframe node to any area on the Workflow canvas. Releasing the mouse button brings up the context menu. From the menu, select **Manipulate --> Substitute Missing Values** as in the example screenshot below. A new **Substitute Missing Values** process node will be created.

OR:

- Drag the **Substitute Missing Values** node from the **Manipulate** palette onto the canvas and connect the input dataframe to the node.



If the wizard does not open automatically, double-click the **Substitute Missing Values** node or right-click on it and select **Modify** to open the wizard.



In the Substitute Missing Values wizard:

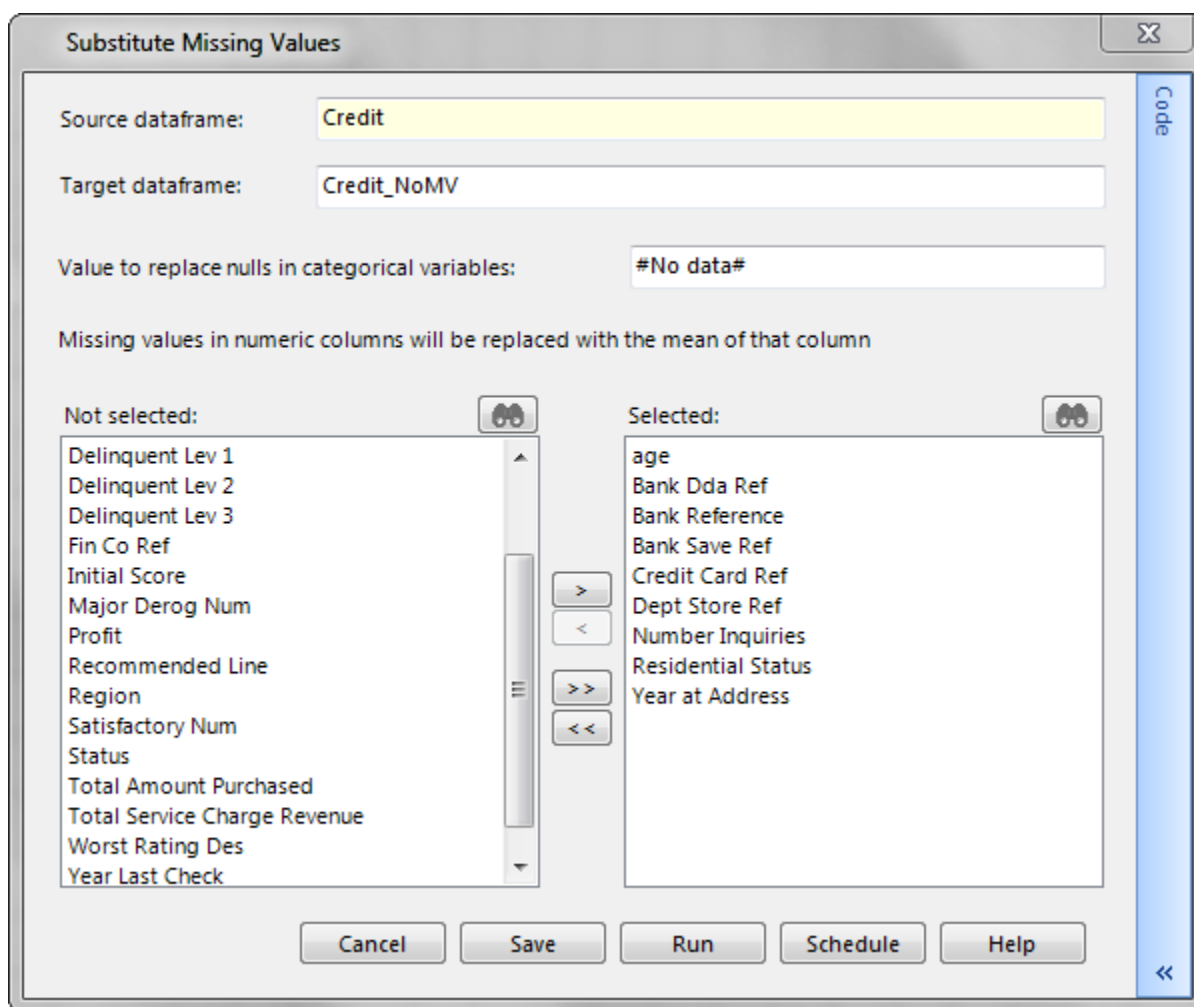
- Specify the name for the output (target) dataframe.
- Select the fields where missing value substitution needs to be performed.
- Specify the string value to replace all missing values in all selected categorical (String) fields.
- Note that missing values in the selected numeric fields will be replaced with the mean value of the respective field.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse

dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.



The dialog box titled "Substitute Missing Values" is shown. It has a close button (X) in the top right corner. The "Source dataframe:" field contains "Credit" and the "Target dataframe:" field contains "Credit\_NoMV". The "Value to replace nulls in categorical variables:" field contains "#No data#". Below this, a text label states: "Missing values in numeric columns will be replaced with the mean of that column". There are two lists of variables: "Not selected:" and "Selected:". The "Not selected:" list contains: Delinquent Lev 1, Delinquent Lev 2, Delinquent Lev 3, Fin Co Ref, Initial Score, Major Derog Num, Profit, Recommended Line, Region, Satisfactory Num, Status, Total Amount Purchased, Total Service Charge Revenue, Worst Rating Des, and Year Last Check. The "Selected:" list contains: age, Bank Dda Ref, Bank Reference, Bank Save Ref, Credit Card Ref, Dept Store Ref, Number Inquiries, Residential Status, and Year at Address. Between the lists are four arrow buttons: ">", "<", ">>", and "<<". At the bottom are five buttons: "Cancel", "Save", "Run", "Schedule", and "Help". On the right side of the dialog is a blue vertical bar labeled "Code" at the top and a double-left arrow "<<" at the bottom.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview. In the example below, missing values in the string fields are replaced with the user-defined string `#No data#`.



Substitute Missing Values X

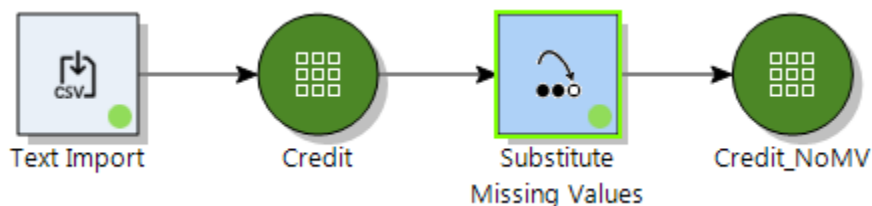
```
Credit_NoMV = angoss.SubstituteMissingValues(
  data = Credit,
  columns = ['age', 'Bank Dda Ref', 'Bank Reference', 'Bank Save Ref', 'Credit Card
Ref', 'Dept Store Ref',
            'Number Inquiries', 'Residential Status', 'Year at Address'],
  value = '#No data#')
```

Code

»

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe is created in the workflow after running the node.



Double-click the output dataframe node and view the Overview Report tab, the Data tab, and/or other dataframe tabs to verify the results.

|    | Initial Score | age | Residential Status |
|----|---------------|-----|--------------------|
| 1  | 463           | 34  | Own Home           |
| 2  | 321           | 39  | #No data#          |
| 3  | 331           | 25  | Other              |
| 4  | 370           | 46  | #No data#          |
| 5  | 399           | 65  | #No data#          |
| 6  | 0             | 57  | Rent               |
| 7  | 492           | 44  | Own Home           |
| 8  | 321           | 39  | #No data#          |
| 9  | 380           | 56  | Rent               |
| 10 | 0             | 22  | Rent               |
| 11 | 471           | 37  | Own Home           |

---

## Variable Transformations

Variable transformations for Spark dataframes are defined in the **Dataframe Editor**, which is provided by the **Variable Transformations** node of the **Manipulate** palette.

Page | 166

Dataframe Editor allows you to define the following data manipulations:

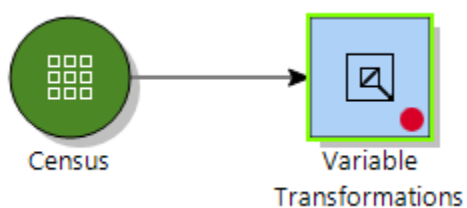
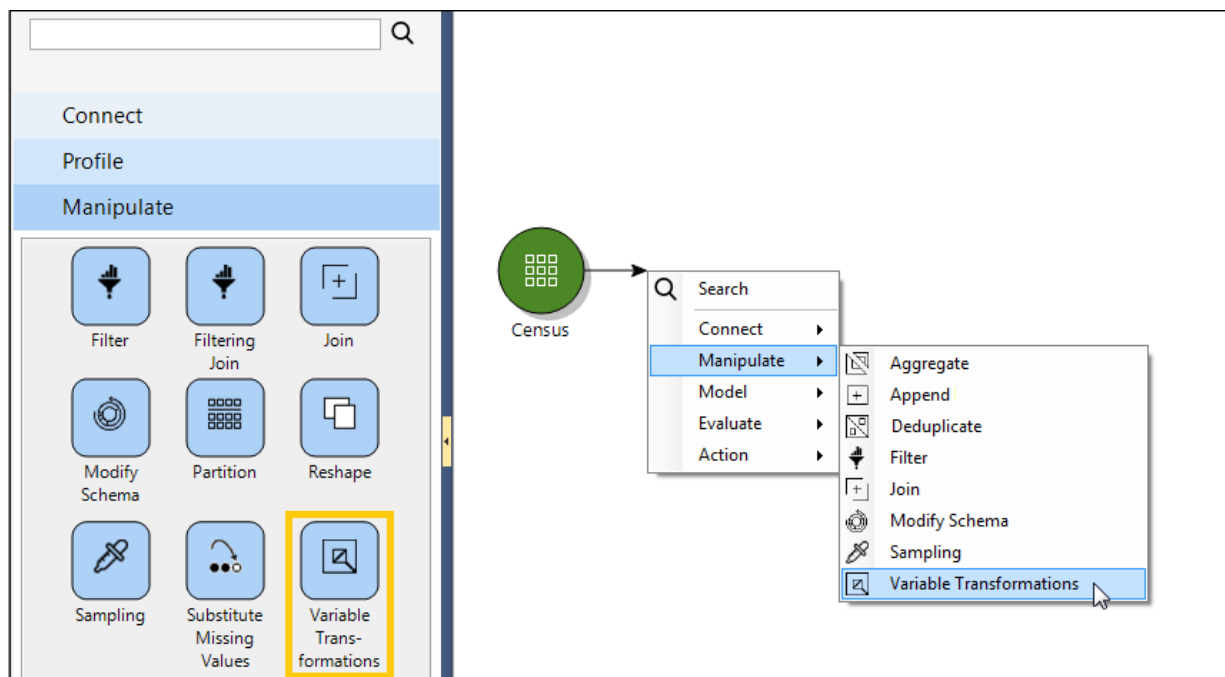
- add new calculated columns (one or multiple at a time)
- delete columns
- edit expressions for the already defined calculated columns and rename them
- specify the display format for any column (such as Currency, Percent, etc.)
- import expressions from other dataframes or expression files
- export expressions

To perform variable transformations for a Spark dataframe in your workflow:

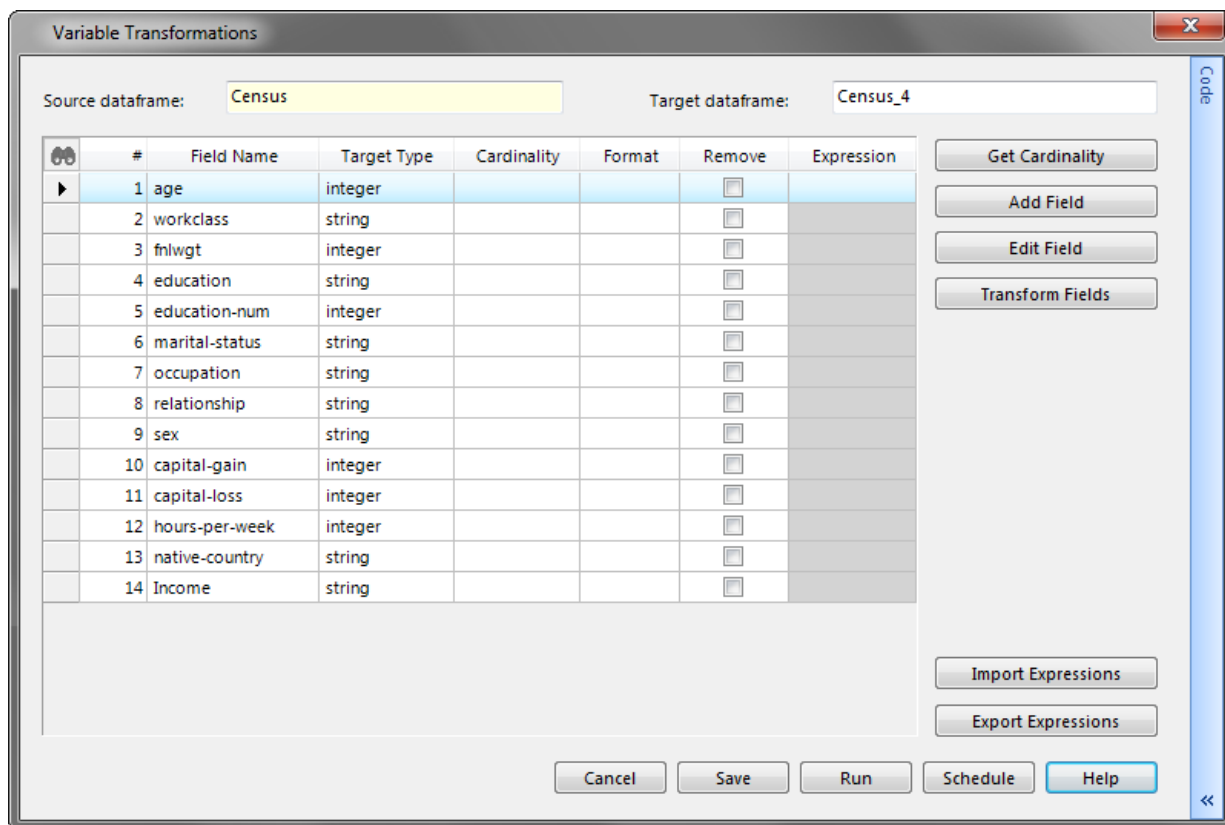
- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will appear. From the context menu, select **Manipulate --> Variable Transformations** as in the example screenshot below. A new **Variable Transformations** process node will be created and its wizard will open.

OR:

- Drag the **Variable Transformations** node from the **Manipulate** palette onto the canvas and connect the input dataframe to it.



If the wizard does not open automatically, double-click the **Aggregate Data** node or right-click on it and select **Modify** to open the wizard.



All columns in this grid are sortable. For example, if you want to remove all String fields, click on the Target Type column header to group fields by type. It will then be convenient to highlight all String fields at once (click and drag the mouse down) and mark them all at once for deletion by placing a checkmark in the Remove column for any one of the highlighted fields.

The [Field Finder](#) button  at the top left corner is useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

### **Field Name**

The name of the field.

This column is read-only. If you wish to change the name of a raw (non-calculated) field, you need to re-import your dataframe and change the field names in the last page of the import wizard (the Field Transformer page). Alternatively, you can add a new field as a copy of an existing raw field and specify the desired name.

### **Target Type**

The data type of the field. Data types include String, Integer, Number, Boolean, Date, Time, and Timestamp.

This column is read-only. If you wish to change the data type of a field, you need to re-import your dataframe and

---

---

|                    |                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | change the data type in the last page of the import wizard (the Field Transformer page).                                                                                                                                                                                                                                                            |
| <b>Cardinality</b> | The number of distinct values in the field. Note that for Spark dataframes, Cardinality is not calculated by default as this can take some time for large volumes of distributed data representing the dataframe. If you need to know the cardinality of a field, select it and click the <b>Get Cardinality</b> button on the right-hand side.     |
| <b>Format</b>      | The display format for the values in a given field. The format can include precision specifier for numeric fields. To specify or modify the format for a field, select the field and click the Edit Field button. For the list of available formats and the ways to specify custom precision, see the topic <a href="#">Specifying Format</a> .     |
| <b>Remove</b>      | Select the variables to remove from the dataframe.                                                                                                                                                                                                                                                                                                  |
| <b>Expression</b>  | Only applies to calculated fields (fields added in the Dataframe Editor using the Expression Editor). Displays the expression specified for the calculated field. The expression can be modified by clicking the Edit Field button and editing the expression in the Expression Editor. See the topic <a href="#">About the Expression Editor</a> . |

## Get Cardinality

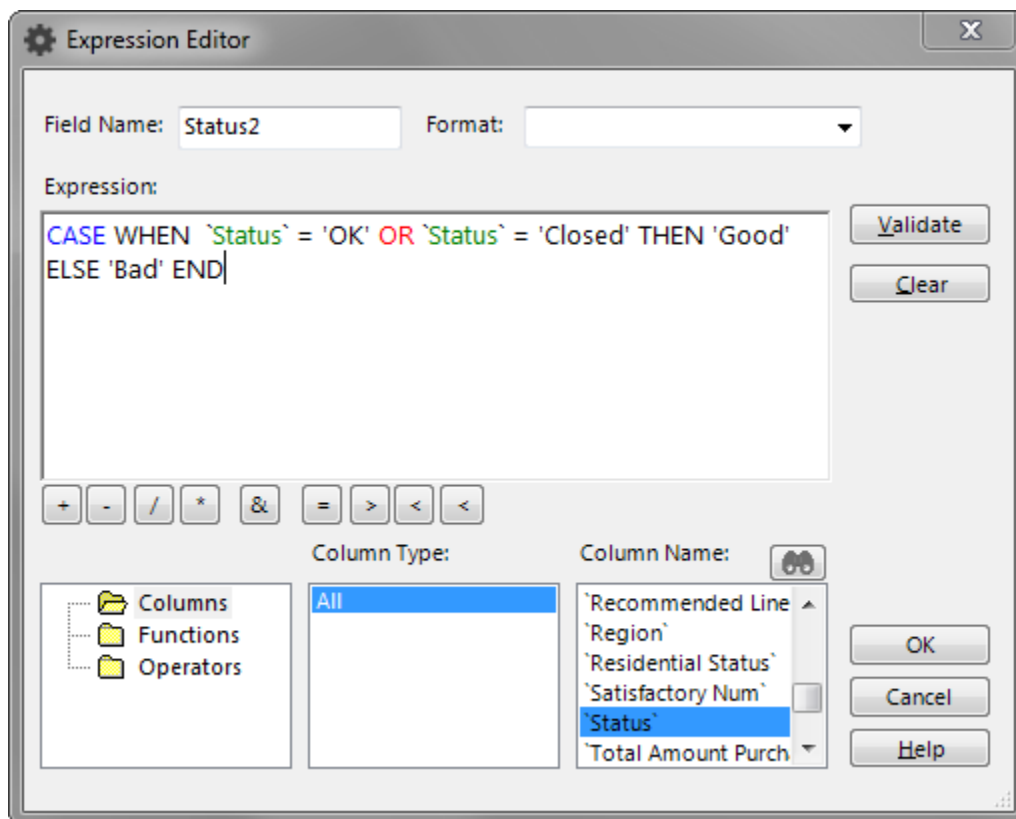
For Spark dataframes, Cardinality is not calculated by default as this can take some time for large volumes of distributed data representing the dataframe. If you need to know the cardinality of a field, select it and click the **Get Cardinality** button on the right-hand side.

## Add Field

To add a new calculated field, click the Add Field button. This will open the Expression Editor window, which allows you to enter a SQL expression or use one of the Helper wizards for most common transformations. See the topic [About the Expression Editor](#).

Note: In the expression syntax for Spark dataframes, field names must be enclosed in the grave accents ( ` ) - usually located to the left of the **1!** key on the keyboard.

Example:



## Edit Field

Allows you to edit the expressions specified for calculated fields (fields created using the Add Field button) using the Expression Editor.

For the original fields of the dataframe, the Edit Field button opens the Format String dialog used to specify the format. For detailed description of format strings, see [Specifying Format](#).

## Transform Fields

Opens an advanced Expression Editor that provides the capability to transform multiple variables at a time: Define an SQL expression and apply it to as many variables as desired. This may be useful in many cases, for example, for standard normalization of multiple numeric variables. The expression specified in the Expression field is applied to all variables in the Selected column at the top right. The variable **[%SelectedField%]** stands for all selected variables. You can enter it quickly at any time by clicking the **Selected field** button. See also the topic [About the Expression Editor](#).

In the example below, for each selected variable **x**, the transformation  $(x - \text{Mean}) / (\text{Standard Deviation})$  will be applied.

Transform Fields

Not Selected:

- Fin Co Ref
- Initial Score
- Major Derog Num
- Number Inquiries
- Recommended Line
- Region
- Residential Status
- Satisfactory Num
- Status
- Status2
- Total Service Charge Revenue
- Worst Rating Des

Selected:

- age
- Avg Monthly Balance
- Amount Lost
- Total Amount Purchased
- Profit

Derived field names extension text:  ☐ Force data to be instantiated

Extension is: ☒ Prefix ☐ Suffix

Format:

Expression:

Operator Category:

Operator:

For detailed description of format strings, see [Specifying Format](#).

## Import Expressions

Expressions for calculated fields in other dataframes can be imported to be applied to any field in the current dataframe, provided that all the dependent fields occurring in the expression are present in the current dataframe.

Expressions can also be imported from expression files (in XML format) created using the Export Expressions command.

## Export Expressions

---

Expressions for calculated fields in the current dataframe can be exported to an external XML file that can be used when adding or editing calculated fields in other dataframes (using the Import Expressions command).

Click **Save** to save your definitions and changes. Click **Run** if you want to save and execute them right away.

## Expression Editor

The Expression Editor allows you to build SQL expressions when creating new fields in your dataframe. These new fields are referred to as *calculated columns*.

Note that Strategy Trees have their own Expression Editor for node calculations. It is described in the topic [Expression Editor for Node Calculations](#) of the Strategy Tree Help section.

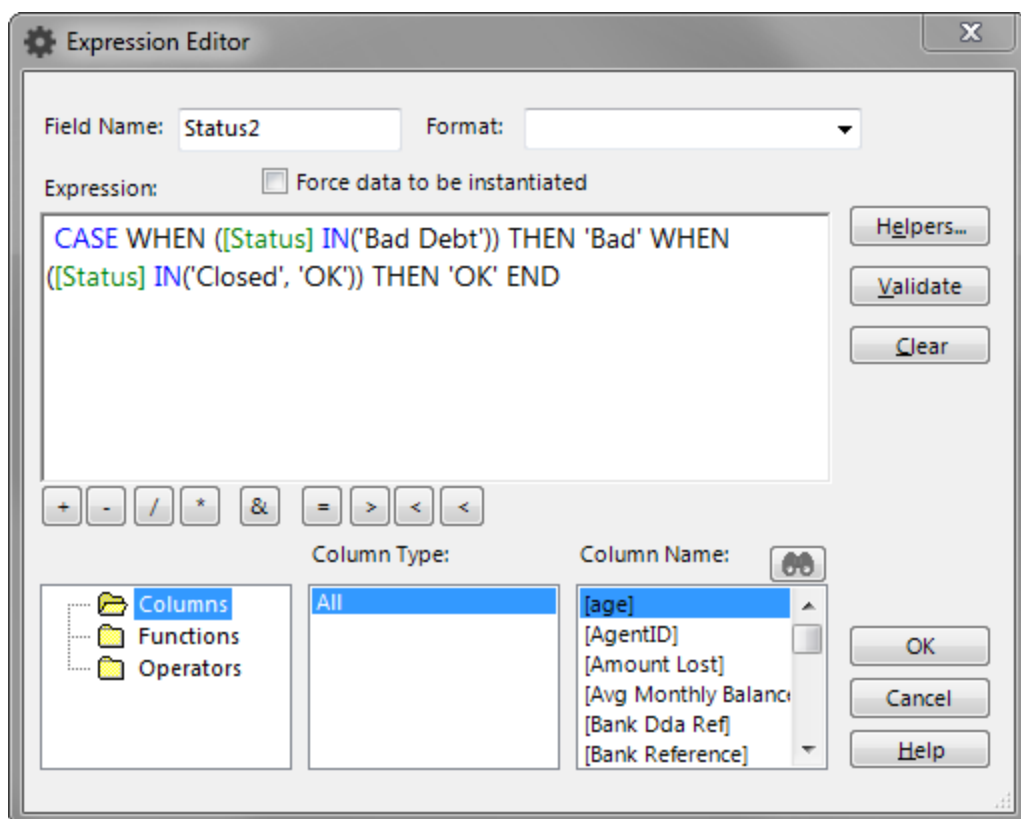
Some of the common types of transformations that can be performed with the help of Expression Editor are:

- mapping the categories of a field to new categories
- merging several categories into one
- creating a discrete representation (binning) of a continuous variable
- performing calculations on existing numeric fields to express the results in a new field
- replacing missing values with a custom value (user-specified or calculated)

The Expression Editor can be opened from the Dataset Editor dialog in the [Variable Transformations](#) node wizard by clicking the **Add Field** or **Edit Field** button.

The Expression Editor uses SQL expressions to create new columns in the dataframe.





**Field Name**  
**Expression**

The name for the new field (variable).

The expression that defines the new variable. You can manually type an expression in the Transact SQL syntax and the Columns, Functions, and Operators reference in the bottom part of the dialog. Note that variable names should be enclosed in square brackets. For reference on the function syntax, click on the Functions folder and then click the Syntax Help link.

**Format**

The display format of the resulting value. If left blank, the application default is assumed. You can set formats such as Currency, Percent, Scientific, and many more. See the topic [Specifying Format](#) for details.

**Force data to be instantiated**

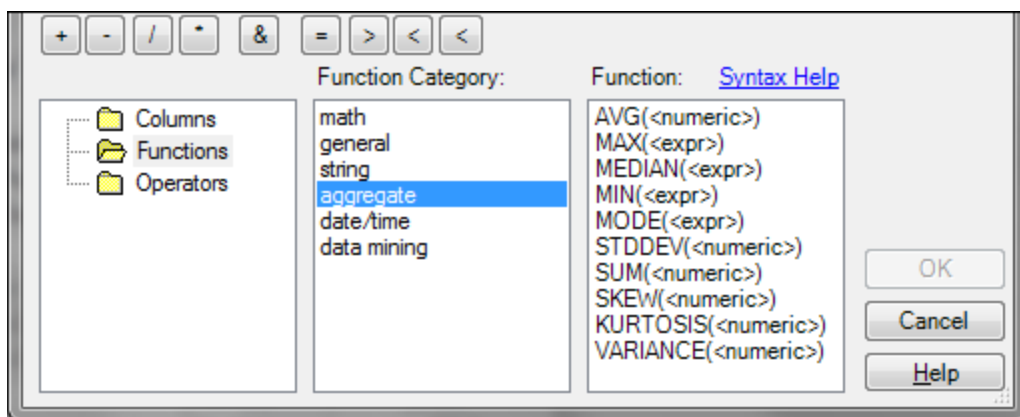
If this option is selected, the expression that defines the values of the new column will be removed in the new dataframe (the new column will be a "raw" column with no expression). This option is similar to the Paste Values option of the Paste function in Microsoft Excel, which only pastes values (without the function that defines them).

## Validate

The Validate button validates the syntax of the expression you entered. The syntax of expressions is based on the standard SQL syntax. There are some additional syntax rules for referring to objects and projects. See also the [Function Syntax](#) topic.

Page | 174

## Columns, Functions and Operators



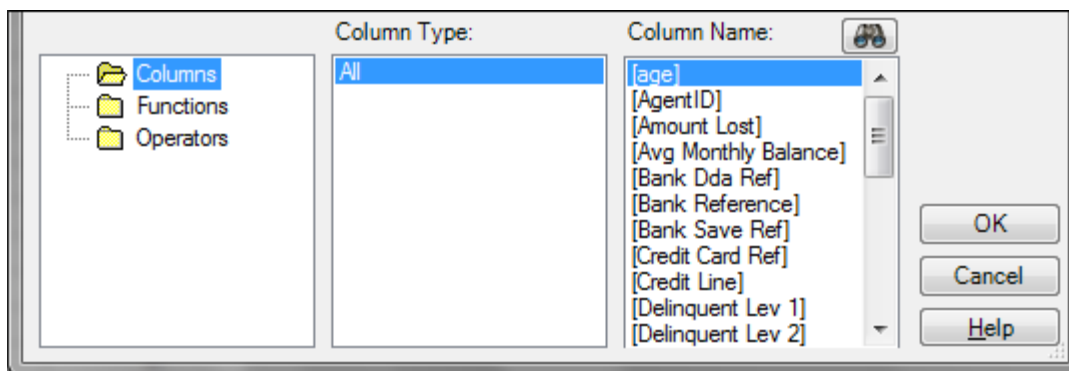
The Columns, Functions, and Operators folders provide help in building your expressions. The Columns list displays the names of all the columns in the dataframe. The Functions list contains standard functions for SQL expressions, various mathematical, string, and date/time functions. The Operators list contains arithmetic, logical, and bitwise operators.

For special notes about calculated columns in the In-Database Analytics Mode, see the [Variable Transformations](#) section.

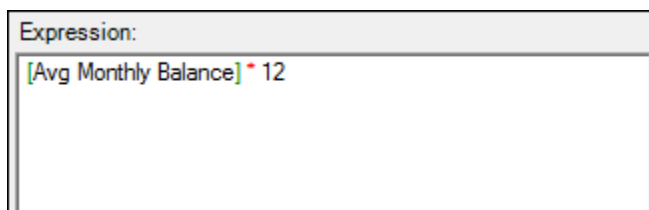
See also: [Function Syntax](#), [Specifying Format](#).

## Columns

Click on the **Columns** folder to open the list of all columns in the dataset.

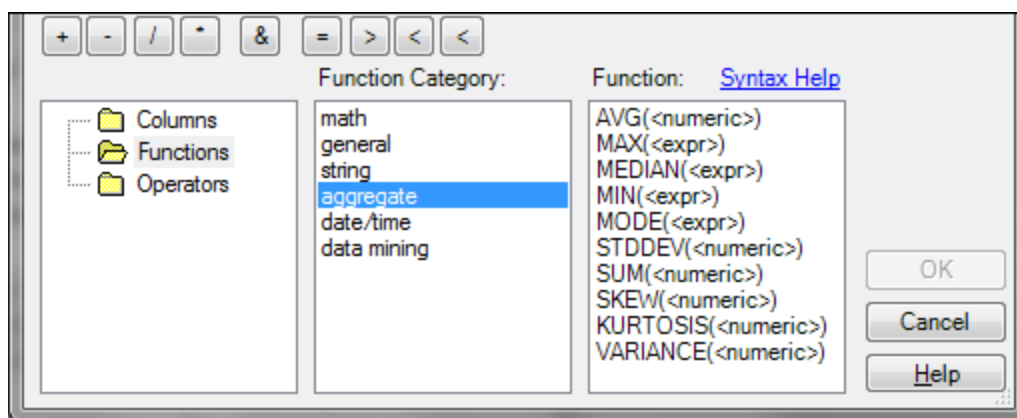


If you want to use a certain column in the calculation being defined, double-click on the column name and it will appear in the expression box. Expressions can always be typed manually into the expression box. Note that column names must be enclosed in the square brackets [ ].



## Functions

The **Functions** folder contains most commonly used mathematical, string, date/time, data mining, and other SQL functions. The **Function** box lists the functions of each category. The type of their arguments is indicated in parentheses. You can click the **Syntax Help** link above the Function box to see the syntax and examples for any function. For information and examples on these functions, see the [Function Syntax](#) topic.



Double-click on any function to make it appear in the expression box.

Expression:

STDDEV(<numeric>)

Replace the type in the parentheses with a value or a column name. A column name can be entered in the expression box by selecting the Columns folder and double-clicking the desired column name. The type of the entered value or column must be compatible with the type of the function argument. For example:

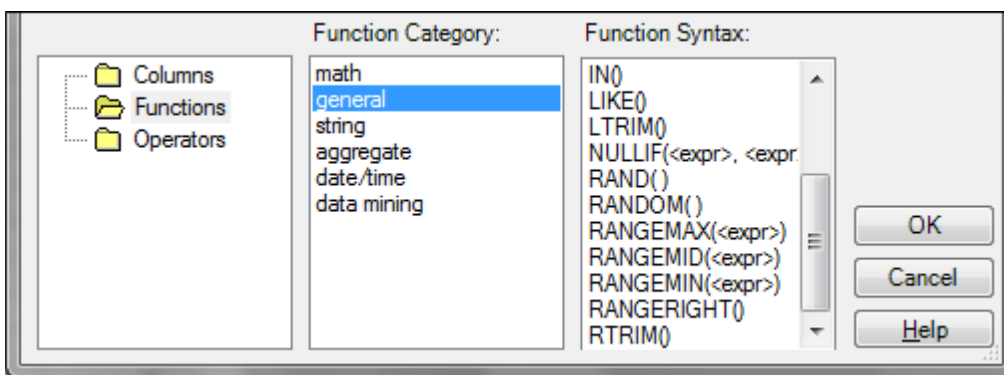
Expression:

STDDEV([Profit])

where [Profit] is a numeric column in the dataset.

As with all expressions, these functions can be typed manually into the expression box.

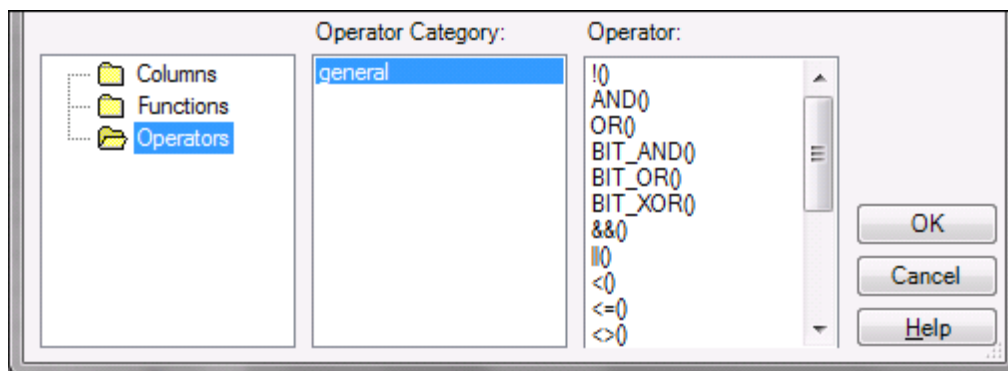
Other function types are:



Aggregate expressions can be used in the Expression Editor at the row level. For example, [age]/avg([age]). Note that aggregate expressions are converted to constants when the formula is copied into a model. For example, [age]/avg([age]) becomes [age]/35.56 when the expression is embedded into a model.

## Operators

The operators list contains most common operators used in SQL Queries. They include addition, subtraction, multiplication and division operators, comparison operators (<,>=,>,<=, etc.), logical operators AND, OR, NOT, etc., and bitwise operators.



See also:

[Function Syntax](#)

[Specifying Format](#)

## Specifying Format

You can specify various formats (currency, percent, etc.) and precision for displaying your data. The format you specify in Dataset Editor or Expression Editor affects all views of the dataframe (except the Data tab and the Correlations tab, since they show unformatted data) and the objects depending on the dataset, such as decision trees.

- To change the format for an existing field, connect your dataset to a new [Variable Transformations](#) node, select the desired field and click the **Edit Field** button.
- If you are adding a calculated field, you can specify the format string for it in the Expression Editor dialog.

You can choose one of the standard formats from the Format String drop-down list or enter a custom format in the Format String field.

a) Standard numeric format strings

| Format String       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Currency (C)</b> | Represents a currency amount. The currency symbol and the default precision are defined by the settings in the Regional Options of your Windows system (see <b>Control Panel   Regional and Language Options</b> ). To specify the Currency format with custom precision, you can type <code>Cnn</code> in the Format String field, where <code>nn</code> is an optional integer called the precision specifier, which ranges from 0 to 99 and controls the number of significant digits or zeros to the right of a decimal. The format string cannot contain white spaces. |

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Percent (P)</b>    | Represents a percent. If the number is negative, the string starts with a minus sign. The converted number is multiplied by 100 in order to be presented as a percentage. You can specify the Percent format with custom precision by typing <code>P<sub>nn</sub></code> in the Format String field, where the precision specifier <code>nn</code> is an integer that indicates the desired number of decimal places.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Scientific (S)</b> | Represents a string of the form <code>"-d.ddd...E+ddd"</code> or <code>"-d.ddd...e+ddd"</code> , where each 'd' indicates a digit (0-9). A minus sign is added if the number is negative. One digit always precedes the decimal point. The precision specifier indicates the desired number of digits after the decimal point. If the precision specifier is omitted, a default of six digits after the decimal point is used. The case of the format specifier indicates whether to prefix the exponent with an 'E' or an 'e'. The exponent always consists of a plus or minus sign and a minimum of three digits. The exponent is padded with zeros to meet this minimum, if required.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Number (N)</b>     | Represents a string of the form <code>"-d,ddd,ddd.ddd..."</code> , where each 'd' indicates a digit (0-9). A minus sign is added if the number is negative. Thousand separators are inserted between each group of three digits to the left of the decimal point. The default precision is defined by the settings in the Regional Options of your Windows system (see <b>Control Panel   Regional and Language Options</b> ). You can specify the Number format with custom precision by typing <code>N<sub>nn</sub></code> in the Format String field, where the precision specifier <code>nn</code> is an integer that indicates the desired number of decimal places.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>General (G, g)</b> | <p>Represents the most compact of either fixed-point or scientific notation, depending on the type of the number and whether a precision specifier is present. You can specify the custom General format by typing <code>G<sub>nn</sub></code> or <code>g<sub>nn</sub></code> in the Format String field, where <code>xx</code> is the precision specifier.</p> <p>Fixed-point notation is used if the exponent that would result from expressing the number in scientific notation is greater than -5 and less than the precision specifier; otherwise, scientific notation is used. The result contains a decimal point if required and trailing zeroes are omitted. If the precision specifier is present and the number of significant digits in the result exceeds the specified precision, then the excess trailing digits are removed by rounding. If scientific notation is used, the exponent in the result is prefixed with 'E' if the format specifier is 'G', or 'e' if the format specifier is 'g'.</p> <p>The exception to the preceding rule is if the number is a Decimal and the precision specifier is omitted. In that case, fixed-point notation is always used and trailing zeroes are preserved.</p> |
| <b>Decimal (D)</b>    | Applicable for the types Integer and String. Represents a string of decimal digits (0-9), prefixed by a minus sign if the number is negative. You can specify the custom Decimal format by typing <code>D<sub>nn</sub></code> in the Format String field, where the precision specifier <code>nn</code> is an integer that indicates the minimum                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | number of digits desired in the resulting string. If required, the number is padded with zeros to its left to produce the number of digits given by the precision specifier. For example, with Format String specified as D10, the number 156 will be represented as 0000000156.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Fixed-Point (F)</b>    | Represents a string of the form "-ddd.ddd..." where each 'd' indicates a digit (0-9). The string starts with a minus sign if the number is negative. The default precision is defined by the settings in the Regional Options of your Windows system (see <b>Control Panel   Regional and Language Options</b> ). You can specify the Fixed-point format with custom precision by typing Fnn in the Format String field, where the precision specifier nn is an integer that indicates the desired number of decimal places. For example, with the Format String specified as F05, the number 5.26 will be represented as 5.26000.                                                                                                                               |
| <b>Round-trip</b>         | Applicable to fields of the Number type only. The round-trip specifier guarantees that a numeric value converted to a string will be parsed back into the same numeric value. When a numeric value is formatted using this specifier, it is first tested using the general format, with 15 spaces of precision. If the value is successfully parsed back to the same numeric value, it is formatted using the general format specifier. However, if the value is not successfully parsed back to the same numeric value, then the value is formatted using 17 digits of precision.                                                                                                                                                                               |
| <b>Hexadecimal (X, x)</b> | Applicable to fields of the Integer type only. Represents a string of hexadecimal digits. If, instead of choosing 'Hexadecimal' from the drop-down list, you enter the format specifier 'X' or 'x', note that the case of the letter indicates whether to use uppercase or lowercase characters for the hexadecimal digits greater than 9. For example, use 'X' to produce "ABCDEF", and 'x' to produce "abcdef". You can specify the custom Hexadecimal format by typing Xnn in the Format String field, where the precision specifier nn is an integer that indicates the minimum number of digits desired in the resulting string. If required, the number is padded with zeros to its left to produce the number of digits given by the precision specifier. |

**Note:** A more detailed description of standard numeric format strings is available in the MSDN Library at <https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings>.

#### b) Custom numeric format strings

A simple way to specify custom precision is to enter a custom numeric format string of the form 00.0000... or ##.#####..... For example, with the format string 000.0000, the number 56.73 will be displayed as 056.7300. The format string ####.#### also increases the precision to four decimal places, but does not pad the number with zeroes, so the number 56.73 is displayed in the same form (56.73). Both format strings 0.0 and #.# round 56.73 to 56.7.

Here is a brief description of custom numeric format strings:

| Format String         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 (zero placeholder)  | If the value being formatted has a digit in the position where the '0' appears in the format string, then that digit is copied to the result string. The position of the leftmost '0' before the decimal point and the rightmost '0' after the decimal point determines the range of digits that are always present in the result string. The "00" specifier causes the value to be rounded to the nearest digit preceding the decimal, where rounding away from zero is always used.                                                                                                                            |
| # (digit placeholder) | If the value being formatted has a digit in the position where the '#' appears in the format string, then that digit is copied to the result string. Otherwise, nothing is stored in that position in the result string. Note that this specifier never displays the '0' character if it is not a significant digit, even if '0' is the only digit in the string. It will display the '0' character if it is a significant digit in the number being displayed. The "###" format string causes the value to be rounded to the nearest digit preceding the decimal, where rounding away from zero is always used. |
| . (decimal point)     | The first '.' character in the format string determines the location of the decimal separator in the formatted value; any additional '.' characters are ignored. The actual character used as the decimal separator is determined by the settings in the Regional Options of your Windows system (see <b>Control Panel   Regional and Language Options</b> ).                                                                                                                                                                                                                                                    |

**Note:** A more detailed description of custom numeric format strings is given in <https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-numeric-format-strings>.

#### c) Standard DateTime format strings

- Short Date
- Long Date
- Short Time
- Long Time
- Sortable Date/Time
- Universal Date/Time

**Note:** A more detailed description of standard DateTime format strings can be found in the MSDN Library at <https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-date-and-time-format-strings>.

A more detailed description of custom DateTime format strings is given in <https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings>.



## Function Syntax

### Introduction

Page | 181

This topic describes the functions that can be used in the Expression Editor for adding calculated columns in datasets and node calculations in Strategy Trees. The list of all supported functions with their syntax and related examples is given below.

Functions take inputs (*arguments*) and produce the result of a certain type. For example, the mathematical function [SQRT](#) produces the square root of its numeric argument, e.g.,  $\text{SQRT}(25)=5.0$ . Some functions have no arguments, such as [RAND](#), which generates a random number, or [CDATE](#), which produces the current date.

When used in an expression, a function name is followed by its arguments in round brackets. Functions that require no arguments must still be followed by round brackets, for example, `RAND()` and `CDATE()`. The argument can be a value, a column name, or an expression of the *data type* that is valid for this function. Otherwise the result will be *null*. For example, the argument of the function `SQRT` must be a nonnegative number. Column names in expressions must be enclosed in square brackets.

All String functions and operators are case-sensitive with respect to their arguments.

See the topic [Expression Editor: Functions and Operators](#) for more general information about functions and operators.

### Arguments

| TYPE       | DESCRIPTION                                                                                                                                                                                                       |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| char       | Character in single quotes. Examples: 's', '@', '2'.                                                                                                                                                              |
| expression | Expression constructed from arguments of other types. For example, the addition of two columns. Note: This isn't a specific data type.                                                                            |
| float      | Floating point number. Examples: 1.4, 32.54, .002.                                                                                                                                                                |
| integer    | Integer number. Examples: 1, 38, -189, 0.                                                                                                                                                                         |
| numeric    | Any type of number.                                                                                                                                                                                               |
| string     | String: one or more characters in single quotes. Some examples: 's', '2', and 'word'.                                                                                                                             |
| date       | Date formatted according to the current system settings, for example, DD/MM/YYYY. Date constants used as function arguments must be enclosed in curly brackets, for example: <code>DAYOFYEAR({2012-03-31})</code> |
| timestamp  | Date and time formatted according to the current system settings, for example, DD/MM/YYYY hh:mm:ss. Timestamp constants used as                                                                                   |

|     |                                                                                                                         |
|-----|-------------------------------------------------------------------------------------------------------------------------|
|     | function arguments must be enclosed in curly brackets, for example:<br>DAYOFYEAR({2012-03-31 17:45:51})                 |
| ... | Variable number of arguments. Some functions operate on a list of items where the number of items is not predetermined. |

**Notation:** Arguments in angle brackets <> must be included. Arguments in square brackets [ ] are optional. These brackets are only used to show whether the argument is required and should **not** be included as part of the argument. Square brackets [ ] are used to indicate a column name.

Page | 182

When more than one argument is required, the arguments are separated by a comma. If a function has no arguments, empty brackets must be used after its name (for example, CDATE()). This does not apply to statement keywords such as CASE, WHEN, THEN, ELSE, and END.

### A Note on Aggregate Functions

Aggregate functions can be used in the Expression Editor at the row level. Note that every aggregate function on Spark requires the OVER() clause after the function name.

For example: `[age]/AVG([age]) OVER()`

Aggregate expressions are converted to constants when the formula is copied into a model. For example, if `AVG([age]) = 35.56`, then the expression `[age]/AVG([age]) OVER()` becomes `[age]/35.56` when embedded into a model.

## The Functions

### ABS

Math Function

**Syntax:** `ABS (<float>)`

ABS returns the absolute value (or modulus) of the **argument**. This has the effect of converting negative numbers to positive values. Zero, and numbers greater than zero, are not affected. Thus, both `ABS(32)` and `ABS(-32)` return 32.

For example, if values of Profit in your dataset can be positive or negative (loss), and we want to flag all customers associated with high profit or loss, we could create a True/False field with an expression similar to the following:

```
CASE WHEN ABS([Profit]) >= 5000 THEN true ELSE false END
```

---

---

## ACOS

### Math Function

**Syntax:** ACOS (<float>)

This trigonometric function is the inverse of [COS](#). It returns the arc cosine of the **argument**. It returns *null* if the argument is outside of the interval [-1, 1].

Page | 183

Some examples:

ACOS(1) = 0

ACOS(1.0001) = null

ACOS(0) = 1.5707963267949

---

## ACOSH

### Math Function

**Syntax:** ACOSH (<float>)

The ACOSH function is the inverse of [COSH](#). It returns the inverse hyperbolic cosine of the **argument**. It returns *null* if the argument is less than 1.

Some examples:

ACOSH(35) = 3.91162276521459

ACOSH(0.2) = null

ACOSH(6) = 2.47788873028848

---

## ARGMAX

### Math Function

**Syntax:** ARGMAX (<numeric>, [<numeric>,.....,<numeric>])

ARGMAX takes one or more numeric arguments and returns the index of the largest argument (i.e., its position in the list of arguments, counting from the left).

Some examples:

ARGMAX(0.25, 0.5, 0.33) = 2

---

---

`ARGMAX(2, -106) = 1`

## ARGMIN

Math Function

Page | 184

**Syntax:** ARGMIN (<numeric>, [<numeric>,...,<numeric>])

ARGMIN takes one or more numeric arguments and returns the index of the smallest argument (i.e., its position in the list of arguments, counting from the left).

Some examples:

`ARGMIN(0.25, 0.5, 0.33) = 1`

`ARGMIN(2, -106) = 2`

## ASCII

Math Function

**Syntax:** ASCII (<char>)

This returns the ASCII code value of the leftmost character of the *argument*. The character set of the argument string is assumed to be ASCII. The function returns 0 if the argument is an empty string, and null if the argument is null.

Some examples:

`ASCII('a') = 97`

`ASCII('ab') = 97`

`ASCII("") = 0`

The inverse of ASCII is [CHR](#). The description of the [CHR](#) function in this Help topic contains a table listing the ASCII code values.

## ASIN

Math Function

**Syntax:** ASIN (<float>)

ASIN is the inverse of [SIN](#). It returns the arc sine of the **argument**. It returns *null* if the argument is outside of the interval [-1, 1].

Some examples:

`ASIN(0.2) = 0.201357920790331`

`ASIN(1.2) = null`

Page | 185

## ASINH

Math Function

**Syntax:** ASINH (<float>)

The ASINH function is the inverse of [SINH](#). It returns the inverse hyperbolic sine of the **argument**.

Some examples:

`ASINH(35.0) = 4.24869926123636`

`ASINH(0.2) = 0.198690110349241`

## ATAN

Math Function

**Syntax:** ATAN (<float>)

ATAN is the inverse of [TAN](#). It returns the arc tangent of the **argument**.

Some examples:

`ATAN(2.0) = 1.10714871779409`

`ATAN(-2,2) = -0.785398163397448`

---

Note: This function can take an optional second argument: `ATAN(<float> [,float])`, in which case it is equivalent to [ATAN2](#).

## ATAN2

### Math Function

**Syntax:** `ATAN2 (<float>, <float>)`

ATAN2 returns the arc tangent of the two **arguments**. It is similar to calculating the arc tangent of  $y/x$ , except that the signs of both arguments are used to determine the quadrant of the result.

Some examples:

`ATAN2(-2,2) = -0.785398163397448`

`ATAN2(3.14159,0) = 1.57079632679489`

## ATANH

### Math Function

**Syntax:** `ATANH (<float>)`

ATANH is the inverse of TANH. It returns the inverse hyperbolic tangent of its numeric argument. The argument must be within the interval  $(-1, 1)$ .

`ATANH(-0.1) = -0.10034`

## AVG

### Aggregate Function

**Syntax:** `AVG (<expression>) OVER()`

AVG is an aggregate function that returns the average value of its argument. The argument must be a numeric column or an expression that involves only numeric columns. For example, if the argument is a numeric column, this is the average value of this column across all records of the dataset (if used for a calculated column in a dataset) or the current node (if used for a node calculation in a [Strategy Tree](#)).

Page | 187

For example, to get the average age for all customers in your dataset, you might use:

```
AVG( [Age] ) OVER()
```

The following example calculates the relative profit (profit normalized by the average profit value).

```
[Profit] / AVG( [Profit] ) OVER()
```

## CASE

### General Function

**Syntax:** CASE WHEN <expression> THEN <expression> [WHEN <expression> THEN <expression>] [ELSE <expression>] END

CASE is a conditional statement providing if-then-else logic (flow control). Used with the keywords WHEN, THEN, ELSE, and END, the CASE statement opens a structure that allows you to branch off to different operations based on specific conditions.

The following example returns the string 'Non-Voter' when the record represents either a person under the age of 18 or a person who is not a citizen. In all other cases, 'Voter' is returned.

```
CASE
  WHEN [Age] < 18 THEN
    'Non-Voter'
  WHEN [LegalStatus] <> 'Citizen' THEN
    'Non-Voter'
  ELSE
    'Voter'
END
```

Indenting the statements is not required but it helps keep the logic visual especially when there are more than one WHEN statement.

The keywords CASE, WHEN, THEN, ELSE, and END do not require parentheses.

As usual, for any String comparisons, the operators are case-sensitive with respect to their String arguments.

See also [IN](#) and [LIKE](#).

## CAST

### General Function

**Syntax:** CAST (<expression> AS <type>)

CAST converts a value from one *type* to another *type* (e.g., the number 123 to the string '123', or vice versa). The argument for CAST is a combination of a value and the desired type with the keyword AS between them.

For example, to convert the value in the [ProductNumber] field from a number to a string:

```
CAST( [ProductNumber] AS STRING )
```

To add the characters EU- to the front of the product number:

```
CONCAT( 'EU-', CAST( [ProductNumber] AS STRING ) )
```

Valid types include: *char*, *float*, *integer*, *numeric*, *string*, and *timestamp*.

## CDATE

### Date/Time Function

**Syntax:** CDATE()

The CDATE() function returns the current date and time as a value in *MM/DD/YYYY hh:mm:ss* format.

For example, CDATE() might return:

```
CDATE() = 01/27/2013 7:42:49 AM
```

See also: [CURRENT\\_DATE](#), [CTIME](#)

## CDATETIME

### Date/Time Function



**Syntax:** CDATETIME()

This is an alternative to [CDATE](#).

Example:

CDATETIME() = 01/27/2013 7:42:49 AM

## CEIL

Math Function

**Syntax:** CEIL(<float>)

CEIL returns the smallest integer value not less than the *argument*. That is, the value is rounded up to the nearest integer that is greater or equal to this value.

Some examples:

CEIL(1.23) = 2

CEIL(-1.23) = -1

This is the opposite of [FLOOR](#).

See also: [CEILING](#)

## CEILING

Math Function

**Syntax:** CEILING(<float>)

This is an alternative to [CEIL](#).

Example:

CEILING(4.16) = 5

## CHARINDEX

String Function

**Syntax:** CHARINDEX(<string>,<string>)

The CHARINDEX function returns the position of the first string argument within the second string argument. If there is more than one occurrence, CHARINDEX returns the position of the first occurrence. If the first string argument is not found within the second one, CHARINDEX returns 0.

Page | 190

Examples:

```
CHARINDEX('I', 'toolbox') = 4
```

```
CHARINDEX('o', 'toolbox') = 2
```

```
CHARINDEX('box', 'toolbox') = 5
```

```
CHARINDEX('toolbox', 'box') = 0
```

See also: [POSSTR](#)

## CHARACTER LENGTH

String Function

**Syntax:** CHARACTER LENGTH(<string>)

This function returns the number of characters in the string **argument** - spaces are included. If the argument is an empty string, 0 is returned.

Some examples:

```
CHARACTER LENGTH('To') = 2
```

```
CHARACTER LENGTH('To be') = 5
```

The following example chops the name in the [LastName] field if it is greater than 25 characters long. First it checks the length of the [LastName] field: if it is greater than 25, it truncates the name using the SUBSTRING function; if it is not, it simply returns the name.

CASE

WHEN CHARACTER LENGTH( [LastName] ) > 25 THEN

SUBSTRING( [LastName] , 1, 25)

ELSE

[LastName]

END

Note that this does not change the value in the [LastName] field but rather places the value in the field created with the Expression Editor.

See also: [LEN](#)

## CHR

### General Function

**Syntax:** CHR(<integer>, ..., <integer>)

The CHR function returns a string consisting of the characters given by the ASCII code values of the integers in the **arguments**. Note that you can use a variable number of arguments but they must all be integers. There are 95 printable ASCII characters, numbered 32 to 126. The first character is a space.

|     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|     | 32  | 33  | 34  | 35  | 36  | 37  | 38  | 39  | 40  |
|     |     | !   | "   | #   | \$  | %   | &   | '   | (   |
| 41  | 42  | 43  | 44  | 45  | 46  | 47  | 48  | 49  | 50  |
| )   | *   | +   | ,   | -   | .   | /   | 0   | 1   | 2   |
| 51  | 52  | 53  | 54  | 55  | 56  | 57  | 58  | 59  | 60  |
| 3   | 4   | 5   | 6   | 7   | 8   | 9   | :   | ;   | <   |
| 61  | 62  | 63  | 64  | 65  | 66  | 67  | 68  | 69  | 70  |
| =   | >   | ?   | @   | A   | B   | C   | D   | E   | F   |
| 71  | 72  | 73  | 74  | 75  | 76  | 77  | 78  | 79  | 80  |
| G   | H   | I   | J   | K   | L   | M   | N   | O   | P   |
| 81  | 82  | 83  | 84  | 85  | 86  | 87  | 88  | 89  | 90  |
| Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |
| 91  | 92  | 93  | 94  | 95  | 96  | 97  | 98  | 99  | 100 |
| [   | \   | ]   | ^   | _   | `   | a   | b   | c   | d   |
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 |
| e   | f   | g   | h   | i   | j   | k   | l   | m   | n   |
| 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 |
| o   | p   | q   | r   | s   | t   | u   | v   | w   | x   |
| 121 | 122 | 123 | 124 | 125 | 126 |     |     |     |     |
| y   | z   | {   |     | }   | ~   |     |     |     |     |

Some examples:

CHR(80,77,78) = PMN

CHR(64) = @

CHR(73,32,97,109) = I am

The inverse of CHR is [ASCII](#).

---

---

## CONCAT

### String Function

Page | 192

**Syntax:** CONCAT(<string>,<string>)

This returns the string that results from concatenating the **arguments**. You must pass two string arguments.

For example:

```
CONCAT('E', 'mail') = Email
```

In the following example a space is first added to the end of the value in the [FirstName] field, then [LastName] is added to that:

```
CONCAT( CONCAT([FirstName], ' '), [LastName] )
```

## COS

### Math Function

**Syntax:** COS(<float>)

The COS function returns the cosine of the **argument**, where the argument is given in radians.

Some examples:

```
COS(3.14159) = -1
```

```
COS(45) = 0.52532198881773
```

## COSH

### Math Function

**Syntax:** COSH(<float>)

The COSH function returns the hyperbolic cosine of the **argument**, where the argument is given in radians.

Some examples:

`COSH(3.14159) = 11.5919226299454`

`COSH(0) = 1`

Page | 193

## COUNT

### General Function

**Syntax:** COUNT(<expression>)

COUNT (\*)

The COUNT function returns the number of records in the dataset (if used in an expression for a calculated column in the dataset) or the number of records in the node (if used in a [node calculation](#) in a Strategy Tree).

The argument expression type must be integer. You can also use the wildcard symbol \* for the function to return the total number of input records, including duplicates and those with null values.

The following example calculates the percentage of males in the dataset. Note that it uses the CAST function to convert the COUNT result from integer to float in order for the entire result to be a floating point number rather than an integer:

```
CAST(COUNT(CASE WHEN [Gender] = 'Male' THEN 1 ELSE null END) AS double) / CAST(COUNT(*) AS double)
```

## COVARIANCE

### General Function

**Syntax:** COVARIANCE (<expression>, <expression>)

The argument expression must involve only numeric columns.

Covariance is the measure of how much two variables vary together (as distinct from variance, which measures how much a single variable varies). If two variables tend to vary together (that is, when one of them is above its expected value, then the other variable tends to be above its expected value too), then the covariance between the two variables will be positive.

On the other hand, if when one of them is above its expected value, the other variable tends to be below its expected value, then the covariance between the two variables will be negative.

Page | 194

If X and Y are independent, then their covariance is zero.

The converse, however, is not true: if X and Y have covariance zero, they need not be independent.

The units of measurement of the covariance(X, Y) are those of X times those of Y. By contrast, correlation, which depends on the covariance, is a dimensionless measure of linear dependence.

The following example calculates the covariance between the variables *hours-per-week* and *capital-gain*:

```
COVARIANCE([hours-per-week],[capital-gain])
```

## CTIME

### Date/Time Function

**Syntax:** CTIME()

CTIME returns the current time as a value in seconds. In a time field, it will appear in an *hh:mm:ss* format.

These examples are all from 5:07:42 PM:

```
CTIME() = 61662
```

The following are values that would appear within a time field:

```
CTIME() = 17:07:42
```

```
CTIME() + 3600 = 18:07:42
```

See also: [CURRENT\\_TIME](#)

## CTIMESTAMP

### Date/Time Function

**Syntax:** CTIMESTAMP()

This is an alternative to [CDATETIME](#)

Example:

CTIMESTAMP() = 01/27/2013 07:42:49

See also: [CDATE](#), [CTIME](#)

## CTZNAME

Date/Time Function

**Syntax:** CTZNAME()

CTZNAME returns the system time zone name.

For example:

CTZNAME() = Eastern Standard Time

## CURRENT\_DATE

Date/Time Function

**Syntax:** CURRENT\_DATE()

This is an alternative to [CDATE](#).

Example:

CURRENT\_DATE() = 01/27/2013 7:42:49 AM

## CURRENT\_TIME

Date/Time Function

**Syntax:** CURRENT\_TIME()

This is an alternative to [CTIME](#).

Examples:

```
CURRENT_TIME() = 06:55:42
```

```
CURRENT_TIME() + 3600 = 07:55:42
```

## CURRENT\_TIMESTAMP

### Date/Time Function

**Syntax:** CURRENT\_TIMESTAMP()

This is an alternative to [CTIMESTAMP](#) and [CDATETIME](#).

Example:

```
CURRENT_TIMESTAMP() = 01/27/2013 06:55:42
```

## CURRENT\_USER

### General Function

**Syntax:** CURRENT\_USER()

The current user is only applicable in multiple user situations. Otherwise, it returns a (null) string since there is no user name.

```
CURRENT_USER()
```

## DATEADD

### Date/Time Function

**Syntax:** DATEADD(<string>,<integer>,<timestamp>)

The DATEADD function adds time to a timestamp value.

- The first **argument** specifies the type of time value to add - valid types are the strings: 'SECOND', 'MINUTE', 'HOUR', 'DAY', 'MONTH', and 'YEAR'.
- The second **argument** specifies the number of units to add. Negative numbers subtract the time.
- The third **argument** is a timestamp that specifies the starting time.

DATEADD returns a timestamp as a value in *MM/DD/YYYY hh:mm:ss* format.

In the following examples, the 0 starting time argument represents *1/1/2000 12:00:00 AM*:



```
DATEADD( 'SECOND', 45, 0 ) = 1/1/2000 12:00:45 AM
```

```
DATEADD( 'MINUTE', 22, 0 ) = 1/1/2000 12:22:00 AM
```

```
DATEADD( 'HOUR', 48, 0 ) = 1/3/2000 12:00:00 AM
```

```
DATEADD( 'DAY', 2, 0 ) = 1/3/2000 12:00:00 AM
```

```
DATEADD( 'MONTH', 4, 0 ) = 5/1/2000 12:00:00 AM
```

```
DATEADD( 'YEAR', 7, 0 ) = 1/1/2007 12:00:00 AM
```

```
DATEADD( 'YEAR', -1, 0 ) = 1/1/1999 12:00:00 AM
```

Other starting times might be specified as:

```
DATEADD( 'HOUR', -5, CDATE() )
```

```
DATEADD( 'DAY', 14, [DateToShip] )
```

## DATEDIFF

### Date/Time Function

**Syntax:** DATEDIFF(<string>,<timestamp>,<timestamp>)

The DATEDIFF function returns the difference between two timestamp values.

- The first **argument** specifies the type of time value to compare - valid types are the strings: 'SECOND', 'MINUTE', 'HOUR', 'DAY', 'MONTH', and 'YEAR'.
- The second **argument** is a timestamp that specifies the starting time.
- The third **argument** is a timestamp that specifies the ending time.

DATEDIFF returns the number based on the first argument type.

In the following examples, the 0 starting time argument represents *1/1/2000 12:00:00 AM*. The ending time is in seconds:

```
DATEDIFF( 'SECOND', 0, 10 ) = 10
```

```
DATEDIFF( 'MINUTE', 0, 120 ) = 2
```

Other arguments might be specified as:

```
DATEDIFF( 'DAY', [OrderDate], CDATE() )
```

```
DATEDIFF( 'MONTH', [OrderDate], [ShippedDate] )
```

You may wish to use the [ROUND](#) function:

```
ROUND( DATEDIFF( 'YEAR', [AccountOpenDate], [AccountCloseDate] ) )
```

## DATENAME

### Date/Time Function

**Syntax:** DATENAME(<string>,<timestamp>)

The DATENAME function returns a string representing a part of a timestamp, that is, it returns a value such as 'Friday', 'December', or '2007'.

- The first **argument** specifies the type of time value to return - valid types are the strings: 'SECOND', 'MINUTE', 'HOUR', 'DAY', 'MONTH', and 'YEAR'.
- The second **argument** is a timestamp that specifies the time or date.

In the following examples, the current date returned by [CDATE](#) is 12/28/2006 04:30:00 PM:

```
DATENAME( 'HOUR', CDATE() ) = '16'
```

```
DATENAME( 'DAY', CDATE() ) = 'Thursday'
```

```
DATENAME( 'MONTH', CDATE() ) = 'December'
```

```
DATENAME( 'YEAR', CDATE() ) = '2006'
```

To return a numeric value rather than a string, see [DATEPART](#).

## DATEPART

### Date/Time Function

**Syntax:** DATEPART(<string>,<timestamp>)

The DATEPART function returns a number representing a part of a timestamp.

- The first **argument** specifies the type of time value to return - valid types are the strings: 'SECOND', 'MINUTE', 'HOUR', 'DAY', 'MONTH', and 'YEAR'.
- The second **argument** is a timestamp that specifies the time or date.

In the following examples, the current date returned by [CDATETIME](#) is 12/28/2006 17:30:15

```
DATEPART( 'HOUR', CDATETIME() ) = 17
```

```
DATEPART( 'DAY', CDATETIME() ) = 28
```

```
DATEPART( 'MONTH', CDATETIME() ) = 12
```

```
DATEPART( 'YEAR', CDATETIME() ) = 2006
```

Page | 199

To return a string value rather than a number, see [DATENAME](#).

See also: [SECOND](#), [MINUTE](#), [HOUR](#), [DAY](#), [WEEK](#), [MONTH](#), [YEAR](#)

## DAY

### Date/Time Function

**Syntax:** DAY(<date>) or DAY (<timestamp>)

The DAY function returns a number representing the day part of the date or timestamp argument. This is the same as DATEPART( 'DAY', <timestamp> )

Example:

If the current date returned by [CDATETIME](#) is 12/28/2006 17:30:15, then

```
DAY(CDATETIME() ) = 28
```

To return a string value rather than a number, see [DATENAME](#).

See also: [DATEPART](#), [SECOND](#), [MINUTE](#), [HOUR](#), [WEEK](#), [MONTH](#), [YEAR](#)

## DAYOFYEAR

### Date/Time Function

**Syntax:** DAYOFYEAR(<date>)

DAYOFYEAR (<timestamp>)

The DAYOFYEAR function returns the day of the year expressed as a value between 1 and 366 representing the specified date or timestamp argument.

Examples:

---

```
DAYOFYEAR({2013-02-05}) = 36
```

```
DAYOFYEAR({2011-12-31}) = 365
```

```
DAYOFYEAR({2012-12-31}) = 366
```

If the current date returned by [CDATE](#) is 2013-03-18, then

```
DAYOFYEAR(CDATE()) = 77
```

See also: [DATEPART](#), [DAY](#)

## DAYOFWEEK

### Date/Time Function

**Syntax:** DAYOFWEEK(<date>)

DAYOFWEEK (<timestamp>)

The DAYOFWEEK function returns a number from 1 (Sunday) to 7 (Saturday) representing the day of the week for the specified date or timestamp argument.

Examples:

```
DAYOFWEEK({10/03/2013}) = 1
```

If the current date returned by [CDATE](#) is 16/03/2013, then

```
DAYOFWEEK(CDATE()) = 7
```

See also: [WEEKDAY](#), [DATEPART](#), [DAY](#)

## DEGREES

### Math Function

**Syntax:** DEGREES(<float>)

The DEGREES function returns the *argument* converted from radians to degrees.

Some examples:

```
DEGREES(3.14159) = 180
```

```
DEGREES(1.0995575) = 63
```

This function is the opposite of [RADIANS](#).

## EXP

### Math Function

**Syntax:** EXP(<float>)

The EXP function returns the value of e (the base of natural logarithms) raised to the power of the **argument**.

Some examples:

EXP(2) = 7.38905609893065

EXP(0) = 1

EXP(-2) = 0.135335283236613

## FLOOR

### Math Function

**Syntax:** FLOOR(<float>)

FLOOR returns the largest integer value not greater than the **argument**. That is, the value is rounded down to the nearest integer. This is the opposite of [CEIL](#).

Some examples:

FLOOR(1.23) = 1

FLOOR(-1.23) = -2

---

## GETDATE

Date/Time Function

**Syntax:** GETDATE()

Page | 202

Returns the current system date in the standard internal format for datetime values.

Example:

GETDATE() = 24/04/2013

See also: [CDATE](#)

## GETDATETIME

Date/Time Function

**Syntax:** GETDATETIME()

Returns the current system date and time in the standard internal format for datetime values.

See also: [CDATETIME](#)

## GETUTCDATE

Date/Time Function

**Syntax:** GETUTCDATE()

Returns the current system date and time in the Coordinated Universal Time (UTC) standard.

## HOUR

Date/Time Function

---

**Syntax:** HOUR (<timestamp>)

The HOUR function returns a number representing the hour part of the date or timestamp argument. This is the same as DATEPART( 'HOUR', <timestamp> )

Example:

Page | 203

If the current date returned by [CDATETIME](#) is 12/28/2006 17:30:15, then

```
HOUR( CDATETIME() ) = 17
```

See also: [DATEPART](#), [SECOND](#), [MINUTE](#), [DAY](#), [WEEK](#), [MONTH](#), [YEAR](#)

## IN

### General Function

**Syntax:** IN(<expression> {,<expression>,...})

This function determines if a given value matches any value in a list. Items in the list are separated by commas. The result is of boolean type (True or False).

The following example flags records in which the [native-country] field contains either Mexico, Canada, or United-States:

```
CASE
  WHEN [native-country] IN('Mexico', 'Canada', 'United-States') then
    true
  ELSE
    false
END
```

See also [CASE](#).

## ISDATE

### Date/Time Function

**Syntax:** ISDATE(<string>)

ISDATE returns a TRUE/FALSE value based on whether the **argument** is a valid date.

Some examples:

```
ISDATE( 'word' ) = FALSE
```

```
ISDATE( '1/1/2000 12:00:45 AM' ) = TRUE
```

```
ISDATE( 'sunday' ) = TRUE
```

```
ISDATE( '12:00:00 AM' ) = TRUE
```

```
ISDATE( 'May 10, 2007' ) = TRUE
```

## LCASE

### String Function

**Syntax:** LCASE(<string>)

This function returns the string **argument** with all characters changed to lowercase.

For example:

```
LCASE('MODEL') = model
```

In the following example, we can flag all records with the name *Smith* regardless of whether the record contains *Smith*, *SMITH*, or even *sMith*. Here, we populate a True/False field with the expression:

```
CASE WHEN LCASE([LastName]) = 'smith' THEN true ELSE false END
```

This is the opposite of [UCASE](#).

## LEN



## String Function

**Syntax:** LEN(<string>)

This is an alternative to [CHARACTER LENGTH](#).

Examples:

```
LEN('UK') = 2
```

```
LEN('United Kingdom') = 14
```

Page | 205

**LIKE**

## General Function

**Syntax:** LIKE <string>

This determines whether or not a given character string matches a specified pattern.

A pattern can include regular characters and wildcard characters. During pattern matching, regular characters must exactly match the characters specified in the character string. Wildcard characters, however, can be matched with any fragments of the character string. Using wildcard characters makes the LIKE operator more flexible than using the = and != string comparison operators.

| WILDCARD       | EXPLANATION                                                                      |
|----------------|----------------------------------------------------------------------------------|
| %              | Any string of zero or more characters.                                           |
| _ (underscore) | Any single character.                                                            |
| [ ]            | Any single character within the specified range ([a-f]) or set ([abcdef]).       |
| [^]            | Any single character not within the specified range ([^a-f]) or set ([^abcdef]). |

In the following example, a match is found when the ProductCode begins with any character and is followed by MMV- plus any number of other characters (e.g., uMMV-2983 or 1MMV-kratt43).

```
CASE
  WHEN [ProductCode] LIKE '_MMV-%' then
    true
  ELSE
    false
END
```

See also [CASE](#).

---

---

## LOG

### Math Function

Page | 206

**Syntax:** LOG(<float>)

The LOG function returns the natural logarithm of the *argument*. The argument must be larger than zero.

Some examples:

LOG(2) = 0.693147180559945

If you want the log of a number  $X$  to some arbitrary base  $B$ , use the formula  $\text{LOG}(X)/\text{LOG}(B)$ .

## LOG10

### Math Function

**Syntax:** LOG10(<float>)

The LOG10 function returns the base-10 logarithm of the *argument*. The argument must be larger than zero.

Some examples:

LOG10(2) = 0.301029995663981

LOG10(100) = 2

## LOG2

### Math Function

**Syntax:** LOG2(<float>)

The LOG2 function returns the base-2 logarithm of the *argument*. The argument must be larger than zero.

Some examples:

`LOG2(2) = 1`

`LOG2(3) = 1.5849625007211561`

## LOWER

String Function

**Syntax:** LOWER(<string>)

This is an alternative to [LCASE](#).

## LTRIM

String Function

**Syntax:** LTRIM(<string>)

The LTRIM function returns the string **argument** with leading space characters removed.

For example:

`LTRIM(' word') = word`

This function is related to [RTRIM](#) and [TRIM](#).

## MAX

Aggregate Function

**Syntax:** MAX(<expr>,<expr>,[...]) OVER()

MAX returns the largest value in the list of **arguments**. You must have at least two arguments which should all be of the same type: numeric or string.

The arguments are compared and the value type returned using these rules:

- If all arguments are integers, they are compared as integers.
- If all arguments are numbers, but not all are integers, they are compared as real numbers.
- If all arguments are strings, they are compared as case-sensitive strings.

Some examples:

```
MAX(2,0) OVER() = 2
```

```
MAX(34.0,3,5.0,767.0) OVER() = 767
```

```
MAX('B','A','C') OVER() = C
```

```
MAX('b','A','C') OVER() = b
```

## MEDIAN

### Aggregate Function

**Syntax:** MEDIAN(<expression>) OVER()

MEDIAN returns the middle value in the numeric column specified by the argument. This is the middle value across all records. When there is an odd number of records, the median is simply the middle number. When there is an even number of records, the median is the mean of the two middle numbers.

For example, to get the median age in a record set, you might use:

```
MEDIAN( [Age] ) OVER()
```

The following example flags records with high incomes if the median age in the dataset is less than 30. However, it returns False when an income is low or in all cases if the median age is greater than 29.

```
CASE
```

```
  WHEN MEDIAN( [age] ) OVER() < 30 THEN
```

```
    CASE
```

```
      WHEN [Income] = ( '>50K' ) THEN
```

```
        True
```

```
    ELSE
      False
    END
  ELSE
    False
  END
```

## MIN

### Aggregate Function

**Syntax:** MIN(<expr>,<expr>,[...]) OVER()

MIN returns the smallest value in the list of **arguments**. You must have at least two arguments which should all be of the same type: numeric or string.

The arguments are compared and the value type returned using these rules:

- If all arguments are integers, they are compared as integers.
- If all arguments are numbers, but not all are integers, they are compared as real numbers.
- If all arguments are strings, they are compared as case-sensitive strings.

Some examples:

```
MIN(2,0) OVER() = 0
```

```
MIN(34.0,3,5.0,767.0) OVER() = 3
```

```
MIN('B','A','C') OVER() = A
```

```
MIN('a','A','C') OVER() = A
```

## MINUTE

### Date/Time Function

**Syntax:** MINUTE (<timestamp>)

The MINUTE function returns a number representing the minute part of the date or timestamp argument. This is the same as DATEPART( 'MINUTE', <timestamp> )

Example:

If the current date returned by [CDATETIME](#) is 12/28/2006 17:30:15, then

```
MINUTE(CDATETIME()) = 30
```

See also: [DATEPART](#), [SECOND](#), [HOUR](#), [DAY](#), [WEEK](#), [MONTH](#), [YEAR](#)

## MKDATE

### Date/Time Function

**Syntax:** MKDATE(<integer>,<integer>,<integer>)

The MKDATE function returns a timestamp value based on the integer arguments.

- The first **argument** specifies the year.
- The second **argument** specifies the month.
- The third **argument** specifies the day.

If the day argument is larger than possible, the month value is increased. Likewise, if the month argument is larger than 12, the year value is increased.

Some examples:

```
MKDATE( 2007, 10, 29 ) = 10/29/2007 12:00:00 AM
```

```
MKDATE( 2007, 10, 32 ) = 11/1/2007 12:00:00 AM
```

```
MKDATE( 2007, 13, 25 ) = 1/25/2008 12:00:00 AM
```

## MKTIME

### Date/Time Function

**Syntax:** MKTIME(<integer>,<integer>,<integer>)

The MKTIME function returns a time value based on the integer arguments.

- The first **argument** specifies the hour.

- The second **argument** specifies the minute.
- The third **argument** specifies the second (time).

If the second (time) argument is larger than 59, the minute value is increased. Likewise, if the minute argument is larger than 59, the hour value is increased. The hour value is simply rolled over after 24.

Some examples:

```
MKTIME( 10, 30, 0 ) = 10:30:00
```

```
MKTIME( 10, 30, 72 ) = 10:31:12
```

```
MKTIME( 10, 61, 0 ) = 11:01:00
```

```
MKTIME( 25, 30, 0 ) = 01:30:00
```

## MKTIMESTAMP

### Date/Time Function

**Syntax:** MKTIMESTAMP(<integer>,<integer>,<integer>,<integer>,<integer>,<integer>)

The MKTIMESTAMP function returns a timestamp value based on the six integer arguments. This is a combination of the [MKDATE](#) and [MKTIME](#) functions.

- The first **argument** specifies the year.
- The second **argument** specifies the month.
- The third **argument** specifies the day.
- The fourth **argument** specifies the hour.
- The fifth **argument** specifies the minute.
- The sixth **argument** specifies the second (time).

The same rules apply here as for MKDATE and MKTIME.

For example:

```
MKTIMESTAMP( 2006, 10, 23, 8, 45, 30 ) = 10/23/2006 8:45:30
```

## MODE

### Aggregate Function

**Syntax:** MODE(<expression>) OVER()

Mode returns the most common value in the column specified by the argument. This is the most common value across all records.

Page | 212

For example, to get the most common country name in a record set, you might use:

```
MODE( [Country] ) OVER()
```

The following example flags records with high incomes if the most common country name in the dataset is United States. However, it returns False when an income is low or in all cases if the most common country name is something else.

```
CASE
  WHEN MODE( [Country] ) OVER() = 'United States' THEN
    CASE
      WHEN [Income] = ( '>50K' ) THEN
        True
      ELSE
        False
    END
  ELSE
    False
END
```

## MONTH

### Date/Time Function

**Syntax:** MONTH (<date>)  
MONTH (<timestamp>)

The MONTH function returns a number representing the month part of the date or timestamp argument. This is the same as DATEPART( 'MONTH', <timestamp> )

Example:

If the current date returned by [CDATETIME](#) is 12/28/2006 17:30:15, then

```
MONTH( CDATETIME() ) = 12
```



See also: [DATEPART](#), [SECOND](#), [MINUTE](#), [HOUR](#), [DAY](#), [WEEK](#), [YEAR](#)

## MONTHNAME

### Date/Time Function

**Syntax:** MONTHNAME(<integer>)

This returns the name of the month based on the integer argument.

For example:

```
MONTHNAME( 1 ) = 'January'
```

## NULLIF

### General Function

**Syntax:** NULLIF(<expr>,<expr>)

NULLIF returns a null value if its two arguments equivalent. It returns the value of the first expression if they are not equivalent. Note that NULLIF is case sensitive when string types are compared.

Some examples:

```
NULLIF( 1, 1 ) = null
```

```
NULLIF( 3, 2 ) = 3
```

```
NULLIF( 'one', 'one' ) = null
```

```
NULLIF( 'one', 'One' ) = 'one'
```

## PARSEPARTS

### String Function

**Syntax:** PARSEPARTS(<string>,<string>,<integer>,<integer>)

The PARSEPARTS function works as follows:

Step 1. The first string argument is split into a list of substrings using the first character of the second argument (the delimiter argument).

Step 2. Each substring is numbered from 0. For example, in the case of PARSEPARTS('a,b,c,d,e', '\$\$', 2, 4) the list would be

Page | 214

| Number | Sub-string |
|--------|------------|
| 0      | a          |
| 1      | b          |
| 2      | c          |
| 3      | d          |
| 4      | e          |

Step 3. The return value is built up from concatenating the substrings in the range defined by the third and fourth arguments, inserting the delimiter argument between each substring. In the case of PARSEPARTS('a,b,c,d,e', '\$\$', 2, 4), the range is from the 2nd to 4th substring. This produces the result "c,\$\$d,\$\$e"

Examples:

```
PARSEPARTS('a,b,c,d,e', '$$', 2, 4) = c,$$d,$$e
```

```
PARSEPARTS('a,b,c,d,e', '$$', 2, 3) = c,$$d
```

## POSSTR

String Function

**Syntax:** POSSTR(<string>,<string>)

The POSSTR function returns the position of the second string argument within the first string argument. If there is more than one occurrence, POSSTR returns the position of the first occurrence. If the second string argument is not found within the first one, POSSTR returns 0.

```
POSSTR('toolbox', 'o') = 2
```

```
POSSTR('toolbox', 'box') = 5
```

```
POSSTR('box', 'toolbox') = 0
```

See also: [CHARINDEX](#)

---

---

## POW

### Math Function

**Syntax:** POW(<float>,<float>)

Page | 215

The POW function returns the value of the first **argument** raised to the power of the second **argument**. This is the same as the POWER function.

Some examples:

POW(2,2) = 4

POW(2,-2) = 0.25

POW(2,3) = 8

POW(3,2) = 9

## POWER

### Math Function

**Syntax:** POWER(<float>,<float>)

This is an alternative to [POW](#).

## QUARTER

### Date/Time Function

**Syntax:** QUARTER(<date>)

QUARTER (<timestamp>)

The QUARTER function returns the quarter of the year in which the date or timestamp argument occurs. It is expressed as a value between 1 and 4.

Some examples:

QUARTER({01/01/2013}) = 1

QUARTER({24/12/2012}) = 4

See also: [DATEPART](#), [MONTH](#), [MONTHNAME](#)

## RADIANS

### Math Function

Page | 216

**Syntax:** RADIANS(<float>)

The RADIANS function returns the **argument** converted from degrees to radians.

Some examples:

RADIANS(180) = 3.14159265358979

RADIANS(63) = 1.09955742875643

RADIANS(90) = 1.5707963267949

This function is the opposite of [DEGREES](#).

## RAND

### General Function

**Syntax:** RAND()

The RAND function returns a random floating-point value in the range 0 to 1.0.

The following values may be returned:

RAND() = 0.44825727492570877

RAND() = 0.7087954482994973

RAND() = 0.98769719549454749

## RANDOM

## General Function

**Syntax:** RANDOM()

This is an alternative to [RAND](#).

## REPEAT

## String Function

**Syntax:** REPEAT(<string>,<integer>)

REPEAT returns a string consisting of the first **argument** repeated by the number of times specified by the second argument. If the second argument is less than 1, REPEAT returns a null string.

Some examples:

```
REPEAT( 'Loo', 2 ) = 'LooLoo'
```

```
REPEAT( '-', 8 ) = '-----'
```

```
REPEAT( 'Can', 0 ) = '(null)'
```

## REPLACE

## String Function

**Syntax:** REPLACE(<string>,<string>,<string>)

This function replaces all or some of the string in the first **argument**. The second **argument** determines what is replaced while the third **argument** is the replacement text. If the value of the second argument is not found, the original value in the first argument is returned.

Some examples:

```
REPLACE( 'overview', 'view', 'take' ) = 'overtake'
```

```
REPLACE( 'overview', 'cast', 'take' ) = 'overview'
```

```
REPLACE( [ProductNumber], '-US', '-EU' )
```

## ROUND

### Math Function

**Syntax:** ROUND(<float>,<integer>)

ROUND returns the first **argument** , rounded to the number of decimals specified by the second **argument**. If the second argument is 0, the result will have no decimal point or fractional part. To round values to the nearest ten, use -1, to the nearest hundred, use -2, and so on.

Some examples:

```
ROUND(1.398, 1) = 1.4
```

```
ROUND(2.298, 0) = 2
```

```
ROUND(26, -1) = 30
```

```
ROUND(75, -2) = 100
```

## ROW\_NUMBER

### General Function

**Syntax:** ROW\_NUMBER()

The ROW\_NUMBER function returns the sequential number of a record (row) in the current dataset, starting at 1 for the first record. You can use this function, for example, to add an index column.

The following example expression for a new calculated column populates its values with record numbers for all records up to #1000 and the rest with nulls:

```
CASE WHEN ROW_NUMBER() <= 1000 THEN ROW_NUMBER() ELSE null END
```

## RTRIM

### String Function

**Syntax:** RTRIM(<string>)

The RTRIM function returns the string **argument** with trailing space characters removed.

For example:

```
RTRIM('word ') = 'word'
```

This function is related to [LTRIM](#) and [TRIM](#).

## SECOND

### Date/Time Function

**Syntax:** SECOND (<timestamp>)

The SECOND function returns a number representing the seconds part of the date or timestamp argument. This is the same as DATEPART( 'SECOND', <timestamp> )

Example:

If the current date returned by [CDATETIME](#) is 12/28/2006 17:30:15, then

```
SECOND( CDATETIME() ) = 15
```

See also: [DATEPART](#), [MINUTE](#), [HOUR](#), [DAY](#), [WEEK](#), [MONTH](#), [YEAR](#)

## SIN

### Math Function

**Syntax:** SIN(<float>)

Returns the sine of the **argument**, where the argument is given in radians.

For example:

```
SIN(3.14159) = 0
```

## SINH

### Math Function

**Syntax:** SINH(<float>)

Page | 220

Returns the hyperbolic sine of the **argument**, where the argument is given in radians.

For example:

```
SINH(1.57079632679) = 2.30129890231
```

## SOUNDEX

### String Function

**Syntax:** SOUNDEX(<string>)

This returns a *soundex* string based on the value in the **argument**. Two strings that sound almost the same should have identical soundex strings. All international alpha characters outside the A-Z range are treated as vowels.

Some examples:

```
SOUNDEX('Hello') = 'H400'
```

```
SOUNDEX('Halo') = 'H400'
```

```
SOUNDEX('Quadratically') = 'Q363'
```

## SQRT

### Math Function

**Syntax:** SQRT(<float>)

The SQRT function returns the square root of the **argument**. The argument must be a nonnegative number.

For example:



---

`SQRT(4.0) = 2`

`SQRT(20.0) = 4.47213595499958`

If the argument is negative, the (null) value is returned.

## SQUARE

### Math Function

**Syntax:** SQUARE(<float>)

This returns the square of the *argument*.

For example:

`SQUARE(5) = 25`

## STDDEV

### Aggregate Function

**Syntax:** STDDEV(<numeric>) OVER()

STDDEV returns the standard deviation in the numeric column specified by the argument. This is the standard deviation across all records.

For example, to get the standard deviation of age in a record set, you might use:

`STDDEV( [Age] ) OVER()`

## SUBSTR

### String Function

**Syntax:** SUBSTR(<string>,<integer>,<integer>)

SUBSTR returns a string found within the first **argument**. The second **argument** determines the starting position where 1 is the first character. The third argument determines the length of the returned string. If there is no string found, a (null) string is returned.

Some examples:

```
SUBSTR( '25-XMT', 4, 3 ) = 'XMT'
```

```
SUBSTR( '25-XMT', 1, 2 ) = '25'
```

```
SUBSTR( '25-XMT', 0, 1 ) = null
```

In the following example returns 'United States' if the two letters in the 10th and 11th position are 'US'. Otherwise, the return string is 'International'.

```
CASE
  WHEN SUBSTR( [ProductCode], 10, 2) = 'US' THEN
    'United States'
  ELSE
    'International'
END
```

## SUBSTRING

String Function

**Syntax:** SUBSTRING(<string>,<integer>,<integer>)

This is an alternative to [SUBSTR](#).

## SUM

Aggregate Function

**Syntax:** SUM(<expression>) OVER()

The aggregate function SUM returns the sum of its argument - a numeric column or an expression involving numeric columns. This is the sum over all records in the dataset if it is used for a [calculated column](#) or over all records in the node if used in a [node calculation](#).

For example, to get the sum of Amount Lost in a record set, you might use:

---

SUM( [Amount Lost] ) OVER()

## TAN

### Math Function

**Syntax:** TAN(<float>)

The TAN function returns the tangent of the **argument**, where the argument is given in radians.

For example:

`TAN(3.14159 + 1)` = 1.5574077246549018

## TANH

### Math Function

**Syntax:** TANH(<float>)

The TANH function returns the hyperbolic tangent of the **argument**, where the argument is given in radians.

For example:

`TANH(1.57079632679)` = 0.91715233566649657

## TRIM

### String Function

**Syntax:** TRIM(<string>)

The TRIM function returns the string **argument** with all leading and trailing spaces removed.

For example:

---

```
TRIM(' word ') = 'word'
```

See also: [LTRIM](#) and [RTRIM](#).

## TRUNC

### Math Function

**Syntax:** TRUNC(<float>,<integer>)

The TRUNC function returns the first **argument** truncated to the number of decimals specified by the second **argument**. The value is not rounded up or down but simply cut off. If the second argument is 0, the result will have no decimal point or fractional part.

Some examples:

```
TRUNC(1.223,1) = 1.2
```

```
TRUNC(1.999,2) = 1.99
```

```
TRUNC(1.999,0) = 1
```

## TRUNCATE

### Math Function

**Syntax:** TRUNCATE(<float>,<integer>)

This is an alternative to [TRUNC](#).

## UCASE

### String Function

**Syntax:** UCASE(<string>)

This function returns the string **argument** with all characters changed to uppercase according to the current character set mapping (the default is ISO-8859-1 Latin1).

For example:

```
UCASE('Part') = 'PART'
```

In the following example, we can flag all records with the name *Smith* regardless of whether the record contains *Smith*, *SMITH*, or even *sMith*. Here, we populate a True/False field with the expression:

```
CASE WHEN UCASE([LastName]) = 'SMITH' THEN true ELSE false END
```

See also: [LCASE](#).

## UPPER(<string>)

String Function

**Syntax:** UPPER(<string>)

This is an alternative to [UCASE](#).

## WEEK

Date/Time Function

**Syntax:** WEEK (<date>)

WEEK (<timestamp>)

The WEEK function returns a number representing the week part of the date or timestamp argument. This is the same as DATEPART('WEEK', <timestamp> )

Example:

If the current date returned by [CDATE](#) is 12/28/2006, then

```
WEEK( CDATE() ) = 52
```

See also: [DATEPART](#), [SECOND](#), [MINUTE](#), [HOUR](#), [DAY](#), [MONTH](#), [YEAR](#)

## WEEKDAY

Date/Time Function

**Syntax:** WEEKDAY (<date>)

WEEKDAY (<timestamp>)

This is an alternative to [DAYOFWEEK](#).

Examples:

```
WEEKDAY({10/03/2013}) = 1
```

If the current date returned by [CDATE](#) is 16/03/2013, then

```
WEEKDAY(CDATE()) = 7
```

See also: [DATEPART](#), [DAY](#)

## YEAR

Date/Time Function

**Syntax:** YEAR (<date>)

YEAR (<timestamp>)

The YEAR function returns a number representing the year part of the date or timestamp argument. This is the same as DATEPART( 'YEAR', <timestamp> )

Example:

If the current date returned by [CDATE](#) is 12/28/2006, then

```
YEAR( CDATE() ) = 2006
```

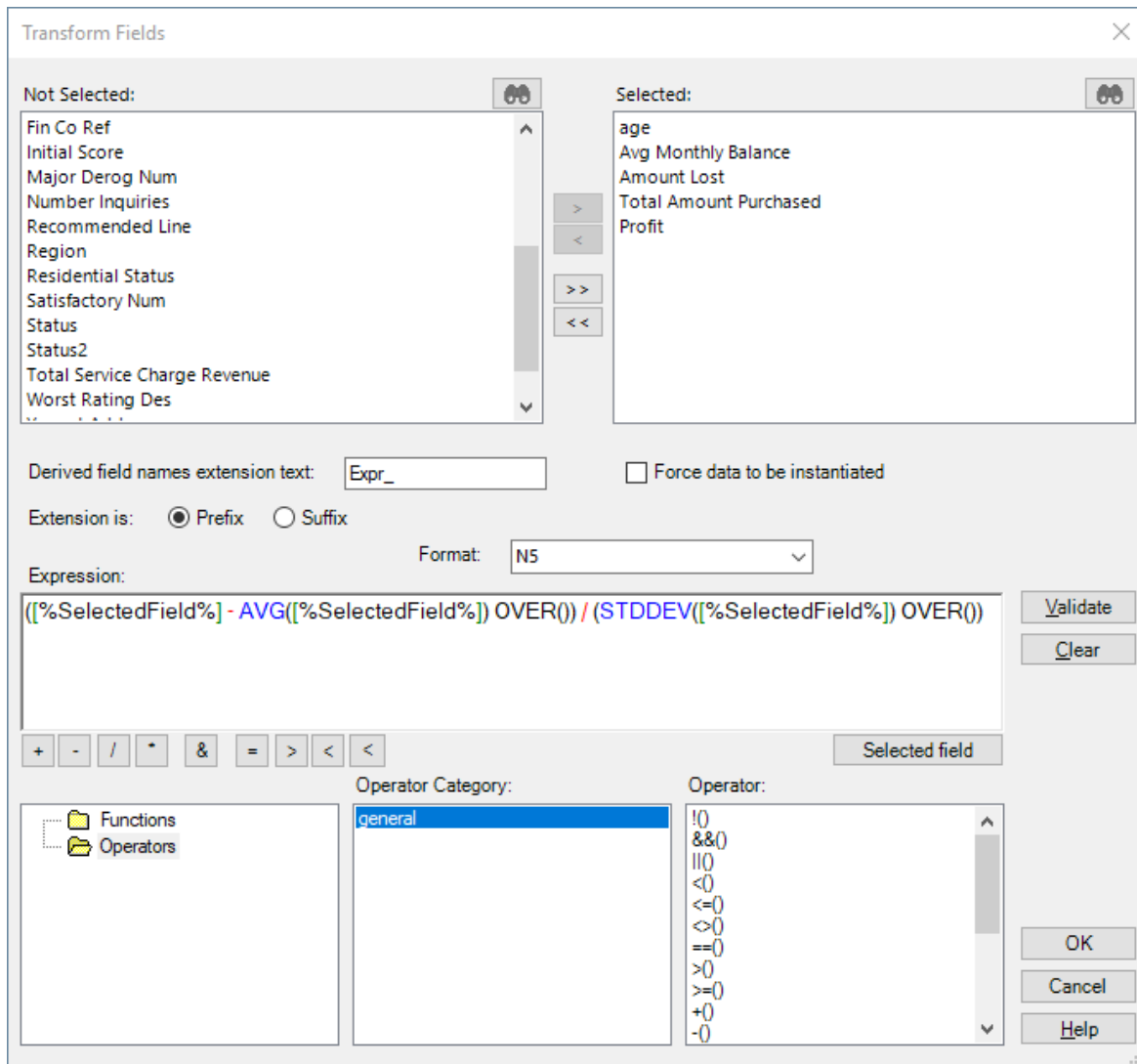
See also: [DATEPART](#), [SECOND](#), [MINUTE](#), [HOUR](#), [DAY](#), [WEEK](#), [MONTH](#)



## Transforming Multiple Variables

The **Transform Fields** button in the [Dataset Editor](#) opens an advanced Expression Editor that provides the capability to transform multiple variables at a time: Define an SQL expression and apply it to as many variables as desired. This may be useful in many cases, for example, for standard normalization of multiple numeric variables. The expression specified in the Expression field is applied to all variables in the Selected column at the top right. The variable [%SelectedField%] stands for all selected variables. You can enter it quickly at any time by clicking the **Selected field** button. See also the topic [About the Expression Editor](#).

In the example below, for each selected variable  $x$ , the transformation  $(x - \text{Mean}) / (\text{Standard Deviation})$  will be applied.



The **Transform Fields** dialog box is shown with the following configuration:

- Not Selected:**
  - Fin Co Ref
  - Initial Score
  - Major Derog Num
  - Number Inquiries
  - Recommended Line
  - Region
  - Residential Status
  - Satisfactory Num
  - Status
  - Status2
  - Total Service Charge Revenue
  - Worst Rating Des
- Selected:**
  - age
  - Avg Monthly Balance
  - Amount Lost
  - Total Amount Purchased
  - Profit
- Derived field names extension text:**  ☐ Force data to be instantiated
- Extension is:** ☒ Prefix ☐ Suffix
- Format:**
- Expression:**

```
([%SelectedField%] - AVG([%SelectedField%]) OVER()) / (STDDEV([%SelectedField%]) OVER())
```
- Buttons:** Validate, Clear, OK, Cancel, Help
- Operator Category:** general
- Operator:**
  - !()
  - &&()
  - ||()
  - <()
  - <=()
  - >()
  - >=()
  - ==()
  - +()
  - ()

For detailed description of format strings, see [Specifying Format](#).

See also:

[Variable Transformations](#)

[Expression Editor: Overview](#)

---

## Data Profiling

To profile your data, open the desired dataframe and analyze statistics and charts provided in the dataframe views. To navigate to the desired view, click on the corresponding tab at the bottom of the window. The following dataframe views are available:

Page | 228

### [Overview Report](#)

- The Overview Report summarizes your data with univariate statistics for each variable. You can also set a weight field in Overview Report. The available statistics are grouped into categories: Basic Measures, Measures of Central Tendency, Measures of Dispersion, Quantiles, Extreme Observations, Tests for Normality, Tests for Location, and Confidence Interval for Population Mean.

### [Chart](#)

- The Chart tab displays univariate charts that show the distribution of data values for each field.

### [Data](#)

- The Data view displays the data in the selected fields of the dataframe in the tabular form.

### [Segment Viewer](#)

- The Segment Viewer visualizes the distributions of the independent variables under each category of the selected segmentation variable.

### [Cross Tabs](#)

- The Cross Tabs view visualizes your data when plotting the categories of one field in relation to the categories of another field, as in pivot tables.

### [Characteristic Analysis](#)

- The Characteristic Analysis view allows you to analyze trends of the categories of a dependent variable in both chart and tabular format.

### [Correlations](#)

- The Correlations allows you to explore the strength of association between the variables in your dataframe. It displays four types of correlation coefficients for the selected fields: Pearson, Spearman's rho, Kendall's tau-b, and Hoeffding coefficients.

### [Saved Charts](#)

- The Saved Charts tab represents the library of charts you saved in other views of the current dataframe.

All charts and tables can be copied and pasted into any Microsoft Office application or image editor using the Edit menu commands, or [exported to PDF](#).



## Weight

Your data can contain a special numeric field whose values weight the values in the other fields that represent your analysis variables. The weight can be already in the data source to be imported or it can be created in Knowledge Studio after importing the data.

Weight can be used at the dataset level, as well as in trees and predictive and cluster models. All charts and reports take weight into account and display weighted statistics when a weight is defined.

A weight can be interpreted in different ways. The most common weight types are frequency and probability (sample) weights.

- Frequency weights are usually integer numbers that indicate how many observations each record represents. They can be used to compress the data: for example, instead of having  $N$  identical records in your data, you can include just one of them and put  $N$  in the same row in an additional field called Weight.
- Probability (or sample) weights represent the portion of specific observations selected into the sample from a population. For example, if the population of the customers who responded positively to your marketing campaign is 50000 and you select 20000 of them into your sample, your sampling fraction is  $2/5$ . Then the value of the weight for such customers is the inverse of the sampling fraction =  $5/2 = 2.5$ .

Any numeric field with the following properties can be used as weight in Knowledge Studio:

- Data type is Integer or Number (note: the Number type represents all real values)
- All values are nonnegative
- No missing values

A weight in Knowledge Studio is treated as a generalized frequency weight (since the weight values don't have to be integer), so the total number of records in the weighted dataset is always equal to the sum of the weight field and is not necessarily integer.

Since the weight values don't have to be integer, a weight can be interpreted as a frequency weight or a probability (sample) weight. Note that if you want the displayed total number of records in the sample remain the same after setting a weight, you can always normalize the weight field by multiplying it by  $N / \text{SUM}(\text{<Weight field>})$  in the Dataset Editor, where  $N$  is the actual (unweighted) sample size.

## Specifying Weight

If the field you would like to use as weight already exists in your dataset, the weight can be set in [Overview Report](#) by selecting the field from the Weight drop-down list (see the screenshot below). Only nonnegative numeric fields with no missing values are available to be set as weight.

If you want to create a weight, you can add it as a calculated column in the [Variable Transformations](#) node.

When a weight is set, the row in Overview Report representing the weight field is highlighted to distinguish it from the analysis variables.

Page | 230

| Calculate |                              | Calculate All |             | Table: [Credit].[Credit Data] |                       | Weight: Region Weight |        | Records: 76,099.00 |              |
|-----------|------------------------------|---------------|-------------|-------------------------------|-----------------------|-----------------------|--------|--------------------|--------------|
| #         | Field Name                   | Data Type     | Cardinality | # of Missing Values           | Minimum               | Maximum               | Mean   | Standard Deviation | Unique Count |
| 1         | Credit Line                  | Number        | 48          | 0                             | 0.00                  | 70,098.00             | 893.28 | 838.53             | 50           |
| 2         | Initial Score                | Number        | 363         | 0                             | 0.00                  | 867.00                | 362.32 | 135.93             | 65           |
| 3         | Recommended Line             | Number        | 24          | 0                             | 0.00                  | 10,000.00             | 451.88 | 449.50             | 27           |
| 4         | AgentID                      | String        | 100         | 0                             | A0                    | A99                   |        |                    | 0            |
| 5         | Number Inquiries             | Number        | 29          | 44136                         | 0.00                  | 99.00                 | 43.47  | 47.41              | 13           |
| 6         | Worst Rating Des             | String        | 10          | 43729                         | 1 satisfactory rating | No File               |        |                    | 0            |
| 7         | Year Last Check              | Number        | 14          | 44152                         | 0.00                  | 99.00                 | 16.73  | 36.72              | 3            |
| 8         | Major Derog Num              | Number        | 10          | 47714                         | 0.00                  | 9.00                  | 0.03   | 0.26               | 2            |
| 9         | Satisfactory Num             | Number        | 31          | 47714                         | 0.00                  | 99.00                 | 2.91   | 3.30               | 15           |
| 10        | Year at Address              | Number        | 70          | 33657                         | 0.00                  | 99.00                 | 5.72   | 7.74               | 16           |
| 11        | Bank Reference               | Number        | 5           | 31157                         | 0.00                  | 6.00                  | 1.00   | 0.06               | 6            |
| 12        | Fin Co Ref                   | Number        | 3           | 57355                         | 0.00                  | 1.00                  | 0.01   | 0.08               | 0            |
| 13        | Dept Store Ref               | Number        | 4           | 38848                         | 0.00                  | 3.00                  | 0.29   | 0.45               | 1            |
| 14        | Bank Dda Ref                 | Number        | 4           | 38012                         | 0.00                  | 4.00                  | 0.92   | 0.27               | 3            |
| 15        | Bank Save Ref                | Number        | 3           | 44075                         | 0.00                  | 1.00                  | 0.87   | 0.34               | 0            |
| 16        | Credit Card Ref              | Number        | 4           | 40397                         | 0.00                  | 4.00                  | 0.89   | 0.32               | 3            |
| 17        | Region                       | String        | 13          | 0                             | Eight                 | Two                   |        |                    | 0            |
| 18        | Avg Monthly Balance          | Number        | 1817        | 0                             | -552.00               | 9,837.00              | 212.29 | 384.96             | 1213         |
| 19        | Total Amount Purchased       | Number        | 2834        | 0                             | 1.00                  | 60,604.00             | 637.35 | 822.40             | 2104         |
| 20        | Total Service Charge Revenue | Number        | 516         | 0                             | 0.00                  | 1,642.00              | 40.77  | 82.25              | 324          |
| 21        | Profit                       | Number        | 8952        | 0                             | -8,647.92             | 1,760.16              | 8.12   | 230.68             | 15298        |
| 22        | Region Weight                | Number        | 2           | 0                             | 1.00                  | 3.00                  | 2.61   | 0.79               | 0            |
| 23        | Delinquent Lev 1             | Number        | 10          | 0                             | 0.00                  | 9.00                  | 0.64   | 1.05               | 0            |
| 24        | Delinquent Lev 2             | Number        | 7           | 0                             | 0.00                  | 6.00                  | 0.17   | 0.51               | 3            |
| 25        | Delinquent Lev 3             | Number        | 10          | 0                             | 0.00                  | 10.00                 | 0.25   | 0.89               | 3            |
| 26        | Amount Lost                  | Number        | 1416        | 0                             | 0.00                  | 9,837.60              | 45.52  | 239.71             | 3792         |
| 27        | age                          | Number        | 83          | 1672                          | 17.00                 | 98.00                 | 38.47  | 14.74              | 4            |
| 28        | Residential Status           | String        | 5           | 31618                         | Live with Parents     | Rent                  |        |                    | 0            |
| 29        | Status                       | String        | 3           | 0                             | Bad Debt              | OK                    |        |                    | 0            |

Overview Report

Dataset Chart

Data

Segment Viewer

Cross Tabs

Correlations

Distributions in all charts (Dataset Chart tab, Segment Viewer tab, Cross Tabs, Correlations) and tables are automatically recalculated based on the defined weight.

The weighted distributions for variables other than the weight are as follows:

If  $W$  is the weight variable,  $F_i$  is the  $i$ th value of a variable  $F$ , and  $\text{SUM}(W)$  is the sum of the weight field over all records where  $F=F_i$ , then

- The weighted total number of records =  $\text{SUM}(W)$
- The weighted frequency for the value  $F_i$  =  $\text{SUM}(W_i)$
- The weighted percentage of the value  $F_i$  =  $100 * \text{SUM}(W_i) / \text{SUM}(W)$

---

## Removing Weight. Switching between Weighted and Unweighted Representations

Once you set the weight, you can easily toggle back and forth between weighted and unweighted representations in all dataset views using the menu command **Edit | Toggle Weight On/Off** or the

toolbar button .

Page | 231

If a weight is set, the weight toggle button becomes highlighted: .

There are three ways to turn the weighting off:

- select “[No Weight]” in the Weight drop-down list
- select the menu command **Edit | Toggle Weight On/Off**
- click the weight toggle button  in the toolbar

The statistics and charts in all dataset views are automatically recalculated and updated whenever you toggle the weight off or on. This makes it very easy to compare various statistics in weighed and unweighted samples.

If you remove the weight, the univariate statistics in the Overview Report are automatically reset to uncalculated. Click **Calculate All** if necessary to recalculate.

NOTE: The univariate statistics shown for the weight field in Overview Report are the same as they were before setting this field as weight (i.e., the weight field is not weighted by itself). The same is true for all other dataset views: they show the *original* distribution of the weight variable (not weighted by itself).

## Relationship between Weights in Datasets and Dependent Models

The weights in a dataset and its dependent models can be changed or reset independently. When you toggle weight on or off in the dataset Overview Report, this does not affect existing trees and other models depending on the dataset.

By default, any new model created on the basis of a weighted dataset inherits the weight from it. More precisely, when inserting a model for a weighted dataset, the weight field defined in the dataset is automatically listed in the model's Attribute Editor as "weight" (in the Usage column). You can override this in the Insert Tree (or Predictive / Cluster Model) Wizard and redefine the field as an analysis variable by changing the usage from "weight" to "independent" if desired. Once the model is inserted, it is independent of the parent dataset in the sense that any changes of weights in the model will not affect the parent dataset and vice versa.

---

## Overview Report

The **Overview Report** is a dataframe view that displays the dataframe structure and univariate statistics such as the mean, maximum, and minimum values, the number of missing values, etc.

Page | 232

You can customize the Overview Report by choosing which statistics to display or hide using the command **Tools | Options** from the main menu. To set global preferences for new dataframes, use **Tools | Preferences**. See the topic [Customizing the Overview Report](#) for details. Note that Preferences are not applied to objects already created.

The statistics available in the Overview Report depend on the dataframe type. For Spark dataframes in Knowledge Studio for Apache Spark, there are 9 univariate statistics available in Overview Report. See also [Options and Preferences: Overview Report](#).

### Calculate and Calculate All commands

Upon importing a new dataframe, only the columns selected in **Tools | Preferences** -> Overview Report are displayed. You can add more statistics for the current dataframe using the **Tools | Options** command.

Click the **Calculate All** button to calculate the statistics for all fields. If you want to see the statistics for one field only, select the desired field and click the **Calculate** button.

A **weight** can be set in Overview Report by selecting the desired field from the *Weight* drop-down list. Note that the list contains only numeric fields with nonnegative values. For details about weights, see the [Weight](#) topic.

All columns in the Overview Report are sortable.

Calculate

Calculate All

Dataset: [SB\_Project02].[Credit\_Binary]

Weight: [No weight]

Records: 29,167

Fields: 29

| #  | Field Name                   | Data Type | Cardinality | Unique Count | # of Missing Values | % of Missing Values | Minimum               | Maximum   | Mean   | Std |
|----|------------------------------|-----------|-------------|--------------|---------------------|---------------------|-----------------------|-----------|--------|-----|
| 1  | Credit Line                  | Number    | 48          | 18           | 0                   | 0.00 %              | 0.00                  | 70,098.00 | 884.78 |     |
| 2  | Initial Score                | Number    | 363         | 27           | 0                   | 0.00 %              | 0.00                  | 867.00    | 366.50 |     |
| ▶  | Recommended Line             | Number    | 24          | 9            | 0                   | 0.00 %              | 0.00                  | 10,000.00 | 460.79 |     |
| 4  | AgentID                      | String    | 100         | 0            | 0                   | 0.00 %              | AO                    | A99       |        |     |
| 5  | Number Inquiries             | Number    | 29          | 5            | 16834               | 57.72 %             | 0.00                  | 99.00     | 43.67  |     |
| 6  | Worst Rating Des             | String    | 10          | 0            | 16695               | 57.24 %             | 1 satisfactory rating | No File   |        |     |
| 7  | Year Last Check              | Number    | 14          | 1            | 16840               | 57.74 %             | 0.00                  | 99.00     | 16.29  |     |
| 8  | Major Derog Num              | Number    | 10          | 2            | 18132               | 62.17 %             | 0.00                  | 9.00      | 0.03   |     |
| 9  | Satisfactory Num             | Number    | 31          | 5            | 18132               | 62.17 %             | 0.00                  | 99.00     | 2.95   |     |
| 10 | Year at Address              | Number    | 70          | 10           | 12677               | 43.46 %             | 0.00                  | 99.00     | 6.19   |     |
| 11 | Bank Reference               | Number    | 5           | 2            | 11761               | 40.32 %             | 0.00                  | 6.00      | 1.00   |     |
| 12 | Fin Co Ref                   | Number    | 3           | 0            | 21905               | 75.10 %             | 0.00                  | 1.00      | 0.01   |     |
| 13 | Dept Store Ref               | Number    | 4           | 1            | 14812               | 50.78 %             | 0.00                  | 3.00      | 0.31   |     |
| 14 | Bank Dda Ref                 | Number    | 4           | 1            | 14314               | 49.08 %             | 0.00                  | 4.00      | 0.93   |     |
| 15 | Bank Save Ref                | Number    | 3           | 0            | 17027               | 58.38 %             | 0.00                  | 1.00      | 0.88   |     |
| 16 | Credit Card Ref              | Number    | 4           | 1            | 15411               | 52.84 %             | 0.00                  | 4.00      | 0.89   |     |
| 17 | Region                       | String    | 13          | 0            | 0                   | 0.00 %              | Eight                 | Two       |        |     |
| 18 | Avg Monthly Balance          | Number    | 1817        | 445          | 0                   | 0.00 %              | -552.00               | 9,837.00  | 206.11 |     |
| 19 | Total Amount Purchased       | Number    | 2834        | 764          | 0                   | 0.00 %              | 1.00                  | 60,604.00 | 625.13 |     |
| 20 | Total Service Charge Revenue | Number    | 516         | 116          | 0                   | 0.00 %              | 0.00                  | 1,642.00  | 39.50  |     |
| 21 | Profit                       | Number    | 8952        | 5736         | 0                   | 0.00 %              | -8,647.92             | 1,760.16  | 8.76   |     |
| 22 | Delinquent Lev 1             | Number    | 10          | 0            | 0                   | 0.00 %              | 0.00                  | 9.00      | 0.63   |     |
| 23 | Delinquent Lev 2             | Number    | 7           | 1            | 0                   | 0.00 %              | 0.00                  | 6.00      | 0.17   |     |
| 24 | Delinquent Lev 3             | Number    | 10          | 1            | 0                   | 0.00 %              | 0.00                  | 10.00     | 0.25   |     |
| 25 | Amount Lost                  | Number    | 1416        | 1406         | 0                   | 0.00 %              | 0.00                  | 9,837.60  | 43.36  |     |
| 26 | age                          | Number    | 83          | 2            | 622                 | 2.13 %              | 17.00                 | 98.00     | 39.26  |     |
| 27 | Residential Status           | String    | 5           | 0            | 11924               | 40.88 %             | Live with Parents     | Rent      |        |     |
| 28 | Status                       | String    | 3           | 0            | 0                   | 0.00 %              | Bad Debt              | OK        |        |     |
| 29 | AccountStatus                | String    | 2           | 0            | 0                   | 0.00 %              | Bad Debt              | OK        |        |     |

III

Overview Report

Dataset Chart

Data

Segment Viewer

Cross Tabs

Characteristic Analysis

Correlations

Saved Charts

The contents of the Overview Report can be copied and pasted into a Microsoft Office application such as Word or Excel using the menu commands **Edit | Select All** and **Edit | Copy** or the keyboard shortcuts CTRL A and CTRL C.

General attributes of the dataframe are displayed at the top of the Overview Report:

|                |                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------|
| <b>Dataset</b> | Displays the dataframe name with the parent project name in the format <b>[project name].[dataframe name]</b> |
| <b>Weight</b>  | The weight field (if specified)                                                                               |
| <b>Records</b> | The total number of records in the dataframe                                                                  |
| <b>Fields</b>  | The total number of fields in the dataframe                                                                   |

## Attributes and univariate statistics displayed in Overview Report:

The following attributes and statistics are available in the Overview Report. Not all of them are displayed by default. See the topic [Customizing the Overview Report](#) for details.

| Name and Type |                                                                                     |
|---------------|-------------------------------------------------------------------------------------|
| #             | Field number. Fields are numbered in the same order as in the original data source. |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Field Name</b>  | Field name. Fields cannot be renamed unless they are calculated fields added with the Data Transformation operation. If you need to rename a raw field, re-import the dataframe and rename the field in the last page of the Insert Dataset Wizard (the Field Transformer page).                                                                                                                                                                                                                                                        |
| <b>Field Label</b> | The label of the field. This is an editable field for specifying an optional field label. In some application views you can choose whether to display the original field names or labels.                                                                                                                                                                                                                                                                                                                                               |
| <b>Data Type</b>   | <p>The data type of the field. Data types include Text, Integer, Number, Boolean, Date, Time, and Timestamp.</p> <p>Data types cannot be changed once the dataframe has been imported. If you need to change the type of a field, either re-import your dataframe and change the types in the last page of the import wizard (the Field Transformer page), or create a copy of the field converting it to the desired type using Data Transformations. Example type conversion command: <code>CAST ([field_name] AS Integer)</code></p> |

### Basic Measures

|                                |                                                                                                             |
|--------------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Cardinality</b>             | The number of distinct values in the field.                                                                 |
| <b>Unique Count</b>            | The number of values that occur exactly once.                                                               |
| <b># of Missing Values</b>     | The number of missing values in the field.                                                                  |
| <b>% of Missing Values</b>     | The percentage of missing values in the field.                                                              |
| <b># of Non-Missing Values</b> | The number of non-missing values in the field.                                                              |
| <b>% of Non-Missing Values</b> | The percent of non-missing values in the field.                                                             |
| <b>Minimum</b>                 | The minimum value in the field. Also applicable to String fields (with respect to the lexicographic order). |
| <b>Maximum</b>                 | The maximum value in the field. Also applicable to String fields (with respect to the lexicographic order). |
| <b>Sum</b>                     | The sum of all values in the field.                                                                         |

### Measures of Central Tendency

|               |                                                                                                                                                                                                                                                                                                            |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Mean</b>   | The mean value of the field.                                                                                                                                                                                                                                                                               |
| <b>Median</b> | The middle value of the ordered set of values in the field across all records, i.e., the value such that an equal number of values are less than and greater than this value (if the cardinality is odd), or the lesser of the two central values (if the cardinality is even). Also, see Quantiles below. |
| <b>Mode 1</b> | The most common value in the field.                                                                                                                                                                                                                                                                        |

**Mode 2** The second most common value in the field.

### Measures of Dispersion

|                                 |                                                                                                                                                                                                                               |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Standard Deviation</b>       | The standard deviation of the field values (the square root of variance).                                                                                                                                                     |
| <b>Range</b>                    | The difference between the maximum and the minimum values in the field.                                                                                                                                                       |
| <b>Interquartile Range</b>      | The difference between the upper and lower quartiles (the 75th and 25th percentiles). See the Quantiles section below.                                                                                                        |
| <b>Coefficient of Variation</b> | The ratio of the standard deviation to the mean value of the field.                                                                                                                                                           |
| <b>Std Error Mean</b>           | The standard error of the mean: The ratio of the standard deviation to the square root of the total number of records.                                                                                                        |
| <b>Variance</b>                 | The variance of the field values (the second central moment). It is equal to the mean of the squared values minus the square of the mean value of the field.                                                                  |
| <b>Skewness</b>                 | The skewness of the value distribution in the field. Characterizes the asymmetry of the distribution of numeric values in the field. A zero value indicates relatively even distribution of values on both sides of the mean. |
| <b>Kurtosis</b>                 | The kurtosis of the value distribution in the field. Characterizes the shape of the distribution of numeric values in the field in terms of the width of peak, tail weight, and lack of shoulders.                            |

Page | 235

### Quantiles

Quantiles are data points separating equal-sized (or approximately equal-sized) subsets of the ordered set of all values in a numeric field. The method used here to compute quantiles is as follows. Let  $x[1], \dots, x[n]$  be the ordered set of all distinct non-missing values in the field  $x$  in the ascending order.  $x[i]$  is called the *ith order statistic*. Let  $0 < p < 1$  and let  $j$  be the integer part of  $np - 1/2$ . Then:

$$Q(p) = x[j] \quad \text{if } j = np - 1/2 \text{ and } j \text{ is even}$$

$$Q(p) = x[j+1] \quad \text{otherwise}$$

In the list below,  $p$  is expressed as percentage (1%, 5%, etc.).

|                  |                                                                                                                   |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| <b>Customize</b> | Custom quantile. If the desired quantile is not among the ones listed below, specify the desired percentage here. |
| <b>1%</b>        | The lower percentile (1st percentile).                                                                            |
| <b>5%</b>        | The 5th percentile.                                                                                               |
| <b>10%</b>       | The lower decile (10th percentile).                                                                               |
| <b>25%</b>       | The lower quartile (the 25th percentile).                                                                         |
| <b>75%</b>       | The upper quartile (the 75th percentile).                                                                         |
| <b>90%</b>       | The upper decile (90th percentile).                                                                               |
| <b>95%</b>       | The 95th percentile.                                                                                              |



**99%** The upper percentile (99th percentile).

### Extreme Observations

**Lowest Observation** Show N lowest values, where N is specified in the Parameters column. The default is 5 values.

**Highest Observation** Show N highest values, where N is specified in the Parameters column. The default is 5 values.

Page | 236

### Tests for Normality

**Shapiro-Wilk Test** The Shapiro-Wilk test for the null hypothesis that the data in the field is normally distributed. It is the ratio of the square of a linear combination of the order statistics  $x[1], \dots, x[n]$  and a sum-of-squares estimator of the variance. The range of values for the statistic is (0, 1]. The null hypothesis is rejected if the value is too small. See Shapiro and Wilk (1965) in [References](#).

**Cramér-Von Mises Criterion** A goodness-of-fit test based on the empirical distribution function (EDF). When used as a test for normality, the Cramér-Von Mises Criterion evaluates the squared difference between the EDF of the data in the field and the standard normal distribution function  $F$ . Let  $x[1], \dots, x[n]$  be the ordered set of all distinct non-missing values in the field  $x$  in the ascending order. Then the statistic is computed as

$$\frac{1}{12n} + \sum_{i=1}^n \left( F(x_i) - \frac{2i-1}{2n} \right)^2$$

The null hypothesis that the data has standard normal distribution is rejected if the value is larger than the chosen threshold.

**Kolmogorov-Smirnov Test** A goodness-of-fit test based on the empirical distribution function (EDF). When used as a test for normality, the Kolmogorov-Smirnov statistic measures the largest vertical difference between the EDF of the data in the field and the standard normal distribution function  $F$  with the same mean and variance. Let  $x[1], \dots, x[n]$  be the ordered set of all distinct non-missing values in the field  $x$  in the ascending order. Then the statistic is the larger of the two values

$$\max_i \left( \frac{i}{n} - F(x_i) \right) \quad \text{and} \quad \max_i \left( F(x_i) - \frac{i-1}{n} \right)$$

**Anderson-Darling Test** A goodness-of-fit test based on the empirical distribution function (EDF). When used as a test for normality, the Anderson-Darling Test evaluates the weighted squared difference between the EDF of the data in the field and the standard normal distribution function  $F$ . Let



$x[1], \dots, x[n]$  be the ordered set of all distinct non-missing values in the field  $x$  in the ascending order, and  $y[i]$  be their standardized values. Then the statistic is computed as

$$-n - \frac{1}{n} \sum_{i=1}^n ((2i - 1) \ln F(x_i) + (2(n - i) + 1) \ln(1 - F(x_i)))$$

Page | 237

The null hypothesis that the data has standard normal distribution  $F$  is rejected if the value is larger than the chosen threshold.

### Tests for Location

#### Student's $t$ -test

A test statistic for the null hypothesis that the mean value of the field is equal to the specified value of  $\mu_0$ . It is set to zero by default. To change the value of  $\mu_0$ , click the **Set  $\mu_0$  values** button. The statistic is calculated as

$$\frac{\mu - \mu_0}{\sigma/\sqrt{n}}$$

where  $\mu$  is the mean value of the field, and  $\sigma$  is the standard deviation, and  $n$  is the number of all distinct nonmissing values in the field.

#### Sign Test

A test statistic for the null hypothesis that the median value of the field is equal to the specified value of  $\mu_0$ . It is set to zero by default. To change the value of  $\mu_0$ , click the **Set  $\mu_0$  values** button. The statistic is calculated as  $(n_1 - n_2)/2$ , where  $n_1$  is the number of values greater than  $\mu_0$ , and  $n_2$  is the number of values less than  $\mu_0$ .

#### Wilcoxon signed-rank test

A test statistic for the null hypothesis that the median value of the field is equal to the specified value of  $\mu_0$  under the assumption of a symmetric distribution. It is set to zero by default. Let  $R(|x_i - \mu_0|)$  be the rank of nonzero  $|x_i - \mu_0|$  and  $n_t$  be the number of values  $x[i]$  not equal to  $\mu_0$ . The test statistic is

$$\sum_{x_i \neq \mu_0} R(|x_i - \mu_0|) - \frac{n_t(n_t + 1)}{4}$$

#### Confidence Interval for Population Mean

A confidence interval for the mean is an estimated range of values which includes the mean of the variable at the probability level specified in the Parameters column. The default probability level is 95%. If this statistic is selected, the lower and upper bounds for the interval are displayed.

## See also:

[Customizing the Overview Report](#)

[Options and Preferences](#)

Page | 238

## Customizing the Overview Report

You can customize the Overview Report by selecting the attributes and statistics to be displayed and changing the font.

To customize the report for the current dataframe, select the **Tools | Options...** command from the menu or click the **Options** button in the toolbar.

To set the columns to be displayed by default in the Overview Report for all *new* dataframes, select **Tools | Preferences** instead of **Options** and choose the Overview Report tab. Changes in Preferences do not affect the dataframes already created.

There are 9 univariate statistics available in Overview Report. See [Options and Preferences: Overview Report](#).

You can expand any group by clicking on the plus sign on the left, or expand all groups by clicking Expand All. The application default includes cardinality, unique count, the number and percentage of missing values, minimum, maximum, mean, and standard deviation. For the description of all available statistics, see the [Overview Report](#) topic.

Select the desired attributes and statistics to be displayed and change the corresponding parameters if necessary.

Customize Font  
Grid Font: Segoe UI 8.25pt

Select columns to display  
☒ Row Number    ☒ Field Name    ☒ Label    ☒ Data Type

Expand All

|                                     |                                     | Name of Measure                         | Parameters          |
|-------------------------------------|-------------------------------------|-----------------------------------------|---------------------|
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Basic Measures                          |                     |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Measures of Central Tendency            |                     |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | Measures of Dispersion                  |                     |
| <input checked="" type="checkbox"/> | <input type="checkbox"/>            | Quantiles                               |                     |
| <input checked="" type="checkbox"/> | <input type="checkbox"/>            | Extreme Observations                    | 5                   |
| <input checked="" type="checkbox"/> | <input type="checkbox"/>            | Tests for Normality                     |                     |
| <input checked="" type="checkbox"/> | <input type="checkbox"/>            | Tests for Location                      | Default $\mu_0 = 0$ |
| <input checked="" type="checkbox"/> | <input type="checkbox"/>            | Confidence Interval for Population Mean | 95.00               |

Click OK. Then click **Calculate** or **Calculate All** in the Overview Report to calculate the newly added statistics.

Clicking **Reset** in the Options dialog will reset everything to the current Preferences. Clicking **Reset** in the Preferences dialog will reset everything to the application defaults described above.

Note: If you include all available statistics, they may take a while to calculate for very large dataframes.

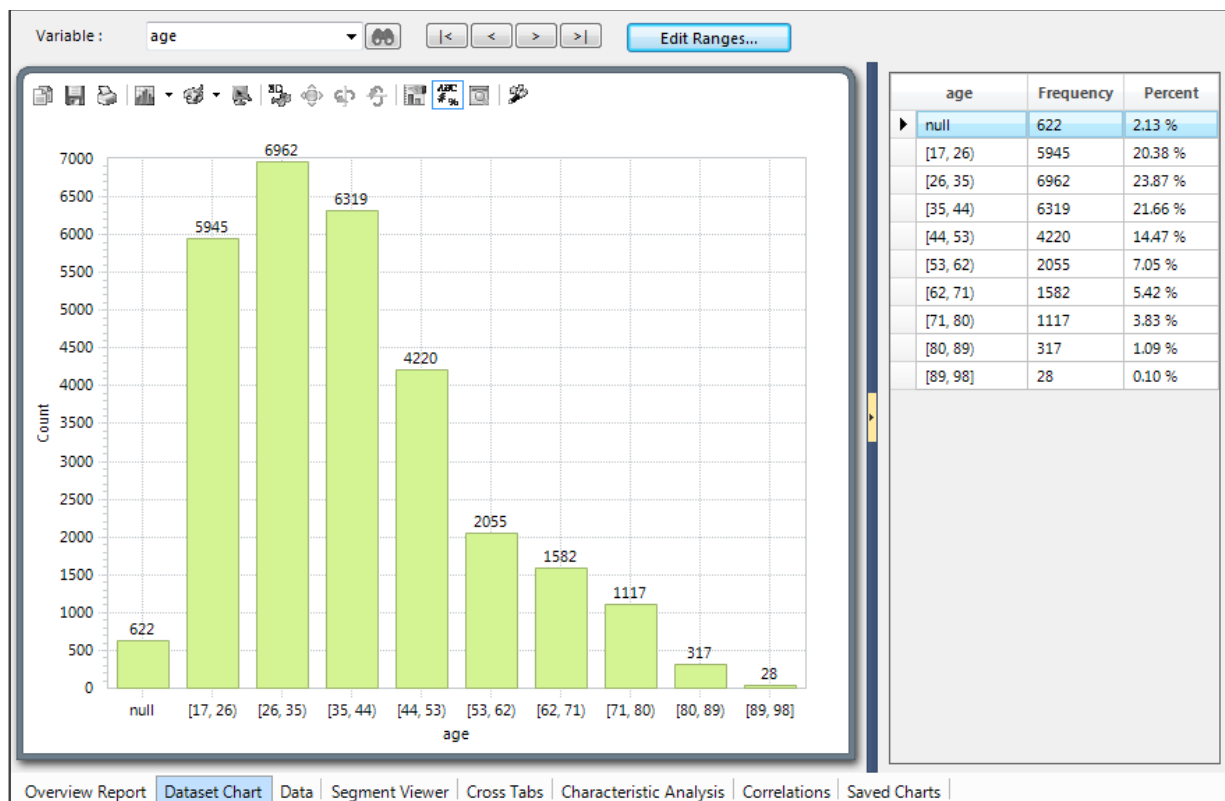
Note that the following statistics are not available in the In-Database Analytics mode, since there are no standard SQL operations for them: Unique Count, Variance, DevSqr, Mode (Mode1 and Mode2), Lower Quartile, Upper Quartile, Median, Skewness, and Kurtosis.

## Dataset Chart

The **Dataset Chart** is a view of the dataset that displays the distribution of data values for a given field in a chart.

If the field is continuous, the distribution is displayed as a histogram. If the field is discrete, the distribution is displayed as a pie chart. Various chart display settings including the chart type, labels, colors, and fonts can be changed using the **Chart** menu or the context menu that appears when right-clicking the chart. If you want to see the chart data, select the **Data Editor** item from the context menu.

You can change the field being displayed by choosing from the drop-down list labeled **Field** or using the navigation buttons to the right of the drop-down list.



By default, the chart for discrete variables shows a maximum of 10 categories. If the cardinality of a discrete variable is larger than 10, the remaining categories are mapped into a category designated by the symbol '####'. If you need to increase the maximum number of categories displayed, click the **Edit Grouping** button to change the threshold value.

**Note:** In Dataset Chart, a numeric, date, time, or timestamp variable is treated as discrete if its cardinality is less or equal to 10. In this case, its distribution is represented in a Pie Chart by default. Otherwise, it is considered to be continuous and its distribution is represented in a Histogram by default.

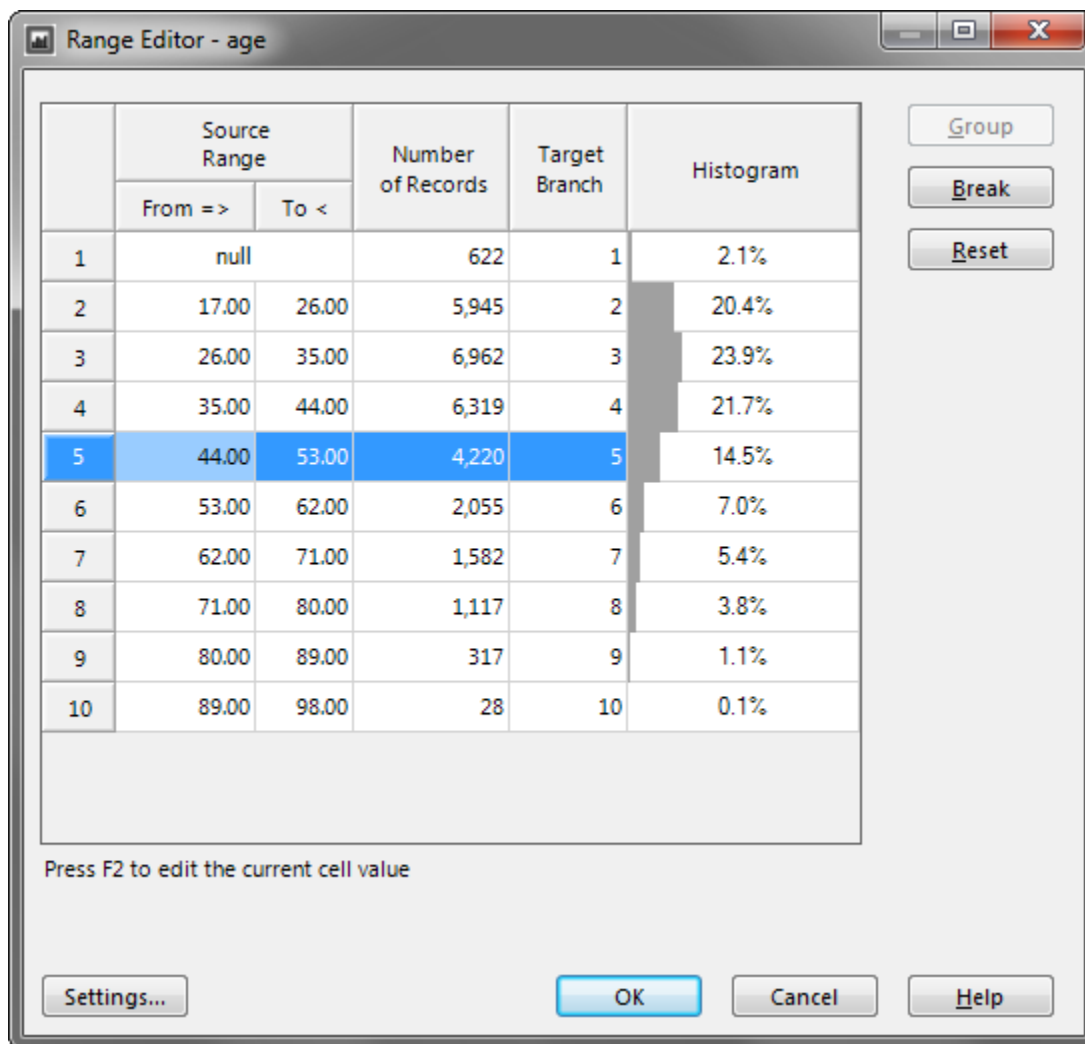
The [Edit Ranges](#) button is only available if the field to display is continuous. This feature allows you to change the displayed ranges of a continuous variable.

The **dataset chart** and the associated data can be copied and pasted into a Microsoft Office document. This allows the rapid and easy preparation of reports, presentations and summaries.

## Editing Ranges

Range Editor allows you to define the boundaries of the bins for a continuous variable and the way its data should be grouped and distributed across the bins.

Page | 241



## Range Display

**Source Range**  
**From=> To<**  
**Number of Records**

Indicates the boundaries of each interval (bin) for the selected variable distribution.

Number of records contained in the bin.

---

---

|                  |                                                                                |
|------------------|--------------------------------------------------------------------------------|
| <b>Target</b>    | Represents the index of the interval.                                          |
| <b>Branch</b>    |                                                                                |
| <b>Histogram</b> | Percentage of the total number of records in the dataset contained in the bin. |

## Range Editing

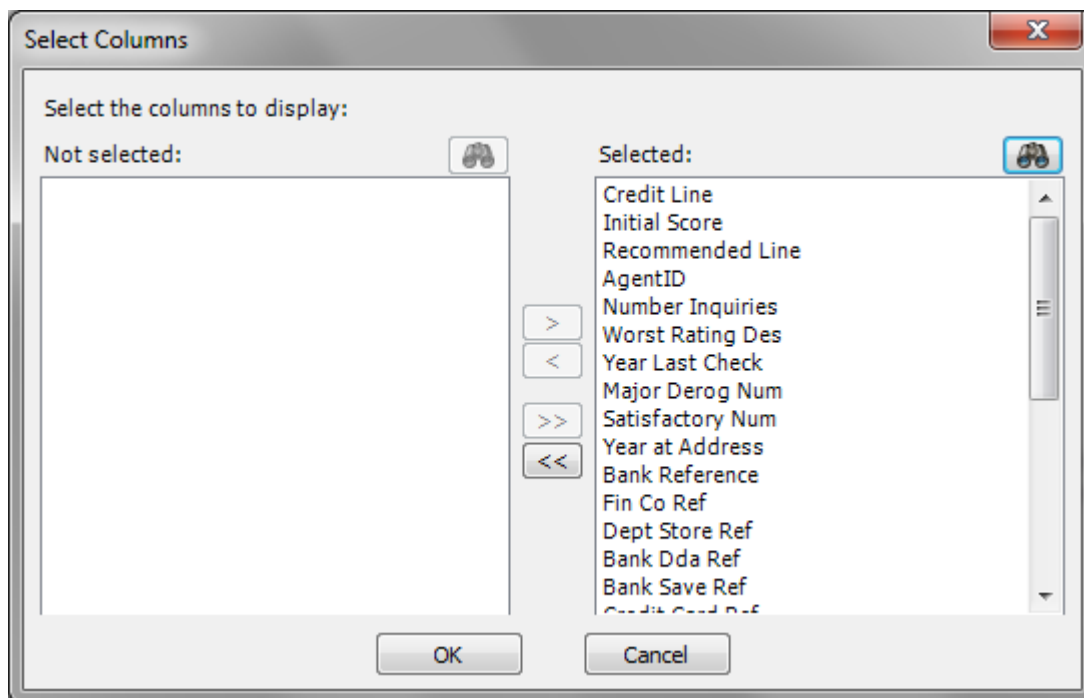
Page | 242

|                 |                                                                                                                                  |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------|
| <b>Group</b>    | Allows you to combine ranges. Only contiguous ranges can be grouped.<br>Example of Group                                         |
| <b>Break</b>    | Allows you to add a new interval to the distribution by specifying break points within an existing interval.<br>Example of Break |
| <b>Reset</b>    | The reset button allows you to return to the original ranges displayed when the range editor was first opened.                   |
| <b>Settings</b> | The settings button allows you to change the default range settings. See Settings.                                               |

## Data View

The **Data** view is a view of the dataset and displays the raw data in the dataset. Missing values in the dataset are displayed as `null`.

The first time you click on the Data tab, you are prompted to select the fields you would like to be displayed. All fields are selected by default. Exclude the fields you don't want to be shown.



Note that new calculated columns added in the Dataset Editor are not displayed by default in the Data tab if it had been opened before you added the new columns. Click the **Options** button in the toolbar to include the newly added columns in the Data view if necessary.

|    | Credit Line | Initial Score | Recommended Line | AgentID | Number Inquiries | Worst Rating Des               |
|----|-------------|---------------|------------------|---------|------------------|--------------------------------|
| 1  | 500         | 463           | 500              | A24     | 98               | 3 satisfactory ratings         |
| 2  | 2500        | 321           | 1                | A76     | (null)           | (null)                         |
| 3  | 500         | 331           | 500              | A66     | 98               | 1 satisfactory rating          |
| 4  | 1000        | 370           | 1                | A79     | (null)           | (null)                         |
| 5  | 1000        | 399           | 500              | A29     | (null)           | (null)                         |
| 6  | 1000        | 0             | 1000             | A82     | 7                | 5 or more satisfactory ratings |
| 7  | 1000        | 492           | 1000             | A61     | (null)           | (null)                         |
| 8  | 500         | 321           | 1                | A32     | (null)           | (null)                         |
| 9  | 500         | 380           | 500              | A56     | 1                | File but no rating             |
| 10 | 500         | 0             | 500              | A64     | 98               | File but no rating             |
| 11 | 500         | 471           | 500              | A72     | 98               | 2 satisfactory ratings         |
| 12 | 500         | 310           | 500              | A40     | 98               | 2 satisfactory ratings         |
| 13 | 500         | 331           | 500              | A22     | 99               | No File                        |
| 14 | 1000        | 399           | 500              | A35     | (null)           | (null)                         |
| 15 | 1000        | 399           | 500              | A63     | (null)           | (null)                         |
| 16 | 1000        | 321           | 1                | A48     | (null)           | (null)                         |
| 17 | 1000        | 321           | 1                | A80     | (null)           | (null)                         |
| 18 | 1000        | 321           | 1                | A47     | (null)           | (null)                         |
| 19 | 1000        | 331           | 1                | A98     | (null)           | (null)                         |

Overview Report | Dataset Chart | **Data** | Segment Viewer | Cross Tabs | Correlations

The data can be copied and pasted into Microsoft Office applications. This allows the rapid and easy preparation of reports, presentations and summaries. To copy the data, use the context menu appearing on right mouse click, or

## From the menu

Edit | Select All

Edit | Copy

To paste the data into a Microsoft Office application such as Excel, open the application and paste the data from the clipboard.

## Segment Viewer

The **segment viewer** helps you to visualize the distributions of the independent variables under each category of the segmentation variable. Specifically, you can determine if the distributions of the independent variables vary across the segmented variable. If you can see this variation, then this may be a variable of interest for you in your analysis.

The first time the **segment viewer** is viewed for a dataset, the [Options](#) dialog appears. You can call up [Options](#) manually:



---

## From the menu:

Tools | Options...

## From the toolbar:

Click on the Options icon

Page | 245

The **segment viewer charts** and the associated data can be copied and pasted into Microsoft Office applications. This allows the rapid and easy preparation of reports, presentations and summaries.

See also:

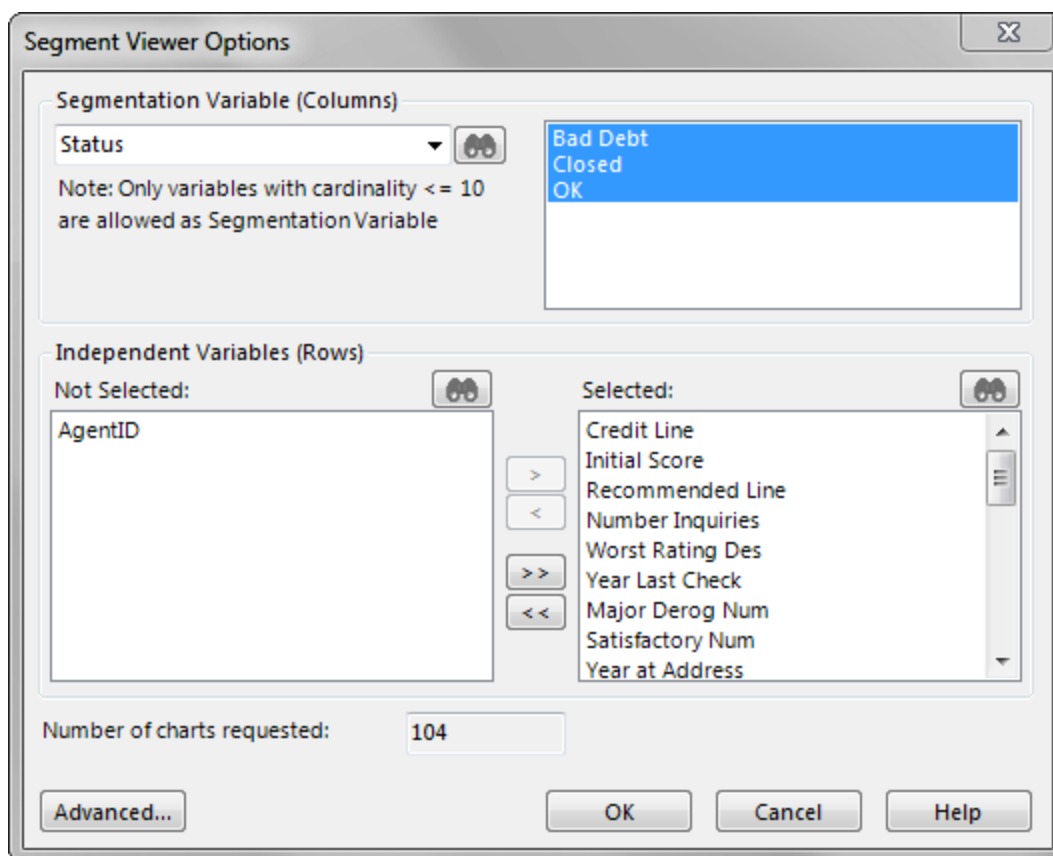
[Segment Viewer: Options](#)

[Segment Viewer Charts](#)

[Copying the Charts](#)

## Segment Viewer Options

The segment viewer options allow you to pick a dependent variable to compare to multiple independent variables.



**Segmentation Variable**

Variable to segment. The categories of the chosen variable are displayed to the right. Charts will be created for the highlighted categories. All the categories are highlighted by default. The segmentation variable should have no more than 10 distinct values.

**Independent Variable**

Variables to compare to the dependent, or segment, variable. One or more independent variables can be selected.

**Number of Charts Requested**

Number of charts that will be displayed in the segment viewer based on the number of categories in the dependent variable multiplied by the number of independent variables chosen.

If the number of charts is greater than 50, a warning message appears. Reduce the number of categories to display for the segmentation variable, if necessary.

Click the Advanced button if you would like to change the binning for the selected variables. See the topic [Segment Viewer: Advanced Options](#) for details.

## Segment Viewer Charts

The segment viewer charts are created based on the parameters specified in the Segment Viewer Options.

The first column, **Whole Dataset**, always displays the distribution of the independent variable for all categories of the segmentation variable. The rest of the columns are the distribution of the independent variable across a each category of the segmentation variable.

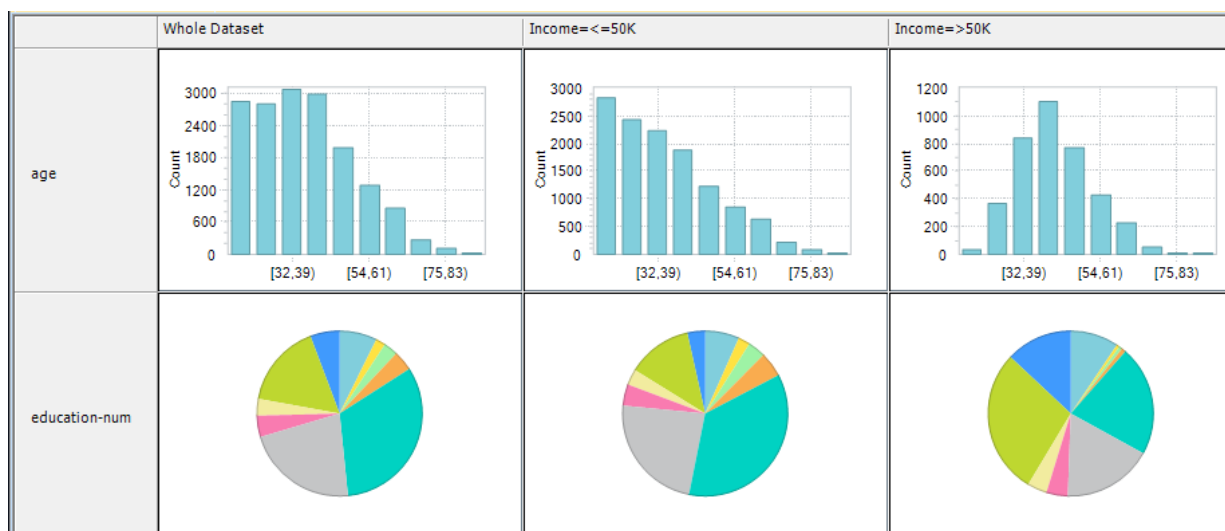
Page | 247

All non-integer numeric variables regardless of their cardinality are always treated as continuous and displayed in the form of bar charts by default.

Integer and date/time variables are displayed in the form of bar charts if their cardinality is higher than the threshold defined in the [Advanced Segment Viewer Options](#). Otherwise, they are considered discrete and are displayed in the form of pie charts.

String and Boolean variables are always displayed as pie charts by default. To change the pie chart settings such as the maximum number of bins displayed, use the [Advanced Segment Viewer Options](#).

Figure 1



In Figure 1, we have the segmentation variable **Income** that contains the categories **<=50K** and **>50K** and the independent variable **Age**.

The first column, **Whole Dataset**, displays the distribution of **Age** for both the **<=50K** and **>50K** categories. You can see this same distribution of **Age** in the **Dataset Chart** (given the same default settings).

Trends can be observed with these charts. For example:

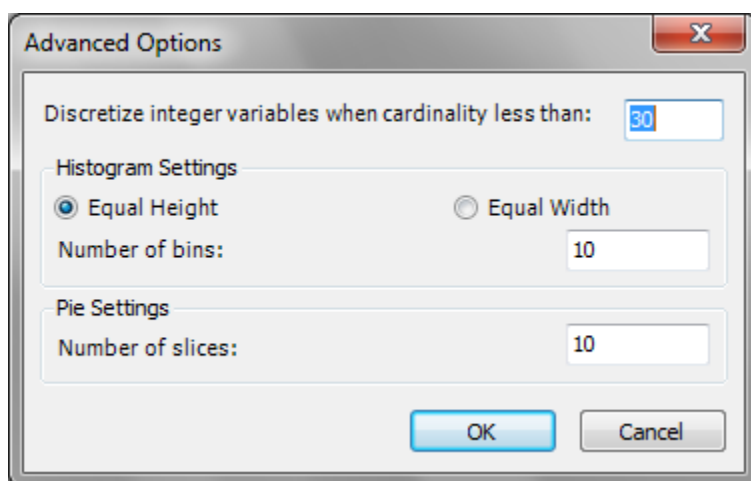
- Compare the distribution of **<=50K** with the whole dataset. Visually the distributions do not look too different.
- Compare the distribution of **>50K** with the whole dataset. Visually you can see a distinct variation with the distributions. Specifically, people who make >50K tend to be older. Although this observation may be intuitive, it illustrates how the segment viewer can show you, visually, the differences in the distributions.

Page | 248

There may be more charts displayed than can fit in the view. Use the **Fit to Page**  button in the toolbar to fit all charts to the width of the Segment Viewer tab.

## Advanced Options

The segment viewer settings allow you to change how the charts are displayed.



**Discretize integer variables when cardinality is less than:**

The distribution of integer variables whose cardinality is less than or equal to this number will be displayed in the form of bar charts (histograms) by default. Otherwise, it will be displayed as pie charts. The default is 30.

## Histogram Settings

### **Equal Height**

Determines how the data is distributed in the chart. When this is checked, the data is distributed so that each bin contains approximately the same number of observations.

---

|                              |                                                                                                                                                    |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Equal Width</i></b>    | Determines how the data is distributed in the chart. When this is checked, the width of the intervals on the x-axis will be the same for all bins. |
| <b><i>Number of bins</i></b> | Number of bins in which to distribute the data. The default is 10.                                                                                 |

## Pie Settings

|                                |                                                                                                                                                                                                               |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Number of slices</i></b> | Sets the number of categories to display on the pie chart. If the number of categories is greater than this number, then the remaining categories are combined into a category called ###. The default is 10. |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Copying all Charts

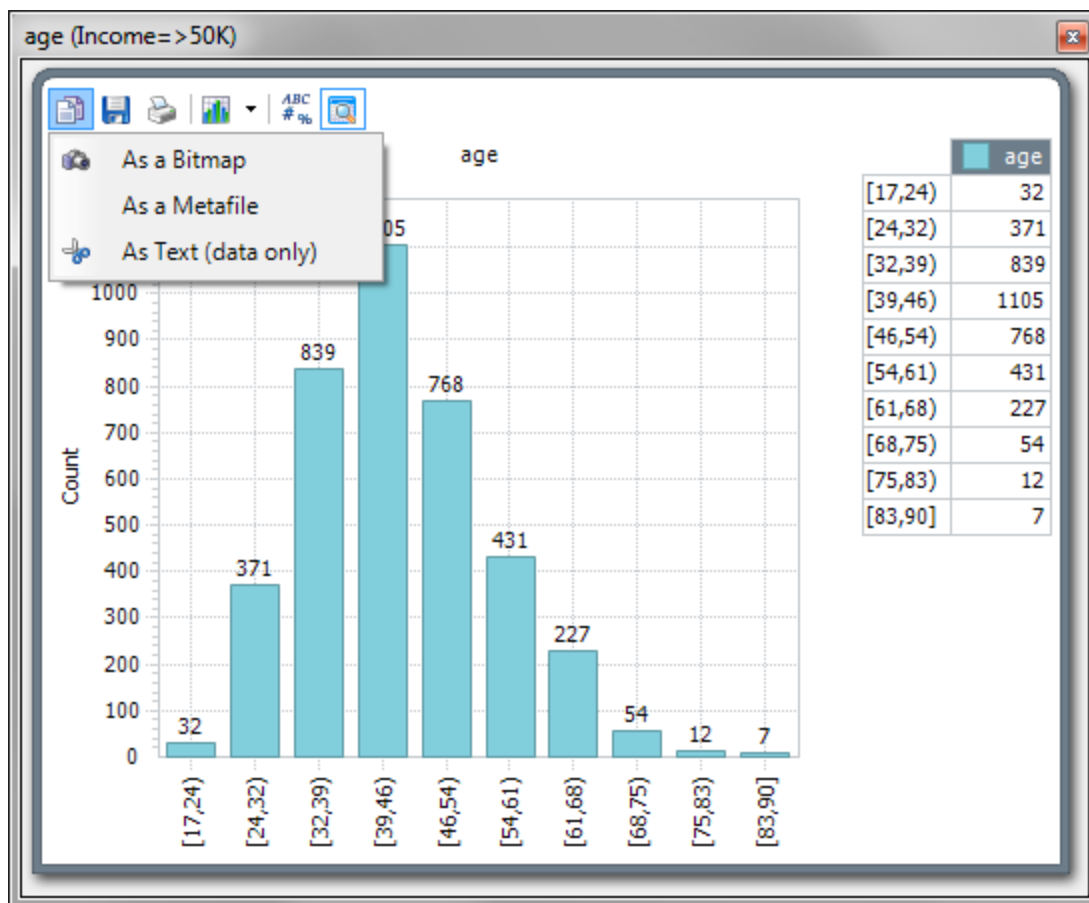
To copy all the charts that are visible:

1. Right-click on the segment viewer workspace
2. From the popup, choose **Copy**
3. Paste into any Office application

Alternatively, you can use the **Edit | Copy** menu command.

## Copying Individual Charts

1. Double-click on a chart in the segment viewer workspace. This will bring up the chart in a separate window.
2. Click the top left icon ("Copy to Clipboard")



1. Choose the type of copy

|                            |          |                                                                                                                                |
|----------------------------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| <b>As Bitmap</b>           | <b>a</b> | The chart is copied into the clipboard as a bitmap image.                                                                      |
| <b>As Metafile</b>         | <b>a</b> | The chart is copied into the clipboard as a metafile. A metafile is a mechanism for retaining and transporting graphical data. |
| <b>As Text (data only)</b> |          | Only the chart data is copied (in the tabular form).                                                                           |

## Cross Tabs

The **Cross Tabs** view helps you to visualize your data in cross tables between two or more variables. The cross tables can be viewed in the chart form or a tabular form (report mode).

You can switch between the [chart mode](#) and the [report mode](#) by clicking the Show As Report icon  in the toolbar or selecting **View | Show As Report** from the menu.

You can choose more than one variable to be shown on each axis of the chart.

Page | 251

The cross tabs chart and the associated data can be copied and pasted into Microsoft Office applications for quick and easy preparation of reports, presentations, and summaries.

The first time you navigate to the **Cross Tabs** view of a dataset, the Cross Tabs Options dialog appears automatically. It guides you through the process of creating a cross table chart and report.

This section deals with specifying the necessary parameters to create the Cross Tabs chart and report. The resulting chart and report are described in the topics [Cross Tabs: Chart](#) and [Cross Tabs: Report](#).

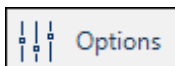
If the options have already been defined and the chart is displayed, you can modify the Cross Tabs settings as follows:

### From the menu, select

Tools | Options...

### From the toolbar

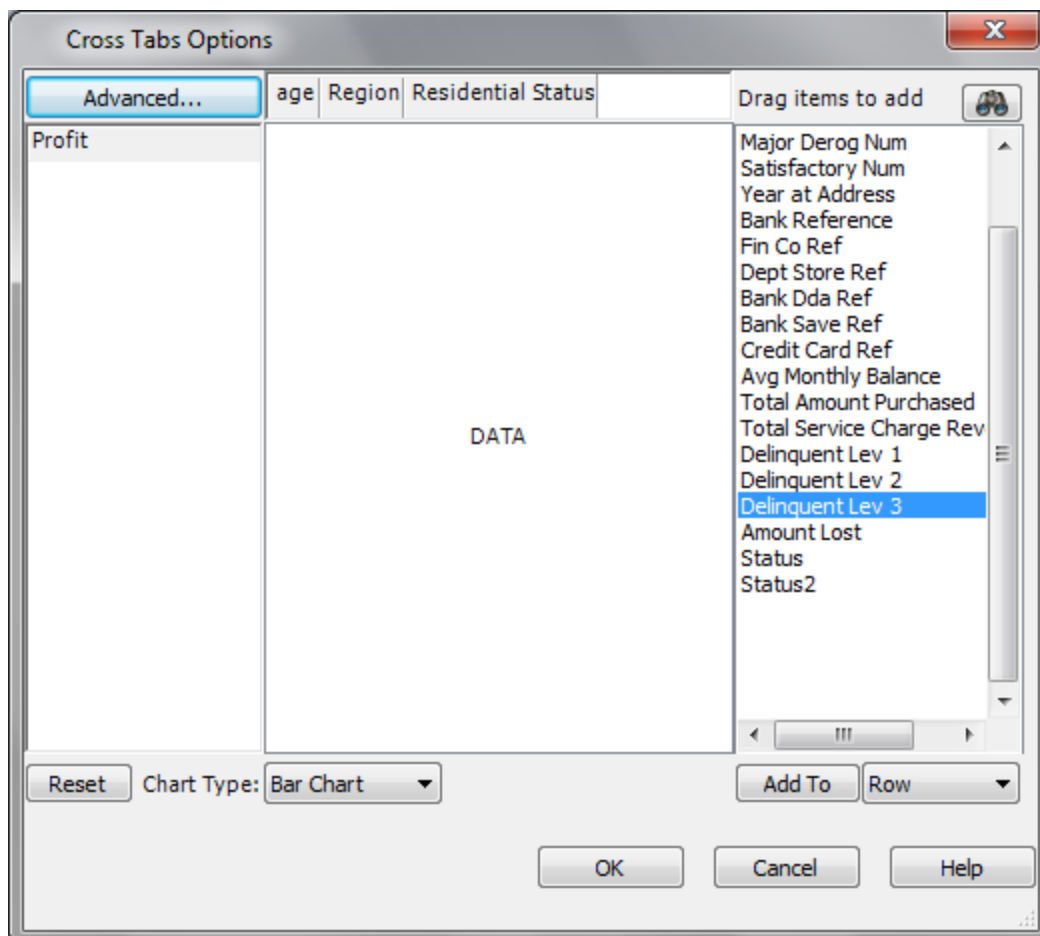
Click on the icon



The Cross Tabs Options dialog prompts you to select one or more variables for each axis (**ROW** and **COLUMN** sections of the grid represent the axes), specify the measure to be displayed for each Cross Tabs cell (in the **DATA** section), specify the chart type, and optionally specify advanced binning options for each variable.

The right panel of the dialog contains the list of available variables. Advanced variable selection, which is especially useful if you have a large number of variables, can be performed using the Field Finder (launched by clicking the binoculars button ).

The dialog supports a drag and drop interface similar to Microsoft Excel Pivot Table layout dialog.



- To add a variable to any of the dimensions (axes), drag it from the list of variables to the desired dimension (ROW or COLUMN sections). To remove a variable from a section, drag it back to the list of variables or any other section of the grid.
- You can drag any **numeric** variable onto the DATA portion of the grid. If no variable is selected for the DATA portion of grid, the assumed measure is **record count**, which is displayed for each Cross Tabs cell. If you add a variable to the DATA section, the default measure is assumed to be the average of that variable in each Cross Tabs cell. To change the measure by specifying a different aggregate function, see [Advanced Options](#).
- You can add more than one numeric variable into the DATA section, thus specifying several measures. In this case, when viewing the results, you will be able to choose which measure to display (one at a time).
- To add several variables at once (to any section of the grid), first highlight them by pressing and holding the SHIFT or CTRL key and selecting the desired fields with the mouse. From the list box beneath the list of variables, choose where you want to add the variables to (Row, Column, or Data) and click the **Add To** button.
- To remove variables from all sections back to the variable list, click the **Reset** button in the bottom left corner of the dialog.



If you leave the ROW or COLUMN axis blank, a simple one-dimensional result is produced which is the same as charts in the [Dataset Chart](#) view.

Double-clicking on a variable added to the ROW or COLUMN section brings up the Advanced Options dialog, which allows you to specify advanced binning options for the variable. Alternatively, you can open the Advanced Options dialog by clicking the **Advanced** button at the top left corner of the dialog. See [Advanced Options](#) for details.

## Chart Type

**Bar Chart** is the default chart type. The other options are **Surface Chart** and two-dimensional **Scatter Plot**.

You can specify a measure other than record count (by adding a variable to the DATA section) only if either Bar Chart or Surface Chart is selected.

The Scatter Plot option requires that the following conditions be satisfied:

1. Only one variable per axis
2. Both axis variables must be numeric
3. No Data field specified

See also:

[Cross Tabs: Advanced Options](#)

[Cross Tabs: Chart](#)

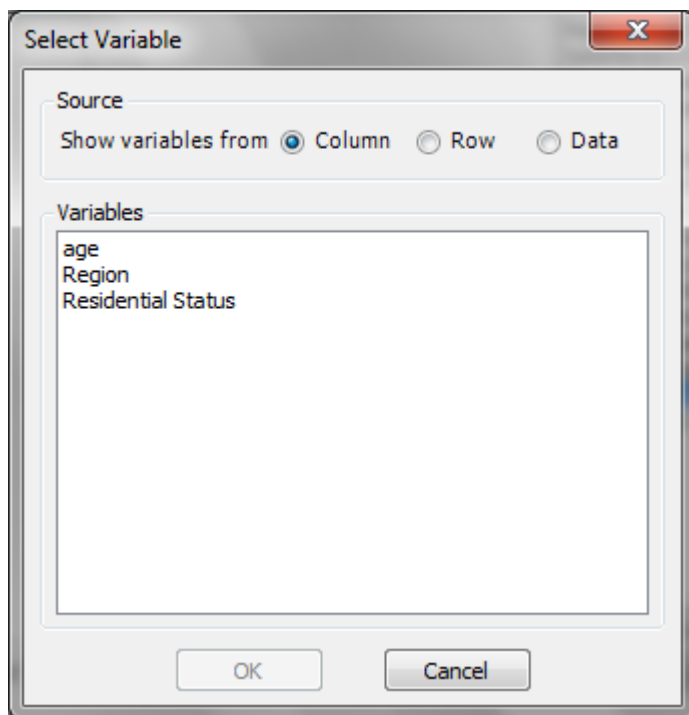
[Cross Tabs: Report](#)

Using the Advanced Options of the [Cross Tabs Options](#) dialog, you can

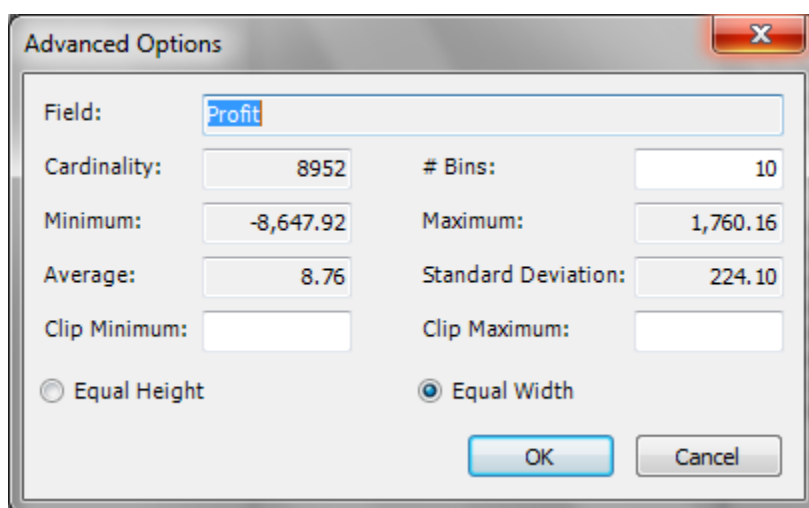
1. specify custom binning for continuous variables
2. specify custom grouping for the categories of discrete variables
3. specify an aggregate function of your choice for the selected measure (determined by the variable(s) in the DATA section, if any)

In any of the above cases, double-click the desired variable in the ROW, COLUMN or DATA sections of the grid.

Alternatively, click the **Advanced** button at the top left corner of the dialog, choose a source (Row, Column, or Data) and then select the desired variable from the selected source:

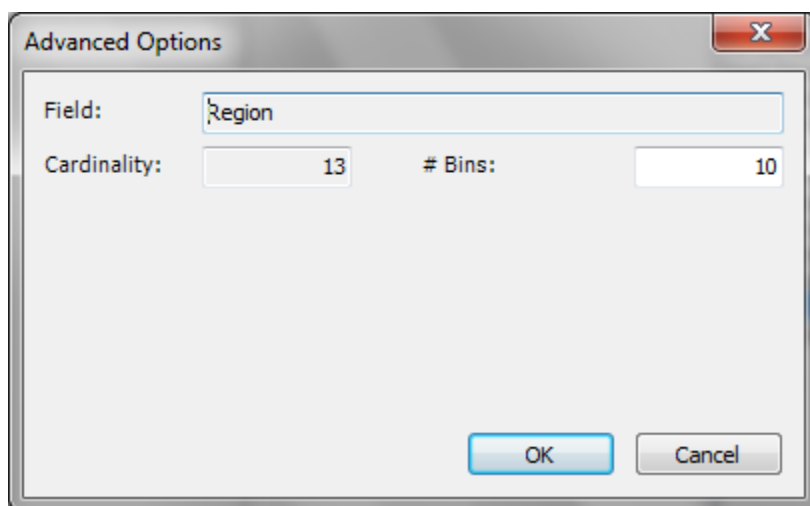


1. If you are changing the binning for a **continuous variable**, a dialog of the following form appears:



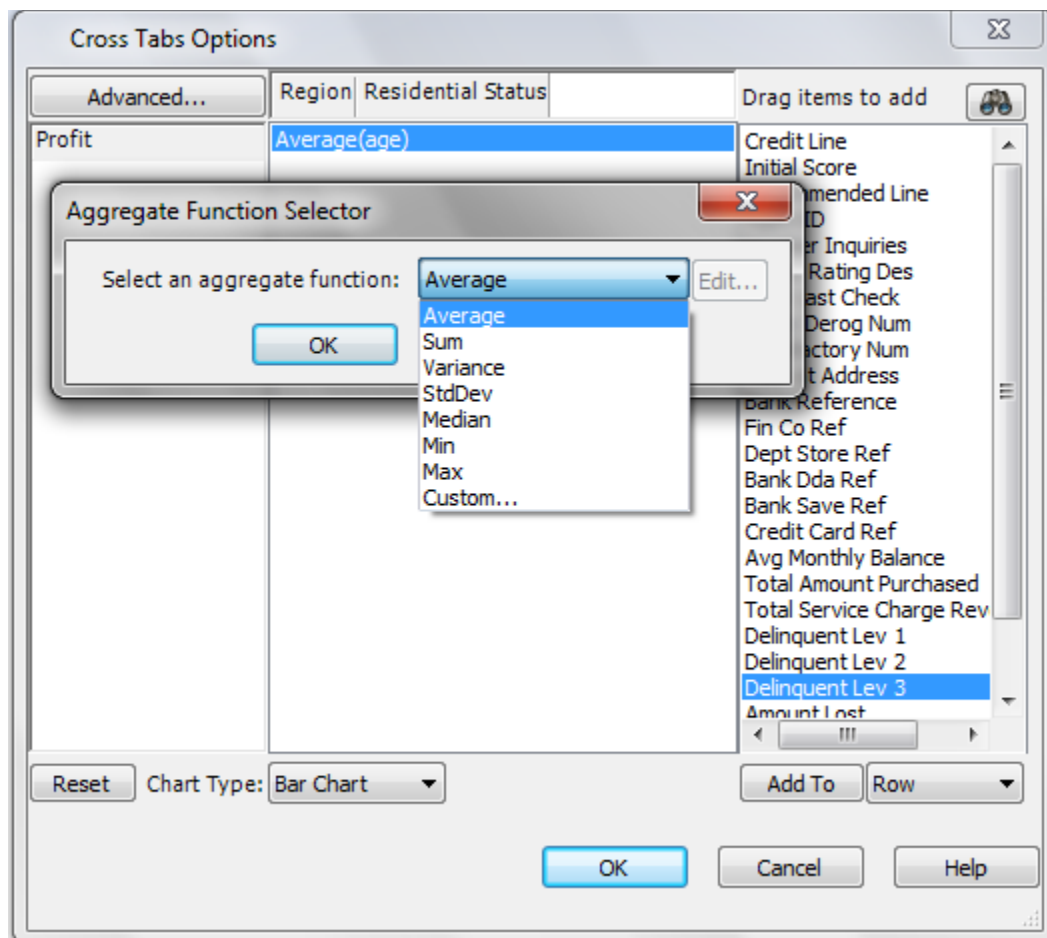
|                           |                                                                                                                                                                 |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Equal Width</b>        | If selected, all intervals (bins) for the selected variable will have the same width.                                                                           |
| <b>Equal Height</b>       | If selected, the data in the intervals (bins) for the selected variable is distributed so that each bin contains approximately the same number of observations. |
| <b>Cardinality</b>        | The cardinality (the number of unique values) of the selected variable.                                                                                         |
| <b># of bins</b>          | The number of bins (intervals) in the displayed range of the selected variable.                                                                                 |
| <b>Minimum</b>            | The minimum value of the selected variable in the whole dataset.                                                                                                |
| <b>Maximum</b>            | The maximum value of the selected variable in the whole dataset.                                                                                                |
| <b>Average</b>            | The average of the selected variable in the whole dataset.                                                                                                      |
| <b>Standard Deviation</b> | The standard deviation of the selected variable in the whole dataset.                                                                                           |
| <b>Clip Minimum</b>       | Sets the left boundary for the displayed range of the selected variable. All values smaller than this value (if any) will be ignored.                           |
| <b>Clip Maximum</b>       | Sets the right boundary for the displayed range of the selected variable. All values larger than this value (if any) will be ignored.                           |

2. If you are changing the grouping for the categories of a **discrete variable**, a dialog of the following form appears:



You can reduce the number of bins if necessary. Categories with the least frequencies will be grouped first.

3. If you are **changing the measure** by changing the aggregate function for the variable in the DATA section, the following dialog appears:



Here you can choose between Average, Sum, Variance, Standard Deviation, Median, Minimum, Maximum, or Custom aggregate functions.

If Custom is selected, you can enter a custom aggregate expression of any complexity using the [Expression Editor](#), which opens automatically. After you finish specifying your custom aggregate expression and close Expression Editor, you can return to the expression by clicking the Edit button.

See also:

[Cross Tabs Overview and Options](#)

[Cross Tabs: Chart](#)

[Cross Tabs: Report](#)

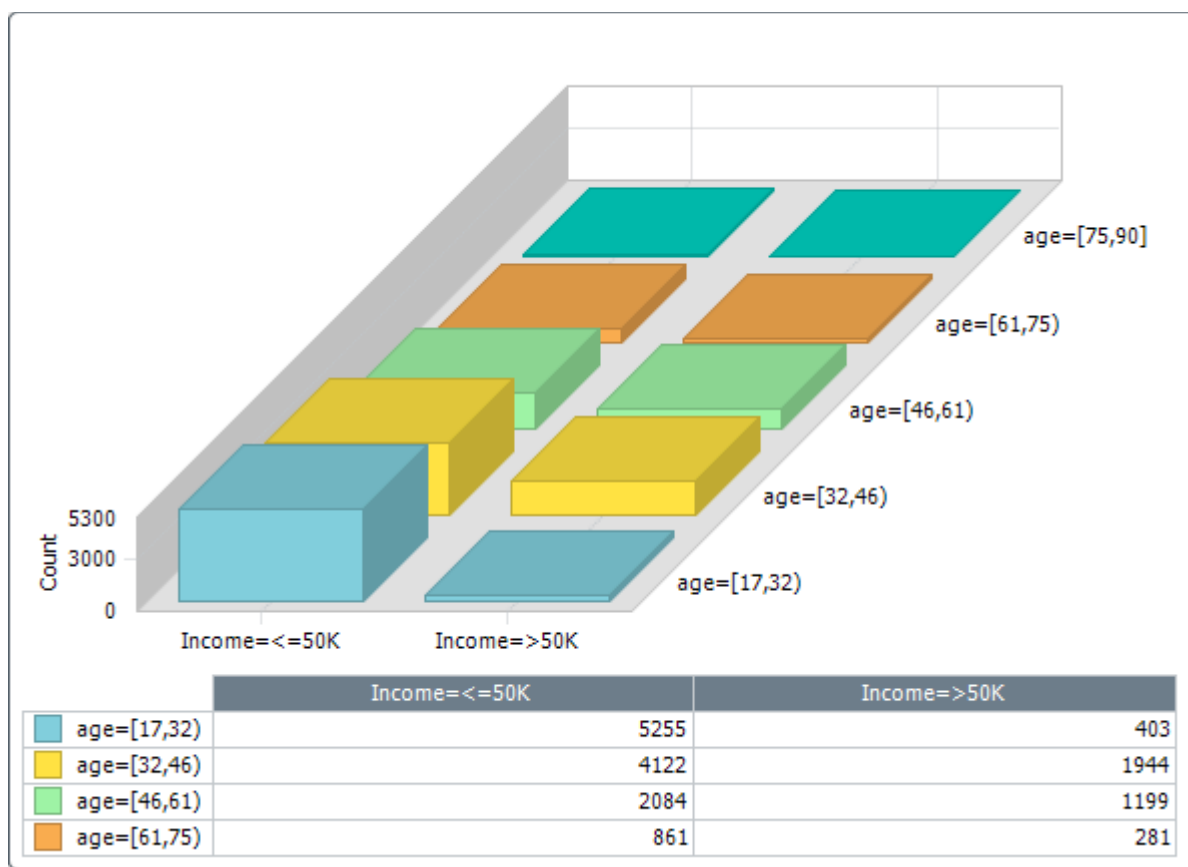
## Cross Tabs Chart

The Chart mode of the **Cross Tabs** view displays the cross tables between two or more variables in the form of a Bar Chart, Surface Chart, or Scatter Plot based on the parameters specified in the [Cross Tabs Options](#) dialog.

Page | 257

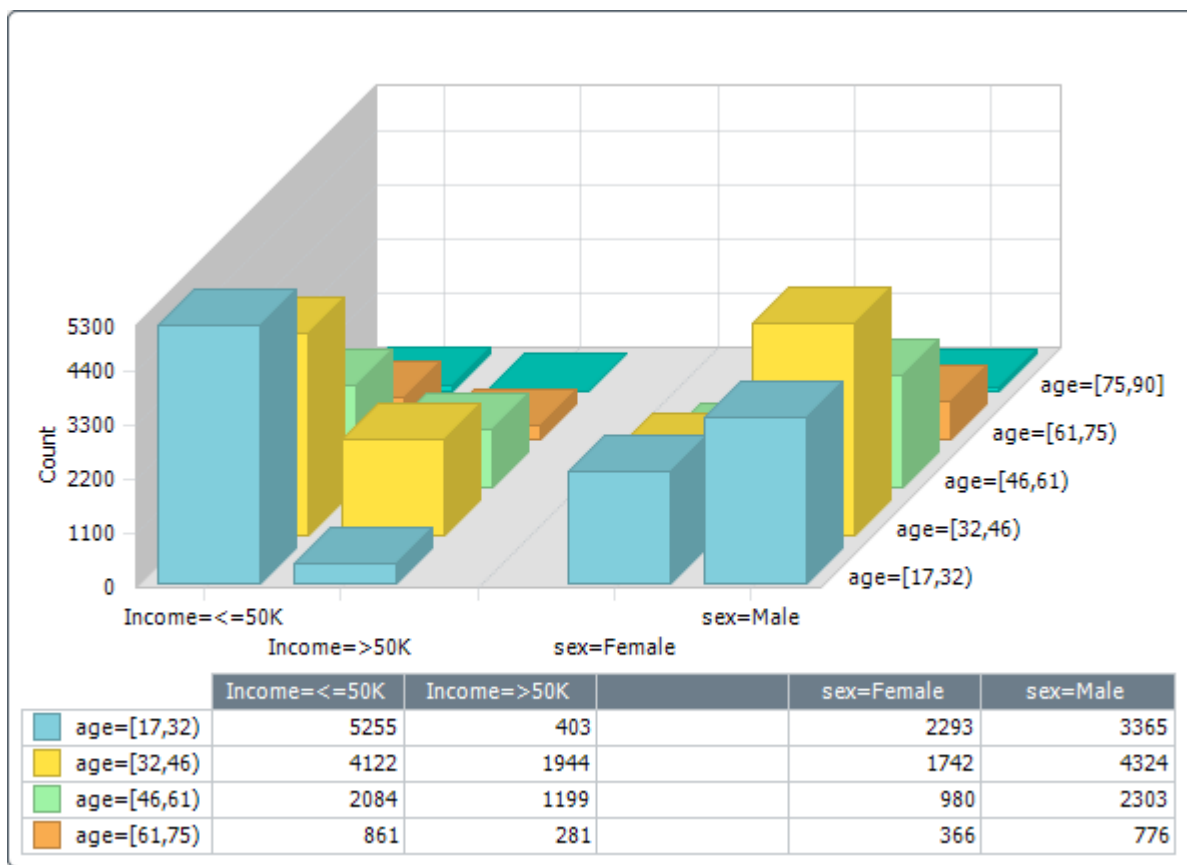
**Note:** You can switch between the chart mode and the [report mode](#) by clicking the Show As Report icon  in the toolbar or selecting **View | Show As Report** from the menu.

In this example, the chart displays the number of people in each **Income** category for each **Age** interval. From this chart we can see that the majority of people whose income is above \$50000 are in the age group between approximately 26 and 54.



If you choose more than one variable for one of the dimensions (axes), they will be separated by an empty row in the chart for clarity, as shown in the following example.

Here **Income** and **sex** variables were added to the ROW section in the [Cross Tabs Options](#) dialog:



If you would like to change the binning for the continuous variables, group some categories of the discrete variables, or specify a different measure, open the [Cross Tabs Options](#) and double-click on the desired variable. See [Advanced Options](#) for details.

See also:

[Cross Tabs Overview and Options](#)

[Cross Tabs: Report](#)

## Cross Tabs: Report

The Report mode of the **Cross Tabs** view displays the cross tables between two or more variables in the form of a tabular report based on the parameters specified in the [Cross Tabs Options](#) dialog.

**Note:** You can switch between the report mode and the [chart mode](#) by clicking the Show As

Report icon  in the toolbar or selecting **View | Show As Report** from the menu.

☐ Show Row Totals
 ☐ Show Column Totals
 Show: Values
 Plot Variable:

|                                   | AccountStatus=Bad Debt | AccountStatus=OK |  |
|-----------------------------------|------------------------|------------------|--|
| Credit Line=[0.00,14,019.60)      | 1424                   | 27739            |  |
| Credit Line=[14,019.60,28,039.20) |                        | 2                |  |
| Credit Line=[28,039.20,42,058.80) |                        | 1                |  |
| Credit Line=[42,058.80,56,078.40) |                        |                  |  |
| Credit Line=[56,078.40,70,098.00) |                        | 1                |  |
|                                   |                        |                  |  |
| age=null                          | 17                     | 605              |  |
| age=[17.00,37.25)                 | 979                    | 14297            |  |
| age=[37.25,57.50)                 | 363                    | 9028             |  |
| age=[57.50,77.75)                 | 57                     | 3305             |  |
| age=[77.75,98.00]                 | 8                      | 508              |  |
|                                   |                        |                  |  |

[Overview Report](#) | 
 [Dataset Chart](#) | 
 [Data](#) | 
 [Segment Viewer](#) | 
 [Cross Tabs](#) | 
 [Correlations](#)

You can choose to show values, column percentages, or row percentages.

☐ Show Row Totals
 ☐ Show Column Totals
 Show: Column Percentages
 Plot Variable:

|                                   | AccountStatus=Bad Debt | AccountStatus=OK |  |
|-----------------------------------|------------------------|------------------|--|
| Credit Line=[0.00,14,019.60)      | 100%                   | 99.99%           |  |
| Credit Line=[14,019.60,28,039.20) |                        | 0.01%            |  |
| Credit Line=[28,039.20,42,058.80) |                        | 0%               |  |
| Credit Line=[42,058.80,56,078.40) |                        |                  |  |
| Credit Line=[56,078.40,70,098.00) |                        | 0%               |  |
|                                   | <b>100%</b>            | <b>100%</b>      |  |
| age=null                          | 1.19%                  | 2.18%            |  |
| age=[17.00,37.25)                 | 68.75%                 | 51.53%           |  |
| age=[37.25,57.50)                 | 25.49%                 | 32.54%           |  |
| age=[57.50,77.75)                 | 4%                     | 11.91%           |  |
| age=[77.75,98.00]                 | 0.56%                  | 1.83%            |  |
|                                   | <b>100%</b>            | <b>100%</b>      |  |

[Overview Report](#) | 
 [Dataset Chart](#) | 
 [Data](#) | 
 [Segment Viewer](#) | 
 [Cross Tabs](#) | 
 [Correlations](#)

## Characteristic Analysis

The Characteristic Analysis view shows the frequencies of each category of a dependent variable across various ranges (bins) of independent variables. This view helps to reveal relative trends in the distribution of the DV categories by plotting them side by side, making it easy to compare the trend of one category against the others.

---

The options dialog provides a choice of displaying charts and report as percent of column or percent of rows. The default is percent of column.

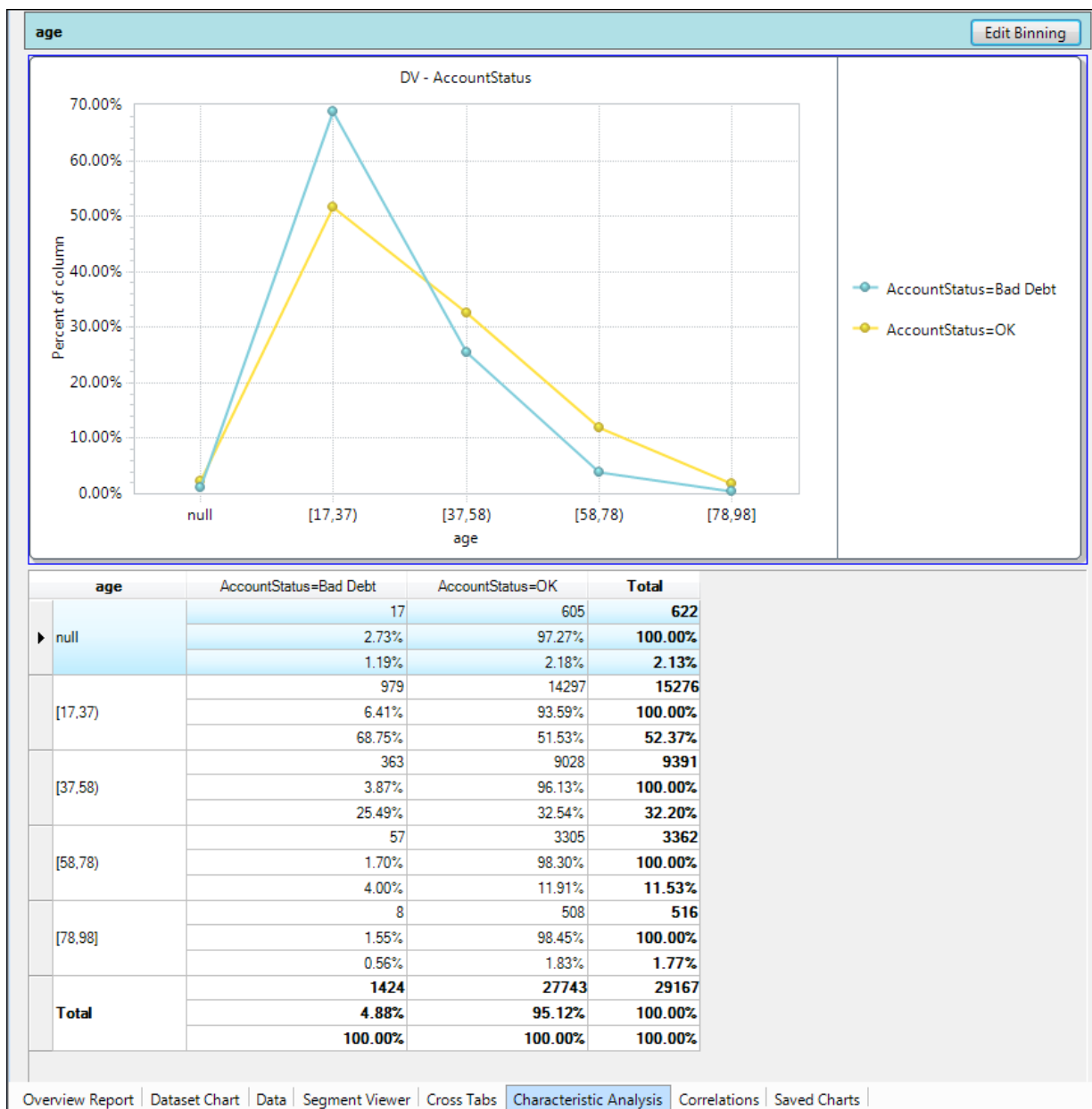
The initial chart is displayed based on the dependent variable selected in [Segment Viewer](#). You can change the dependent variable in the Options dialog.

Page | 260

All graphs are line charts by default.

You can change the grouping of the independent variable by clicking the Edit Binning button at the top right corner. This brings up the familiar Range Editor dialog.





In the example above, we see consistently similar trends of both categories of the dependent variable (Account Status) across the selected age ranges.

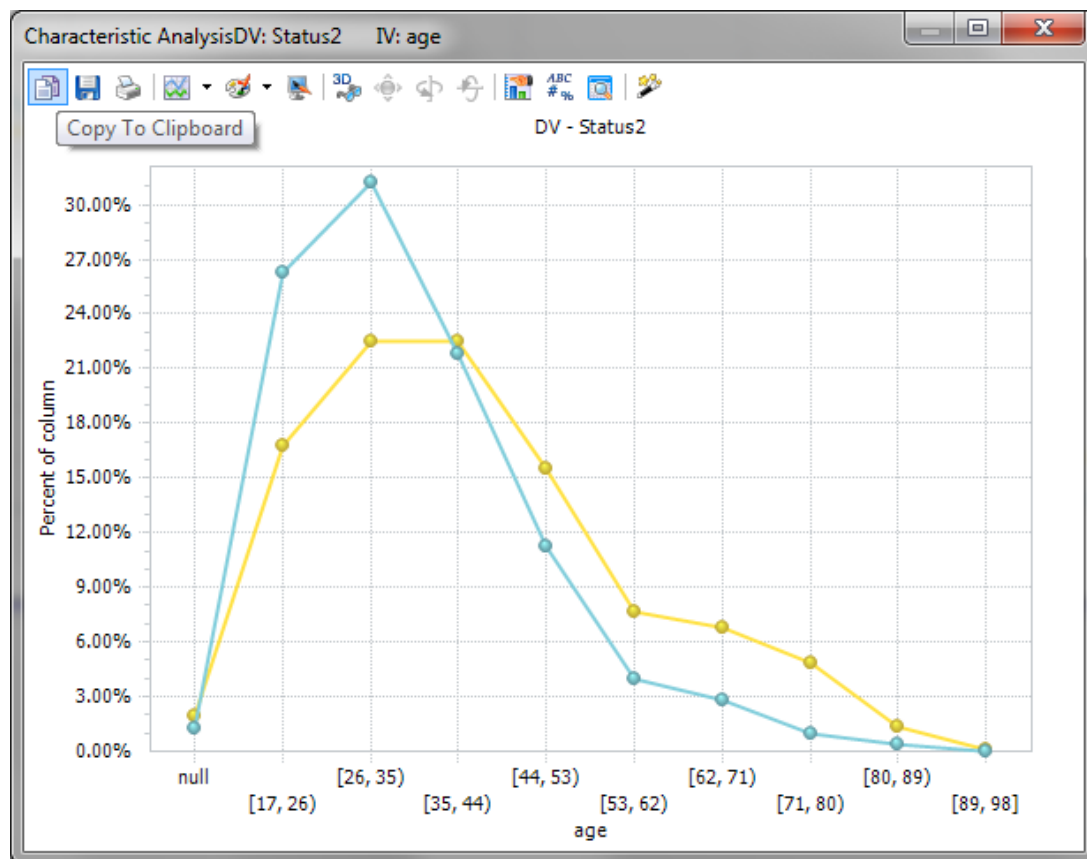
In the example below, we see that the High and Low income trends go against each other in the first two age groups [17,35) and [35,54), whereas they start to go together after age 54.



Thus, by looking closer into the behavior of the DV categories within various segments of a variable, Characteristic Analysis may help detect interesting relationships between this variable and the DV even when other methods do not rank this variable as highly significant.

All charts and tables in this view can be easily copied and pasted into any Microsoft Office application or saved as PDF.

- To copy a chart: double-click the desired chart to open it in a separate window and click the Copy icon in the toolbar at the top of the chart.



- To save the whole Characteristic Analysis view as PDF with all of its charts and tables, select the **Copy to PDF** command from the **File** menu.
- To copy a table, right-click on any cell and choose **Select All** from the context menu. Then right-click and select **Copy**.

| age        |  | Status2=Bad Debt | Status2=OK | Total    |
|------------|--|------------------|------------|----------|
| null       |  | 5682             | 195955     | 201637   |
|            |  | 2.82%            | 97.18%     | 100.00%  |
|            |  | 1.23%            | 1.92%      | 1.89%    |
| [17, 44)   |  | 366277           | 6314085    | 6680362  |
|            |  | 5.48%            | 94.52%     | 100.00%  |
|            |  | 79.37%           | 61.73%     | 62.49%   |
| [44, 71)   |  | 83219            | 3077004    | 3160223  |
|            |  | 2.63%            | 97.37%     | 100.00%  |
|            |  | 18.03%           | 30.08%     | 29.56%   |
| ▶ [71, 98] |  | 6276             | 641151     | 647427   |
|            |  | 0.97%            | 99.03%     | 100.00%  |
|            |  | 1.36%            | 6.27%      | 6.06%    |
| Total      |  | 461454           | 10228195   | 10689649 |
|            |  | 4.32%            | 95.68%     | 100.00%  |
|            |  | 100.00%          | 100.00%    | 100.00%  |

## Correlations

The **Correlations** view of the dataset object allows you to explore how or to what extent fields in your dataset are associated with each other. It displays *correlation coefficients* (default option) or *covariances* for the fields you select.

You have an option of displaying these statistics in the matrix form, in pairs, or in a chart. See the topic Correlations View for more details.

Covariance and Pearson correlation coefficients measure the degree of linear association between pairs of variables.

The values of correlation coefficients are always between -1 and +1. The value of +1 indicates that two fields are perfectly related in a positive linear sense, -1 indicates that they are perfectly related in a negative linear sense, and 0 means there is no linear relationship between the two fields.

The correlation coefficient of two variables is obtained by dividing their covariance by the product of their standard deviations.

Spearman's correlation is a special case of the Pearson correlation applied to the ranks of the variable values rather than the values themselves. It is a measure of monotone association between two variables.

Missing values are handled on a pairwise basis, i.e., the results of the correlations or covariance computation treats missing values in a pairwise mode. For a given row, as long as the two intersecting values are not missing, the numbers contribute to the calculation.

---

When you select the Correlations tab of a dataset for the first time, the Options dialog appears automatically, allowing you to select the desired fields, the type of statistics to be displayed, and the way to treat discrete and continuous variables (see [Options](#)) . You can also call up the [Options](#) dialog manually:

## From the menu

Tools | Options...

## From the toolbar

Click on the Options icon

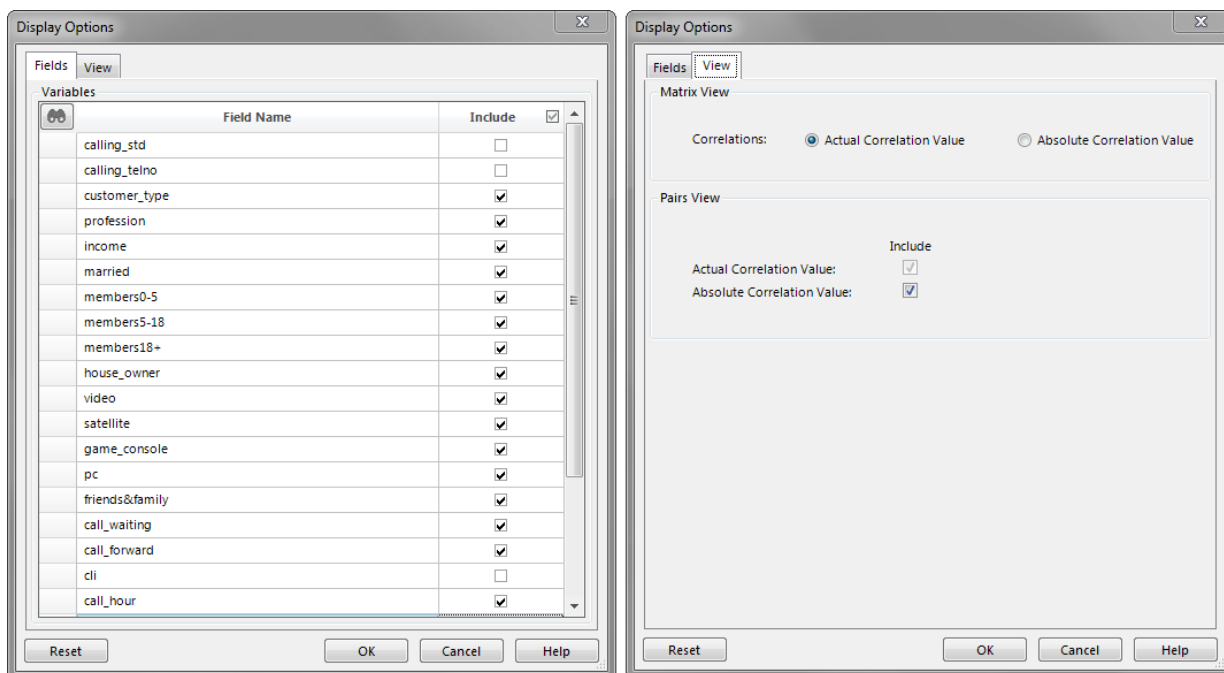
Page | 265

See the topic [Correlations View](#) for detailed description of the view.

The Display Options dialog for the Correlations view allows you to select the desired fields and the additional statistics to be displayed. For general information about the Correlations view, see the topic [About the Correlations View](#).

Select the fields to analyze for correlation. If you want to select or deselect all fields, click the checkbox in the **Include** column header. Note that the calculation of correlations for a very large number of variables may take substantial time and memory resources.

The attributes and options displayed in the Correlations Display Options dialog are as follows. Select the desired variables in the Fields tab, then click the **View** tab to select whether to display actual or absolute correlation values in the Matrix View and whether to include absolute correlation values in the Pairs view.



## Reset

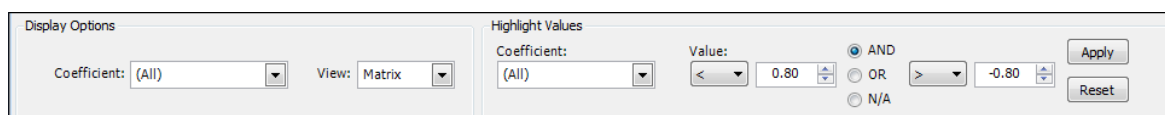
The **Reset** button undoes the changes you made in the Display Options dialog, i.e., resets the settings to the those of the currently displayed correlations view or the application defaults.

See also: [About the Correlations View](#)

The correlation coefficients are shown based on the parameters specified in the [Display Options](#) dialog (see the topic [Correlations: Options](#) for details).

The **View** drop down list provides three ways to display correlation coefficients: *Matrix*, *Pairs*, and *Chart*.

From the **Coefficient** list, select the type of coefficients to display (Pearson, Spearman, Kendall's Tau-b, or Hoeffding's).



- **Matrix View**

Shows the correlation or covariance matrix for the selected fields.

|                        | WOE_Residential Status | WOE_Initial Score | WOE_Region | WOE_age    |
|------------------------|------------------------|-------------------|------------|------------|
| WOE_Residential Status | P: 1                   | P: 0.31931        | P: 0.07141 | P: 0.12701 |
|                        | S: 1                   | S: 0.49013        | S: 0.01134 | S: 0.17247 |
|                        | K: 1                   | K: 0.40827        | K: 0.0104  | K: 0.14673 |
|                        | H: 0.27447             | H: 0.05062        | H: 5E-05   | H: 0.0061  |
| WOE_Initial Score      | P: 0.31931             | P: 1              | P: 0.24811 | P: 0.35921 |
|                        | S: 0.49013             | S: 1              | S: 0.31464 | S: 0.35671 |
|                        | K: 0.40827             | K: 1              | K: 0.28344 | K: 0.29778 |
|                        | H: 0.05062             | H: 0.41376        | H: 0.00824 | H: 0.02959 |
| WOE_Region             | P: 0.07141             | P: 0.24811        | P: 1       | P: 0.03486 |
|                        | S: 0.01134             | S: 0.31464        | S: 1       | S: 0.04879 |
|                        | K: 0.0104              | K: 0.28344        | K: 1       | K: 0.04379 |
|                        | H: 5E-05               | H: 0.00824        | H: 0.037   | H: 0.00016 |
| WOE_age                | P: 0.12701             | P: 0.35921        | P: 0.03486 | P: 1       |
|                        | S: 0.17247             | S: 0.35671        | S: 0.04879 | S: 1       |
|                        | K: 0.14673             | K: 0.29778        | K: 0.04379 | K: 1       |
|                        | H: 0.0061              | H: 0.02959        | H: 0.00016 | H: 0.46593 |

### ○ Pairs

Each pair of variables and their correlation or covariance is displayed in a single row.

|   |                        |                   | Pearson     |          |         |         |         |
|---|------------------------|-------------------|-------------|----------|---------|---------|---------|
|   | Variable 1             | Variable 2        | Correlation | Absolute | P-value | LCL     | UCL     |
| ► | WOE_Residential Status | WOE_Initial Score | 0.31931     | 0.31931  | 0       | 0.30896 | 0.32957 |
|   | WOE_Residential Status | WOE_Region        | 0.07141     | 0.07141  | 0       | 0.05998 | 0.08281 |
|   | WOE_Residential Status | WOE_age           | 0.12701     | 0.12701  | 0       | 0.1157  | 0.13828 |
|   | WOE_Initial Score      | WOE_Region        | 0.24811     | 0.24811  | 0       | 0.23731 | 0.25885 |
|   | WOE_Initial Score      | WOE_age           | 0.35921     | 0.35921  | 0       | 0.34917 | 0.36916 |
|   | WOE_Region             | WOE_age           | 0.03486     | 0.03486  | 0       | 0.0234  | 0.04632 |

You can select a specific variable to view its correlations with other fields.

### ○ Chart

Correlation or covariance matrix is displayed in the form of a chart. The variables are displayed along the horizontal axes and their correlations or covariances are shown on the vertical axis. You can choose from different chart types and display options from the Chart menu or from the context menu brought up by right-clicking on the chart.

You have an option of displaying only the charts for a particular variable (chosen from the *Variable* drop-down list) or ignoring the correlation or covariance of a variable with itself (the Suppress Same Column checkbox).

- 1.
2. Having identified and highlighted the desired variables based on the correlation analysis in this view, you can run the **Insert | Variable List** command to save them in a [variable list](#) for future use. For example, if you want to use only the selected variables to train a model, use the field finder button  when selecting the independent variables for the model and import the saved variable list.

Page | 268

See also: [About the Correlations View](#), [Correlations: Options](#),

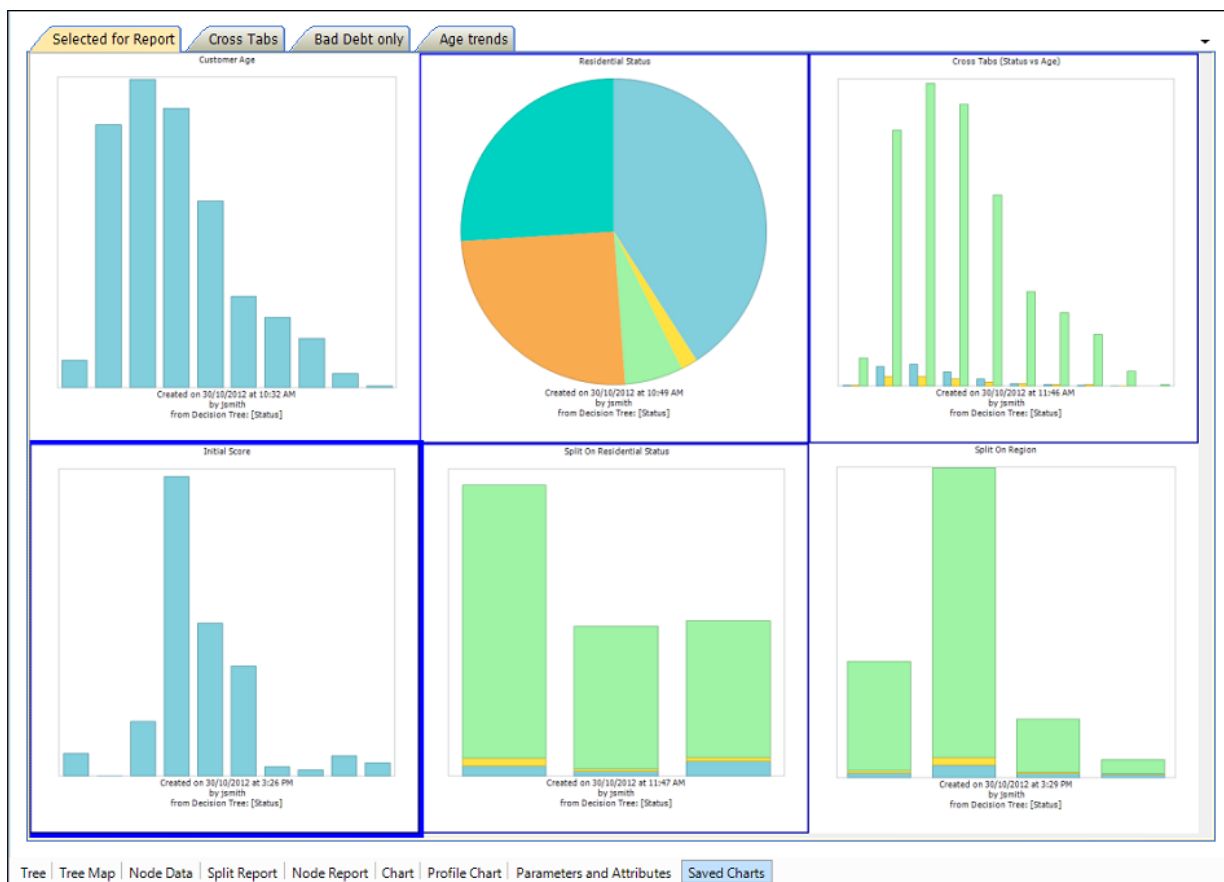
## Saved Charts

The Saved Charts tab is available for datasets, decision trees, and strategy trees. This tab represents the library of charts you saved in other views of the object it belongs to.

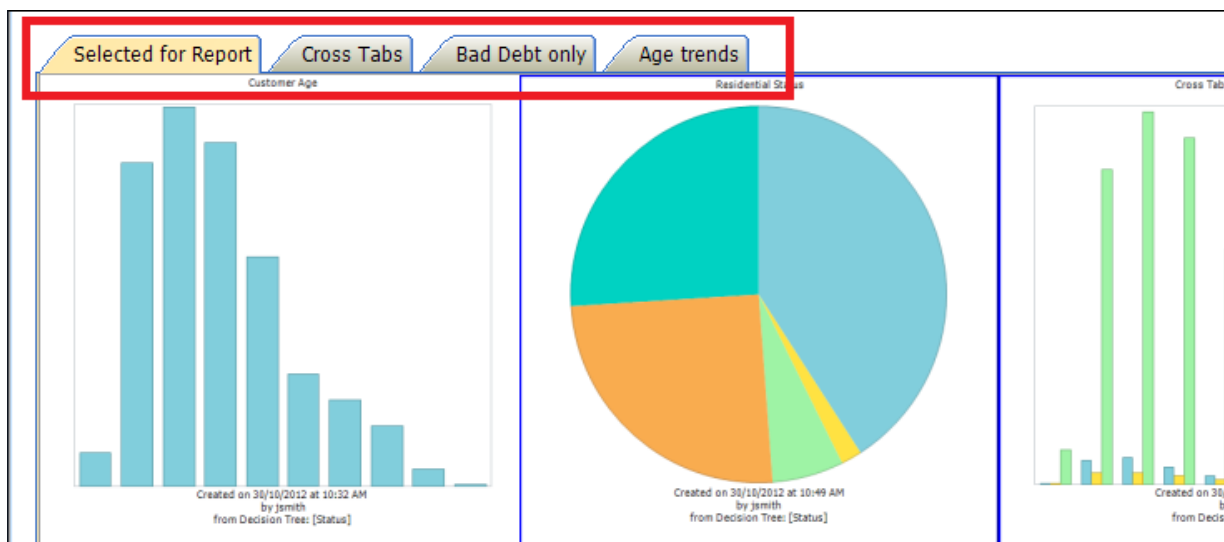
- The Saved Charts tab of a dataset contains charts saved in the Dataset Chart, Segment Viewer, Cross Tabs, and Correlations views of the dataset.
- The Saved Charts tab of a tree contains node charts saved in the Chart view of the tree.

In the Saved Charts tab you can view, organize, and print the charts, or copy them to Microsoft Office documents for reporting.





Charts are organized in groups. Each group is represented by a separate tab at the top of the chart grid. The tabs are highlighted by the red frame in the example screenshot below. Click on the tab names to switch between the chart groups.

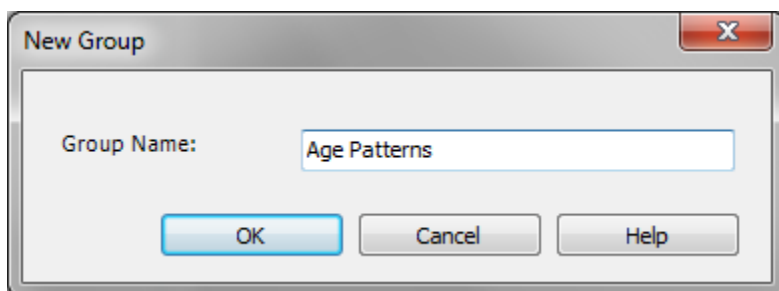


To view any chart in detail in a separate window, double-click on it and resize its window as necessary. In this window you can view the chart details, copy or print it, etc. For more information, see the topic [Single Chart View](#).

## Chart Groups

You can create new chart groups and distribute the charts in these groups.

To create a new chart group, select the **New Group** command from the **Groups** menu, or right-click on the group tab name and select **New** from the context menu. Then specify the desired name in the dialog:



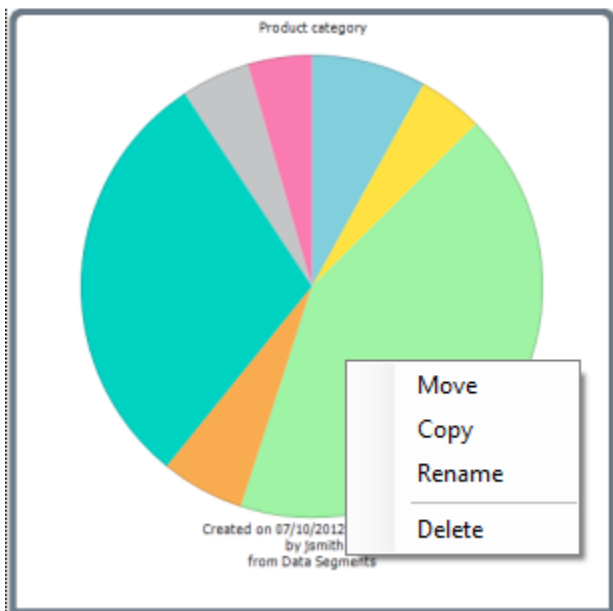
A new blank chart group with the new name will be created.

To rename an existing chart group, right-click on the group tab name and select the **Rename** command from the context menu. Enter the new name and click OK.

## Operations with Charts

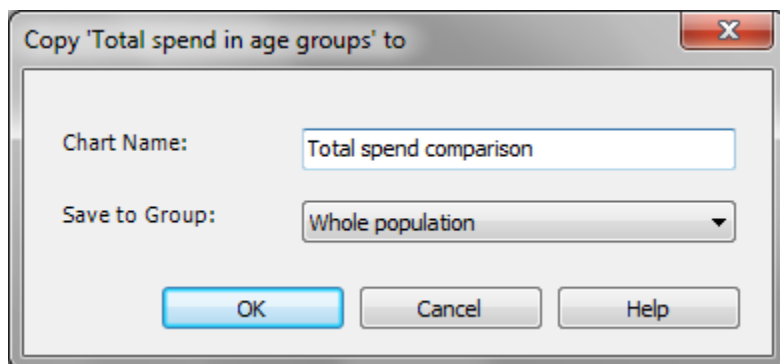
To change the position of any chart within its group, drag and drop it to the desired position.

To delete, copy, rename, or move a chart to a different group, right-click on the chart and select the appropriate command from the context menu.



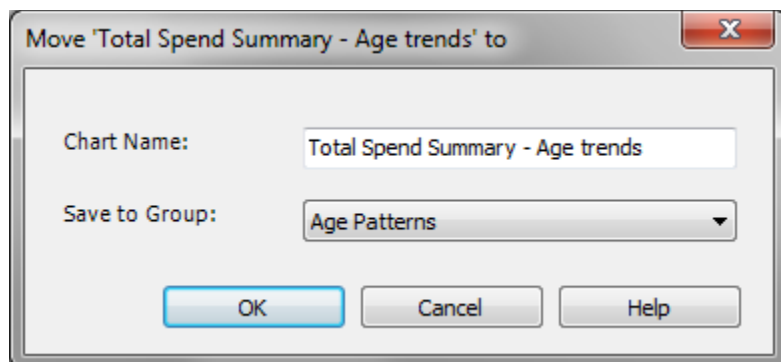
Page | 271

- To delete a chart, select **Delete** from the context menu.
- To rename a chart, select **Rename** from the context menu, specify the new name for the chart, and click OK.
- Charts can be copied within the same chart group or to a different chart group. To copy a chart, right-click on it and select **Copy** from the context menu. Specify the name for the new chart and select the destination group. The current group is the default destination.



- There are two ways to move a chart to another group:
  - a) drag and drop the chart to another group tab, or
  - b) right-click on the chart and select **Move** from the context menu.

In the Move dialog, specify the new name for the chart and select the desired destination group from the Save to Group list



## See also:

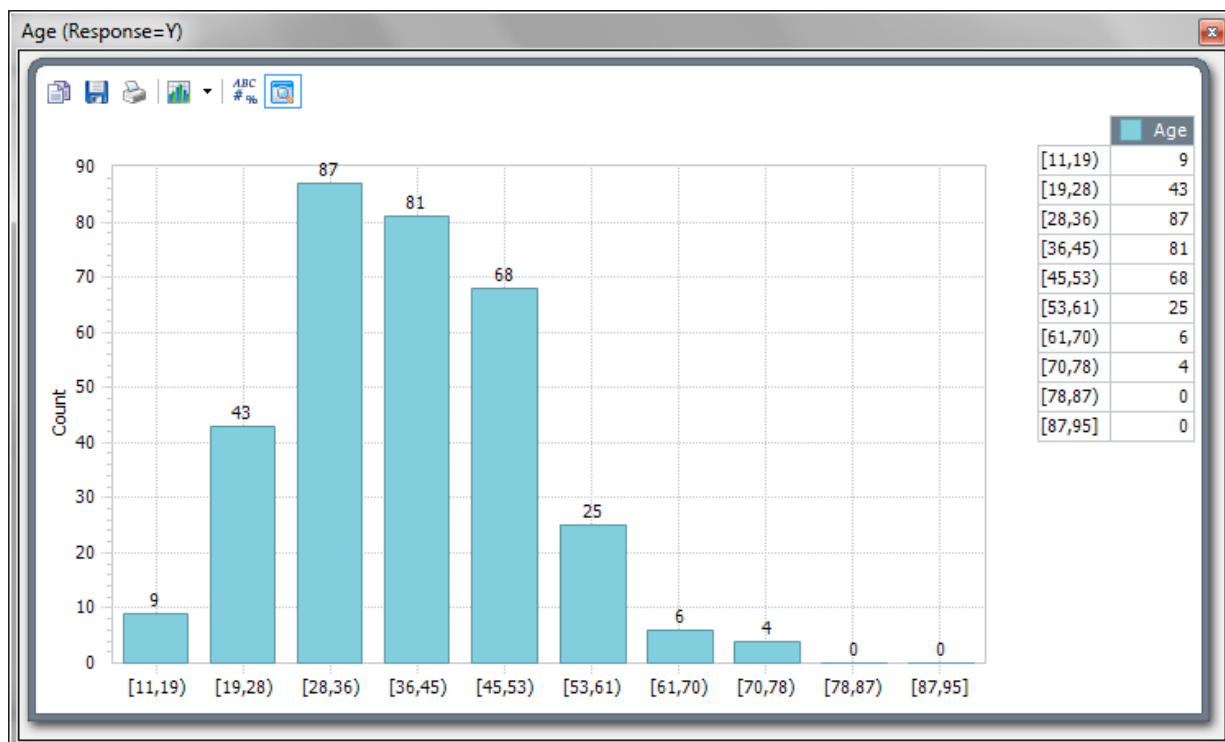
[Single Chart View](#)

## Single Chart View

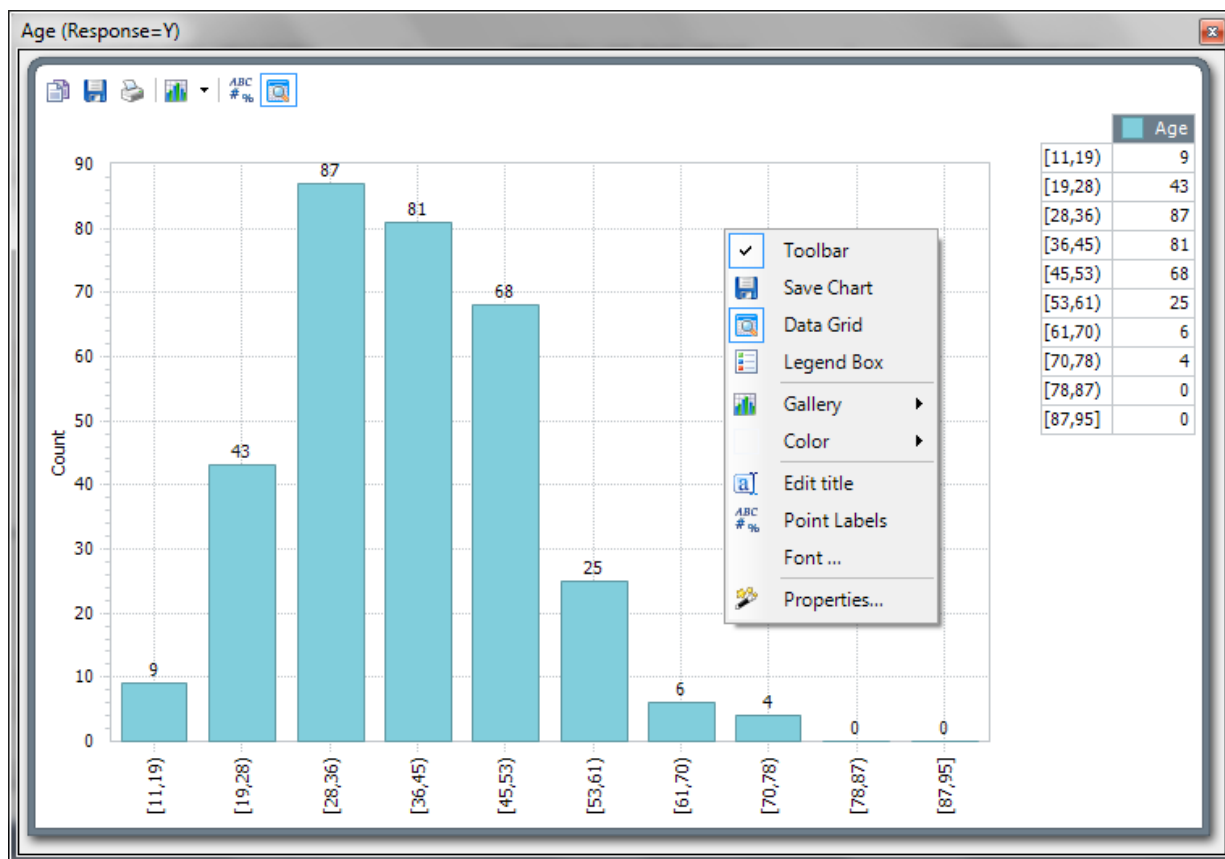
Several views in the application contain multiple charts: [Segment Viewer](#) in datasets, [Chart](#) in trees, and [Saved Charts](#) in both datasets and trees. In any of these views, you can open each individual chart in its own separate window, where you can view its details, copy or print it, etc.

Double-click on the desired chart to view it in a separate window. The window provides a detailed view of the chart and shows the data grid representing the chart data. You can resize the window by clicking and dragging on its corners or borders.

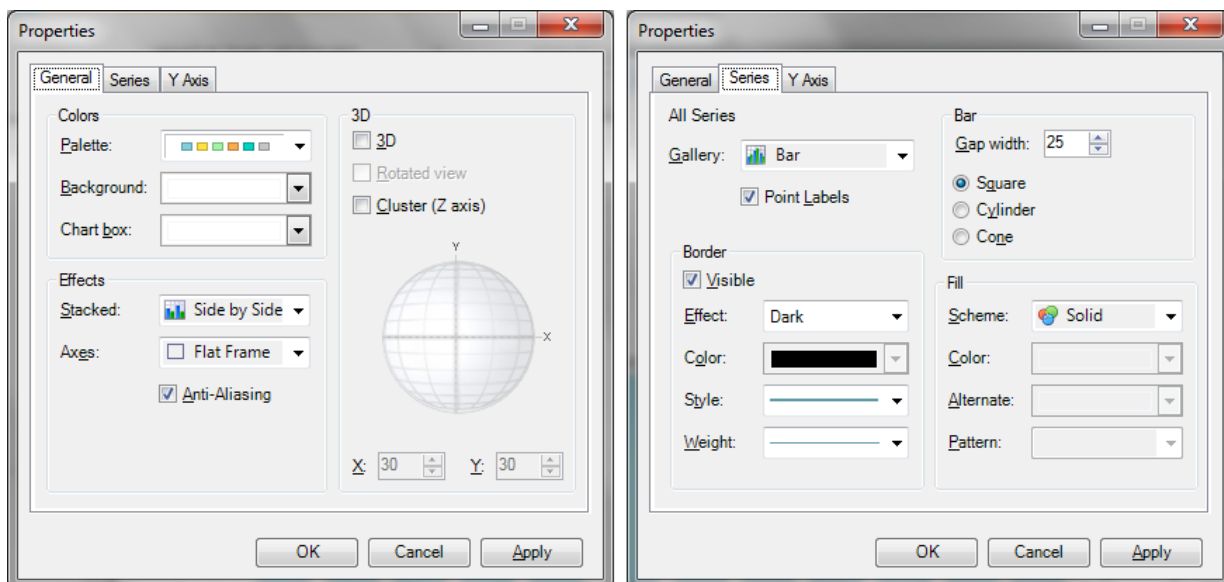
Here is an example of a chart from Segment Viewer opened in its own window.



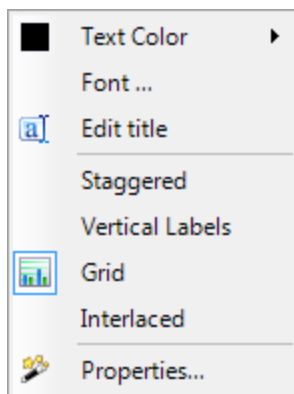
To change the display properties of the chart, use the chart toolbar or right-click anywhere on the chart background. The following context menu appears when right-clicking on the chart background. For example, select Point Labels to show or hide the values on top of the bars; select Data Grid to show or hide the table with the chart data.



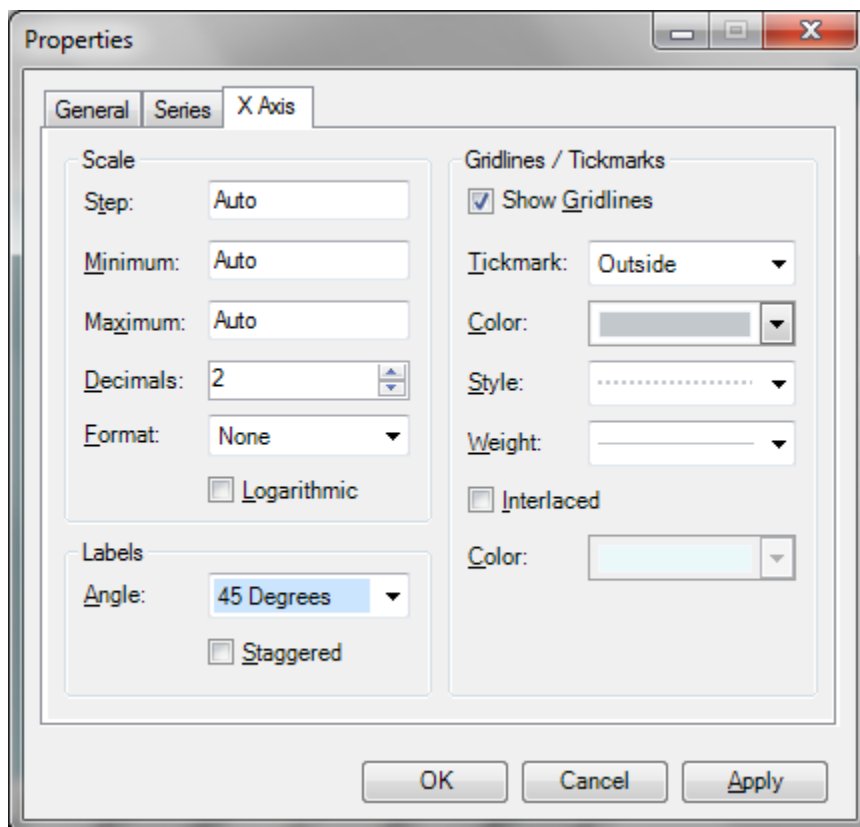
To change advanced properties of the chart, select **Properties** from the context menu. In the **Properties** dialog, you can choose the color palette, 3D or 2D view, effects, properties of the axis labels, show or hide grid lines, etc.



To change the orientation of the axis labels, right-click on any label of the desired axis (vertical or horizontal). The context menu will look like this:

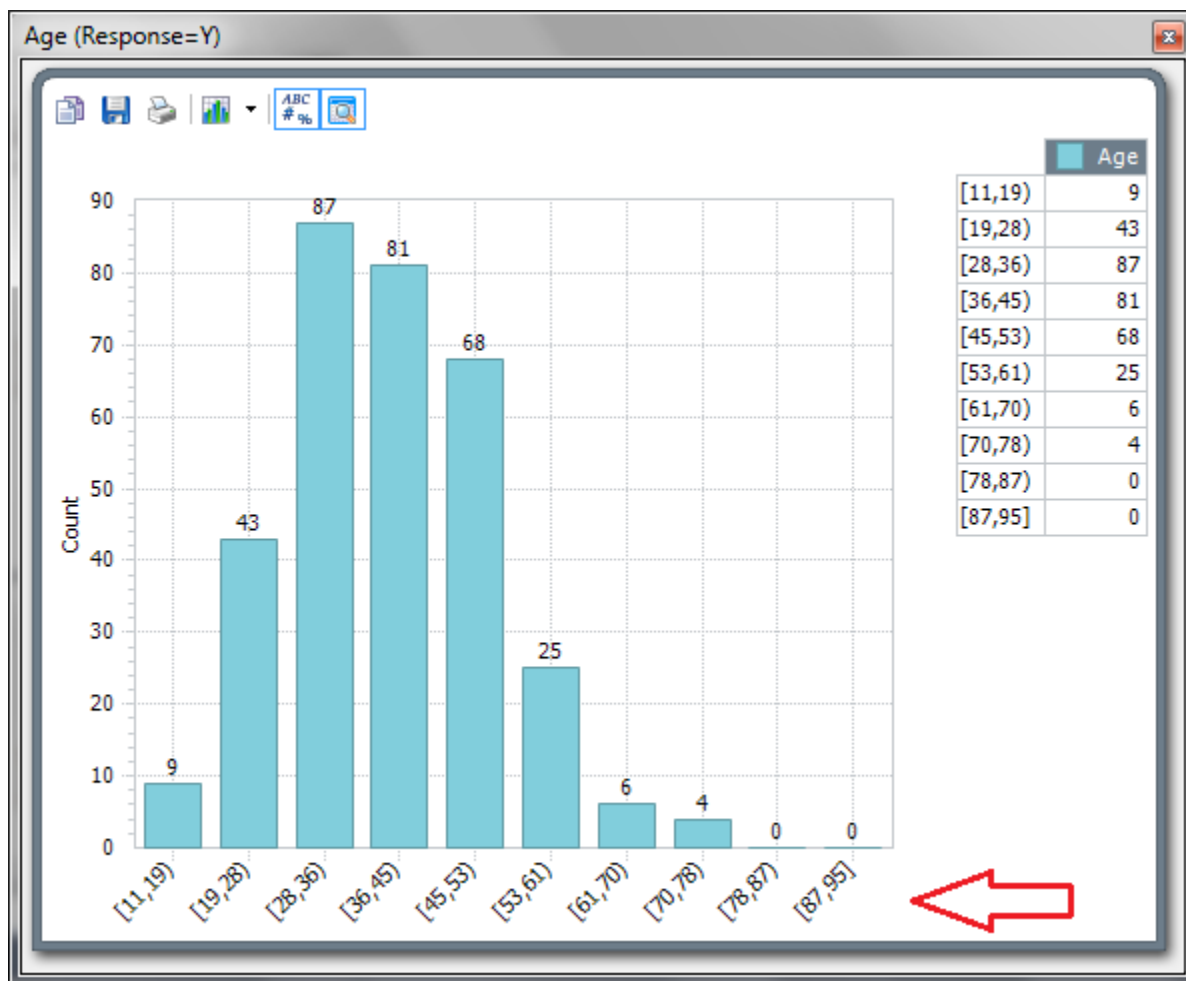


Select *Staggered* or *Vertical Labels* if desired. If you want to place the labels at a 45-degree angle, select *Properties* and set *Angle* to 45 Degrees.



As a result, the X-axis labels are displayed as follows:





## Chart Toolbar

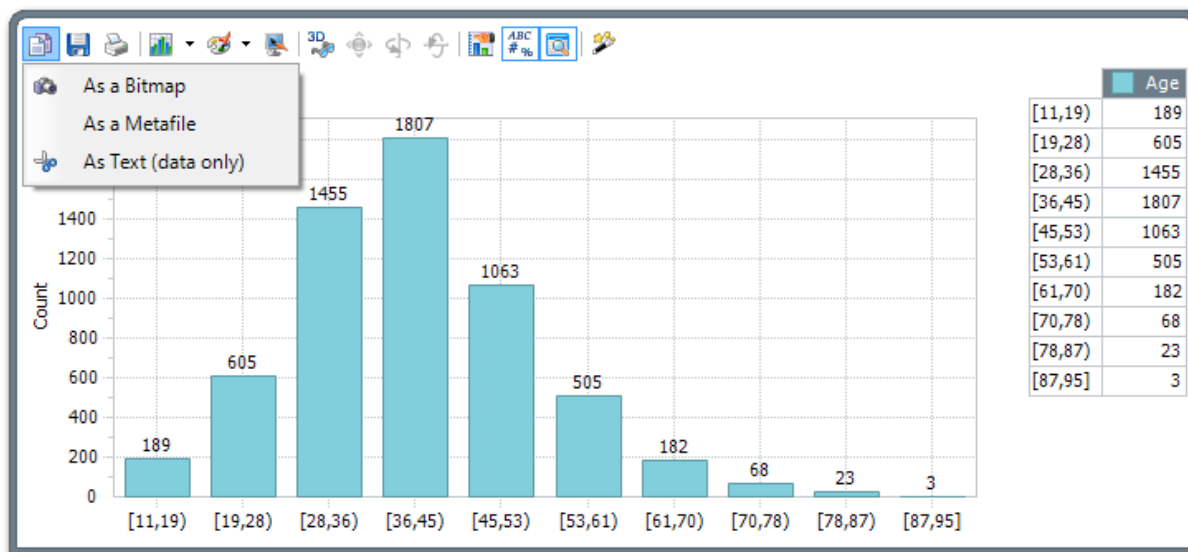
Depending on the charts' parent view - Segment Viewer, Cross Tabs, Chart (in trees), etc., - its toolbar at the top left corner of the chart window can have one of the following two forms:



All icons in the Chart toolbar except the *Save*, *Copy*, and *Print* icons are shortcuts to the corresponding options and commands in the Properties dialog described above.

To copy a chart, click on the *Copy* icon in its toolbar, as shown in the screenshot below. Select *As a Bitmap* or *As a Metafile* from the context menu if you want to copy the chart as an image. Select

As *Text (data only)* if you want to copy the underlying data, which is displayed in the data grid on the right-hand side. Then paste the copied image or text from the clipboard into the desired application.



If you are viewing a chart from the Segment Viewer or Node Chart, you can save it in the Chart Library by clicking on the Save button .

**See also:** [Saved Charts](#)

## Variable Selection

The **Variable Selection** node in the **Profile** palette provides a way to select variables based on measures of predictive power with respect to a dependent variable. After computing the necessary measures and ranking variables, the node produces the output dataframe that contains only the variables selected by the user based on the statistics computed.

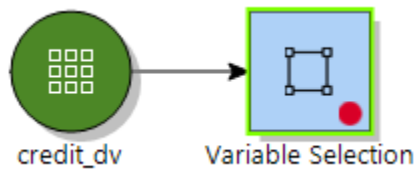
To add a variable selection operation to your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. This brings up a context menu that allows you to select the desired operation. From the context menu, select **Profile --> Variable Selection** as shown in the example below. A new **Variable Selection** process node will be created.

OR:

- Drag the **Variable Selection** node from the **Profile** palette onto the canvas and connect the input dataframe to the node.

If the wizard does not open automatically, double-click the Variable Selection node or right-click on it and select **Modify**.



In the Variable Selection wizard:

- Specify the name for the output dataframe with the variables to be selected.
- Select the dependent variable.
- Choose the measures you wish to use to rank and select the variables. The set of available measures depends on the type of the selected dependent variable (discrete or continuous).

Measures available in the case of a discrete dependent variable:

- *Entropy Explained*
- *Information Value* (for binary DV only)
- *Pearson Chi-Square Statistic*
- *Gini*

Measures available in the case of a continuous dependent variable:

1.
  - *Variance Explained*
  - *Pearson Correlation*
  - *Spearman Correlation*

Variable Selection

Source dataframe: `credit_dv`

Target dataframe: `credit_prep`

Dependent variable: `Status2`

Measures of predictive power

☒ Entropy Explained    ☒ Pearson Chi-Square Statistic    ☒ Pearson Correlation

☒ Information Value    ☒ Gini    ☒ Spearman Correlation

☒ Number of variables: `1`    Criteria: `Union`       

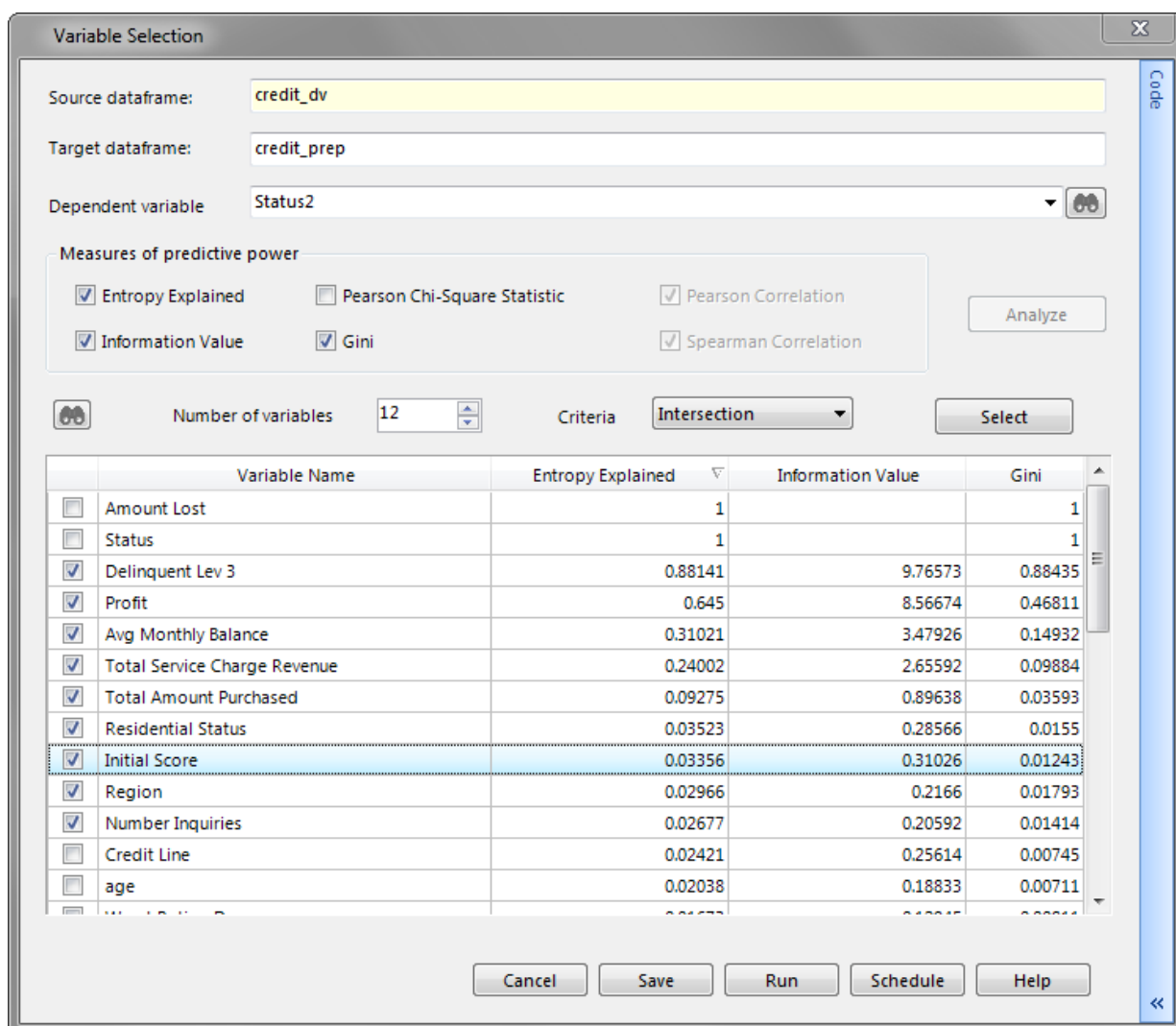
|                          | Variable Name       | Entropy Explained | Information Value | Pearson Chi-Square | Gini |
|--------------------------|---------------------|-------------------|-------------------|--------------------|------|
| <input type="checkbox"/> | age                 |                   |                   |                    |      |
| <input type="checkbox"/> | AgentID             |                   |                   |                    |      |
| <input type="checkbox"/> | Amount Lost         |                   |                   |                    |      |
| <input type="checkbox"/> | Avg Monthly Balance |                   |                   |                    |      |
| <input type="checkbox"/> | Bank Dda Ref        |                   |                   |                    |      |
| <input type="checkbox"/> | Bank Reference      |                   |                   |                    |      |
| <input type="checkbox"/> | Bank Save Ref       |                   |                   |                    |      |
| <input type="checkbox"/> | Credit Card Ref     |                   |                   |                    |      |
| <input type="checkbox"/> | Credit Line         |                   |                   |                    |      |
| <input type="checkbox"/> | Delinquent Lev 1    |                   |                   |                    |      |
| <input type="checkbox"/> | Delinquent Lev 2    |                   |                   |                    |      |
| <input type="checkbox"/> | Delinquent Lev 3    |                   |                   |                    |      |
| <input type="checkbox"/> | Dept Store Ref      |                   |                   |                    |      |

- Click **Analyze** to compute the selected measures. The columns are sortable. Click on the column header for the desired measure to sort on it.
- The variables can be selected in several ways:
  - Automatically: Specify the desired number *N* of variables with the highest values of the selected statistics. Choose the criterion (*Union* or *Intersection*) and click **Select**. If *Union* is chosen, then the wizard selects the top *N* variables with the highest value of **at least one** of the selected statistics. If *Intersection* is chosen, then the wizard selects the top *N* variables with the highest values of **all** of the selected statistics (if this condition can be satisfied).
  - Manually - simply selecting the checkbox on the left of the desired variable. You can highlight multiple rows by clicking on a row and dragging the mouse down, or by clicking the rows while holding down the SHIFT or CTRL key. Then click in the checkbox for any of the highlighted rows to select all of them.

c) Using the [Field Finder](#) button , which is especially useful when there are a lot of variables (hundreds or thousands). It also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

Here is an example where the selected dependent variable is discrete, and the top 12 variables are requested whose Entropy Explained, Information Value, and Gini index are all higher than those of other variables (9 are found).

Page | 281



The Variable Selection dialog box shows the following configuration:

- Source dataframe: credit\_dv
- Target dataframe: credit\_prep
- Dependent variable: Status2
- Measures of predictive power:
  - ☒ Entropy Explained
  - ☐ Pearson Chi-Square Statistic
  - ☒ Pearson Correlation
  - ☒ Information Value
  - ☒ Gini
  - ☒ Spearman Correlation
- Number of variables: 12
- Criteria: Intersection

The table below shows the top 12 variables selected based on the criteria:

| Variable Name                | Entropy Explained | Information Value | Gini    |
|------------------------------|-------------------|-------------------|---------|
| Amount Lost                  | 1                 |                   | 1       |
| Status                       | 1                 |                   | 1       |
| Delinquent Lev 3             | 0.88141           | 9.76573           | 0.88435 |
| Profit                       | 0.645             | 8.56674           | 0.46811 |
| Avg Monthly Balance          | 0.31021           | 3.47926           | 0.14932 |
| Total Service Charge Revenue | 0.24002           | 2.65592           | 0.09884 |
| Total Amount Purchased       | 0.09275           | 0.89638           | 0.03593 |
| Residential Status           | 0.03523           | 0.28566           | 0.0155  |
| Initial Score                | 0.03356           | 0.31026           | 0.01243 |
| Region                       | 0.02966           | 0.2166            | 0.01793 |
| Number Inquiries             | 0.02677           | 0.20592           | 0.01414 |
| Credit Line                  | 0.02421           | 0.25614           | 0.00745 |
| age                          | 0.02038           | 0.18833           | 0.00711 |

An example for the case of a continuous dependent variable:

**Variable Selection**

Source dataframe:

Target dataframe:

Dependent variable:

**Measures of predictive power**

☒ Variance Explained
 ☐ Pearson Chi-Square Statistic
 ☒ Pearson Correlation
 ☐ Information Value
 ☐ Gini
 ☐ Spearman Correlation

7
  Union

|                                     | Variable Name                | Variance Explained | Pearson Correlation |
|-------------------------------------|------------------------------|--------------------|---------------------|
| <input type="checkbox"/>            | Delinquent Lev 1             | 0.04335            | 0.20282             |
| <input type="checkbox"/>            | Total Service Charge Revenue | 0.01981            | 0.10395             |
| <input type="checkbox"/>            | Delinquent Lev 2             | 0.01129            | 0.09683             |
| <input checked="" type="checkbox"/> | Total Amount Purchased       | 0.01403            | 0.0933              |
| <input type="checkbox"/>            | Credit Line                  | 0.00178            | 0.03641             |
| <input type="checkbox"/>            | Bank Dda Ref                 | 0.00042            | 0.02966             |
| <input type="checkbox"/>            | Credit Card Ref              | 0.0003             | 0.02761             |
| <input type="checkbox"/>            | Major Derog Num              | 0.00035            | 0.01976             |
| <input type="checkbox"/>            | Dept Store Ref               | 0.0005             | 0.01937             |
| <input checked="" type="checkbox"/> | age                          | 0.00151            | 0.01862             |
| <input checked="" type="checkbox"/> | Year at Address              | 0.00152            | 0.01757             |
| <input checked="" type="checkbox"/> | Initial Score                | 0.00107            | 0.01684             |
| <input type="checkbox"/>            | Fin Co Ref                   |                    | 0.01332             |

**Code**

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.

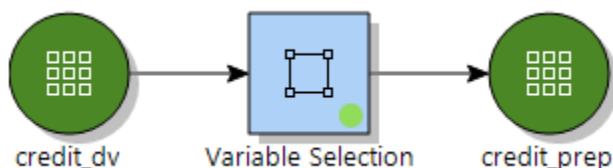
An example Python code:

```
fields = {'Status2', 'age', 'Initial Score', 'Number Inquiries',
          'Region', 'Year at Address'}
credit_prep = angoss.SelectColumns(data = credit_dv, columns = fields)
```

Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

As a result, the node representing the output dataframe appears on the workflow canvas. Double-click on the output dataframe to view the result.

Page | 283



The output dataframe contains only the variables selected in the Variable Selection node wizard.

### **A note regarding the calculation of statistics in the Variable Selection wizard:**

For the correlation statistics, there will be differences when missing values (NULLs) are present in the sets of variables. This amounts to what is meant by “listwise” and “pairwise” correlation. A correlation between X and Y can only be on a set where both values of X and Y are not NULL.

In the Variable Selection wizard, the correlations are “pairwise”, which means that if a NULL is present in either DV or IV, the pair is ignored.

In the Correlations tab of a dataset or dataframe, it is “listwise”, which means that if a NULL is present in any of the selected columns, the entire row is ignored. Note that in this case, otherwise valid pairs may be thrown away, which will result in different correlations being shown.

For tables that contains no missing data, there will be no differences between the Variable Selection statistics and Correlations tab statistics.

# Modeling

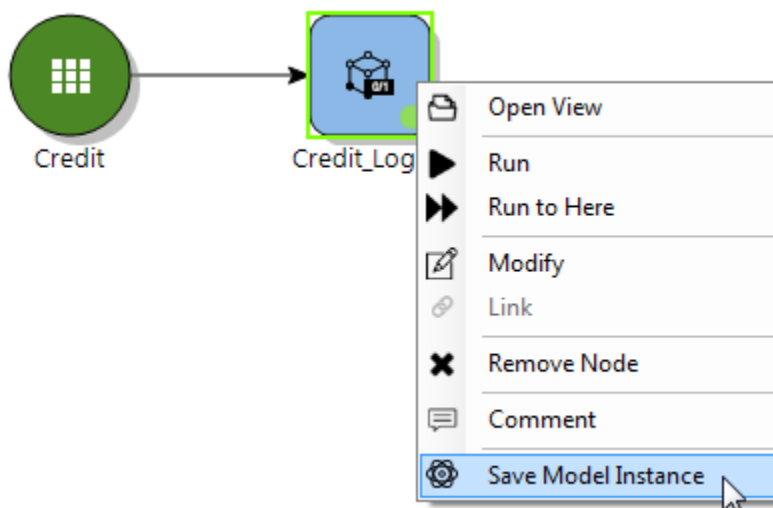
## Model Instances

In any workflow, we distinguish **model process nodes** and **model instance nodes**.

There is an option for all model process nodes including trees to save a model instance that cannot be overwritten by re-running the model node.

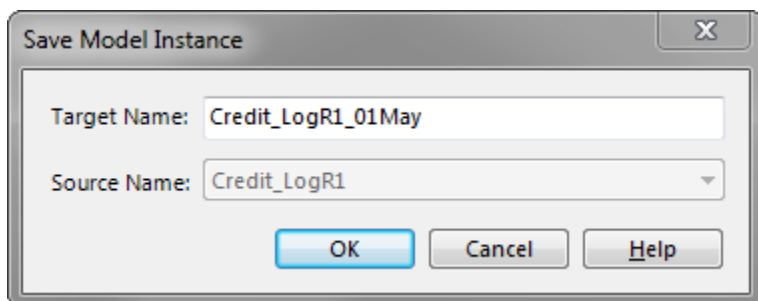
When you add a model to your workflow by dragging it from the Modelling group onto the workflow canvas, you create a **model process node**. It defines an *operation* of creating a model - a description of inputs, outputs, and training parameters - that can be re-run as part of running the workflow. The model object resulting from running a model process node and containing model results (the equation and model training statistics) is linked to the model process node and can be opened by right-clicking the model process node and selecting the **Open View** command. This model object is called a **model instance**.

Every time you re-run the model process node, the corresponding model instance behind it is overwritten. However, you can save the model instance for future use for scoring or other purposes. To save a model instance, right-click on the model process node and select the **Save Model Instance** command.

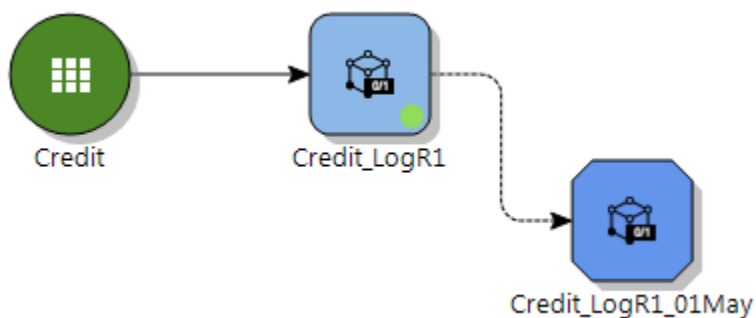


Specify the name for the new model instance. The name of the original model is shown in the Source Name field.

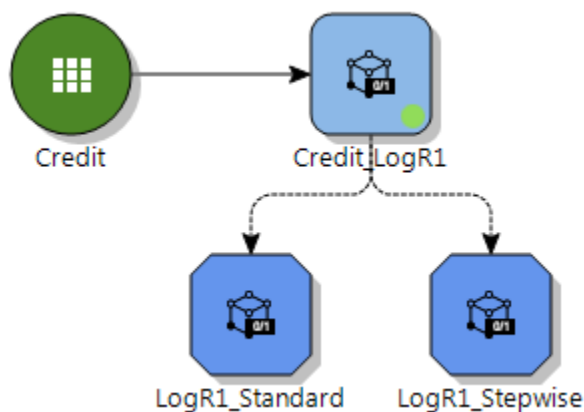




This will save a copy of the model instance resulting from the most recent run of the model process node. The new model instance is shown next to the model process node. It is connected to the process node by a dotted line indicating that this model instance cannot be overwritten by re-running the workflow.



You can save any number of model instance copies for the same model process node.



The connection between the source model node and the instance node can be deleted at any time, if desired. You will still be able to open and score the model instance after deleting the connection.

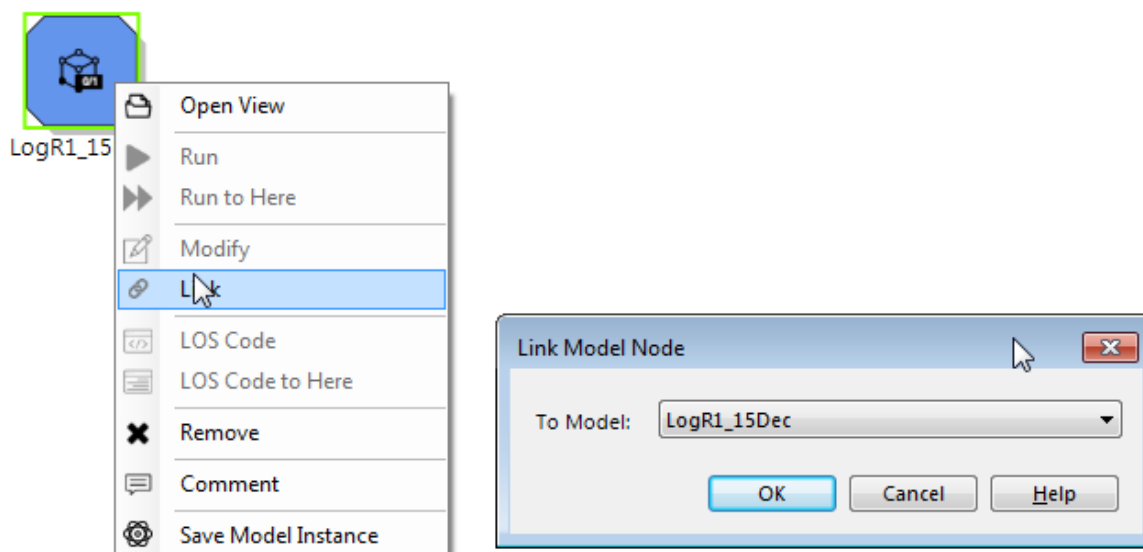
You can also save a model instance of another model instance. This is just creating a copy of a model instance. The copy is not connected to the original instance in the workflow.

## Operations with Model Instances

Page | 286

By definition, model instance nodes cannot be run and cannot be overwritten by re-running the workflow. The main operations defined for model instance nodes available in the right-click context menu are:

- **Open View:** Opens the model results view. For Decision and Strategy Trees, it opens the interactive tree view.
- **Save Model Instance:** Creates another model instance copy
- **Link:** Links the model instance node to a selected physical model in the Project Explorer. Any model instance node can be linked to any physical model - an output of a previous workflow execution available in the Project Explorer.



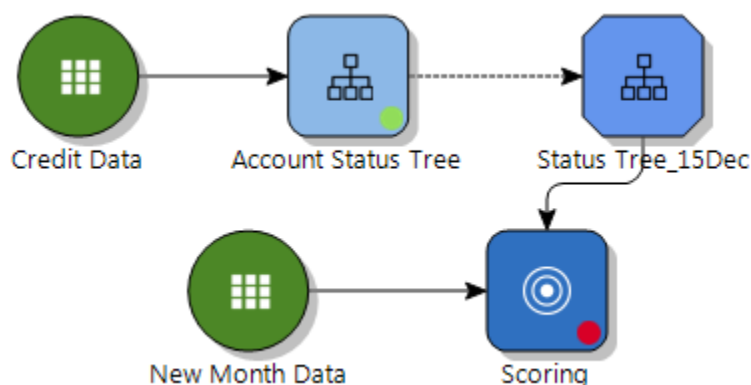
## Scoring and Validation

Model instances of all model types can be validated and scored. (Note: Validation does not apply to Market Basket Analysis). This means you can connect a Validation or Scoring node to any model instance node. In particular, for Decision Tree, Strategy Tree, and Market Basket Analysis models, validation and scoring are applicable ONLY to model instance nodes, and not applicable to model process nodes. So for these three model types you have to save model instances before scoring or validation. The reason is that manual user input is always required for scoring these model types.

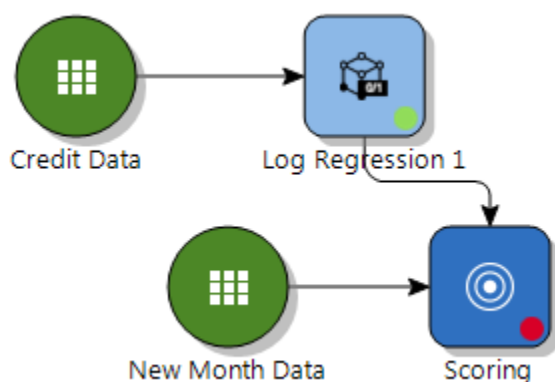
For example, trees are interactive and they need to be manually grown or modified before scoring, and in Market Basket Analysis you need to manually select the association rules to be deployed.

In the example below, the Account Status Tree node cannot be directly connected to the Scoring node. It can be scored only through creating a model instance node.

Page | 287



For Logistic Regression, however, the model process node can be directly connected to the scoring node:



See also: [Workflows](#)

## Decision Trees

### Decision Trees and Splits

A **decision tree** is a statistical model used for classifying data. The information is represented in a tree-like structure where each node is either:

- a *terminal node*, also called a *leaf*, which is a classification of an instance, or
- a decision node, where some test is applied on a single attribute value, with one branch and sub-tree for each possible outcome of the test.

There are two types of decision trees, depending on the type of the dependent variable (*discrete* or *continuous DV*).

Page | 288

The nodes of a decision tree with *discrete (categorical)* DV show the distribution of the DV values (categories) in the form of frequencies and percentages. For example, for DV = AccountStatus (with values 'OK' and 'Bad Debt'), the tree nodes have the following form:

|          |       |         |
|----------|-------|---------|
| Bad Debt | 513   | 17.16%  |
| OK       | 2,477 | 82.84%  |
| Total    | 2,990 | 100.00% |

The nodes of a decision tree with *continuous* DV show the values of Average and Standard Deviation (StdDev) of that DV. For example, for DV = Profit (with values ranging from - 8,600.00 to +1,700.00), the tree nodes have the following form:

|         |                 |
|---------|-----------------|
| Average | 8.76            |
| StdDev  | 224.10          |
| Total   | 29,167 100.00 % |

It is convenient to think of a decision tree as a set of rules, every [split](#) representing a condition. Using the menu command **Insert | Code Generation**, you can generate rules for any tree in several different languages (code types) such as SQL, SAS, SPSS, etc.

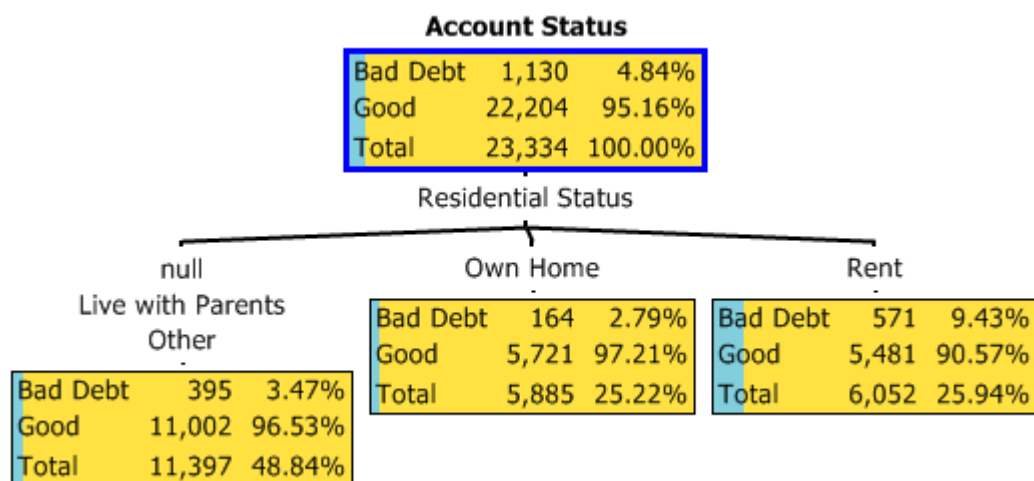
A detailed description of splitting algorithms used in Knowledge Studio decision trees can be found in the topic [Supported Measures](#) and in Mathematical Formulations (see [Mathematical Formulations: Decision Tree](#))

See also:

[What is a Split](#)

[Inserting a Decision Tree](#)

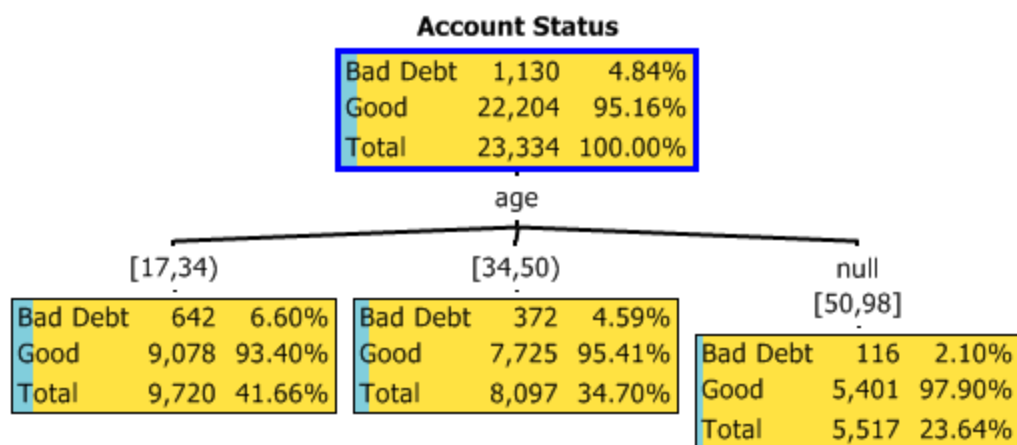
A split represents a rule. The rule determines how records from the parent node are to be distributed across the child nodes. The child nodes are the values (either ranges or categories) of a fixed independent variable (IV) which we refer to as the split variable. The rule is based on the split variable. Within the child node, the distribution of values of the variable of interest, which we refer to as the dependent variable (DV), are shown.



This example shows a parent node (which happens to be a root node) and a split. The data we are using is from the credit industry and our variable of interest, or the DV, is **AccountStatus**.

The split variable is **Residential Status**, and each child node represents a subset of the root node where the value of **Residential Status** is equal to the value (or one of the values) displayed above the node. For example, the rightmost node contains all records where *Residential Status* = 'Rent'. The condition *Residential Status* = 'Rent' is the *rule* of the rightmost node.

The split shown below is on a *continuous* independent variable, and its rules are therefore represented by ranges, or intervals, of the split variable **age**.



In our example, there are 12119 records that satisfy the rule **age** = [17, 34), i.e.,  $17 \leq \text{age} < 34$ .

See the Help section [Creating and Modifying Splits](#) for more details.

See also:

---

[Inserting a Decision Tree](#)[Find Split](#)[Split Report](#)[Automatic Grow](#)[Measure](#)[Tree View](#)[Creating Calculated Column from Split](#)

Page | 290

## Inserting a Decision Tree

A decision tree node can be added to the workflow in one of the following ways:

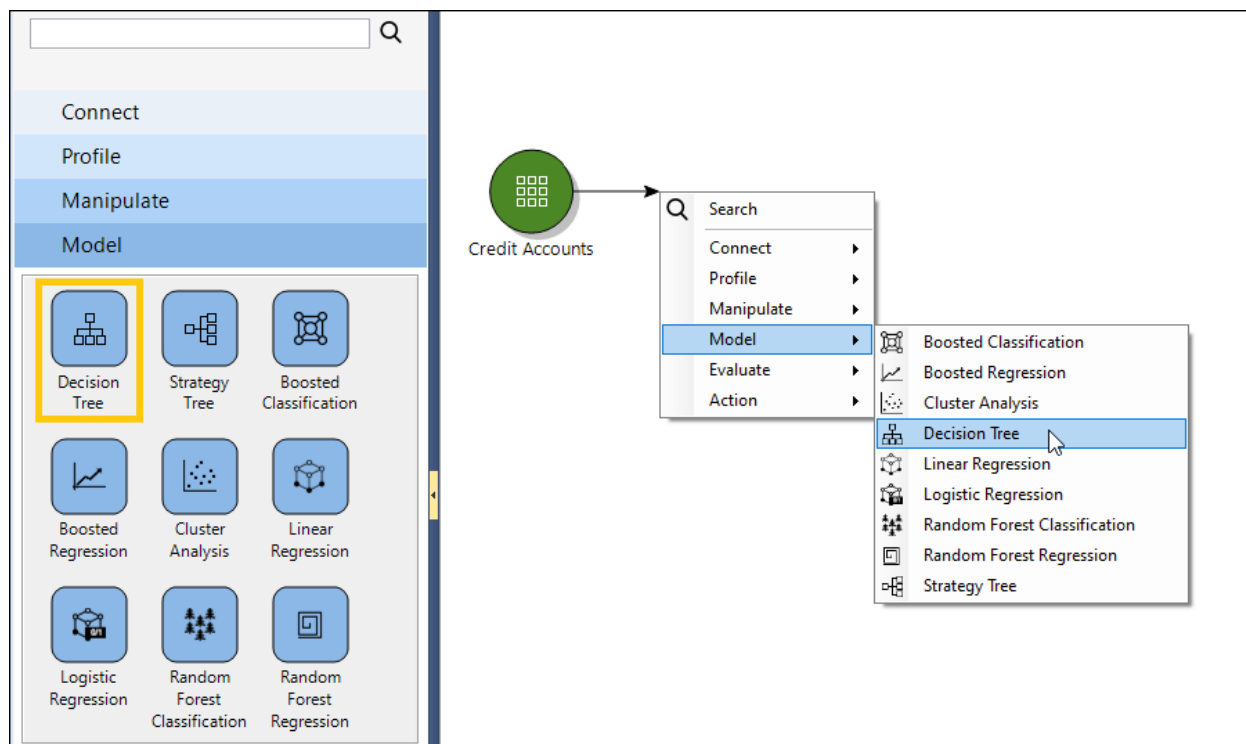
- Drag a connection arrow from a dataset node to any area on the Workflow canvas background. From the context menu that opens upon releasing the mouse button, select **Model --> Decision Tree** as in the example screenshot below. A new **Decision Tree** process node will be created and the wizard will open.

OR

- Right-click on the Workflow canvas and select **Model --> Decision Tree** from the context menu. The new Decision Tree node is gray until its inputs are defined. Connect a dataset to the tree node.

OR

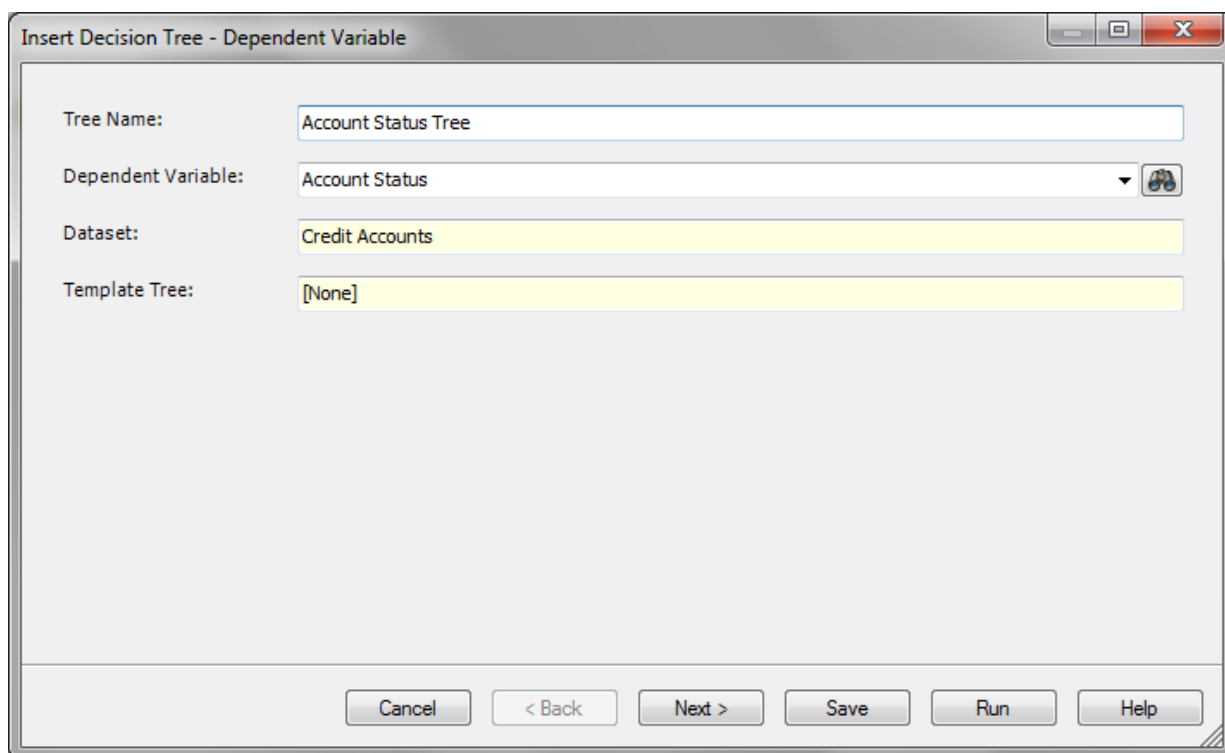
- Open the **Model** palette and drag the **Decision Tree** node onto the workflow canvas. The **Decision Tree** node is gray while its inputs are undefined. Connect the input dataset to the **Decision Tree** node by drawing a connection arrow from the dataset node to the tree node.



Double-click on the Decision Tree node added to the canvas, or right-click on it and select **Modify** from the context menu.

Proceed to [Decision Tree: Select Dependent Variable](#)

Select the dependent variable and specify the tree name.



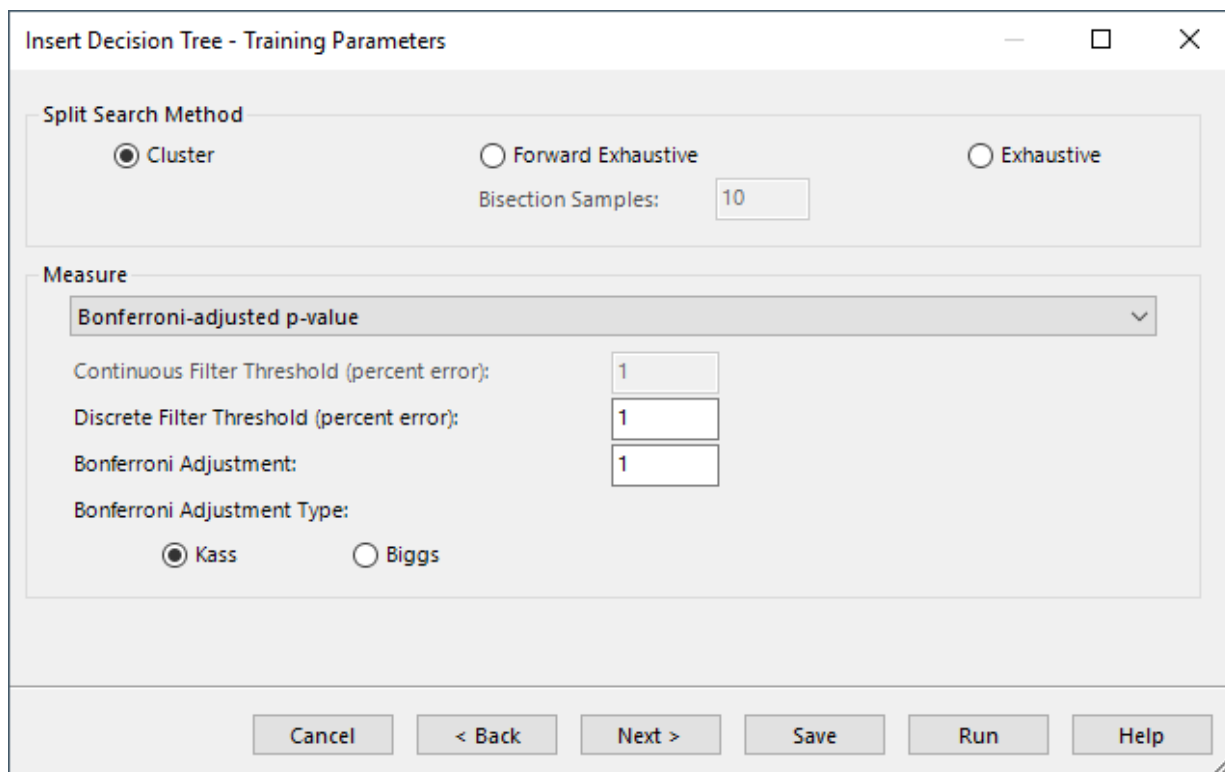
|                           |                                                                                                                                                           |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Tree Name</b>          | The name of the tree as it will appear in the project.                                                                                                    |
| <b>Dependent Variable</b> | The dependent (target) variable for the analysis. The drop down list contains all available variables in the dataset.                                     |
| <b>Dataset</b>            | The name of the parent dataset of the tree.                                                                                                               |
| <b>Template Tree</b>      | The tree to copy parameters from. If you want to insert a copy of a tree that already exists in your project, connect that tree to the current tree node. |

Click Next to specify the training parameters. If you would like to save your settings or insert the root node of the tree accepting all other defaults, click **Save** or **Run**.

Proceed to the [Training Parameters](#) topic.

The training parameters set in this wizard page govern the splitting mechanism of the **Find Split** and **Automatic Grow** operations.





## Split Search Method

The *Split Search Method* determines the iterative process of merging groups of values into bins until an optimal binning is achieved according to certain criteria of the split significance. This procedure locates the pairs of most similar groups and merges them together. Upon re-evaluation of the binning, this process is repeated. Merges are subject to the constraints defined by the type of independent variable (nominal, ordinal, or continuous) and whether or not the groups are in fact "similar" by comparison of the cost function to a merging threshold value. Special data structures are used to make the search for similarity between two groups as fast as possible.

### **Cluster**

The *Cluster* method finds groups that maximize similarity within the groups and dissimilarity between the groups. Merging stops when either there is only one group left or there is no pair of groups that is "similar" enough with respect to the "merging threshold". The Cluster method tends to find more natural patterns than Exhaustive. This method is the application default. See the [Find Split](#) topic for more technical details.

### **Exhaustive**

The *Exhaustive* method performs an exhaustive search to find groups that maximize statistical significance. Merging stops when there is only one group left. While merging, the sequence of merges is recorded in a special history buffer, noting the "best", global, solution found to date. When the merging process is finished, a re-initialization to the start point is performed, and the sequence of merges recorded in the aforementioned "history buffer" is performed. This process stops as soon as the "best"

### **Forward Exhaustive**

history entry (i.e., the noted best solution) is encountered. Splits derived from the Exhaustive method display the most significant relationships. See the [Find Split](#) topic for more technical details.

This split search method is optimized for ordered variables. It begins with no split points, and adds them iteratively. It combines binary split search applied to the bins of the current (k-th) iteration and ternary splits on the bins obtained at the previous iteration (k-1). In addition, at each iteration it uses a discrete mesh of potential split points to increase the chances of achieving the global minimum p-value, rather than stopping at a local minimum. The **Bisection Samples** parameter determines the number of nodes in this mesh. The default is 10. The admissible values are nonnegative integers. Zero means no sampling (i.e., a complete sweep is performed). The higher the value, the more likely the method is to find the optimal global solution – at the expense of performance. These optimizations affect only ordered variables – the variables whose Usage is 'ordinal' or 'continuous'. For splits on nominal variables, the regular Exhaustive method is used.

Page | 294

See the [Find Split](#) topic for more technical details on the splitting process.

## Selecting the Measure

The [measure](#) determines the method of evaluating the significance, or importance, of a split. The best split, or the most significant split, is the first split produced by a **Find Split** command. Choose the measure for your decision tree model. (The default is Entropy Variance, unless changed in [Preferences](#)). The available measures are:

### Entropy Variance

Entropy is the measure used to determine the importance of each split in a tree. Four statistics are calculated for this measure, **input entropy**, **output entropy**, **ratio entropy** and **info**. See Mathematical Formulations for detailed formulas for these values.

For decision trees, entropy measures the variability of the distribution of the dependent variable (DV) values with respect to the independent variables (IV). The value of entropy is a number in the interval [0,1]. An entropy of one indicates high variability in the distribution and an entropy approaching zero indicates low variability. As the variability is reduced, there is a gain in information.

In general, the output entropy value will decrease (approach zero) as the importance of the split increases. The value of info shows the importance of a split expressed in a percentage. The higher the percentage, the more important the split.

### Entropy Variance

This measure uses statistics based on entropy or variance to determine the importance of each split in a tree. The statistics calculated for this measure are:

- for discrete dependent variables:
-

- **information value** (applicable to binary DV only)
  - **input entropy**
  - **output entropy**
  - **ratio entropy**
  - **info** (ratio entropy expressed as a percentage)
- for continuous dependent variables:
    - 
    - **input variance**
    - **output variance**
    - **ratio variance**
    - **info** (ratio variance expressed as a percentage)

For decision trees, entropy measures the variability of the distribution of the dependent variable (DV) values with respect to the independent variables (IV). The value of entropy is a number in the interval [0,1]. An entropy of one indicates high variability in the distribution and an entropy approaching zero indicates low variability. As the variability is reduced, there is a gain in information.

In general, the output entropy value will decrease (approach zero) as the importance of the split increases. The value of info shows the importance of a split expressed as a percentage. The higher the percentage, the more important the split.

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

#### Gini Variance

This measure uses the Gini coefficient, or Gini index, to determine the importance of each split in a tree. The statistics calculated for this measure are:

- for discrete dependent variables:
  - 
  - **information value** (applicable to binary DV only)
  - **input index**
  - **output index**
  - **ratio index**
- for continuous dependent variables:
  - 
  - **input variance**
  - **output variance**
  - **ratio variance**

**Info** is **ratio index** or **ratio variance** expressed as a percentage.

For decision trees, the Gini index compares the equality of the distributions of the dependent variable (DV) values with respect to the independent variables (IV). The value of the Gini index is a number in the interval [0,1]. A Gini index approaching 0 indicates that the distributions are equal, a Gini index approaching 1 indicates that the distributions are not equal.

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

Page | 296

#### Unadjusted p-value

In trees based on the *Unadjusted p-value* measure, the significance of splits is ranked using the p-value. The statistics calculated for this measure are:

- for discrete dependent variable:
  - - **information value** (applicable to binary DV only)
    - **p-value**
    - **chi-square**
    - **df** (degrees of freedom)
    - **confidence** (1 - p)
- for continuous dependent variable:
  - - **p-value**
    - **confidence** (1 - p)
    - **F-statistic**
    - **df numerator**
    - **df denominator**

**Info** is confidence expressed as a percentage:  $(1 - p) * 100\%$

p-value expresses the probability that the difference being observed occurs by chance. The higher the p-value the more likely the observed differences in the DV distribution are random.

chi-square: Chi-Squared Test uses the chi-square statistic measuring the statistical association between two categorical variables.

df – Degrees of freedom: The number of observations minus the number of necessary relations, which is obtainable from these observations ( $df = n - r$ ). The definition of "necessary relationship" is simply defined as the relationship between a dependent variable (Y) and each independent variable (X) in the research.

The F-ratio statistic is calculated as part of the ANOVA procedure. The larger this number is, the greater the distance between the means or average values of the nodes in the split. ANOVA Procedure - (ANalysis Of VAriance). A procedure used to detect statistically significant effects induced by an independent variable on a continuous dependent variable. The ANOVA procedure

employs an F-test to measure the differences between a given set of population means, where  $F = \text{Mean Square for Treatments} / \text{Mean Square for Error}$  ( $F = \text{MST}/\text{MSE}$ ).

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

Unadjusted log p-value

Page | 297

In trees based on the *Unadjusted log p-value* measure, the significance of splits is ranked using the logarithm of p-value. The statistics calculated for this measure are:

- for discrete dependent variable:
  - - **information value** (applicable to binary DV only)
    - **p-value**
    - **log p-value** (the natural logarithm of p-value)
    - **chi-square**
    - **df** (degrees of freedom)
    - **confidence** ( $1 - p$ )
- for continuous dependent variable:
  - - **p-value**
    - **log p-value** (the natural logarithm of p-value)
    - **confidence** ( $1 - p$ )
    - **F-statistic**
    - **df numerator**
    - **df denominator**

The split significance is ranked by  **$-\log(p)$**  (negative logarithm of p-value).

The F-Ratio Statistic is calculated as part of the ANOVA procedure. The larger this number is, the greater the distance between the means or average values of the nodes in the split. ANOVA Procedure - (ANalysis Of VAriance). A procedure used to detect statistically significant effects induced by an independent variable on a continuous dependent variable. The ANOVA procedure employs an F-test to measure the differences between a given set of population means, where  $F = \text{Mean Square for Treatments} / \text{Mean Square for Error}$  ( $F = \text{MST}/\text{MSE}$ ).

The p-value is the probability that the difference being observed occurs by chance. The higher the p-value the more likely the observed differences in the DV distribution are random.

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

Bonferroni-adjusted p-value

In trees based on the *Bonferroni-adjusted p-value* measure, the significance of splits is ranked using the p-value with Bonferroni adjustment. The statistics calculated for this measure are:

- for discrete dependent variable:
  - 
  - **information value** (applicable to binary DV only)
  - **p-value**
  - **adjusted p-value** (p-value with Bonferroni adjustment)
  - **the Bonferroni factor**
  - **chi-square**
  - **df** (degrees of freedom)
  - **confidence** (1 - p)
  - **adjusted confidence** (1 - adjusted p)
- for continuous dependent variable:
  - 
  - **p-value**
  - **adjusted p-value** (p-value with Bonferroni adjustment)
  - **the Bonferroni factor**
  - **confidence** (1 - p)
  - **adjusted confidence** (1 - adjusted p)
  - **F-statistic**
  - **df numerator**
  - **df denominator**

**Info** is adjusted confidence expressed as a percentage:  $(1 - \text{adjusted } p) * 100\%$

The Bonferroni adjustment ( $B * p$ ) uses multiple test comparisons and adjusts the significance level (p-value) accordingly. The p-value used as the cutoff is divided by the total number of comparisons to be made. Any single comparison would then be compared to this adjusted p-value. For example, for the cutoff,  $p < 0.05$  and using 10 comparisons, any single test statistic would be compared to  $p < 0.005$ .

The Bonferroni adjustment factor for the p-value is used to minimize the discovery of chance relationships in a dataset. Bonferroni adjustments automatically adjust the level of significance used when testing the validity of a relationship.

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

Bonferroni-adjusted log p-value

In trees based on the *Bonferroni-adjusted log p-value* measure, the significance of splits is ranked using the logarithm of the adjusted p-value. The statistics calculated for this measure are:

- for discrete dependent variable:
  - - **information value** (applicable to binary DV only)
    - **p-value**
    - **log p-value** (the natural logarithm of p-value)
    - **adjusted p-value** (p-value with Bonferroni adjustment)
    - **log adj p-value** (the natural logarithm of the Bonferroni-adjusted p-value)
    - **the Bonferroni factor**
    - **chi-square**
    - **df** (degrees of freedom)
    - **confidence** (1 - p)
    - **adjusted confidence** (1 - adjusted p)
- for continuous dependent variable:
  - - **p-value**
    - **log p-value** (the natural logarithm of p-value)
    - **adjusted p-value** (p-value with Bonferroni adjustment)
    - **log adj p-value** (the natural logarithm of the Bonferroni-adjusted p-value)
    - **the Bonferroni factor**
    - **confidence** (1 - p)
    - **adjusted confidence** (1 - adjusted p)
    - **F-statistic**
    - **df numerator**
    - **df denominator**

The split significance is ranked by  **$-\log(\text{adj } p)$**  (negative logarithm of the Bonferroni-adjusted p-value).

The Bonferroni adjustment ( $B * p$ ) uses multiple test comparisons and adjusts the significance level (p-value) accordingly. The p-value used as the cutoff is divided by the total number of comparisons to be made. Any single comparison would then be compared to this adjusted p-value. For example, for the cutoff,  $p < 0.05$  and using 10 comparisons, any single test statistic would be compared to  $p < 0.005$ .

The Bonferroni adjustment factor for the p-value is used to minimize the discovery of chance relationships in a dataset. Bonferroni adjustments automatically adjust the level of significance used when testing the validity of a relationship.

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

Page | 300

See [Supported Measures](#) for more details.

## Filter Thresholds

### ***Continuous/Discrete Filter Threshold (percent error)***

Generally speaking, when analyzing data using the unadjusted or adjusted p-value measure, you run the risk that your data may display results obtained through chance, which can be potentially misleading. By adjusting the filter threshold value, you can lower the possibility that chance will play a part in your analysis - you determine how rigorously your data will be analyzed.

Continuous and discrete filter thresholds are for trees with continuous and discrete dependent variable, respectively. These thresholds are related to the selected measure as follows:

- a) They are not applicable for non-P-value measures (Entropy Variance and Gini Variance)
- b) For P-value measures (Unadjusted p-value and Bonferroni-adjusted p-value), the thresholds are based on the confidence level (C) of a split, which is described in the section "Mathematical Formulations" (Decision Trees). The *Info* parameter in Split Report (same as *Rank* in the Go To Split dialog) is the value of C expressed in percentage. For example, if C=0.95, then Info = Rank = 95%.

**Note:** If you want the parameter C to be displayed at every split of the current tree (for trees with P-value measure), select **Tools | Options** from the main menu, go to the Tree Split Parameters tab, and check the checkbox for C.

By specifying the filter threshold, you define the minimum value of Info (or Rank) that is of interest to you, so that Find Split or AutoGrow would not show the splits on independent variables whose rank is lower than 100% minus the specified error.

For example, if you are interested only in the independent variables whose Rank is higher than 98%, specify 2 as the filter threshold (percent error). In this case, variables with Rank less



than 98% ( = 100% - 2%) will not appear in the Go To Split dialog and no statistics will be displayed for them in Split Report.

## Bonferroni Adjustment

Page | 301

### **Bonferroni Adjustment**

*This option is applicable only to the Bonerroni-adjusted measures.*

To minimize the discovery of chance relationships, Bonferroni adjustments let you adjust the level of significance used when testing the validity of a relationship.

This type of adjustment compensates for amount of redundant variables in your dataset. Normally, if a concept is captured by more than one variable, it has a disproportionate opportunity to enter the analysis by chance. Use the Bonferroni adjustment to compensate for this possibility.

### **Bonferroni Adjustment Type (Kass / Biggs)**

Bonferroni adjustments may be quite conservative. Increasing the size of the adjustment increases the level of conservatism. It is not meaningful to set this value greater than the total number of variables in your dataset

Two options are available for the Bonferroni adjustment type: Kass and Biggs (named after the authors of these methods, G.V. Kass and D. Biggs). The Kass type is the default. These options influence the number of child nodes in splits, and their effect depends on the type (nominal, ordinal, or continuous) and the cardinality of the split variable. For example, according to G. Ritschard, the Kass adjustment type tends to penalize high-cardinality predictors more than low-cardinality ones, and nominal predictors more than ordinal or continuous predictors. For details, see the articles by these authors listed in the [References](#) section of Mathematical Formulations.

Thus, for instance, if you increase the Default number of bins (the initial number of bins for the Find Split operation) in Attribute Editor before growing the tree with Find Split or Automatic Grow, the Kass adjustment type may be more preferable as the Find Split algorithm will be less likely to pick variables with lower relevance to the target. The Biggs type may be preferable in other contexts. For guidance, see the articles cited above.

See also:

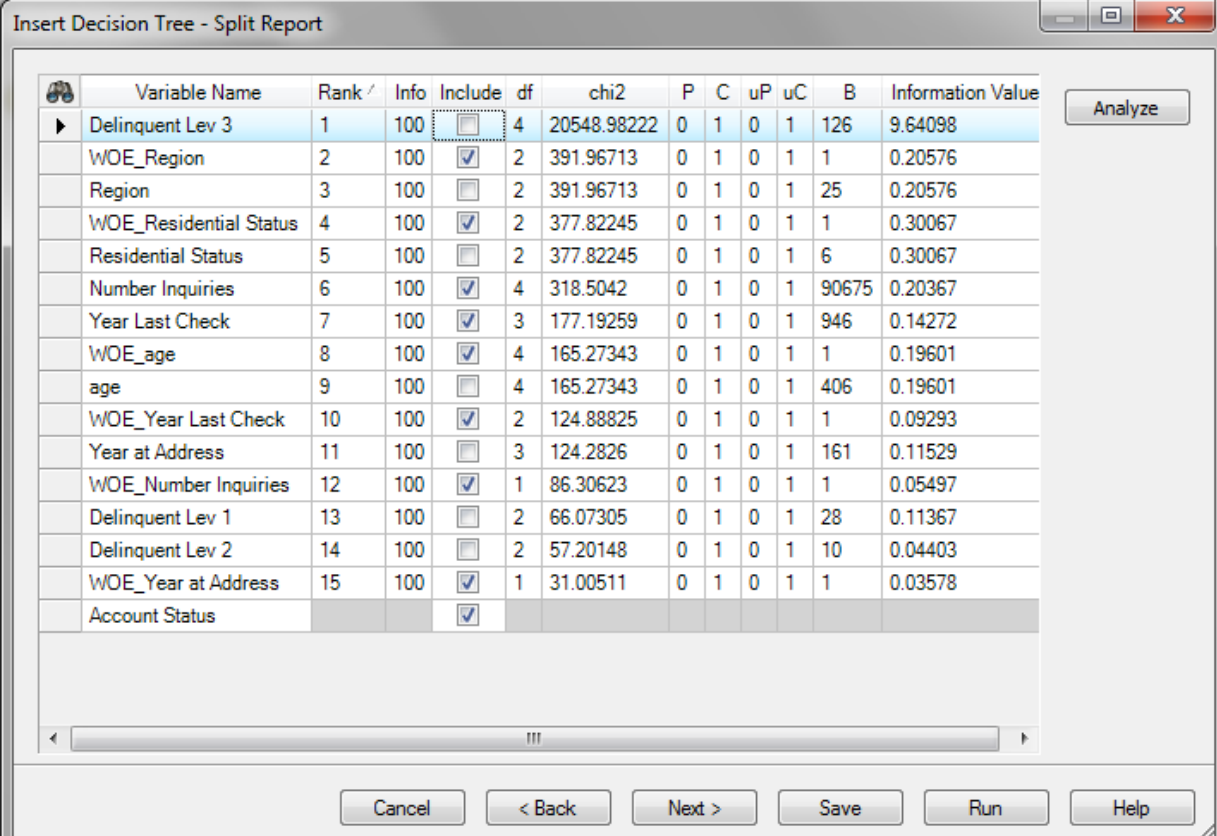
[Find Split](#)

[Automatic Grow](#)

[Split Report](#)

[Mathematical Formulations: Decision Tree](#)

This step allows you to choose the independent variables you want to use in your model. By default, all the variables are included. To exclude a variable uncheck the corresponding box in the Include column.



| Variable Name          | Rank | Info | Include                             | df | chi2        | P | C | uP | uC | B     | Information Value |
|------------------------|------|------|-------------------------------------|----|-------------|---|---|----|----|-------|-------------------|
| Delinquent Lev 3       | 1    | 100  | <input type="checkbox"/>            | 4  | 20548.98222 | 0 | 1 | 0  | 1  | 126   | 9.64098           |
| WOE_Region             | 2    | 100  | <input checked="" type="checkbox"/> | 2  | 391.96713   | 0 | 1 | 0  | 1  | 1     | 0.20576           |
| Region                 | 3    | 100  | <input type="checkbox"/>            | 2  | 391.96713   | 0 | 1 | 0  | 1  | 25    | 0.20576           |
| WOE_Residential Status | 4    | 100  | <input checked="" type="checkbox"/> | 2  | 377.82245   | 0 | 1 | 0  | 1  | 1     | 0.30067           |
| Residential Status     | 5    | 100  | <input type="checkbox"/>            | 2  | 377.82245   | 0 | 1 | 0  | 1  | 6     | 0.30067           |
| Number Inquiries       | 6    | 100  | <input checked="" type="checkbox"/> | 4  | 318.5042    | 0 | 1 | 0  | 1  | 90675 | 0.20367           |
| Year Last Check        | 7    | 100  | <input checked="" type="checkbox"/> | 3  | 177.19259   | 0 | 1 | 0  | 1  | 946   | 0.14272           |
| WOE_age                | 8    | 100  | <input checked="" type="checkbox"/> | 4  | 165.27343   | 0 | 1 | 0  | 1  | 1     | 0.19601           |
| age                    | 9    | 100  | <input type="checkbox"/>            | 4  | 165.27343   | 0 | 1 | 0  | 1  | 406   | 0.19601           |
| WOE_Year Last Check    | 10   | 100  | <input checked="" type="checkbox"/> | 2  | 124.88825   | 0 | 1 | 0  | 1  | 1     | 0.09293           |
| Year at Address        | 11   | 100  | <input type="checkbox"/>            | 3  | 124.2826    | 0 | 1 | 0  | 1  | 161   | 0.11529           |
| WOE_Number Inquiries   | 12   | 100  | <input checked="" type="checkbox"/> | 1  | 86.30623    | 0 | 1 | 0  | 1  | 1     | 0.05497           |
| Delinquent Lev 1       | 13   | 100  | <input type="checkbox"/>            | 2  | 66.07305    | 0 | 1 | 0  | 1  | 28    | 0.11367           |
| Delinquent Lev 2       | 14   | 100  | <input type="checkbox"/>            | 2  | 57.20148    | 0 | 1 | 0  | 1  | 10    | 0.04403           |
| WOE_Year at Address    | 15   | 100  | <input checked="" type="checkbox"/> | 1  | 31.00511    | 0 | 1 | 0  | 1  | 1     | 0.03578           |
| Account Status         |      |      | <input checked="" type="checkbox"/> |    |             |   |   |    |    |       |                   |

## Using the Split Report

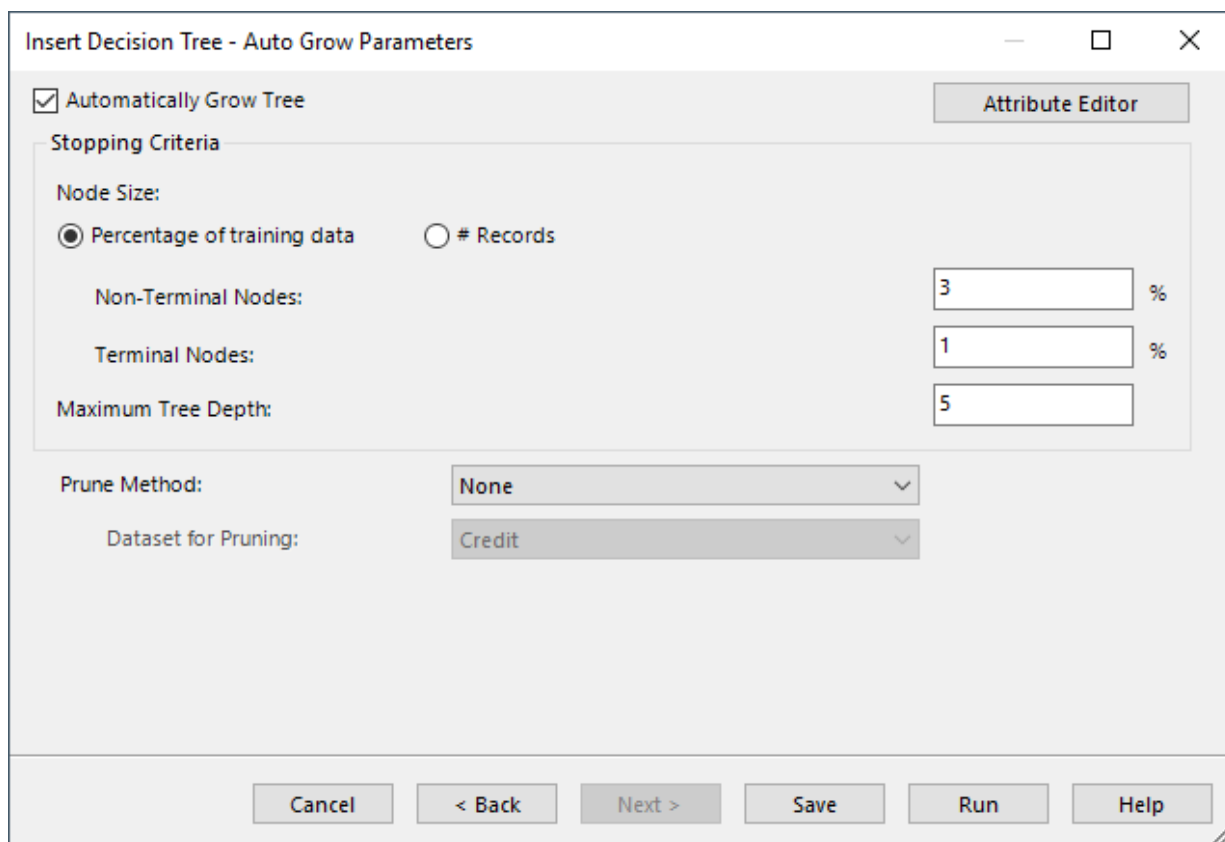
The split report displays a list of all the variables available for you use in your model. Additional split report parameters are also part of the split report. To calculate and display the split report statistics click the **Analyze** button.

After initializing the split report with the **Analyze** button, the variables are ranked in order of importance, or significance, with respects to the dependent variable. If the variable is not considered important, the **Rank** and **Info** columns will remain blank for these variables. The additional split parameters that are calculated are related the Measure. Different split parameters are displayed depending on which measure you chose in the previous wizard step. For more information on the columns in this report, see the [Decision Tree Split Report](#).

The split report statistics provide useful information that can help you decide which variables to include in your model. Because the variables are ranked by importance, the split report can show

you if and which variables are highly correlated. Depending on your defined business problem, you may or may not want to include these variables.

This step of the Decision Tree Wizard allows you to specify the Auto Grow parameters if you want the tree to be grown automatically upon insertion of the root node. You can also modify the attributes of the tree variables in the Tree Attribute Editor at this step using the **Advanced** button.



The box “Automatically Grow Tree” is unchecked by default (thus, by default, only the root node is created by the wizard). Check this box if you want the tree to be grown automatically upon insertion of the root node.

**Node Limits**  
**Percentage or Number of Records**

If the **Percentage** option is selected, the node size limits must be specified in percentage of total. This option is selected by default and is recommended, since it is logical to set these limits relative to the size of the input dataset. Select the option **# Records** if you want to specify the node size limits in absolute numbers of records.

**Minimum Node Size for**

The minimum number of records for a node to contain before the **Find Split** or **Automatic Grow** operations are

|                                              |                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Non-Terminal Nodes</b>                    | allowed to split it. For the option " <i>Percentage of training data</i> ", the default is 3%.                                                                                                                                                                                                                                                                               |
| <b>Minimum Node Size for Terminal Nodes</b>  | The minimum number of records allowed for terminal nodes. For the option " <i>Percentage of training data</i> ", the default is 1%.                                                                                                                                                                                                                                          |
| <b>Maximum Tree Depth for Automatic Grow</b> | The maximum number of levels allowed for the <b>Automatic Grow</b> command to grow the decision tree. The number of levels includes the root node, so the depth of a tree with a single split is 2. Enter the value of zero (0) for unlimited depth (in that case, the depth is limited only by the minimum node size constraints defined in this dialog). The default is 5. |

## Pruning Options

In order to use the pruning feature, you must have at least one more dataset available in the project aside from the parent dataset of the new tree. Normally, you should first insert partitions of the original dataset and then insert your tree based on the first partition and specify the second partition as the dataset for pruning. The following pruning algorithms are available:

- Critical-Value
- Error-Complexity
- Reduced-Error

See the Decision Tree section of Mathematical Formulations for a detailed description of these pruning methods.

See also: [Supported Measures](#)

## Supported Measures

The measure determines the method of evaluating the significance, or importance, of a split. The best split, or the most significant split, is the one that is displayed after a [Find Split](#) command. Each method uses different statistics. The values of these statistics are displayed in the [Split Report](#). A brief description of the available measures is given below. A more detailed description is available in the [Decision Tree](#) section of Mathematical Formulations.

## Unadjusted - Raw P-value Measure

P-value is the measure used to determine the significance of each split in a tree. Six statistics are calculated for this measure, **Info**, **P**, **C**, **F**, **df1**, **df2**. See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

F is the F-Ratio Statistic - A value calculated as part of the ANOVA procedure. The larger this number is, the greater the distance between the means or average values of the nodes in the split. ANOVA Procedure - (ANalysis Of VAriance). A procedure used to detect statistically significant effects induced by an independent variable on a continuous dependent variable. The ANOVA procedure employs an F-test to measure the differences between a given set of population means, where  $F = \text{Mean Square for Treatments} / \text{Mean Square for Error}$  ( $F = \text{MST}/\text{MSE}$ ).

The P-value is the probability that the difference being observed occurs by chance. The higher the P-value the more likely the observed differences were random. The **Info** is **1-P**.

## Adjusted - P-value Bonferroni Adjustment

Adjusted P-value is the measure used to determine the significance of each split in a tree. Eight statistics are calculated for this measure, **Info**, **df**, **Chi2**, **P**, **C**, **uP**, **uC**, and **B**. See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

The Bonferroni adjustment uses multiple test comparisons and adjusts the significance level (P-value) accordingly. The P-value used as the cutoff is divided by the total number of comparisons to be made. Any single comparison would then be compared to this adjusted P-value. For example, for the cutoff,  $p < 0.05$  and using 10 comparisons, any single test statistic would be compared to  $p < 0.005$ . The **Info** is **1-P**.

In a nutshell:

- P is the Bonferroni-adjusted P value.
- C is confidence.  $C = 1 - P$ .
- B is the B factor (Bonferroni adjustment factor for the P value).
- Bonferroni adjustments are used to minimize the discovery of chance relationships in a dataset. Bonferroni adjustments automatically adjust the level of significance used when testing the validity of a relationship.
- uP – unadjusted P-value for Chi2 and F tests.
- uC is unadjusted C.  $uC = 1 - uP$
- Chi-Squared Test - (Chi2). A test using the chi2 statistic measuring the statistical association between two categorical variables.
- df – the number of observations minus the number of necessary relations, which is obtainable from these observations ( $df = n - r$ ). The definition of "necessary relationship" is simply

defined as the relationship between a dependent variable (Y) and each independent variable (X) in the research.

## Entropy Variance - Non P-value Information Gain Measure

This measure uses statistics based on entropy or variance to determine the importance of each split in a tree. The statistics calculated for this measure are:

- for discrete dependent variables: **input entropy**, **output entropy**, **ratio entropy**, **info** (ratio entropy expressed as percentage), and **Information Value**
- for continuous dependent variables: **input variance**, **output variance**, **ratio variance**, and **info** (ratio variance expressed as percentage)

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

For decision trees, entropy measures the variability of the distribution of the dependent variable (DV) values with respect to the independent variables (IV). The value of entropy is a number in the interval [0,1]. An entropy of one indicates high variability in the distribution and an entropy approaching zero indicates low variability. As the variability is reduced, there is a gain in information.

In general, the output entropy value will decrease (approach zero) as the importance of the split increases. The value of info shows the importance of a split expressed in a percentage. The higher the percentage, the more important the split.

## Gini Variance - Non P-value Gini Measure

This measure uses the Gini coefficient, or Gini index, to determine the importance of each split in a tree. The statistics calculated for this measure are:

- for discrete dependent variables: **input index**, **output index**, **ratio index**, **info** (ratio entropy index in percentage), and **Information Value**
- for continuous dependent variables: **input variance**, **output variance**, **ratio variance**, and **info** (ratio variance expressed in percentage)

See [Decision Tree](#) in Mathematical Formulations for detailed equations for these values.

For decision trees, the Gini index compares the equality of the distributions of the dependent variable (DV) values with respect to the independent variables (IV). The value of the Gini index is a number in the interval [0,1]. A Gini index approaching 0 indicates that the distributions are equal, a Gini index approaching 1 indicates that the distributions are not equal.

## Creating and Modifying Splits

### Find Split

The **Find Split** operation searches for the independent variables (IV) that are most significant to the dependent variable (DV) based on criteria defined by the [measure](#). The independent variables that are found to be significant are called the split variable(s). The most significant independent variable is displayed with child nodes representing a category or range found in the independent variable. See more technical description of the splitting algorithm below.

Page | 307

The **Find Split** command may find multiple significant splits. The splits are ranked according to the measure and the first and most significant split is displayed in the tree view. The other splits can be viewed using [Next Split](#), [Previous Split](#), or [Go To Split](#) commands available through the right-click context menu or under the Reshape menu.

To do a **Find Split**, click on the node you want to split. The node becomes selected, which is indicated by a blue outline along the border.

|        |        |         |
|--------|--------|---------|
| <= 50K | 12,435 | 76.38%  |
| > 50K  | 3,846  | 23.62%  |
| Total  | 16,281 | 100.00% |

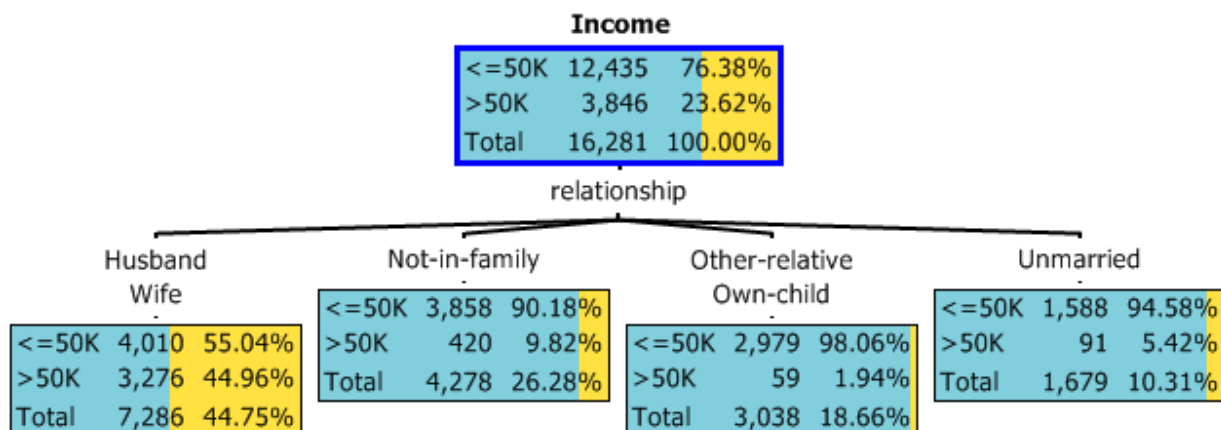
#### From the menu:

Grow | Find Split

#### From the context menu:

right-click on the node and select Find Split

For example, applying the Find Split operation to the node above may yield the following split:



For advanced options that define the characteristics of branches produced by Find Split, see the topic [Using the Attribute Editor](#).

NOTE: You can undo any sequence of Find Split operations using the Undo command under the Edit menu, or pressing Ctrl Z on your keyboard.

## Technical Details of the Splitting Algorithm

The underlying algorithm of the Find Split operation is as follows. (Note that it also governs the Automatic Grow operation, since Automatic Grow is just a recursive application of Find Split.)

The Knowledge Seeker tree growth algorithm (KSTGA) accessed via the **Find Split** or **Automatic Grow** functions attempts to solve a constrained combinatorial optimization problem. Given  $N$  input categories of a predictor, where each containing a measurement of the predictable, a cost function appropriate to the nature of the predictable (discrete vs. continuous), the algorithm finds  $M$  proper subsets of the  $N$  categories such that

- (a) the subsets meet constraints imposed by the nature of the predictor (ordered, unordered, floating) as well as special properties of the solution (each output category must meet minimum 'size' requirements), and
- (b) the cost function is minimized.

As an example, if  $N = 5$  and  $M = 2$ , and we just use numbers for the categories (they can be any value -- strings, integers, or other types) for notational simplicity, then:

```
in = 1
    2
    3
    4
    5
```



are the input categories, and

```
out = 1 4 5
      2 3
```

might be the solution. The "1 4 5" (and "2 3") can be called a "group", and the act of making a group from smaller groups is called "merging".

Page | 309

Each input category of the predictor, or IV, has some information about the predictable, or "DV". For discrete DV's, this information is the discrete distribution of the DV for that value of the IV. For a continuous DV, this would be the mean and variance of the DV for that value of the IV.

The cost function measures the statistical similarity of the DV for two or more groups of categories. For example, with a discrete DV, a chi-square test is used, and for a continuous DV, a simple analysis of variance is conducted. The larger the cost function value is, the more "similar" the groups are with respect to the DV.

The problem is one of finding a sequence of merges that produce groups that give the smallest value of the associated cost function -- that is to say, when the distribution of the DV in each group is \*most different\*, thus "best explains" the DV.

The search for the merges is conducted with respect to the class of the IV. If the IV is declared to be an 'ordered' variable, then merges are only conducted between "adjacent" groups after they have been put into "order". For the 'unordered' case, any set of groups can be merged without regard to 'order'.

The 'floating' case is a simple variation on the 'ordered': any missing value is removed from the input categories and held to the side. The problem is then considered to be 'ordered' and run normally. The last step of the algorithm will then take the missing category and search for the group which it "best fits" and place it there.

In addition, there are algorithm parameters that specify when a merge is actually performed, minimum 'group' size (as defined by the number of cases in a group, as opposed to the number of IV values), a merging threshold (below which groups are not merged), a break-apart threshold (above which groups are not dissociated), and whether or not an "exhaustive" search is conducted.

The basic structure of the algorithm is then:

```
- initialize-start-point
- merge-categories
  if(exhaustive)
    - initialize-start-point
    - do-restore-exhaustive
  else if(break_apart)
    - do-break-apart
- handle-small-nodes
  if(floating)
```

---

---

**- *handle-floating***

Description of the steps:

### **INITIALIZE-START-POINT**

Page | 310

The start point is N groups, each one consisting of a single input IV value. For the 'floating' constraint, at this point,

### **MERGE-CATEGORIES**

The merge-phase locates the pairs of most-similar groups and merges them together. This process is then repeated. As noted above, merges are subject to the constraints defined by the type of IV (ordered, unordered) and whether or not the groups are in fact "similar" by comparison of the cost function to a merging threshold value.

#### **For the non-exhaustive case (cluster):**

Merging stops when either

a) there is only one group left,

OR

b) there is no pair of groups that is "similar" enough with respect to the "merging threshold".

#### **For the exhaustive case:**

Merging stops when

a) there is only one group left (i.e., the merging threshold is ignored)

AND

b) while merging, record the sequence of merges in a special history buffer, noting the "best", global, solution found to date.

Special data structures are used to make the search for similarity between two groups as fast as possible.

### **DO-RESTORE-EXHAUSTIVE**

If the exhaustive analysis is selected, then after merging, a re-initialization to the start point is performed, and the sequence of merges recorded in the aforementioned "history buffer" is performed. This process stops as soon as the "best" history entry (i.e., the best solution) is encountered.

### **DO-BREAK-APART**

During the merge phase, it is possible that groups were merged when, from the standpoint of the global solution, they should not have been merged. The "break apart" mechanism attempts to partially correct this tendency to be a bit greedy.

It works by examining each group, and chopping it into smaller groups (again, subject to the usual constraints), examining the broken group against a break-apart threshold (it won't break it apart if the broken pieces of the group are themselves still similar with respect to the DV), and re-testing the solution for optimality (i.e., did the break-up actually make the solution better).

## HANDLE-SMALL-NODES

Any group that has too few cases (or samples, or records) is merged with the group that is most similar to it with respect to the DV.

## HANDLE-FLOATING

If the IV is 'floating', then the missing-value category is merged into the most similar group.

See also:

[Mathematical Formulations: Decision Tree](#)

[Measure](#)

[Automatic Grow](#)

[Force Split](#)

[Using the Attribute Editor](#)

## Automatic Grow

The **Automatic Grow** command allows you to automatically grow a tree to the size specified in the automatic grow parameters. Each split on a node is the strongest statistical split for that node.

The **Automatic Grow** can be started on any node. Any previous **Find Split** or **Force Split** results above the current node will remain in the tree.

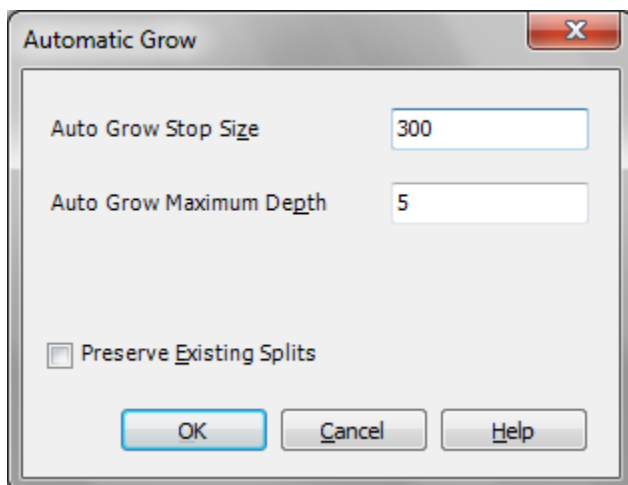
To use this command you must have at least a root node of a tree and the tree must be active. The **Automatic Grow** command is called:

### From the menu

Grow | Automatic Grow...

### From the popup menu:

on the parent node, right-click | Automatic Grow...



|                                 |                                                                                                                                                                          |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Auto Grow Stop Size</b>      | Minimum number of records in a node needed before the <b>Automatic Grow</b> will attempt to split the node.                                                              |
| <b>Auto Grow Maximum Depth</b>  | Determines the number of levels to grow in a tree. A value of <b>zero</b> leaves growth limits to other factors. The number of levels is measured from the current node. |
| <b>Preserve Existing Splits</b> | If checked, the existing tree is preserved and the growing starts from the terminal nodes. Otherwise, all splits are erased and Automatic Grow starts from scratch.      |

You can change the default Auto Grow parameters at any time using the **Tools | Options** or **Tools | Preferences** menu commands (Tree Growth tab). Options change the defaults for the current tree only, whereas Preferences change them globally (for all trees to be inserted in the future). See the topic [Preferences: Tree Growth](#) for details.

If you change the Auto Grow Maximum Depth parameter to 0 (unlimited depth), the Automatic Grow command grows the tree until the other conditions are met (for Minimum Node Creation Size, Auto-Grow Stop Size, and Minimum DV Values) or there is no more significant splits found for any node (whichever happens first).

The Automatic Grow process is controlled by the parameters in the Tree Growth and Tree Training tabs of the **Options** (or **Preferences**) dialog box.

## Limiting Automatic Grow based on DV values

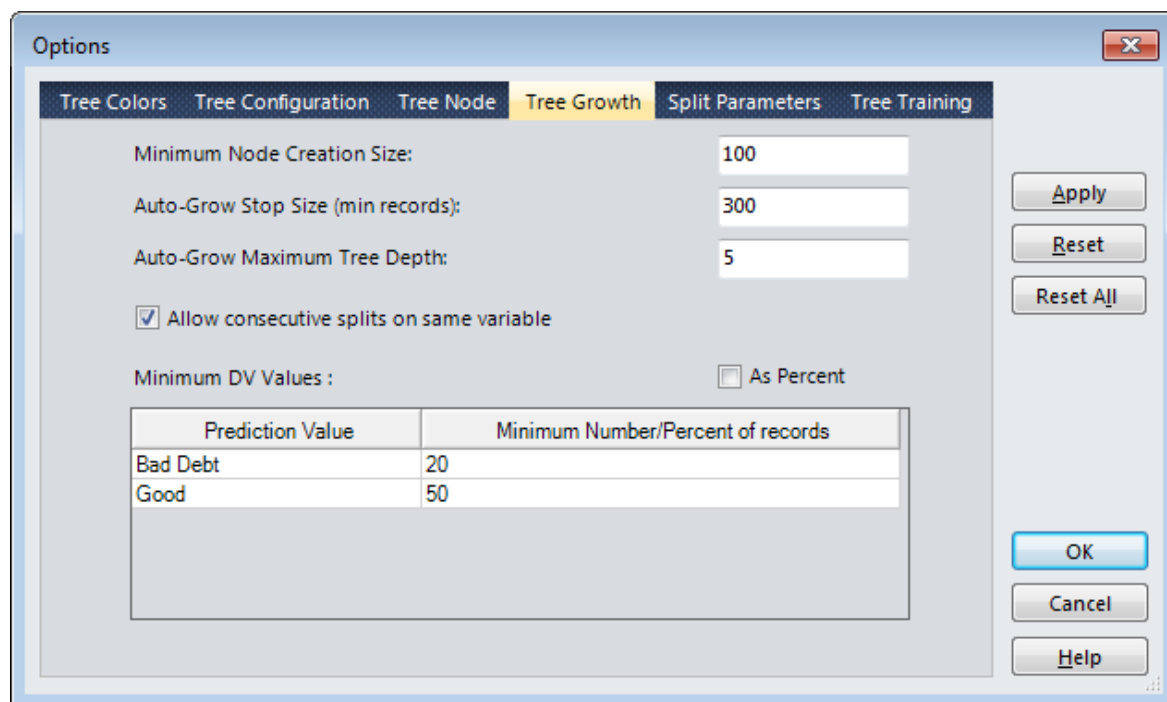
Aside from specifying the Auto-Grow Stop Size, Maximum Tree Depth, and Minimum Node Creation Size, you can also limit your Automatic Grow operation depending on the number (or percentage) of specific DV values in the nodes. This is done in the Tree Growth tab of the tree options dialog (**Tools | Options**).

For example, suppose that the dependent variable is AccountStatus, with values 'Bad Debt', 'Closed', and 'OK'.

If you specify the thresholds for Minimum DV Values as shown in the screenshot below, then the nodes where the number of records with AccountStatus = 'Bad Debt' is less than 150 or with AccountStatus = 'Closed' or 'OK' is less than 320 will **not** be created.

Page | 313

If you want to express your thresholds in terms of percentage of total number of records, select the checkbox 'As Percent' and enter the desired percentages.



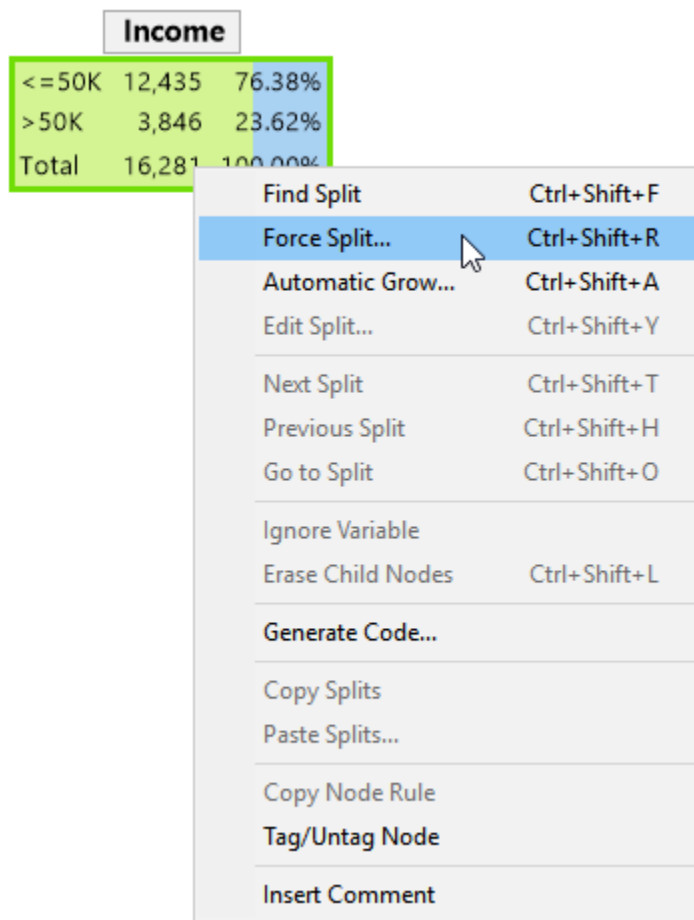
See also:

[Find Split](#), [Force Split](#)

## Force Split

Unlike Find Split, the **Force Split** command allows you to choose the independent variable to split on and gives you complete control over the bins of the split variable. In the default mode, Force Split does not use any algorithm to determine the most informative splits or rank independent variables. The split control is fully manual - the initial binning is "Equal Width" or "Equal Height", and the user is free to define their own custom bins starting from that point.

Right-click on the node of the tree that you want to split and select **Force Split** from the context menu. Alternatively, select the **Force Split** command from the **Grow** menu or use the keyboard shortcut CTRL + SHIFT + R.



A series of steps takes you through the **Force Split** process. This includes choosing the split variable and defining the binning for it.

- The **Force Split** command does not use the decision tree algorithm and therefore ignores the settings in the Attribute Editor that are relevant to the Find Split algorithm.
- If the cardinality of a discrete (nominal) split variable is greater than the *Max Bins* threshold defined in the Tree Attributes tab of Preferences, then the split variable is mapped into the number of bins equal to  $Max\ Bins = N$  ( $N-1$  most populous categories from the variable remain in their own category, while the others are mapped to the overflow bin, represented as ###). The default *Max Bins* is 50. For details, see the topic [Preferences: Tree Attributes](#).
- The **Settings** command allows you to increase the number of bins (for example if you have more than 10 categories, you can show them all rather than using the overflow bin). A high cardinality split variable will be limited to one hundred bins/categories. There will be ninety-nine categories (or ranges) and the hundredth (100th) bin will be the overflow bin, represented as ###.
- If the selected variable is continuous, then it is initially mapped into a number of intervals defined by the *Max Intervals* parameter in the Tree Attributes tab of Preferences. The default is 10. For details, see the topic [Preferences: Tree Attributes](#). These bins can be changed by editing the splits directly or using the Settings command.

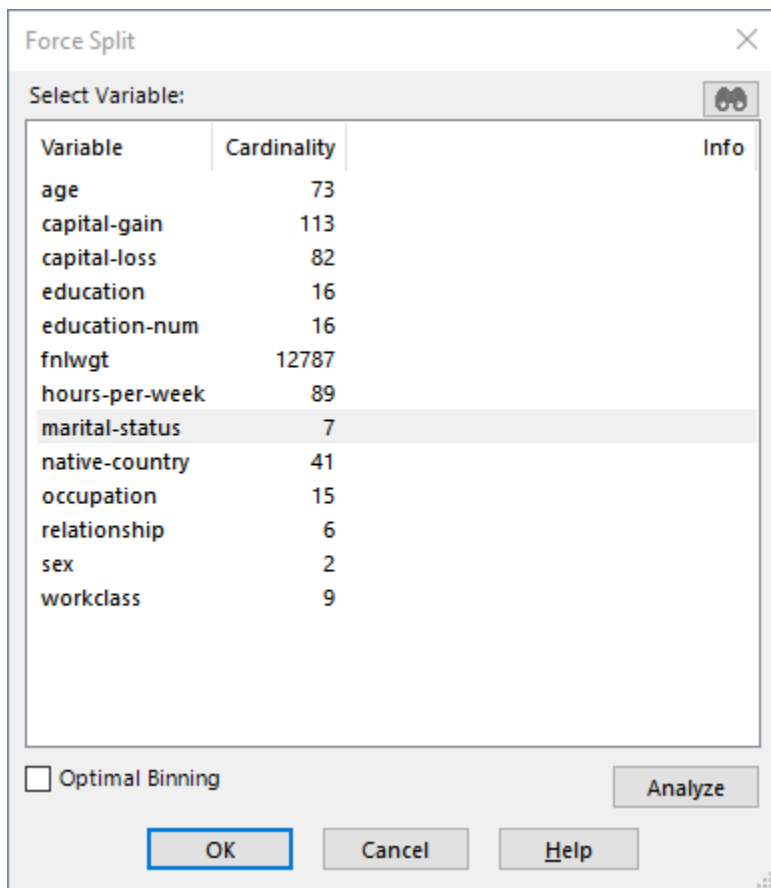
- When performing a force split on a node that contains existing splits, all the splits will be replaced by the force split command. This is the same as performing an Erase All before doing a force split. If you would like to preserve some of the child node information, you may consider the **Copy and Paste splits** command.

See the topic: [Force Split: Choosing the split variable](#).

Page | 315

The **Force Split** dialog allows you to choose the variable to split on. The variables are displayed in alphabetical order. Only the variables included in the [Attribute Editor](#) are shown in the list.

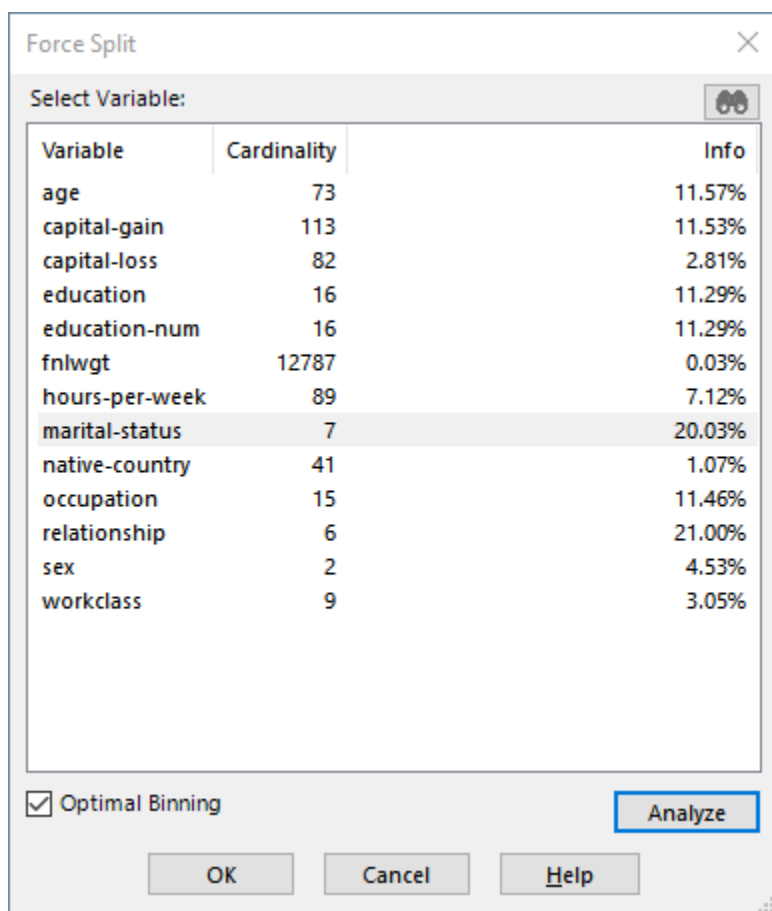
If the currently selected node already has child nodes, the existing split variable is highlighted in the Force Split dialog. Otherwise, the top variable in the list is highlighted.



By default, Force Split does **not** use any algorithm to determine the most informative splits or rank independent variables. Unless you select the *Optimal Binning* option, the split control is fully manual - the initial binning is "Equal Width" or "Equal Height", and the user is free to define their own custom bins starting from that point.

Click **Analyze** if you want to compute the relative significance of the variables with respect to the dependent variable. It will be displayed in the Info column. The way Info is calculated depends on the selected measure. See the topic [Measure](#) for details.

Select the **Optimal Binning** option in this dialog if you want to apply an intelligent algorithm to automatically produce the optimal binning for the selected variable - the most informative binning with respect to the dependent variable distribution. This is equivalent to performing the **Find Split** command followed by the **Go To Split** command and selecting the desired variable. For the details of the optimal binning algorithm, see the [Find Split](#) topic.



The Force Split dialog box contains a table of variables and their significance. The 'marital-status' variable is highlighted. At the bottom, the 'Optimal Binning' checkbox is checked, and the 'Analyze' button is highlighted.

| Variable       | Cardinality | Info   |
|----------------|-------------|--------|
| age            | 73          | 11.57% |
| capital-gain   | 113         | 11.53% |
| capital-loss   | 82          | 2.81%  |
| education      | 16          | 11.29% |
| education-num  | 16          | 11.29% |
| fnlwgt         | 12787       | 0.03%  |
| hours-per-week | 89          | 7.12%  |
| marital-status | 7           | 20.03% |
| native-country | 41          | 1.07%  |
| occupation     | 15          | 11.46% |
| relationship   | 6           | 21.00% |
| sex            | 2           | 4.53%  |
| workclass      | 9           | 3.05%  |

☒ Optimal Binning

Analyze

OK Cancel Help

Select the desired variable and click **OK**. This will bring up the Range Editor dialog. The Range Editor functions depend on the type of the independent variable selected. See the topics [Range Editor: Discrete](#) and [Range Editor: Continuous](#).

## Range Editor for Discrete Variables

The range editor allows you to modify the display of the branches of your discrete variable and how it will appear in the current split.



Range Editor - marital-status

|   | Source Value          | Number of Records | Target Branch | Histogram |
|---|-----------------------|-------------------|---------------|-----------|
| 1 | Divorced              | 2,190             | 1             |           |
| 2 | Married-AF-spouse     | 14                | 2             |           |
| 3 | Married-civ-spouse    | 7,403             | 3             |           |
| 4 | Married-spouse-absent | 210               | 4             |           |
| 5 | Never-married         | 5,434             | 5             |           |
| 6 | Separated             | 505               | 6             |           |
| 7 | Widowed               | 525               | 7             |           |

Group

Break

Reset

Settings...

OK

Cancel

Help

|                          |                                                                                                                                                                                                                                                                                                  |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source Value</b>      | Categories arranged as bins. Prior to any categories being grouped together, the number of categories displayed is equal the number of bins requested. The number of bins is set in the Settings. If there are more categories than requested bins, the remaining categories are grouped as ###. |
| <b>Number of Records</b> | Number of records associated with the bin.                                                                                                                                                                                                                                                       |
| <b>Target Label</b>      | The Target Label is only a reference.                                                                                                                                                                                                                                                            |
| <b>Histogram</b>         | Percentage of total records contained in the bin.                                                                                                                                                                                                                                                |

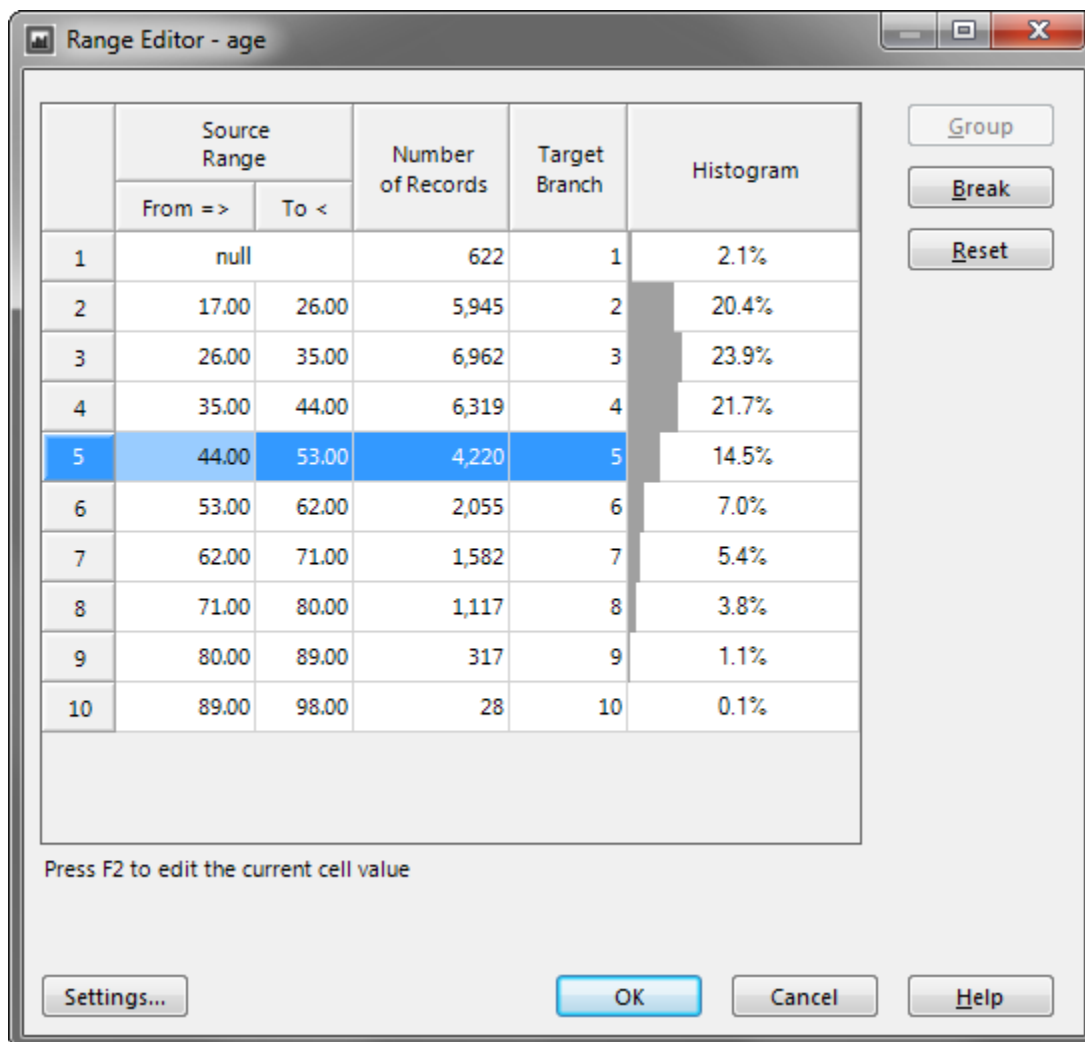
Bin Editing

---

|                 |                                                                                                              |
|-----------------|--------------------------------------------------------------------------------------------------------------|
| <b>Group</b>    | Allows you to combine bins.<br>Example                                                                       |
| <b>Break</b>    | Allows you to break apart grouped bins.<br>Example                                                           |
| <b>Reset</b>    | The reset button allows you to return to the original bins displayed when the range editor was first called. |
| <b>Settings</b> | The settings button allows you to change the default bin settings. See Settings for Discrete Variables .     |

## Range Editor for Continuous Variables

The range editor allows you to modify the display of the ranges as they will appear in the current split.



## Range Display

|                          |                                                                                                                                                  |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source Range</b>      | Indicates the beginning and ending range of values that for each bin.                                                                            |
| <b>From=&gt; To&lt;</b>  |                                                                                                                                                  |
| <b>Number of Records</b> | Number of records contained in the bin.                                                                                                          |
| <b>Target Label</b>      | Label for the bin as it will appear in the expression editor. The label can be modified by double-clicking on the cell and entering a new label. |
| <b>Histogram</b>         | Percentage of records in the bin.                                                                                                                |

---

## Range Editing

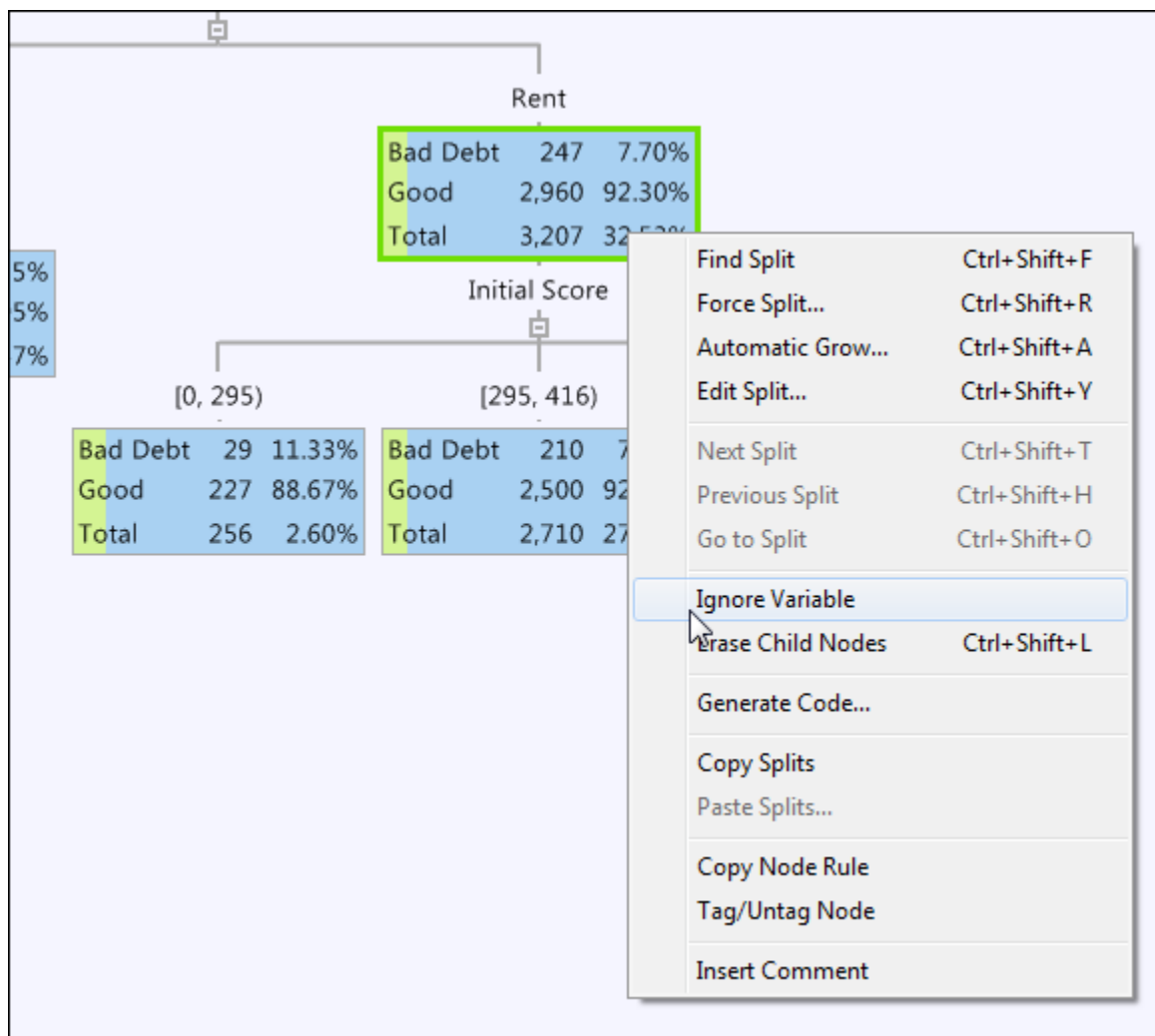
|                 |                                                                                                                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Group</b>    | Allows you to combine ranges. Only contiguous ranges can be grouped.<br>Example                                                                                                            |
| <b>Break</b>    | Allows you to create new ranges by breaking the existing ranges.<br>Example                                                                                                                |
| <b>Reset</b>    | The reset button allows you to return to the original ranges displayed when the range editor was first called.                                                                             |
| <b>DV %</b>     | Using the DV % button you can specify the desired percentage of each DV value in each child node of the split. See the topic <a href="#">Force/Edit Split with Requested DV Percentage</a> |
| <b>Settings</b> | The settings button allows you to change the default range settings. See Settings.                                                                                                         |

## Ignore Variable

The **Ignore Variable** command removes the split on the independent variable displayed under the currently selected node. If there are any background splits at the same node (for example, as is usually the case after Find Split), then the next split is displayed.

You can also exclude variables from the Attribute Editor, but the next split is not displayed in that case.

In the example given below, to ignore the split on **Initial Score**, right-click on the node and select **Ignore Variable** from the context menu.



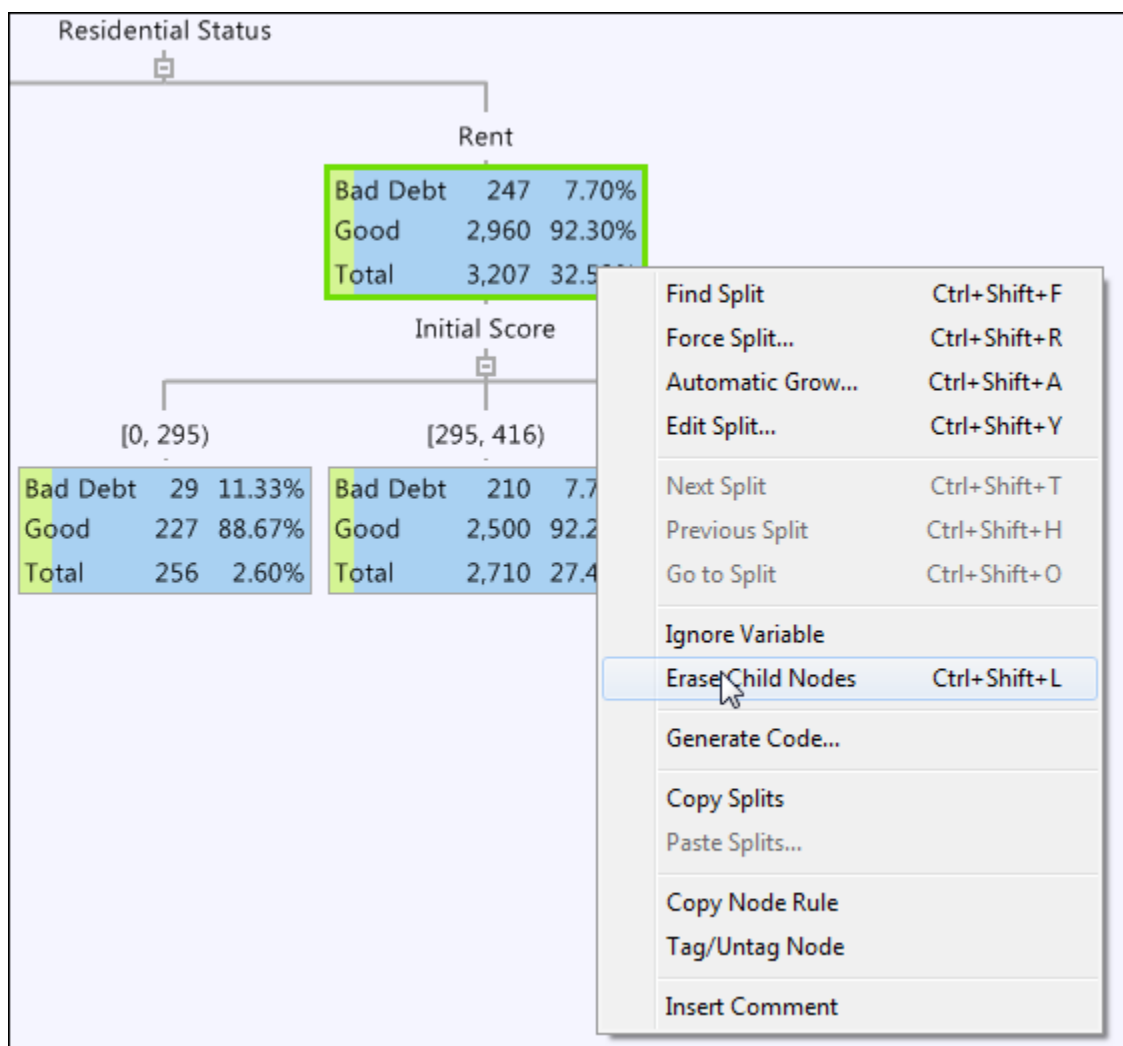
Alternatively, click on the parent node and select the **Ignore Variable** command from the Reshape menu.

See also: [Edit Split](#), [Find Split](#), [Force Split](#), [Erase Child Nodes](#)

## Erase Child Nodes

The **Erase Child Nodes** command deletes all child nodes of the currently selected node. If there are several levels of child nodes under the current node, all levels are erased.

To erase the child nodes of a specific node, right-click on it and select the command **Erase Child Nodes** from the context menu, as shown below. Alternatively, click on the parent node and use the keyboard shortcut CTRL + SHIF + L.



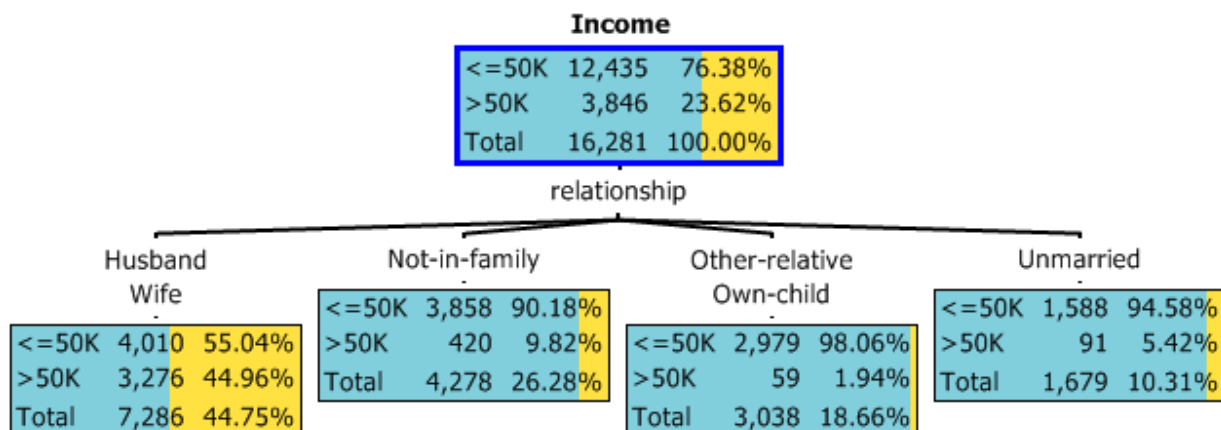
You can also use the **Erase child Nodes** command from the Reshape menu.

See also: [Edit Split](#), [Find Split](#), [Force Split](#), [Ignore Variable](#)

## Editing Splits

The **Edit Split** command allows you to modify the display of the ranges (for continuous variables) or bins (for discrete/categorical variables) in the currently selected split.

*Note:* If you have more than one significant split at the currently selected node (for example, resulting from a Find Split), then performing Edit Split on this node will affect only the visible split, and the other splits at this node will be preserved (the **Next/Previous/Go to Split** commands will still be enabled).



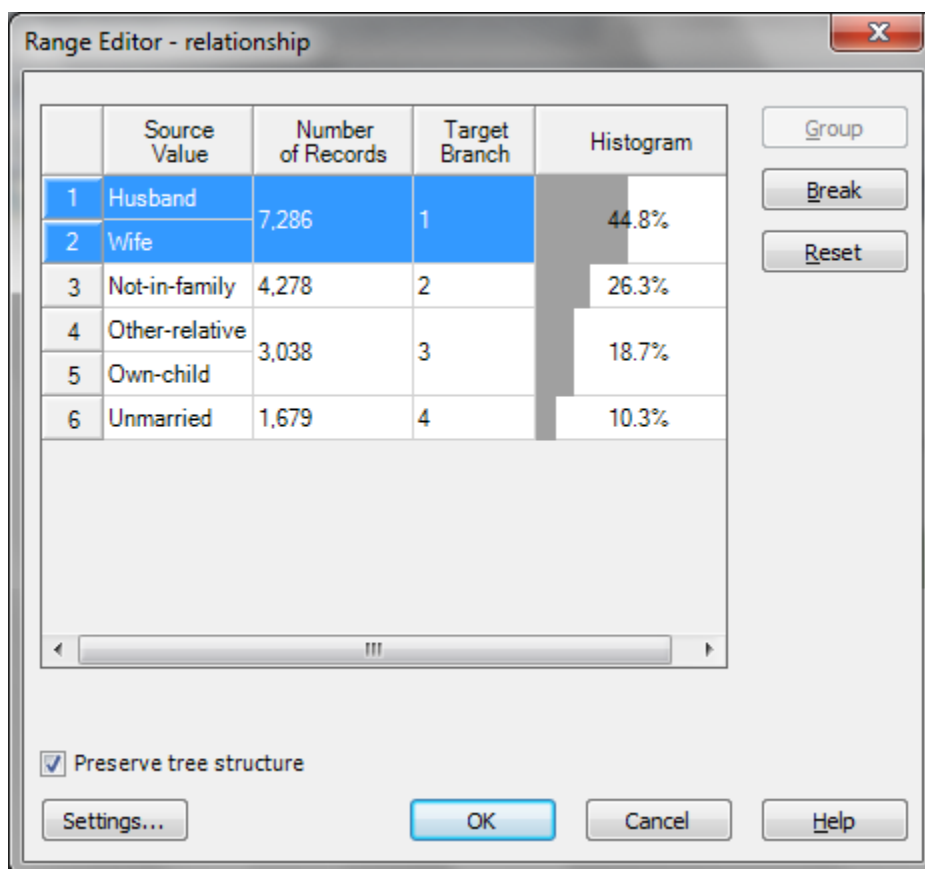
### From the menu:

Grow | Edit Split

### From the context menu:

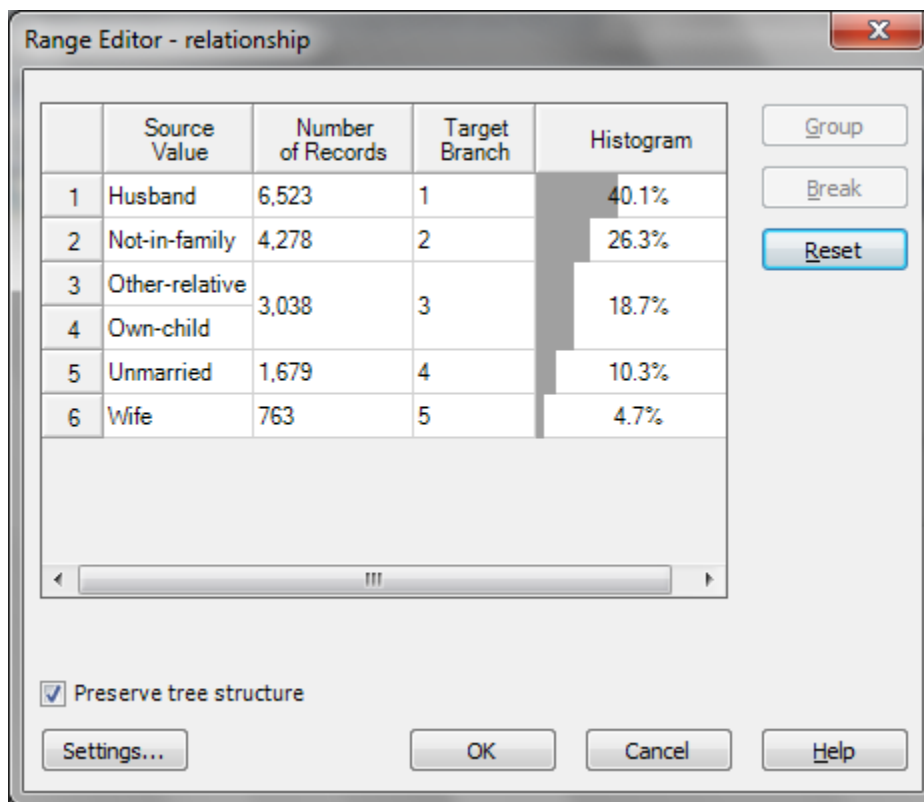
Right click on the parent node and select **Edit Split**

When you select the **Edit Split** command, it displays the distribution of bins/ranges at the current node in **Range Editor**, whose functions and options are exactly the same as in the case of the **Force Split** command. For example when editing the split shown above, we get:



For example, if you want to ungroup the categories **Husband** and **Wife** in this case, highlight the group and click the **Break** button. As a result, you will get the following distribution:





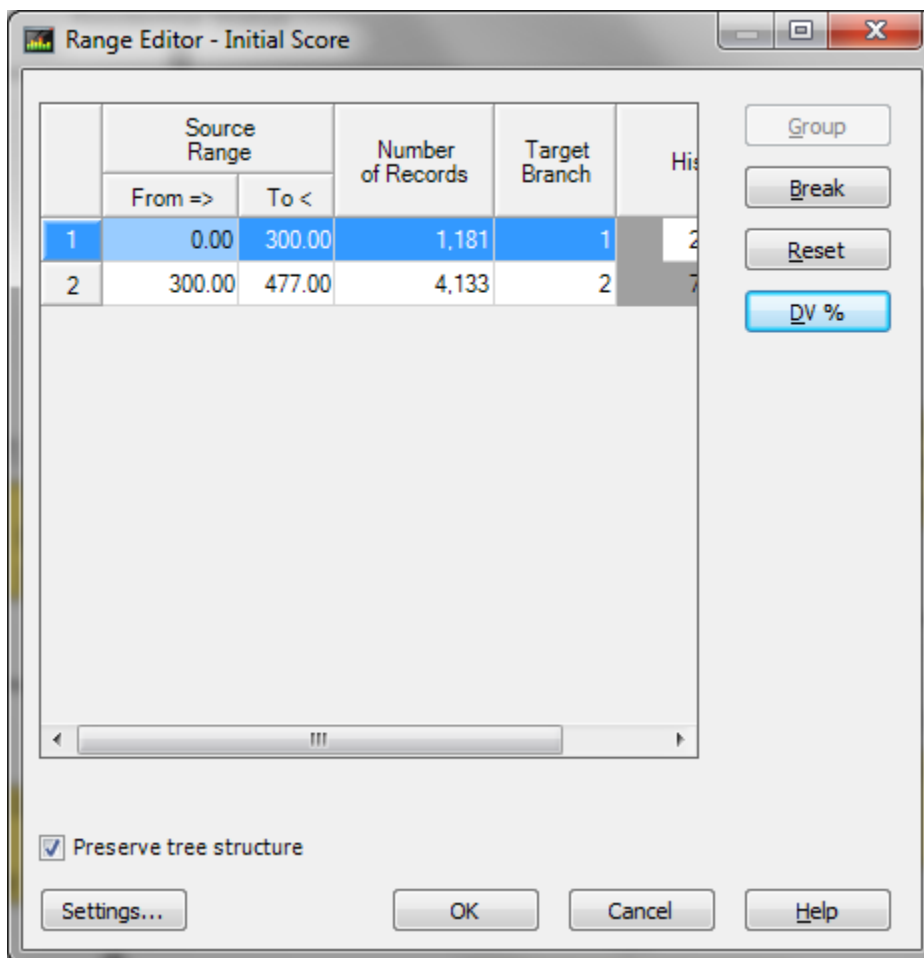
**Note:** If you perform an Edit Split operation of a node with grandchildren, you have the option of preserving the splits unaffected by the Edit Split whenever possible. By default, the unaffected splits will be preserved (the **Preserve tree structure** checkbox in Range Editor is selected).

Refer to the topics Range Editor (Discrete) and Range Editor (Continuous) for details about using the Range Editor for discrete and continuous independent variables.

**Note:** In the case of a continuous independent variable, real numbers that represent interval boundaries in Range Editor are displayed with fixed precision (2 digits after the decimal point) if the cells are not in Edit mode. To enter the Edit mode, click on a cell or press the F2 key. This will display the actual value with the original precision.

## Force/Edit Split with Requested DV Percentage

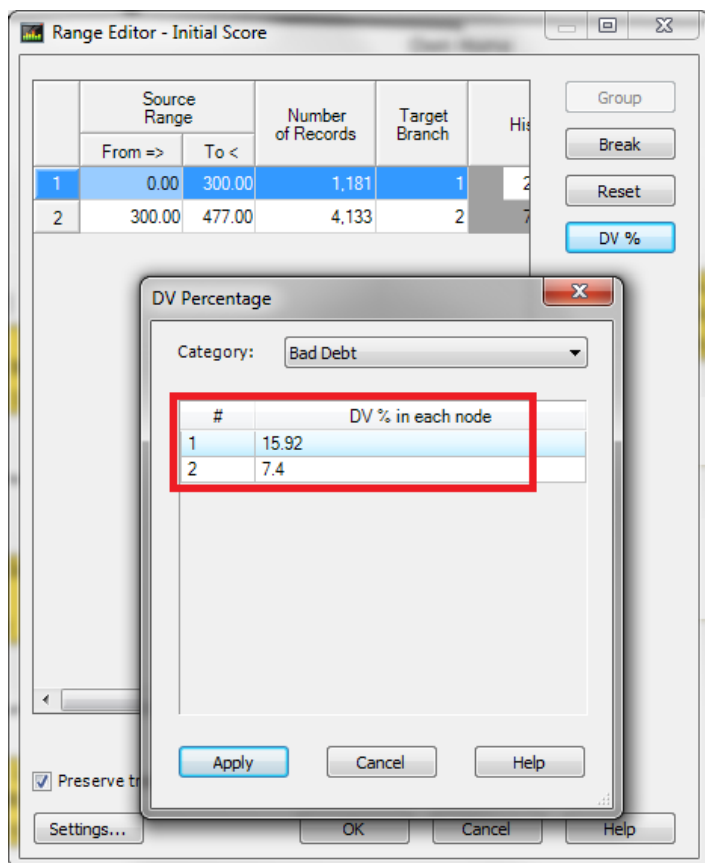
When [forcing a split](#) or [editing a split](#) on a continuous variable, the **DV %** button in the Range Editor provides the ability to specify the desired percentage of each category of the dependent variable in each of the child nodes in the split. This function is applicable only in Decision Trees with discrete dependent variable.



Clicking the **DV %** button brings up the **DV Percentage** dialog.

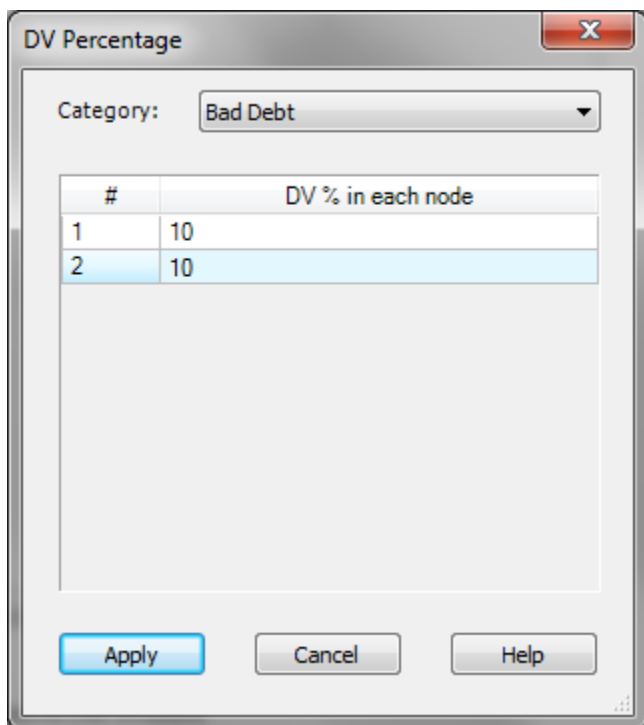
- First, choose the desired DV category from the Category drop-down list.
- In the grid below, specify the desired percentage for the chosen DV category in each child node of the split.

The example below illustrates the process of modifying the DV % when editing the split shown on the right-hand side. Initially, when the Bad Debt category is chosen, the dialog shows the current percentage of Bad Debt in each of the two child nodes of the split (15.92% and 7.4%).



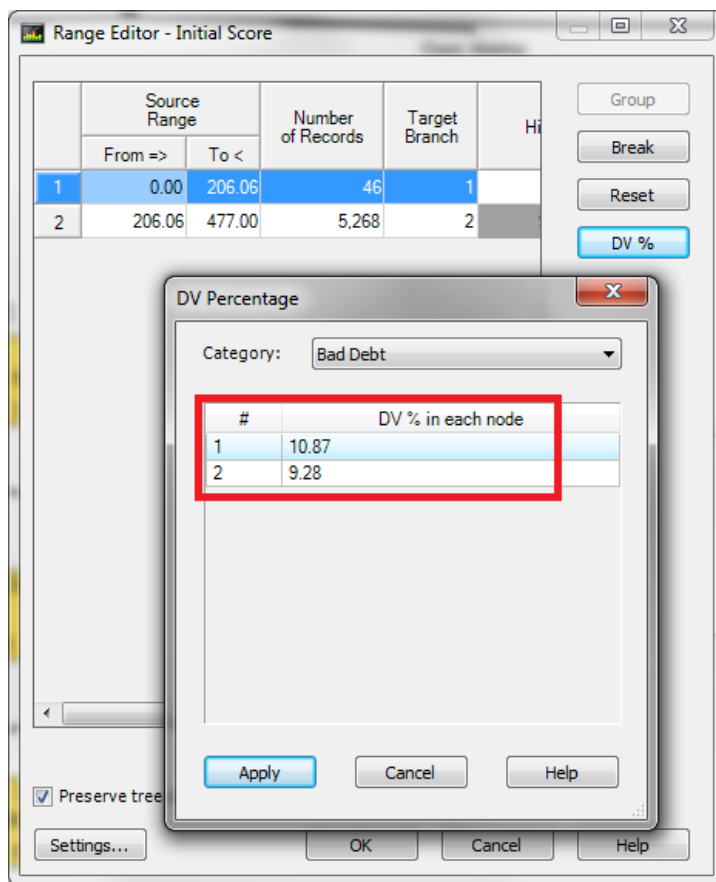
|               |       |        |  |
|---------------|-------|--------|--|
| Initial Score |       |        |  |
| [0, 300)      |       |        |  |
| Bad Debt      | 494   | 9.30%  |  |
| OK            | 4,820 | 90.70% |  |
| Total         | 5,314 | 26.03% |  |
| [300, 477]    |       |        |  |
| Bad Debt      | 188   | 15.92% |  |
| OK            | 993   | 84.08% |  |
| Total         | 1,181 | 5.78%  |  |
| Bad Debt      | 306   | 7.40%  |  |
| OK            | 3,827 | 92.60% |  |
| Total         | 4,133 | 20.24% |  |

For example, if the percentages of *Bad Debt* in the two child nodes are required to be approximately 10% in each node, specify **10** instead of the current 15.92 and 7.4, as shown below:



Click **Apply**. The boundaries of the intervals for the child nodes will be automatically adjusted to satisfy the new DV percentages specified. As shown below, the intervals  $[0, 300)$  and  $[300, 477]$  become  $[0, 206.06)$  and  $[206.06, 477]$ . Click OK in the Range Editor.

Note that it is not always possible to set the distribution exactly as specified. In this example, precisely 10% of *Bad Debt* in each child node is not achievable, so the percentages are automatically adjusted to 10.87 and 9.28, as can be seen in the resulting split.



|                          |       |        |  |
|--------------------------|-------|--------|--|
| Initial Score            |       |        |  |
| [0, 206)      [206, 477] |       |        |  |
| Bad Debt                 | 494   | 9.30%  |  |
| OK                       | 4,820 | 90.70% |  |
| Total                    | 5,314 | 26.03% |  |

|          |    |        |          |       |        |
|----------|----|--------|----------|-------|--------|
| Bad Debt | 5  | 10.87% | Bad Debt | 489   | 9.28%  |
| OK       | 41 | 89.13% | OK       | 4,779 | 90.72% |
| Total    | 46 | 0.23%  | Total    | 5,268 | 25.80% |

## See also:

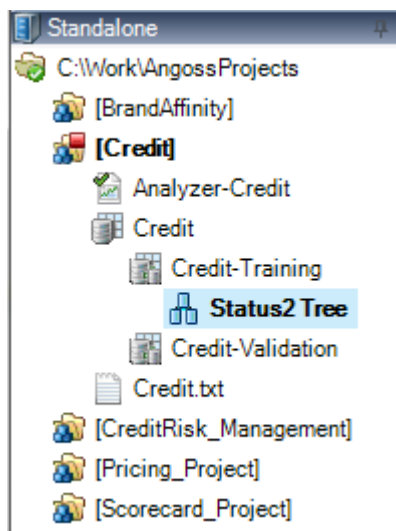
[Force Split](#)

[Edit Split](#)

## Saving Changes

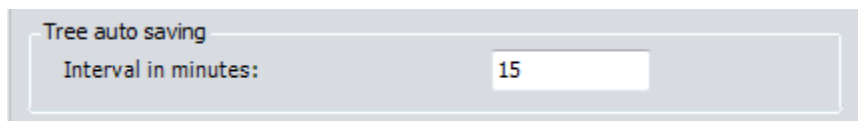
The changes you made to your decision trees or strategy trees can be saved by selecting the **Save Tree** command from the **File** menu or pressing **Ctrl+S** on your keyboard.

To indicate that a tree has been modified and the changes have not yet been saved, the name of the tree in Project Explorer is displayed in bold font. The names of the projects that have unsaved trees are also displayed in bold. For example, the AccountStatus tree in the CreditRisk project in this screenshot has been modified and the changes have not been saved yet:



When closing the tree or project, you will be prompted to save the trees you have modified.

The Auto-save feature is provided. You can set the time interval for auto saving trees using the **Tools | Preferences** menu command (in the **General** tab of the Preferences dialog). The default interval is 15 minutes.



## Navigating between Splits

The **Next Split** command displays the next split in the order of significance within the set of splits produced by the [Find Split](#) command on the selected node. The list of independent variables ordered by their significance with respect to the dependent variable is displayed in the [Split Report](#).

The Next Split command is enabled only if there is more than one split created by the **Find Split** command.

### From the menu:

Reshape | Next Split

### From the popup menu:

Right click on parent node | Next split

**Previous split** displays the previous significant split in the list of splits created by the [Find Split](#) command. This command is only enabled if there is more than one split created by the **Find Split** command.

Page | 331

### From the menu:

Reshape | Previous Split

### From the popup menu:

Right click on parent node | Previous split

The **Go To Split** command allows you to go to a different split on the node where a **Find Split** operation was performed previously. If there are no other significant splits calculated, this command is disabled.

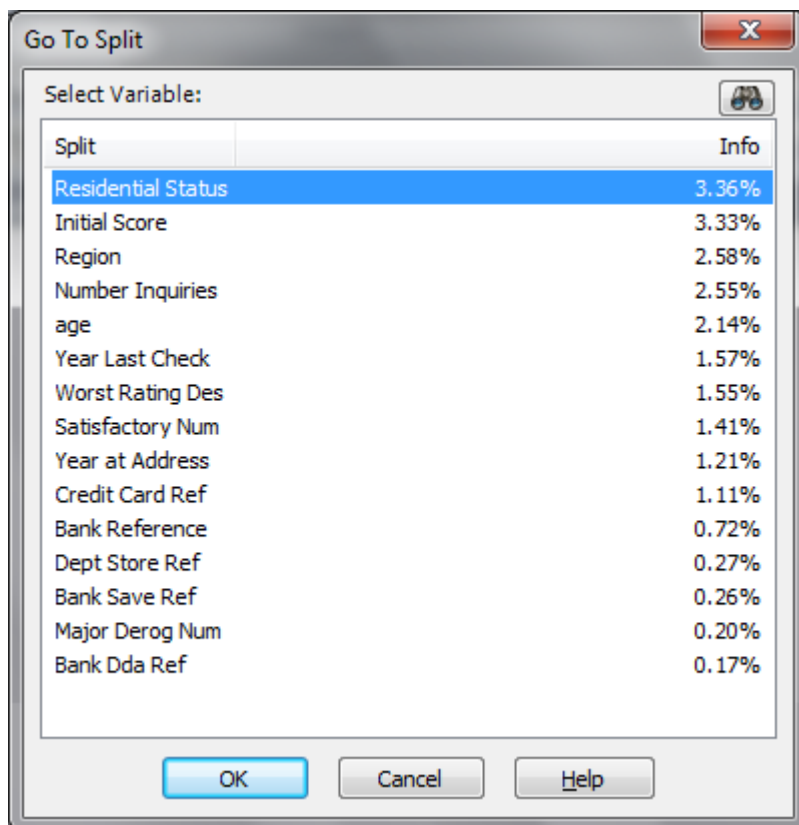
### From the menu:

Grow | Go To Split

### From the popup menu:

right-click on the parent node and select Go To Split

The **Go To Split** dialog displays the list of all the significant splits calculated for the given node in the order of their significance. The value displayed in the **Info** column is the same as in the [split report](#) for the current split. If the tree uses a **log p-value** measure, the splits are ranked by  $-\log(p)$  or  $-\log(\text{adjusted } p)$ , and the corresponding header is shown instead of *Info*. See the [Measures](#) topic for more details.



See also:

[Next Split](#)

[Previous Split](#)

[Measure](#)

## Viewing a Decision Tree

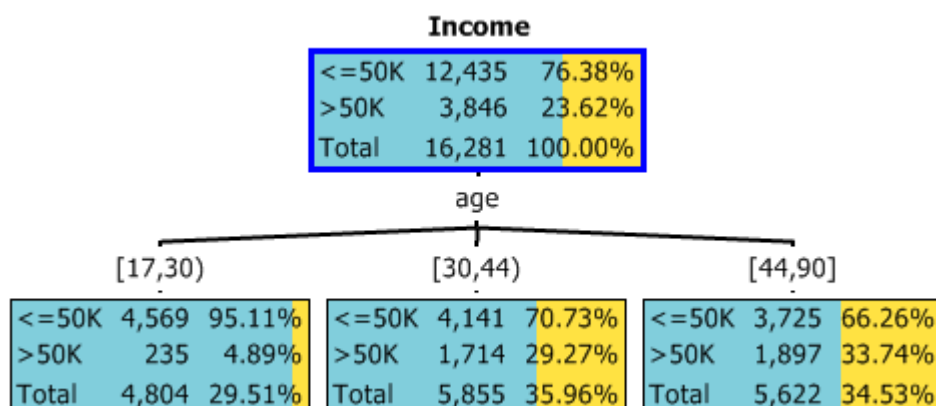
### Tree View

For decision trees, this is the main analysis view.

Initially, the tree view displays the root node, which represents all the records in the dataset. From there, records are split into groups called child nodes. These nodes in turn are further split until terminal nodes or leaves - which can no longer be split - are created.

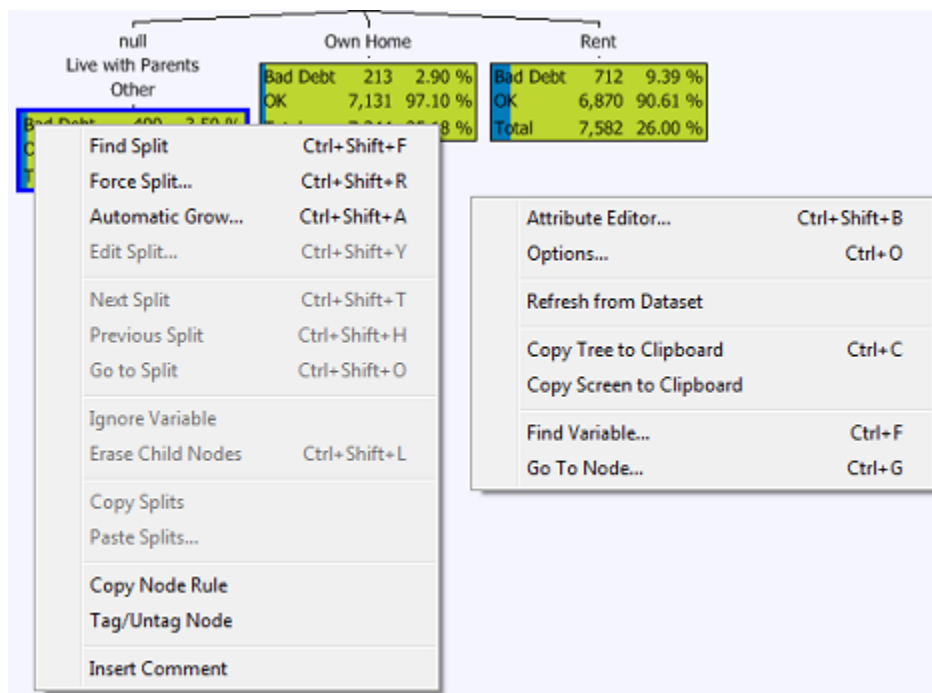
The tree view is interactive - you use it to split nodes one-by-one or to grow the tree automatically and then adjust it.





The name of the dependent variable (DV) is shown as a title at the top of the tree. If you want the DV name to be displayed in every node, use the **Tools | Options** command and select the *Show DV Name* option in the Tree Node tab of the Options dialog. The Options command is also available in the toolbar and in the context menu of the tree background.

Right-click menus in the Tree view are context-sensitive based on whether a node is clicked or the tree background. The context menu for the nodes contains only node-specific commands, such as *Find Split* and *Erase Child Nodes*. The commands applicable to the entire tree, such as *Attribute Editor* and *Find Variable*, are in the context menu for the tree background. This also applies to the [Tree Map](#) view.



Usually the tree view is used to look for the most relevant (and statistically significant) variables that correlate to the values of the dependent variable. After running the **Find Split** command, the most significant split is displayed first. You can view other splits using the [Next Split](#), [Previous Split](#), and [Go To Split](#) commands. As you move down through the tree, you are building a series of rules such as this one:

```
If a customer's age is between 17 and 30 then there is a 95.11 percent
chance that Income will be <=50K and a 4.89 percent chance that Income
will be >50K.
```

Page | 334

When you click or hover the mouse pointer over a node, the tooltip displays the node rule. If you need to copy the node rule to the clipboard, just right click the node and select **Copy Node Rule** from the context menu (or select **Edit | Copy Node Rule** from the menu). For example, the rule for the middle child node in the tree above is `[age]>=30 AND [age]<41`

Notice that the further you traverse down the tree, the more you build on the rule set. After just a few splits, you may have a set of rules that classifies records quite tightly.

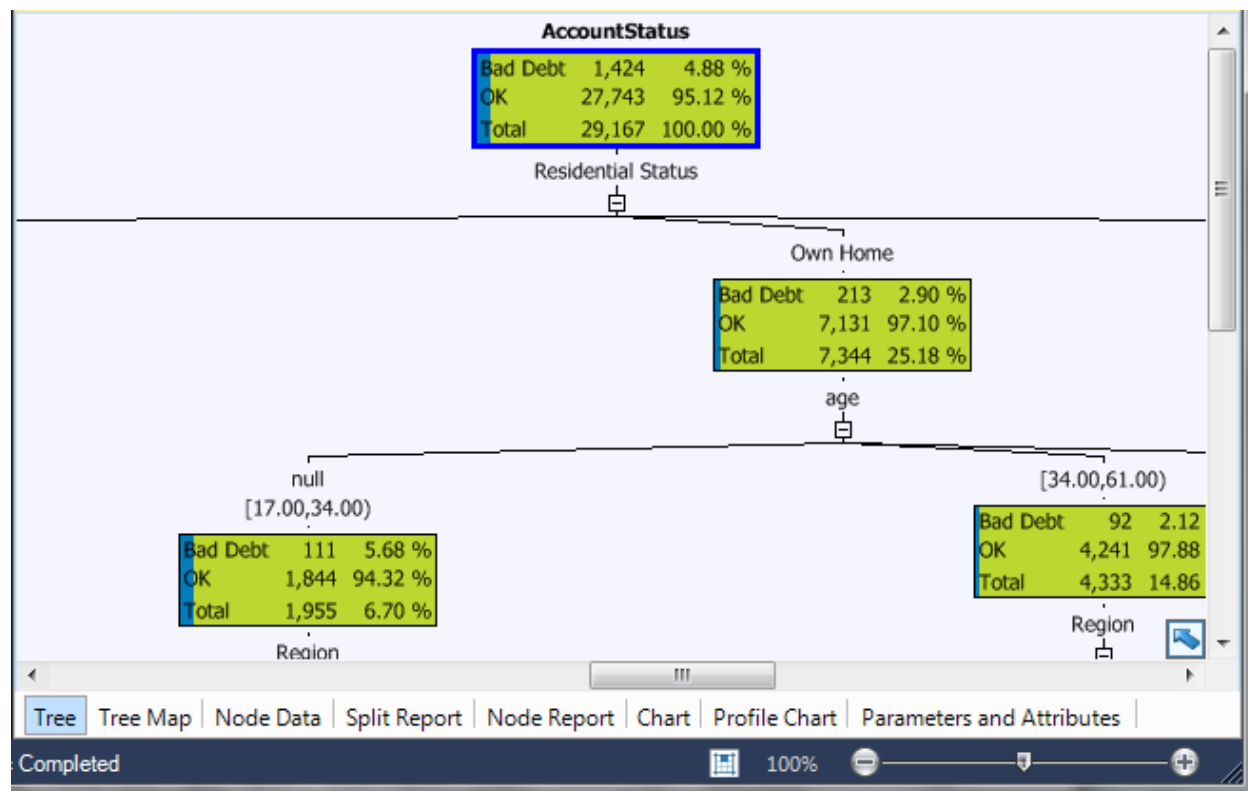
The primary tree commands are found on the **Grow**, **Reshape**, and **View** menus. The most common commands are also found on the context menu (right-click on a node to bring this menu up).

## Pan and Zoom

You can easily navigate across large trees using pan and zoom features.

Pan control allows the user to click and drag the tree instead of using the scroll bars. Just click anywhere in the tree view, hold the left mouse button down and start dragging to move around the tree.

The Zoom control allows you to zoom in or out to view more of the tree. A standard zoom control with a slider is provided in the Status bar, at the bottom right corner, as seen in the screenshot below. Aside from zooming by moving the slider and clicking the +/- icons, you can also use mouse wheel scroll with the CTRL button, or various keyboard shortcuts (CTRL+, CTRL--, CTRL-0).



The zoom commands are listed under the View menu: Zoom In / Zoom Out / Reset (to 100 %). The zoom percentage indicated by the slider position is relative to the current font size.

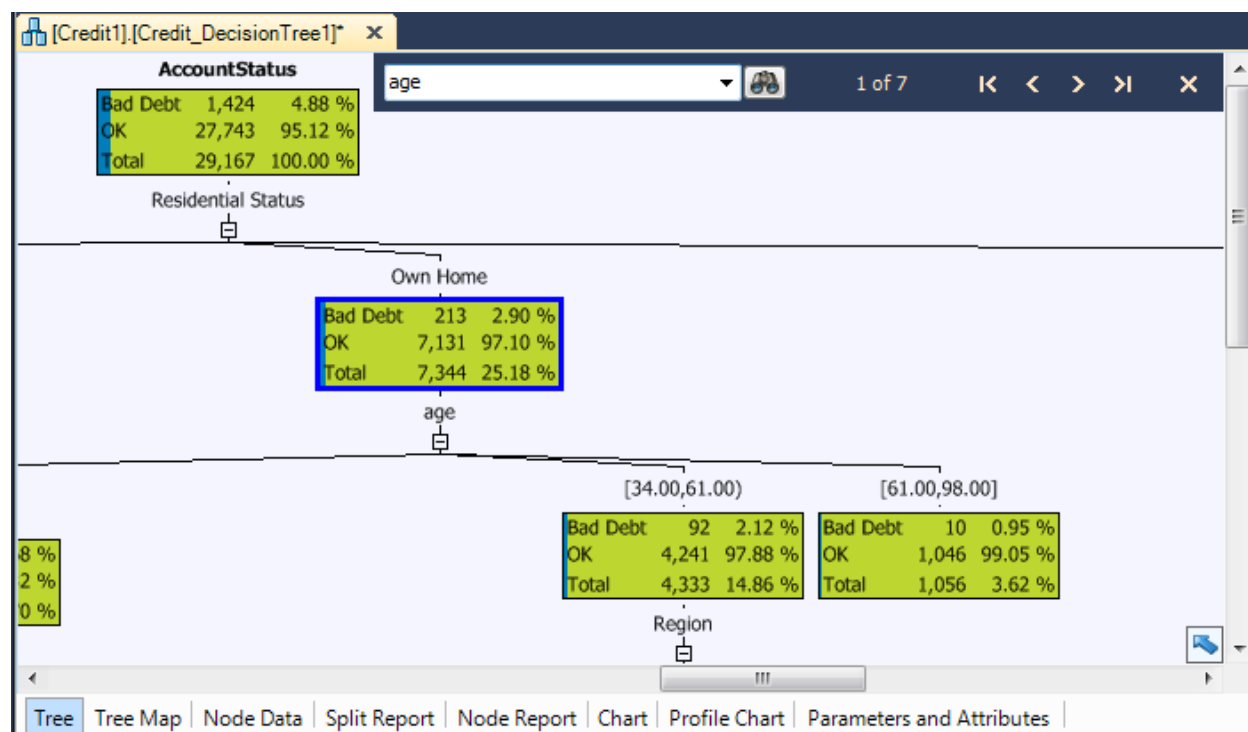
To the left of the zoom slider, there is a button to re-center the tree at the root node.

## Searching for a Split Variable

The Find Variable command allows you to search for a split variable in the tree and navigate between splits, which is especially convenient in large trees with multiple splits on the same variable.

The Find Variable control can be opened with a CTRL-F keyboard shortcut or via the edit menu. See the screenshot below. You can enter the name of the variable using the keyboard or select it in the drop-down list. The control will automatically go to the first split on that variable, and show how many splits in the tree use that variable. The command finds the variables in the same order as the Node Numbers: from left to right, top to bottom.

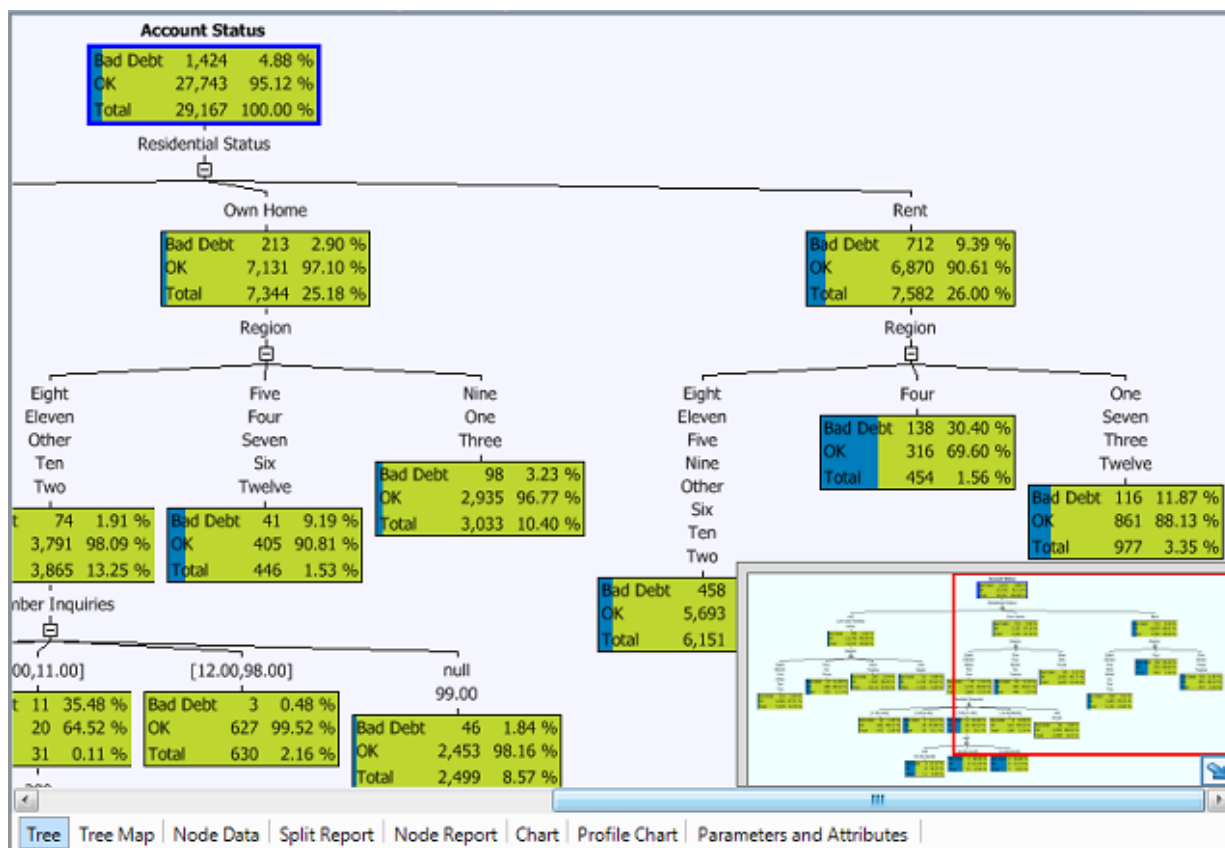
The Previous and Next buttons in the form of arrows find the previous and next occurrence of the variable. Pressing Enter while the focus is on the Find control finds the next occurrence of the variable. Pressing F3 when the focus is not on the Find control does the same. Pressing the Escape key closes the Find box.



## Mini-map

An interactive tree mini-map in the bottom right part of the Tree View shows the location of the current view of the tree relative to the whole tree and allows the user to jump easily to any part of the tree with a single click. The mini-map can be shown or hidden by clicking on the blue arrow in the bottom right corner. You can resize the mini-map by dragging its border or the top left corner.

A transparent rectangle with red border on the mini-map indicates the part of the tree shown in the Tree view. The fastest way to navigate to the desired part of the tree is to click on that part in the mini-map view. The Tree View will be updated immediately to show the part you clicked on.



## Tree Options

You can specify various Tree options by clicking the *Options* icon in the toolbar or selecting **Tools | Options** from the main menu. They are applied to the current tree only, while the same settings specified via **Tools | Preferences** are global and will be applied to all *new* trees (not existing ones). See the Preferences section for details. Also, see the subsection "Precision and Format of Numbers Displayed in Trees" provided below.

## Node Comments

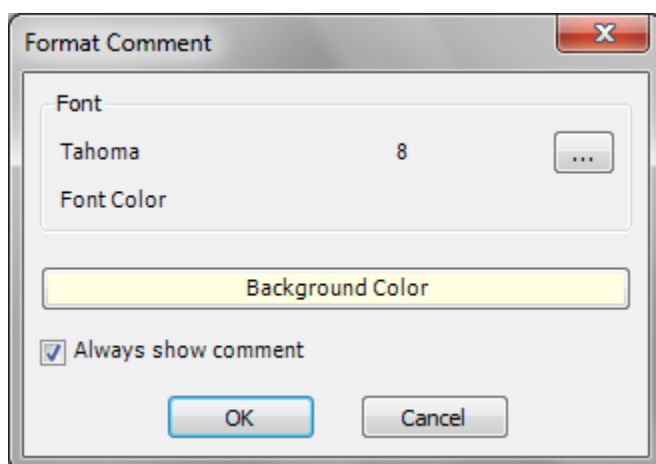
In addition to node tags, node comments is a convenient way to mark some nodes of your interest and put your comments, explanations, or instructions (for example, specify the way to treat the observations in a particular node).

Node comments can be added for presentation purposes if you plan to print or send images of your tree or even share your project.

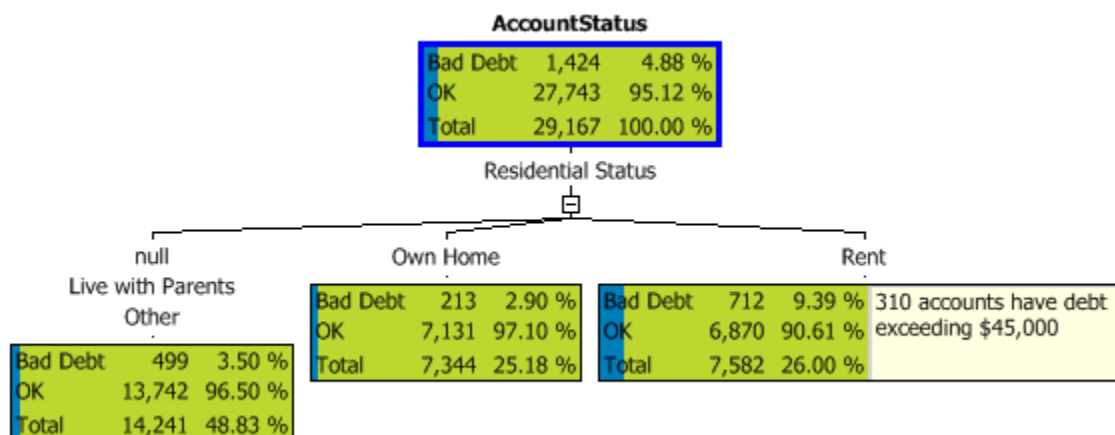
To insert a node comment, right-click the desired node and select **Insert Comment** from the context menu or select the node and choose **Edit | Insert Comment** from the main menu. Type your text in the comment window and click anywhere outside this window when finished. The comment will be saved with the node. This is indicated by a small red triangle at the top right corner of the node. Now the commands **Edit Comment**, **Delete Comment**, and **Format Comment** become available in the context menu of the node and under the Edit menu.

Page | 338

The text of the node comment appears as a tooltip when your mouse pointer is over the top right corner of the node. You can also choose to always display the text of the comment next to the node by right-clicking it and selecting **Format Comment** from the context menu.



After you select the checkbox 'Always show comment', the comment will be displayed in a box to the right of the node as in this example:



## Precision and Format of Numbers Displayed in Trees

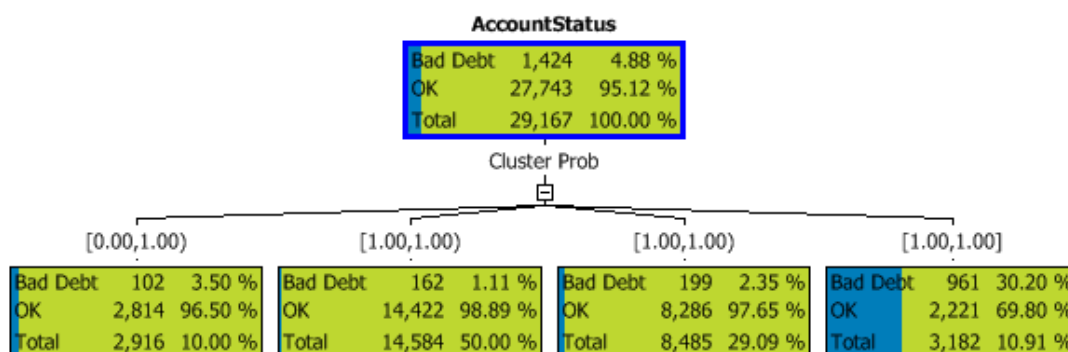
**You can adjust the precision and format of all numerical values displayed in trees:** value percentages of the discrete dependent variables, average and standard deviation of continuous dependent variables, values of independent variables in the split rules (displayed above the nodes), tree split parameters such as Ratio Entropy, Chi2, P value, etc. (if you chose the option to display them), and even the value frequencies, which may be non-integer if the tree is weighted.

Page | 339

To adjust the precision and format of numbers in the current tree only, use **Tools | Options** and Tree Attribute Editor (**Tools | Attribute Editor**).

To set global precision and format preferences for all new trees to be created, use **Tools | Preferences** and Dataset Editor. Changes made in Dataset Editor can also be applied to existing trees using the menu command **Tools | Refresh From Dataset**.

To adjust the precision and format for independent variables in the current tree only, use the Tree Attribute Editor (**Tools | Attribute Editor**). For example, in the following split on Cluster Probability, the default precision of two digits after the decimal point is not sufficient, as many values of this variable may be close to 1 (such as 0.99998) and will therefore be displayed as 1.00 after rounding, so some intervals in marginal rules may look like this: [1.00, 1.00):



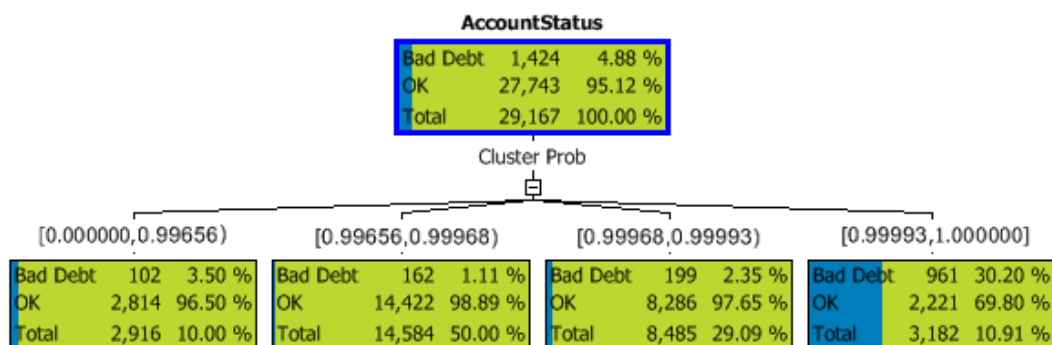
To increase the precision for the *Cluster Prob* variable, open the Tree Attribute Editor by selecting **Tools | Attribute Editor** from the menu, select the row for the Cluster Prob, scroll to the right and enter 0.000000 in the **Format String** field if you want 5 digits after the decimal point. (For details about the ways to specify format strings and precision, see the topic [Specifying Format](#).)

Attribute Editor

| #  | Variable Name                | Max Branches | Open Left                | Open Right               | Cardinality | Format String |
|----|------------------------------|--------------|--------------------------|--------------------------|-------------|---------------|
| 18 | Avg Monthly Balance          | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 1817        |               |
| 19 | Total Amount Purchased       | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 2834        |               |
| 20 | Total Service Charge Revenue | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 516         |               |
| 21 | Profit                       | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 8952        |               |
| 22 | Delinquent Lev 1             | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 10          |               |
| 23 | Delinquent Lev 2             | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 7           |               |
| 24 | Delinquent Lev 3             | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 10          |               |
| 25 | Amount Lost                  | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 1416        |               |
| 26 | age                          | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 83          |               |
| 27 | Residential Status           | 0            |                          |                          | 5           |               |
| 28 | Status                       | 0            |                          |                          | 3           |               |
| 29 | AccountStatus                |              |                          |                          | 2           |               |
| 30 | Cluster Index                | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 5           |               |
| 31 | Cluster Prob                 | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 19960       | 0.000000      |
| 32 | Distance to Center           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 29164       |               |

Copy Paste OK Cancel Help

Upon clicking OK in Attribute Editor, the values of the split variable in the tree will be displayed with requested precision:



If you need to change the default precision of an independent variable in all trees created for this dataset, use Dataset Editor instead. Open the parent dataset and select **Tools | Dataset Editor** from the menu. Select the desired variable and click the **Edit** button. Enter 0.00000 or another preferred format string in the Format field. NOTE: The format changes made in Dataset Editor will be applied to all new trees based on this dataset, but if you want them to



be applied to an existing tree, you need to open the tree and select **Tools | Refresh From Dataset**.

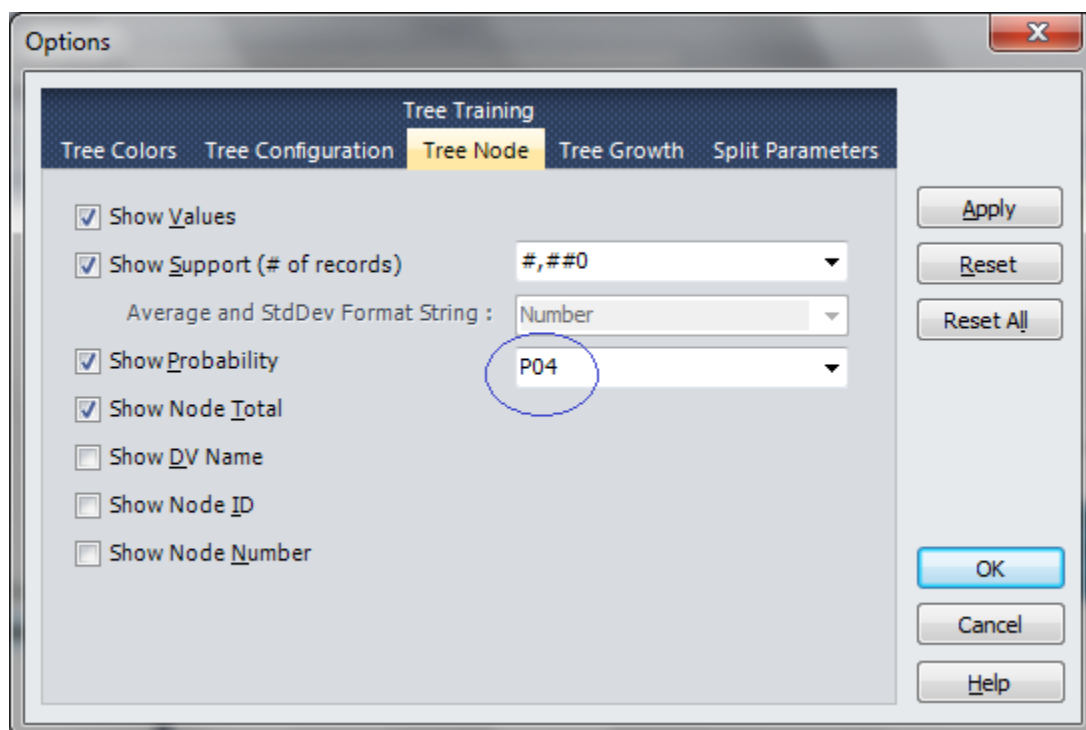
In the above example, if you set the precision of *Cluster Prob* to 0.00000 in Dataset Editor, the values of *Cluster Prob* will be displayed with 5 digits after the decimal point in all new trees based on this dataset and also in the existing trees after you select **Tools | Refresh From Dataset** in the Tree view.

Page | 341

For details about the ways to specify format strings and precision, see the topic [Specifying Format](#).

To change the format and precision of the DV value percentages displayed in the nodes of the current tree, use **Tools | Options** (the Tree Node Settings tab). To change the global default (for all new trees to be created), use **Tools | Preferences** instead.

By default, DV percentages in tree nodes are rounded to two decimal places. For example, if you want the percentages of the values "Bad Debt" and "OK" in the above tree to be displayed with four digits after the decimal point, click the Options button in the toolbar. Type "P04" in the Show Probability box. (For details about the ways to specify format strings and precision, see the topic [Specifying Format](#).)



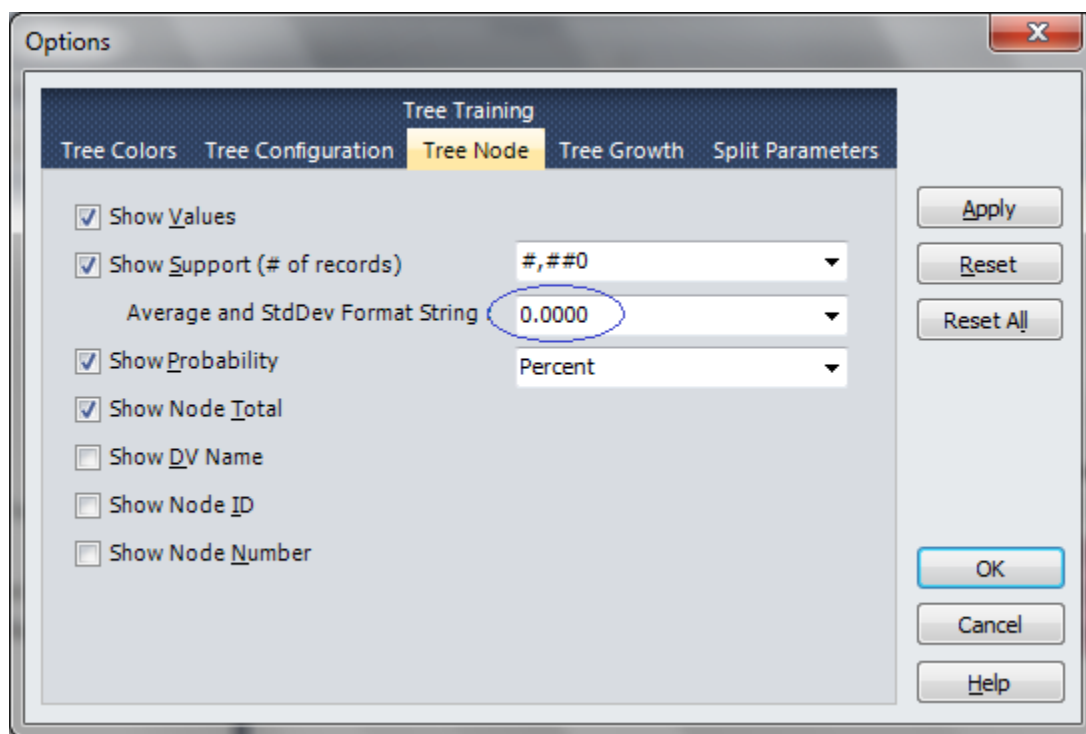
Upon clicking OK, the percentages of the AccountStatus values will be displayed with four digits after the decimal point:

|          |        |          |   |
|----------|--------|----------|---|
| Bad Debt | 1,424  | 4.8822   | % |
| OK       | 27,743 | 95.1178  | % |
| Total    | 29,167 | 100.0000 | % |

If the DV is continuous, the default precision of Average and Standard Deviation is two decimal places. For example:

|         |                 |
|---------|-----------------|
| Average | 8.76            |
| StdDev  | 224.10          |
| Total   | 29,167 100.00 % |

If you need to increase the precision of Average and StdDev to 4 decimal places, enter 0.0000 as Average and StdDev Format String in the **Options** dialog. (For this change to affect all new trees with continuous DVs to be created, use **Tools | Preferences**, Tree Node Settings tab.):



Upon clicking OK, the Average and StdDev for *Profit* will be displayed with four digits after the decimal point:

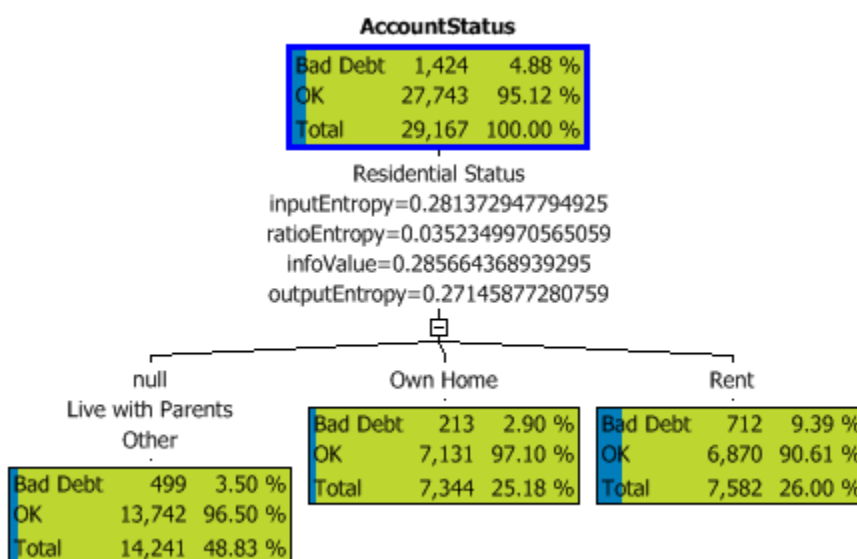
|         |                 |
|---------|-----------------|
| Average | 8.7579          |
| StdDev  | 224.1036        |
| Total   | 29,167 100.00 % |

If your tree is weighted (for details, see the topics [Attribute Editor: About Weights](#) and [Weight](#)), you may need to see the fractional parts of the node totals. By default, the total number of records in a node is displayed as an integer. To increase its precision, for example, to two decimal places, enter 0.00 in the **Show Support (# of records)** field in the Tree Node Settings tab of the **Options** dialog. If you need the thousands separator, enter #,##0.00. See the topic [Specifying Format](#) for information about format strings.

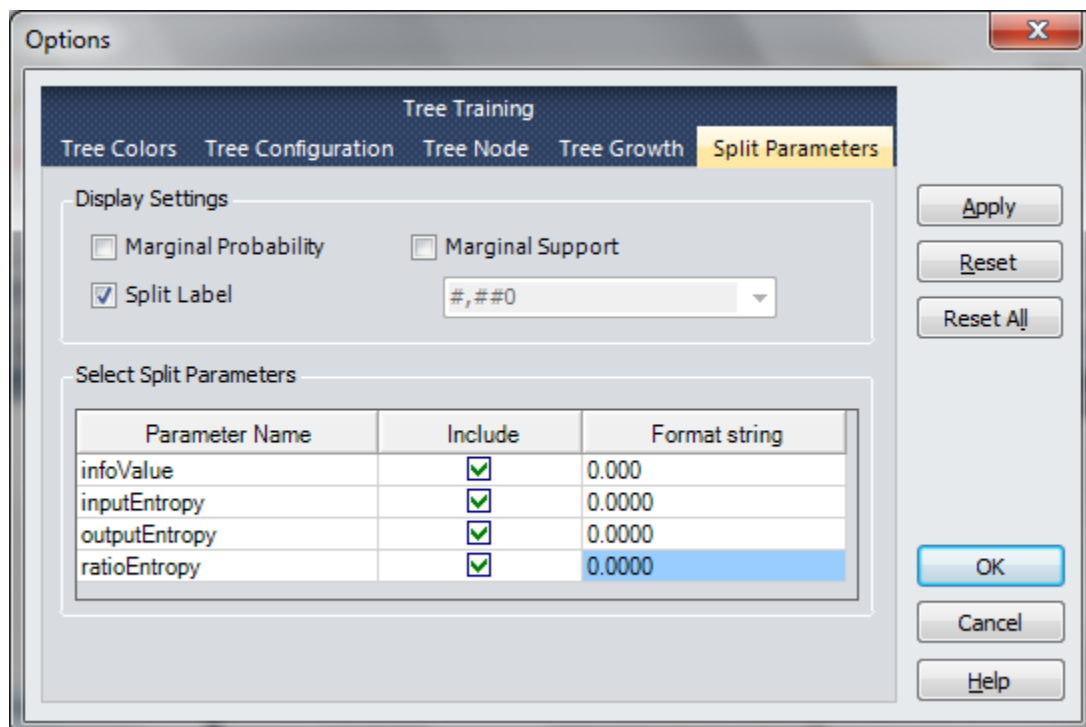
Page | 343

|         |                    |
|---------|--------------------|
| Average | 8.7579             |
| StdDev  | 224.1036           |
| Total   | 29,167.00 100.00 % |

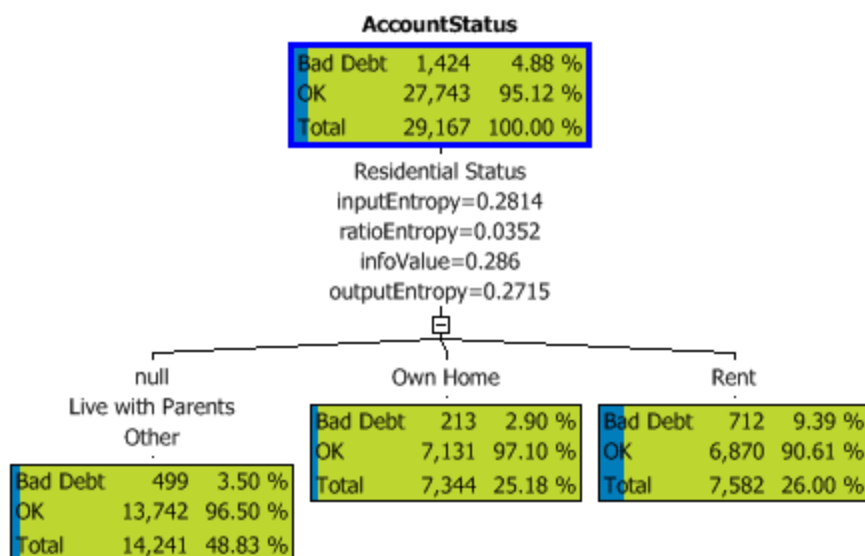
By default, the Tree Split Parameters such as Ratio Entropy are displayed with up to 17 digits after the decimal point. For example:



If you want them to be displayed with lower precision, select the Tree Split Parameters tab of the **Tools | Options** dialog (or the **Tools | Preferences** dialog if you want the change to be applied to all trees to be created in the future) and enter the desired precision in the Format String field. For example, with the following settings, Info Value will be shown with three decimal places, while Input, Output, and Ratio Entropy will be shown with five:



Upon clicking OK, the tree becomes as follows:



## See Also

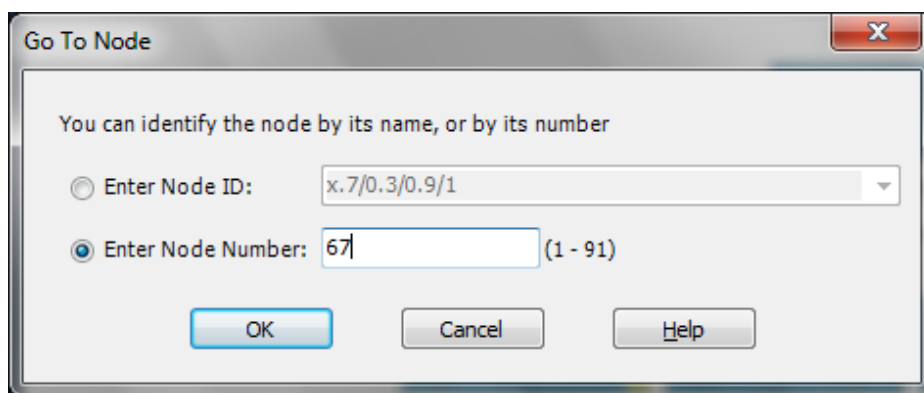
[Tree Map view](#)  
[Node Data view](#)  
[Split Report view](#)  
[Node Report view](#)  
[Profile Chart](#)  
[Find Split](#)  
[Force Split](#)  
[Edit Split](#)

## Go To Node

If you enabled the display of Node Numbers or Node IDs in your tree (in the *Tree Node Settings* tab of the **Options** or **Preferences** dialog), you may need to locate a specific node by its number or ID.

(For example, suppose you scored a tree model and included the Node IDs or Node Number column in the scoring result. Then at some point you need to know which tree node a certain record was assigned to.)

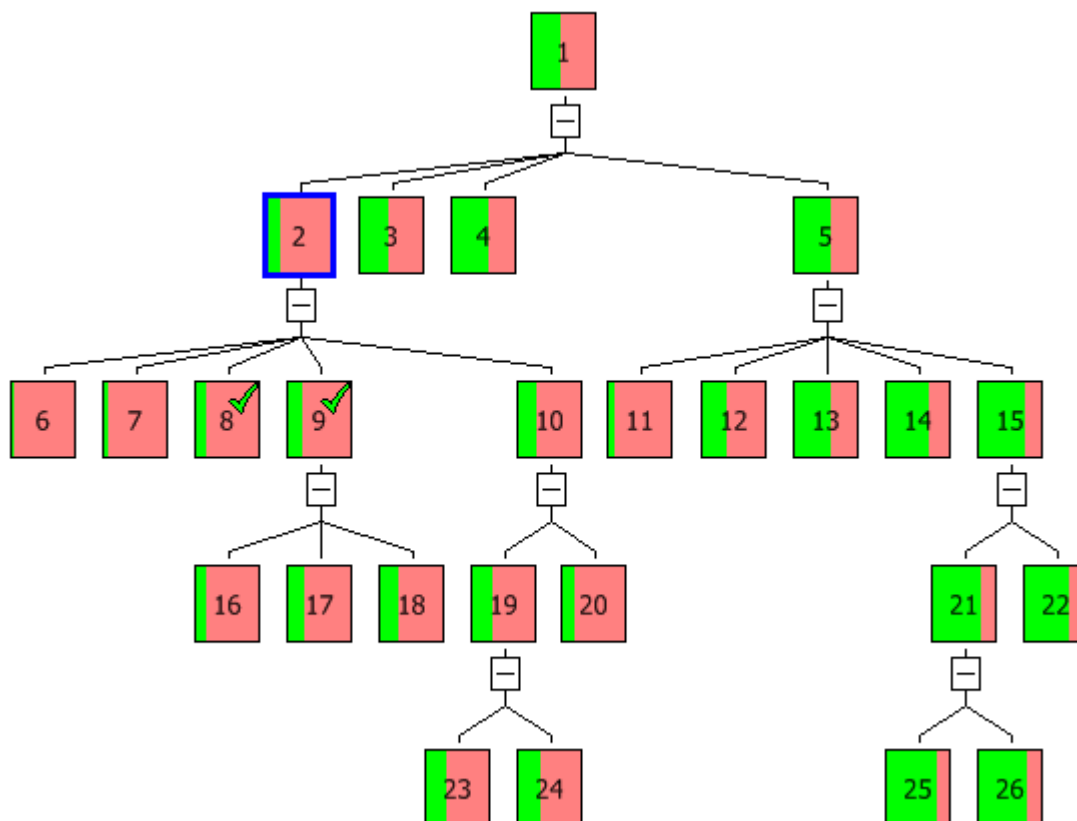
For this purpose, use the **Go To Node** command under the **Edit** menu. The same command is invoked by the keyboard shortcut **Ctrl+G**.



For more on Tree Node Settings and other display options, see the topic [Preferences: Tree Node Settings](#).

## Tree Map

The **Tree Map** view displays the tree without the detailed information for each node. This view is good for navigating through a large tree. You can perform all the functions that are available for the **Tree** view.



## See Also

[Tree view](#)  
[Node Data view](#)  
[Split Report view](#)  
[Node Report view](#)  
[Find Split](#)  
[Force Split](#)  
[Edit Split](#)

## Node Data

The **Node Data** view displays the data contained within the active tree node. The node is considered active when it is highlighted by a mouse click while in the **tree** view or **tree map** view.

|   | age | workclass        | fnlwgt | education    | education-num | marital-status |
|---|-----|------------------|--------|--------------|---------------|----------------|
| ▶ | 34  | Private          |        | 10th         | 6             | Never-married  |
| 2 | 26  | Private          | 82091  | HS-grad      | 9             | Never-married  |
| 3 | 72  | ?                | 132015 | 7th-8th      | 4             | Divorced       |
| 4 | 25  | Private          | 220931 | Bachelors    | 13            | Never-married  |
| 5 | 32  | Self-emp-not-inc | 109282 | Some-college | 10            | Never-married  |
| 6 | 24  | Self-emp-not-inc | 188274 | Bachelors    | 13            | Never-married  |
| 7 | 22  | Private          | 248446 | 5th-6th      | 3             | Never-married  |
| 8 | 17  | Private          | 269430 | 10th         | 6             | Never-married  |

Tree | Tree Map | **Node Data** | Split Report | Node Report | Chart | Profile Chart | Parameters and Attribu...

Data is listed in a worksheet-style window where the fields appear as columns and the records appear as rows. This information cannot be changed.

## See Also

[Tree View](#)  
[Tree Map View](#)  
[Node Data View](#)  
[Split Report View](#)  
[Node Report View](#)  
[About Node dataset](#)

## Split Report

The **Split Report** displays all splits at the currently selected node of the decision tree. Although only the most statistically significant split is displayed in the tree, all statistically significant splits are calculated. The split report shows all the calculated splits in rank order. Variables with a significance less than the filter threshold will be displayed but will not have statistics computed. To adjust the filter threshold, select the menu command **Tools | Options** when Split Report is displayed.

|  Variable Name | Rank | Info     | Input Entropy | Output Entropy | Ratio Entropy | Information Value |
|-------------------------------------------------------------------------------------------------|------|----------|---------------|----------------|---------------|-------------------|
| relationship                                                                                    | 1    | 21.00091 | 0.78871       | 0.62307        | 0.21001       | 1.51571           |
| marital-status                                                                                  | 2    | 20.02513 | 0.78871       | 0.63077        | 0.20025       | 1.36872           |
| ▶ age                                                                                           | 3    | 11.5691  | 0.78871       | 0.69746        | 0.11569       | 1.07773           |
| capital-gain                                                                                    | 4    | 11.5587  | 0.78871       | 0.69754        | 0.11559       |                   |
| occupation                                                                                      | 5    | 11.4599  | 0.78871       | 0.69832        | 0.1146        | 0.75986           |
| education-num                                                                                   | 6    | 11.28876 | 0.78871       | 0.69967        | 0.11289       | 0.71556           |
| education                                                                                       | 7    | 11.28876 | 0.78871       | 0.69967        | 0.11289       | 0.71556           |
| hours-per-week                                                                                  | 8    | 7.12078  | 0.78871       | 0.73255        | 0.07121       | 0.46187           |
| sex                                                                                             | 9    | 4.52947  | 0.78871       | 0.75298        | 0.04529       | 0.29493           |
| capital-loss                                                                                    | 10   | 3.08356  | 0.78871       | 0.76439        | 0.03084       | 0.21253           |
| workclass                                                                                       | 11   | 3.04825  | 0.78871       | 0.76467        | 0.03048       | 0.18992           |
| native-country                                                                                  | 12   | 1.06576  | 0.78871       | 0.7803         | 0.01066       | 0.07839           |
| fnlwt                                                                                           | 13   | 0.02743  | 0.78871       | 0.78849        | 0.00027       | 0.00168           |
| Income                                                                                          |      |          |               |                |               |                   |

[Tree](#) | [Tree Map](#) | [Node Data](#) | [Split Report](#) | [Node Report](#) | [Chart](#) | [Profile Chart](#) | [Parameters and Attribu...](#)

As you navigate the nodes in the other tree views, the split report is automatically updated.

The **split report** also displays statistics for each split. The type of statistics that are displayed in this report is dependent on the [measure](#) used when creating the decision tree.

The splits in Split Report are ranked by significance, which is expressed by the *Info* parameter.

## See Also

[Tree view](#)

[Tree Map view](#)

[Node Data view](#)

[Node Report view](#)

## Node Report

The Node Report view provides you with information about the terminal nodes in the tree.



Dependent Variable:  Value to Chart:  ☐ Tagged nodes only

|                                                                                                                | Size of Group | Cumulative Size | Number of Status = Bad Debt | Cumulative Status = Bad Debt |
|----------------------------------------------------------------------------------------------------------------|---------------|-----------------|-----------------------------|------------------------------|
| (([Residential Status] = 'Rent') AND ([Region] = 'Four'))                                                      | 454           | 454             | 138                         | 138                          |
| (([Residential Status] = 'Rent') AND ([Region] IN ('Five', 'One', 'Seven', 'Three', 'Twelve'))                 | 979           | 1433            | 116                         | 254                          |
| ▶ (([Residential Status] = 'Rent') AND ([Region] IN ('Eight', 'Eleven', 'Nine', 'Other', 'Six', 'Ten', 'Two')) | 6149          | 7582            | 458                         | 712                          |
| (([Residential Status] is null OR [Residential Status] IN ('Live with Parents', 'Other'))                      | 14241         | 21823           | 499                         | 1211                         |

Tree | Tree Map | Node Data | Split Report | **Node Report** | Chart | Profile Chart | Parameters and Attributes

**Node Rules**

The rule for the terminal node in the decision tree.

**Size of group**

The number of observations in a given node.

**Cumulative Size**

The cumulative number of observations up to and including this node.

**Number of <DV> = <DV Value>**

The number of observations of this category of the dependent variable in this node.

**Cumulative <DV> = <DV Value>**

The cumulative number of observations of this category up to and including this node

Dependent Variable:  Value to Chart:  ☐ Tagged nodes only

|                                                                                                                | Status = Bad Debt | Status = Bad Debt rate (%) | Cumulative Status = Bad Debt rate (%) | Lift (%) |
|----------------------------------------------------------------------------------------------------------------|-------------------|----------------------------|---------------------------------------|----------|
| (([Residential Status] = 'Rent') AND ([Region] = 'Four'))                                                      | 38                | 30                         | 30                                    | 622.59   |
| (([Residential Status] = 'Rent') AND ([Region] IN ('Five', 'One', 'Seven', 'Three', 'Twelve'))                 | 54                | 12                         | 18                                    | 363.05   |
| ▶ (([Residential Status] = 'Rent') AND ([Region] IN ('Eight', 'Eleven', 'Nine', 'Other', 'Six', 'Ten', 'Two')) | 12                | 7                          | 9                                     | 192.34   |
| (([Residential Status] is null OR [Residential Status] IN ('Live with Parents', 'Other'))                      | 11                | 4                          | 6                                     | 113.66   |

Tree | Tree Map | Node Data | Split Report | **Node Report** | Chart | Profile Chart | Parameters and Attributes

**<DV> = <DV Value> Rate (%)**

Ratio of the number of observations of the dependent variable category (Number of <DV>=<DV Value>) over the total number of observations in the node (Size of group).

**Cumulative <DV> = <DV Value> Rate (%)**

Ratio of the cumulative number of observations (cumulative <DV>=<DV Value>) over the cumulative number of observations up to and including this node (cumulative size).

**Lift**

The cumulative ratio of selected categories moving top-to-bottom, from node to node. It is computed relative to the overall percentage that the selected category represents in the entire dataset

Node Report also includes an option to display node rules with a column per Independent Variable (see Node Report Options below). In this mode, each independent variable occurring in at least one node rule is represented by a column in the node report. For any given rule (row), if a condition for the variable occurs in the rule, this condition (in the form of a value, a set of values, or an interval) is displayed in the column representing this variable.

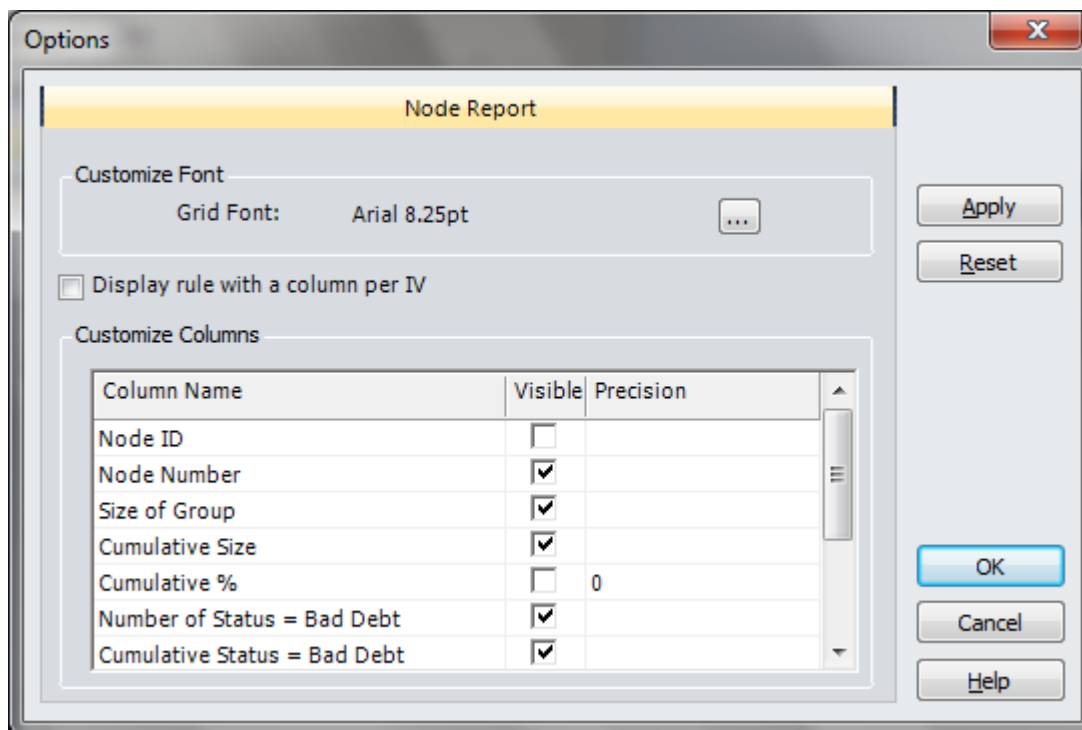
Dependent Variable: Status    Value to Chart: Bad Debt    ☐ Tagged nodes only

|   | [Residential Status]               | [Region]                                              | Size of Group | Cumulative Size | Number of Status = Bad Debt | Cumulative |
|---|------------------------------------|-------------------------------------------------------|---------------|-----------------|-----------------------------|------------|
| ▶ | Rent                               | Eight<br>Eleven<br>Nine<br>Other<br>Six<br>Ten<br>Two | 6149          | 7582            | 458                         |            |
|   | null<br>Live with Parents<br>Other |                                                       | 14241         | 21823           | 499                         |            |
|   | Own Home                           |                                                       | 7344          | 29167           | 213                         |            |

Tree | Tree Map | Node Data | Split Report | Node Report | Chart | Profile Chart | Parameters and Attributes

## Node Report Options

To change the Node Report display options, click the  **Options** icon in the toolbar or select **Tools | Options** from the menu. You can select columns to display or hide, specify precision for columns, change the font, and choose to display rules with a column per IV:



## See Also

[Preferences: Node Report](#)

[Tree view](#)

[Tree Map view](#)

[Node Data view](#)

[Split Report view](#)

## Node Chart

The Node Chart (the **Chart** tab of the Tree object) helps you to visualize the distributions of variables within a tree node.

The chart look and functionality is exactly the same as in the [Dataset Chart](#) (for the Single Variable option) or Cross Tabs (for the Cross Tab option), except that it shows the variable distributions in a single tree node rather than the whole dataset. Refer to the [Dataset Chart](#) topic for details.

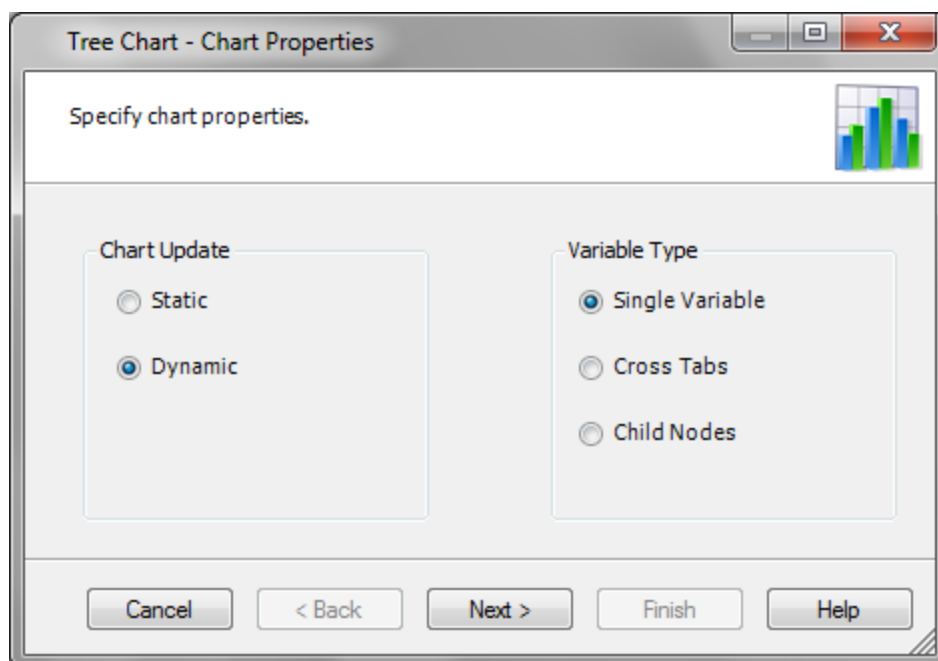
One of the most convenient features of the tree chart is that it can be dynamic: you can traverse the tree clicking on different nodes and at the same time view the chart for the currently selected node.

Note that you can view the tree and the chart side-by-side by docking the Chart view next to the Tree view. To this end, click on the Chart tab name and drag the tab to the desired position on the screen.

The first time you click the **Chart** tab in a tree, the **Add Chart** wizard appears automatically. If it does not start automatically, then:

- Click the button  **Add** in the toolbar, or
- Select **Tools | Add Chart** from the menu.

In the first page of the Add Chart Wizard, choose the Chart Properties.



With regard to the update type, the chart can be Static or Dynamic.

If you choose the **Dynamic** option (default), the Chart tab will be dynamically updated as you traverse the tree selecting different nodes in a manner completely synchronized with the Tree view. At any instant, it will display the chart for the node selected at that instant in the Tree view. To view the tree and the node chart side-by-side, dock the Chart view next to the Tree view. To this end, click on the Chart tab name and drag the tab to the desired position on the screen.

If you choose the **Static** option, the Chart tab will always display the chart for the node selected in the Tree view at the time of adding the chart. It will not be updated as you traverse the tree selecting different nodes, and will always show the chart for one fixed node.

**Single Variable:** this option means that the chart will be for a single variable (same type as the [Dataset Chart](#) object).

**Cross Tabs:** The chart will be of the same type as the [Cross Tabs](#) chart of the Dataset object.

**Child Nodes:** Each child of the selected node will be represented by a bar.

Page | 353

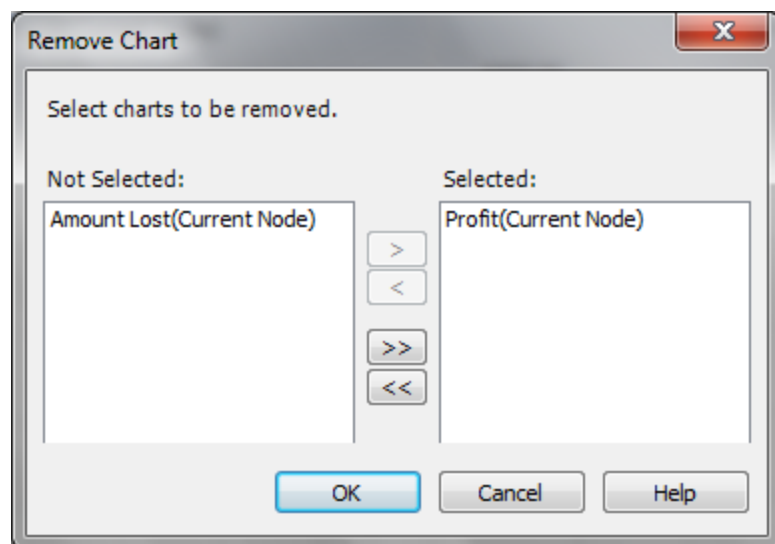
If the DV is discrete, the height of each bar represents the size of the corresponding child node (number of records), and the proportion of each DV value is shown in a specific color.

If the DV is continuous, the height of each bar represents the average DV value in the corresponding child node.

If you selected the Single Variable option, see the [Dataset Chart](#) topic in the Data Profiling section.

If you selected the Cross Tabs option, see the [Cross Tabs](#) topic in the Data Profiling section.

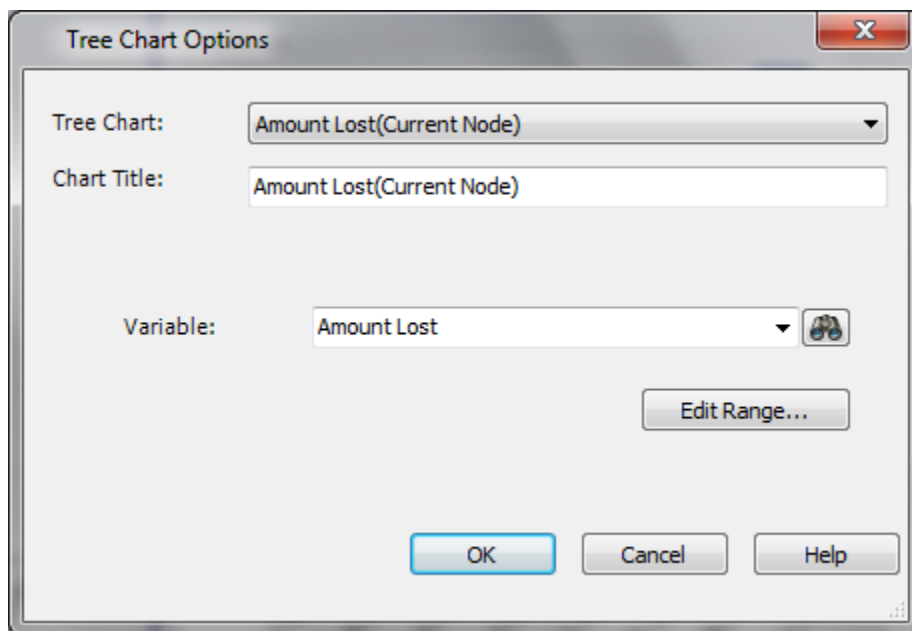
To remove a chart you added before, click the **Remove** button in the toolbar (or select **Tools | Remove Chart** from the menu). Double-click on the name of the chart to remove and click OK:



To change the charted variable or the ranges (bins) in a Node Chart, click the button in the toolbar or select **Tools | Options** from the menu.



In the Tree Chart Options dialog, select the desired chart from the Tree Chart list.



To change the variable for the selected chart, choose the desired variable from the drop-down list.

To change the binning of the selected variable, click the **Edit Range** button (or **Edit Grouping** button, if the variable is discrete).

## Profile Chart

The profile chart shows the distribution of the dependent variable (DV) **in the terminal nodes** of the tree in the form of a bar chart. Each bar represents a terminal node in the tree, its width representing the size of the node (i.e., the number of records in the node).

Profile Chart is completely interactive and synchronized with the Decision Tree view, so you can click on the chart to navigate the tree, right-click on any node to choose a command from the same pop-up menu as in the Decision Tree view, so you can perform any split operation such as Find Split. If you tag a node in Profile Chart, it will be automatically tagged in the Decision Tree view, and vice versa.

### Discrete DV

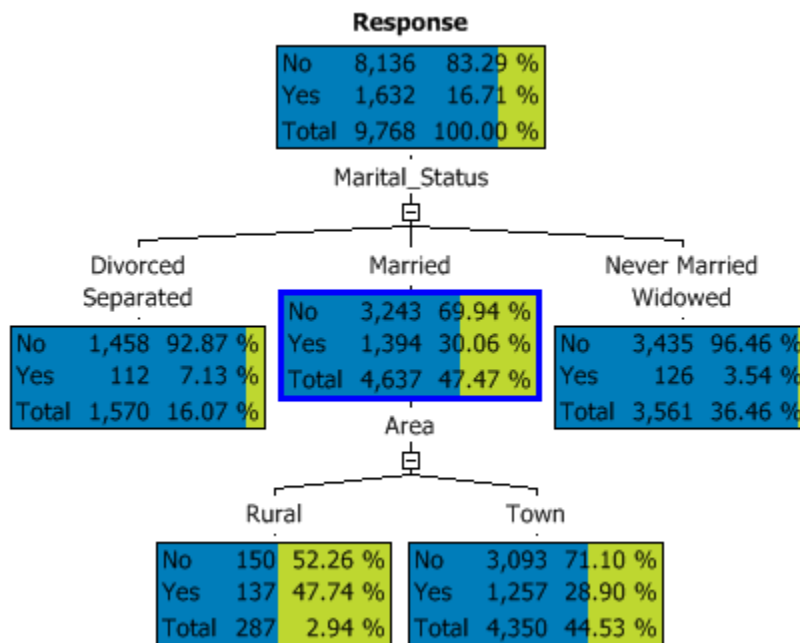
In the case of a discrete DV, you must specify the DV outcome of your interest using the drop-down menu at the top of the Profile Chart view.

The height of each bar is the probability of the specified DV outcome for the corresponding node (equivalent to the percentage shown for this outcome in the Tree view). The width represents the size of the node relative to other terminal nodes.

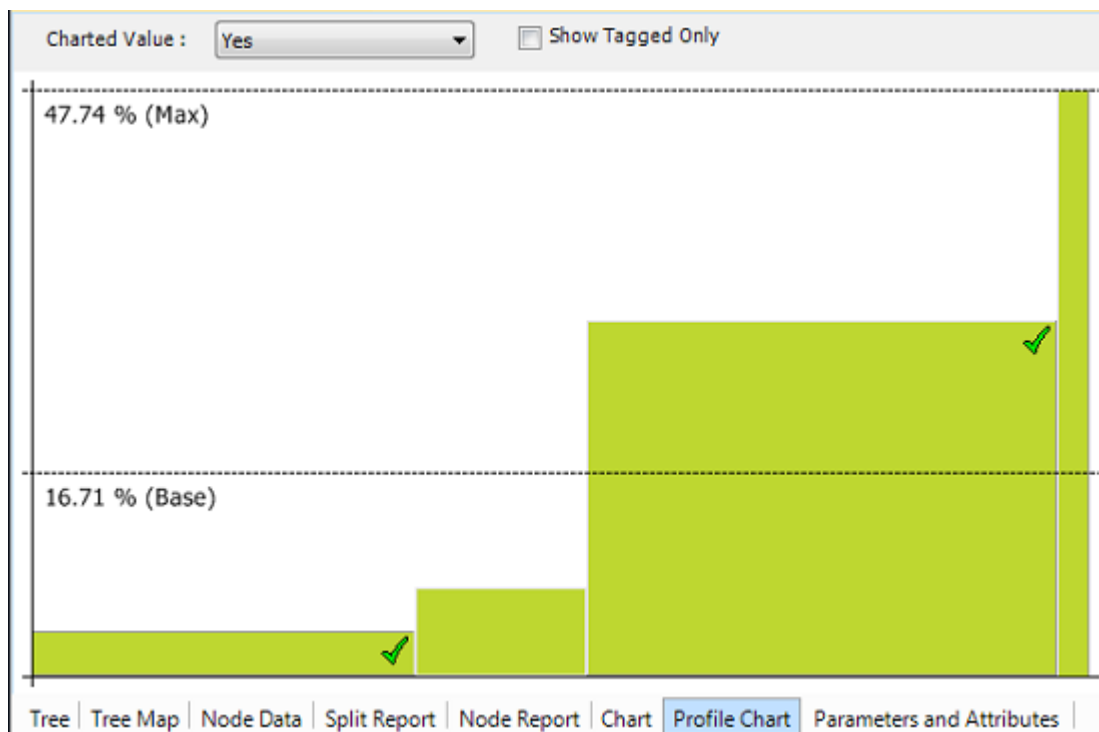
The base line represents the probability of the specified DV outcome in the whole dataset (the root node of the tree). The Max line represents the maximum probability of the specified DV outcome over all terminal nodes.

The nodes are sorted by the probability of the outcome. To the left of the screen are the nodes with the lowest probability of the specified outcome. To the right of the screen are the nodes with higher probability of the specified outcome.

For example, consider the following tree:



The Profile Chart for this tree is shown below (the charted DV value is 'Yes'). Note that two leaves are tagged in the tree view and Profile Chart shows these tags. You can choose to show only the tagged nodes by selecting the checkbox 'Show Tagged Only'. The base line (0.17) is the rounded probability of the 'Yes' outcome in the whole dataset (the root node). The Max line (0.48) represents the maximum probability of the 'Yes' outcome over all terminal nodes.



## Continuous DV

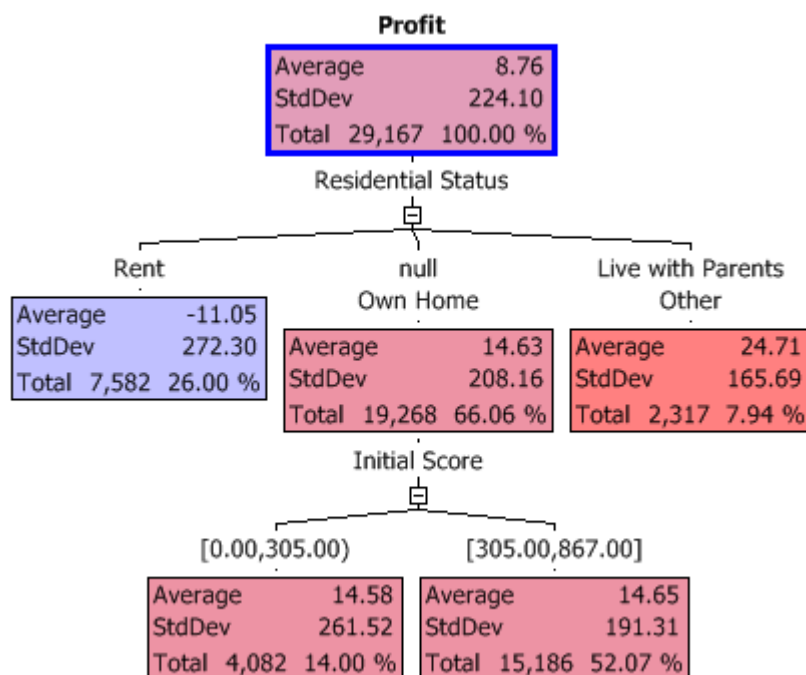
In the case of a continuous DV, the center line of each bar represents the average DV value in the corresponding node, the top and bottom of the box are average plus / minus one standard deviation in the node. As in the discrete case, the width of each bar represents the size of the corresponding node.

The center line of the chart represents the baseline, which is the average DV value in the whole dataset (the root node).

The Min and Max lines are the minimum of ( Avg - StdDev) and the maximum of ( Avg + StdDev), respectively, taken over all terminal nodes.

For example, consider the following tree with continuous DV called Profit.:





The Profile Chart for this tree is shown below. The average value of the DV is the only available charted value. Note that two leaves are tagged in the tree view and Profile Chart shows these tags. You can choose to show only the tagged nodes by selecting the checkbox 'Show Tagged Only'.

The base line (8.76) represents the average DV value in the root node of the tree. The Max line (287.83) and the Min line (-391.65) in this case correspond to (Avg +/- StdDev) for the node with the rule ([Residential Status] IN ('Live with Parents', 'Rent')) AND (0 <= [Initial Score] <305), which is represented by the leftmost bar.



## Parameters and Attributes

The **Parameters and Attributes** tab displays a complete summary of the tree configuration:

- the settings you specified when inserting the tree in the Tree Wizard (also available under **Tools | Options** in the Tree view)
- the tree parameters from the [Tree Attribute Editor](#)
- expressions for calculated columns (if any are used in the tree)
- general tree statistics (the tree depth, the total number of nodes, the number of terminal nodes)

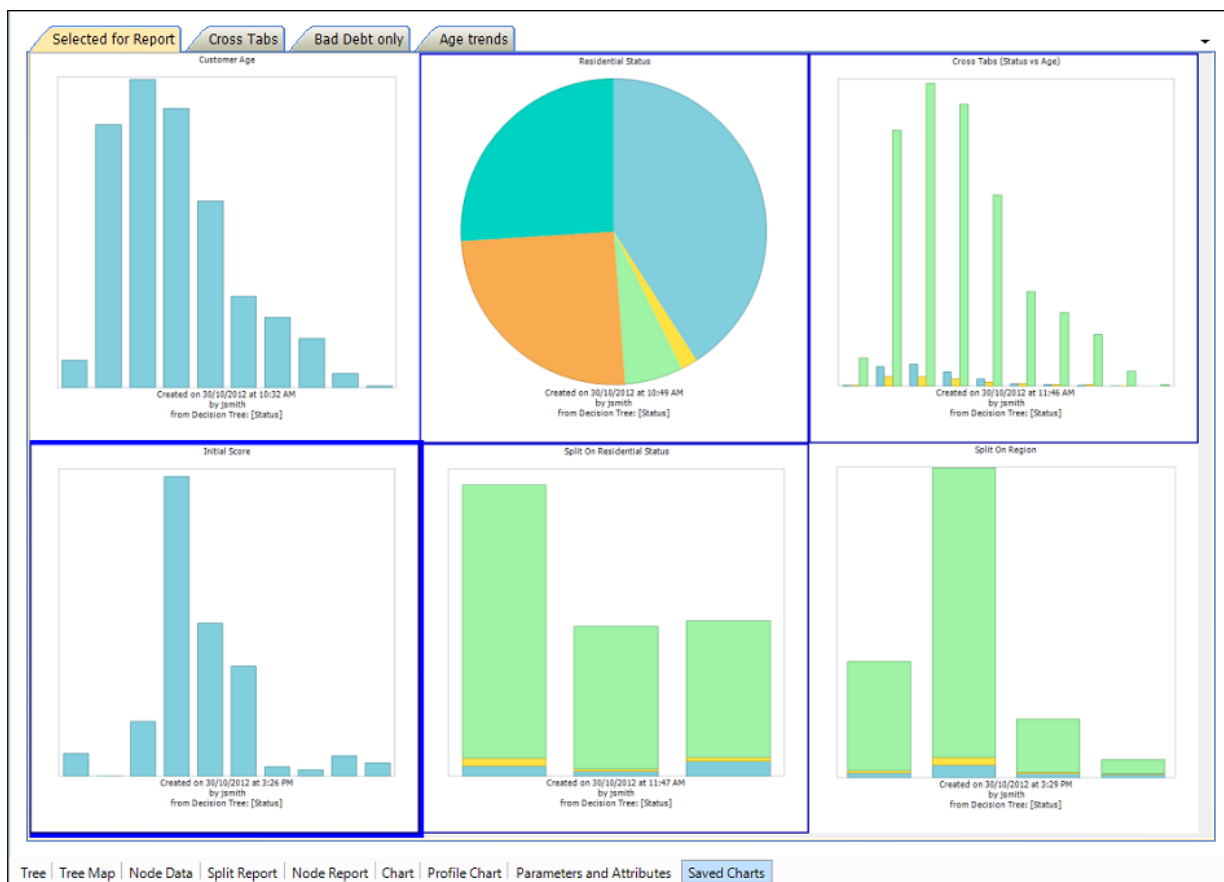
You can specify the precision for the numbers displayed in the Parameters and Attributes tab by selecting **Tools | Options** from the menu and choosing the desired notation (standard or scientific) and precision.

The Saved Charts tab is available for datasets, decision trees, and strategy trees. This tab represents the library of charts you saved in other views of the object it belongs to.

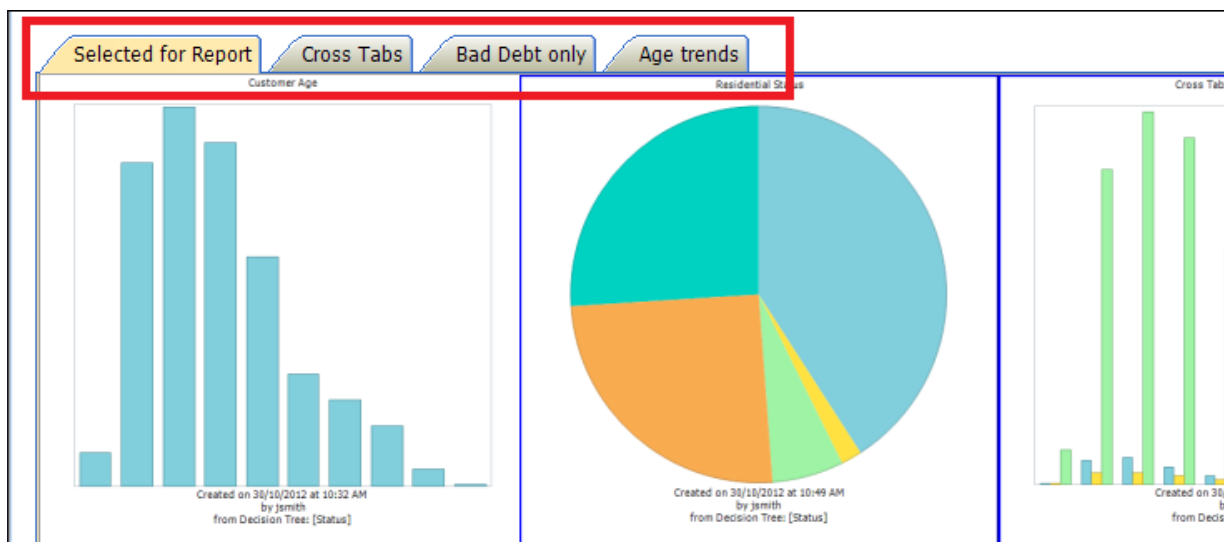
- The Saved Charts tab of a dataset contains charts saved in the Dataset Chart, Segment Viewer, Cross Tabs, and Correlations views of the dataset.
- The Saved Charts tab of a tree contains node charts saved in the Chart view of the tree.

Page | 359

In the Saved Charts tab you can view, organize, and print the charts, or copy them to Microsoft Office documents for reporting.



Charts are organized in groups. Each group is represented by a separate tab at the top of the chart grid. The tabs are highlighted by the red frame in the example screenshot below. Click on the tab names to switch between the chart groups.

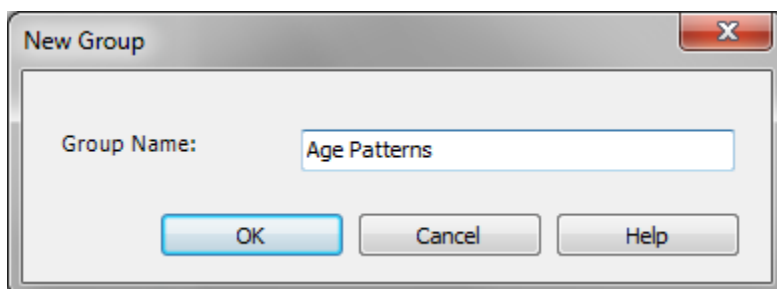


To view any chart in detail in a separate window, double-click on it and resize its window as necessary. In this window you can view the chart details, copy or print it, etc. For more information, see the topic [Single Chart View](#).

## Chart Groups

You can create new chart groups and distribute the charts in these groups.

To create a new chart group, select the **New Group** command from the **Groups** menu, or right-click on the group tab name and select **New** from the context menu. Then specify the desired name in the dialog:



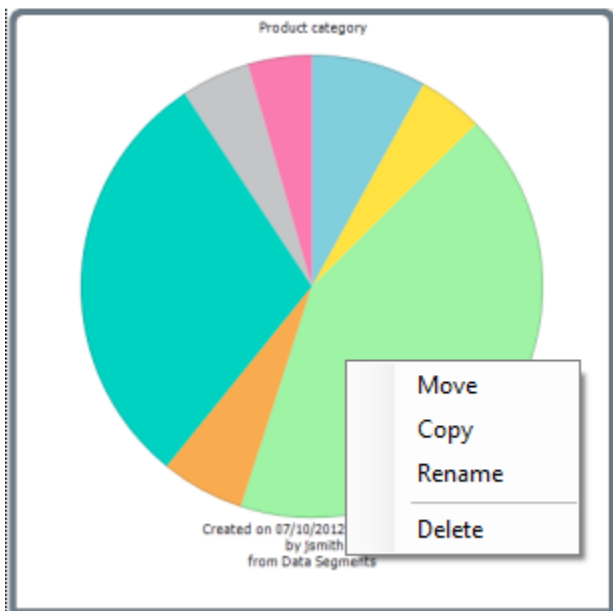
A new blank chart group with the new name will be created.

To rename an existing chart group, right-click on the group tab name and select the **Rename** command from the context menu. Enter the new name and click OK.

## Operations with Charts

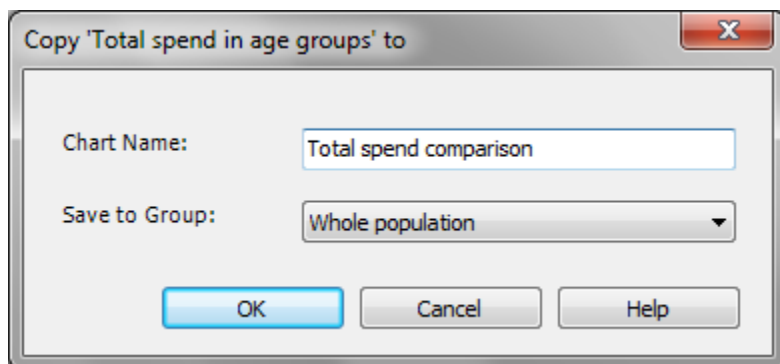
To change the position of any chart within its group, drag and drop it to the desired position.

To delete, copy, rename, or move a chart to a different group, right-click on the chart and select the appropriate command from the context menu.



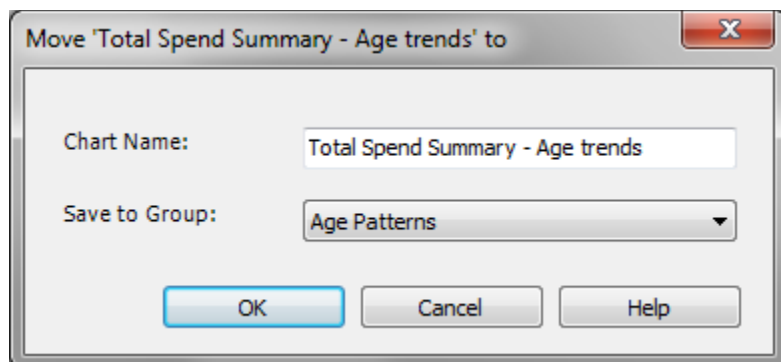
Page | 361

- To delete a chart, select **Delete** from the context menu.
- To rename a chart, select **Rename** from the context menu, specify the new name for the chart, and click OK.
- Charts can be copied within the same chart group or to a different chart group. To copy a chart, right-click on it and select **Copy** from the context menu. Specify the name for the new chart and select the destination group. The current group is the default destination.



- There are two ways to move a chart to another group:
  - a) drag and drop the chart to another group tab, or
  - b) right-click on the chart and select **Move** from the context menu.

In the Move dialog, specify the new name for the chart and select the desired destination group from the Save to Group list



### See also:

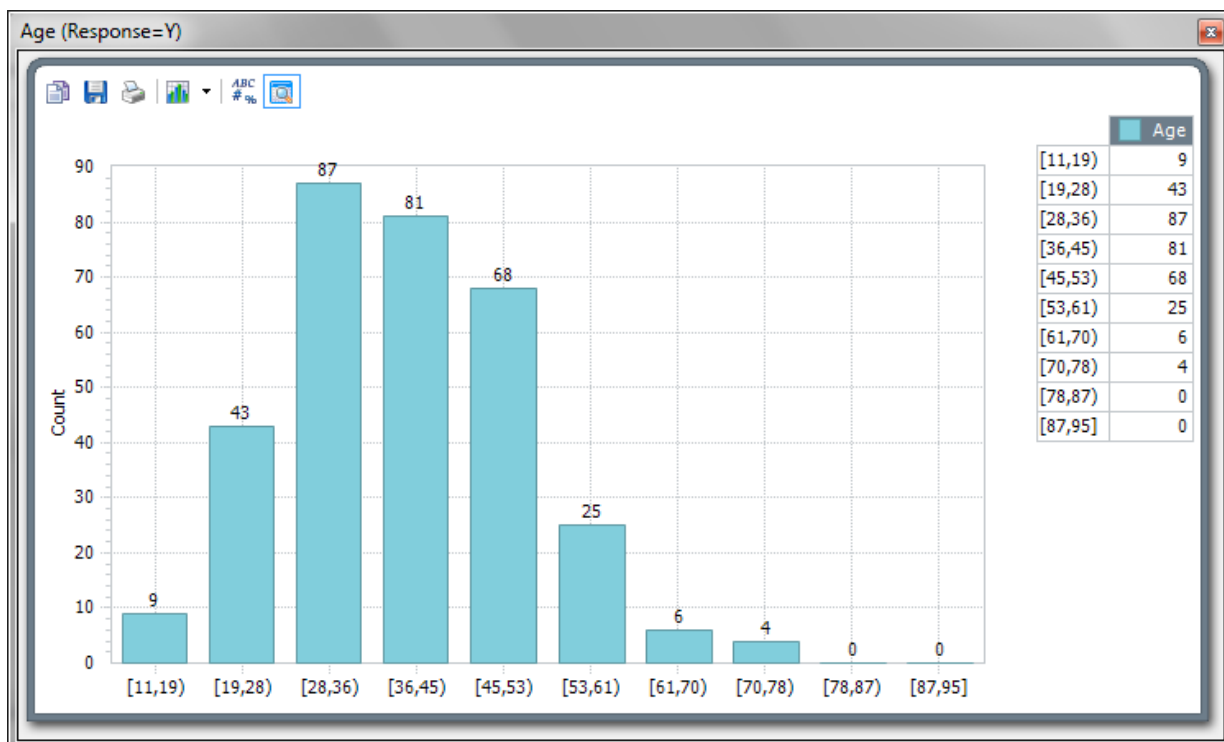
[Single Chart View](#)

## Single Chart View

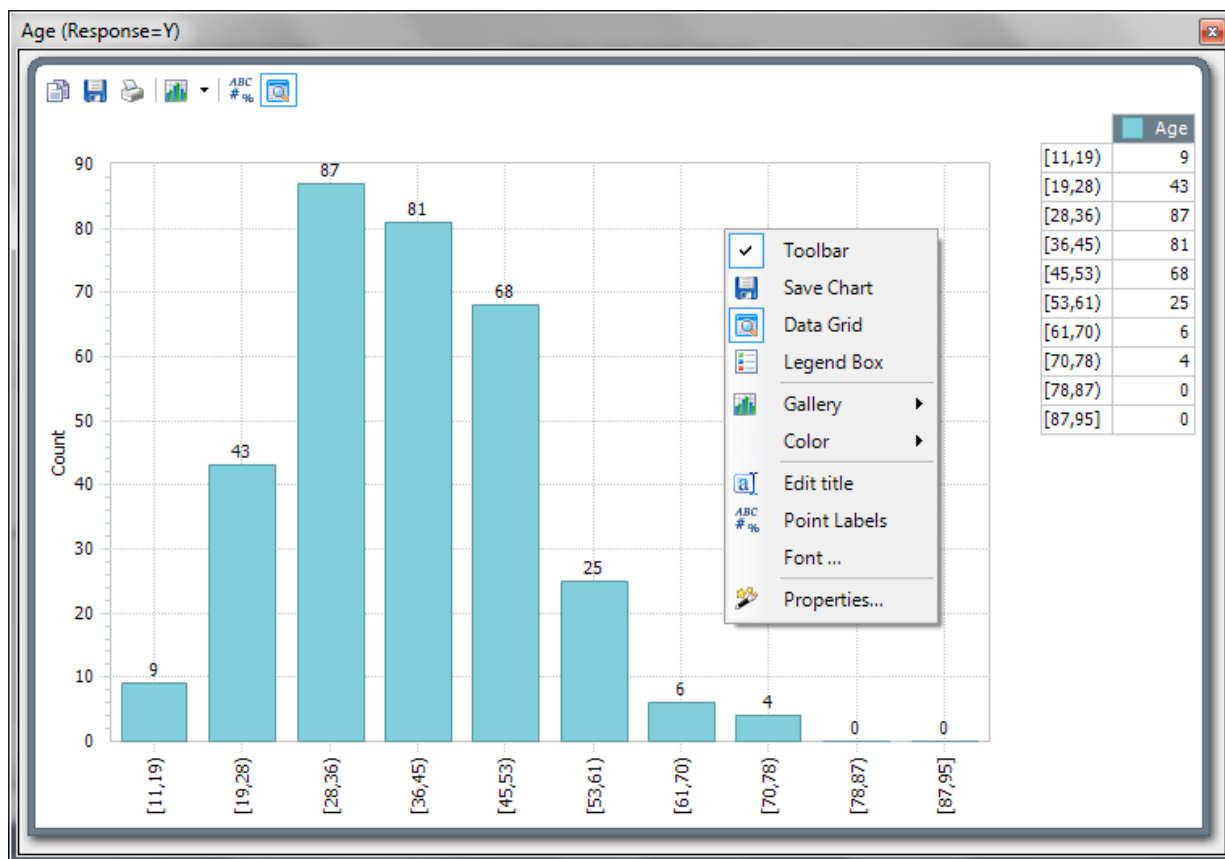
Several views in the application contain multiple charts: [Segment Viewer](#) in datasets, [Chart](#) in trees, and [Saved Charts](#) in both datasets and trees. In any of these views, you can open each individual chart in its own separate window, where you can view its details, copy or print it, etc.

Double-click on the desired chart to view it in a separate window. The window provides a detailed view of the chart and shows the data grid representing the chart data. You can resize the window by clicking and dragging on its corners or borders.

Here is an example of a chart from Segment Viewer opened in its own window.

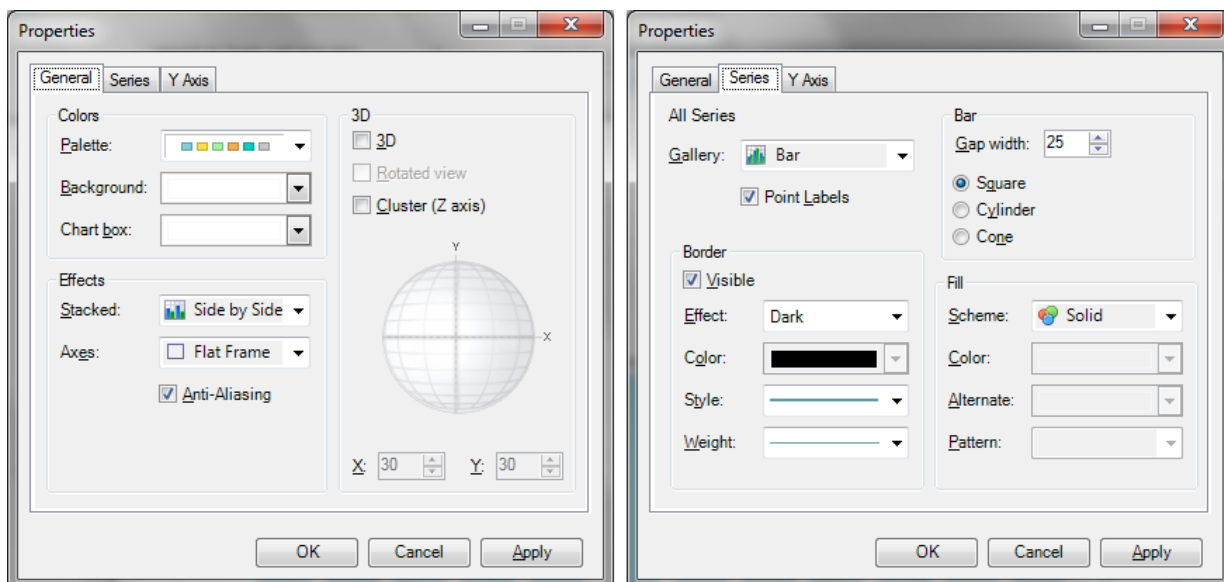


To change the display properties of the chart, use the chart toolbar or right-click anywhere on the chart background. The following context menu appears when right-clicking on the chart background. For example, select Point Labels to show or hide the values on top of the bars; select Data Grid to show or hide the table with the chart data.

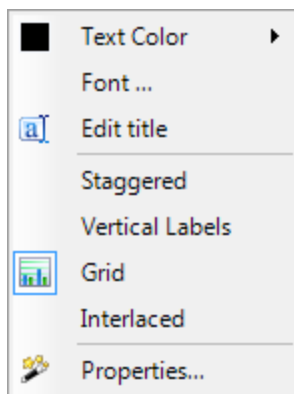


To change advanced properties of the chart, select **Properties** from the context menu. In the **Properties** dialog, you can choose the color palette, 3D or 2D view, effects, properties of the axis labels, show or hide grid lines, etc.

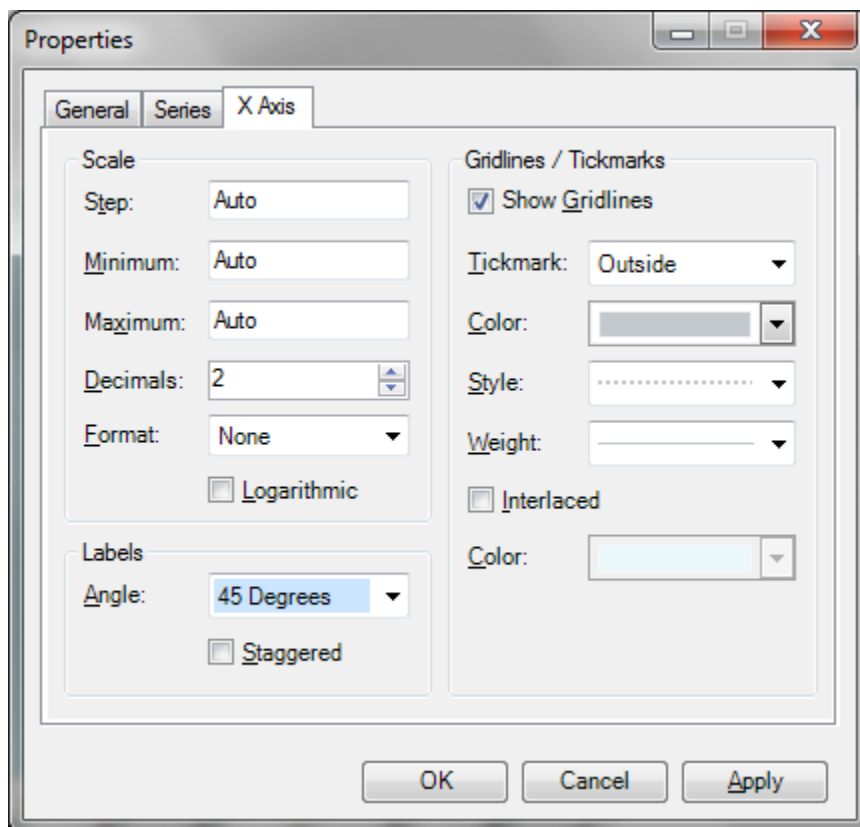




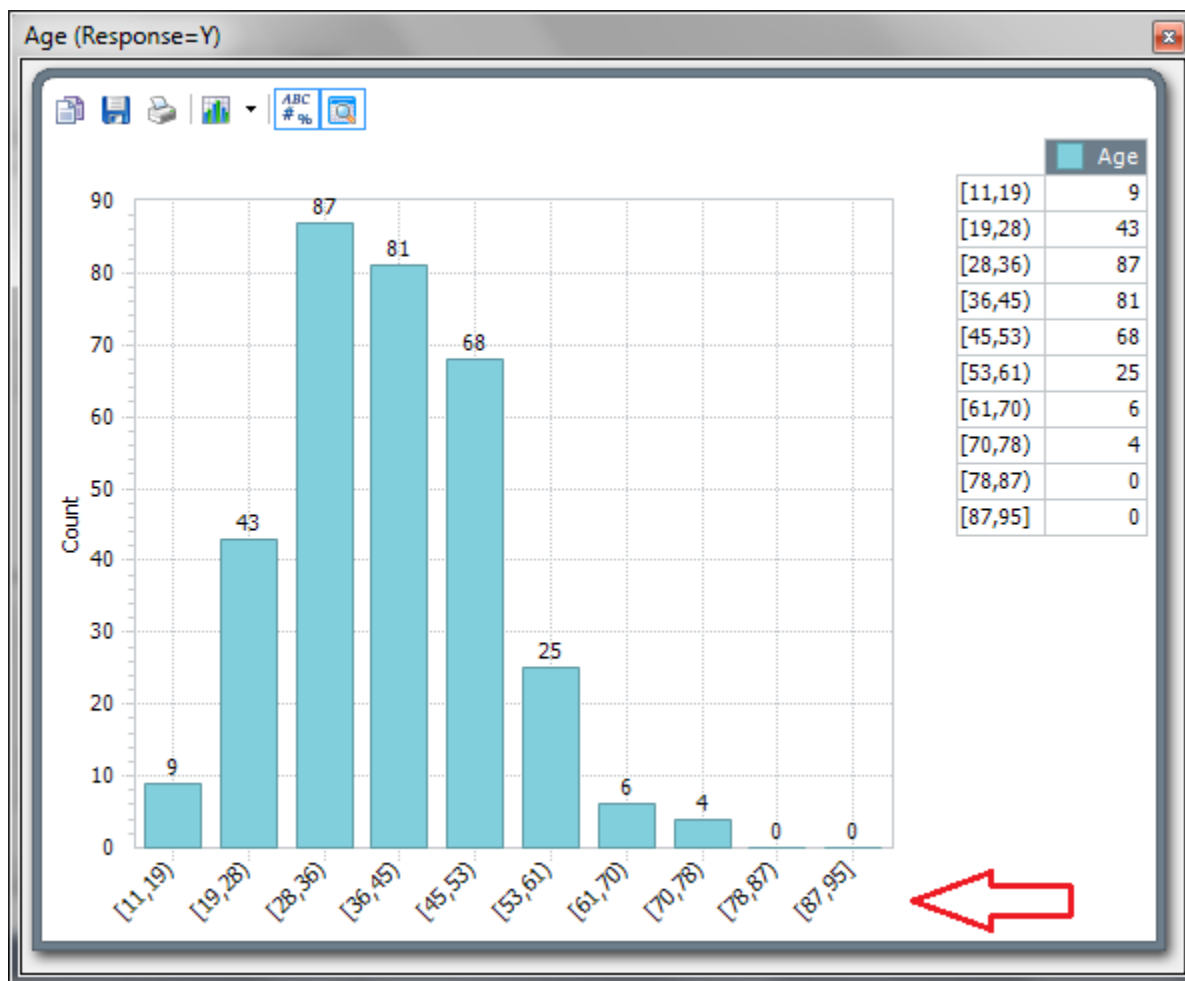
To change the orientation of the axis labels, right-click on any label of the desired axis (vertical or horizontal). The context menu will look like this:



Select *Staggered* or *Vertical Labels* if desired. If you want to place the labels at a 45-degree angle, select *Properties* and set *Angle* to 45 Degrees.



As a result, the X-axis labels are displayed as follows:



## Chart Toolbar

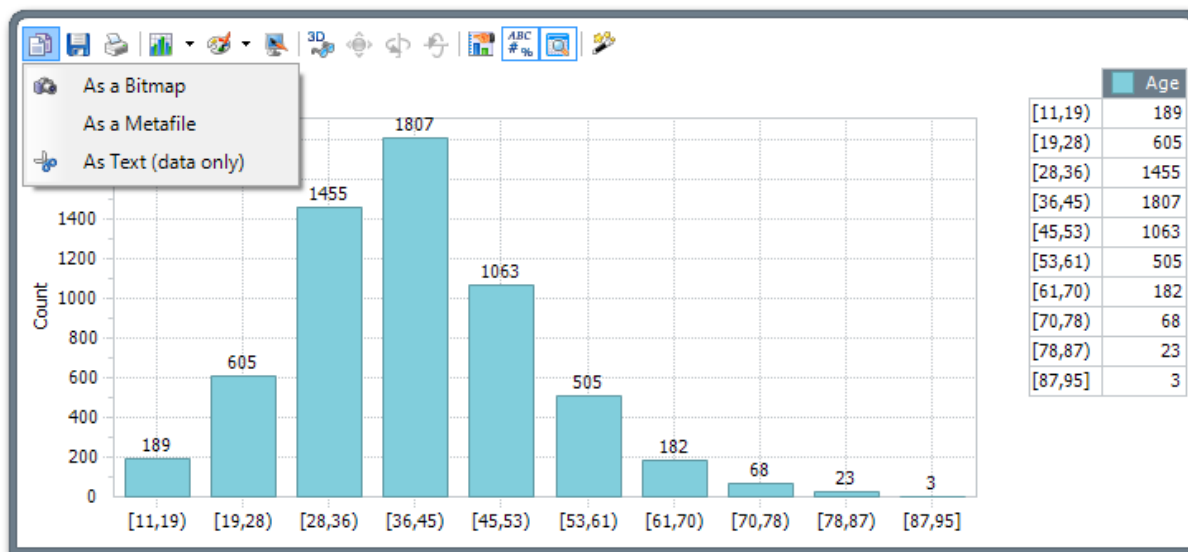
Depending on the charts' parent view - Segment Viewer, Cross Tabs, Chart (in trees), etc., - its toolbar at the top left corner of the chart window can have one of the following two forms:



All icons in the Chart toolbar except the *Save*, *Copy*, and *Print* icons are shortcuts to the corresponding options and commands in the Properties dialog described above.

To copy a chart, click on the *Copy* icon in its toolbar, as shown in the screenshot below. Select *As a Bitmap* or *As a Metafile* from the context menu if you want to copy the chart as an image. Select

As *Text (data only)* if you want to copy the underlying data, which is displayed in the data grid on the right-hand side. Then paste the copied image or text from the clipboard into the desired application.



If you are viewing a chart from the Segment Viewer or Node Chart, you can save it in the Chart Library by clicking on the Save button .

**See also:** [Saved Charts](#)

## Copying Splits and Trees

The **Copy/Paste Splits** commands allow you to copy the splits from one tree and paste them to another. This feature can be useful in a variety of situations.

- See how the split in one node behaves when copied to another node of the same tree
- See how the split in one tree behaves when copied to the node of a different tree created in a different dataset.

## Copy Splits

The **Copy Splits** command copies into memory all the splits of the tree, starting from the active node.

**Highlight the node you want to copy**

**From the menu:**

Edit | Copy Split

## From the popup menu:

Right click on parent node | Copy Split

Page | 369

## Paste Splits

The **Paste Splits** command pastes splits that were copied to the clipboard with the **Copy Splits** command. The tree structure from the original Copy Split is maintained.

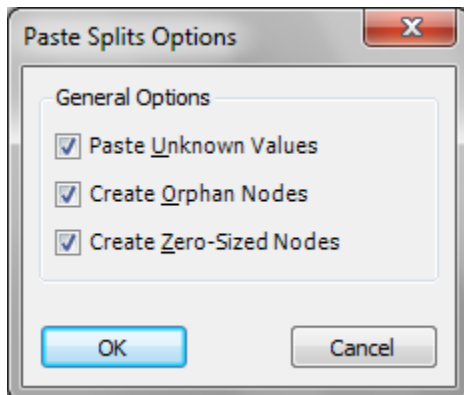
Highlight the node where you want to paste the split.

## From the menu:

Edit | Paste Split

## From the popup menu:

Right-click on the parent node  
Select the Paste Split command



Choose the desired Paste Split options  
Click OK to Paste the splits

## Paste Split Options

## Paste Unknown Values

This option applies to the independent variable (IV) value labels (predicate values of marginal rules) shown above each child node in splits. If some IV values in the source split are not present in the destination parent node (i.e., are 'unknown' to it), you can choose to show them anyway by selecting 'Paste unknown values'.

Page | 370

## Create Orphan Nodes

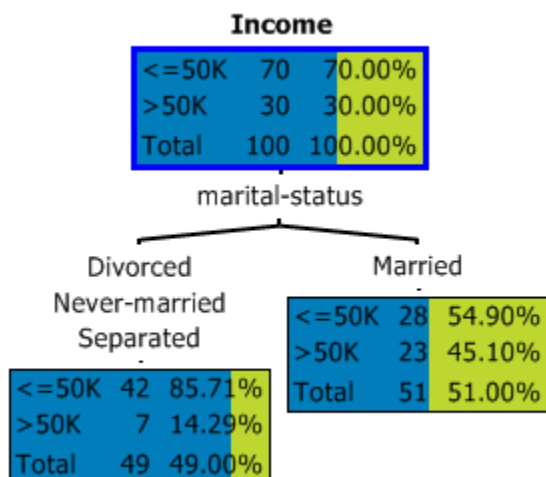
Orphan nodes are created when a split is pasted to a tree whose dataset contains records that were not in the original dataset. This may occur when you copy splits from a tree created with a small dataset and paste to a tree created with a larger dataset.

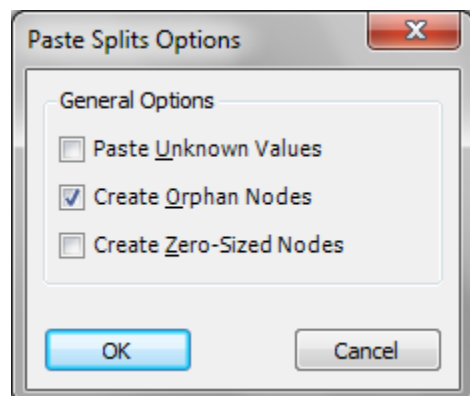
For example, in **Tree A**, the split has three categories, color=**red, blue, green**. This split is pasted to **Tree B** but the dataset used to create **Tree B** contains four categories for color, color=**red, blue, green, yellow**. There is no rule in **Tree A** for color=**yellow**. To include the records for the rule where color=**yellow**, you must make sure the **Create Orphan Nodes** option is selected. If this option is not selected, the sum of records in the child nodes may appear to be less than the number of records in the parent node.

**Example:** The **Create Orphan Nodes** option

Pasting splits in the case where only the Create Orphan Nodes option is selected.

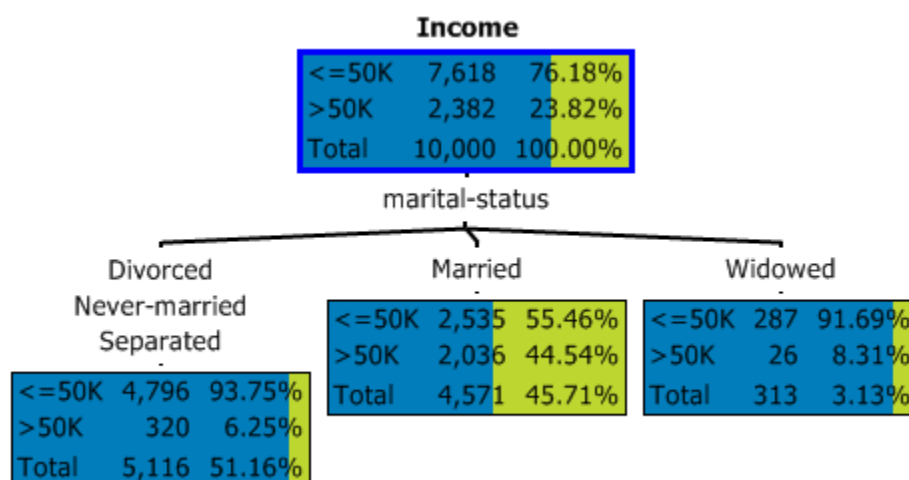
**Tree A**





We copied the split from the root node from **Tree A** and pasted the split to **Tree B**. The dataset for **Tree B** has the same data structure but contains more records. The split copied from **Tree A** does not have any records where **marital-status=Widowed**. However, **Tree B** contains data where **marital-status=Widowed**. If you did not select **Create Orphan Nodes**, records where **marital-status=Widowed** would never be displayed after the Paste Split operation. By selecting **Create Orphan Nodes**, these records will appear after the Paste Split operation in **Tree B**.

### Tree B



## Create Zero-sized Nodes

Zero-sized nodes are nodes that contain no records. They are created when a split is pasted to a tree created with a dataset that does not contain records that were in the original dataset. This may occur when you copy splits from a tree created in a large dataset and pasted to a tree created with a smaller dataset.

For example, in **Tree A**, the split has four categories, color=**red, blue, green, yellow**. This split is pasted to **Tree B** but the dataset used to create **Tree B** only has three categories for color, color=**red, blue, green**. If **Create zero-sized nodes** option is selected, **Tree B** will create a node with the rule color=**yellow** but node will contain zero records.

- If this option is not selected, the split will only contain nodes for color= **red, blue, green**.
- If this option is selected but option Paste Unknown values is not selected, some zero-sized nodes may not be created because they correspond to unknown categories in the destination node.

Page | 372

Example: The **Create Zero-sized Nodes** option

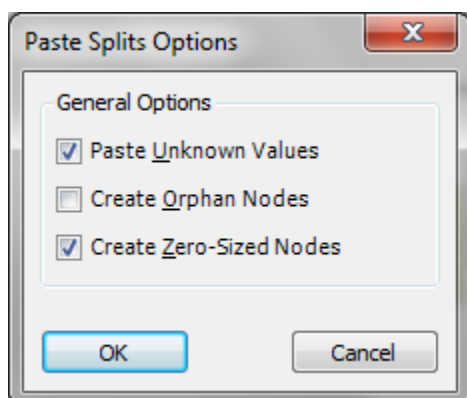
Pasting splits with the options Create Zero-sized Nodes and Paste Unknown Values.

### Tree C

| Income |        |         |  |
|--------|--------|---------|--|
| <=50K  | 7,618  | 76.18%  |  |
| >50K   | 2,382  | 23.82%  |  |
| Total  | 10,000 | 100.00% |  |

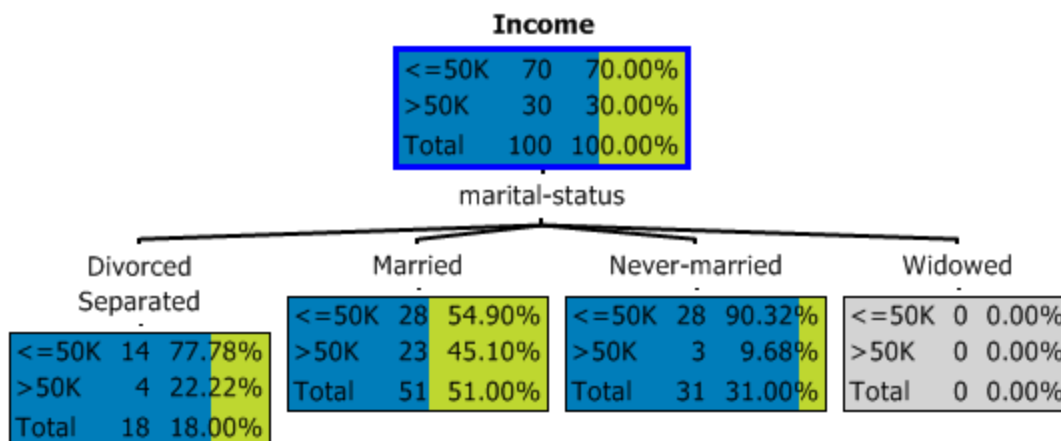
| marital-status     |                    |                    |                  |
|--------------------|--------------------|--------------------|------------------|
| Divorced Separated | Married            | Never-married      | Widowed          |
| <=50K 1,623 91.03% | <=50K 2,535 55.46% | <=50K 3,173 95.20% | <=50K 287 91.69% |
| >50K 160 8.97%     | >50K 2,036 44.54%  | >50K 160 4.80%     | >50K 26 8.31%    |
| Total 1,783 17.83% | Total 4,571 45.71% | Total 3,333 33.33% | Total 313 3.13%  |



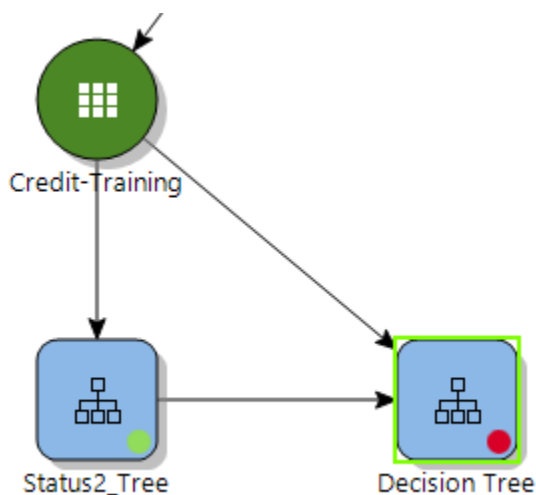
We copied the split from the root node from **Tree C** and pasted the split to **Tree D**. The dataset for **Tree D** has the same data structure but contains fewer records. This dataset does not have any records where marital-status=Widowed. However, the split from **Tree C** contains a rule marital-status=Widowed. **Tree D** shows the rule and the pasted node with the rule marital-status=Widowed as a zero-sized node.

### Tree D

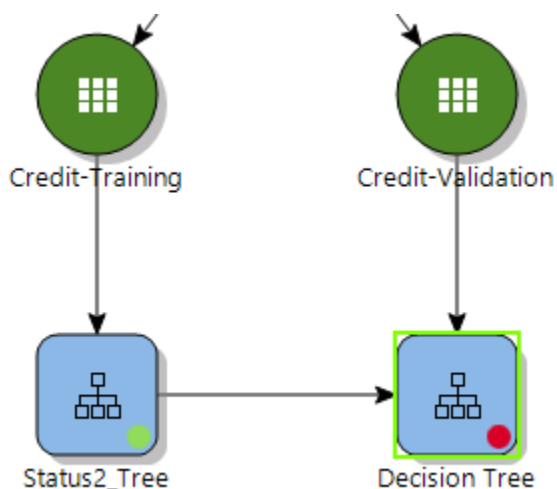




To copy a Decision Tree, add a new Decision Tree node to the workflow and connect the tree you want to copy to the new tree node. Then connect the dataset that the new tree should be based on. In the following example, the new tree is based on the same dataset as the original tree.

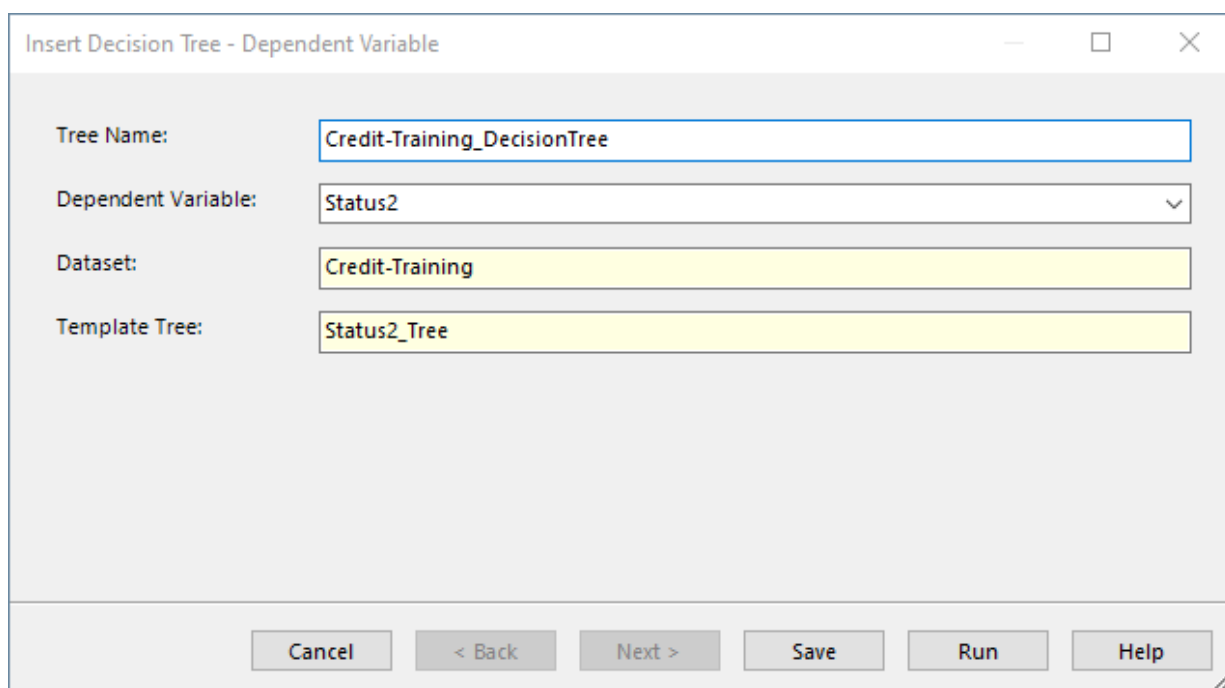


You can also connect a different dataset (other than the parent of the original tree), as in the second example below. This is equivalent to copying splits of the original tree to a new tree (consisting of the root node only) based on the new dataset. Note that the splits on the variables that do not exist in the other dataset will not appear in the new tree. It is assumed that the second dataset contains a variable with the same name and compatible data type as the dependent variable of the original tree.



Double-click on the new tree node or right-click on it and select **Modify** to open the wizard.

The name of the original tree is displayed in the Template Tree field. Specify the name of the new decision tree in the Tree Name field.



Click **Save** to save the settings or click **Run** to save the settings and run the process immediately.

The tree will be copied along with its attributes.

---

To copy the whole tree as an image in order to paste it to a presentation or another document, use the command **Edit | Copy Tree to Clipboard**. This command is also available on the right-click context menu of the tree background. Right-click anywhere on the tree background and select "Copy Tree to Clipboard". Then paste it to the desired document.

The command **Copy Node Data to Clipboard** copies the data, in a tabular form, from the active tree node and copies it to the clipboard. The copied data can then be pasted into another application, such as Microsoft Excel spreadsheet.

Page | 375

**Select the node whose data you want to copy**

### **From the main menu:**

**Edit | Copy Node Data to Clipboard**

### **From the background context menu:**

**Right click anywhere on the tree background and select "Copy Node Data to Clipboard"**

This command is only available when the decision tree model is the active view.

The command **Copy Screen to Clipboard** copies a bitmap image of the visible tree to the clipboard. This image can then be pasted into another application, such as Microsoft Word or Excel.

### **From the main menu:**

**Edit | Copy Screen to Clipboard**

### **From the background context menu:**

**Right click anywhere on the tree background and select "Copy Screen to Clipboard"**

This command is only available when the decision tree model is the active view.

---

## Modifying Tree Attributes

### Using the Attribute Editor

The Attribute Editor allows you to modify variable attributes for the tree. The attribute values can be set and modified when inserting the tree in the tree wizard or modified after the tree has been inserted. Some of the tree parameters determined by the values set in the Attribute Editor are:

Page | 376

- whether a variable is included in the model
- the role of variables in the model (dependent, independent, or weight)
- the maximum number of nodes in splits produced by Find Split and AutoGrow
- the maximum number of bins or intervals in the Force Split Range Editor
- whether the values of a variable are treated as ordered or not
- whether grouped values of numeric variables are displayed in intervals or as discrete values
- the way missing values are treated

While a decision tree is open, the Attribute Editor can be invoked as follows:

#### From the menu:

Tools | Attribute Editor...

#### From the background context menu:

Right-click anywhere on the tree background and select Attribute Editor...

Attribute Editor

| #  | Variable Name  | Include | Role        | Usage      | P-Value | Default # Bins | Intervals Type | Break-apart | Missing values |
|----|----------------|---------|-------------|------------|---------|----------------|----------------|-------------|----------------|
| 1  | age            | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 2  | workclass      | yes     | independent | nominal    | 0.05    | 9              |                | yes         | use            |
| 3  | fnlwgt         | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 4  | education      | yes     | independent | nominal    | 0.05    | 50             |                | yes         | use            |
| 5  | education-num  | yes     | independent | ordinal    | 0.05    |                |                | yes         | float          |
| 6  | marital-status | yes     | independent | nominal    | 0.05    | 7              |                | yes         | use            |
| 7  | occupation     | yes     | independent | nominal    | 0.05    | 50             |                | yes         | use            |
| 8  | relationship   | yes     | independent | nominal    | 0.05    | 6              |                | yes         | use            |
| 9  | sex            | yes     | independent | nominal    | 0.05    | 2              |                | yes         | use            |
| 10 | capital-gain   | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 11 | capital-loss   | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 12 | hours-per-week | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 13 | native-country | yes     | independent | nominal    | 0.05    | 50             |                | yes         | use            |
| 14 | Income         | yes     | dependent   | discrete   |         |                |                |             | use            |

OK Cancel Help

Scroll to the right to see more attributes:

Attribute Editor

| #  | Variable Name  | Missing values | Ordered display | Max Branches | Open Left                | Open Right               | Cardinality | Format String |
|----|----------------|----------------|-----------------|--------------|--------------------------|--------------------------|-------------|---------------|
| 1  | age            | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 73          | 0             |
| 2  | workclass      | use            | present         | 0            |                          |                          | 9           |               |
| 3  | fnlwgt         | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 12787       | 0             |
| 4  | education      | use            | present         | 0            |                          |                          | 16          |               |
| 5  | education-num  | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 16          | 0             |
| 6  | marital-status | use            | present         | 0            |                          |                          | 7           |               |
| 7  | occupation     | use            | present         | 0            |                          |                          | 15          |               |
| 8  | relationship   | use            | present         | 0            |                          |                          | 6           |               |
| 9  | sex            | use            | present         | 0            |                          |                          | 2           |               |
| 10 | capital-gain   | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 113         | 0             |
| 11 | capital-loss   | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 82          | 0             |
| 12 | hours-per-week | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 89          | 0             |
| 13 | native-country | use            | present         | 0            |                          |                          | 41          |               |
| 14 | Income         | use            |                 |              |                          |                          | 2           |               |

OK Cancel Help

The global defaults for the tree attributes can be set in the Tree Attributes tab of the Preferences dialog accessible from the Tools menu. See the topic [Preferences: Tree Attributes](#) for details.

**Tip:** you can change attribute values for more than one variable at a time. To this end, click in the attribute column of your choice and drag your mouse down while holding down the left mouse button to highlight several cells in the column. (Alternatively, while holding down the SHIFT or CTRL key on your keyboard, click on several cells whose values you want to change in the selected column.) When finished selecting multiple cells, set the value for one of them (the last one you highlighted). This will set the same value for all highlighted cells in the selected column.

Page | 378

You can also click the Field Finder (binoculars) button to quickly find and select the desired variable and use the search capability.

Tree attributes can be copied from one tree to another. Open the tree whose attributes you want to copy and open its Attribute Editor. Click the Copy button and close the Attribute Editor. Then open the other tree, open its Attribute Editor and click the Paste button. All attributes for the variables that are common for both trees will be copied. Click OK to apply the changes and close the Attribute Editor.

NOTE: After you apply changes made in Attribute Editor and return to the tree view, you can still undo them using the Undo command, provided that you haven't made changes to a different tree or dataset.

## Tree Attributes

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Variable Name</b> | Name of the variable in the decision tree.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Include</b>       | Determines if a variable is to be included in the analysis.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Role</b>          | Determines how a variable is used in an analysis. A number of options are available:<br><br><b>Dependent:</b> Indicates that this variable is the dependent variable, or the variable that you want to explore. There can only be one dependent variable (DV).<br><br><b>Independent:</b> Indicates that this variable should be compared to the dependent variable to determine if it has predictive value.<br><br><b>Weight:</b> Indicates that this variable should be used as a weight that controls how much emphasis each record should be given in the analysis. If this option is selected, you must also select the type of weighting in the Weight Type column.<br><br><a href="#">Example</a> |
| <b>Usage</b>         | Algorithms can treat values in a number of ways. In general, if the values of a field are ranked (like 1, 2, 3, or 1-low, 2- medium and 3-high) then they are 'ordinal'. If there is no ordering, i.e., if they are just categories or qualities, they are considered to be 'nominal' values (red, blue and green).                                                                                                                                                                                                                                                                                                                                                                                      |

**No binning:** Each category within the variable will form its own separate node in the tree. The search for similar groupings is turned off.

**Ordinal:** Treats values as an ordered set (e.g., 1-low, 2-medium, 3-high or 1, 2, 3). Only adjoining values will be grouped together when an algorithm is looking for similarities. For example, 1-low can be paired with 2-medium, or 2-medium can be paired with 3-high but 1-low cannot be paired with 3-high.

**Nominal:** Values combine freely. For example, red may combine with blue or green. All possible groupings of field categories are examined so that similar items are grouped together to become a node in the tree.

**Continuous:** The variable is split into intervals. Intervals are calculated in a number of different ways, based on your selections in the Intervals Type column.

**Discrete:** Only available with the dependent variable. Allows you to keep dependent variables ungrouped.

### Example

#### ***p-value***

Defines the level of confidence required to merge values into groupings. This setting affects the bushiness of the decision tree - a low setting (0.001) will tend to produce few branches (splits) from each node, whereas a high setting (0.20) will produce a bushy tree with more branches. The default setting is 0.05. With a p-value setting of 0.05 you would typically only be wrong considering values as category components 5 times out of 100. Similarly, a setting of 0.001 will assure you that you have a 99.9% accuracy in the category assignments but that you will also have fewer resulting categories. A setting of 0.20 would result in more branches but with less certainty of the accuracy (80% confidence).

#### ***Default of Bins*** #

- If **Usage** is **continuous**, the value in this column determines the number of intervals for the selected variable according to the following rule: To form splits in a decision tree, continuous values are initially divided into discrete intervals where each interval has approximately the same number of observations. These discrete intervals are then treated as ordered and combined accordingly. The number specified here determines how many **initial** intervals are formed for this continuous variable prior to the ordered grouping. Therefore, the resulting number of nodes may be less than or equal to the value in the Interval column. For example, if Interval=8, the resulting number nodes can be less than or equal to ( $\leq$ ) 8.

See the topic "About Intervals/Bins" for details.

- If **Usage** is **nominal**, the value in this column determines the maximum number of categories (bins) to display for the selected variable. The default value is equal to the cardinality or 50 (whichever is less). You may enter any integer number between 1 and the cardinality of the variable. If the maximum number of bins **N** is equal to the cardinality of the variable, then each of its values is represented as a separate category. If **N** is less than the cardinality, the following mapping is applied: the first **N-1** unique values are represented as separate categories and the rest are mapped into one category denoted by triple number sign ("###").

#### ***Intervals Type***

This option affects the Find Split and Automatic Grow operations and applies only to the variables with *continuous* Usage (see above). It defines

the level at which the binning maps for continuous independent variables are calculated.

**Static:** The binning map for the variable is precomputed at the root node. The intervals remain static throughout the tree.

**Dynamic:** The binning map for the variable is calculated dynamically at every node.

Although the Static type is the default, the Dynamic type provides more intelligent mapping and is therefore recommended in most cases. Note that splitting a variable with dynamic interval type is up to two times slower than with the static type, but this may not be noticeable on fast computers.

### Break-apart

Allows you to optimize the growing of the tree by choosing either speed or accuracy. There are two options:

**Yes:** this setting gives the most accurate results. **Yes** is the default.

**No:** this setting is used when you have variables with many categories and you want to speed up the analysis. However, the results may not be optimal.

### Missing Values

Allows you to specify how missing values in your data are handled in the tree. Note that the effect and the applicability of the missing value options depends on the variable usage (independent, dependent, or weight) and its Usage.

#### 1. For the dependent variable (DV):

**Use** (applicable to discrete DV only): Missing values will be treated like any other value of the DV. More precisely, a missing value will be considered a separate DV category designated as **null** and will be shown in every node together with other DV values. This option is not applicable to continuous DV.

**Ignore:** All records containing missing values in the dependent variable will be excluded from the analysis. Note that in this case the root node will contain fewer records than the parent dataset. If the DV of a Decision Tree is continuous, the records with missing values in the DV are always excluded, so **Ignore** is the default and the only available option for continuous DV in Decision Trees. This does not apply to Strategy Trees.

Note that if you ignore missing values in the DV, it will affect all tree views except for the Data tab. which will still show records with missing values in the DV. (The Data tab always shows the actual raw data for any selected node, which is determined by the node rule only, not by the settings in Attribute Editor).

#### 2. For the independent variables (IVs):

**Use:** A missing value in the selected variable will be considered one of its normal categories and will be used to calculate the split like any other category. It is designated as **null** and shown in every split on this variable - in the node rules, in the Range Editor of Force Split and Edit Split, etc.

This option is the default for discrete independent variables (those whose Usage is "*nominal*" or "*no binning*")

**Ignore:** Missing values in the independent variable will be ignored. This means that any split on this variable will not include the condition "IF <variable\_name> is NULL", so the child nodes in this split will not contain records with missing values in this variable. In this case the sum of the numbers of records in the child nodes of a split will not necessarily add up to the number of records in their parent node.

**Float:** Missing values in the variable will not be used to calculate the split, but will then be grouped with the branch (child node) that presents the most similar behavior in terms of the target (DV) distribution. This option is the default for continuous and ordinal independent variables.



### 3. For variables used as weight: Not applicable.

#### **Ordered Display**

Allows you to define how ordered variables are displayed. There are three options:

**Present:** displays all values which are present in the node. If a particular value of the variable is not present in the node, it will not be listed.

**All:** displays all values that fit within the range in the node, whether they are present in that particular node or not. For example, if a node contains the values 1, 2, 4, and 5 (and the value 3 exists somewhere in the dataset), the value 3 will be listed for the node.

**Range:** displays values as a range. For example, if a split groups the values 1, 2, 3, and 4 together, the values will appear as [1, 4].

Page | 381

#### **Max Branches**

The Max Branches value determines the maximum number of branches (leaves) produced by Find Split on the given variable. For example, if you want Find Split to produce binary splits for a specific independent variable, specify "2" in the Max Branches column for this variable. If you want to grow a binary tree (i.e., a tree with binary splits only), enter "2" in the Max Branches column for all variables. See above for the tip on changing attributes for more than one variable at a time.

Note that this setting does not apply to the Force Split operation. Also, note the difference from the Default # of Bins attribute. The Find Split algorithm starts calculating the split with the number of branches determined by Default # of Bins value. As opposed to Default # of Bins, Max Branches never leads to the creation of the other ("###") category for discrete variables.

#### **Open Left**

The splits created by the independent variable are not bound by the minimum data values found in the dataset.

#### **Open Right**

The splits created by the independent variable are not bound by the maximum data values found in the dataset.

#### **Cardinality**

The cardinality (number of unique values) of the variable.

#### **Format String**

Allows you to specify various formats (such as currency, percent, etc.) and precision for displaying values in the tree. You can either choose one of the standard formats from the Format String drop-down list or enter a custom format. For detailed description of format strings, see [Specifying Format](#).

## Weight in Decision Trees

Weight in datasets and models can be viewed as a generalized frequency weight (the weight values don't have to be integer). A weight can be set at the dataset level, but it can also be set and changed independently in the dependent models.

See the [Weight](#) topic for general information about weights and on setting weight at the dataset level.

## Weights in Datasets and Dependent Models

Page | 382

The weights in a dataset and its dependent models can be changed or reset independently. When you toggle weight on or off in the dataset Overview Report, this does not affect existing trees and other models depending on the dataset.

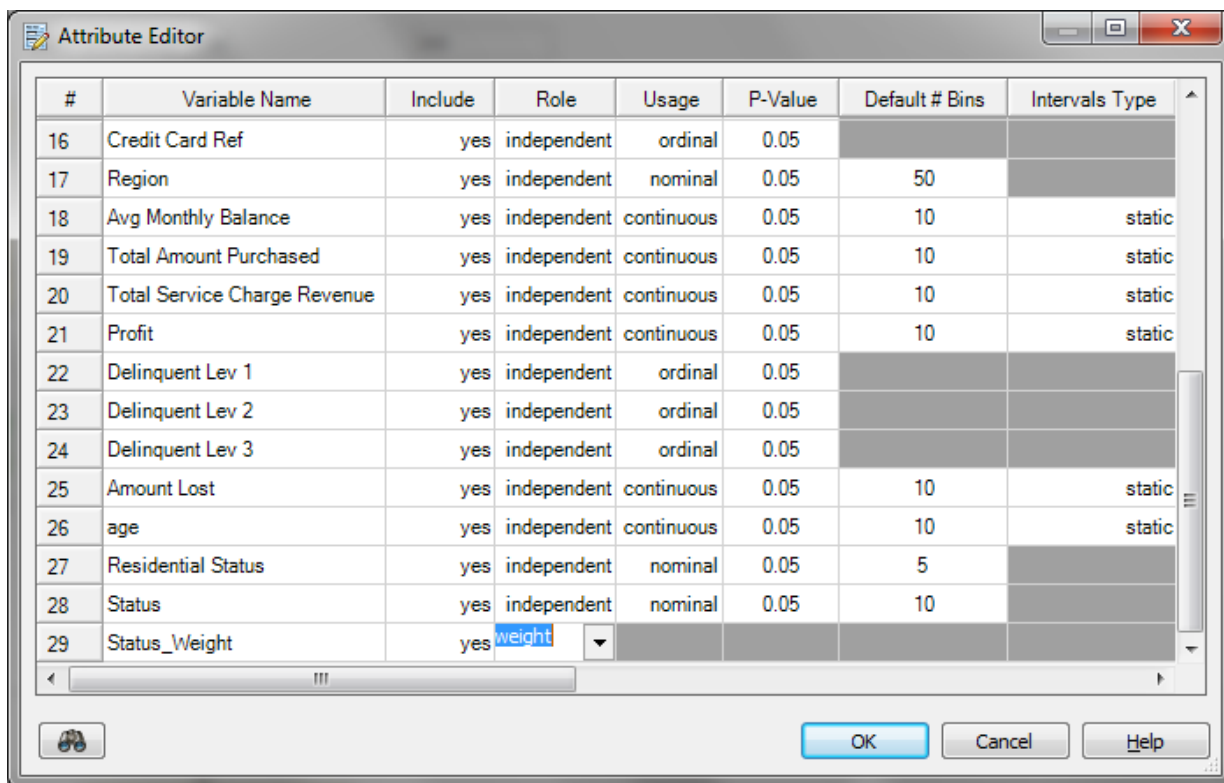
By default, any new model created on the basis of a weighted dataset inherits the weight from it. More precisely, when inserting a model for a weighted dataset, the weight field defined in the dataset is automatically listed in the model's Attribute Editor as "weight" (in the Usage column). You can override this in the wizard of the corresponding model node and redefine the field as an analysis variable by changing Usage from "weight" to "independent" if desired. Once the model is inserted, it is independent of the parent dataset in the sense that any changes of weights in the model will not affect the parent dataset and vice versa.

## Weight in Decision Trees

To specify weight for an existing decision tree, open the tree and select **Tools | Attribute Editor** from the menu.

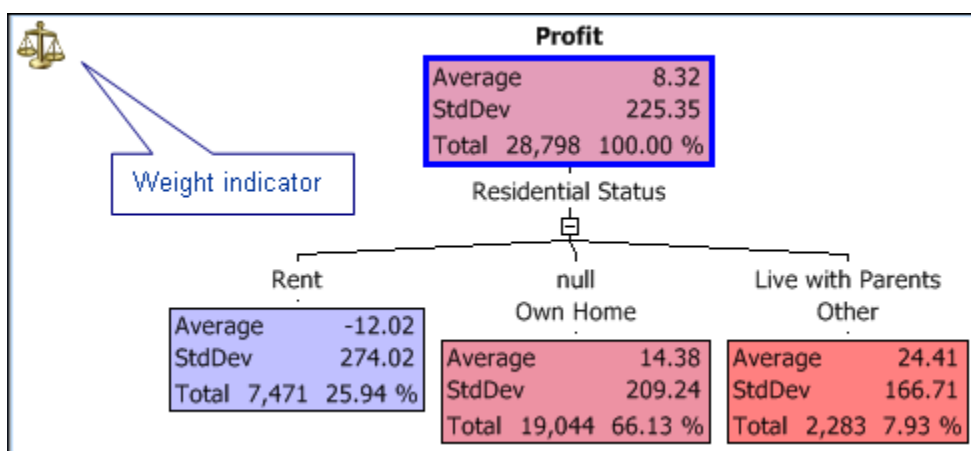
To specify weight for a new decision tree in the Decision Tree Wizard, open Attribute Editor by clicking the Advanced button in the last page of the wizard. (It is assumed that the field you want to set as weight already exists in the parent dataset. See the [Weight](#) topic for information on creating a weight field.)

Click in the **Usage** column for the variable you want to set as weight and select "**weight**" from the drop-down list. Note that this usage option is available only for nonnegative numeric variables. The weight variable does not have to be integer.



Note that any existing splits in the tree will be deleted after you specify weight.

If the weight is defined, the letter “W” shown in the top left corner of the Tree view serves as an indicator that the tree is weighted. In addition to Attribute Editor, the information about the weight is available in the Parameters and Attributes tab of the tree.



Since weight is treated as a (generalized) frequency weight, the effect of setting a weight in trees is as follows:

The number of records in all tree nodes is recalculated based on the defined weight (equals the sum of the weight field over all records in each node).

NOTE: Non-integer node totals are possible, since the sum of the weight field over a node is not necessarily integer. By default, the node totals displayed in tree nodes are formatted as integer numbers, but you can change this setting by selecting Tools | Options and changing the format in the *Show Support (# of records)* field, for example, to 0.00 or #,##0.00 (for two digits after the decimal point).

The Dependent Variable distribution in all nodes is recalculated based on the defined weight. The weighted distribution in a given node is as follows:

If  $W$  is the weight variable,  $F_i$  is the  $i$ th value of the discrete dependent variable  $F$ , and  $SUM(W_i)$  is the sum of the weight field over all records in the node where  $F=F_i$ , then

- The weighted total number of records =  $SUM(W)$  (over the node)
- The weighted frequency for the value  $F_i$  =  $SUM(W_i)$  (over the node)
- The weighted percentage of the value  $F_i$  =  $100 * SUM(W_i) / SUM(W)$  (sums taken over the node)

The effect in other tree views: All charts in the Tree Chart and Profile Chart are calculated based on the defined weight. Also, note that the columns of Node Report may display non-integer values.

## Variable Usage

By setting the **Usage** options, you can define the way in which the values of a field are treated by the tree algorithm and, therefore, the way they will be displayed in the splits (how they will be grouped, whether they will be represented in ranges, etc). For example, if the values of a field are ranked (like 1, 2, 3, or low, medium and high) then they are 'ordinal'. If there is no ordering, i.e., if they are just categories or qualities, they are considered to be 'nominal' values (red, blue and green).

### Usage set to 'no binning'

When set to **no binning**, each value of the independent variable forms a separate branch, i.e., individual values are presented as unique nodes in the tree.

Example

### Usage set to 'ordinal'

Values of this Usage are treated as an ordered set (e.g., 1, 2, 3, or a,b,c...). Only values immediately succeeding one another will be grouped together when an algorithm is looking for similarities. For example, 'a' can be paired with 'b', but not with 'c'.

This Usage is the default for independent variables of types Number, Integer, Date, Time, or Timestamp if their cardinality is less than 50.

You can assign the Usage 'ordinal' to String variables (which are 'nominal' by default). In this case, when creating the split, the algorithm will order the values alphabetically and then try to group contiguous values together to create a branch. If your data is not alphabetically ordered, you may want to create a calculated column that transforms it. For example, use the Binning helper in Dataset Editor to transform the category 'low' to '**1-low**', 'medium' to '**2-medium**' and 'high' to '**3-high**'.

Example

Page | 385

## Usage set to 'nominal'

Values of this Usage are treated as an unordered set. All possible groupings of field categories are examined so that similar items are grouped together. This Usage is the default for String and Boolean variables.

Example

## Usage set to 'continuous'

This Usage is available for independent variables of types Number, Integer, Date, Time, or Timestamp and it is the default if the cardinality is higher than 50. A variable of this Usage is split into intervals that are calculated in a number of different ways based on your selections in the Intervals Type column and the [Intervals/Bins](#) column. See also the topic [Using the Attribute Editor](#) for more information about the Intervals/Bins column.

## Usage set to 'discrete'

This Usage is only available for dependent variables. It keeps the variable ungrouped. Each value is represented as an individual category. Dependent variables of types String and Boolean are always discrete. Dependent variables of types Number, Integer, Date, Time, or Timestamp are considered discrete by default if their cardinality is less than or equal to 10. Otherwise, if their cardinality is greater than 10, their default Usage is continuous.

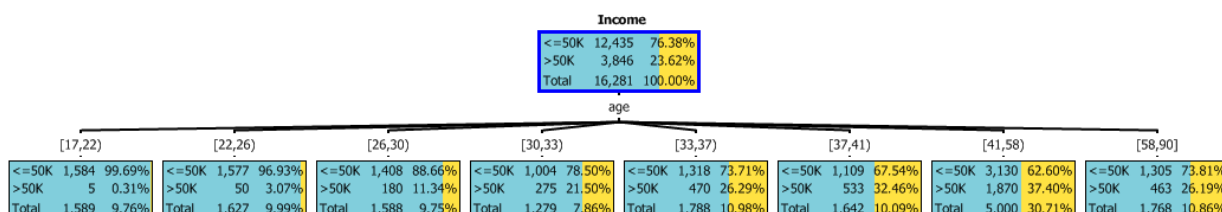
## Intervals/Bins

The **Default # Bins** attribute affects both Find Split and Force Split commands.

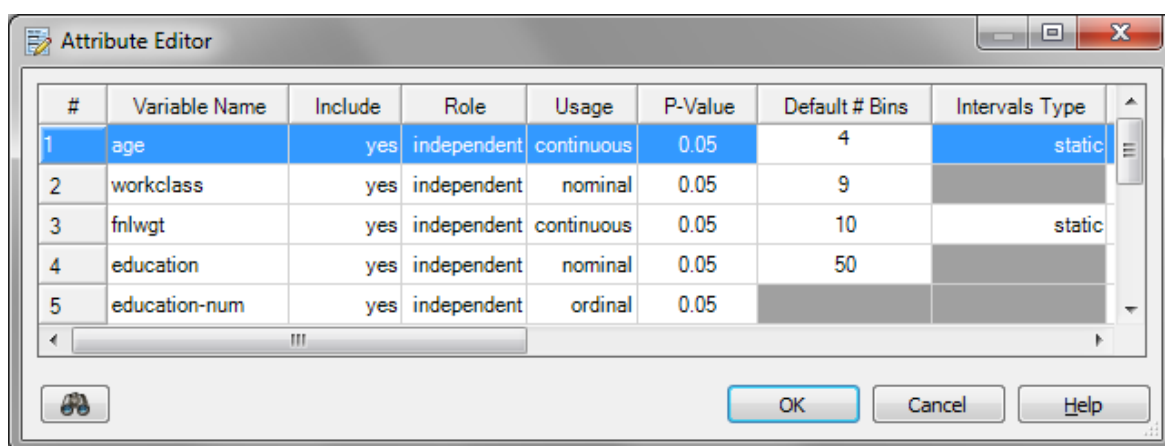
For **Find Split**, this attribute affects the maximum number of child nodes that can be produced by Find Split for a given independent variable (IV). More precisely, it determines the maximum **initial** number of intervals (for continuous IVs) or bins (for discrete IVs) created at the initial stage of the Find Split operation. During the steps of the Find Split algorithm, some intervals (or bins) may be merged (grouped), so the number of intervals or bins (and therefore child nodes) in the resulting split cannot exceed the value of this attribute.

For **Force Split**, the **Default # Bins** attribute determines the default number of intervals (for continuous IVs) or bins (for discrete IVs) in the Force Split Range Editor dialog. The actual number of intervals in the Range Editor for a given IV is the lesser of the Intervals/Bins value and the cardinality of the variable.

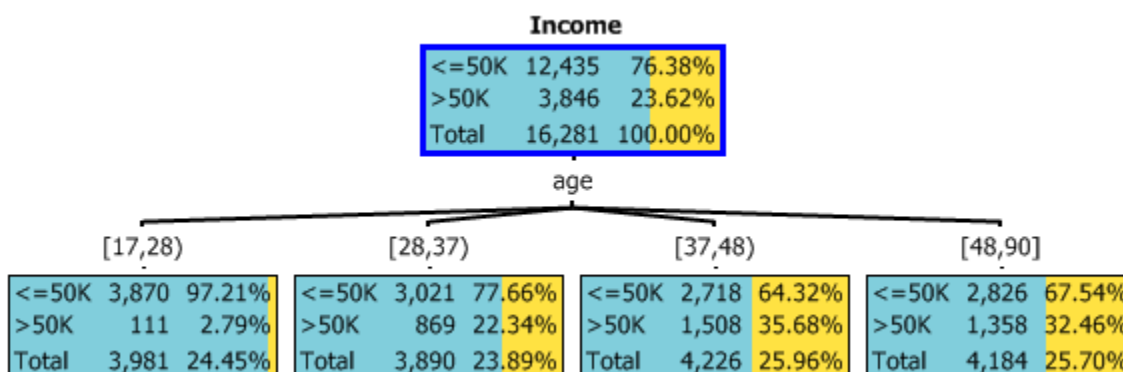
Example: After the Find Split operation, a split on the variable *age* is found. The Default # Bins value for *age* was set to 10. As shown below, Find Split produces eight child nodes:



In Attribute Editor, change the Intervals/Bins value for *age* to 4:



After applying the change, the split on *age* will be removed from the tree. Performing Find Split after this change will produce only four child nodes:



## Node Datasets

In any tree, you can select one or several nodes and create a separate dataset that would contain the records from the selected node(s) only. The resulting dataset is called a node dataset.

---

Creating a node dataset can be useful in many cases. For example, if you used a decision tree to isolate a certain segment and you are not interested in analyzing any other data, then you may want to insert a node dataset based on this segment.

A node dataset has the same properties as any other dataset in the project. It is only referred to as a node dataset because of where the data originated. In the Project Outline, a node dataset is shown as a child object of the tree it is based on.

Page | 387

NOTE: A node dataset inserted from a node of a weighted tree inherits the weight from the tree. If necessary, you can turn the weighting off in the node dataset after it is created.

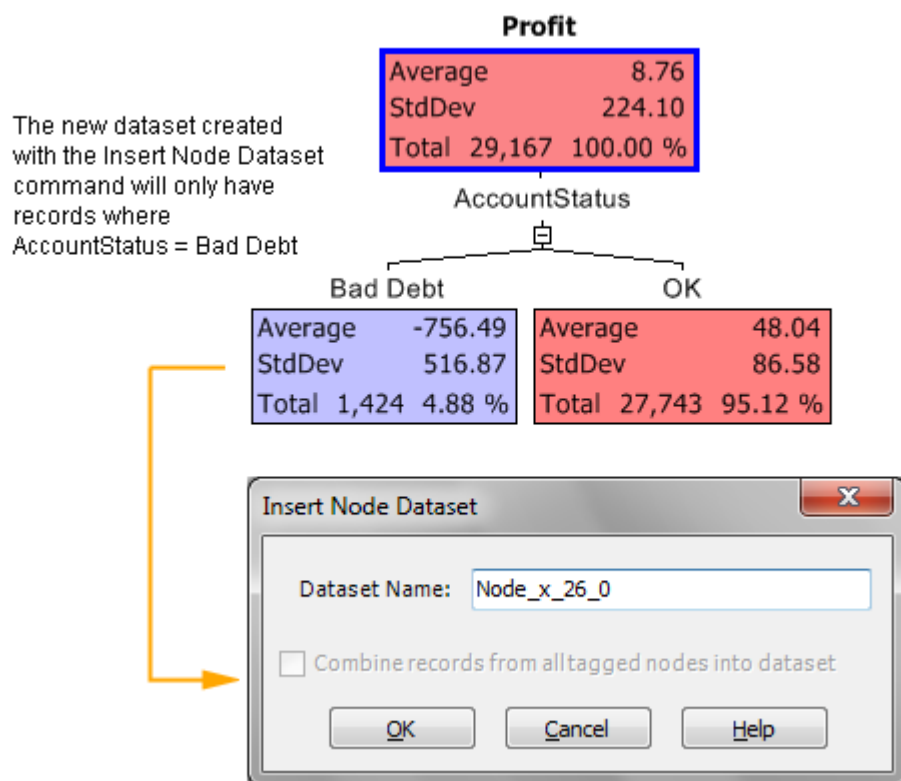
To insert a node dataset:

- Click on the node you want to use to create a node dataset
- To merge data from more than one node, tag each of the desired nodes (right-click and select Tag/Untag Node)

### From the menu:

#### Insert | Node Dataset

The Insert Node Dataset command can be used to stratify the data, as an alternative to the Partition Wizard. For example, if your dataset contains credit account data and you want to analyze only the bad debt accounts, insert a decision tree with any dependent variable (other than *AccountStatus*) and do a **Force Split** on the variable *AccountStatus*.



The default name for the node dataset is its Node ID (note that you can view Node IDs for each node if you include Node ID in the Tree Node Settings tab of the Tree Options dialog via **Tools | Options**). You can specify a different name, if desired.

The new dataset will appear as a child object of the decision tree in the Project Outline. In our example, in the Overview Report for the new dataset, the total number of records equals the number of records in the source node (1424), and it contains only the records where AccountStatus = 'Bad Debt'.

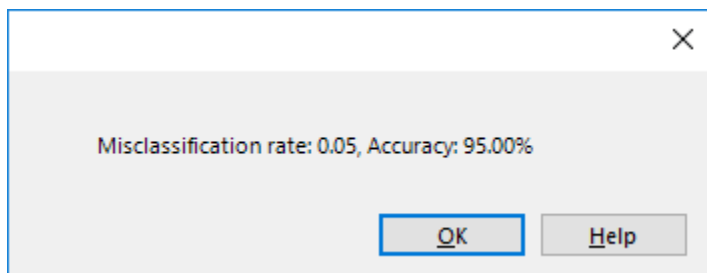
## Accuracy Estimate: Resubstitution

The **Tools | Resubstitution** command can be used to quickly estimate the level of accuracy for the current tree. Note that the Resubstitution command performs a static analysis of the model. Generally, the error rate will be underestimated, since no new data is used to test the tree. For comprehensive validation that compares the model with data, use Model Analyzer or the commands **Tools | Tree Validation** or **Tools | Validate** (if available in your configuration).

Resubstitution error results are presented in one of two ways - based on the type of the dependent variable (DV) used.



## Discrete DV

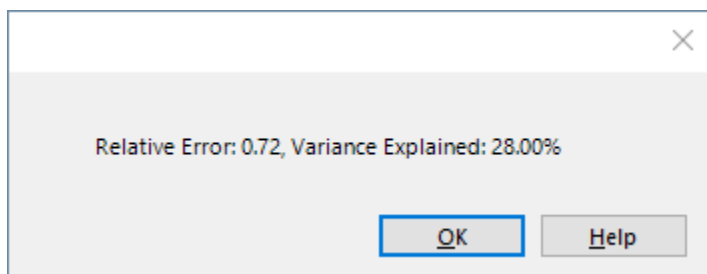


Page | 389

The value of Accuracy tells you how often you will be correct in assigning records from a sample into the DV's modal (predominant) category. For example, if the reported accuracy is 60%, then you will correctly assign, about 60% of the time, an unclassified record in the modal category of the node that corresponds to the unclassified record's attributes.

You can verify the accuracy of the calculation by looking at the percentage of observations in each of the **terminal** nodes of the decision tree that fall outside the modal category. The weighted sum of these percentages over all terminal nodes of the tree is the overall error rate (the error rate for a given node is weighted by the number of observations in the node relative to the total size of the tree).

## Continuous DV



The value of Variance Explained tells you how much better off you are using the decision tree to predict some outcome compared to using the average value as a guess of the dependent variable. For example, if the variance is 8%, you will likely guess a given sample's outcome 8% more accurately using the tree than you would by predicting that it will have the average value shown in the root node of the tree.

See also:

Model Analyzer

[Validation](#)

## Creating Calculation from Split

Every [split](#) on an independent variable V in a Decision Tree is a rule that can be represented by a conditional expression

```
CASE WHEN V in V_1 THEN <Node 1> WHEN V in V_2 THEN <Node 2> .... WHEN V in V_N THEN <Node N> END
```

Page | 390

where V\_1, V\_2, .... V\_N are subsets of the range of values of the variable V at the node being split.

Once you created a split with any tree growing command such as [Automatic Grow](#), [Find Split](#) or [Force Split](#), you can use the split as a binning transformation for creating a new expression for a calculated column with the conditional expression defined by the split. Thus, it can be a convenient alternative to creating binning transformations in the [Variable Transformations](#) node.

The command Calculation from Split creates an expression based on the split and writes it to an expression file that can be subsequently used in any Data Transformation step in your workflow simply by importing it into the Dataset Editor.

For example, suppose you have a Decision Tree with the dependent variable AccountStatus with values *Bad Debt* and *OK*. If you created a split on **age** such as the one shown below, you may want to bin the **age** variable (create discrete categories for it) exactly the way it is mapped by the split, and then work with the new, binned age variable instead.

| AccountStatus |        |         |  |
|---------------|--------|---------|--|
| Bad Debt      | 983    | 4.81%   |  |
| OK            | 19,434 | 95.19%  |  |
| Total         | 20,417 | 100.00% |  |

| age  |         |         |         |
|------|---------|---------|---------|
| null | [17,30) | [30,45) | [45,98] |

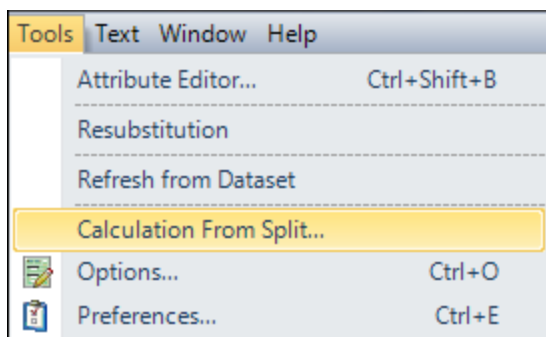
|          |     |        |
|----------|-----|--------|
| Bad Debt | 10  | 2.33%  |
| OK       | 419 | 97.67% |
| Total    | 429 | 2.10%  |

|          |       |        |
|----------|-------|--------|
| Bad Debt | 404   | 6.45%  |
| OK       | 5,862 | 93.55% |
| Total    | 6,266 | 30.69% |

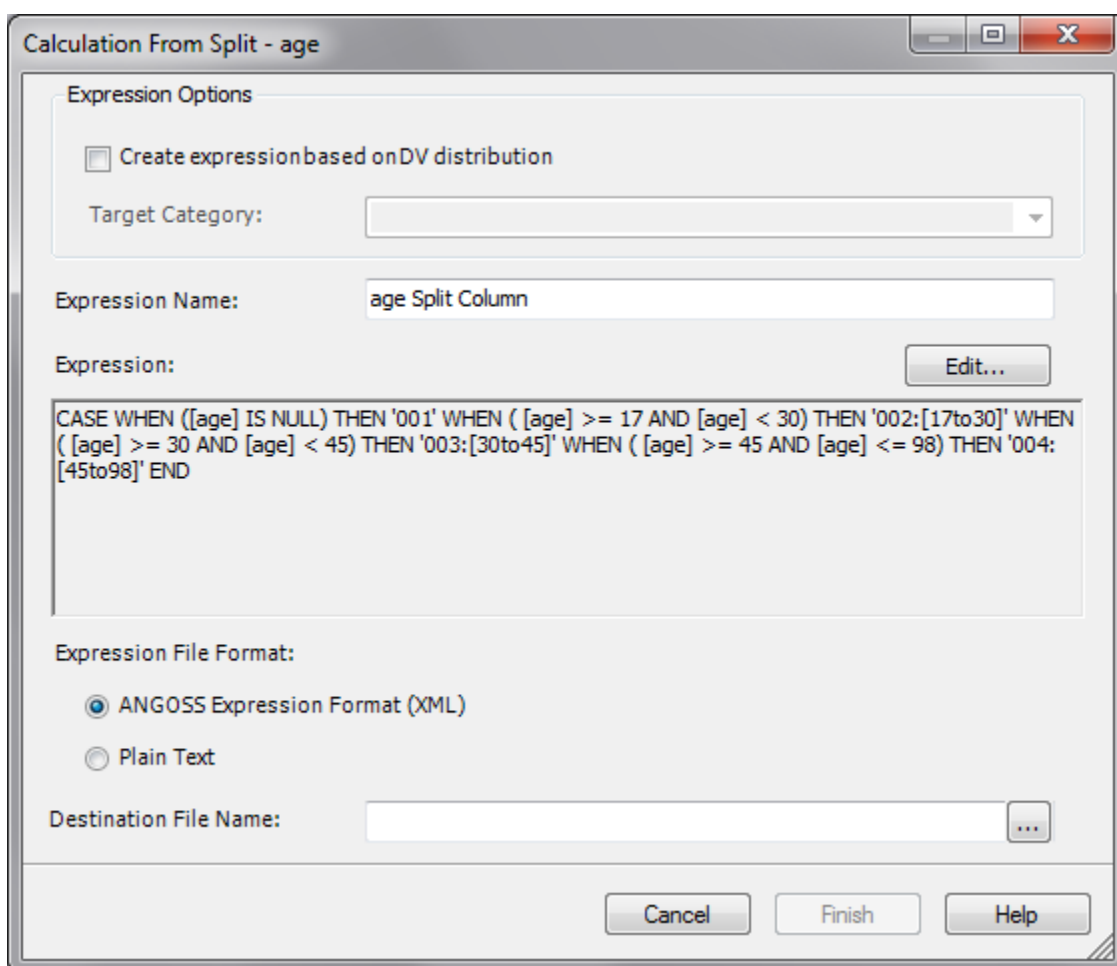
|          |       |        |
|----------|-------|--------|
| Bad Debt | 412   | 5.44%  |
| OK       | 7,157 | 94.56% |
| Total    | 7,569 | 37.07% |

|          |       |        |
|----------|-------|--------|
| Bad Debt | 157   | 2.55%  |
| OK       | 5,996 | 97.45% |
| Total    | 6,153 | 30.14% |

Click on the parent node in the split and select the menu command **Tools | Calculation From Split**.

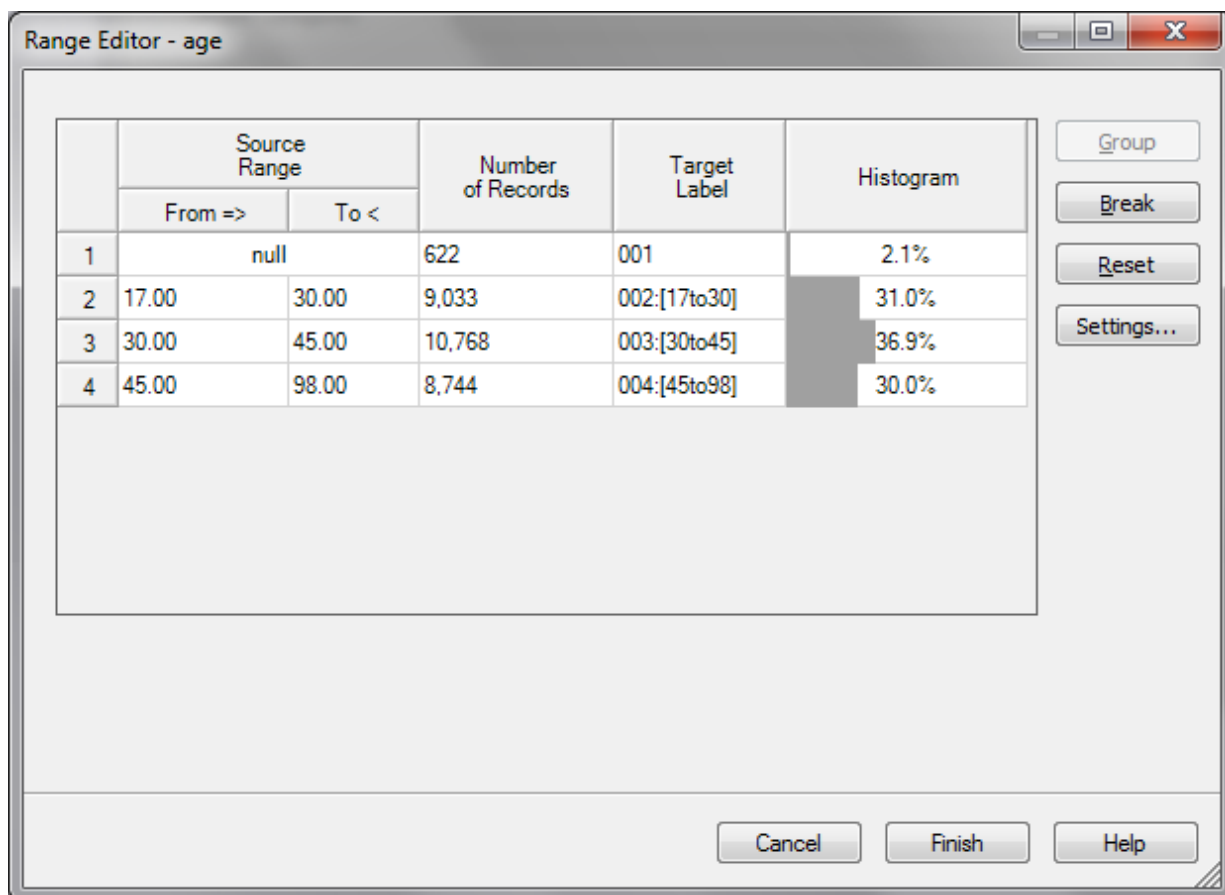


This brings up the Calculation from Split dialog. It allows you to choose the expression options, edit bins, specify the expression name, and save it in the an expression file.



By default, the new calculation is a simple binning, assigning bin labels to each bin in the split (the option *Create expression based on DV distribution* is not selected).

Click the **Edit** button if you would like to change the binning before creating the expression. This brings up the Range Editor dialog, where you can group the bins, edit their boundaries, and edit the bin labels in the Target Label column for the final expression.

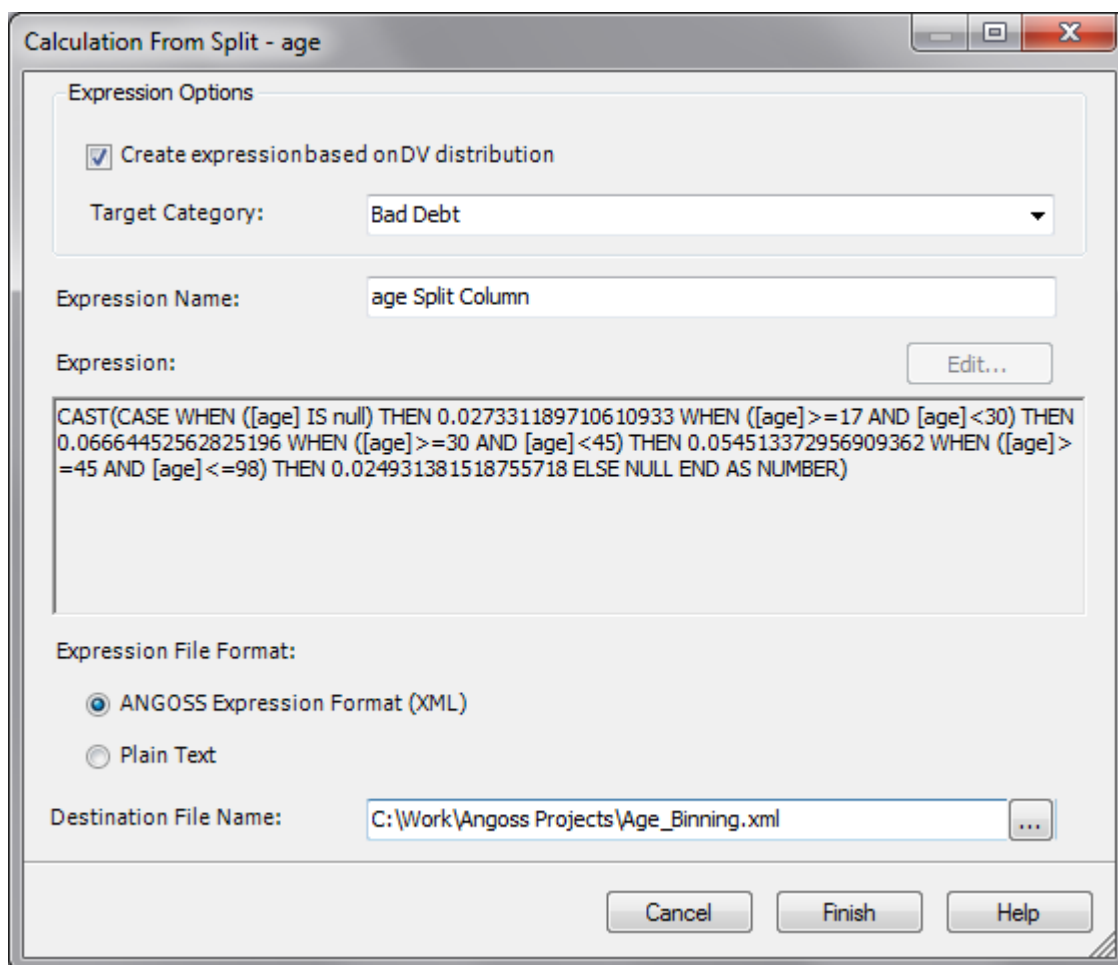


## Expressions based on the DV distribution

If you want the expression to reflect the distribution of the dependent variable (represent the scoring rule of the split), choose the option **Create expression based on DV distribution**.

### Discrete DV

If the dependent variable is discrete, this option creates a calculation whose value for each bin is the probability of one of the dependent variable outcomes. Select the target category. For example, given the split shown above, the value *Bad Debt* is selected as Target Category in the screenshot below. The resulting CASE WHEN expression produces the probabilities of Bad Debt in each of the bins (child nodes) in the selected split.



Calculation From Split - age

Expression Options

☒ Create expression based on DV distribution

Target Category: Bad Debt

Expression Name: age Split Column

Expression: [Edit...](#)

```
CAST(CASE WHEN ([age] IS null) THEN 0.027331189710610933 WHEN ([age] >= 17 AND [age] < 30) THEN 0.06664452562825196 WHEN ([age] >= 30 AND [age] < 45) THEN 0.054513372956909362 WHEN ([age] >= 45 AND [age] <= 98) THEN 0.024931381518755718 ELSE NULL END AS NUMBER)
```

Expression File Format:

☒ ANGOSS Expression Format (XML)

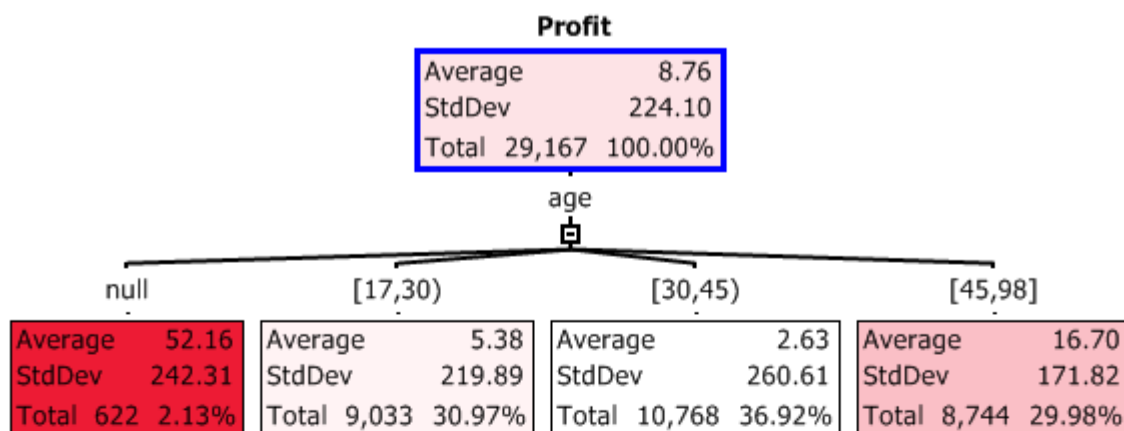
☐ Plain Text

Destination File Name: C:\Work\Angoss Projects\Age\_Binning.xml [...](#)

[Cancel](#) [Finish](#) [Help](#)

## Continuous DV

If the dependent variable is continuous, the command creates a calculation whose value for each bin is the average value of the dependent variable.



Calculation From Split - age

Expression Options

☒ Create expression based on DV distribution

Target Category:

Expression Name:

age Split Column

Expression:

Edit...

CAST(CASE WHEN ([age] IS null) THEN 52.162508038585273 WHEN ([age] >= 17 AND [age] < 30) THEN 5.3774416030111709 WHEN ([age] >= 30 AND [age] < 45) THEN 2.6341381872213923 WHEN ([age] >= 45 AND [age] <= 98) THEN 16.703778591033853 ELSE NULL END AS NUMBER)

Expression File Format:

☒ ANGOSS Expression Format (XML)
☐ Plain Text

Destination File Name:

C:\Work\Angoss Projects\AVG\_Profit\_By\_Age.xml

...

Cancel

Finish

Help

Choose the expression file format and specify the name of the output expression file. The default is the XML Expression Format (re-importable). It allows you to reuse expressions by importing the

expressions file into the Dataset Editor for Data Transformation or into Node Calculations for Strategy Trees.

Click **Finish** to create the expression file. You can use it in your workflow by importing it by means of the **Import Expressions** button available in the Dataset Editor for Data Transformation or in the Calculations dialog of Strategy Trees.

## See also:

[Find Split](#)

[Force Split](#)

[Edit Split](#)

[Variable Transformations](#)

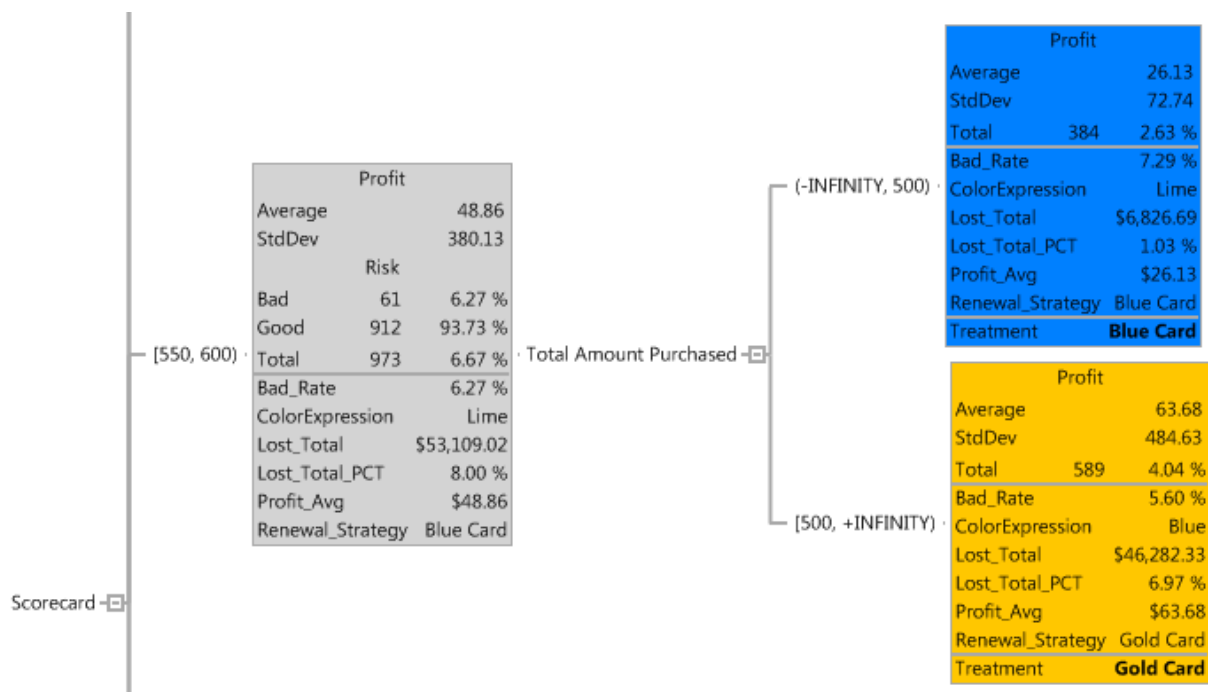
[Function Syntax](#)

## Strategy Trees

A **strategy tree** is a series of rules and actions that are developed to help achieve a business objective when interacting with a customer. A tree metaphor, very similar to decision trees, is used to define conditions and associated actions. Each segment (or node) can also display tree calculations that can compute key performance indicators such as profit, losses, purchasing behaviors or other important measures.

Figure1 shows a fragment of a strategy tree with five calculations, such as  $\text{Profit\_Avg} = \text{AVG}([\text{Profit}])$  and treatments *Gold Card*, *Blue Card*, and *Cancel Card* assigned to three terminal nodes.

Figure 1 - Example strategy tree (fragment)



## Strategy Trees Compared to Other Model Types

The Strategy tree model differ from other models. To understand strategy tree models it is helpful to compare them to other model types:

- **Strategy Trees are created manually based on business rules supplied by the user.**  
By contrast, predictive models are generated automatically by applying a mathematical algorithm to historic data.

- **Strategy Trees prescribe an action.**

By contrast, a classifying predictive model predicts the likelihood of a particular outcome. A strategy tree may use this likelihood to prescribe an action. For example, a Strategy tree may state, “if the probability of a customer responding to offer A is greater than 10%, then send then offer A”. Consider Strategy Tress to be Prescriptive Models.

- **Strategy Trees do not need dependent variable but can have dependent variables assigned locally.**

By contrast, a decision tree has a single dependent variable that is used for each and every split. To help optimize the business rules, a dependent variable can be used when constructing



Strategy Trees, but this is only to help find optimal groupings and the dependent variable can vary throughout the tree.

- **Strategy Trees can display the distribution of any variable.**

By contrast, a decision tree has a single dependent variable and its distribution is always displayed in each and every node. With Strategy Trees, you can display the value distribution for any variable in your dataset and apply a great variety of calculations.

Page | 397

## Inserting a Strategy Tree

A Strategy Tree node can be added to the workflow in one of the following ways:

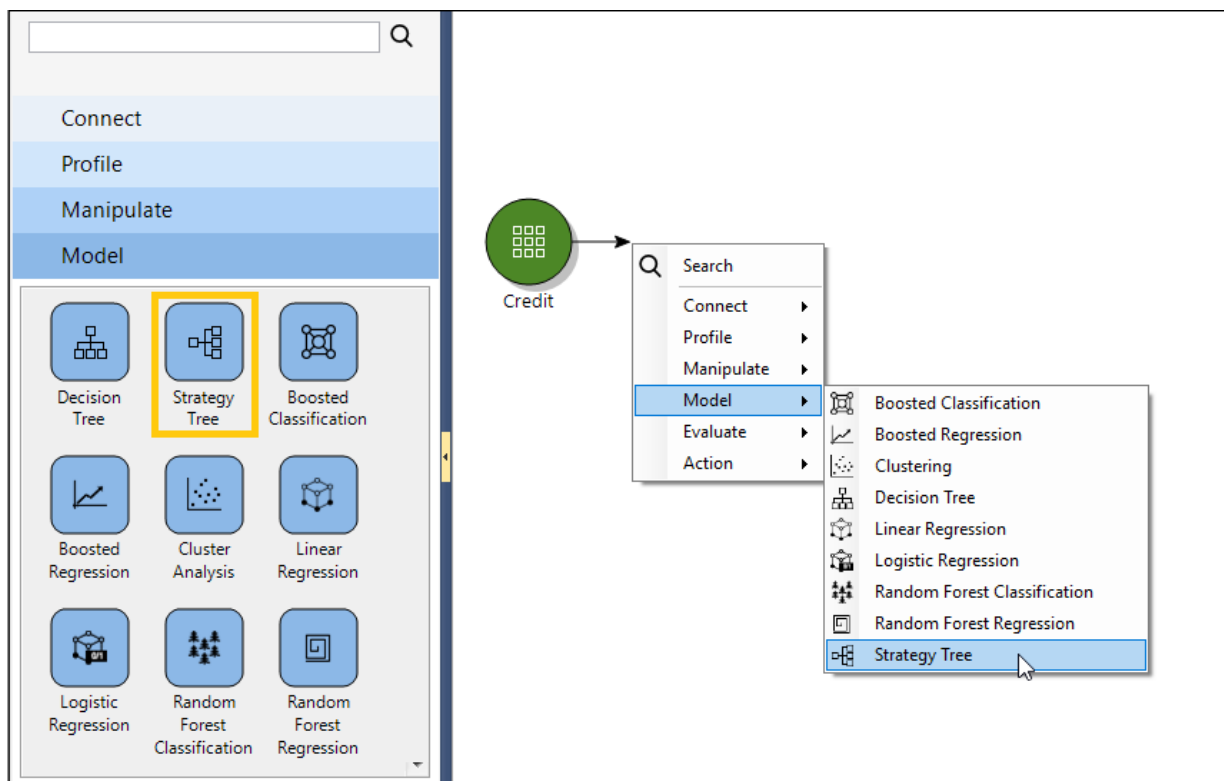
- Drag a connection arrow from a dataset node to any area on the Workflow canvas background. From the context menu that opens upon releasing the mouse button, select **Model --> Strategy Tree** as in the example screenshot below. A new **Strategy Tree** process node will be created and the wizard will open.

OR

- Right-click on the Workflow canvas and select **Model --> Strategy Tree** from the context menu. Connect a dataset node to the tree node.

OR

- Open the **Model** palette and drag the **Strategy Tree** node onto the workflow canvas. Connect a dataset node to the **Strategy Tree** node.

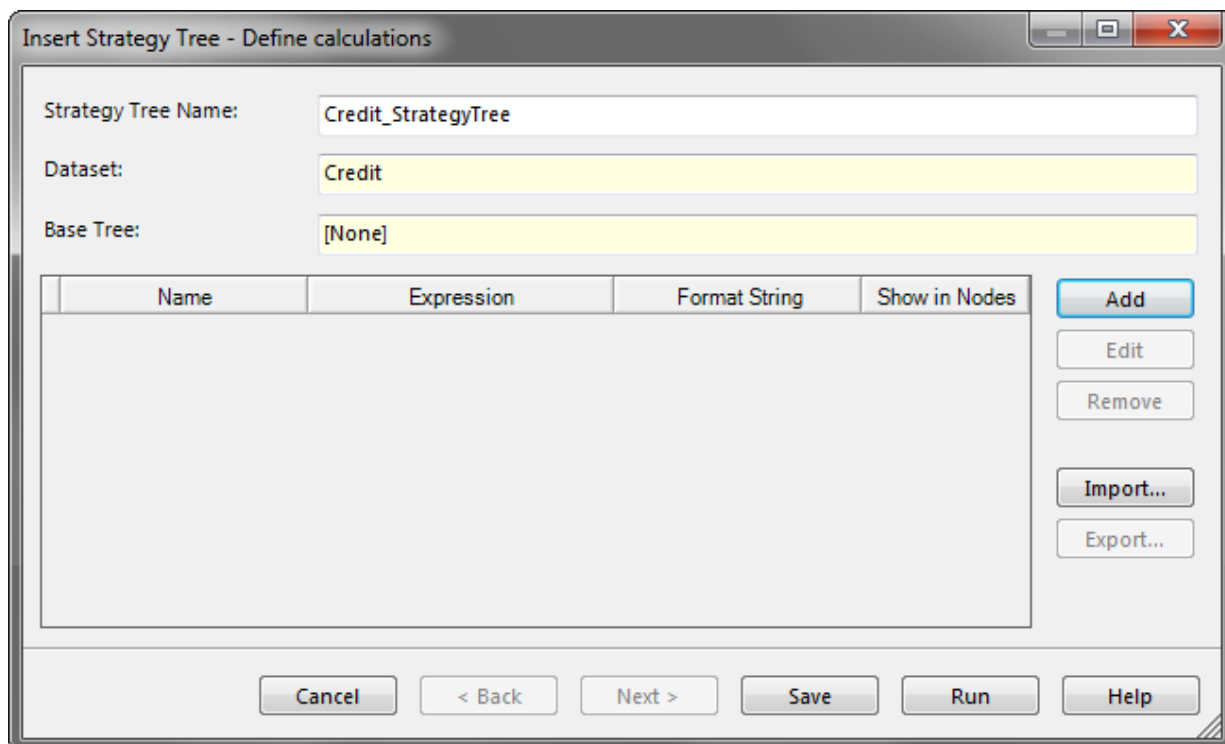


If the wizard does not open automatically, double-click on the Strategy Tree node or right-click on it and select **Modify** from the context menu.



In the Insert Strategy Tree wizard:

- Specify the name for the new tree. The name can contain spaces, but cannot contain the following characters: " \* + , . / : ; < = > ? \ |
- Add calculations or import calculations if necessary. This is optional. You may add calculations later, after the tree has been inserted.
- The Base Tree name is specified only if another Strategy Tree node is connected to the new tree node. In this case the current tree has two inputs: a dataset and another tree. All applicable training parameters and attributes of the base tree will be copied and used for the new tree..



Calculations can also be imported from another Strategy Tree or from a previously saved Expressions file.

To edit or remove a calculation, select it and click **Edit** or **Remove**. When you click Add or Edit, this opens the [Expression Editor for Node Calculations](#).

After you have finished adding or editing calculations, click **Save** to save the settings, or click **Run** to save the settings and run the process immediately. After running the operation, double-click the strategy tree node to open the interactive tree view.

See also:

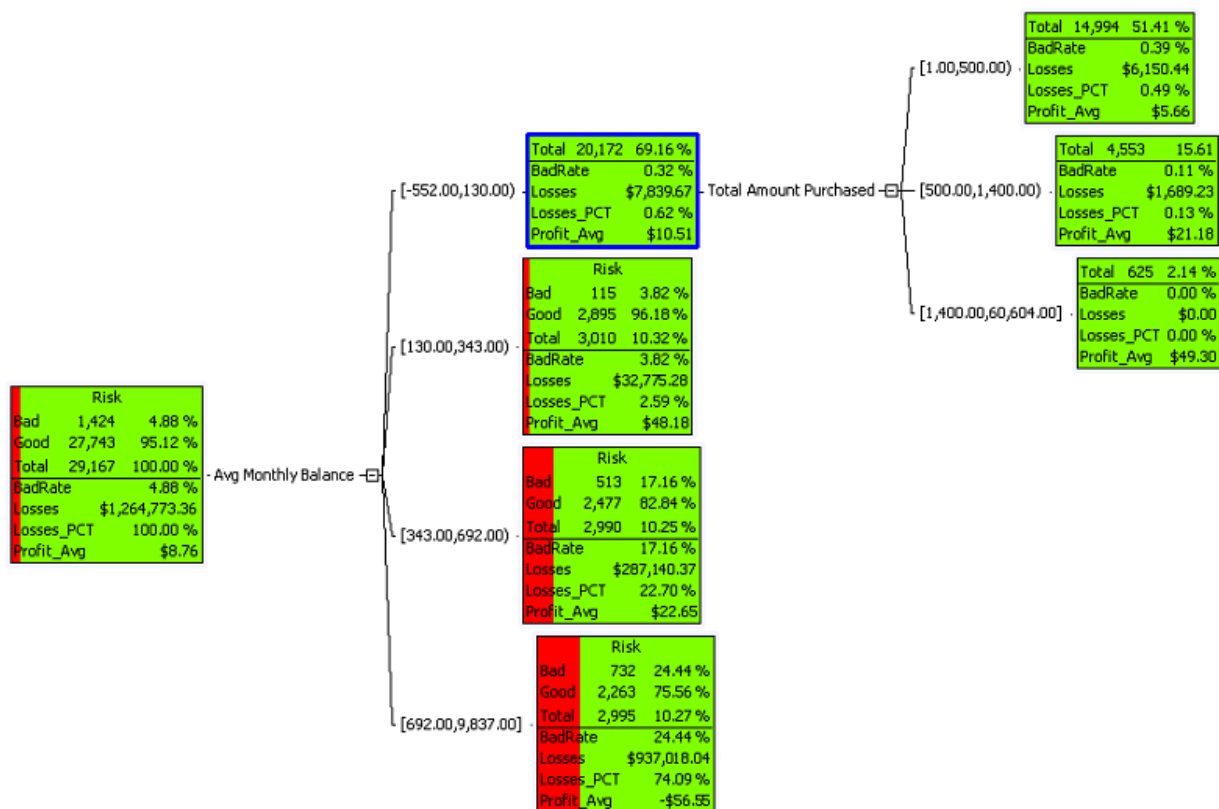
[About Node Calculations](#)

[Inserting a Node Calculation](#)

## Viewing a Strategy Tree

The **Tree** view displays the graphical representation of the tree.

Strategy Trees are oriented horizontally by default, but you can change the orientation using the **Tools | Options** command.



The Tree view for Strategy Trees has the same capabilities as that for Decision Trees. See the [Tree](#) topic for Decision Trees for details.

The **Node Data** view displays the data contained within the active tree node. The node is considered active when it is highlighted by a mouse click while in the **tree** view or **tree map** view.

|   | age | workclass        | fnlwgt | education    | education-num | marital-status |
|---|-----|------------------|--------|--------------|---------------|----------------|
| 1 | 34  | Private          |        | 10th         | 6             | Never-married  |
| 2 | 26  | Private          | 82091  | HS-grad      | 9             | Never-married  |
| 3 | 72  | ?                | 132015 | 7th-8th      | 4             | Divorced       |
| 4 | 25  | Private          | 220931 | Bachelors    | 13            | Never-married  |
| 5 | 32  | Self-emp-not-inc | 109282 | Some-college | 10            | Never-married  |
| 6 | 24  | Self-emp-not-inc | 188274 | Bachelors    | 13            | Never-married  |
| 7 | 22  | Private          | 248446 | 5th-6th      | 3             | Never-married  |
| 8 | 17  | Private          | 269430 | 10th         | 6             | Never-married  |

Tree
Tree Map
Node Data
Split Report
Node Report
Chart
Profile Chart
Parameters and Attribu...

Data is listed in a worksheet-style window where the fields appear as columns and the records appear as rows. This information cannot be changed.

The strategy tree node report provides you with information about the tree calculations in the terminal nodes of the tree.

| <input type="checkbox"/> Tagged nodes only   |               |                 |            |                              |                   |
|----------------------------------------------|---------------|-----------------|------------|------------------------------|-------------------|
| Node Rules                                   | Size of Group | Cumulative Size | Treatments | Bad_Rate                     | Cumulative Node % |
| ((Initial Score]<300) AND ((Scorecard]<550)  | 4             | 824             | 5.65       | Cancel Card                  |                   |
| ((Initial Score]<300) AND ((Scorecard]>=550) | 5             | 2358            | 21.82      | Accept Increase If Requested |                   |
| ((Initial Score]>=300) AND ((Scorecard]<500) | 6             | 187             | 23.1       | Cancel Card                  |                   |

| Treatment                    | Row Count | % of Rows | Description | Cost | Revenue |
|------------------------------|-----------|-----------|-------------|------|---------|
| Cancel Card                  | 2236      | 15.33     | N/A         | N/A  | N/A     |
| Accept Increase If Requested | 9686      | 66.42     | N/A         | N/A  | N/A     |
| Auto Increase Line           | 1688      | 11.58     | N/A         | N/A  | N/A     |
| Refer                        | 973       | 6.67      | N/A         | N/A  | N/A     |

Tree | Tree Map | Node Data | **Node Report** | Chart | Profile Chart | Parameters and Attributes | Saved Charts

**Node ID**

The node ID. By default this is not displayed and must be turned on in the Options command.

**Node Rules**

The rules for the terminal node in the tree.

**Size of group**

The number of observations in a given terminal node.

**Cumulative Size**

The cumulative number of observations up to and including this node.

**Treatment**

The treatment that is assigned to the given node. You can highlight one or more rows and assign them treatments by right clicking or Treatments command.

**Tree Calculation Values**

Each of the tree calculations is displayed in the node report. The title of the column matches the title of the label when the expression was defined. You can sort the node report by clicking on calculation headers.

**Cumulative Calculation Values**

If you turn this option on (it is off by default), then the calculation values is cumulatively summed. This is useful for selecting cut off

|                                |                                                                                                                                                                                                                                                                                |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                | points and managing resources. The amounts are accumulated by the current sort order.                                                                                                                                                                                          |
| <b>Node Percent</b>            | If you turn this option on (it is off by default), then node calculation is displayed a percentage of the sum node calculation for all nodes. This will only make sense in some circumstances, namely when the calculation is a SUM or COUNT.                                  |
| <b>Cumulative Node Percent</b> | If you turn this option on (it is off by default), then node calculation is displayed a cumulative percentage of the sum node calculation for all nodes. As with Node Percent, This will only make sense in some circumstances, namely when the calculation is a SUM or COUNT. |

The report can be sorted on a particular column. For example, you want to see all the nodes that have a treatment.

To sort the columns, click on the column heading.

## Treatment Summary

There is a summary of the treatment nodes located at the bottom of the report. This gives you a quick report on the total number of records that have been labeled with a given treatment.

The **chart** helps you to visualize the distributions of the independent variables within a node of a tree model

The first time the **chart** is viewed for a tree, the **Add Chart** wizard appears. You can start the **Add Chart** wizard manually:

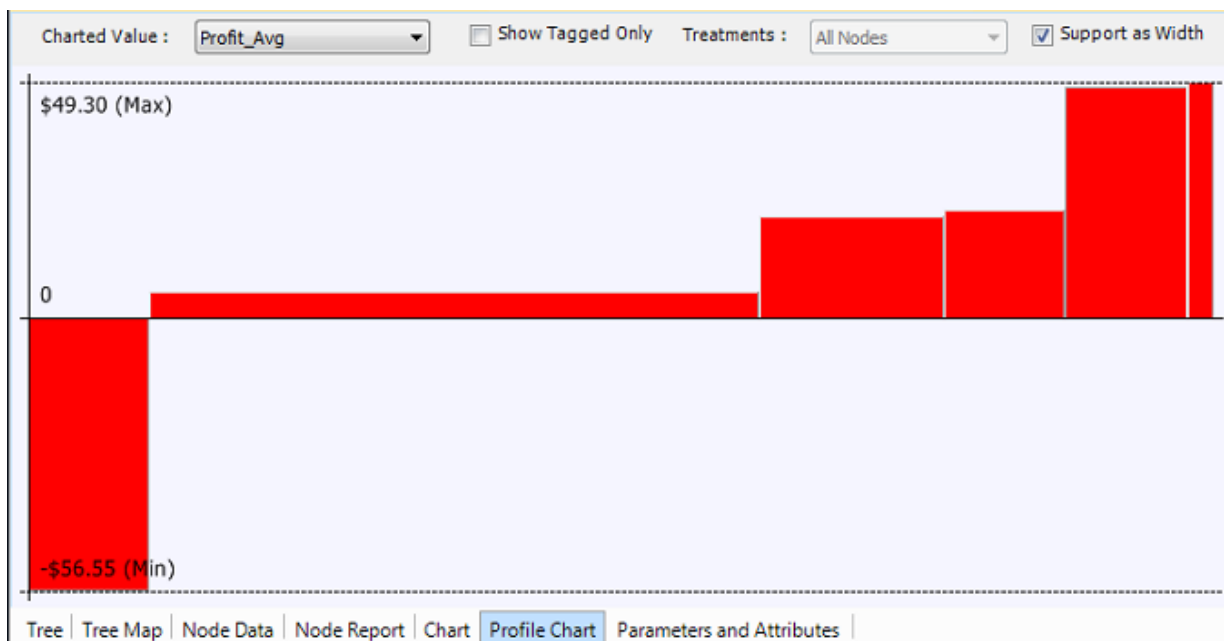
### From the menu:

Tools | Add Chart...

## Using the Tree Chart

The tree chart provides useful information while traversing the tree. Splitting the screen and docking the dynamic chart with the tree view allows you to traverse the tree and immediately see the distributions of the specified variable displayed in the chart.

The **Profile Chart** view displays each terminal node in tree. Its purpose is to display the tree in a single chart.



Before the profile chart can be displayed the tree must be grown at least one level and you have at least one tree calculation.

Clicking on a bar will cause the tree to navigate to that node. You can also right click to access commands such as assigning treatments and tagging nodes.

#### **Charted Value**

The charted value presents a list of each of the tree calculations. Choose the value that you would like charted.

#### **Show Tagged**

If this is checked, only nodes that are tagged are displayed.

#### **Treatments by Default**

By default, all terminal nodes are displayed. Changing the treatments option allows you to plot only nodes that are assigned to a specific treatment.

#### **Support as Width**

When checked, the width of the bar represents the number of observations in the node. If unchecked, each bar is the same width regardless of the number of rows.

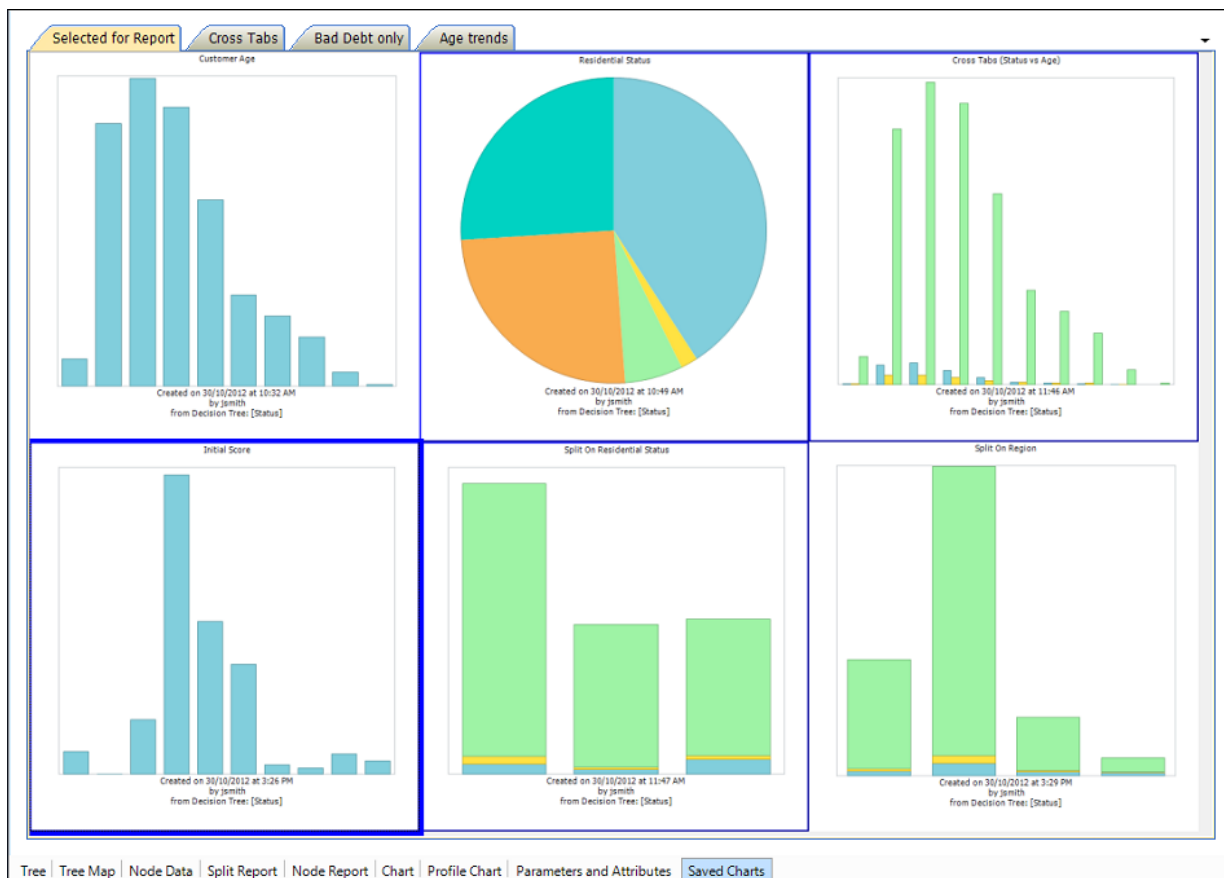
If the tree calculations are not up-to-date, then they will not be charted. You may need to use the Recalculate command for the profile chart to plot all of the terminal nodes.

The Saved Charts tab is available for datasets, decision trees, and strategy trees. This tab represents the library of charts you saved in other views of the object it belongs to.

- The Saved Charts tab of a dataset contains charts saved in the Dataset Chart, Segment Viewer, Cross Tabs, and Correlations views of the dataset.

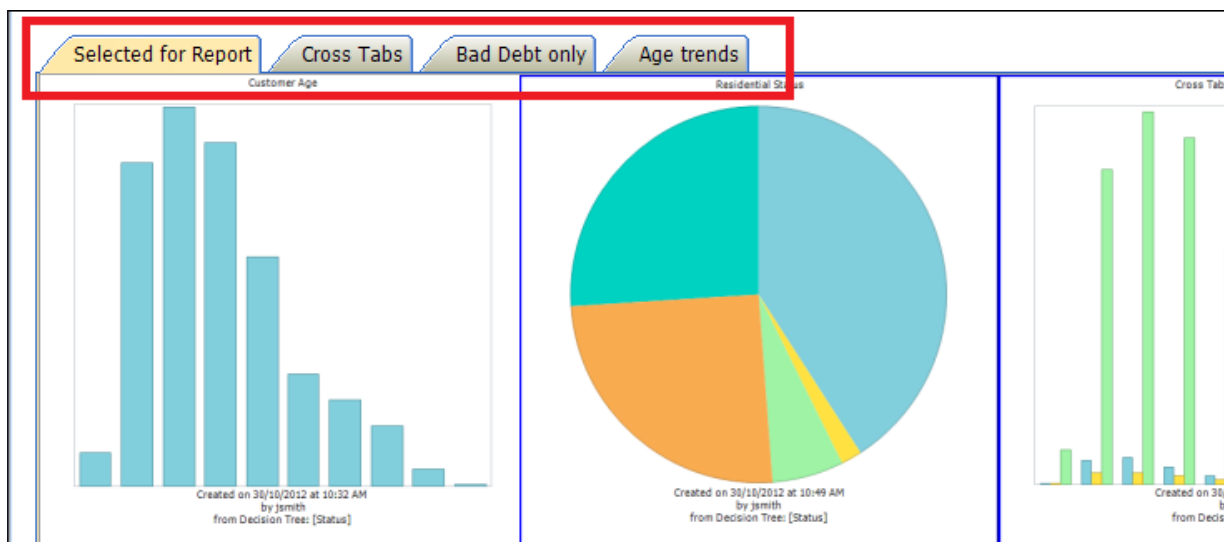
- The Saved Charts tab of a tree contains node charts saved in the Chart view of the tree.

In the Saved Charts tab you can view, organize, and print the charts, or copy them to Microsoft Office documents for reporting.



Charts are organized in groups. Each group is represented by a separate tab at the top of the chart grid. The tabs are highlighted by the red frame in the example screenshot below. Click on the tab names to switch between the chart groups.



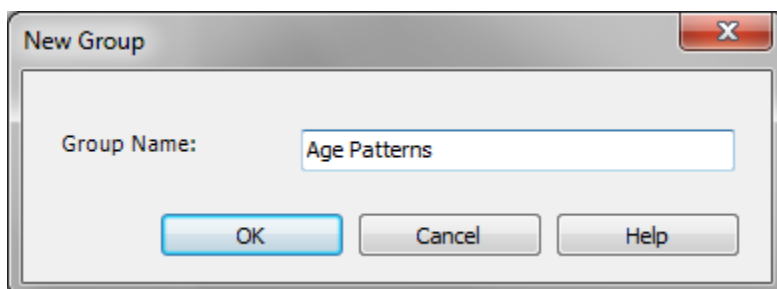


To view any chart in detail in a separate window, double-click on it and resize its window as necessary. In this window you can view the chart details, copy or print it, etc. For more information, see the topic [Single Chart View](#).

## Chart Groups

You can create new chart groups and distribute the charts in these groups.

To create a new chart group, select the **New Group** command from the **Groups** menu, or right-click on the group tab name and select **New** from the context menu. Then specify the desired name in the dialog:



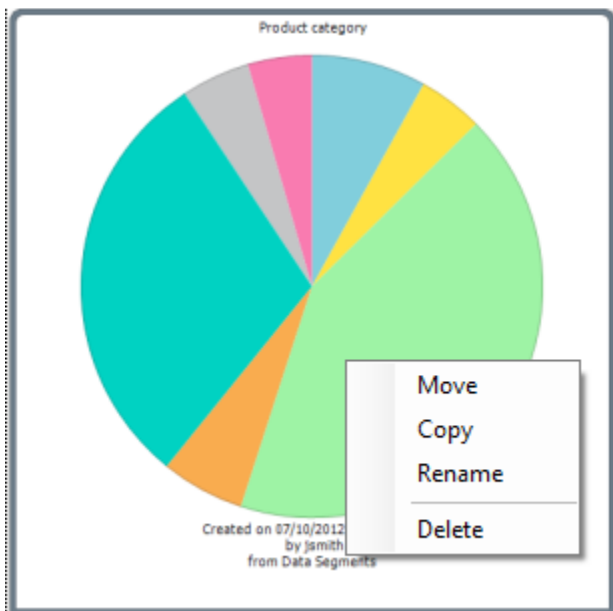
A new blank chart group with the new name will be created.

To rename an existing chart group, right-click on the group tab name and select the **Rename** command from the context menu. Enter the new name and click OK.

## Operations with Charts

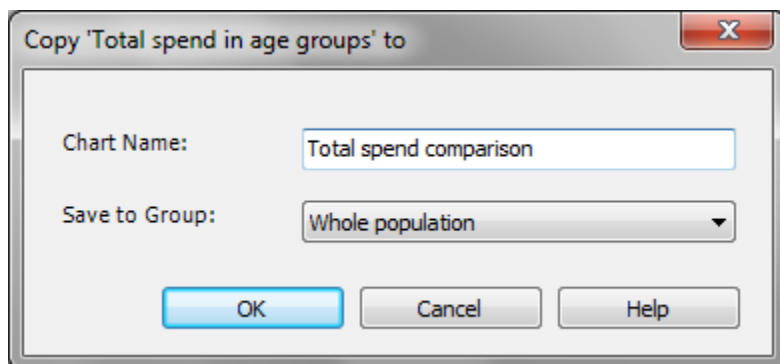
To change the position of any chart within its group, drag and drop it to the desired position.

To delete, copy, rename, or move a chart to a different group, right-click on the chart and select the appropriate command from the context menu.



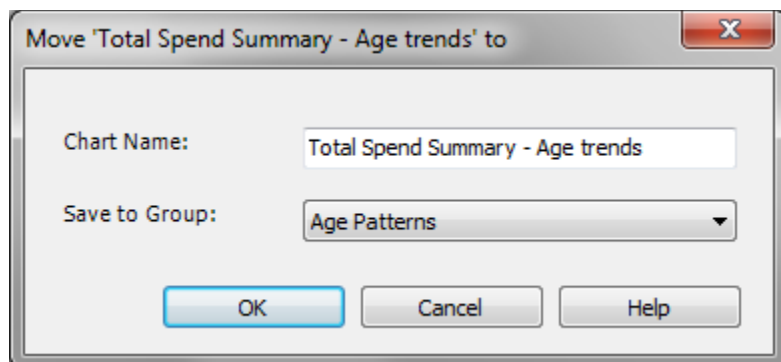
Page | 406

- To delete a chart, select **Delete** from the context menu.
- To rename a chart, select **Rename** from the context menu, specify the new name for the chart, and click OK.
- Charts can be copied within the same chart group or to a different chart group. To copy a chart, right-click on it and select **Copy** from the context menu. Specify the name for the new chart and select the destination group. The current group is the default destination.



- There are two ways to move a chart to another group:
  - a) drag and drop the chart to another group tab, or
  - b) right-click on the chart and select **Move** from the context menu.

In the Move dialog, specify the new name for the chart and select the desired destination group from the Save to Group list



### See also:

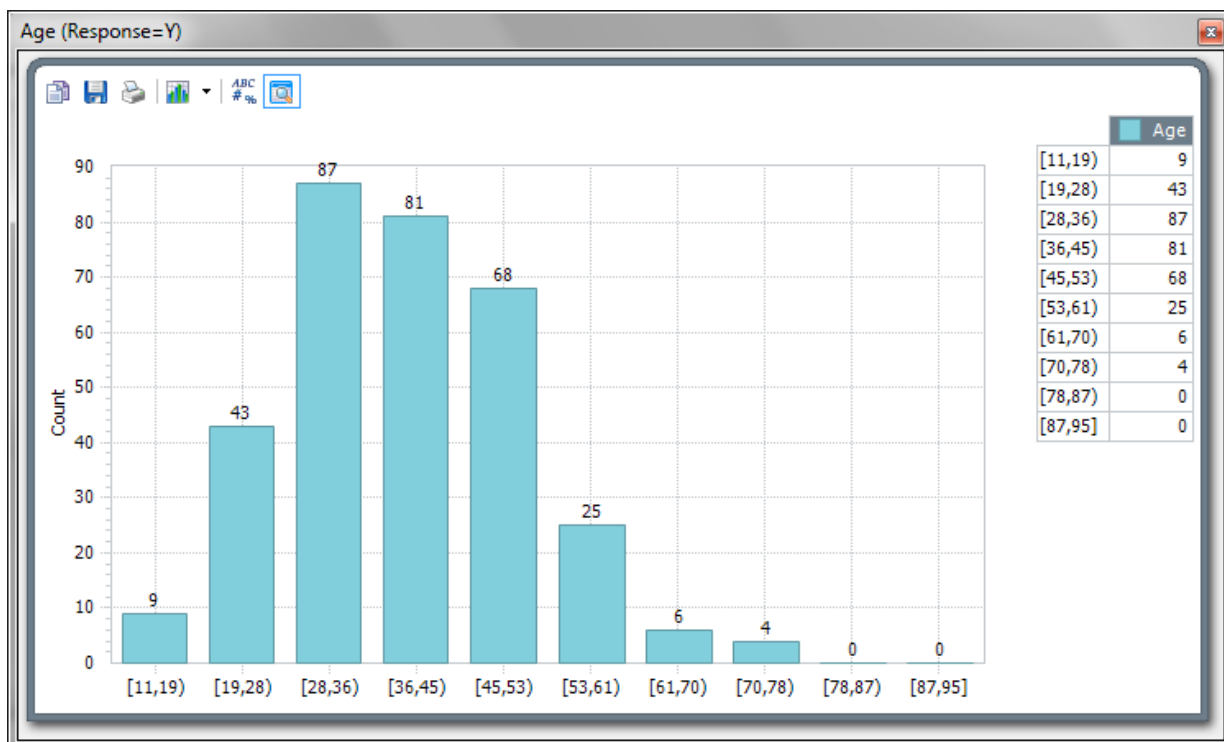
[Single Chart View](#)

## Single Chart View

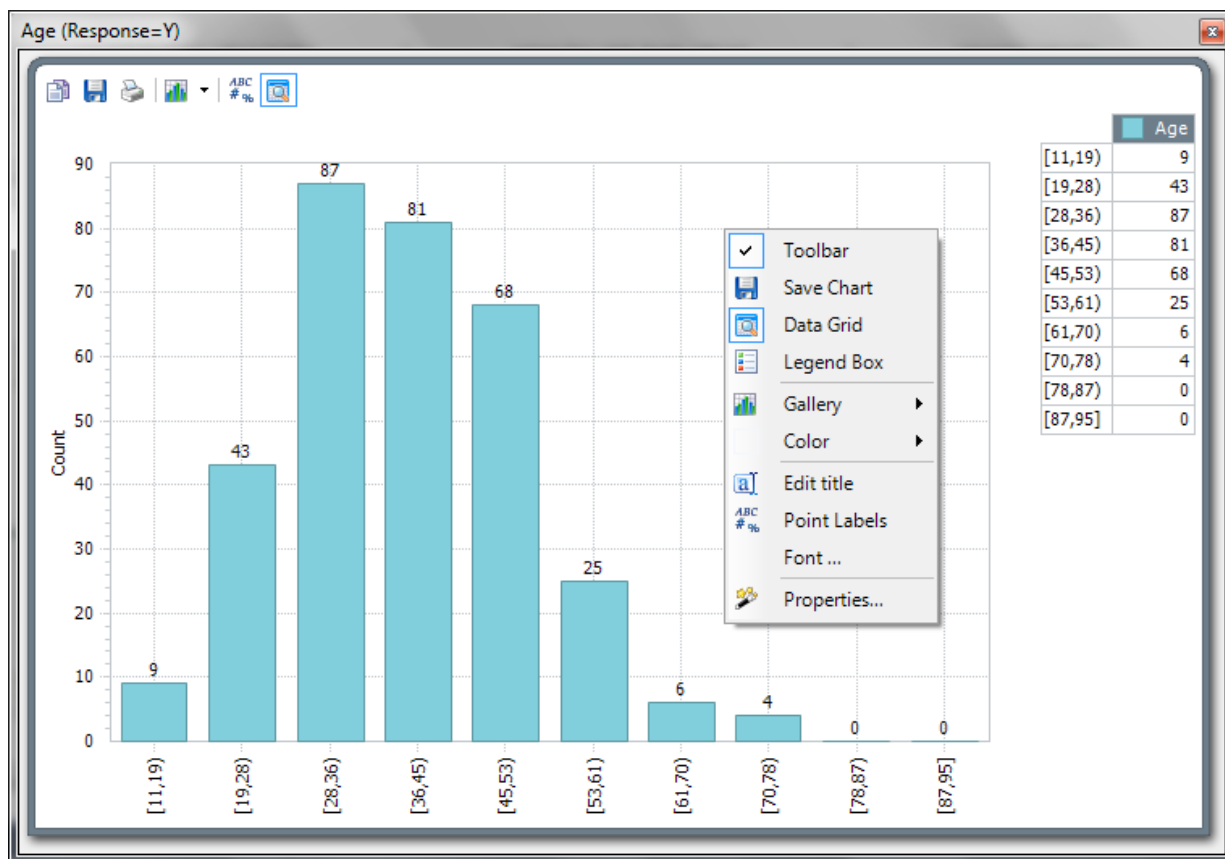
Several views in the application contain multiple charts: [Segment Viewer](#) in datasets, [Chart](#) in trees, and [Saved Charts](#) in both datasets and trees. In any of these views, you can open each individual chart in its own separate window, where you can view its details, copy or print it, etc.

Double-click on the desired chart to view it in a separate window. The window provides a detailed view of the chart and shows the data grid representing the chart data. You can resize the window by clicking and dragging on its corners or borders.

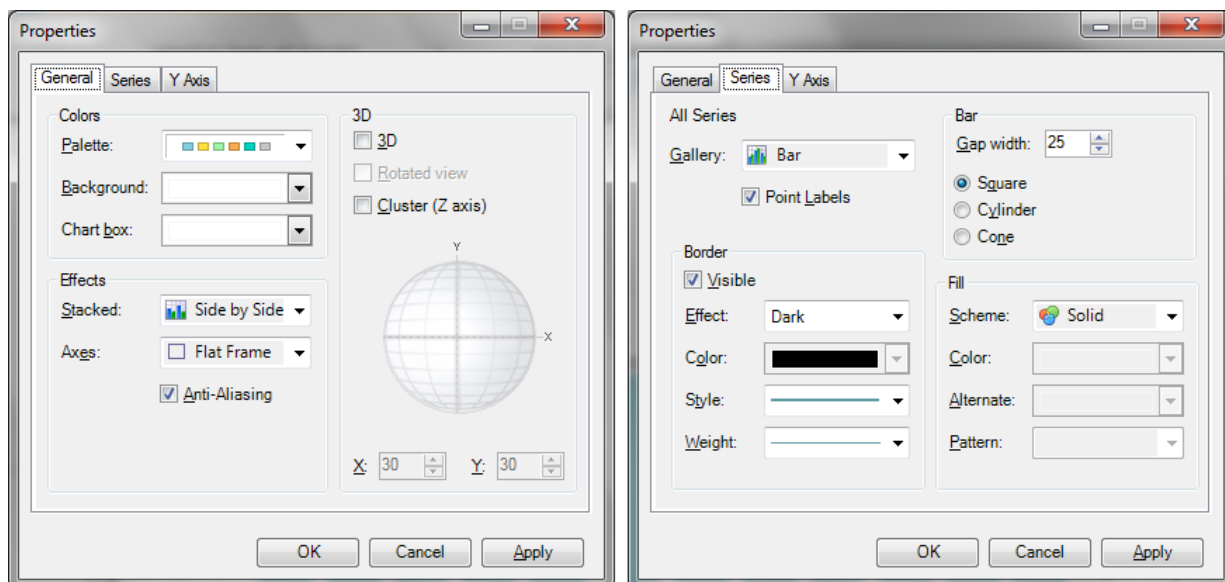
Here is an example of a chart from Segment Viewer opened in its own window.



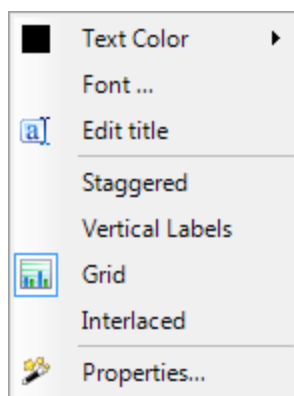
To change the display properties of the chart, use the chart toolbar or right-click anywhere on the chart background. The following context menu appears when right-clicking on the chart background. For example, select Point Labels to show or hide the values on top of the bars; select Data Grid to show or hide the table with the chart data.



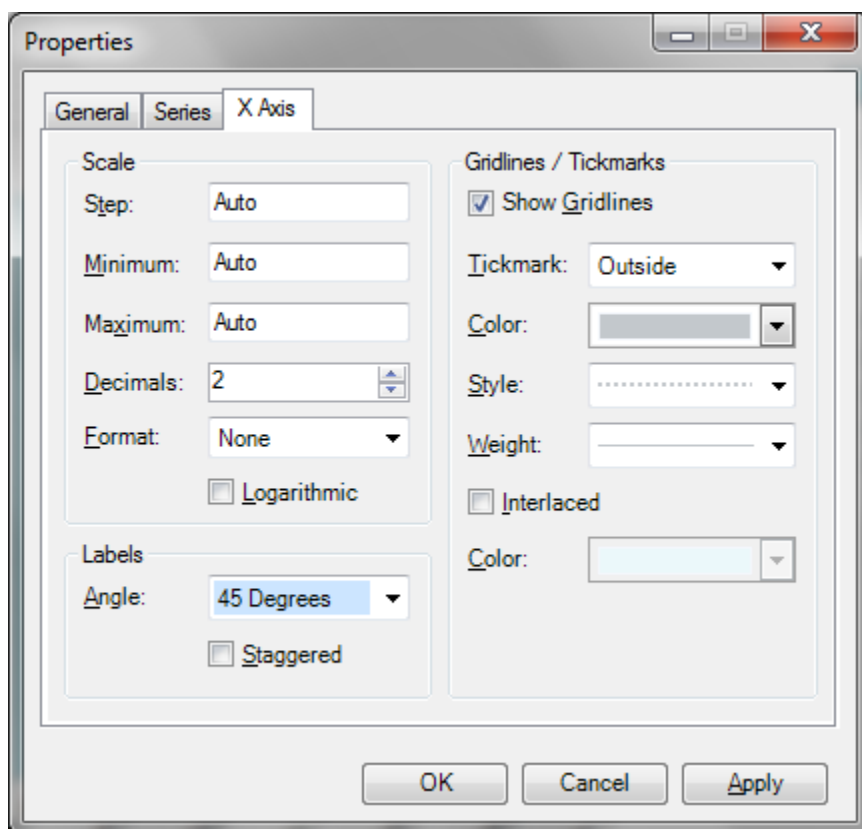
To change advanced properties of the chart, select **Properties** from the context menu. In the **Properties** dialog, you can choose the color palette, 3D or 2D view, effects, properties of the axis labels, show or hide grid lines, etc.



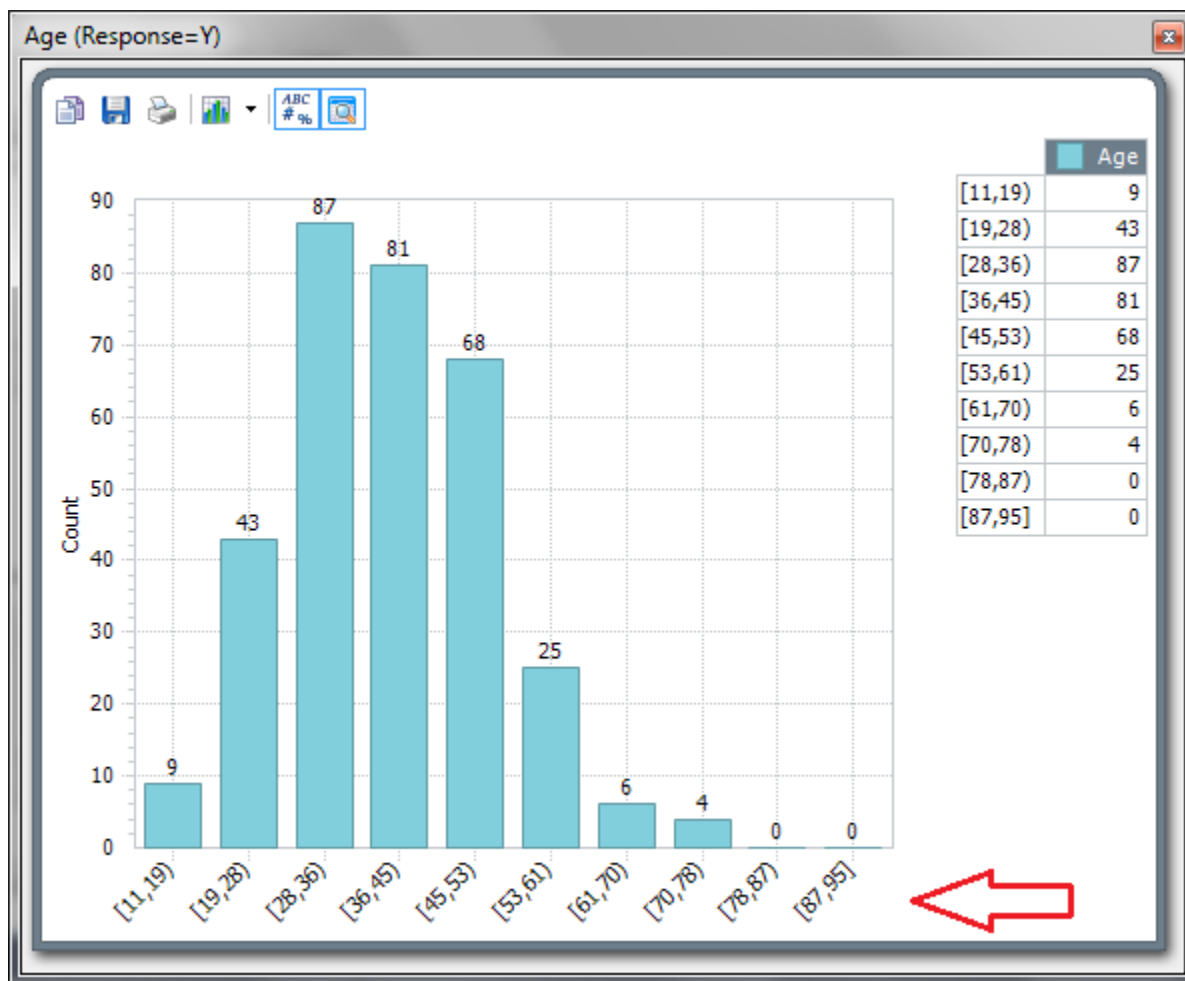
To change the orientation of the axis labels, right-click on any label of the desired axis (vertical or horizontal). The context menu will look like this:



Select *Staggered* or *Vertical Labels* if desired. If you want to place the labels at a 45-degree angle, select *Properties* and set *Angle* to 45 Degrees.



As a result, the X-axis labels are displayed as follows:



## Chart Toolbar

Depending on the charts' parent view - Segment Viewer, Cross Tabs, Chart (in trees), etc., - its toolbar at the top left corner of the chart window can have one of the following two forms:

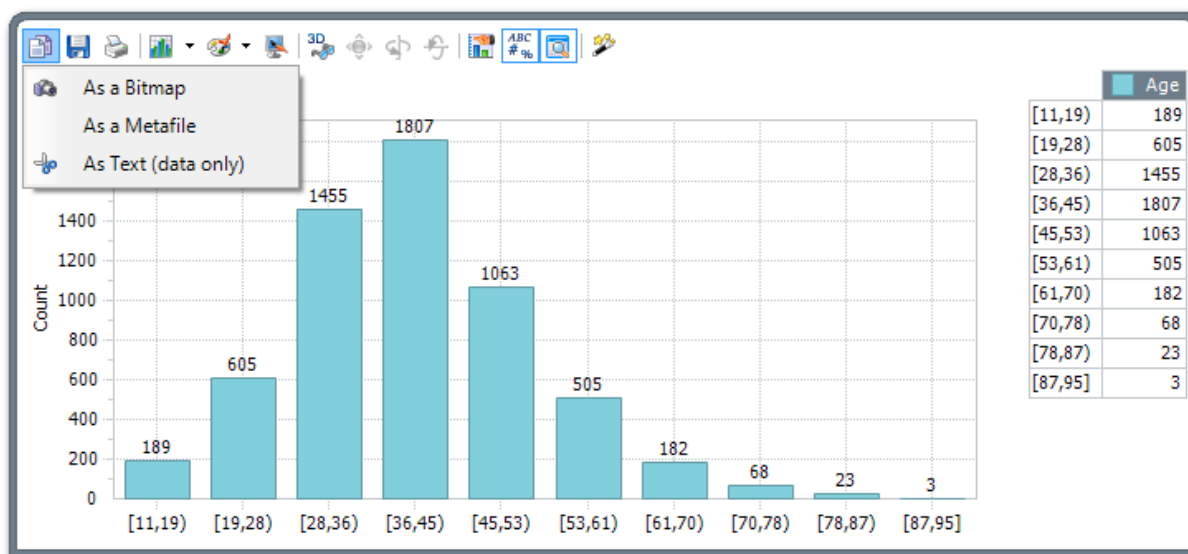


All icons in the Chart toolbar except the *Save*, *Copy*, and *Print* icons are shortcuts to the corresponding options and commands in the Properties dialog described above.

To copy a chart, click on the *Copy* icon in its toolbar, as shown in the screenshot below. Select *As a Bitmap* or *As a Metafile* from the context menu if you want to copy the chart as an image. Select



As *Text (data only)* if you want to copy the underlying data, which is displayed in the data grid on the right-hand side. Then paste the copied image or text from the clipboard into the desired application.



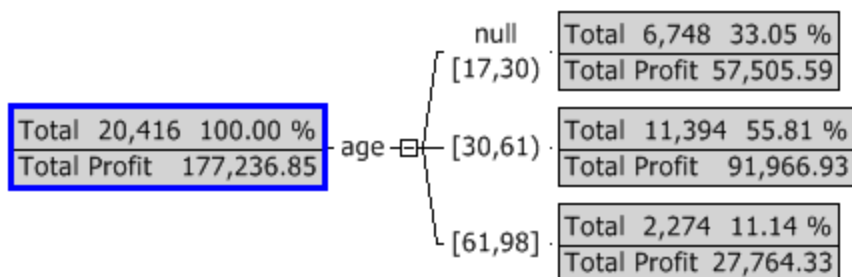
Page | 413

If you are viewing a chart from the Segment Viewer or Node Chart, you can save it in the Chart Library by clicking on the Save button .

**See also:** [Saved Charts](#)

## Adding Node Calculations

Node calculations are expressions that are calculated and applied to each node in the strategy tree. Expressions can include various functions, operators, constants, and field names as variables.



In this example, **Total Profit** is a calculation from the expression **SUM[Profit]** where **[Profit]** is a field in the dataset that the strategy tree is based on.

Node calculations can be added in two ways:

- in the **Insert Strategy Tree** wizard
- through the **Tools | Calculations...** menu command
- by right-clicking any node and selecting **Calculations** from the context menu

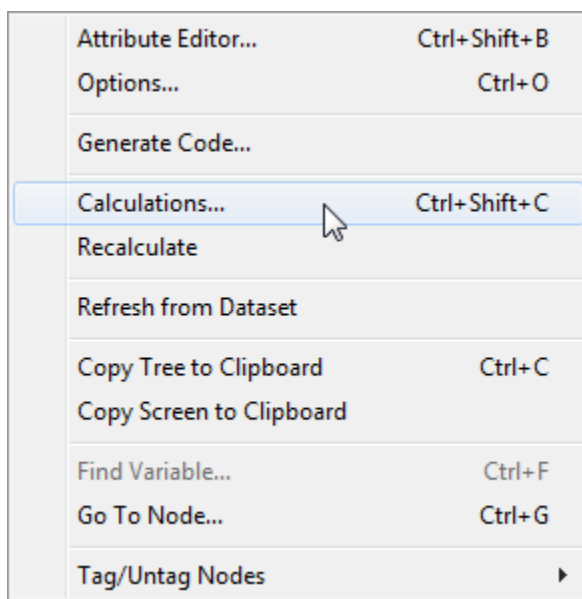
When you add or edit node calculations, they are added or updated for all nodes of the tree. After tree growing operations such as Find Split, Force Split, and others, the existing calculations are automatically added to the newly created nodes.

Page | 414

Calculations inserted for a Strategy Tree are calculated and displayed for every tree node.

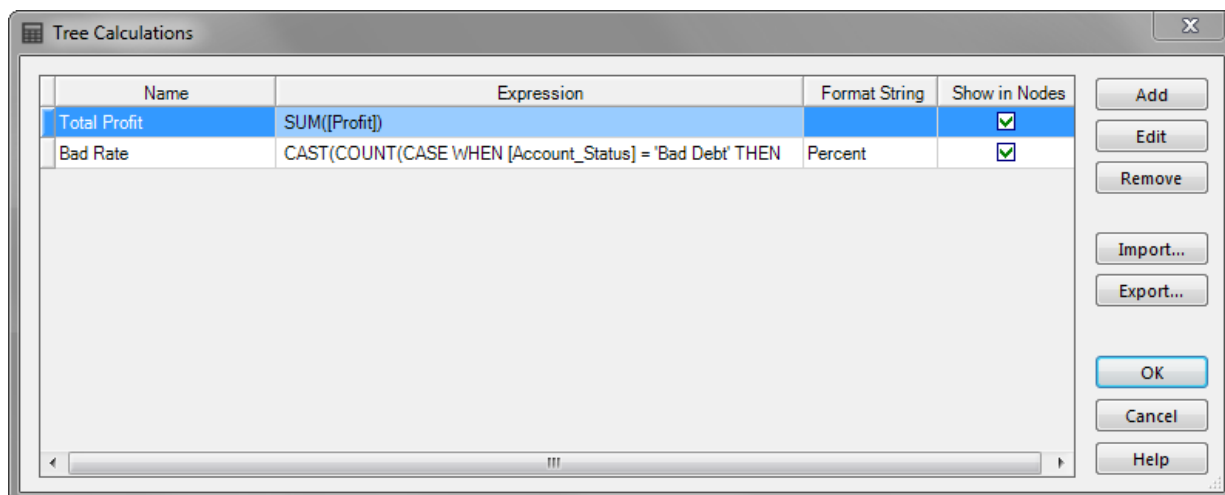
To insert a calculation, open the Strategy Tree (double-click on the Strategy Tree node on the workflow canvas or in the Project Explorer panel) and use one of the following ways:

- Right-click anywhere on the Strategy Tree and select **Calculations** from the context menu:



- Click the icon  in the toolbar
- Select the **Calculations** command from the **Tools** menu.
- Use the keyboard shortcut CTRL + SHIFT + C.

The Tree Calculations window opens. It displays existing node calculations (if any) and provides a set of tools for adding and managing node calculations.



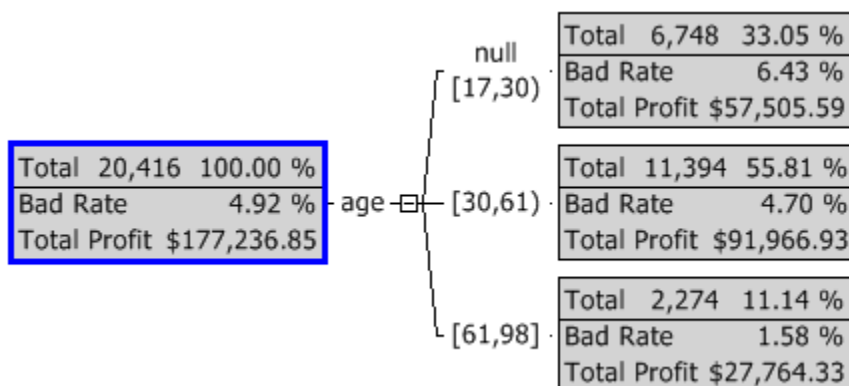
**Add** Add a calculation to the node. This button calls the [expression editor](#) to help in creating the calculations.

**Edit** Edit an existing node calculation

**Remove** Remove a node calculation

In this example, two calculations already exist in the strategy tree. See Figure 1.

Figure 1 - Node Calculation for strategy tree

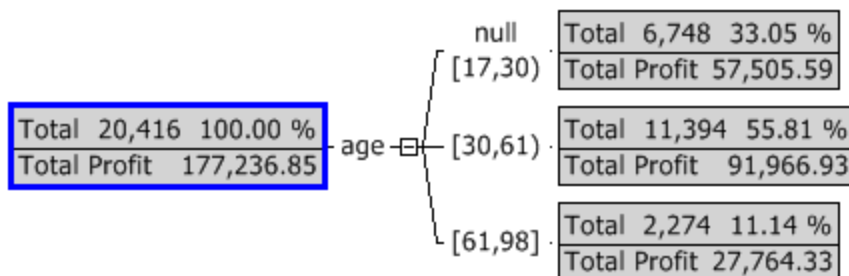


Click OK to calculate the expressions and apply them to the nodes in the strategy tree.

## Expression Editor

Node calculations are SQL expressions that are calculated and applied to each node in the strategy tree. Expressions can include various functions, operators, constants, and field names as variables.

Page | 416



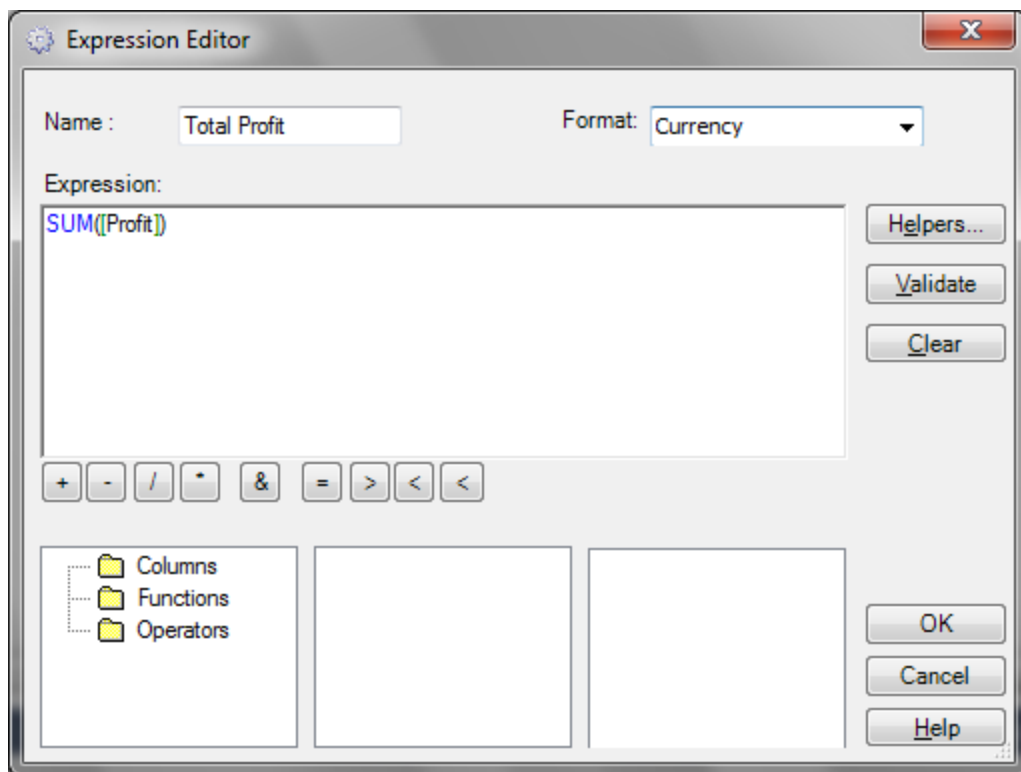
In this example, Total Profit is a calculation from the expression SUM[Profit] where [Profit] is a field in the dataset that the strategy tree is based on.

Node calculations can be added in the following ways:

- in the Insert Strategy Tree Wizard
- using the **Tools | Calculations** menu command
- by right-clicking any node and selecting **Calculations** from the context menu

When you add or edit node calculations, they are added or updated for all nodes of the tree. After tree growing operations such as Find Split, Force Split, and others, the existing calculations are automatically added to the newly created nodes.

To add node calculations after the strategy tree is inserted, right-click any node and select **Calculations** from the context menu. Then click Add. This brings up the Expression Editor for node calculations:



The calculations created by the expressions will appear in all nodes in the strategy tree.

- Helpers**      Helpers are wizards that help you create the most common expressions for the node calculations. Each Helper is described in detail in a separate topic.
- Validate**      Allows you to validate the syntax of the expression entered in the expression box.
- Clear**          Clears the expression from the expression box.

The expression syntax for node calculations is very flexible. Note that you can also use *node-specific* syntax to refer to the value of any expression at the root node or parent node.

To refer to the root or parent node, prefix an identifier of an existing expression with **[\$root\$]** or **[\$parent\$]**, respectively. For example:

**A = sum([field\_name1])**

**ratio = [A]/[\$parent\$].[A]**

**global = [A]/[\$root\$].[A]**

At any node, "ratio" is the value of expression 'A' at this node divided by the value of A at its parent node, while "global" divides A by the value of A at the root node of the tree.

Also possible is [\$self\$]. You can also chain them up like this:

**[\$parent\$].[\$parent\$].[a]**

Page | 418

which would refer to the value of 'a' in the grand-parent node.

Other examples:

a = sum([Profit])

b = a/[\$root\$].[a]

c = a/[\$parent\$].[a]

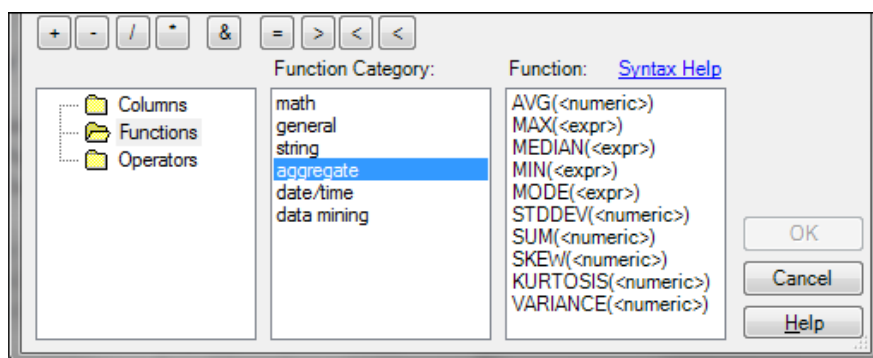
d = a/[\$parent\$].[\$parent\$].[a]

A = SUM([Amount\_Lost])

Loss\_Index=[A]/[\$root\$].[A]

Note the flexibility: you can reuse these calculations in other trees. For example, if you copy your tree to another dataset, these calculations will still be valid as it will refer to the root or parent nodes in the new tree rather than using constants from the source tree.

## Columns, Functions and Operators



The Columns, Functions, and Operators folders provide help in building your expressions. Columns will display all the columns in the dataset the tree is based on. The Functions and Operators lists provide standard functions and operators for SQL expressions.

See also:

[Columns](#)

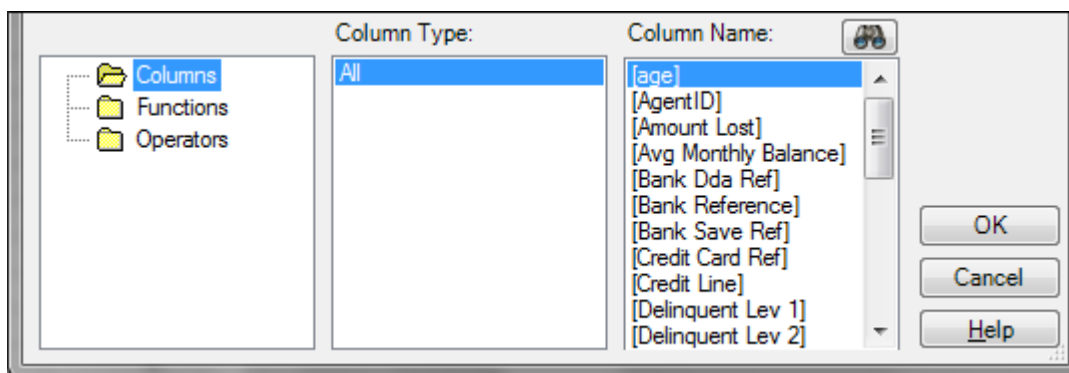
[Functions and Operators](#)

[Function Syntax](#)

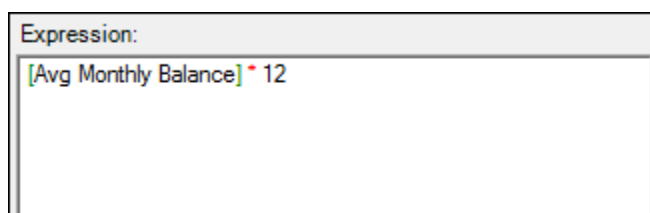
[Expression Helpers](#)

Page | 419

By highlighting the **Columns** folder, a list of columns from the dataset will appear under **Column Name**. The SQL expression must use the exact name of the column in your data.



If you want to perform a calculation using a column, double-click on the column name and it will appear in the expression box. The column can now be used as part of the calculation.

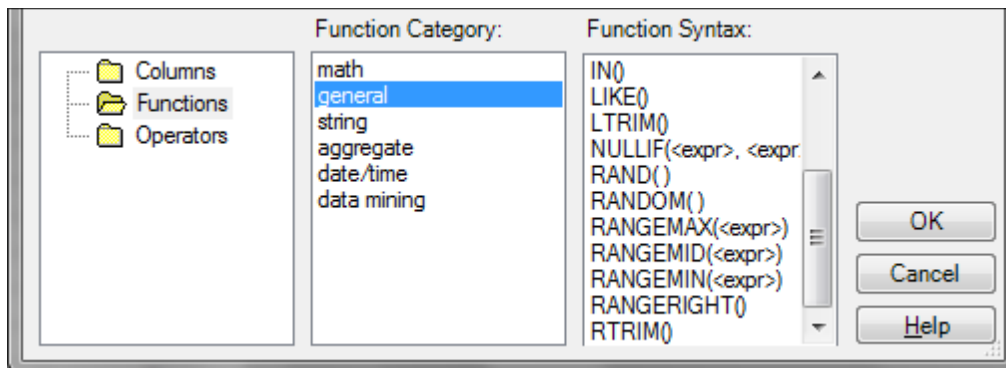


Expressions can always be typed manually into the expression box. When doing this, be aware that the column name must always be surrounded by the square "[" brackets.

## Functions

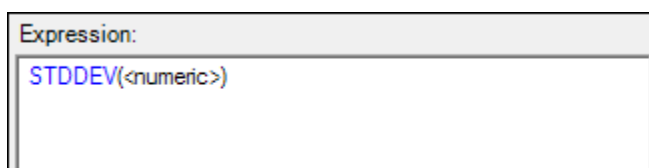
The **Functions** folder contains most commonly used mathematical, string, date/time, data mining, and other SQL functions. The **Function Syntax** box lists the functions of each category. The type

of their arguments is indicated in parentheses. For information and examples on these functions, see the [Function Syntax](#) topic.

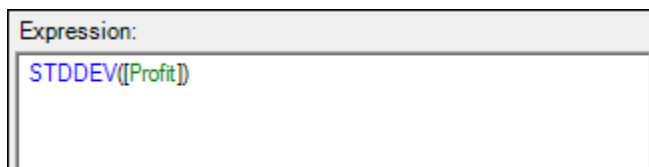


Page | 420

Double-click on a function, and it will appear in the expression box.



Replace the type in the parentheses with a value or a column name. A column name can be entered in the expression box by selecting the Columns folder and double-clicking the desired column name. The type of the entered value or column must be compatible with the type of the function argument. For example:



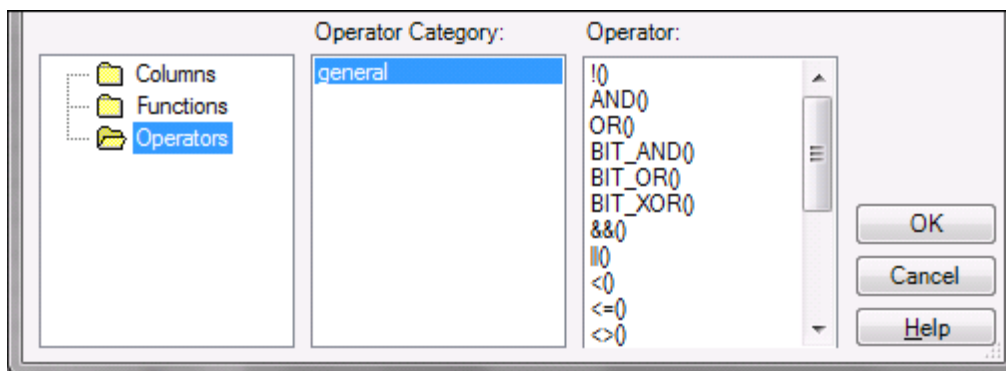
where [Profit] is a column in the dataset.

As with all expressions, these functions can be typed manually into the expression box.

## Operators

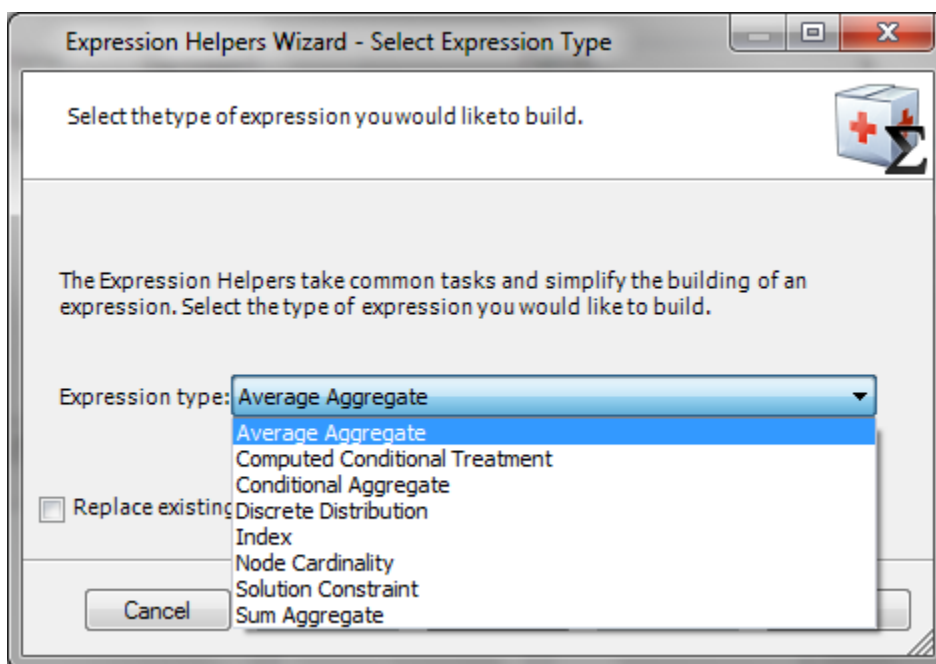
The operators list contains most common operators used in SQL Queries. They include addition, subtraction, multiplication and division operators, comparison operators (<,>,<=,>=, etc.), logical operators AND, OR, NOT, etc., and bitwise operators.





## Expression Helpers

The purpose of the helpers is to help you create most commonly used expressions using a simple wizard-like process. After you construct an expression using a helper and click Finish in the helper wizard, the expression appears in Expression Editor, so you can edit it further, if necessary, before the calculation is added.



## Average Aggregate and Sum Aggregate

The Average Aggregate and Sum Aggregate helpers allow you to create a new node calculation that is an aggregate function (sum or average) of a numeric variable. Example expressions create by these helpers are:

**AVG([Age])**

**SUM([Profit])**

where **Age** and **Profit** are variables in the parent dataset of the strategy tree.

For details, see the topic [Average and Sum Aggregate Helpers](#).

## Conditional Aggregate

A conditional aggregate allows you to display a node calculation involving aggregate functions that is based on a particular condition imposed on the nodes. For example, if you would like to see the average account balance only for the nodes where the average age is greater than 50, the expression constructed by the helper is

**CASE WHEN AVG([Age]) > 50 THEN AVG([Account Balance]) ELSE 0 END**

For details, see the topic [Conditional Aggregate](#).

## Computed Conditional Treatment

Allows you to assign any alphanumeric expression to the nodes depending on a condition you specify. The condition must include an aggregate function (SUM or AVG) and is applicable to numeric variables only.

**Note:** the result of applying this helper is not a treatment but a calculation whose result is a string value. If you want to transform this result into a treatment, use the menu command **Tools | Treatment From Calculation...** after adding the calculation.

For details, see [Computed Conditional Treatment](#).

## Index

This type of node calculation is the normalized aggregate function (sum or average) of a selected numeric variable. The normalization coefficient is the inverse of the sum (average) of the variable at the root node. Thus, the value of the expression at a given node is the sum (average) of the selected variable at this node divided by the sum (average) of the same variable at the root node. It follows from the definition that the value of the index expression at the root node is always equal to 1.

For example, suppose you have your sales data over several regions and the question is how well each region is doing relative to other regions as well as overall (i.e., you want to see only relative sales performance rather than exact dollar amounts). You have a strategy tree with the split forced on the Region variable. Using the index helper, you can represent the overall sales as 1 (shown at the root node) and the regional sales as numbers between 0 and 1 (shown in the child nodes).

For details, see [Index Helper](#).

## Discrete Distribution

Allows you to show the percentage, or count, of a discrete variable category within each node.

For details, see [Discrete Distribution](#).

## Node Cardinality

Shows the cardinality (the number of distinct values) of the specified variable within each node.

For details, see [Node Cardinality](#).

Page | 423

## Solution Constraint

The Solution Constraint Helper is provided to assist with the optimization task workflow. It allows you to assign constant values to specific nodes selected for optimization. These constant values will serve as the upper or lower limits for the values of the optimized solution in the corresponding nodes. These values can be used in the Optimization Helper.

For details, see the topics [Solution Constraint](#) and About Optimization.

## Option: Replace Existing Expressions

If you are editing an expression and this option is unchecked (which is the default), the expression created by the helper will be appended to the existing expression in the expression builder. Otherwise, the existing expression displayed in the expression builder will be overwritten.

## Average and Sum Aggregate Helpers

An aggregate function is a single-valued function that performs a computation on a set of values rather than on a single argument. The simplest examples are MIN(), MAX(), SUM(), and AVG(). When used in a node calculation of a Strategy Tree, aggregate functions perform a computation on all records within the given node.

For example, if Profit is a column in your dataset, the expressions to calculate the average and total Profit within each node is AVG([Profit]) and SUM([Profit]), respectively.

| Status2        |              |          |
|----------------|--------------|----------|
| Bad Debt       | 1,005        | 4.92 %   |
| OK             | 19,411       | 95.08 %  |
| Total          | 20,416       | 100.00 % |
| Average Profit | \$8.68       |          |
| Total Profit   | \$177,236.85 |          |

The Average Aggregate and Sum Aggregate helpers assist in creating expressions containing sum and average aggregate functions. You can edit the output of these helpers to substitute any other supported aggregate function for SUM or AVG or append custom statements.

The expression syntax is standard SQL. Column names should be enclosed in square brackets. See [Function Syntax](#) for details.

---

## Supported Aggregate Functions

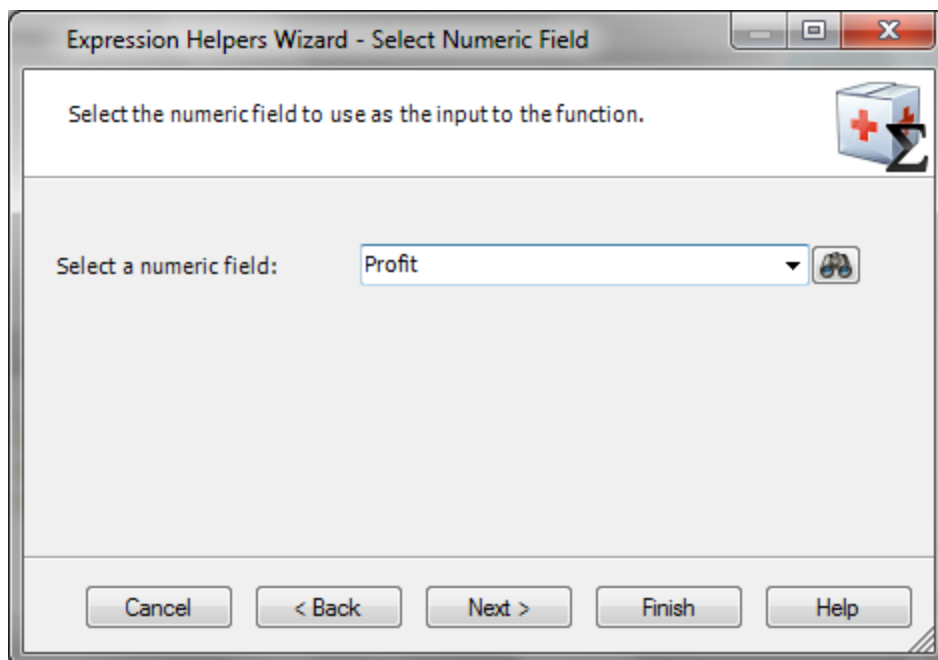
|                                    |                                                                                                                                                                     |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>AVG(&lt;numeric&gt;)</i></b> | Allows you to take the average of a group of values. For example, AVG(Age) where Age is a field in the dataset                                                      |
| <b><i>MAX(&lt;expr&gt;)</i></b>    | Gives you the maximum value of a group of values. The value can be numeric or string. If the value is of type string, the ordering is alphabetical.                 |
| <b><i>MEDIAN(&lt;expr&gt;)</i></b> | Gives you the median value that occurs within a group of values. The value can be numeric or string. If the value is of type string, the ordering is alphabetical.  |
| <b><i>MIN(&lt;expr&gt;)</i></b>    | Gives you the minimum value that occurs within a group of values. The value can be numeric or string. If the value is of type string, the ordering is alphabetical. |

---

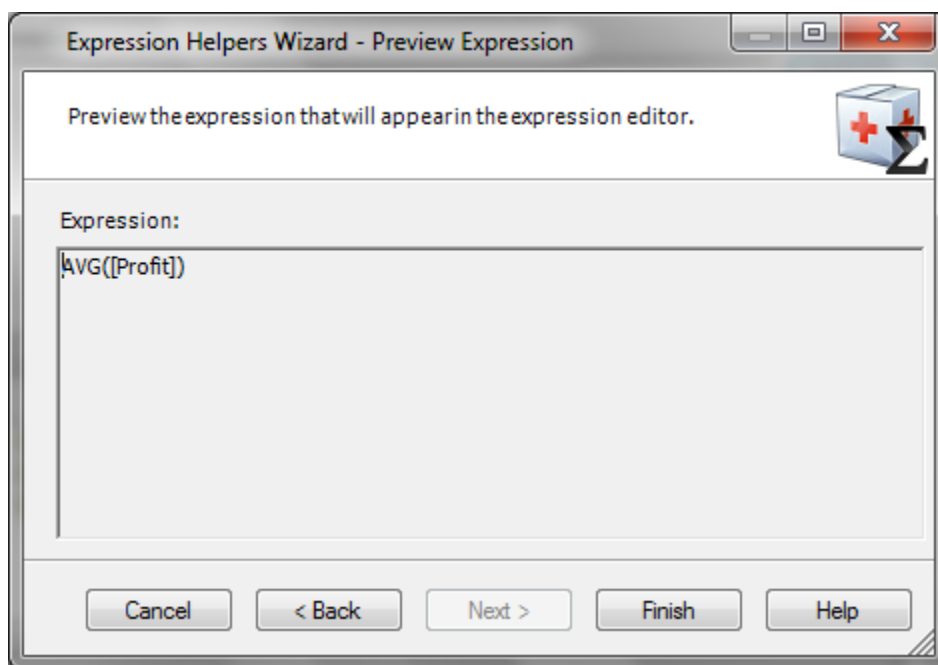
---

|                                       |                                                                                                                                                                          |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>MODE(&lt;expr&gt;)</i></b>      | Gives you the mode of the values that occurs within a group of values. The value can be numeric or string. If the value is of type string, the ordering is alphabetical. |
| <b><i>STDDEV(&lt;numeric&gt;)</i></b> | Calculates the standard deviation of a group of numeric values.                                                                                                          |
| <b><i>SUM(&lt;numeric&gt;)</i></b>    | Calculates the sum of a group of numeric values.                                                                                                                         |
| <b><i>VARIANCE(&lt;expr&gt;)</i></b>  | Calculates the variance within a group of numeric values                                                                                                                 |

After choosing either the *Average Aggregate* or *Sum Aggregate* as Expression type, click Next and select the desired numeric field as the argument for your aggregate function.



Click Next to preview the constructed expression:



Click Finish to add the expression to the Expression Editor and edit it if necessary. See the topic [Preview Expression](#) for details.

See also: [Expression Helpers: Overview](#), [Conditional Aggregate](#), [Computed Conditional Treatment](#).

## Conditional Aggregate

The Conditional Aggregate Helper allows you to add an aggregate calculation at the nodes that meet a certain condition.

Page | 427

The helper produces an expression of the form **CASE WHEN A THEN B ELSE C**, where A is a relational expression, B is a simple aggregate expression, and C is a constant. If desired, you can edit the resulting expression to replace A, B, and C with custom expressions.

Condition A can be of two types:

- Based on an aggregate function of a continuous numeric field. For example: *"If the sum of the field X at a given node is greater than R, then ..."*.

Example: **CASE WHEN SUM([Amount Lost]) > 5000 THEN ...**

- Based on the frequency of a value of a discrete field. For example: *"If the percentage of the value V of the field X at a given node is less than R, then ..."*.

Example: **CASE WHEN CAST(COUNT(CASE WHEN [Status] = 'Bad Debt' THEN 1 ELSE null END) AS DOUBLE) / CAST(COUNT(\*) AS DOUBLE) >= 0.8 THEN ...**

Note: The CAST ... AS DOUBLE operator takes care of the data type compatibility and conversion. It is inserted automatically whenever necessary.

Expression B is a simple aggregate. For example, **SUM ([Profit])** or **AVG ([Profit])**

## Examples:

1) In the case of a continuous condition field, an example of the expression *Exp1* built by the helper is

**CASE WHEN SUM([Amount Lost]) > 5000 THEN AVG([Age]) ELSE 0**

In this example, the average of Age will be displayed as the value of *Exp1* at all nodes such that the total of Amount Lost exceeds 5000. At all other nodes, the value of *Exp1* will be zero.

2) In the case of a discrete condition field, an example of the expression *Exp1* built by the helper is

**CASE WHEN CAST(COUNT(CASE WHEN [Status] = 'Bad Debt' THEN 1 ELSE null END) AS DOUBLE) / CAST(COUNT(\*) AS DOUBLE) > 0.8 THEN AVG([Age]) ELSE 0**

In this example, the average of Age will be displayed as the value of *Exp1* at all nodes such that the percentage of records with Status = 'Bad Debt' exceeds 80%. At all other nodes, the value of *Exp1* will be zero.

**Note:**

- all String and Boolean fields are discrete;
- in this helper, variables of the types Number, Integer, Time, or Timestamp whose cardinality is less than 50 are considered to be discrete.

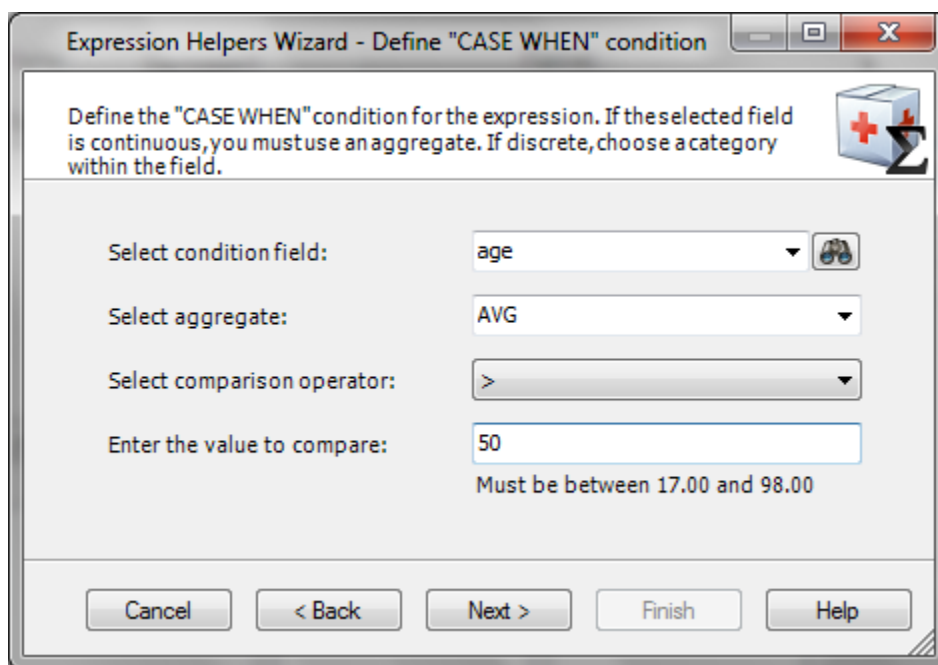
Page | 428

The wizard allows you to specify the calculation in two steps:

1. specify the condition imposed on the nodes
2. specify the calculation to be displayed depending on the condition

## The case of a continuous condition field.

In this case, the condition is based on an aggregate function of a continuous numeric field at a given node.

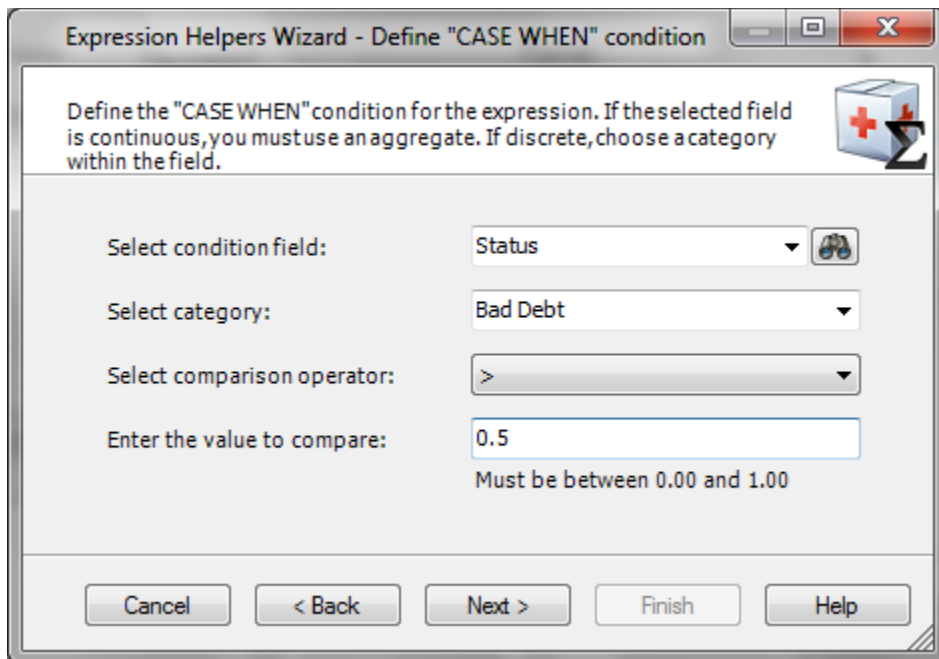


In the above example, the condition constructed at this step is "If the average age at a given node is greater than 50...", or **CASE WHEN AVG([Age]) > 50 THEN ...**

## The case of a discrete condition field.



In this case, the IF condition is based on the percentage of a value of a discrete field at a given node.



Page | 429

In the above example, the IF condition constructed at this step is *"If the percentage of Bad Debt in the field Status at a given node is greater than 50%, then ..."*.

The resulting expression will look like this:

```
CASE WHEN CAST(COUNT(CASE WHEN [Status] = 'Bad Debt' THEN 1 ELSE null END) AS DOUBLE) / CAST(COUNT(*) AS DOUBLE) > 0.8 THEN AVG([Age]) ELSE 0
```

Note: The CAST ... AS DOUBLE operators take care of the type compatibility and conversion and are inserted automatically whenever necessary.

## Summary:

|                               |                                                                                                                                                             |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Select condition field</b> | Select the field to be used for the CASE WHEN part of the expression. In the above example, the field <b>age</b> is selected.                               |
| <b>Select aggregate</b>       | Only available when the condition field is continuous. Choose the aggregate function for the condition. In the above example, it is <b>AVG[age]</b> .       |
| <b>Select category</b>        | Only enabled if the selected condition field is discrete. This list contains all categories of the field. Pick the category on which to base the condition. |

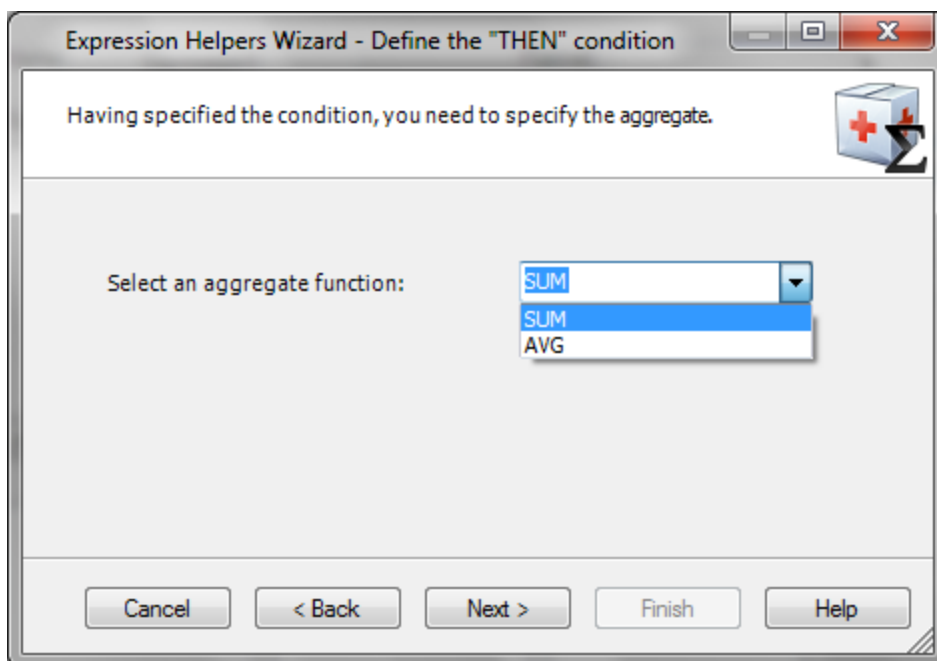
**Select comparison operator**

Choose a comparison operator for the condition.

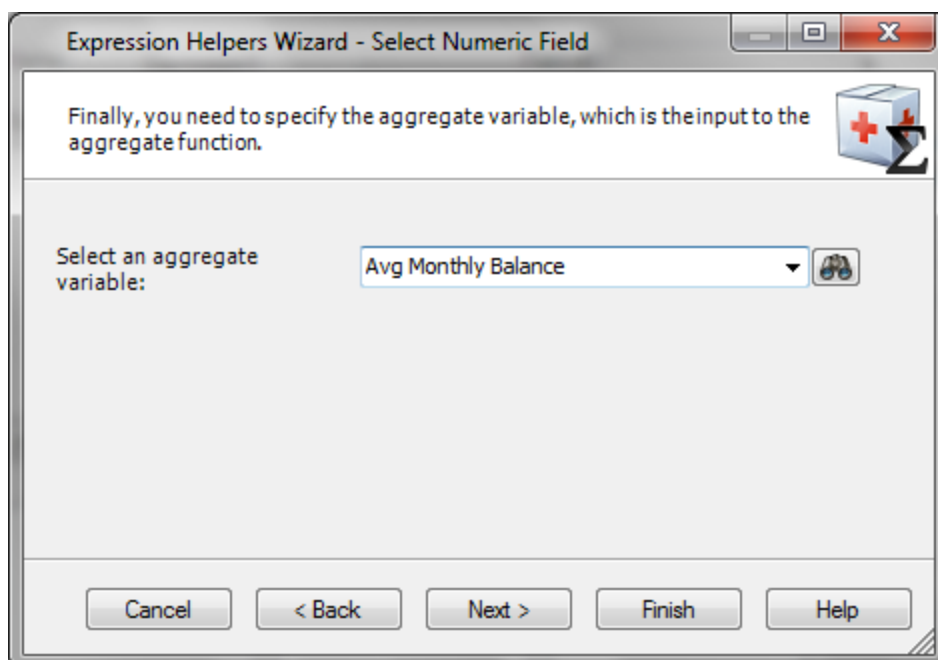
**Enter the value to compare**

Enter the threshold value for your comparison condition. If the selected field is continuous, this must be the value to compare the average or the sum of the selected variable. If the condition field is discrete, the threshold value must be a number between 0 and 1 (the percentage of the selected category divided by 100 is compared to this threshold).

Click Next. At the next step, choose the aggregate function for the calculation to be displayed if the condition specified at the previous step **holds true**.



Click Next. Select a field to be the argument for the aggregate function you selected at the preceding step:



Assuming that we chose SUM as an aggregate function and Avg Monthly Balance as the argument, we obtain the expression **CASE WHEN SUM([Amount Lost]) > 5000 THEN SUM([Avg Monthly Balance]) ELSE 0**

Click Next to preview the resulting expression and add it to the Expression Editor. See Preview Expression for details.

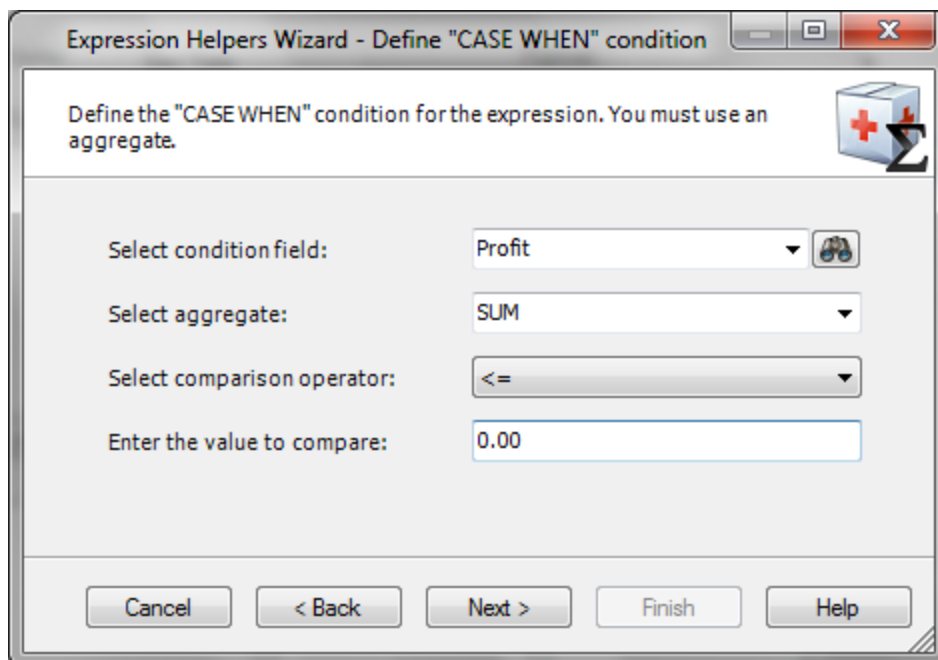
See also:

[Computed Conditional Treatment](#)

## Computed Conditional Treatment

The Computed Conditional Treatment helper allows you to assign any string-valued expression to the nodes depending on a condition you specify.

The first step of the wizard prompts for the condition field, the aggregate function to be applied to it (SUM or AVG), the comparison operator, and the constant value to compare.



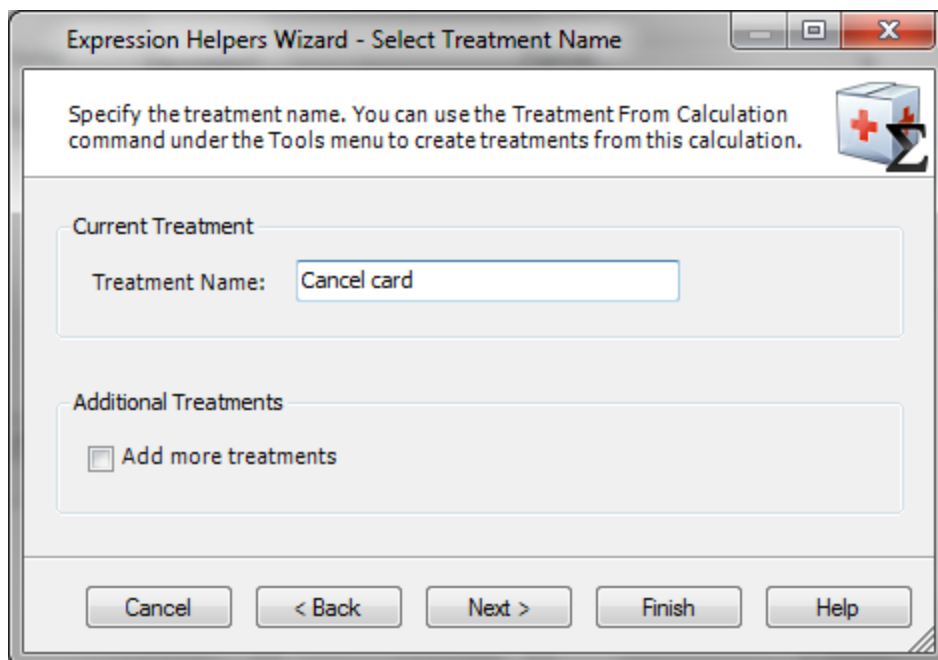
In the above example, this step of the wizard constructs the following condition:

**CASE WHEN SUM([Profit]) <= 0 THEN ...**

Click **Next >** to specify the expression text.

At the second page of the Computed Conditional Treatment Wizard, enter the string value to be displayed at the nodes that satisfy the condition specified at the previous step. For example, it can be a treatment such as "Cancel card", "Platinum Card", "Increase Credit Line by 5000", etc.

Selecting the **Add more treatments** option enables you to define more than one condition and assign text expressions to nodes based on the other conditions you specify.



Click the **Next >** button to preview the resulting expression or define a new condition (if the **Add more treatments** checkbox is checked).

**Note:** the result of applying this helper is not a treatment but a calculation whose result is a string value. If you want to transform this into a treatment, use the menu command **Tools | Treatment From Calculation...** after adding the calculation.

See also:

[Expression Helpers: Overview](#)

## Index Helper

This type of node calculation is the normalized aggregate function (sum or average) of a selected numeric variable. See the [Expression Helpers](#) topic for the summary.

The meaning of the expression being inserted is either

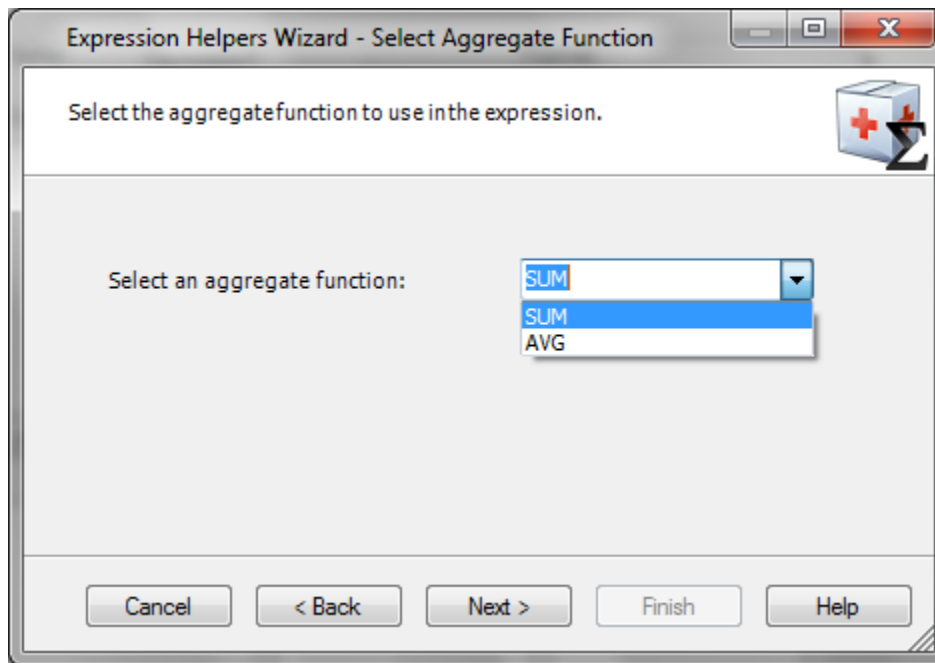
**(SUM (Variable X) over the node) / (SUM (Variable X) over the whole dataset)**

or

**(AVG (Variable X) over the node) / (AVG (Variable X) over the whole dataset)**

By the whole dataset we mean the parent dataset of the tree. The whole dataset is represented by the root node of the tree.

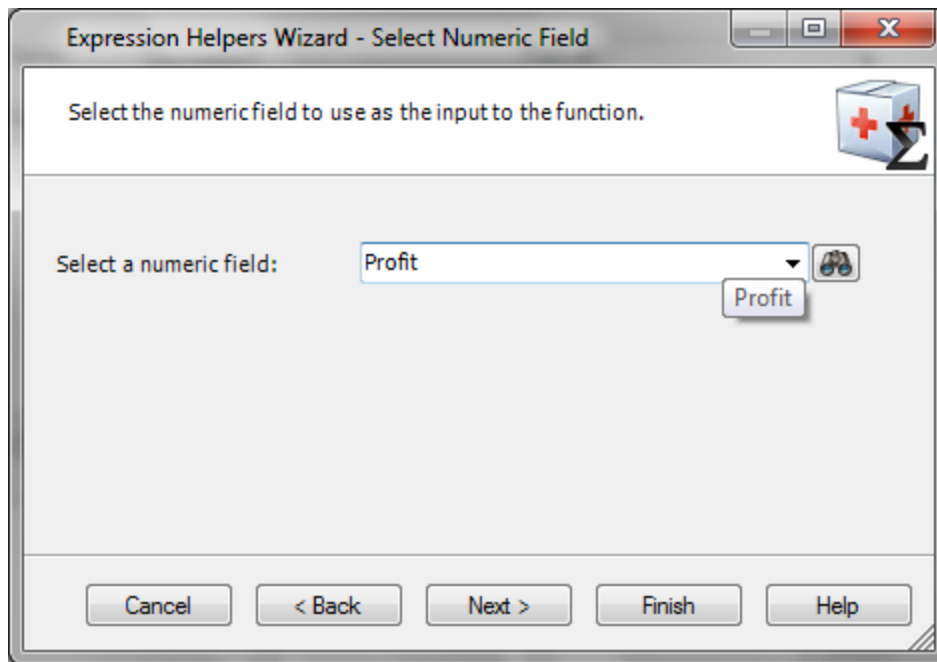
At the first step of the Index Helper wizard, select the desired aggregate function:



Page | 434

Select the desired aggregate function for the expression. Note that you can always edit the expression after it is inserted in the Expression Editor if you need to construct a more complex aggregate function.

Select a numeric field to be the argument for the aggregate function chosen at the previous step.

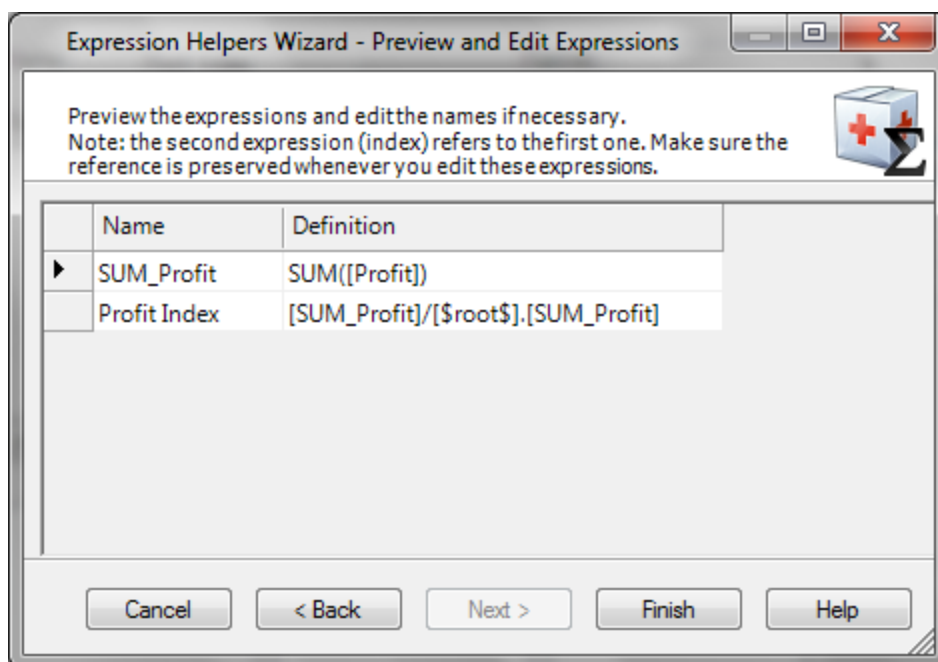


In this example, the Helper computes the ratio

**(SUM(Profit) over the node) / (SUM (Profit) over the whole dataset)**

The constructed expression is shown in the Preview window. Note that two expressions are created.

This example shows the portion of total profit represented by each node. The denominator of the Profit Index expression is the reference to the value of SUM\_Profit at the root node.



Because the expression `[$root$].[SUM_Profit]` is used, if you copy the tree to a different dataset, the Index expression will be calculated with respect to the target dataset rather than the source. Press Finish to return to Expression Editor and edit the expression there if necessary.

In general, the expression syntax allows you to refer to the value of any expression at the root node. For example, you can manually create two expressions

`A = SUM([Profit])`

`Profit_Index=[A]/[$root$].[A]`

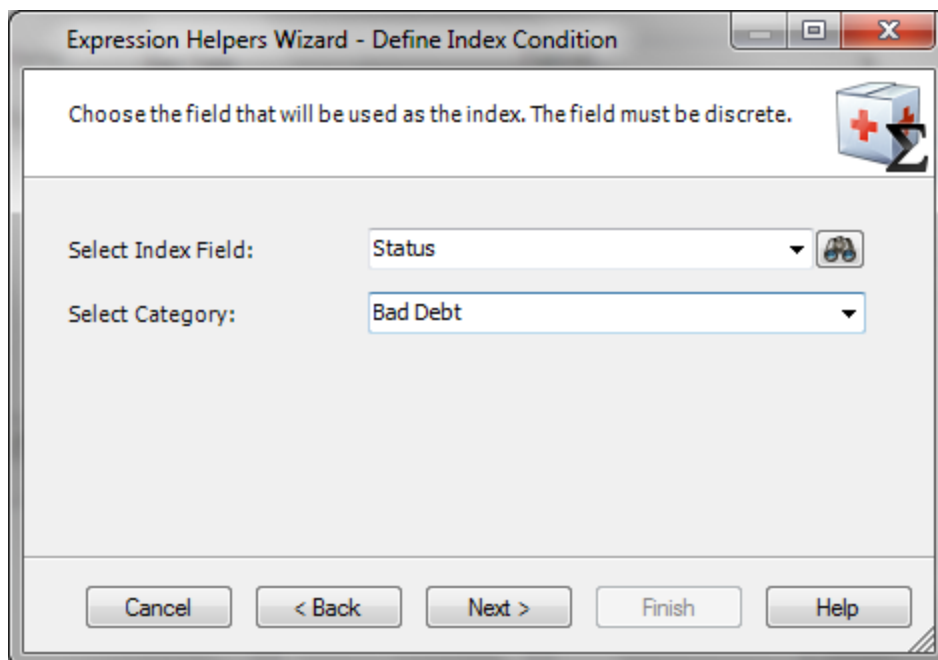
In some cases, this is a more preferable way to specify calculations because of its flexibility: you can reuse these calculations in other trees. If you copy them to another dataset they will still be valid as they will refer to the root node of the new tree.

See also: [Expression Helpers: Overview](#)

### Discrete Distribution Helper

At the first step of the Discrete Distribution Helper Wizard, select a discrete field and the category whose frequency you want to be displayed in the nodes.

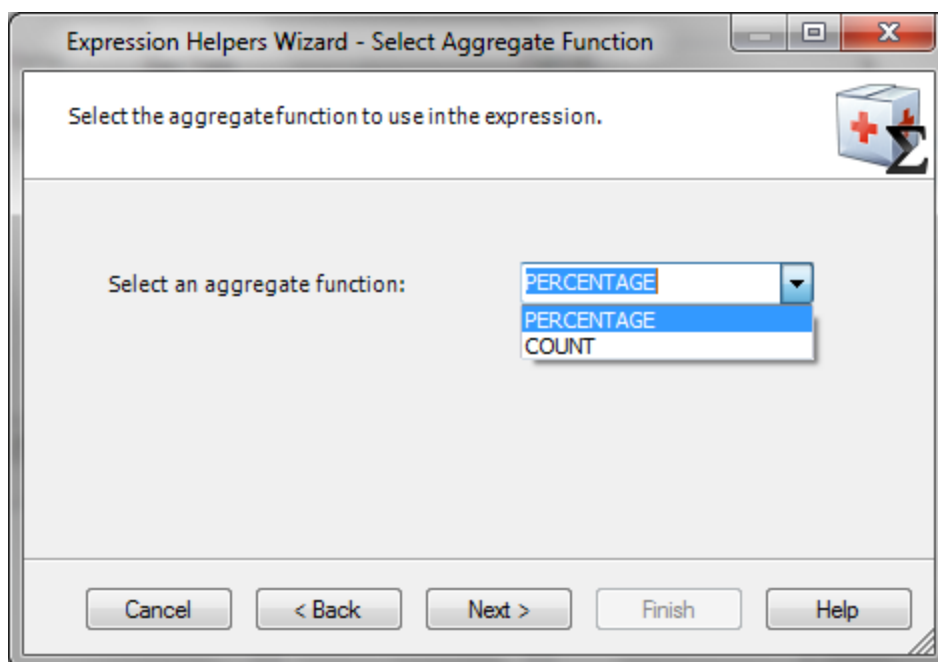


**Note:**

- all String and Boolean fields are discrete;
- in this helper, variables of the types Number, Integer, Time, or Timestamp whose cardinality is less than 50 are considered to be discrete.

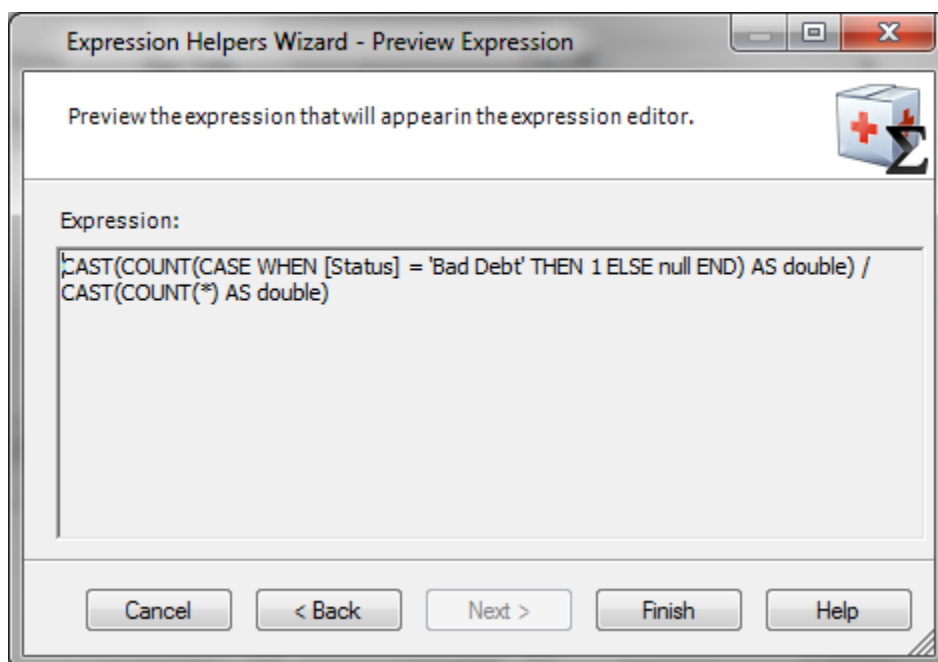
Select whether you want to display the percentage or the count of the category selected at the previous step. The default option is PERCENTAGE.

If the PERCENTAGE option is selected, the Format of the expression in the Expression Editor window is automatically set to **Percent**.



If PERCENTAGE is selected, then the result is as follows.

Note: The CAST ... AS DOUBLE operator takes care of the data type compatibility and conversion. It is inserted automatically whenever necessary.



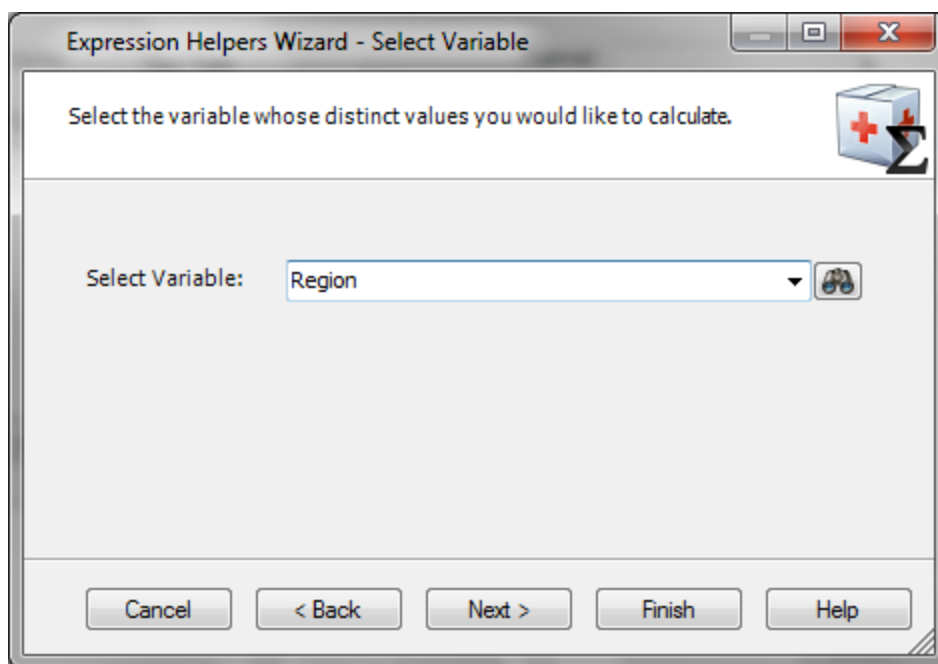
See also:

[Expression Helpers: Overview](#)

## Node Cardinality Helper

Page | 439

In the Node Cardinality helpers wizard, select a variable whose cardinality you want to be displayed in the tree nodes. This is the number of distinct values of the selected variable in a given node.



Click **Next** to preview the resulting expression. See the topic Preview Expression for details.

Click **Finish** to add it to the Expression Editor. For example, if the selected variable is "Region", the helper produces the expression **COUNT(DISTINCT [Region])**

Note that missing values are treated as a separate category when calculating the cardinality of a field.

See also: [Expression Helpers: Overview](#).

## Solution Constraint Helper

The Solution Constraint Helper is provided to assist you in defining constraints for the solutions to the optimization problem (see the topic About Optimization for details).

The helper allows you to assign specific constant values to the nodes you selected for optimization. These constant values will serve as the upper or lower boundaries in the inequalities limiting the

values of the solution in the corresponding nodes. You will use them later in the Optimization Helper.

Thus, the helper allows you to enter expressions with conditions on specific Node IDs of the form:

```

CASE
  WHEN [$node_id$] = <Node ID 1> THEN <constant 1>
  WHEN [$node_id$] = <Node ID 2> THEN <constant 2>
  .....
  ELSE <constant N>
END

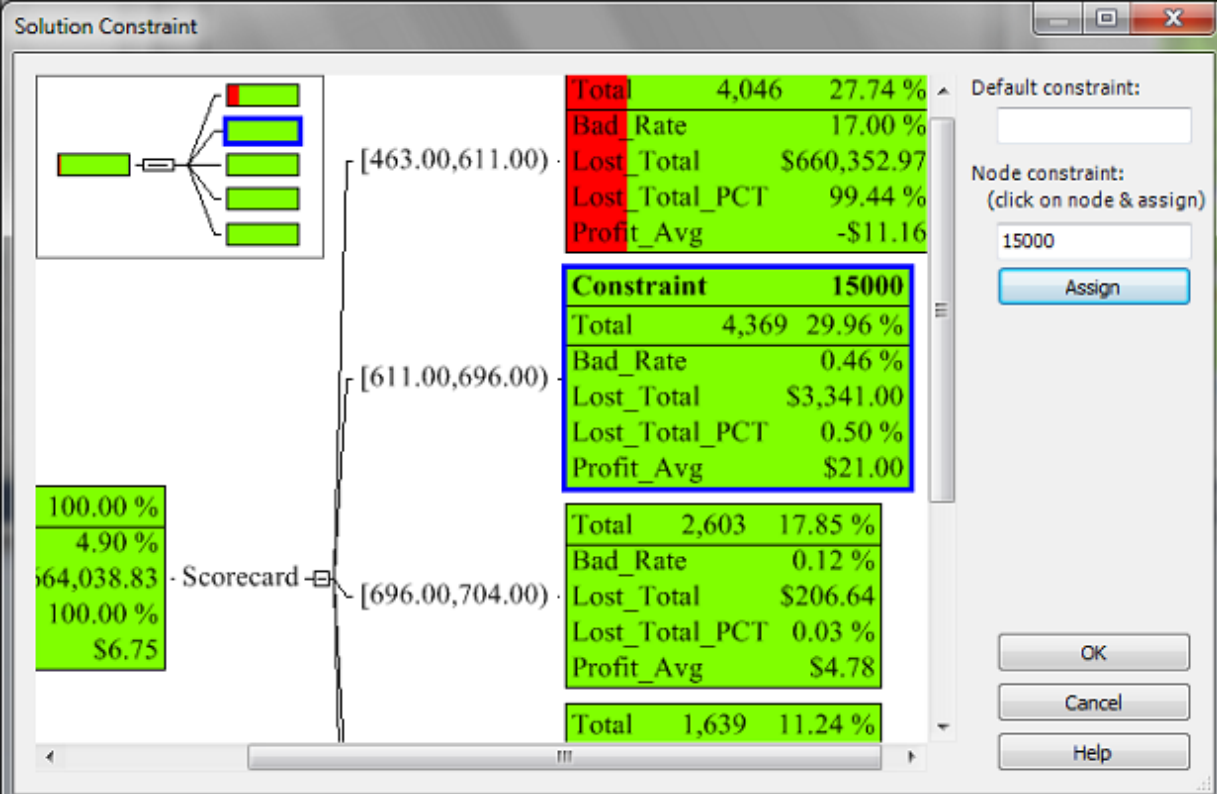
```

Page | 440

You can specify:

- different constraints for specific nodes
- one default constraint for all other nodes (for the ELSE condition in the above expression)

When you click the Helpers button, select Solution Constraint as the Expression type, and click Next, the Solution Constraint dialog opens. It shows both the Strategy Tree and its map in the same window.



**Solution Constraint**

Strategy Tree:

- Scorecard
  - [100.00 %]
  - [4.90 %]
  - [64,038.83]
  - [100.00 %]
  - [\$6.75]

Map:

- [463.00,611.00)
  - Total: 4,046 27.74 %
  - Bad\_Rate: 17.00 %
  - Lost\_Total: \$660,352.97
  - Lost\_Total\_PCT: 99.44 %
  - Profit\_Avg: -\$11.16
- [611.00,696.00)
  - Constraint: 15000**
  - Total: 4,369 29.96 %
  - Bad\_Rate: 0.46 %
  - Lost\_Total: \$3,341.00
  - Lost\_Total\_PCT: 0.50 %
  - Profit\_Avg: \$21.00
- [696.00,704.00)
  - Total: 2,603 17.85 %
  - Bad\_Rate: 0.12 %
  - Lost\_Total: \$206.64
  - Lost\_Total\_PCT: 0.03 %
  - Profit\_Avg: \$4.78
- Total: 1,639 11.24 %

Default constraint:

Node constraint: (click on node & assign)

15000

Assign

OK

Cancel

Help

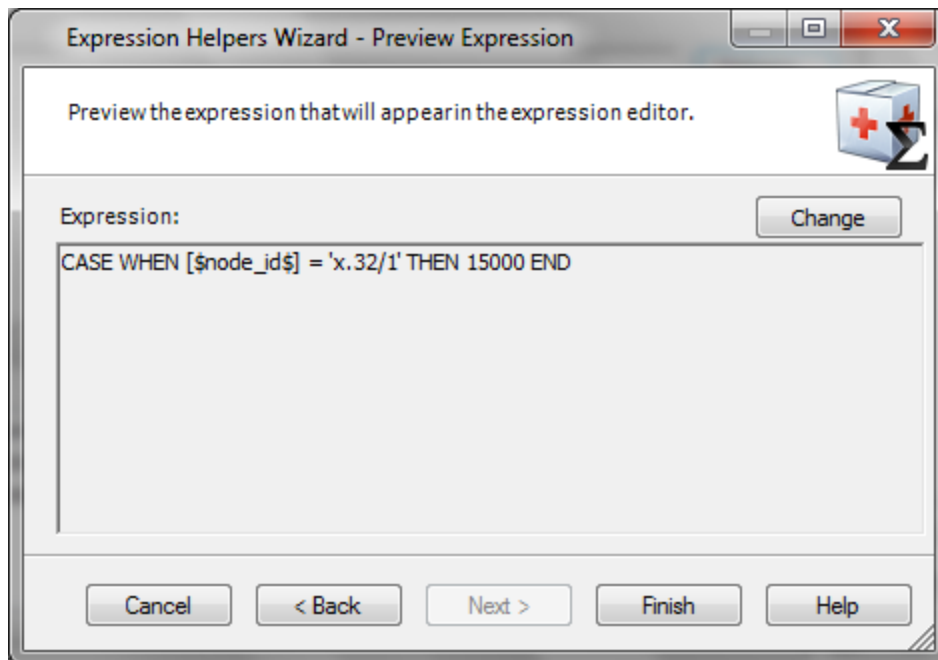
To assign a constraint to a specific node:

- click on the desired node in the tree (or in the tree map in the top left corner)
- enter the desired value in the Node constraint field and click Assign. If your constraint is the same for all nodes, you can enter its value in the Default constraint field instead, and click Assign. The assigned value will appear in the Constraint line at the top of the selected node.
- once you are finished defining different constraints for specific nodes, you can define a default constraint for all other nodes. Enter its value in the Default constraint field instead, and click Assign.

Page | 441

Click OK when done.

The next screen shows the resulting expression. For example:



Click **Change** if you want to go back to the Solution Constraint dialog and change anything. Otherwise, click **Finish**. The expression will be displayed in the Expression Editor, where you can manually edit it as you wish before adding it to the strategy tree. Specify the name of your new constraint expression in the Expression Name field. When done, click **OK** in the Expression Editor.

For details on the Optimization feature, see the topic [About Optimization](#).

See also: [Expression Helpers: Overview](#)

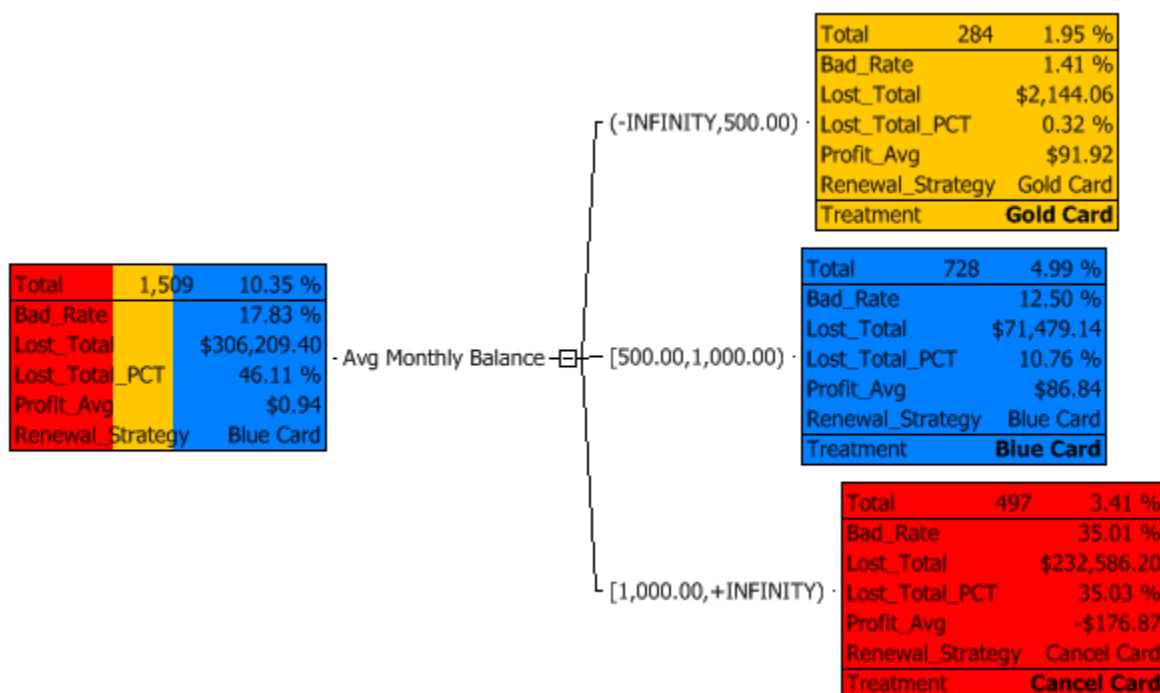
## Treatments

A **treatment** is an action, output, or goal associated with a terminal node of a Strategy Tree as defined by the user. Only one treatment can be associated with a given terminal node. You can create and assign new treatments dynamically or assign previously used treatments.

Page | 442

A single treatment may in fact mean a sequence of regular actions. For example, a treatment named "Repeated offers" could mean the sending of certain promotional materials to a certain group of customers at regular intervals.

A treatment is expressed in the form of an alphanumeric text string, which is assigned using a special command.



Knowledge Studio has the following capabilities in dealing with treatments:

- Color can be assigned to any treatment to improve the visualization and clarity. For example, you can assign blue color to the "Accept" treatment and red to "Reject".
- A monetary amount (cost or revenue) can be assigned to each treatment.
- Your strategy can be summarized based on treatments and view profiles and statistics of nodes with various treatments assigned. For example, treatments can be viewed and assigned not only on the main tree view, but also in Node Report and Profile Chart.
- A Treatments Report showing the list of treatments (including costs) can be generated.
- The command **Tools | Treatments from Calculation** allows you to convert the result of a calculation to a treatment and assign it.

- Treatments can be imported from and exported to an external file.
- When copying and pasting splits, you have an option of pasting treatments.

Note that splitting a node (e.g., using Find Split or Force Split) will remove assigned treatments from the current node (if any) since it will no longer be a terminal node after this operation.

Page | 443

See also: [Managing Treatments](#)

## Managing Treatments

For general information about treatments, see the topic [About Treatments](#).

To assign, remove, or edit a treatment, select a terminal node and click the Treatments icon  in the toolbar or select the Treatments command from the Tools menu:

### From the menu:

Tools | Treatments...

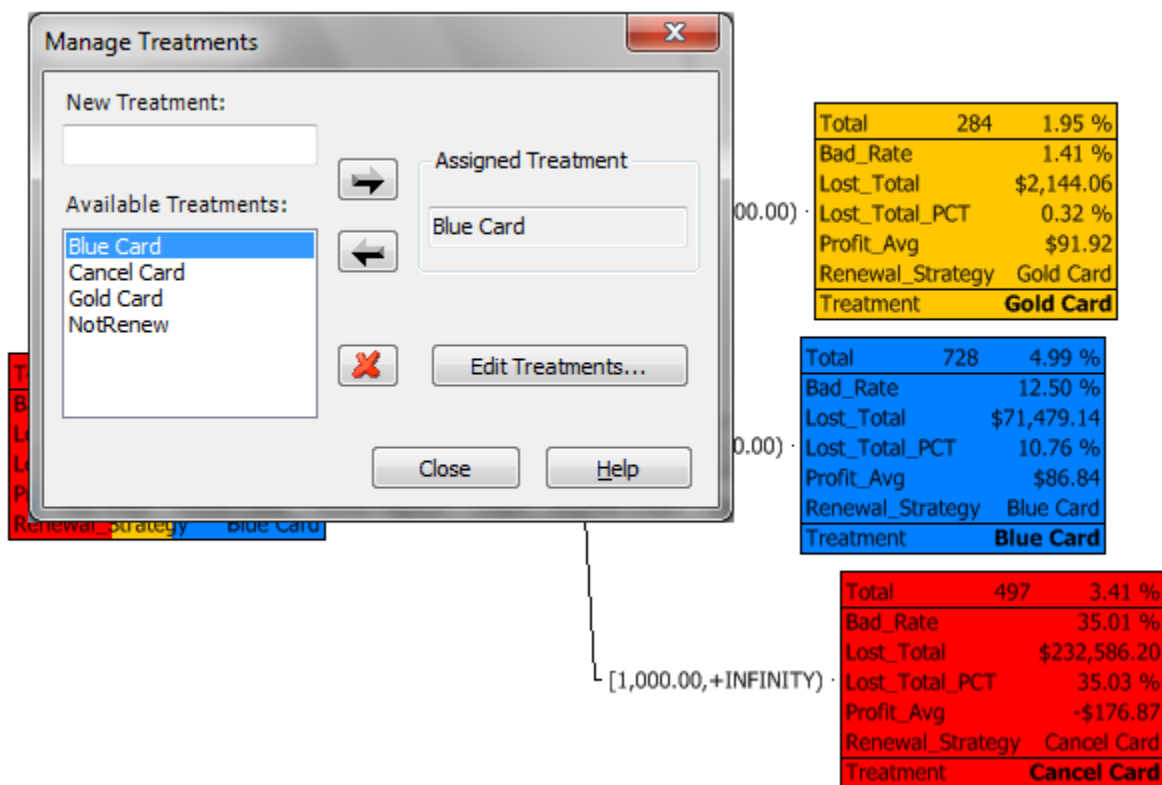
### From the toolbar:

click the icon



This will bring up the Manage Treatments dialog box. If the desired treatment already exists in the list of available treatments, select it and click the right arrow button to assign it to the selected node. Otherwise, enter the name of the new treatment and click the right arrow button.

In this example, the "Blue Card" treatment is created and assigned to the middle leaf:

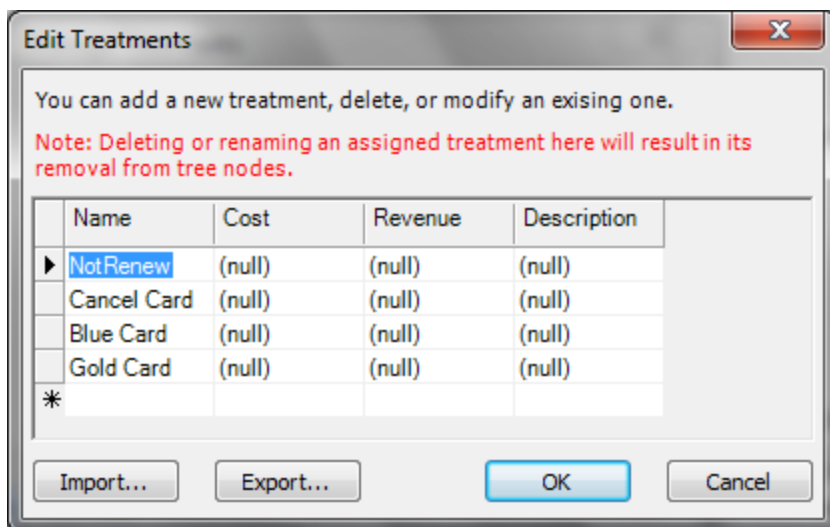


To **unassign** a treatment from a node, select this node first (its treatment is displayed in the Assigned Treatment box) and click the left arrow. Note that the treatment will remain in the Available Treatments list.

*Note that splitting a node (e.g., using Find Split or Force Split) will remove the assigned treatment from the current node (if any) because it will no longer be a terminal node.*

To **edit** a treatment (rename, or assign cost and revenue, or enter a description), click the **Edit Treatments** button.



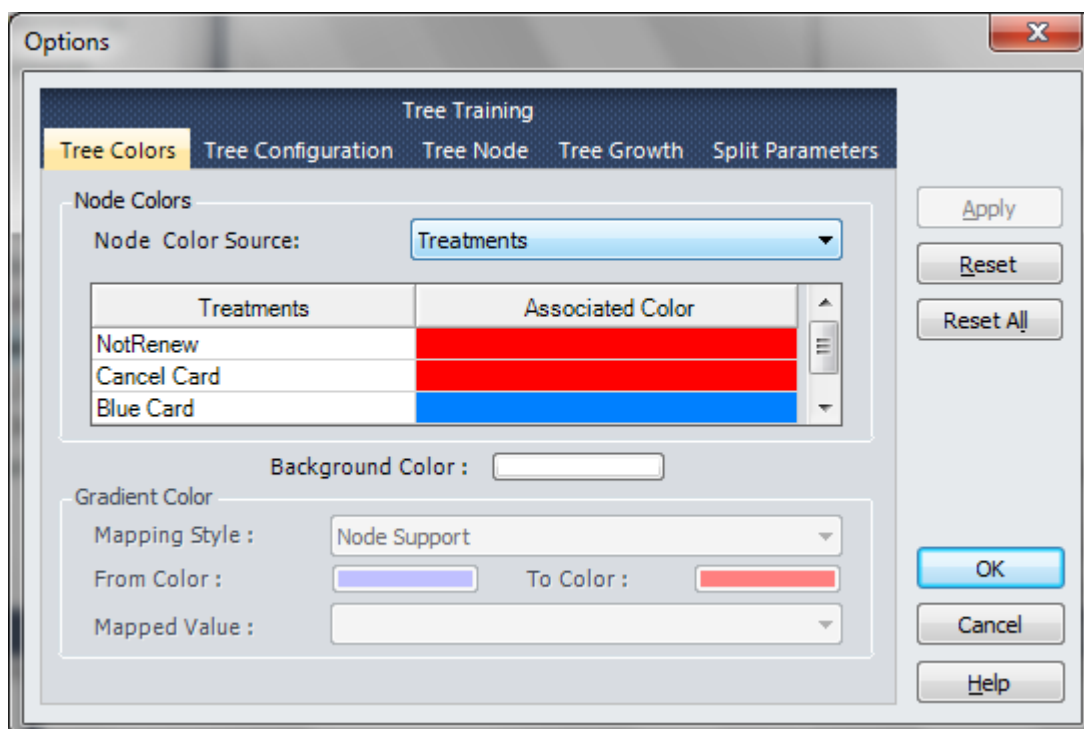


*Note that renaming a treatment is equivalent to replacing it with a new one, so it will be removed from all nodes (if it is assigned to any).*

To delete a treatment completely (i.e., to remove it from all nodes and the list of available treatments), click the Edit Treatment button, select the treatment and press the Delete key.

To delete all available treatments, click the button .

You can assign a color to a treatment using the Options command. Select **Tools | Options** from the menu and click the Tree Colors tab in the Options dialog. Choose 'Treatments' as Node Color Source and assign the desired color to your treatment. The node that has this treatment will have the corresponding color.



See also: [About Treatments](#)

## Generating Treatments From Calculations

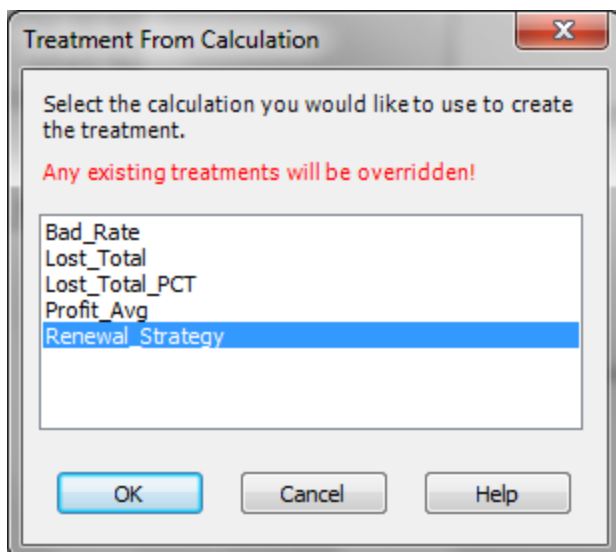
In some cases, your treatments may consist in assigning numerical values (for example, dollar amounts to each record in the selected node). If you already have those values calculated in the Strategy Tree nodes, you may want to convert them to treatments.

For this purpose,

- Click on any node. In this example, the "Blue Card" Renewal\_Strategy is a string-valued calculation (which may have resulted, for example, from the [Computed Conditional Treatment Helper](#)). We need to convert it to an actual treatment.

|                  |             |         |
|------------------|-------------|---------|
| Total            | 728         | 4.99 %  |
| Bad_Rate         |             | 12.50 % |
| Lost_Total       | \$71,479.14 |         |
| Lost_Total_PCT   | 10.76 %     |         |
| Profit_Avg       | \$86.84     |         |
| Renewal_Strategy | Blue Card   |         |

- select **Tools | Treatment From Calculation** from the menu



- in the Treatment From Calculation dialog, select the calculation you would like to convert to a treatment and click **OK**.

Since calculations are displayed in all nodes of the tree, the corresponding treatment will be assigned to every terminal node.

As a result, your nodes will look like this (in this case, we generated a treatment from the *Renewal\_Strategy* calculation, so our treatment coincides with the value of *Renewal\_Strategy* at this node:

|                  |     |                  |
|------------------|-----|------------------|
| Total            | 728 | 4.99 %           |
| Bad_Rate         |     | 12.50 %          |
| Lost_Total       |     | \$71,479.14      |
| Lost_Total_PCT   |     | 10.76 %          |
| Profit_Avg       |     | \$86.84          |
| Renewal_Strategy |     | Blue Card        |
| Treatment        |     | <b>Blue Card</b> |

The treatment in this case is "Blue Card". You can rename the resulting treatment by editing it. To rename a treatment,



- click the Treatments button  in the toolbar, or select **Tools | Treatments** from the menu.
- click **Edit Treatments** in the Manage Treatments dialog
- rename the treatment you have just generated

## Treatment Report

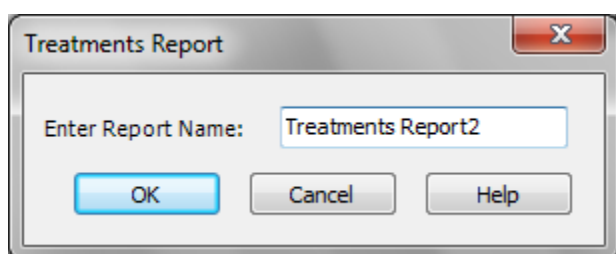
The **Treatment Report** is a simple report that shows all assigned treatments in the Strategy Tree.

Page | 448

In addition, it also shows the following node information for any node with a treatment:

- the node size (the number and percentage of records)
- the values of all node calculations
- all the treatment attributes from the Edit Treatments dialog (cost and revenue, if defined)

To generate a treatment report, select **Tools | Treatment Report** from the menu.



Enter the desired name for the report and click OK.

An example of a treatment report:

Treatments Report - [Project3].[Credit\_Strategy]

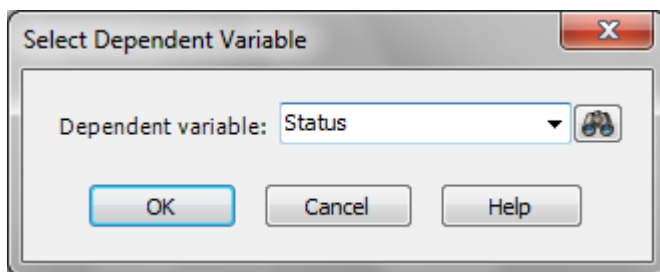
| Treatment            | Size(Count) | Size(Percent) | Description | Cost | Revenue | Amt_Lost_Total | AvailableLineTotal | AvgProfit | Bad_Rate |
|----------------------|-------------|---------------|-------------|------|---------|----------------|--------------------|-----------|----------|
| No increase          | 3,024.00    | 10.37%        | N/A         | N/A  | N/A     | \$740,551.82   | \$2,453,935.00     | -35.18    | 23.74 %  |
| \$2000 Line Increase | 13,044.00   | 44.72%        | N/A         | N/A  | N/A     | \$15,231.42    | \$13,096,712.00    | 11.44     | 0.66 %   |
| \$5000 Line Increase | 1,510.00    | 5.18%         | N/A         | N/A  | N/A     | \$44,268.90    | \$1,034,850.00     | 90.07     | 5.30 %   |
| \$3000 Line increase | 5,199.00    | 17.82%        | N/A         | N/A  | N/A     | \$4,764.58     | \$4,876,954.00     | 20.97     | 0.37 %   |

## Creating Splits

### Find Split

The **Find Split** command prompts you to choose a dependent variable

Page | 449



and then searches for the independent variables (IV) that are most significant to this dependent variable (DV) based on a criteria defined by the [measure](#). The Independent variables that are found to be significant are called the split variable(s). The most significant split variable is displayed as child nodes where each child node represents a category or range found in the independent variable. The Find Split algorithm in Strategy Trees is the same as in Decision Trees. See the [Find Split](#) topic in the Decision Tree section for details.

The **Find Split** command may find multiple significant splits. The splits are ranked according to the measure and the first and most significant split is displayed in the tree. The other splits can be viewed using [Next Split](#) or [Go To Split](#)

Strategy trees do not require dependent variables. However, the results of the **Find Split** command will produce a split that contains the distribution of the dependent variable.

#### Root Node of strategy tree

|       |        |          |
|-------|--------|----------|
| Total | 20,416 | 100.00 % |
|-------|--------|----------|

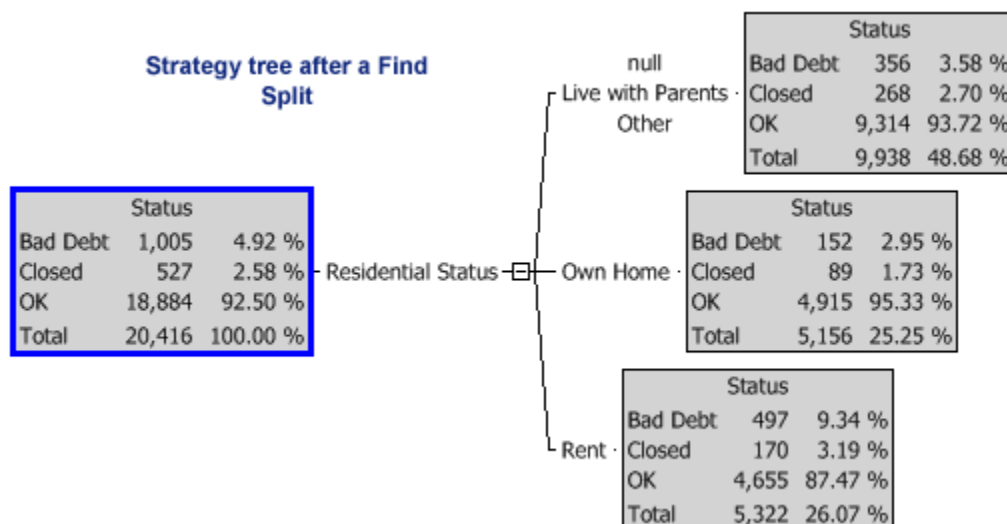
To perform a **Find Split**:

#### From the menu:

Grow | Find Split

#### From the popup menu:

right-click on the parent node and select Find Split



## Force Split

Unlike **Find Split**, **Force Split** allows you to choose the variable that you want as your split variable. Like **Find Split**, the node to split must be active:

Root Node of strategy tree

|       |        |          |
|-------|--------|----------|
| Total | 20,416 | 100.00 % |
|-------|--------|----------|

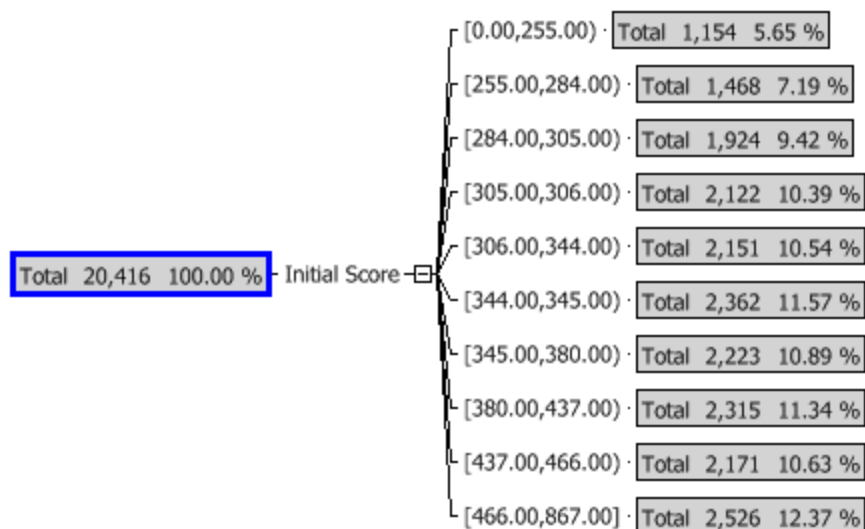
### From the menu:

Grow | Force Split...

### From the popup menu:

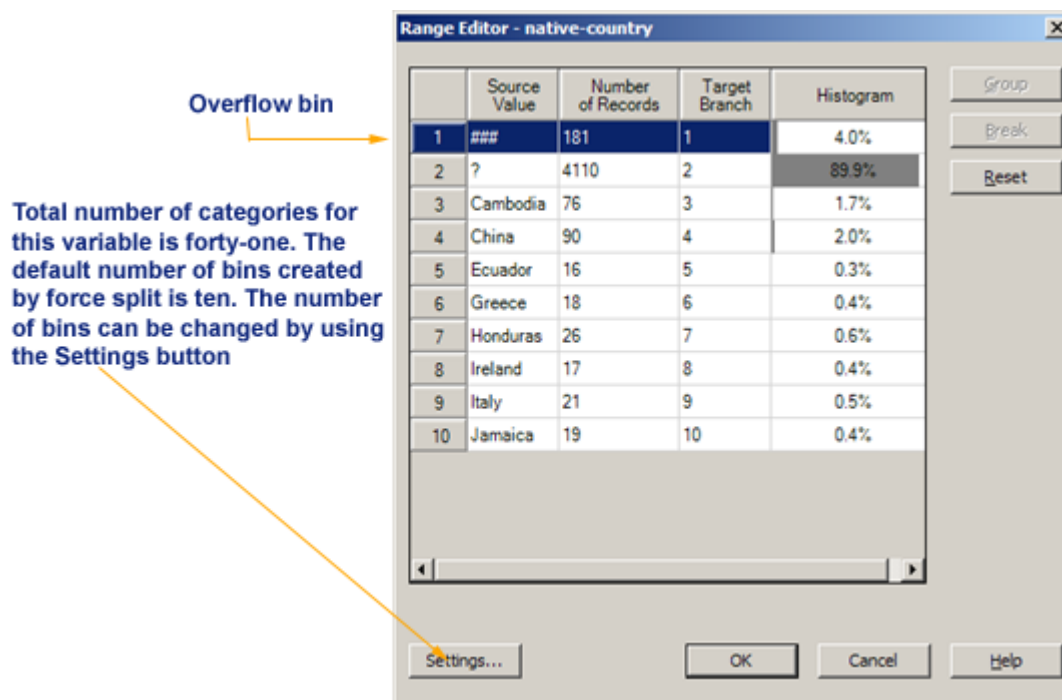
on the parent node, right-click | Force Split...

A series of steps takes you through the **Force Split** process. This includes choosing the split variable and defining the binning of the variable as it will appear in the tree. The resulting split of the above example is:



## About Force Split

- The **Force Split** command does not use the settings in the attribute editor.
- If the cardinality of the split variable is greater than ten (>10), then by default the split variable is divided into ten bins, nine of the bins will be categories (or ranges) from the variable and the tenth bin will be the overflow bin, represented as ###.
- [Example](#)



- A high cardinality split variable will be limited to one hundred bins/categories. There will be ninety-nine categories (or ranges) and the hundredth (100th) bin will be the overflow bin, represented as ###.
- When performing a force split on a node that contains existing splits, all the splits will be replaced by the force split command. This is the same as performing an Erase All before doing a force split.

## Navigating between Splits

**Next split** displays the next significant split in the list of splits created by the [Find Split](#) command. This command is only enabled if there is more than one split created by the **Find Split** command.

### From the menu:

Reshape | Next Split

### From the popup menu:

Right click on parent node | Next split



**Previous split** displays the previous significant split in the list of splits created by the [Find Split](#) command. This command is only enabled if there is more than one split created by the **Find Split** command.

### From the menu:

Reshape | Previous Split

### From the popup menu:

Right click on parent node | Previous split

The **Go To Split** command allows you to go to a different split on the parent node. If there are no other significant splits calculated, this item will be disabled.

### From the menu:

Grow | Go To Split

### From the popup menu:

on the parent node, right-click | Go To Split

The **Go To Split** dialog will appear and display a list of all the significant splits calculated for the given node in rank order. The rank corresponds to the **Info** column in the [split report](#).

## Modifying Splits

The **Copy/Paste Splits** commands allow you to copy the splits from one tree and paste them to another. This feature can be useful in a variety of situations.

- See how the split in one node behaves when copied to another node of the same tree
- See how the split in one tree behaves when copied to the node of a different tree created in a different dataset.

## Copy Splits

The **Copy Splits** command copies into memory all the splits of the tree, starting from the active node.

Highlight the node you want to copy

### From the menu:

Edit | Copy Split

### From the popup menu:

Right click on parent node | Copy Split

Page | 454

## Paste Splits

The **Paste Splits** command pastes splits that were copied to the clipboard with the **Copy Splits** command. The tree structure from the original Copy Split is maintained.

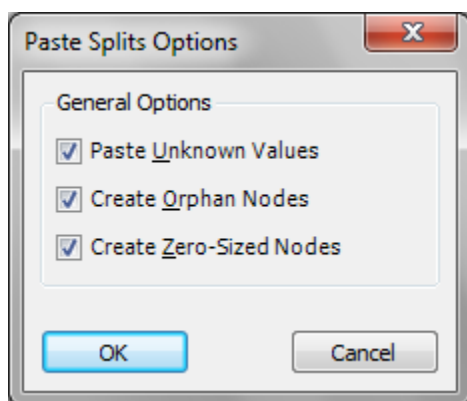
Highlight the node where you want to paste the split.

### From the menu:

Edit | Paste Split

### From the popup menu:

Right-click on the parent node  
Choose Paste Split



Choose the desired Paste Split Options

Click OK to Paste the splits

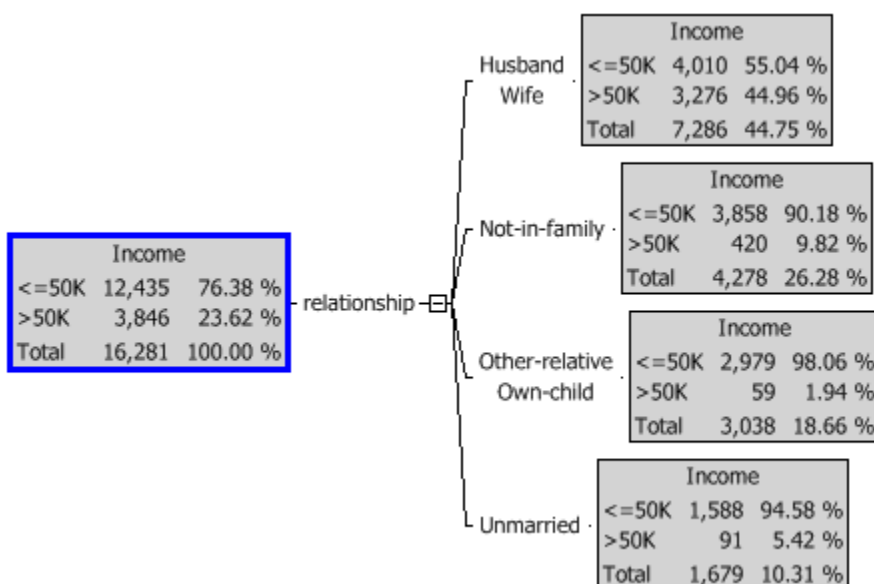
## Paste Split Options

The meaning of Paste Split Options in Strategy Trees is the same as in Decision Trees. For details, see the [Copy Splits / Paste Splits](#) topic in the Decision Tree section.

Page | 455

## Edit Split

The **Edit Split** command allows you to modify the display of the ranges (for continuous variables) or bins (for discrete/categorical variables) for the current split.



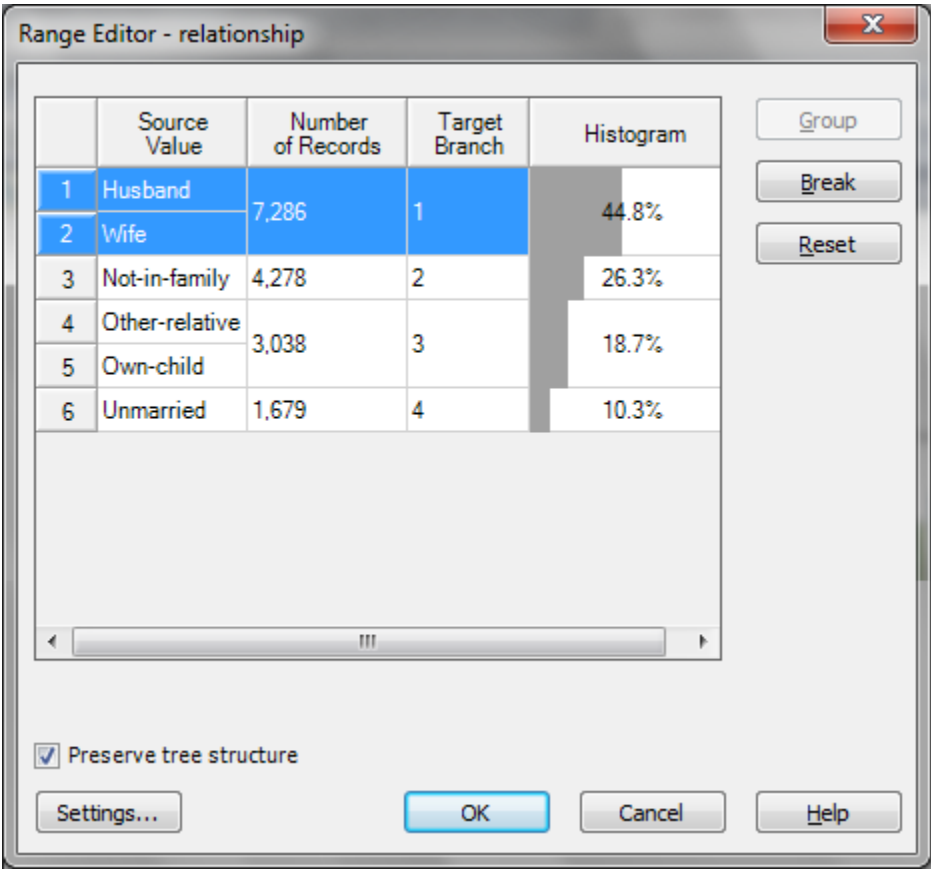
### From the menu:

Grow | Edit Split

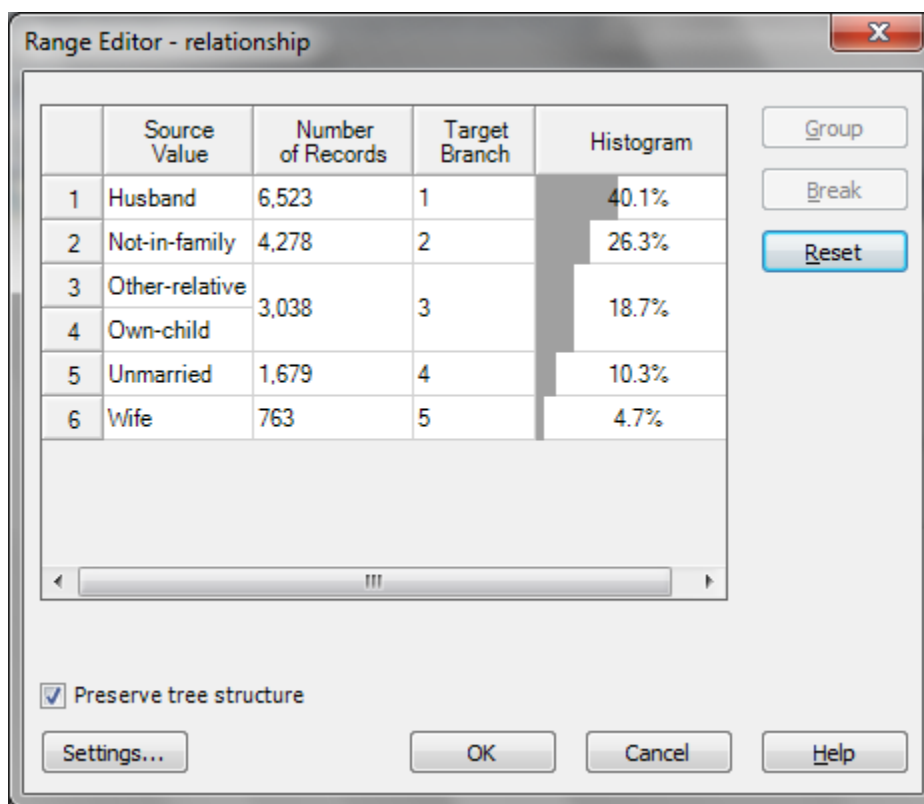
### From the popup menu:

Right click on parent node | Edit Split

The **Edit Split** command takes the existing bin/ranges from the variable and displays it in the range editor.



From the range editor, you can modify the bins. For example, you may want to separate the categories **Husband** and **Wife**. Highlight the group and click the **Break** button.



## Erase Child Nodes

The **Erase Child Nodes** command deletes all nodes beneath the current node. First, select a node whose child nodes you want to delete.

### From the main menu:

Grow | Erase Child Nodes

or

Right click on the node and select Erase Child Nodes

## Ignore Variable

The **Ignore Variable** command removes the variable of the current split from the analysis. The same effect can be accomplished by ignoring the variable in the Attribute Editor.

### From the menu:

Grow | Ignore Variable

Page | 458

### From the popup menu:

Right click on parent node | Ignore Variable

## Modifying Tree Attributes

### Using the Attribute Editor

The Attribute Editor allows you to modify variable attributes for the tree. The attribute values can be set and modified when inserting the tree in the tree wizard or modified after the tree has been inserted. Some of the tree parameters determined by the values set in the Attribute Editor are:

- whether a variable is included in the model
- the role of variables in the model (dependent, independent, or weight)
- the maximum number of nodes in splits produced by Find Split and AutoGrow
- the maximum number of bins or intervals in the Force Split Range Editor
- whether the values of a variable are treated as ordered or not
- whether grouped values of numeric variables are displayed in intervals or as discrete values
- the way missing values are treated

While a decision tree is open, the Attribute Editor can be invoked as follows:

### From the menu:

Tools | Attribute Editor...

### From the background context menu:

Right-click anywhere on the tree background and select Attribute Editor...

Attribute Editor

| #  | Variable Name  | Include | Role        | Usage      | P-Value | Default # Bins | Intervals Type | Break-apart | Missing values |
|----|----------------|---------|-------------|------------|---------|----------------|----------------|-------------|----------------|
| 1  | age            | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 2  | workclass      | yes     | independent | nominal    | 0.05    | 9              |                | yes         | use            |
| 3  | fnlwgt         | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 4  | education      | yes     | independent | nominal    | 0.05    | 50             |                | yes         | use            |
| 5  | education-num  | yes     | independent | ordinal    | 0.05    |                |                | yes         | float          |
| 6  | marital-status | yes     | independent | nominal    | 0.05    | 7              |                | yes         | use            |
| 7  | occupation     | yes     | independent | nominal    | 0.05    | 50             |                | yes         | use            |
| 8  | relationship   | yes     | independent | nominal    | 0.05    | 6              |                | yes         | use            |
| 9  | sex            | yes     | independent | nominal    | 0.05    | 2              |                | yes         | use            |
| 10 | capital-gain   | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 11 | capital-loss   | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 12 | hours-per-week | yes     | independent | continuous | 0.05    | 10             | static         | yes         | float          |
| 13 | native-country | yes     | independent | nominal    | 0.05    | 50             |                | yes         | use            |
| 14 | Income         | yes     | dependent   | discrete   |         |                |                |             | use            |

OK Cancel Help

Page | 459

Scroll to the right to see more attributes:

Attribute Editor

| #  | Variable Name  | Missing values | Ordered display | Max Branches | Open Left                | Open Right               | Cardinality | Format String |
|----|----------------|----------------|-----------------|--------------|--------------------------|--------------------------|-------------|---------------|
| 1  | age            | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 73          | 0             |
| 2  | workclass      | use            | present         | 0            |                          |                          | 9           |               |
| 3  | fnlwgt         | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 12787       | 0             |
| 4  | education      | use            | present         | 0            |                          |                          | 16          |               |
| 5  | education-num  | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 16          | 0             |
| 6  | marital-status | use            | present         | 0            |                          |                          | 7           |               |
| 7  | occupation     | use            | present         | 0            |                          |                          | 15          |               |
| 8  | relationship   | use            | present         | 0            |                          |                          | 6           |               |
| 9  | sex            | use            | present         | 0            |                          |                          | 2           |               |
| 10 | capital-gain   | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 113         | 0             |
| 11 | capital-loss   | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 82          | 0             |
| 12 | hours-per-week | float          | range           | 0            | <input type="checkbox"/> | <input type="checkbox"/> | 89          | 0             |
| 13 | native-country | use            | present         | 0            |                          |                          | 41          |               |
| 14 | Income         | use            |                 |              |                          |                          | 2           |               |

OK Cancel Help

The global defaults for the tree attributes can be set in the Tree Attributes tab of the Preferences dialog accessible from the Tools menu. See the topic [Preferences: Tree Attributes](#) for details.

**Tip:** you can change attribute values for more than one variable at a time. To this end, click in the attribute column of your choice and drag your mouse down while holding down the left mouse button to highlight several cells in the column. (Alternatively, while holding down the SHIFT or CTRL key on your keyboard, click on several cells whose values you want to change in the selected column.) When finished selecting multiple cells, set the value for one of them (the last one you highlighted). This will set the same value for all highlighted cells in the selected column.

You can also click the Field Finder (binoculars) button to quickly find and select the desired variable and use the search capability.

Tree attributes can be copied from one tree to another. Open the tree whose attributes you want to copy and open its Attribute Editor. Click the Copy button and close the Attribute Editor. Then open the other tree, open its Attribute Editor and click the Paste button. All attributes for the variables that are common for both trees will be copied. Click OK to apply the changes and close the Attribute Editor.

**NOTE:** After you apply changes made in Attribute Editor and return to the tree view, you can still undo them using the Undo command, provided that you haven't made changes to a different tree or dataset.

## Tree Attributes

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Variable Name</b> | Name of the variable in the decision tree.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Include</b>       | Determines if a variable is to be included in the analysis.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Role</b>          | Determines how a variable is used in an analysis. A number of options are available:<br><br><b>Dependent:</b> Indicates that this variable is the dependent variable, or the variable that you want to explore. There can only be one dependent variable (DV).<br><br><b>Independent:</b> Indicates that this variable should be compared to the dependent variable to determine if it has predictive value.<br><br><b>Weight:</b> Indicates that this variable should be used as a weight that controls how much emphasis each record should be given in the analysis. If this option is selected, you must also select the type of weighting in the Weight Type column.<br><br><a href="#">Example</a> |
| <b>Usage</b>         | Algorithms can treat values in a number of ways. In general, if the values of a field are ranked (like 1, 2, 3, or 1-low, 2- medium and 3-high) then they are 'ordinal'. If there is no ordering, i.e., if they are just categories or qualities, they are considered to be 'nominal' values (red, blue and green).                                                                                                                                                                                                                                                                                                                                                                                      |



**No binning:** Each category within the variable will form its own separate node in the tree. The search for similar groupings is turned off.

**Ordinal:** Treats values as an ordered set (e.g., 1-low, 2-medium, 3-high or 1, 2, 3). Only adjoining values will be grouped together when an algorithm is looking for similarities. For example, 1-low can be paired with 2-medium, or 2-medium can be paired with 3-high but 1-low cannot be paired with 3-high.

**Nominal:** Values combine freely. For example, red may combine with blue or green. All possible groupings of field categories are examined so that similar items are grouped together to become a node in the tree.

**Continuous:** The variable is split into intervals. Intervals are calculated in a number of different ways, based on your selections in the Intervals Type column.

**Discrete:** Only available with the dependent variable. Allows you to keep dependent variables ungrouped.

### Example

#### ***p-value***

Defines the level of confidence required to merge values into groupings. This setting affects the bushiness of the decision tree - a low setting (0.001) will tend to produce few branches (splits) from each node, whereas a high setting (0.20) will produce a bushy tree with more branches. The default setting is 0.05. With a p-value setting of 0.05 you would typically only be wrong considering values as category components 5 times out of 100. Similarly, a setting of 0.001 will assure you that you have a 99.9% accuracy in the category assignments but that you will also have fewer resulting categories. A setting of 0.20 would result in more branches but with less certainty of the accuracy (80% confidence).

#### ***Default of Bins*** #

- If **Usage** is **continuous**, the value in this column determines the number of intervals for the selected variable according to the following rule: To form splits in a decision tree, continuous values are initially divided into discrete intervals where each interval has approximately the same number of observations. These discrete intervals are then treated as ordered and combined accordingly. The number specified here determines how many **initial** intervals are formed for this continuous variable prior to the ordered grouping. Therefore, the resulting number of nodes may be less than or equal to the value in the Interval column. For example, if Interval=8, the resulting number nodes can be less than or equal to ( $\leq$ ) 8.

See the topic "About Intervals/Bins" for details.

- If **Usage** is **nominal**, the value in this column determines the maximum number of categories (bins) to display for the selected variable. The default value is equal to the cardinality or 50 (whichever is less). You may enter any integer number between 1 and the cardinality of the variable. If the maximum number of bins **N** is equal to the cardinality of the variable, then each of its values is represented as a separate category. If N is less than the cardinality, the following mapping is applied: the first **N-1** unique values are represented as separate categories and the rest are mapped into one category denoted by triple number sign ("###").

#### ***Intervals Type***

This option affects the Find Split and Automatic Grow operations and applies only to the variables with *continuous* Usage (see above). It defines

the level at which the binning maps for continuous independent variables are calculated.

**Static:** The binning map for the variable is precomputed at the root node. The intervals remain static throughout the tree.

**Dynamic:** The binning map for the variable is calculated dynamically at every node.

Although the Static type is the default, the Dynamic type provides more intelligent mapping and is therefore recommended in most cases. Note that splitting a variable with dynamic interval type is up to two times slower than with the static type, but this may not be noticeable on fast computers.

### Break-apart

Allows you to optimize the growing of the tree by choosing either speed or accuracy. There are two options:

**Yes:** this setting gives the most accurate results. **Yes** is the default.

**No:** this setting is used when you have variables with many categories and you want to speed up the analysis. However, the results may not be optimal.

### Missing Values

Allows you to specify how missing values in your data are handled in the tree. Note that the effect and the applicability of the missing value options depends on the variable usage (independent, dependent, or weight) and its Usage.

#### 1. For the dependent variable (DV):

**Use** (applicable to discrete DV only): Missing values will be treated like any other value of the DV. More precisely, a missing value will be considered a separate DV category designated as **null** and will be shown in every node together with other DV values. This option is not applicable to continuous DV.

**Ignore:** All records containing missing values in the dependent variable will be excluded from the analysis. Note that in this case the root node will contain fewer records than the parent dataset. If the DV of a Decision Tree is continuous, the records with missing values in the DV are always excluded, so **Ignore** is the default and the only available option for continuous DV in Decision Trees. This does not apply to Strategy Trees.

Note that if you ignore missing values in the DV, it will affect all tree views except for the Data tab. which will still show records with missing values in the DV. (The Data tab always shows the actual raw data for any selected node, which is determined by the node rule only, not by the settings in Attribute Editor).

#### 2. For the independent variables (IVs):

**Use:** A missing value in the selected variable will be considered one of its normal categories and will be used to calculate the split like any other category. It is designated as **null** and shown in every split on this variable - in the node rules, in the Range Editor of Force Split and Edit Split, etc.

This option is the default for discrete independent variables (those whose Usage is "*nominal*" or "*no binning*")

**Ignore:** Missing values in the independent variable will be ignored. This means that any split on this variable will not include the condition "IF <variable\_name> is NULL", so the child nodes in this split will not contain records with missing values in this variable. In this case the sum of the numbers of records in the child nodes of a split will not necessarily add up to the number of records in their parent node.

**Float:** Missing values in the variable will not be used to calculate the split, but will then be grouped with the branch (child node) that presents the most similar behavior in terms of the target (DV) distribution. This option is the default for continuous and ordinal independent variables.

### 3. For variables used as weight: Not applicable.

#### **Ordered Display**

Allows you to define how ordered variables are displayed. There are three options:

**Present:** displays all values which are present in the node. If a particular value of the variable is not present in the node, it will not be listed.

**All:** displays all values that fit within the range in the node, whether they are present in that particular node or not. For example, if a node contains the values 1, 2, 4, and 5 (and the value 3 exists somewhere in the dataset), the value 3 will be listed for the node.

**Range:** displays values as a range. For example, if a split groups the values 1, 2, 3, and 4 together, the values will appear as [1, 4].

#### **Max Branches**

The Max Branches value determines the maximum number of branches (leaves) produced by Find Split on the given variable. For example, if you want Find Split to produce binary splits for a specific independent variable, specify "2" in the Max Branches column for this variable. If you want to grow a binary tree (i.e., a tree with binary splits only), enter "2" in the Max Branches column for all variables. See above for the tip on changing attributes for more than one variable at a time.

Note that this setting does not apply to the Force Split operation. Also, note the difference from the Default # of Bins attribute. The Find Split algorithm *starts* calculating the split with the number of branches determined by Default # of Bins value. As opposed to Default # of Bins, Max Branches never leads to the creation of the other ("###") category for discrete variables.

#### **Open Left**

The splits created by the independent variable are not bound by the minimum data values found in the dataset.

#### **Open Right**

The splits created by the independent variable are not bound by the maximum data values found in the dataset.

#### **Cardinality**

The cardinality (number of unique values) of the variable.

#### **Format String**

Allows you to specify various formats (such as currency, percent, etc.) and precision for displaying values in the tree. You can either choose one of the standard formats from the Format String drop-down list or enter a custom format. For detailed description of format strings, see [Specifying Format](#).

## Variable Usage Types

By setting the **Usage** options for variables, you can define the way in which the values of a field are treated by the tree algorithm and, therefore, the way they will be displayed in the splits (how they will be grouped, whether they will be represented in ranges, etc). For example, if the values of a field are ranked (like 1, 2, 3, or low, medium and high) then they are 'ordinal'. If there is no

ordering, i.e., if they are just categories or qualities, they are considered to be 'nominal' values (red, blue and green).

## Usage set to 'no binning'

Page | 464

When set to **no binning**, each value of the independent variable forms a separate branch, i.e., individual values are presented as unique nodes in the tree.

Example

## Usage set to 'ordinal'

Values of this Usage are treated as an ordered set (e.g., 1, 2, 3, or a,b,c...). Only values immediately succeeding one another will be grouped together when an algorithm is looking for similarities. For example, 'a' can be paired with 'b', but not with 'c'.

This Usage is the default for independent variables of types Number, Integer, Date, Time, or Timestamp if their cardinality is less than 50.

You can assign the Usage 'ordinal' to String variables (which are 'nominal' by default). In this case, when creating the split, the algorithm will order the values alphabetically and then try to group contiguous values together to create a branch. If your data is not alphabetically ordered, you may want to create a calculated column that transforms it. For example, use the Binning helper in Dataset Editor to transform the category 'low' to '**1-low**', 'medium' to '**2-medium**' and 'high' to '**3-high**'.

Example

## Usage set to 'nominal'

Values of this Usage are treated as an unordered set. All possible groupings of field categories are examined so that similar items are grouped together. This Usage is the default for String and Boolean variables.

Example

## Usage set to 'continuous'

This Usage is available for independent variables of types Number, Integer, Date, Time, or Timestamp and it is the default if the cardinality is higher than 50. A variable of this Usage is split into intervals that are calculated in a number of different ways based on your selections in the Intervals Type column and the [Intervals/Bins](#) column. See also the topic [Using the Attribute Editor](#) for more information about the Intervals/Bins column.

## Weight

Weight in Altair datasets and models can be viewed as a generalized frequency weight (the weight values don't have to be integer). A weight can be set at the dataset level, but it can also be set and changed independently in trees and models.

---

See the [Weight](#) topic for general information about weights and on setting weight at the dataset level.

## Weights in Datasets and Dependent Models

Page | 465

The weights in a dataset and its dependent models can be changed or reset independently. When you toggle weight on or off in the dataset Overview Report, this does not affect existing trees and other models depending on the dataset.

By default, any new model created on the basis of a weighted dataset inherits the weight from it. More precisely, when inserting a model for a weighted dataset, the weight field defined in the dataset is automatically listed in the model's Attribute Editor as "weight" (in the Usage column). You can redefine the field as an analysis variable in the Attribute Editor by changing the usage from "weight" to "independent" if desired. Once a model is inserted, it is independent of the parent dataset in the sense that any changes of weights in the model will not affect the parent dataset and vice versa.

## Weight in Strategy Trees

To specify weight for a Strategy Tree, open the tree and select **Tools | Attribute Editor** from the menu.

It is assumed that the field you want to set as weight already exists in the parent dataset. See the [Weight](#) topic for information on available tools for creating a weight field. If the parent dataset already had weight when you were inserting the tree, the same weight is set in the tree by default.

Click in the **Usage** column for the variable you want to set as weight and select "**weight**" from the drop-down list. Note that this usage option is available only for nonnegative numeric variables. The weight variable does not have to be integer.

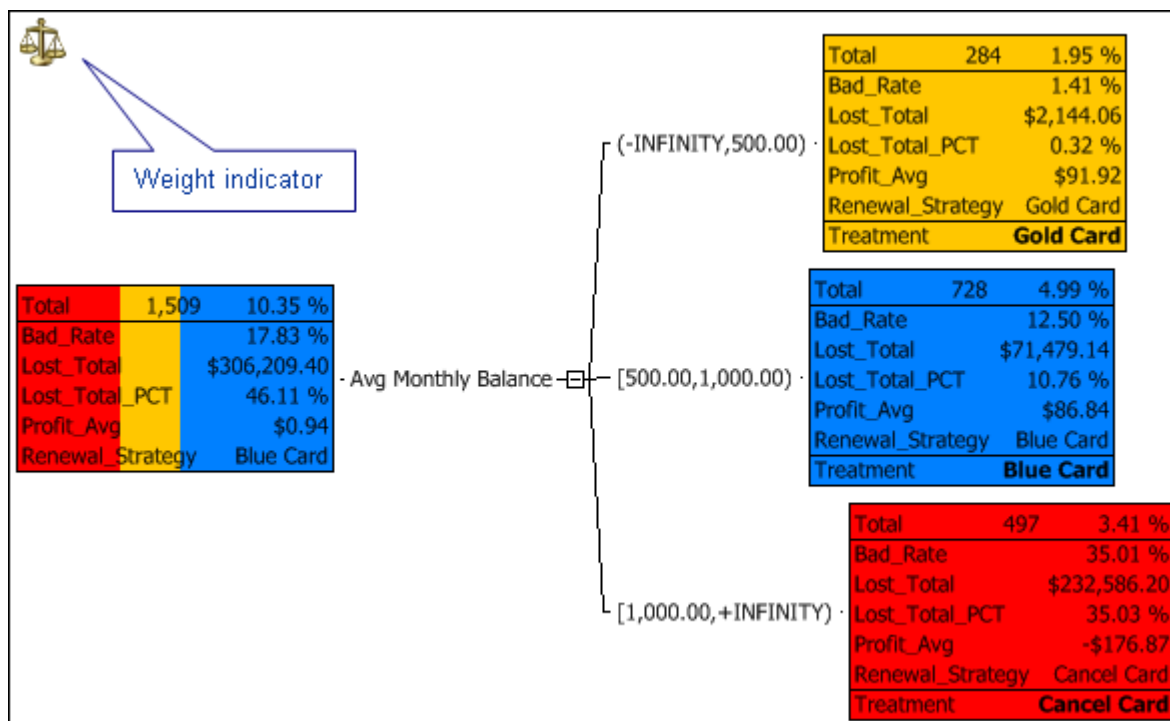
Attribute Editor

| #  | Variable Name                | Include | Role        | Usage      | P-Value | Default # Bins | Intervals Type |
|----|------------------------------|---------|-------------|------------|---------|----------------|----------------|
| 16 | Credit Card Ref              | yes     | independent | ordinal    | 0.05    |                |                |
| 17 | Region                       | yes     | independent | nominal    | 0.05    | 50             |                |
| 18 | Avg Monthly Balance          | yes     | independent | continuous | 0.05    | 10             | static         |
| 19 | Total Amount Purchased       | yes     | independent | continuous | 0.05    | 10             | static         |
| 20 | Total Service Charge Revenue | yes     | independent | continuous | 0.05    | 10             | static         |
| 21 | Profit                       | yes     | independent | continuous | 0.05    | 10             | static         |
| 22 | Delinquent Lev 1             | yes     | independent | ordinal    | 0.05    |                |                |
| 23 | Delinquent Lev 2             | yes     | independent | ordinal    | 0.05    |                |                |
| 24 | Delinquent Lev 3             | yes     | independent | ordinal    | 0.05    |                |                |
| 25 | Amount Lost                  | yes     | independent | continuous | 0.05    | 10             | static         |
| 26 | age                          | yes     | independent | continuous | 0.05    | 10             | static         |
| 27 | Residential Status           | yes     | independent | nominal    | 0.05    | 5              |                |
| 28 | Status                       | yes     | independent | nominal    | 0.05    | 10             |                |
| 29 | Status_Weight                | yes     | weight      |            |         |                |                |

OK Cancel Help

Note that any existing splits in the tree will be deleted after you specify weight.

If the weight is defined, the letter “W” shown in the top left corner of the Tree view serves as an indicator that the tree is weighted. In addition to the Attribute Editor, the information about the weight is available in the Parameters and Attributes tab of the tree.



Since weight in Altair Data Analytics applications is treated as a (generalized) frequency weight, the effect of setting a weight in trees is as follows:

The number of records in all tree nodes is recalculated based on the defined weight (equals the sum of the weight field over all records in each node).

NOTE: Non-integer node totals are possible, since the sum of the weight field over a node is not necessarily integer. By default, the node totals displayed in tree nodes are formatted as integer numbers, but you can change this setting by selecting Tools | Options and changing the format in the *Show Support (# of records)* field, for example, to 0.00 or #,##0.00 (for two digits after the decimal point).

If a node is produced by a Find Split (and therefore shows a DV distribution), the weighted distribution of the dependent variable is as follows:

If  $W$  is the weight variable,  $F_i$  is the  $i$ th value of the discrete dependent variable  $F$ , and  $SUM(W_i)$  is the sum of the weight field over all records in the node where  $F=F_i$ , then

- The weighted total number of records =  $SUM(W)$  (over the node)
- The weighted frequency for the value  $F_i$  =  $SUM(W_i)$  (over the node)
- The weighted percentage of the value  $F_i$  =  $100 * SUM(W_i) / SUM(W)$  (sums taken over the node)

The effect in other tree views: All charts in the Tree Chart and Profile Chart are calculated based on the defined weight. Also, note that the columns of Node Report may display non-integer values.

If the dependent variable is continuous, the average and standard deviation are weighted.

All node calculations are also calculated based on the defined weight.

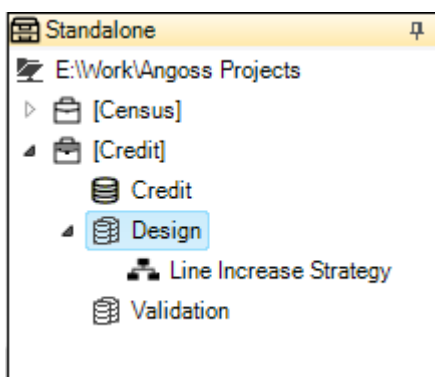
## Validating and Scoring a Strategy

Page | 468

### Strategy Validation

As a prerequisite for strategy development and subsequent validation, the dataset is usually partitioned into **Design** and **Validation** samples. The strategy is designed on the **Design** partition and then validated on the **Validation** partition.

The purpose of **Strategy Validation** is to monitor the changes in KPIs in the data segments defined by a given Strategy Tree. More precisely, it is to compare the values of KPIs expressed by [node calculations](#) in the development dataset and a validation (testing) dataset. The operation produces a *Strategy Validation Tree* whose nodes show the original KPI values, the KPI values calculated on the validation sample, and the comparison metric requested by the user.



Strategy validation is used to:

- assess the statistical validity of the strategy design before going into production
- assess if a strategy that has passed the testing or early stage of production meets the intent of the original design
- assess if the original strategy design assumptions are still valid after using the strategy in production for some time
- assess the performance of a strategy on a node-by-node basis

Strategy validation results are displayed in the form of a validation tree that combines information from the strategy trees applied to the design and validation samples along with user-defined comparison metrics.



| Name       | Design     | Validation | Comparison |
|------------|------------|------------|------------|
| Bad_Rate   | 5.04%      | 4.81%      | -0.23%     |
| Count      | 8750       | 20,417.00  | 11,667.00  |
| Lost_Total | 385,097.35 | 879,676.01 | 494,578.66 |
| Profit_Avg | 9.91       | 8.26       | -1.65      |

Scorecard

[506, 556]

| Name       | Design     | Validation | Comparison |
|------------|------------|------------|------------|
| Bad_Rate   | 18.07%     | 15.35%     | -2.72%     |
| Count      | 736        | 1,687.00   | 951.00     |
| Lost_Total | 111,666.29 | 227,660.94 | 115,994.65 |
| Profit_Avg | -62.20     | -54.05     | 8.15       |

[556, 588]

| Name       | Design     | Validation | Comparison |
|------------|------------|------------|------------|
| Bad_Rate   | 5.59%      | 5.52%      | -0.07%     |
| Count      | 4509       | 10,443.00  | 5,934.00   |
| Lost_Total | 223,476.21 | 498,747.72 | 275,271.51 |
| Profit_Avg | 11.65      | 10.28      | -1.37      |

[588, 620]

| Name       | Design    | Validation | Comparison |
|------------|-----------|------------|------------|
| Bad_Rate   | 1.60%     | 1.79%      | 0.19%      |
| Count      | 3505      | 8,287.00   | 4,782.00   |
| Lost_Total | 49,954.85 | 153,267.35 | 103,312.50 |
| Profit_Avg | 22.81     | 18.41      | -4.40      |

The results of each of the node calculations from the **design** dataset and the **validation** dataset are displayed on the nodes. The comparison column shows you the results of the comparison, where the method of comparison is defined in the validation wizard. In the above example, the comparison method is the difference (**validation - design**).

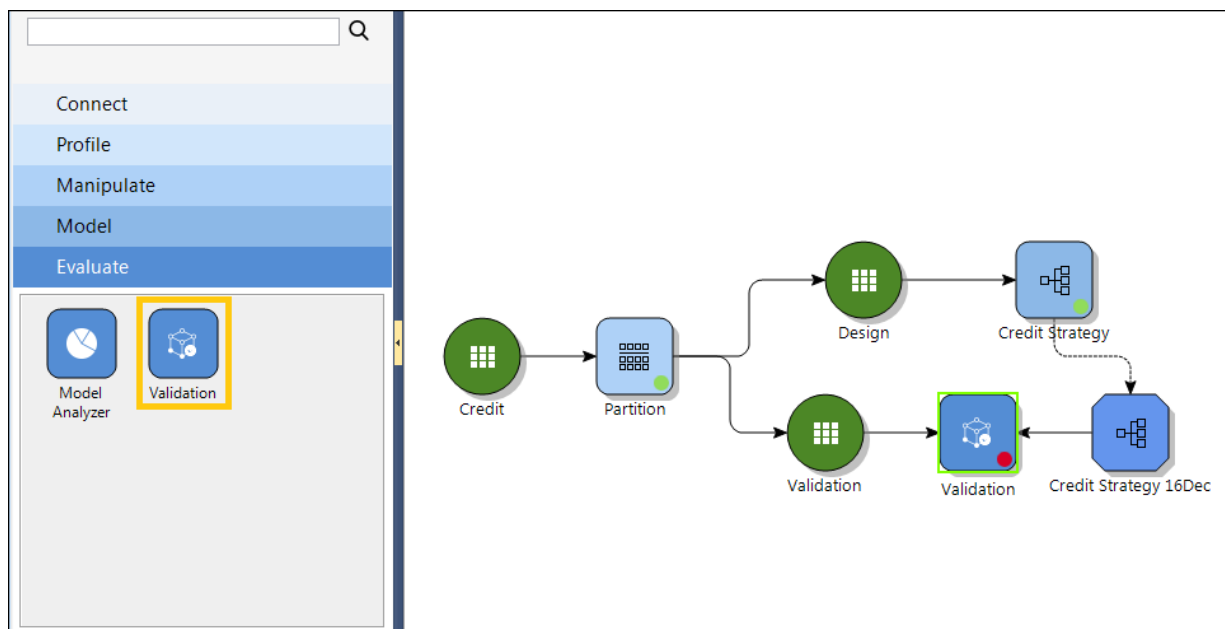
Proceed to the topic [Starting Strategy Validation](#).

The purpose of **Strategy Validation** is to monitor the changes in KPIs in the data segments defined by a given Strategy Tree. More precisely, it is to compare the values of KPIs expressed by [node calculations](#) in the development dataset and a validation (testing) dataset. The operation produces a *Strategy Validation Tree* whose nodes show the original KPI values, the KPI values calculated on the validation sample, and the comparison metric requested by the user.

Strategy Validation is applied to Strategy Tree [instances](#), since it is important to know what the state of the strategy tree was at the time of validation. If you apply validation to a Strategy Tree *process* node, a model instance will be automatically saved before validation, and the validation will be applied to the instance.

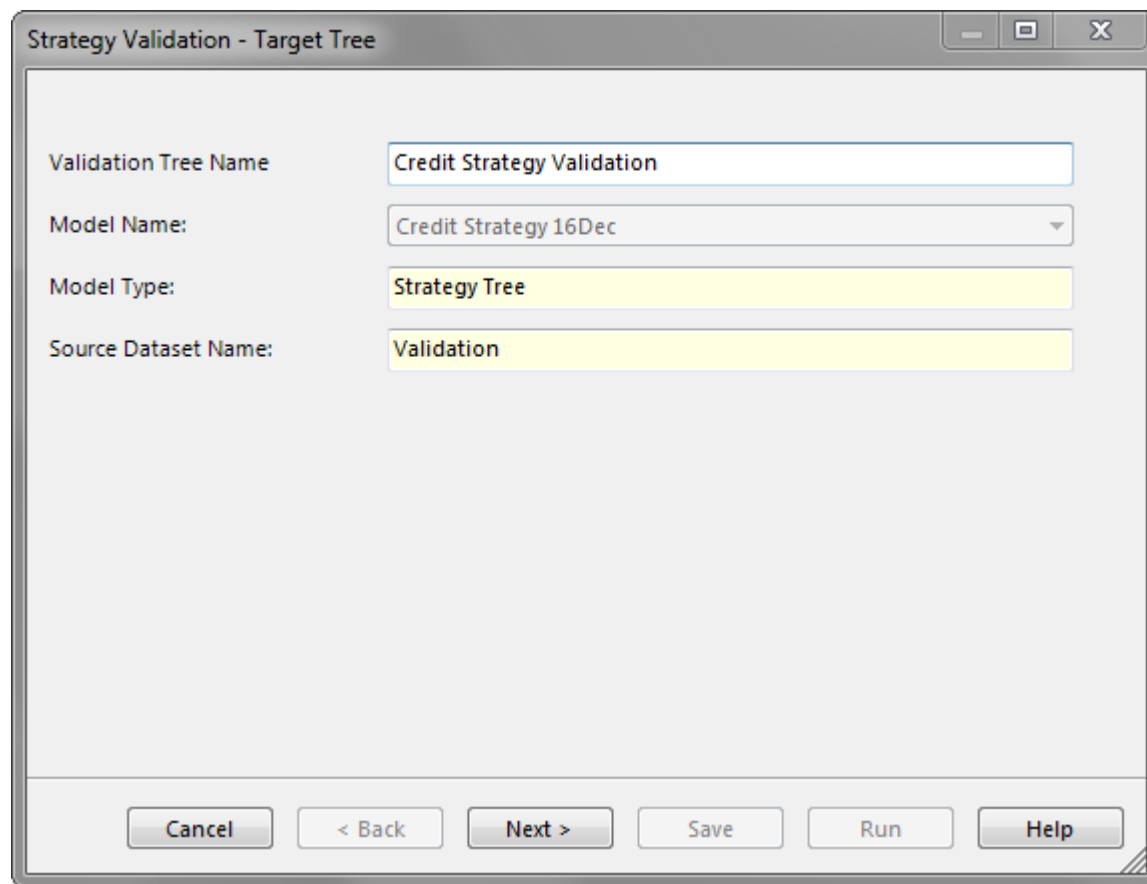
To validate a strategy, drag the **Validation** node from the **Evaluate** palette onto the Workflow canvas and connect the Strategy Tree node and the input dataset to it. If the Strategy Tree node is a process node (rather than a [model instance](#) node), you will be prompted to create a model instance first.

Alternatively, right-click anywhere on the workflow canvas and select the context menu command **Evaluate --> Validation**. This will create a new Model Validation node. Connect the input dataset and the Strategy Tree to it.



Double-click the **Validation** node or right-click on it and select **Modify** to open the wizard.

Specify the name for the output Strategy Validation Tree, which will be embedded in the Model Validation node:



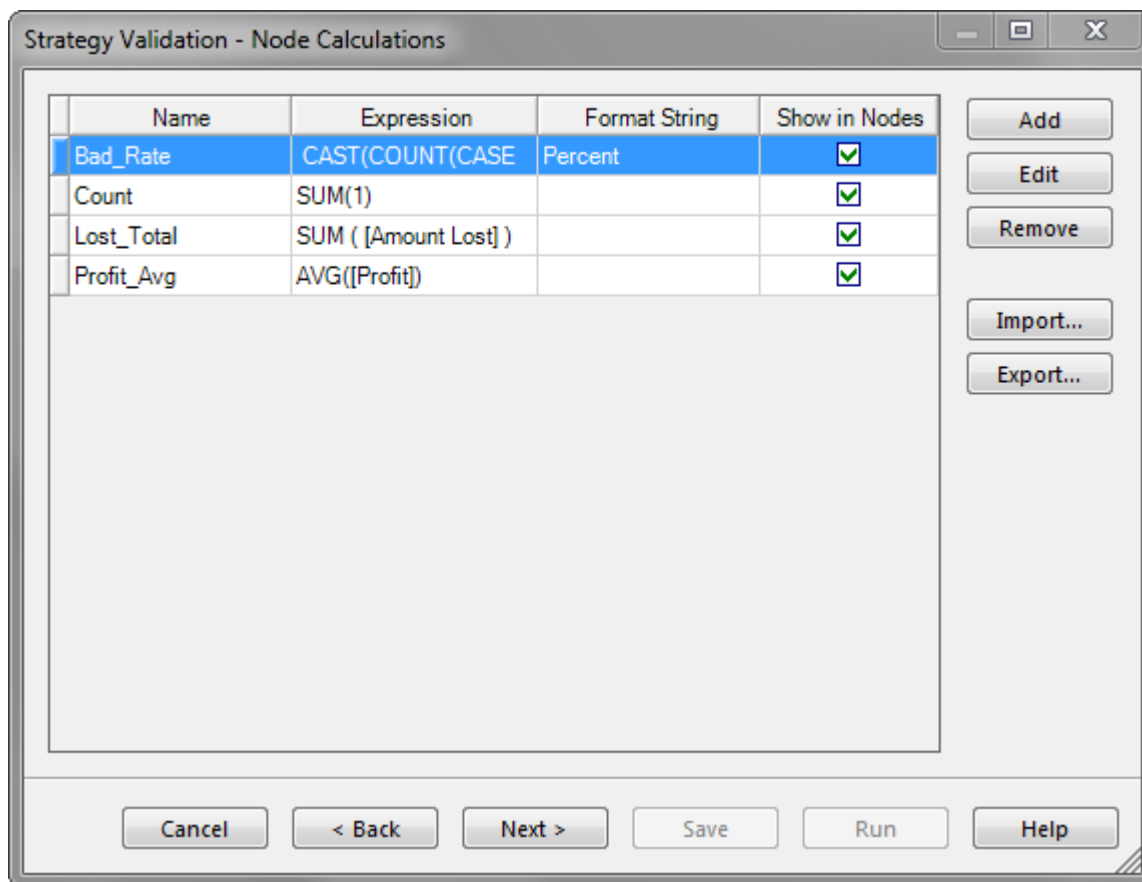
The dialog box titled "Strategy Validation - Target Tree" contains the following fields and buttons:

| Field                | Value                      |
|----------------------|----------------------------|
| Validation Tree Name | Credit Strategy Validation |
| Model Name:          | Credit Strategy 16Dec      |
| Model Type:          | Strategy Tree              |
| Source Dataset Name: | Validation                 |

Buttons at the bottom: Cancel, < Back, Next >, Save, Run, Help.

Click **Next** to proceed to the [Node Calculations](#) page to review or edit the existing KPI calculations or add new ones.

In the Node Calculations page, review the existing calculations, edit them or add new ones if necessary. You can also import calculations from other strategy trees or previously saved expression files.



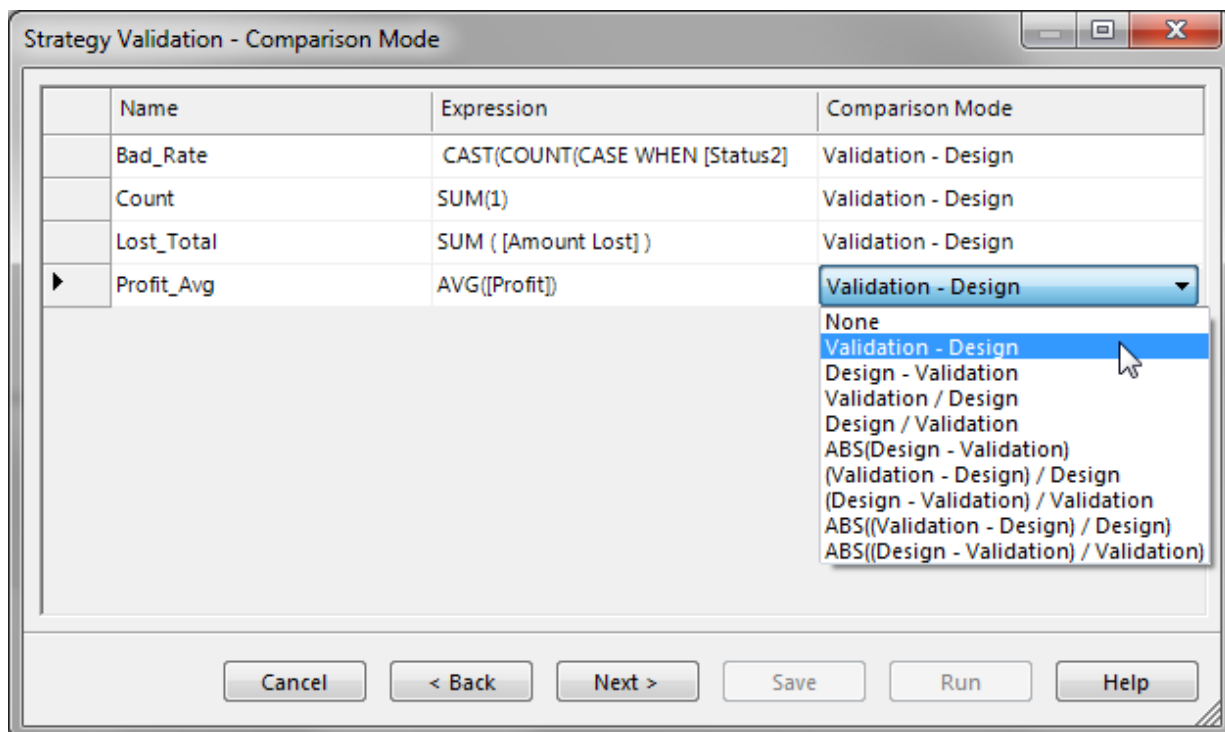
|                      |                                                                                                                        |
|----------------------|------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>          | The name of the calculation as it appears in the node of the strategy model.                                           |
| <b>Expression</b>    | SQL expression representing the node calculation.                                                                      |
| <b>Format String</b> | The format of the expression to be displayed in the node. See the topic <a href="#">Specifying Format</a> for details. |
| <b>Show in Nodes</b> | Choose whether to show or hide the calculation values in the tree nodes.                                               |

#### Operations:

|               |                                                                                                                              |
|---------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Add</b>    | Add a calculation to the node. This button calls the <a href="#">expression editor</a> to help in creating the calculations. |
| <b>Edit</b>   | Edit an existing node calculation in the expression editor.                                                                  |
| <b>Remove</b> | Remove a node calculation.                                                                                                   |

Click **Next** to proceed to the [Comparison Mode](#) page to define the type of comparison metric for the KPIs.

In the Comparison Mode page of the wizard, choose the desired comparison metric.

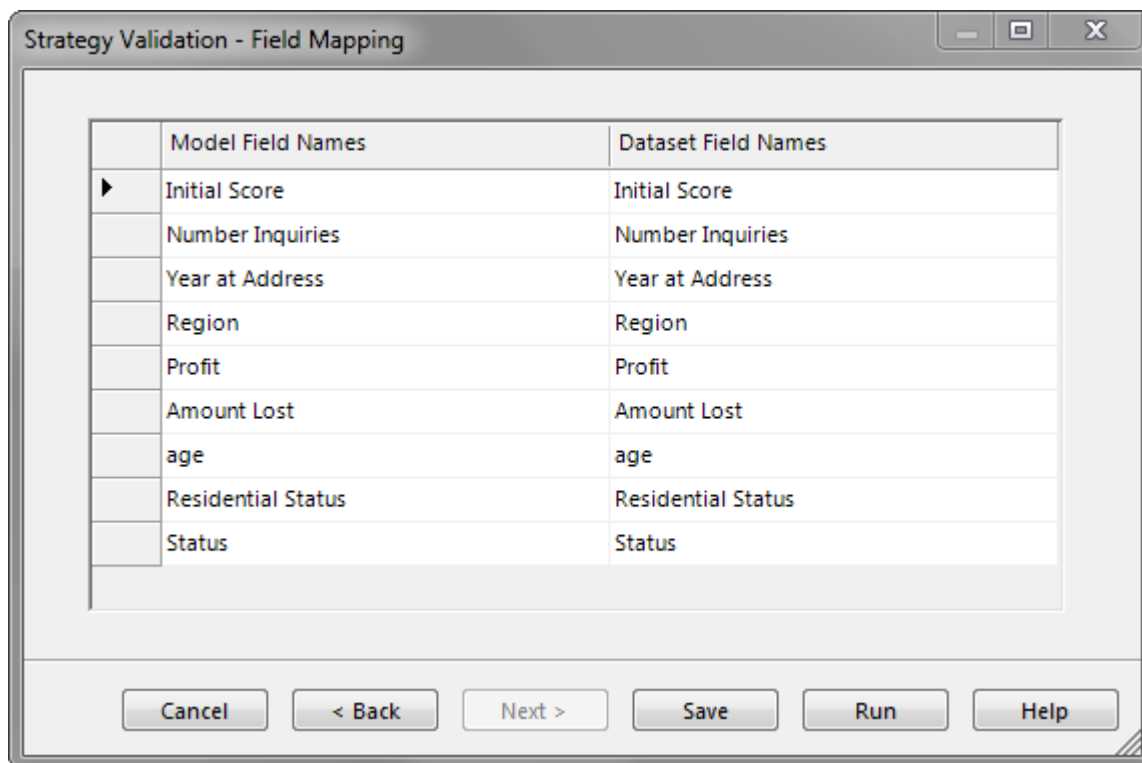


|                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>            | Name of the calculation as it appears in the node of the strategy model.                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Expression</b>      | SQL expression representing the node calculation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Comparison Mode</b> | <p>The following comparison modes are available:</p> <ul style="list-style-type: none"> <li>"None" does not perform any comparison</li> <li>"Validation - Design" subtracts the data in the learning dataset (Design) from the data in the test dataset (Validation)</li> <li>"Design - Validation" subtracts the data in the test dataset (Validation) from the data in the learning dataset (Design)</li> <li>"Validation / Design" divides the data in the test dataset (Validation) by the data in the learning dataset (Design)</li> </ul> |

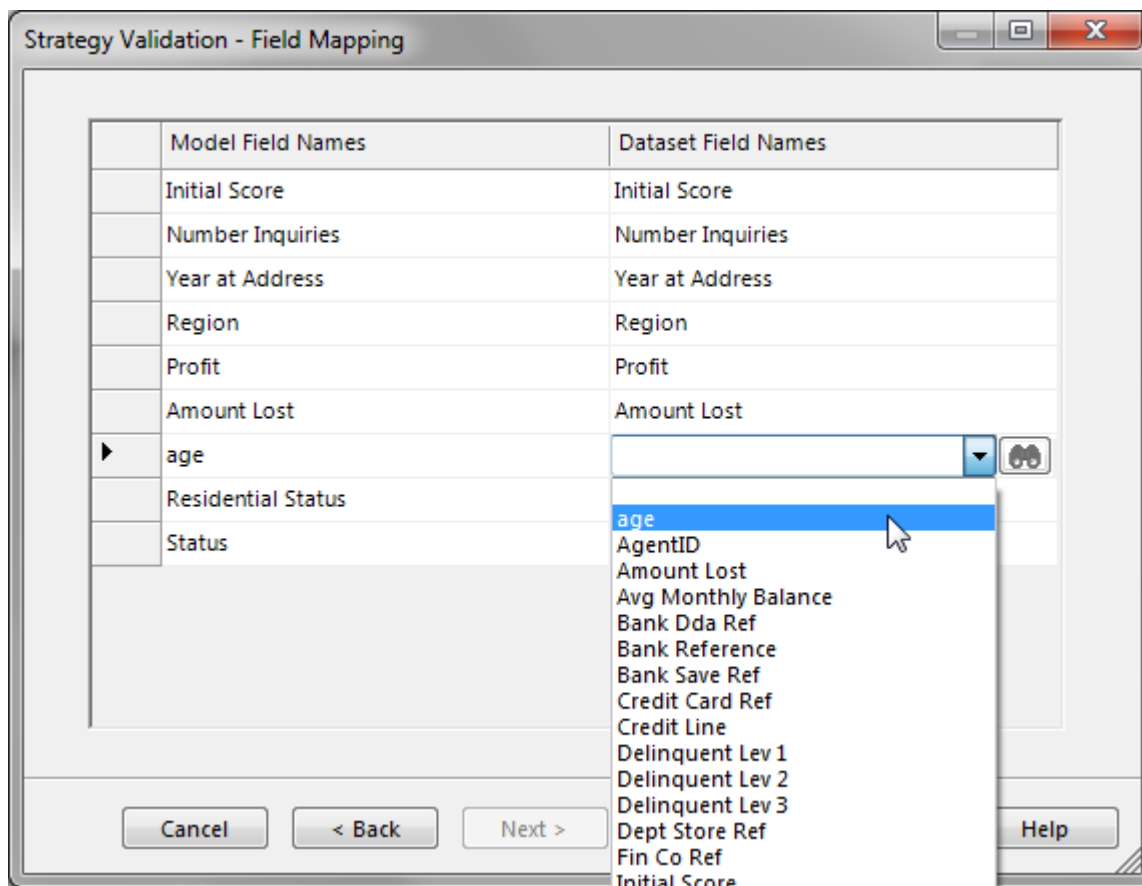
- "Design / Validation" divides the data in the learning dataset (Design) by the data in the test dataset (Validation)
- "ABS(Design - Validation)" takes the absolute value of the difference between the data in the learning dataset (Design) and the data in the test dataset (Validation)
- "(Validation - Design) / Design": provides the ratio of the difference between the data in the test dataset (Validation) and in the learning dataset (Design), and the data in the learning dataset (Design).
- "ABS ((Validation - Design) / Design)" is the absolute value of the preceding expression.
- "(Design - Validation) / Validation": provides the ratio of the difference between the data in the learning dataset (Design) and in the test dataset (Validation), and the data in the test dataset (Validation).
- "ABS ((Design - Validation) / Validation)" is the absolute value of the preceding expression.

Click **Next** to proceed to the [Field Mapping](#) page to map the strategy variables to the input dataset fields if necessary.

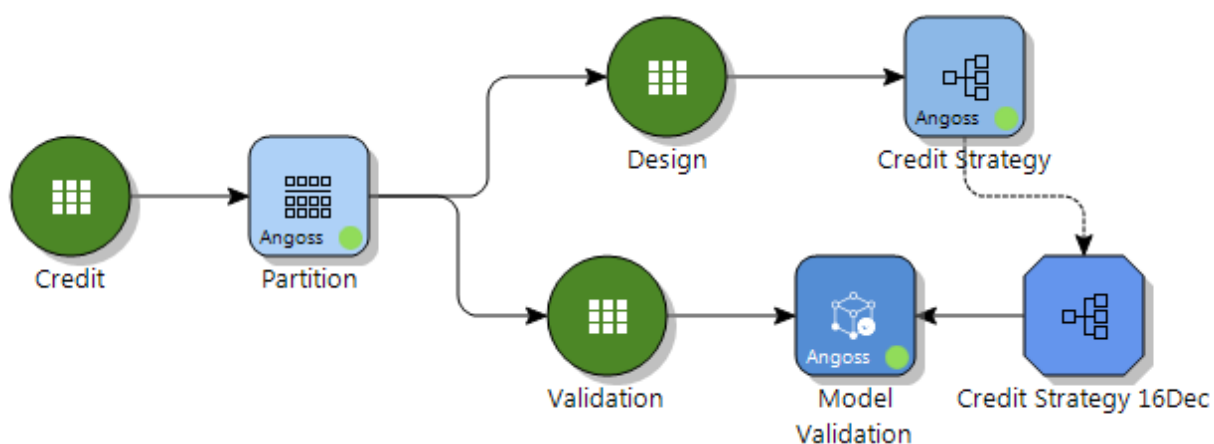
In the general case, the field names in the input dataset for strategy validation or scoring may be different from those in the original design dataset used to build the strategy tree. This step of the wizard allows you to map the fields if necessary. If identical or similar field names are found, they are matched automatically.



If there is a dataset field that is not mapped or is mapped incorrectly, click in the corresponding cell and select the right field as shown below.



Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately. After running the operation, double-click the **Model Validation** node to view the results.



Proceed to the [Validation Tree View](#) topic.



The default view displayed when you open the Model Validation node after Strategy Validation is a report that summarizes the validation inputs:

- Date and time of execution
- Input model name
- Input dataset name
- KPIs (expressions) defined
- Comparison metric (mode) for each KPI

### Strategy Validation Report

Input Model Name: Credit Strategy 16Dec

Input Dataset: Validation

Created: Saturday, May 06, 2017 07:33:14 PM



| Comparison Information |                                                                                                         |                     |
|------------------------|---------------------------------------------------------------------------------------------------------|---------------------|
| Expression Name        | Expression Value                                                                                        | Comparison Mode     |
| Bad_Rate               | CAST(COUNT(CASE WHEN [Status2] = 'Bad Debt' THEN 1 ELSE null END) AS double) / CAST(COUNT(*) AS double) | Validation - Design |
| Count                  | SUM(1)                                                                                                  | Validation - Design |
| Lost_Total             | SUM ( [Amount Lost] )                                                                                   | Validation - Design |
| Profit_Avg             | AVG([Profit])                                                                                           | Validation - Design |

© COPYRIGHT 2017 ANGOSS SOFTWARE CORPORATION
Created: Saturday, May 06, 2017 07:33:14 PM

[Report](#) | [Tree](#) | [Tree Map](#) | [Node Design Data](#) | [Node Validation Data](#) | [Node Report](#) | [Profile Chart](#)

The *Tree* view of the Model Validation node, also referred to as the Strategy Validation Tree, displays the results of the node calculations from both the Design and Validation datasets, along with the values of the KPI comparison metric.

| Name       | Design     | Validation | Comparison |
|------------|------------|------------|------------|
| Bad_Rate   | 5.04%      | 4.81%      | -0.23%     |
| Count      | 8750       | 20,417.00  | 11,667.00  |
| Lost_Total | 385,097.35 | 879,676.01 | 494,578.66 |
| Profit_Avg | 9.91       | 8.26       | -1.65      |

Scorecard

[506, 556]

| Name       | Design     | Validation | Comparison |
|------------|------------|------------|------------|
| Bad_Rate   | 18.07%     | 15.35%     | -2.72%     |
| Count      | 736        | 1,687.00   | 951.00     |
| Lost_Total | 111,666.29 | 227,660.94 | 115,994.65 |
| Profit_Avg | -62.20     | -54.05     | 8.15       |

[556, 588]

| Name       | Design     | Validation | Comparison |
|------------|------------|------------|------------|
| Bad_Rate   | 5.59%      | 5.52%      | -0.07%     |
| Count      | 4509       | 10,443.00  | 5,934.00   |
| Lost_Total | 223,476.21 | 498,747.72 | 275,271.51 |
| Profit_Avg | 11.65      | 10.28      | -1.37      |

[588, 620]

| Name       | Design    | Validation | Comparison |
|------------|-----------|------------|------------|
| Bad_Rate   | 1.60%     | 1.79%      | 0.19%      |
| Count      | 3505      | 8,287.00   | 4,782.00   |
| Lost_Total | 49,954.85 | 153,267.35 | 103,312.50 |
| Profit_Avg | 22.81     | 18.41      | -4.40      |

|                   |                                                                                                                                                                                  |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>       | The node calculation name (KPI name).                                                                                                                                            |
| <b>Design</b>     | The value of the node calculation in the original strategy model based on the Design dataset.                                                                                    |
| <b>Validation</b> | The value of the node calculation in the strategy model based on the Validation dataset.                                                                                         |
| <b>Comparison</b> | The value of the comparison metric (comparison between the KPIs calculated in the segments of the Design and Validation datasets), as defined in the Strategy Validation wizard. |

The validation tree node report provides you with information about the validation results. As with the strategy tree node report, only the results for the terminal nodes are displayed

☐ Tagged nodes only

| Node Rules                                | Design - Bad_Rate | Validation - Bad_Rate | Comparison - Bad_Rate | Design - Count | Validation - Count |
|-------------------------------------------|-------------------|-----------------------|-----------------------|----------------|--------------------|
| ((Scorecard)>=463 AND (Scorecard)<532)    | 34.82 %           | 33.39 %               | -1.43 %               | 1143           | 1141               |
| ((Scorecard)>=532 AND (Scorecard)<551)    | 14.69 %           | 14.66 %               | -0.03 %               | 1484           | 1453               |
| ((Scorecard)>=551 AND (Scorecard)<611)    | 5.07 %            | 6.02 %                | 0.94 %                | 1419           | 1512               |
| ((Scorecard)>=611 AND (Scorecard)<665)    | 1.03 %            | 0.72 %                | -0.31 %               | 1555           | 1534               |
| ▶ ((Scorecard)>=665 AND (Scorecard)<=790) | 0.12 %            | 0.15 %                | 0.02 %                | 8982           | 8943               |

Report | Tree | Tree Map | Node Design Data | Node Validation Data | **Node Report** | Profile Chart

Columns can be added and removed from the report by using the Options.

**Options**

**Node Report**

Customize Font  
Grid Font: Arial 8.25pt

☐ Display rule with a column per IV

Customize Columns

| Column Name           | Visible                             | Precision |
|-----------------------|-------------------------------------|-----------|
| Node ID               | <input type="checkbox"/>            |           |
| Node Number           | <input type="checkbox"/>            |           |
| Treatments            | <input checked="" type="checkbox"/> |           |
| Design - Bad_Rate     | <input checked="" type="checkbox"/> |           |
| Validation - Bad_Rate | <input checked="" type="checkbox"/> |           |
| Comparison - Bad_Rate | <input checked="" type="checkbox"/> |           |
| Design - Count        | <input checked="" type="checkbox"/> |           |

Apply  
Reset  
OK  
Cancel  
Help

The report can be sorted on a particular column. For example, you want to see all the nodes that have a treatment.

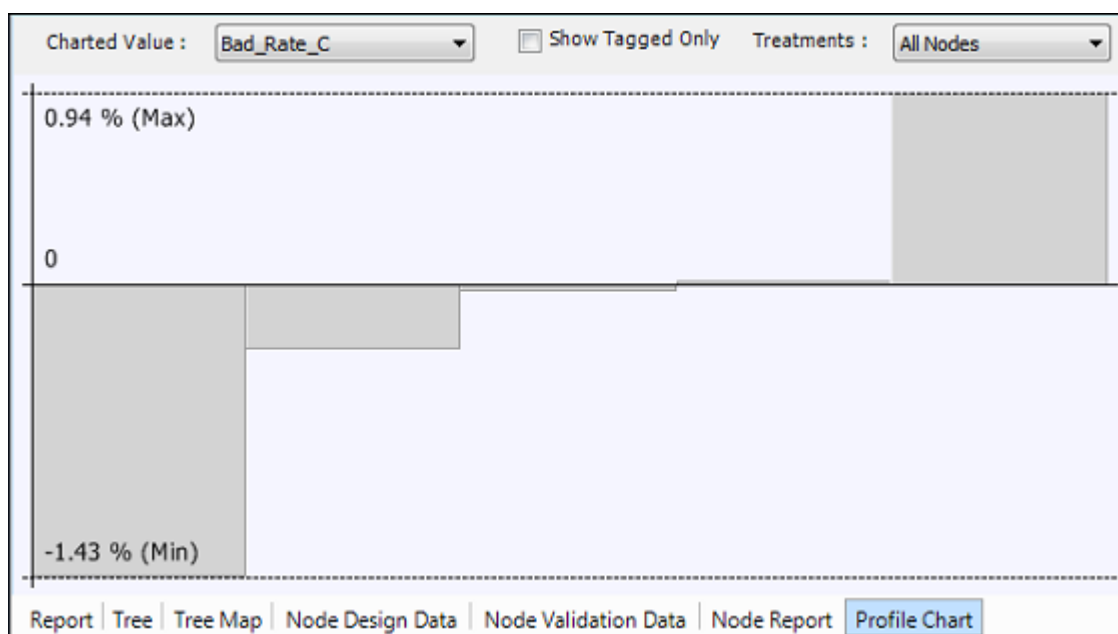
To sort the columns, click on the column heading.

## Treatment Summary

There is a summary of the treatment nodes located at the bottom of the report. This gives you a quick report on the total number of records that have been labeled with a given treatment.

Page | 480

The strategy validation **Profile Chart** view displays each terminal node in tree. Its purpose is to display the tree in a single chart. It differs from the design strategy tree model profile chart in that the charted values available are the node calculations from both the design and the validation.



Clicking on a bar will cause the tree to navigate to that node. You can also right click to access commands such as assigning treatments and tagging nodes.

### **Charted Value**

The charted value presents a list of each of the tree calculations from both the design and the validation trees. Choose the value that you would like charted.

### **Show Tagged Treatments by Default**

If this is checked, only nodes that are tagged are displayed. By default, all terminal nodes are displayed. Changing the treatments option allows you to plot only nodes that are assigned to a specific treatment.

### **Support as Width**

When checked, the width of the bar represents the number of observations in the node. If unchecked, each bar is the same width regardless of the number of rows.

---

## Strategy Scoring

**Scoring a strategy** is the process of assigning treatments to the records of a new dataset using the rules and treatment assignments in a Strategy Tree.

Page | 481

If the Strategy Tree has node calculations, you also have an option to assign the calculation values to each record being scored.

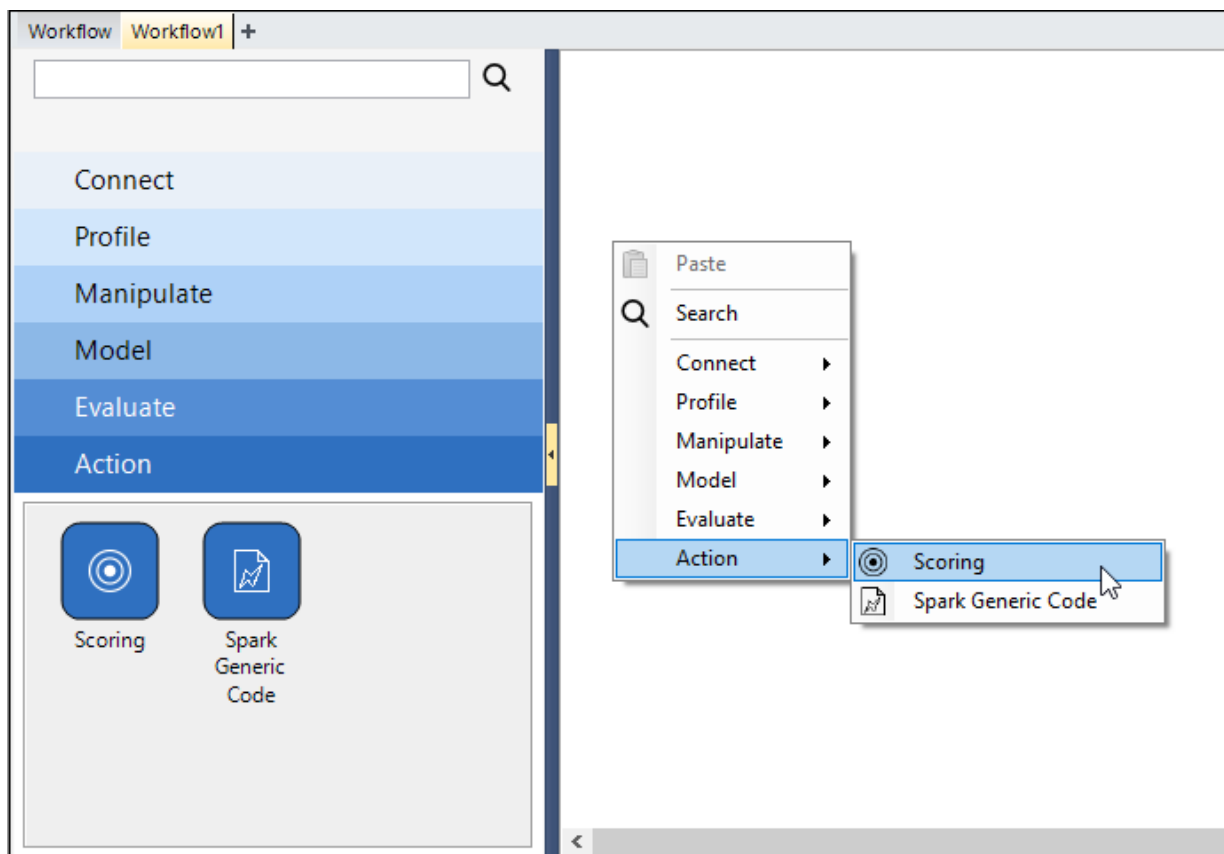
The values of treatments and calculations assigned to the input records in the process of strategy scoring are written to the corresponding new fields of the output dataset.

### Use case examples:

- Strategy scoring can be used to apply strategies to test data for analysis and reporting with the purpose of estimating their effects. For example, after applying a strategy that assigns credit limit increase amounts, you can compute new derived variables such as the new (revised) credit limit, estimated future utilization, interest income and write-offs, and then display them in a new strategy tree or simply report the total effect. This may be helpful in explaining the benefits and risks of a new strategy.
- Treatment lists generated by scoring a strategy can be used by an operational application. A list of actions generated by deploying a collections strategy can be processed by a call center operator. A list of actions generated by deploying a marketing campaign strategy can help build separate lists of customers or prospects to be contacted via specific channels (Email, phone, mail) defined by the strategy.

The strategy scoring process reads the input dataset and associates each input record with a strategy tree node according to the rules defined by the tree splits. The treatment and the KPI found in the node (if any) are assigned to the record.

As with any other Knowledge Studio model type, Scoring is performed using the **Scoring** node from the **Action** palette. See the topic [Starting the Score Strategy Wizard](#).



### See also: [Starting the Score Strategy Wizard](#)

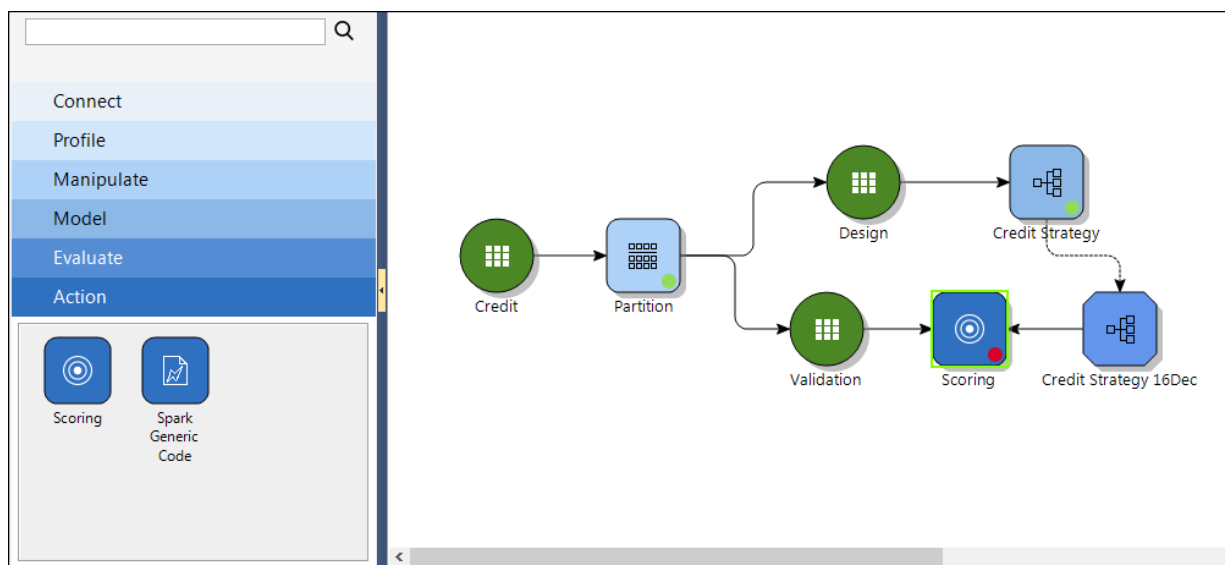
For general information about the notion and the purpose of strategy scoring, see the topic [Scoring a Strategy](#).

As with any other Knowledge Studio model type, Scoring is performed using the **Scoring** node from the **Action** palette.

Strategy scoring is applied to Strategy Tree [instances](#), since it is important to know what the state of the strategy tree was at the time of scoring. If you apply scoring to a Strategy Tree *process* node, a model instance will be automatically saved before scoring, and the scoring will be applied to the instance.

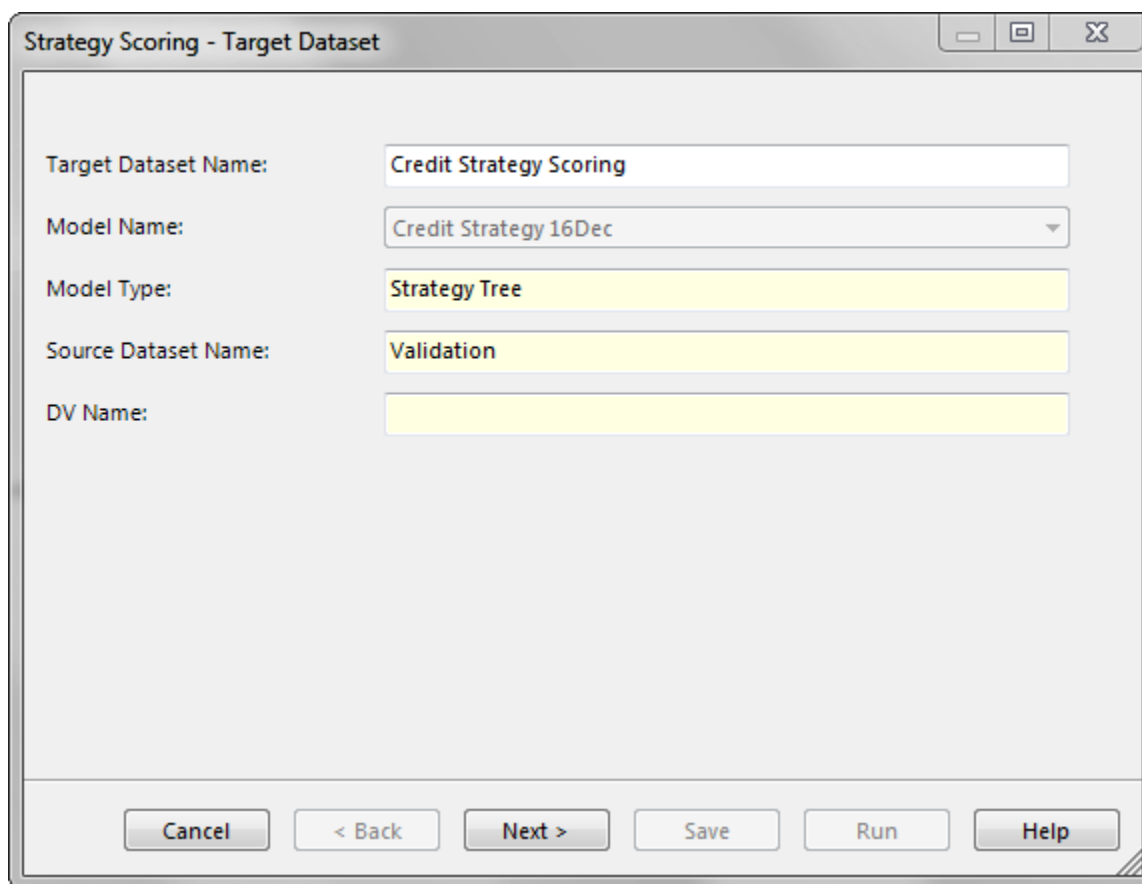
To score a strategy, drag the **Scoring** node from the **Action** palette onto the Workflow canvas and connect the Strategy Tree node and the input dataset to it. If the Strategy Tree node is a process node (rather than a [model instance](#) node), you will be prompted to create a model instance first, in which case you need to specify the name for the tree instance.

Alternatively, right-click anywhere on the workflow canvas and select the context menu command **Action → Scoring**. This will create a new Scoring node. Connect the input dataset and the Strategy Tree to it.



Double-click the **Scoring** node or right-click on it and select **Modify** to open the wizard.

Specify the name for the output dataset.



Strategy Scoring - Target Dataset

Target Dataset Name: Credit Strategy Scoring

Model Name: Credit Strategy 16Dec

Model Type: Strategy Tree

Source Dataset Name: Validation

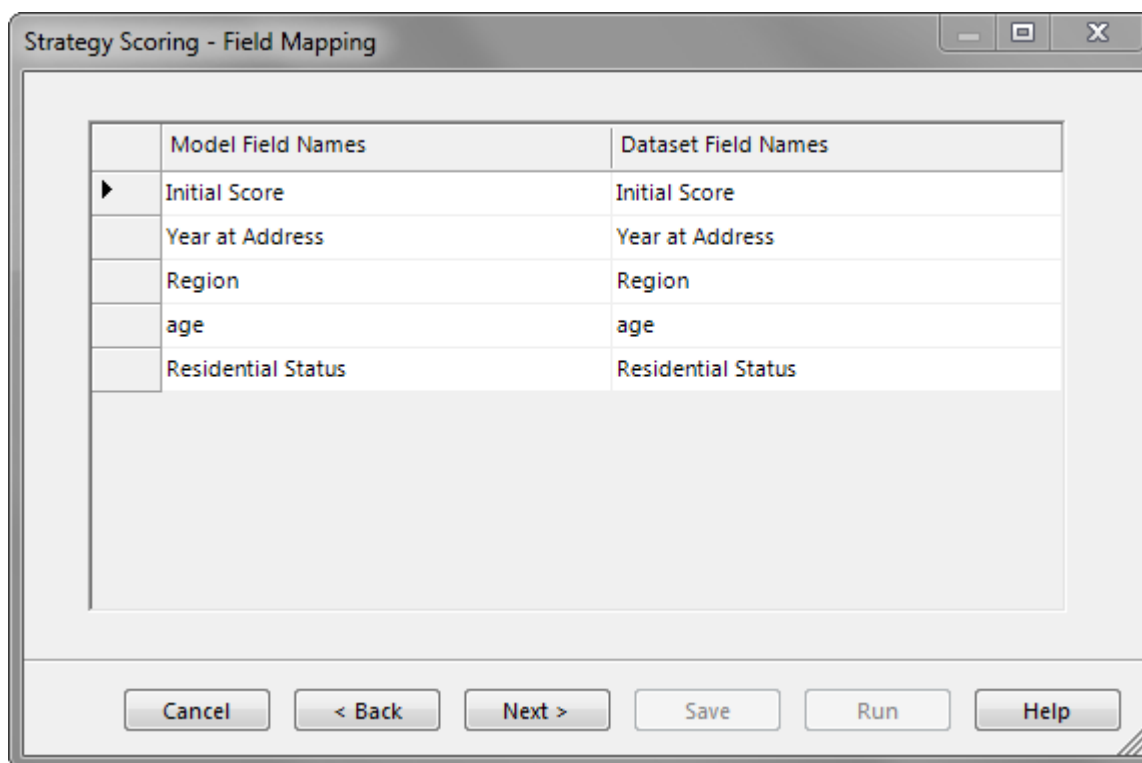
DV Name:

Cancel < Back Next > Save Run Help

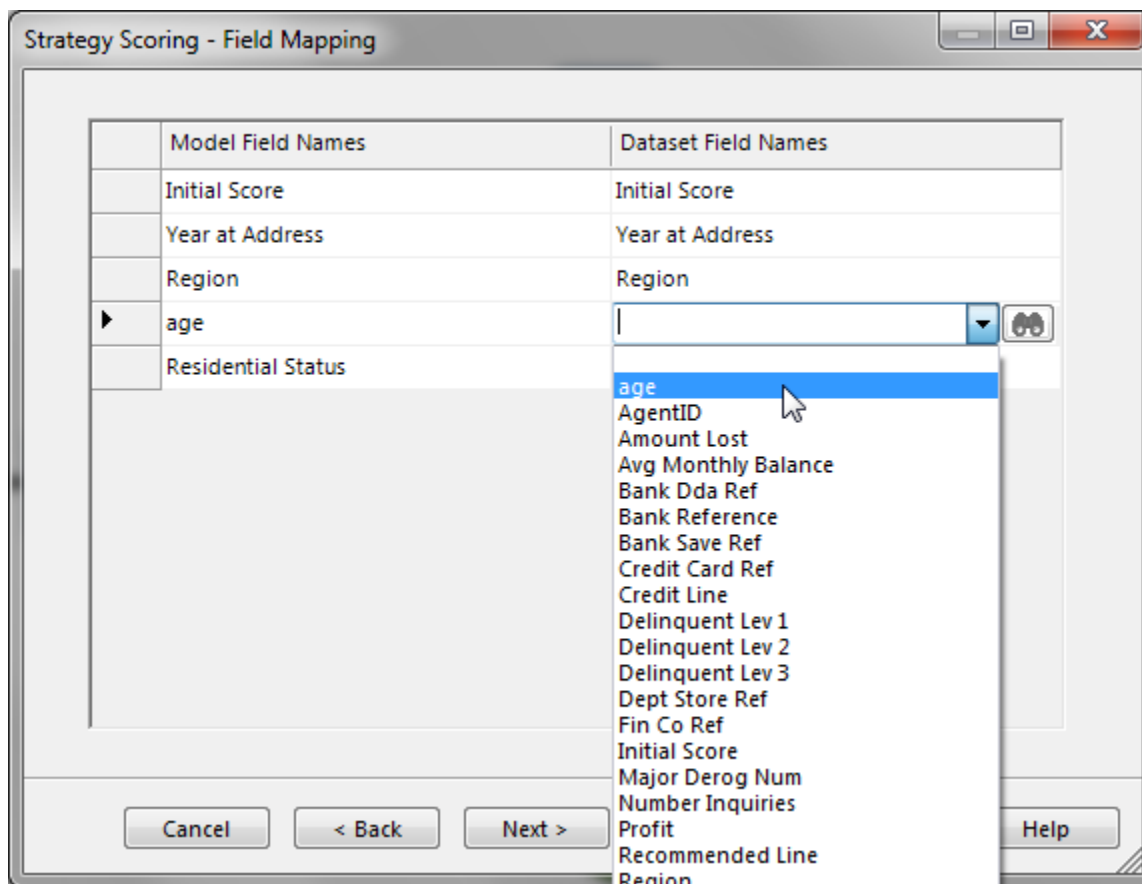
Proceed to the topic [Score Strategy: Field Mapping](#).

In the general case, the field names in the input dataset for strategy scoring may be different from those in the original design dataset used to build the strategy tree. This step of the wizard allows you to map the fields if necessary. If identical or similar field names are found, they are matched automatically.





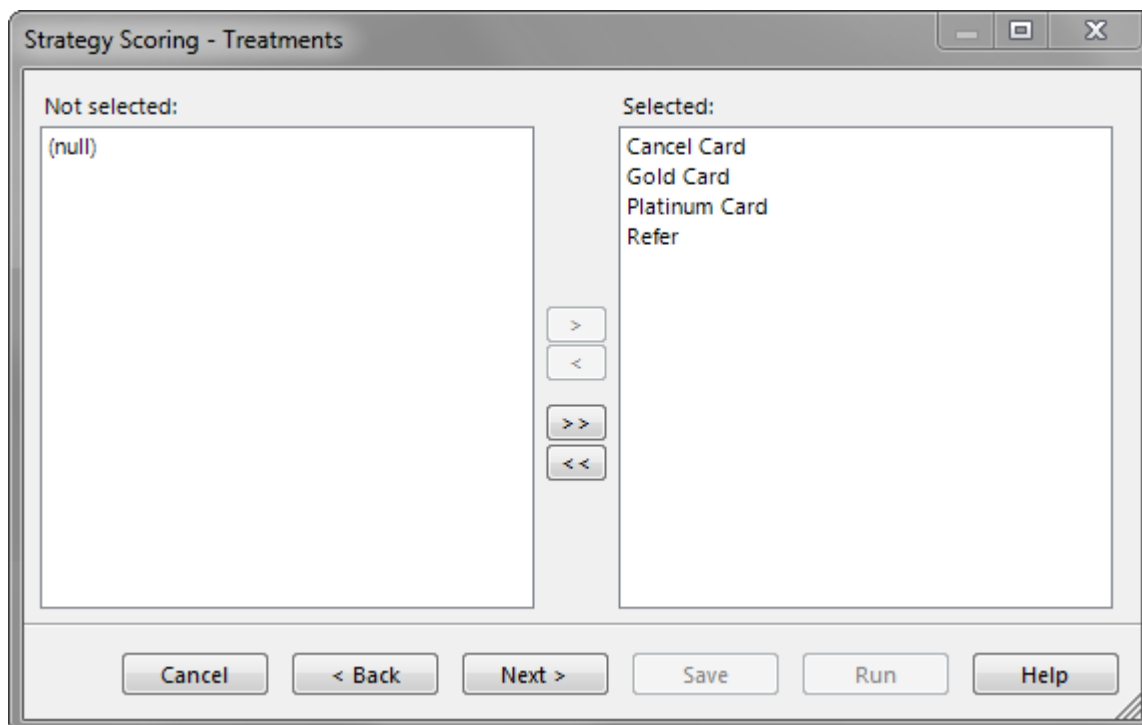
If there is a dataset field that is not mapped or is mapped incorrectly, click in the corresponding cell and select the right field as shown below.



Click **Next** to proceed to the [Treatments](#) page to select the desired Treatments.

Select one or more treatments from the list of all treatments found in the tree. By default, all treatments are selected for list generation. You can move a treatment from one list to the other using arrow buttons or double-clicking on the treatment name.

Note: (null) is one of the options for treatments. It is not selected by default. If it is selected, then nodes without a treatment will also be written in the output with a NULL in the treatment column.

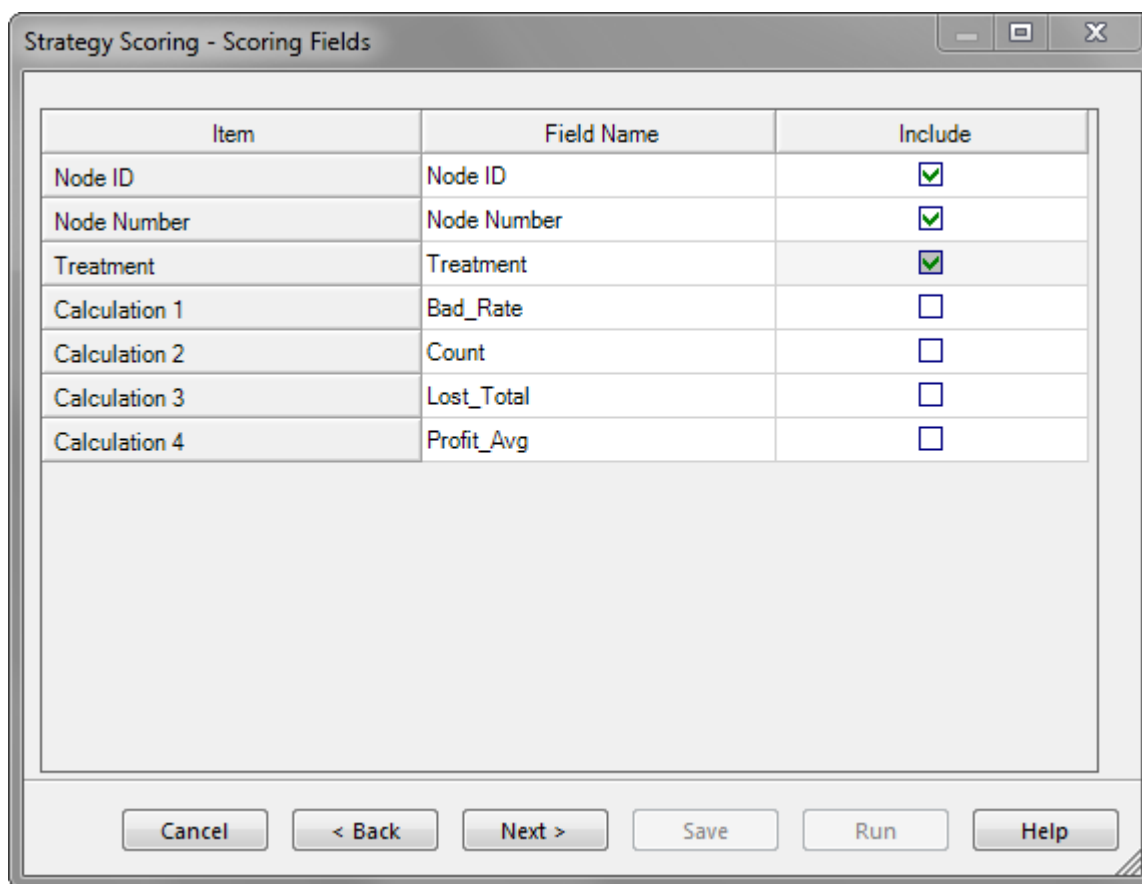


Click **Next** to proceed to the [Scoring Fields](#) page to select the new fields to be generated for the output dataset in addition to the Treatment field.

In the Scoring Fields page, select the fields to include in the scoring output dataset. The fields included by default are Treatment, Node ID, and Node Number.

Calculations are not included by default. Select the ones required in the output.

You can rename the fields if necessary. To rename a field, just edit it in the Field Name column.

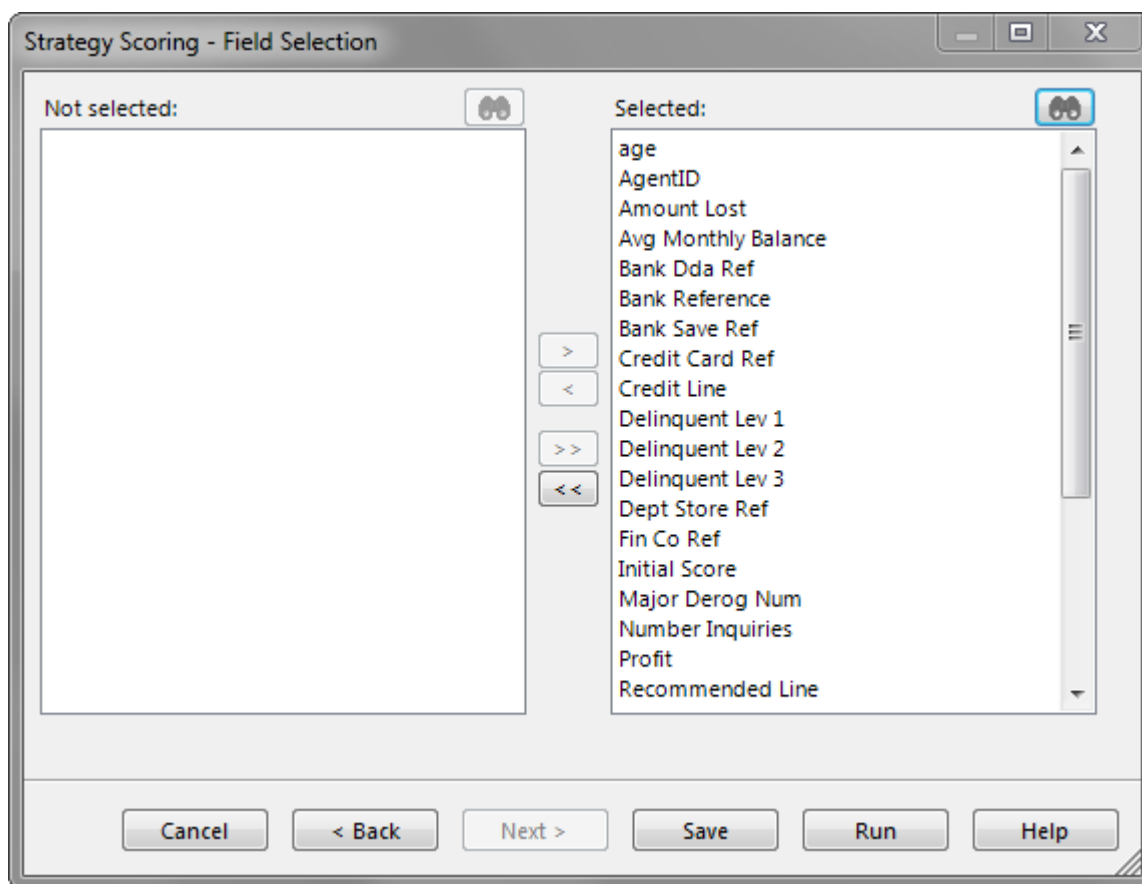


The dialog box titled "Strategy Scoring - Scoring Fields" contains a table with three columns: "Item", "Field Name", and "Include". The table lists several items, some of which are already selected (checked) in the "Include" column. Below the table are several buttons: "Cancel", "< Back", "Next >", "Save", "Run", and "Help".

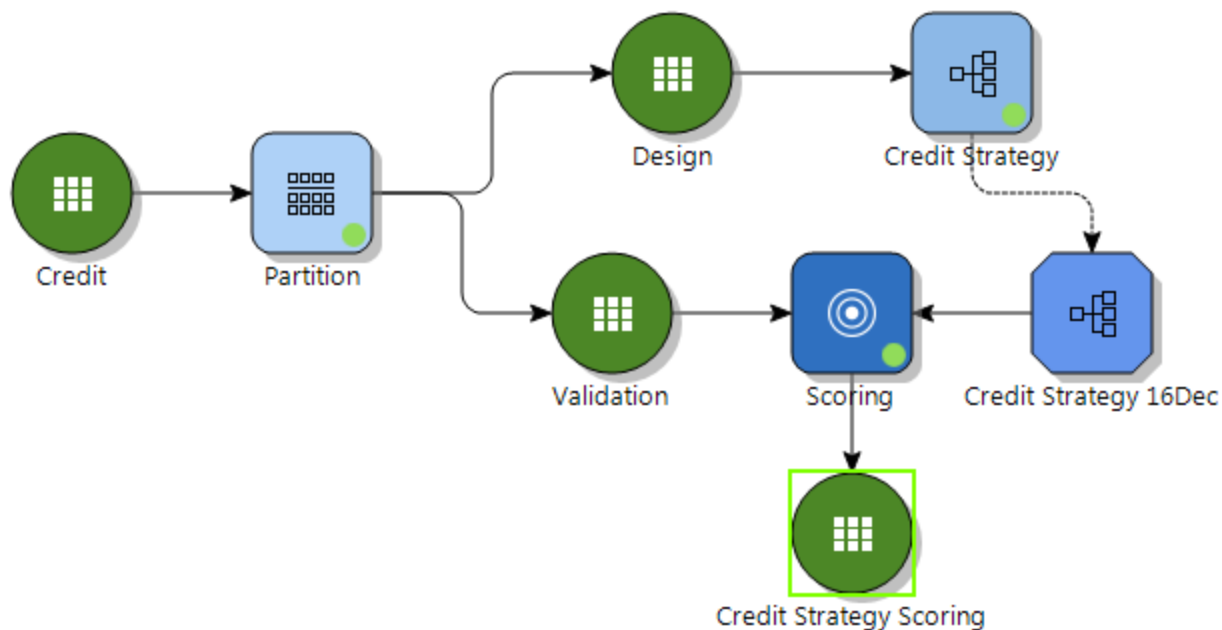
| Item          | Field Name  | Include                             |
|---------------|-------------|-------------------------------------|
| Node ID       | Node ID     | <input checked="" type="checkbox"/> |
| Node Number   | Node Number | <input checked="" type="checkbox"/> |
| Treatment     | Treatment   | <input checked="" type="checkbox"/> |
| Calculation 1 | Bad_Rate    | <input type="checkbox"/>            |
| Calculation 2 | Count       | <input type="checkbox"/>            |
| Calculation 3 | Lost_Total  | <input type="checkbox"/>            |
| Calculation 4 | Profit_Avg  | <input type="checkbox"/>            |

Click **Next** to proceed to the page [Fields To Include In The Output](#) to select the fields from the input dataset to include in the output.

At the last step of the Score Strategy Wizard, select the fields from the input dataset to include in the output. If you don't select any, the output will contain only the Treatment field and the new scoring fields you may have selected in the [Scoring Fields](#) page.



Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately. After running the operation, double-click the output dataset node to view the results.



## Boosted Classification

Unlike in Random Forest, in **Boosted Classification** the iterations are not independent from each other. Every tree is grown based on the input from the previous tree.

The general outline of the algorithm is as follows:

1. Generate a vector of weights
2. Generate a vector of alphas that is going to weight every tree
3. For  $i=1$  to the maximum number of iterations specified:
  - grow tree  $i$  weighted by the weight vector;
  - score the tree;
  - update the weight vector increasing the importance of the misclassified points;
  - generate the alpha for the tree based on the error ratio.
4. Return a set of trees and their corresponding alphas for weighting

The implementation is based on the AdaBoost loss function, where AdaBoost stands for "Adaptive Boosting".

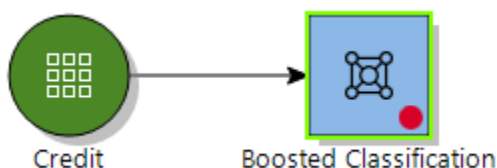
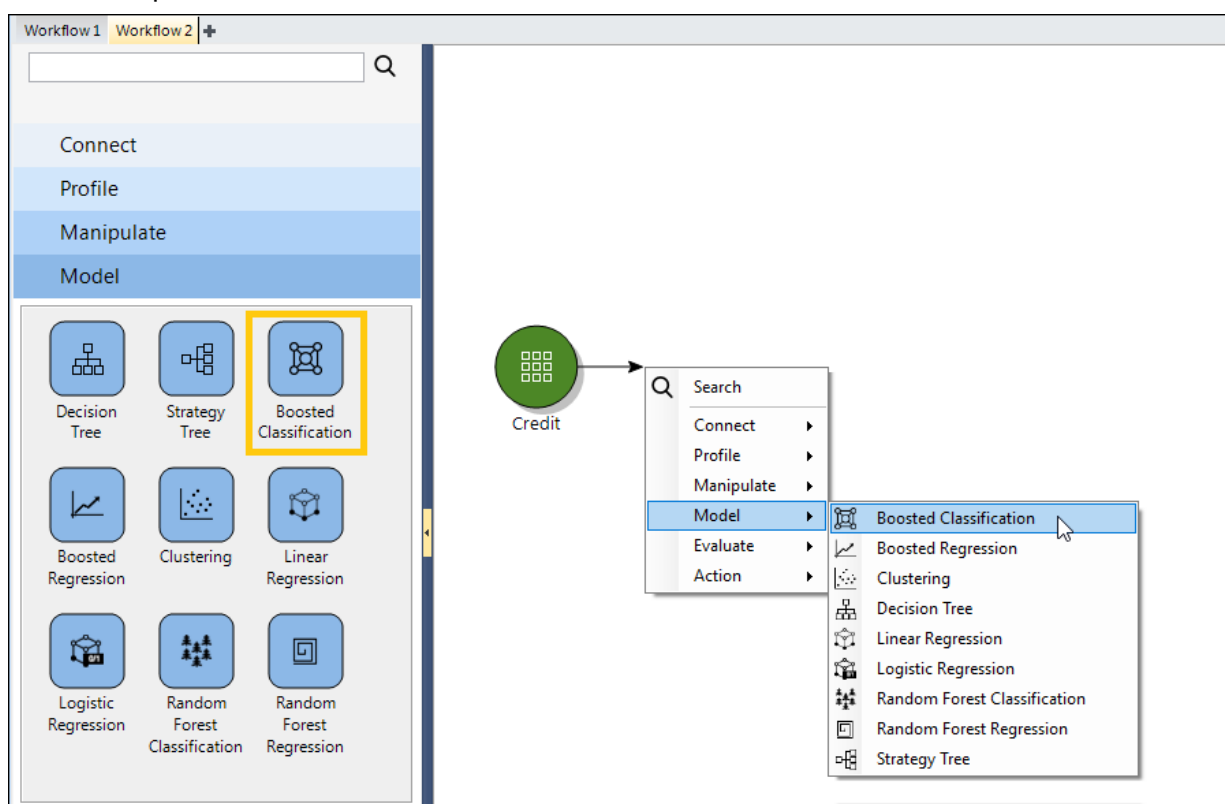
To build a Boosted Classification model on a Spark dataframe in your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the context menu, select **Model --> Boosted Classification** as in the example screenshot below. A new **Boosted Classification** process node will be created.

Page | 491

OR:

- Drag the **Boosted Classification** node from the **Model** palette onto the canvas and connect the input dataframe to it.

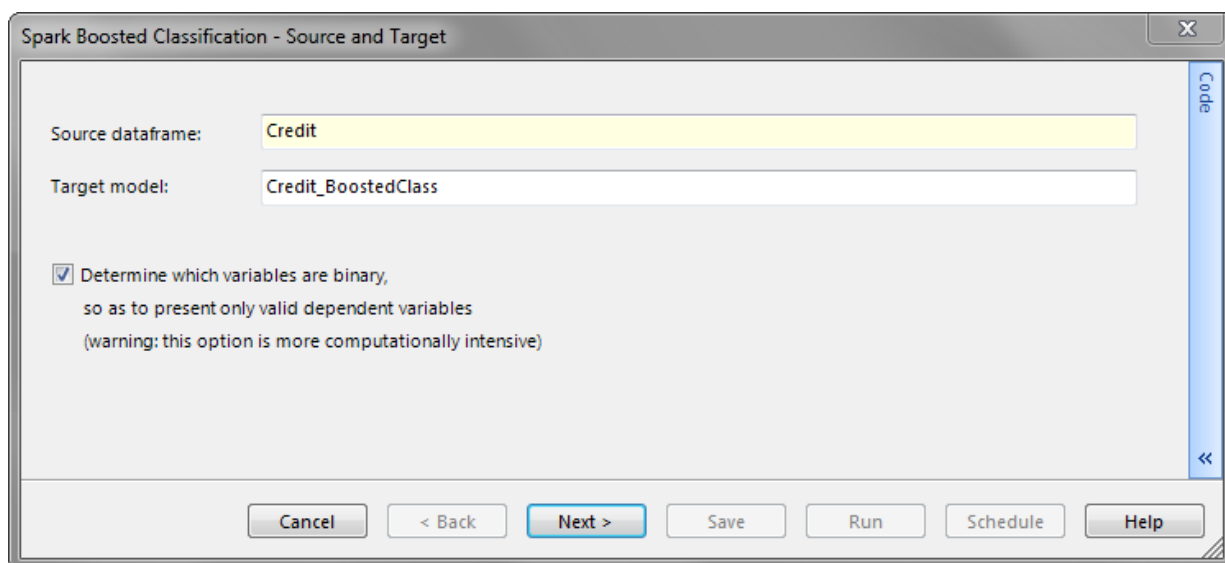


If the wizard does not open automatically, double-click the **Boosted Classification** node or right-click on it and select **Modify** to open the wizard.

In the first page of the Boosted Classification wizard:

- Specify the name for the model
- If necessary, choose the option to exclude constant variables and identify binary variables. (If you select this option, note that proceeding to the next page may take some time for large and wide dataframes.)

Page | 492



Click Next. In the next page:

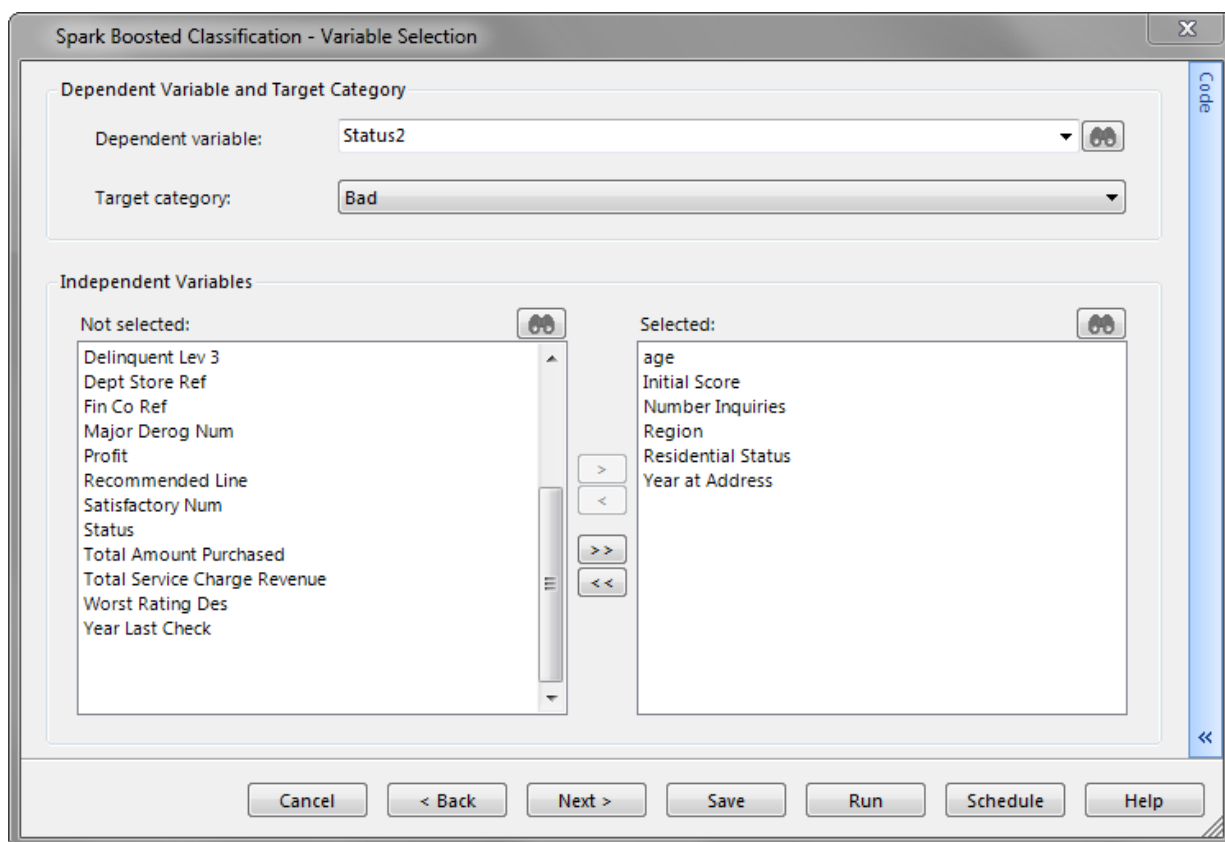
- Select the dependent variable and the target category. The dependent variable must be binary.
- Select the independent variables to include in the model.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.





Click Next. In the next page, specify the model parameters:

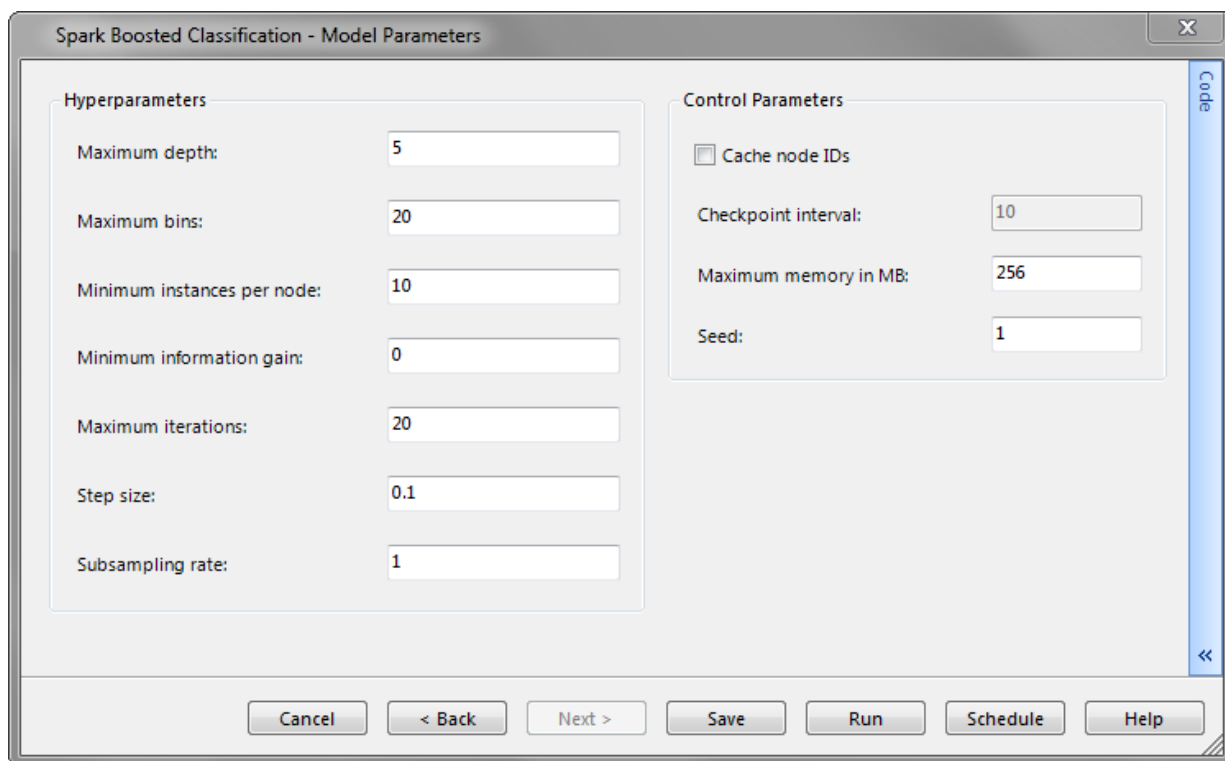
Hyperparameters:

- Maximum tree depth (number of levels, including the root node)
- Maximum number of bins in each split
- Minimum number of instances (records) per node
- Minimum information gain
- Checkpoint interval
- Maximum number of iterations
- Step size
- Seed (random seed)
- Subsampling rate

Control parameters:

- Cache node IDs (turned off by default)
- Checkpoint interval (enabled if the *Cache node ID* option is selected; the default is 10)

- Maximum memory in Megabytes (the default is 256 MB)
- Random seed



The dialog box titled "Spark Boosted Classification - Model Parameters" contains two main sections: "Hyperparameters" and "Control Parameters".

**Hyperparameters:**

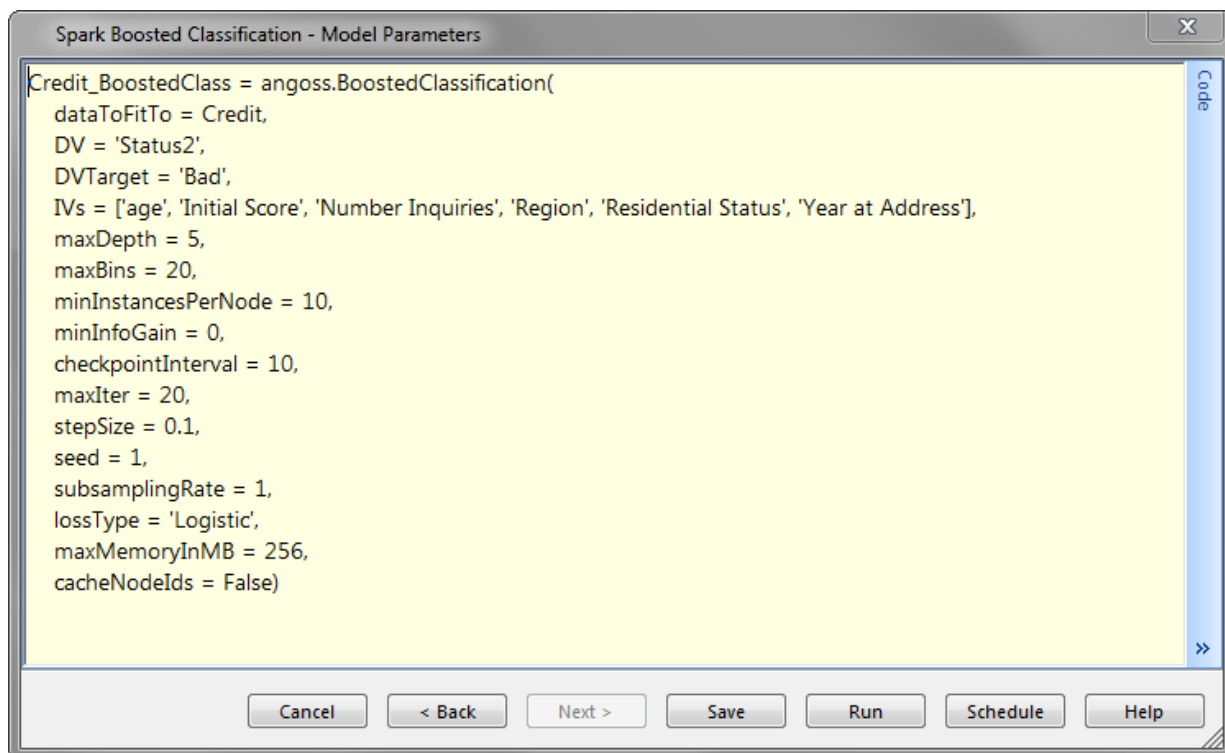
- Maximum depth: 5
- Maximum bins: 20
- Minimum instances per node: 10
- Minimum information gain: 0
- Maximum iterations: 20
- Step size: 0.1
- Subsampling rate: 1

**Control Parameters:**

- ☐ Cache node IDs
- Checkpoint interval: 10
- Maximum memory in MB: 256
- Seed: 1

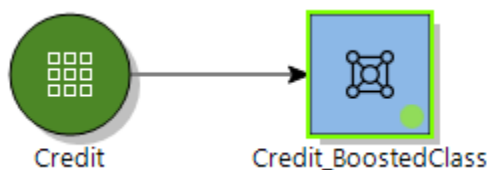
At the bottom of the dialog are buttons: Cancel, < Back, Next >, Save, Run, Schedule, and Help. On the right side, there is a blue vertical bar labeled "Code" with a double arrow icon at the bottom.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output model is created in the workflow after running the node.



Double-click the output model node to view the results.

## Boosted Regression

To build a Boosted Regression model on a Spark dataframe in your workflow:

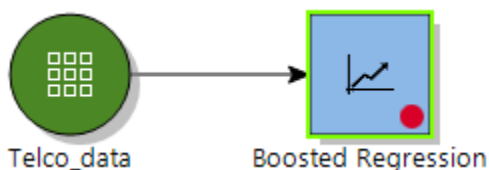
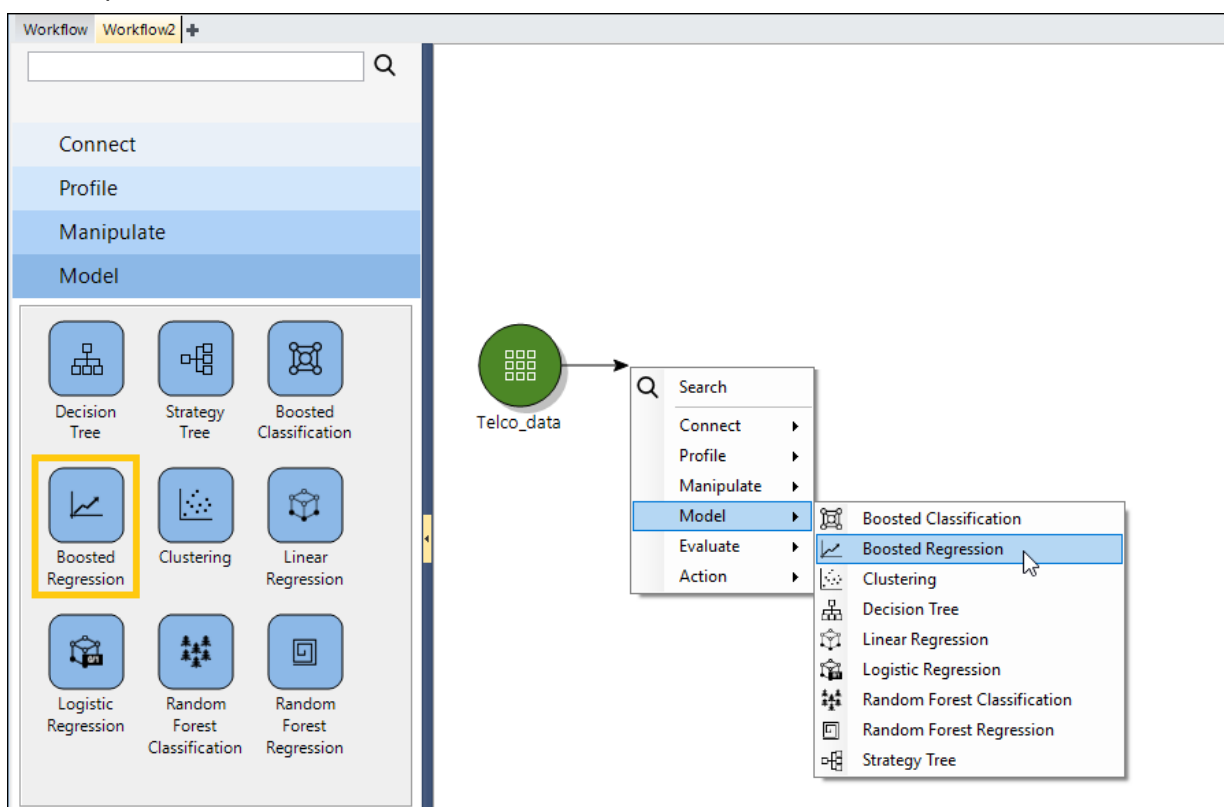
- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the

context menu, select **Model --> Boosted Regression** as in the example screenshot below. A new **Boosted Regression** process node will be created.

OR:

- Drag the **Boosted Regression** node from the **Model** palette onto the canvas and connect the input dataframe to it.

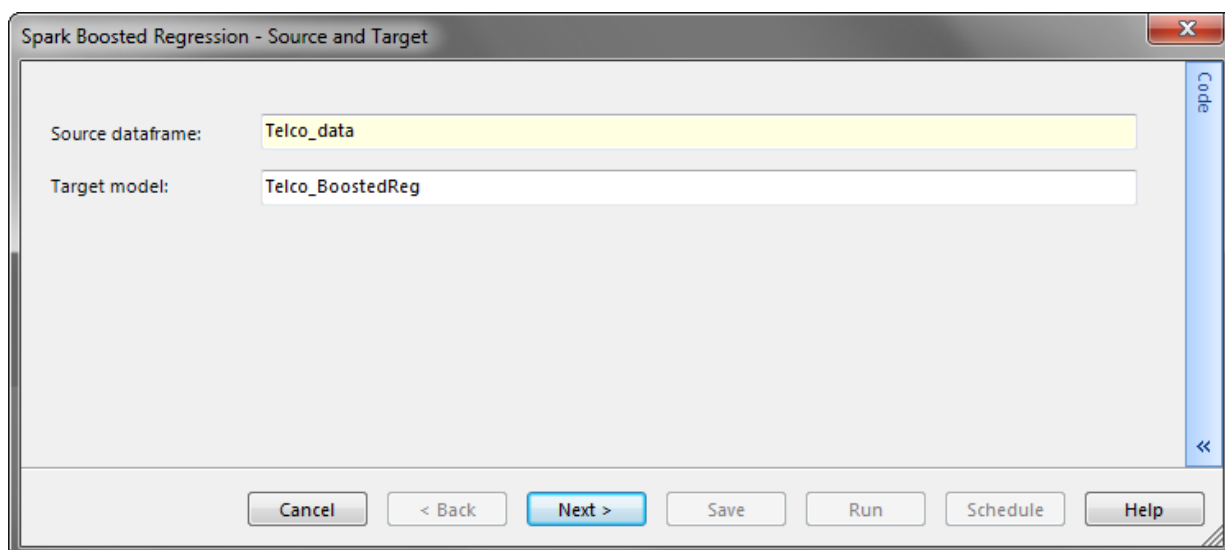
Page | 496



If the wizard does not open automatically, double-click the **Boosted Regression** node or right-click on it and select **Modify** to open the wizard.

In the first page of the Boosted Regression wizard:

- Specify the name for the model
- If necessary, choose the option to exclude constant variables. (If you select this option, note that proceeding to the next page may take some time for large and wide dataframes.)



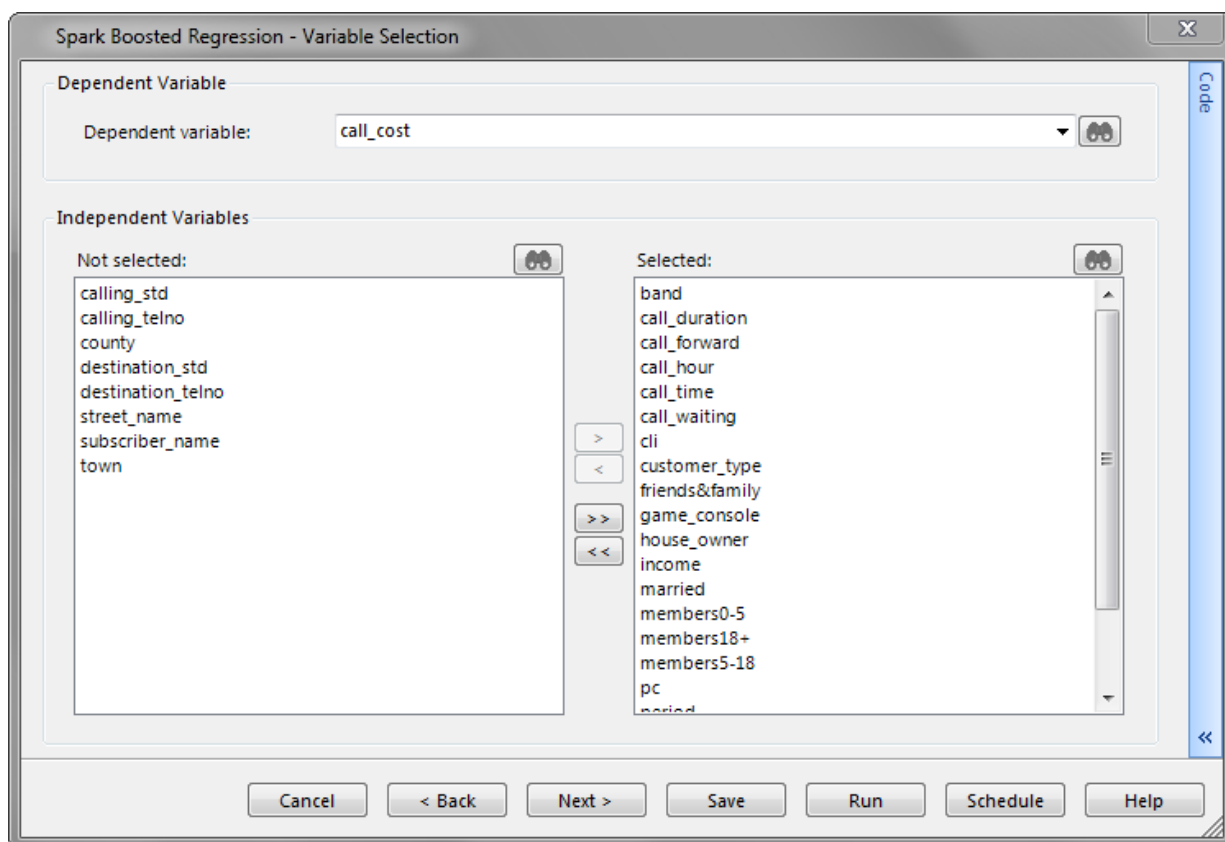
Click Next. In the next page:

- Select the dependent variable. Only numeric variables are allowed as dependent.
- Select the independent variables to include in the model.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

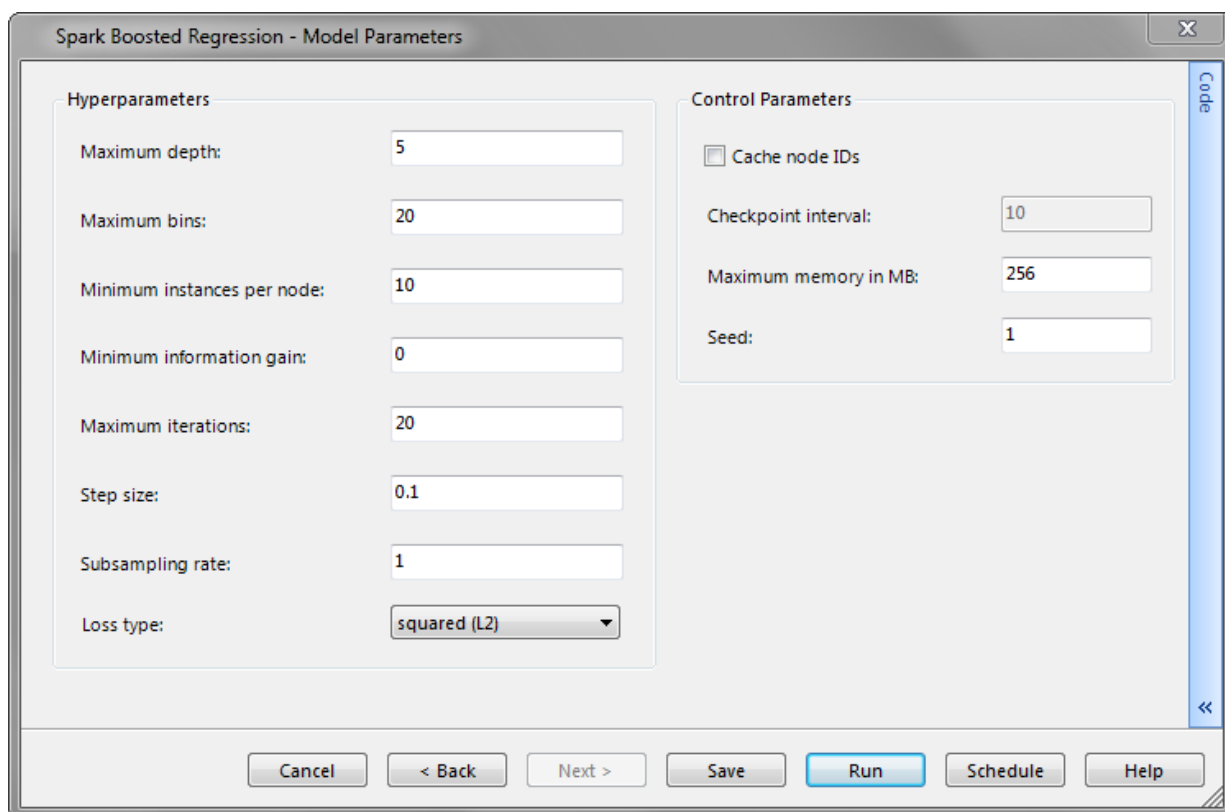


Click Next. In the next page, specify the model parameters:

- Maximum tree depth (number of levels, including the root node)
- Maximum number of bins in each split
- Minimum number of instances (records) per node
- Minimum information gain
- Maximum number of iterations
- Step size
- Subsampling rate
- Loss type: squared (L2) or absolute (L1)

Control parameters:

- Cache node IDs (turned off by default)
- Checkpoint interval (enabled if the *Cache node ID* option is selected; the default is 10)
- Maximum memory in Megabytes (the default is 256 MB)
- Random seed



The dialog box is titled "Spark Boosted Regression - Model Parameters". It is divided into two main sections: "Hyperparameters" and "Control Parameters".

**Hyperparameters:**

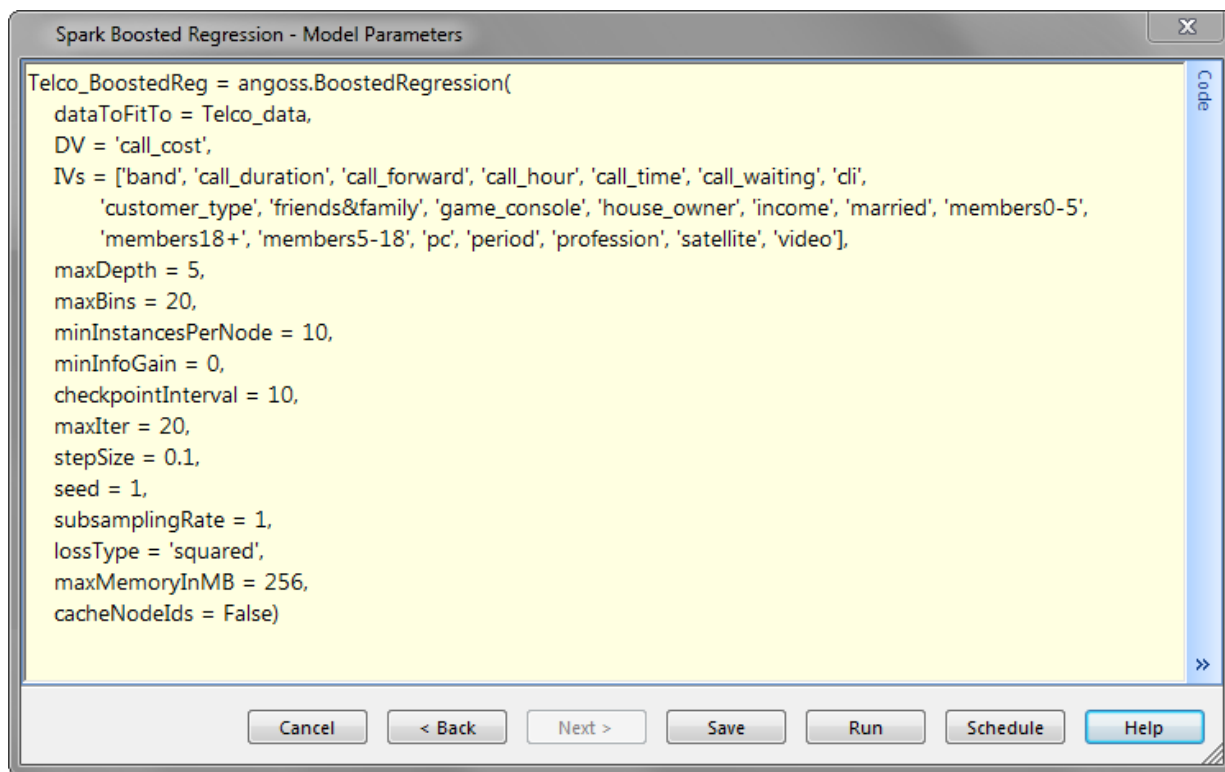
- Maximum depth: 5
- Maximum bins: 20
- Minimum instances per node: 10
- Minimum information gain: 0
- Maximum iterations: 20
- Step size: 0.1
- Subsampling rate: 1
- Loss type: squared (L2) (dropdown menu)

**Control Parameters:**

- ☐ Cache node IDs
- Checkpoint interval: 10
- Maximum memory in MB: 256
- Seed: 1

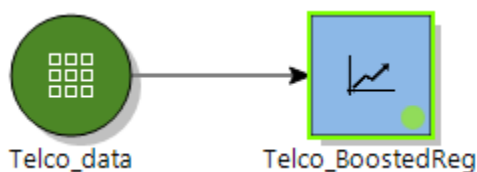
At the bottom of the dialog, there is a row of buttons: Cancel, < Back, Next >, Save, Run (highlighted in blue), Schedule, and Help. On the right side of the dialog, there is a vertical blue bar labeled "Code" at the top and a double arrow icon at the bottom.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output model is created in the workflow after running the node.



Double-click the output model node to view the results.

## Clustering

Only K-means clustering is available for Spark dataframes in this version.

To build a Clustering model on a Spark dataframe in your workflow:

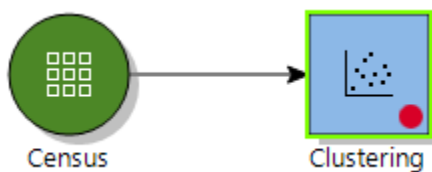
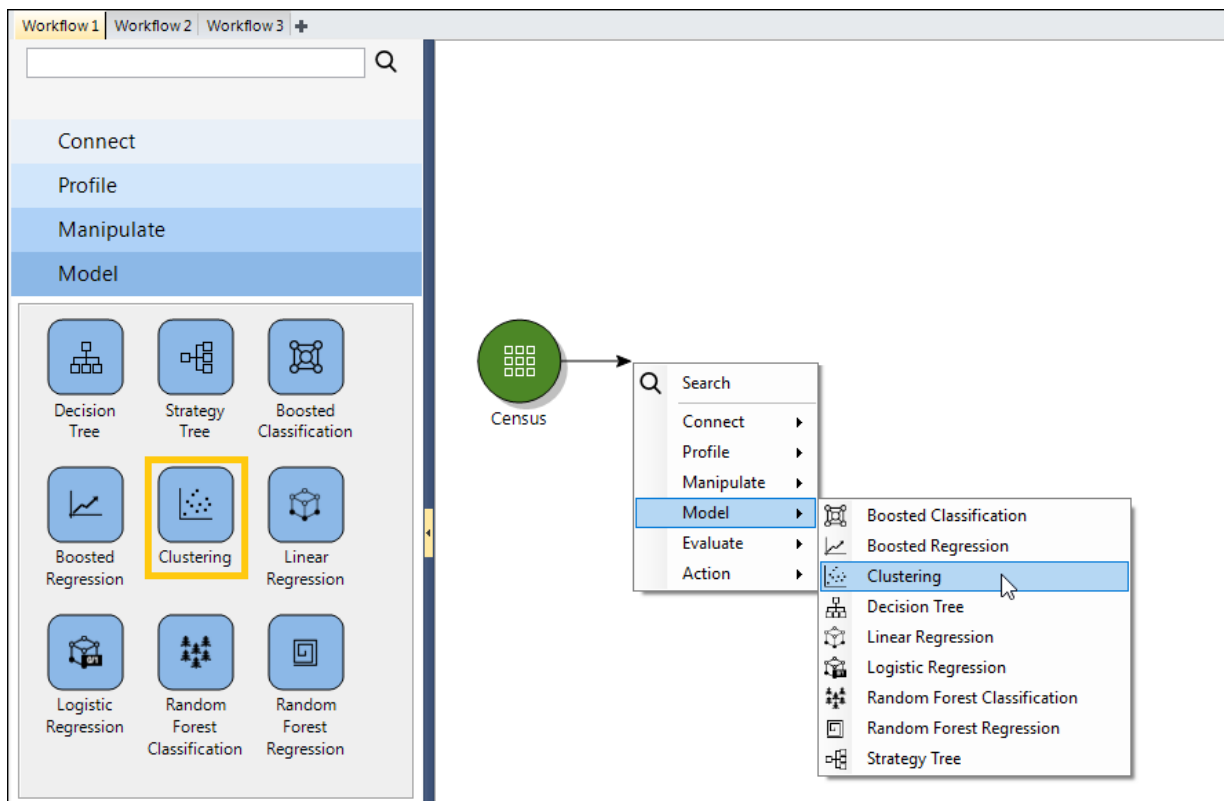


- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the context menu, select **Model** → **Clustering** as in the example screenshot below. A new **Clustering** process node will be created.

OR:

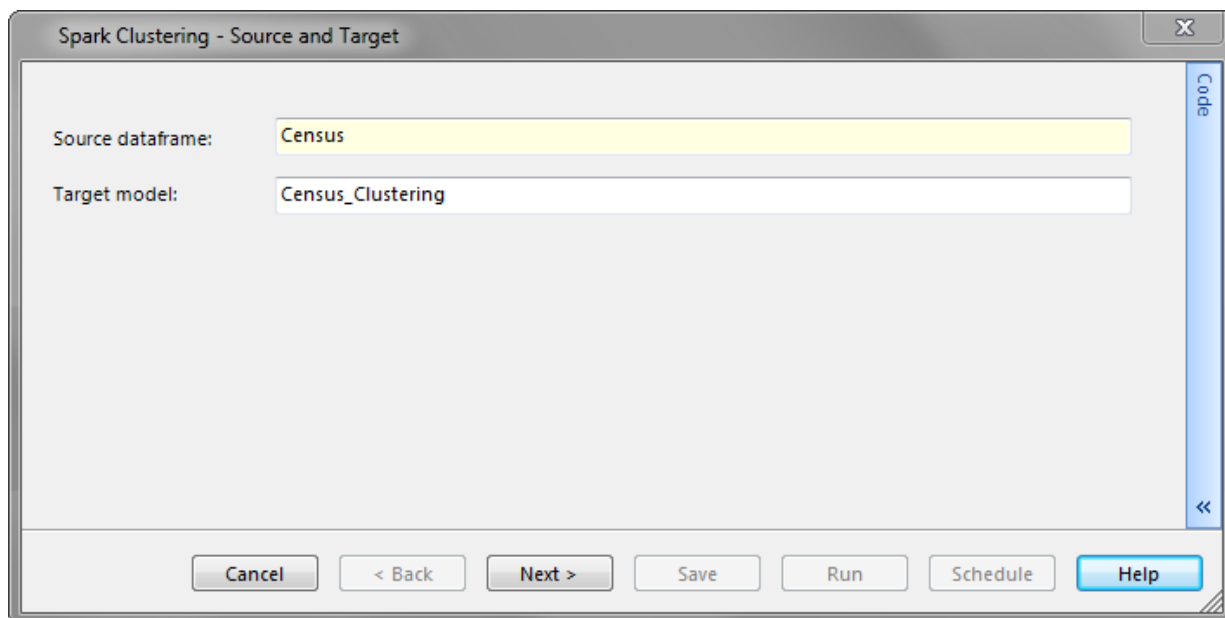
- Drag the **Clustering** node from the **Model** palette onto the canvas and connect the input dataframe to it.

Page | 501



If the wizard does not open automatically, double-click the **Clustering** node or right-click on it and select **Modify** to open the wizard.

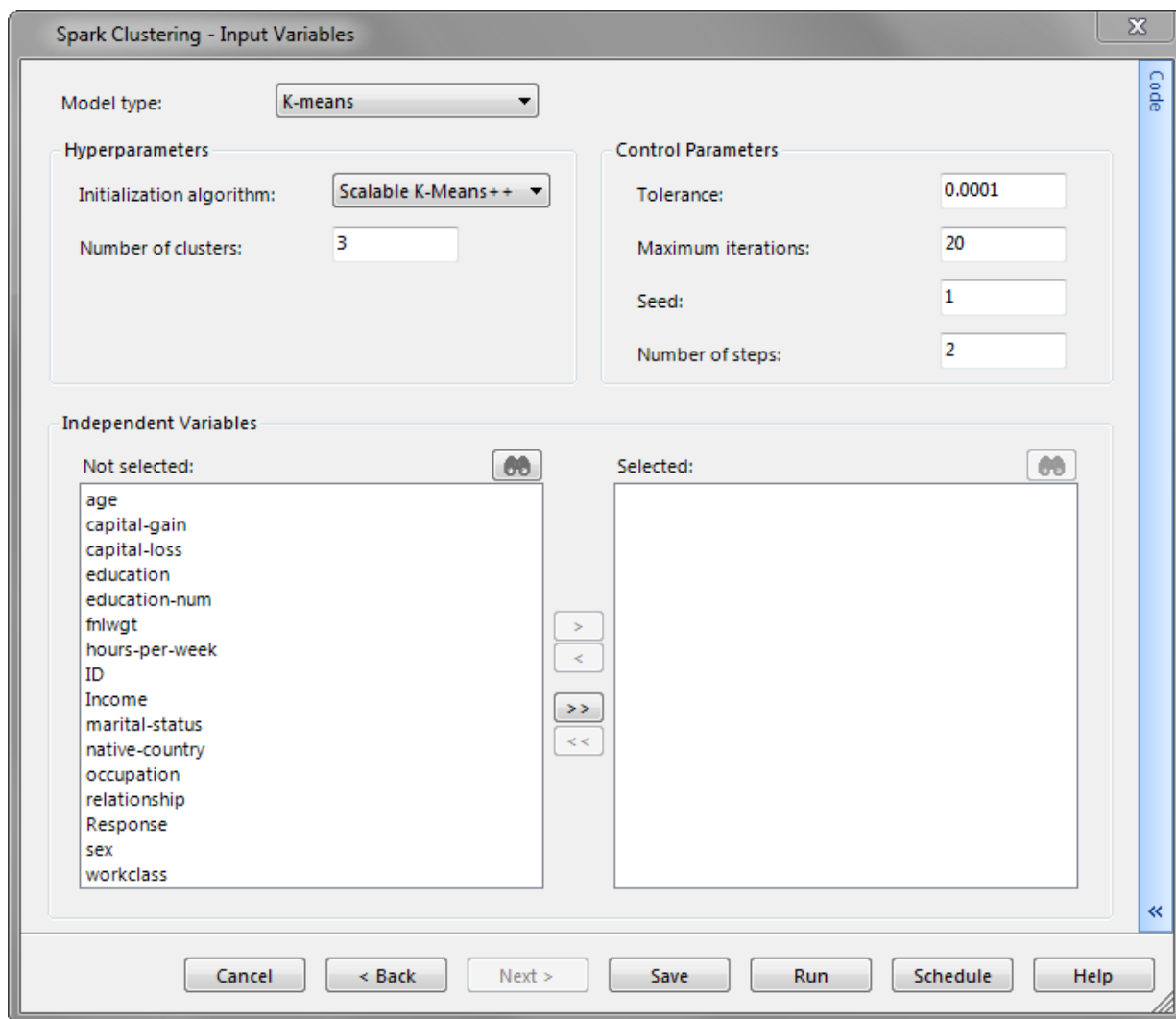
Specify the name for the model in the first page of the Clustering wizard:



The dialog box titled "Spark Clustering - Source and Target" has a close button (X) in the top right corner. It contains two input fields: "Source dataframe:" with the value "Census" and "Target model:" with the value "Census\_Clustering". A vertical "Code" sidebar is on the right. At the bottom, there are buttons for "Cancel", "< Back", "Next >", "Save", "Run", "Schedule", and "Help".

Click **Next**. In the next page:

- Select the input variables for the model.
- The model type is K-means. Other types will be supported in the future versions.
- Specify the cluster initialization algorithm (*Random* or *Scalable K-Means++*)
- Specify the required number of clusters. (The default is 3.)
- Specify the Control parameters:
  - Tolerance (the default is 0.0001)
  - Maximim number of iterations (the default is 20)
  - Random seed
  - Number of steps (the default is 2)



The dialog box is titled "Spark Clustering - Input Variables". It contains the following sections:

- Model type:** A dropdown menu set to "K-means".
- Hyperparameters:**
  - Initialization algorithm:** A dropdown menu set to "Scalable K-Means++".
  - Number of clusters:** A text input field containing the value "3".
- Control Parameters:**
  - Tolerance:** A text input field containing the value "0.0001".
  - Maximum iterations:** A text input field containing the value "20".
  - Seed:** A text input field containing the value "1".
  - Number of steps:** A text input field containing the value "2".
- Independent Variables:**
  - Not selected:** A list box containing the following variables: age, capital-gain, capital-loss, education, education-num, fnlwgt, hours-per-week, ID, Income, marital-status, native-country, occupation, relationship, Response, sex, and workclass.
  - Selected:** An empty list box.
  - Between the two list boxes are four arrow buttons: ">", "<", ">>", and "<<".

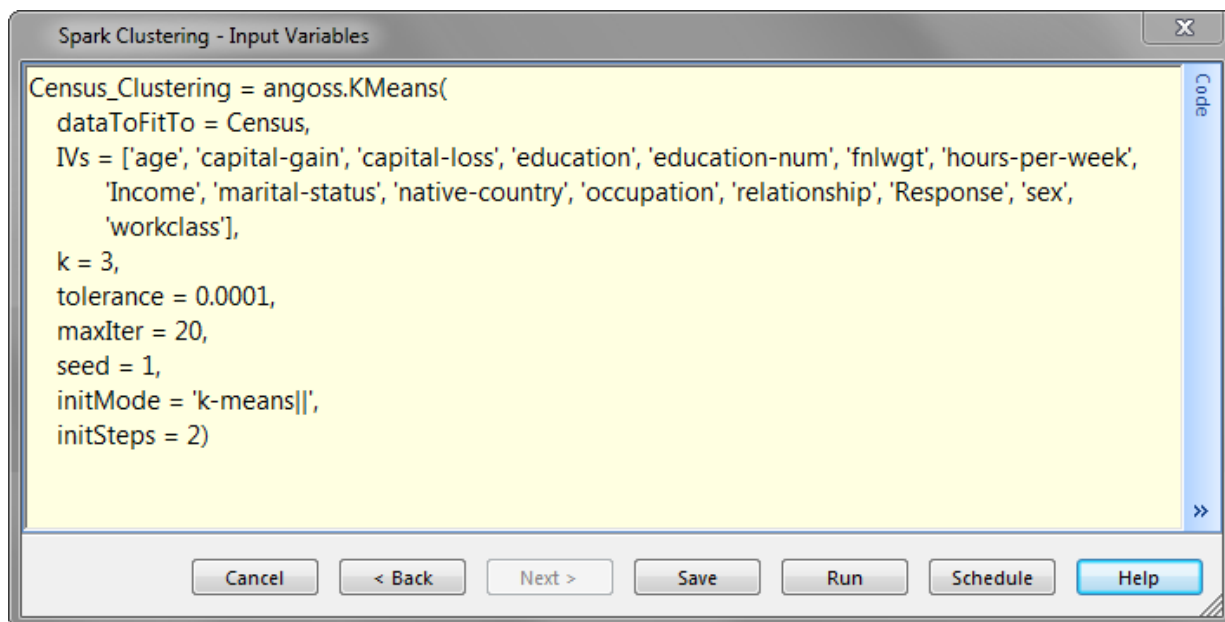
At the bottom of the dialog are several buttons: "Cancel", "< Back", "Next >", "Save", "Run", "Schedule", and "Help". On the right side, there is a vertical blue bar labeled "Code" with a double arrow button at the bottom.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

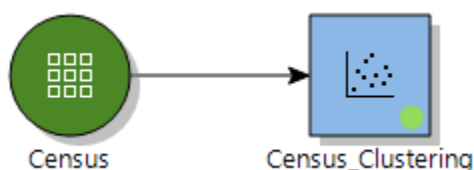
The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output model is created in the workflow after running the node.



Double-click the output model node to view the results. An example of the resulting report is given below:

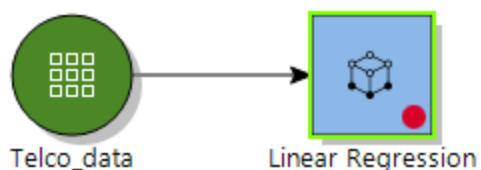
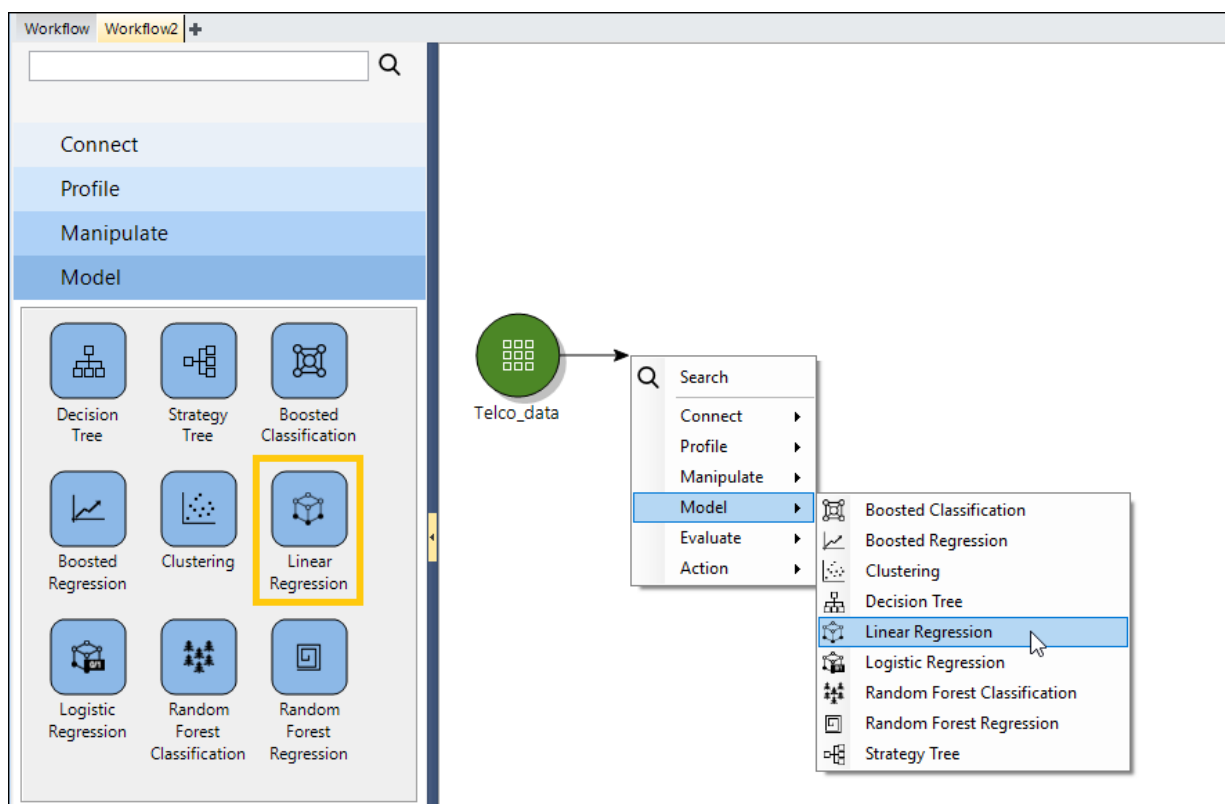
## Linear Regression

To build a Linear Regression model on a Spark dataframe in your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the context menu, select **Model --> Linear Regression** as in the example screenshot below. A new **Linear Regression** process node will be created.

OR:

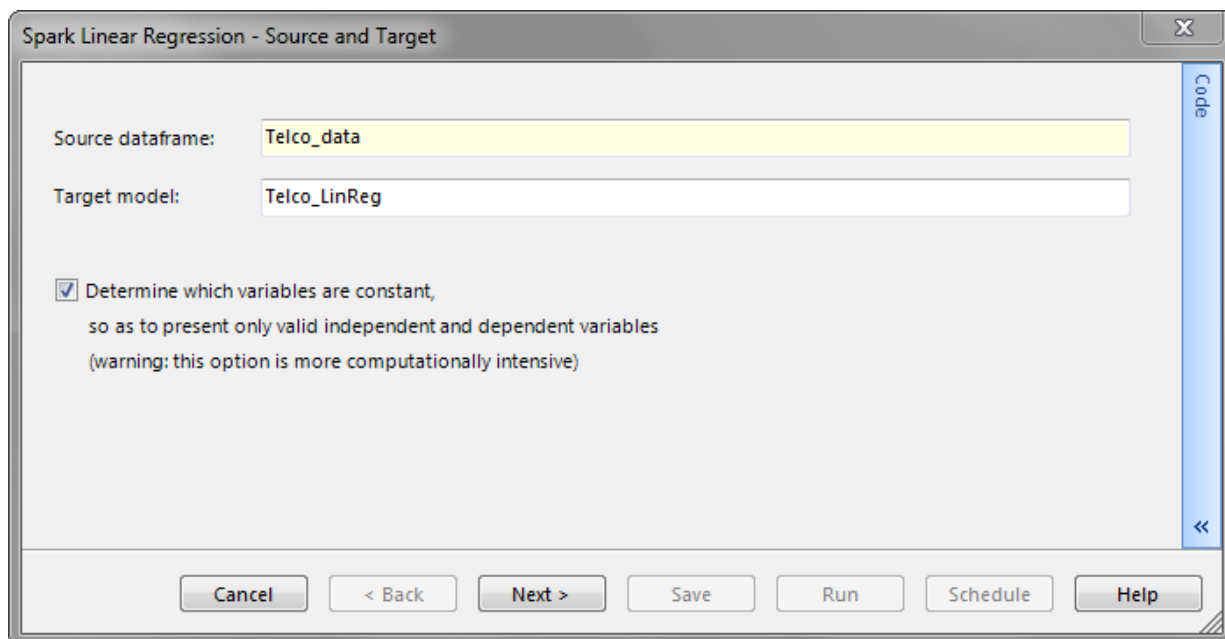
- Drag the **Linear Regression** node from the **Model** palette onto the canvas and connect the input dataframe to it.



If the wizard does not open automatically, double-click the **Linear Regression** node or right-click on it and select **Modify** to open the wizard.

In the first page of the Linear Regression wizard:

- Specify the name for the model
- If necessary, choose the option to isolate constant variables and exclude them from the choice of variables at the variable selection step. (If you select this option, note that proceeding to the next page may take some time for large and wide dataframes.)



Click Next. In the next page:

- Select the dependent variable and independent variables.
- Specify the fitting parameters:
  - Maximum number of iterations
  - Regularization parameter
  - Elastic net parameter
  - Tolerance threshold (singular value threshold)
  - Aggregation depth
  - Choose whether to fit (include) intercept

Spark Linear Regression - Variable Selection

Dependent Variable

Dependent variable:  

Fitting Parameters

Maximum iterations:  ☒ Fit Intercept

Regularization parameter:  ☒ Standardization

Elastic net parameter:

Tolerance:

Aggregation depth:

Independent Variables

Not selected: 

calling\_telno  
destination\_std  
destination\_telno  
street\_name  
subscriber\_name  
town

Selected: 

band  
call\_duration  
call\_forward  
call\_hour  
call\_waiting  
cli  
customer\_type  
friends\_family  
game\_console  
house\_owner  
income  
married

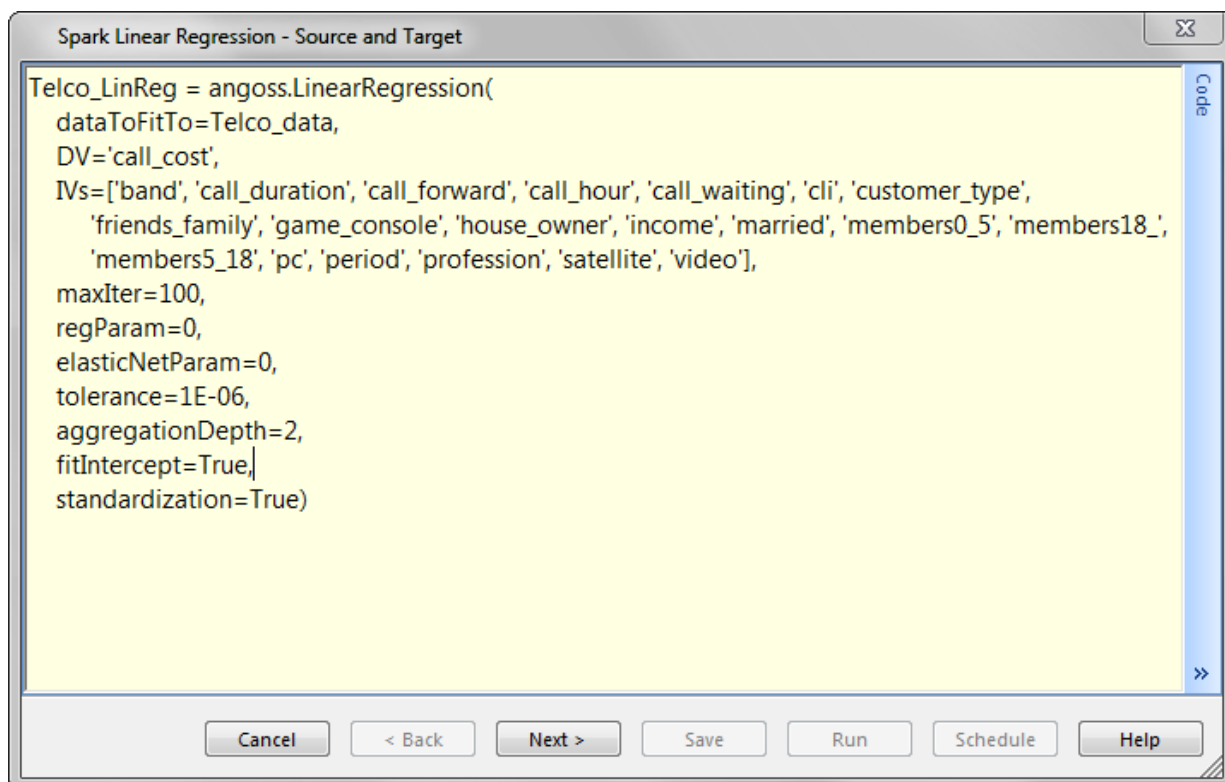
Cancel < Back Next > Save Run Schedule Help

**Tip:**

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output model is created in the workflow after running the node.



Double-click the output model node to view the results. An example of the resulting report is given below:



| Spark Linear Regression                                                                                |                                            |                |                 |                |                   |
|--------------------------------------------------------------------------------------------------------|--------------------------------------------|----------------|-----------------|----------------|-------------------|
| Model Name: Telco_LinReg<br>Dependent Variable: call_cost<br>Created: Monday, Apr 03, 2017 03:12:46 AM |                                            |                |                 |                |                   |
| Independent Variable Statistics                                                                        |                                            |                |                 |                |                   |
| Variable                                                                                               | Category                                   | Coefficient    | Standard Error  | T-Statistic    | P-Value           |
| band                                                                                                   | Local charge                               | -265.972254985 | 10.3800807829   | -25.623331894  | 0.0               |
| band                                                                                                   | Long distance international charge         | 545.999794465  | 10.9992498401   | 49.6397301998  | 0.0               |
| band                                                                                                   | Long distance premium rate national charge | 285.920932281  | 12.9346142963   | 22.1050992114  | 0.0               |
| period                                                                                                 | off peak                                   | -528.589205125 | 10.1893367539   | -51.8767038417 | 0.0               |
| call_duration                                                                                          |                                            | 1.68521705301  | 0.0145657070256 | 115.697579942  | 0.0               |
| call_forward                                                                                           |                                            | -45.2479215564 | 17.9833694149   | -2.51609809667 | 0.0118817804641   |
| call_hour                                                                                              |                                            | 4.28637855401  | 0.860674570039  | 4.98025467839  | 6.45643472907e-07 |
| call_waiting                                                                                           |                                            | 6.34722549312  | 14.3746388407   | 0.441557214998 | 0.65881921965     |
| cli                                                                                                    |                                            | -40.6557167099 | 12.4029060872   | -3.2779186123  | 0.0010493328505   |

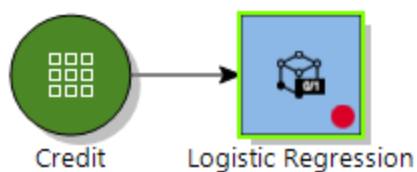
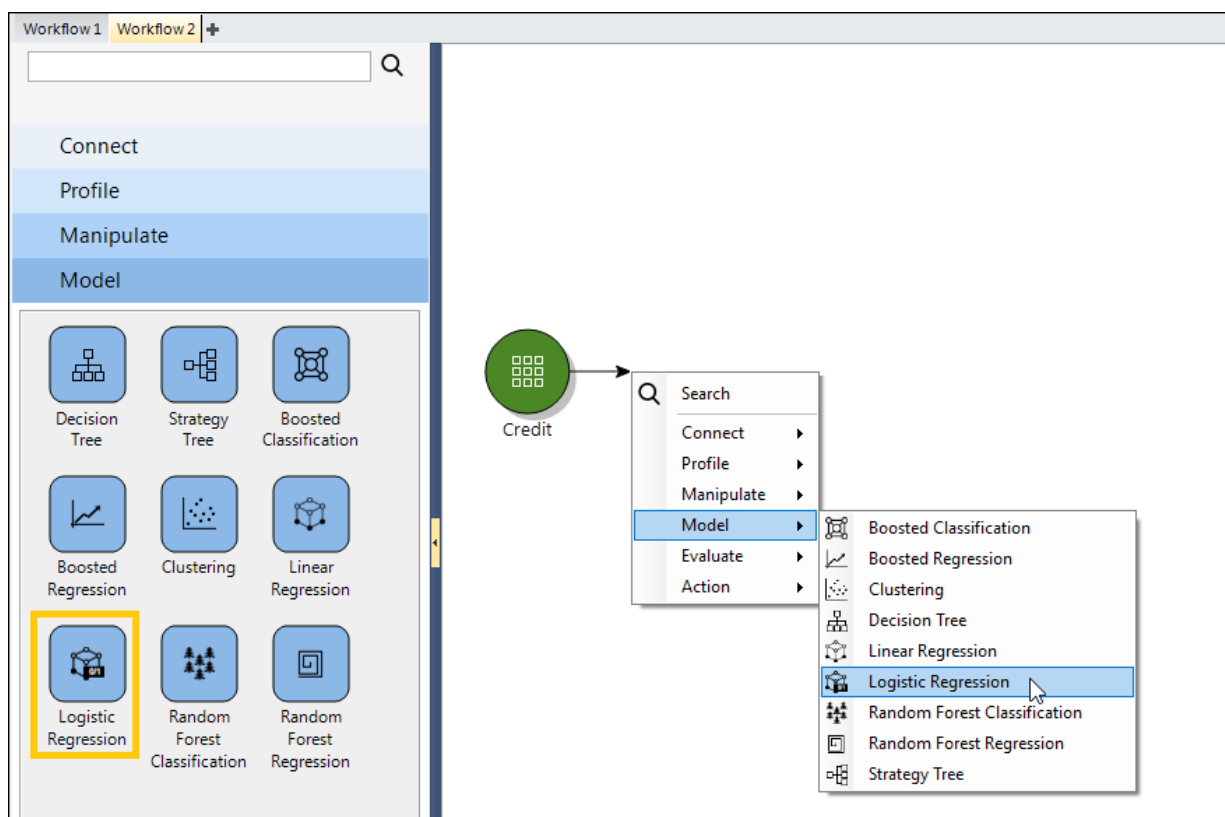
## Logistic Regression

To build a Logistic Regression model on a Spark dataframe in your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the context menu, select **Model --> Logistic Regression** as in the example screenshot below. A new **Logistic Regression** process node will be created.

OR:

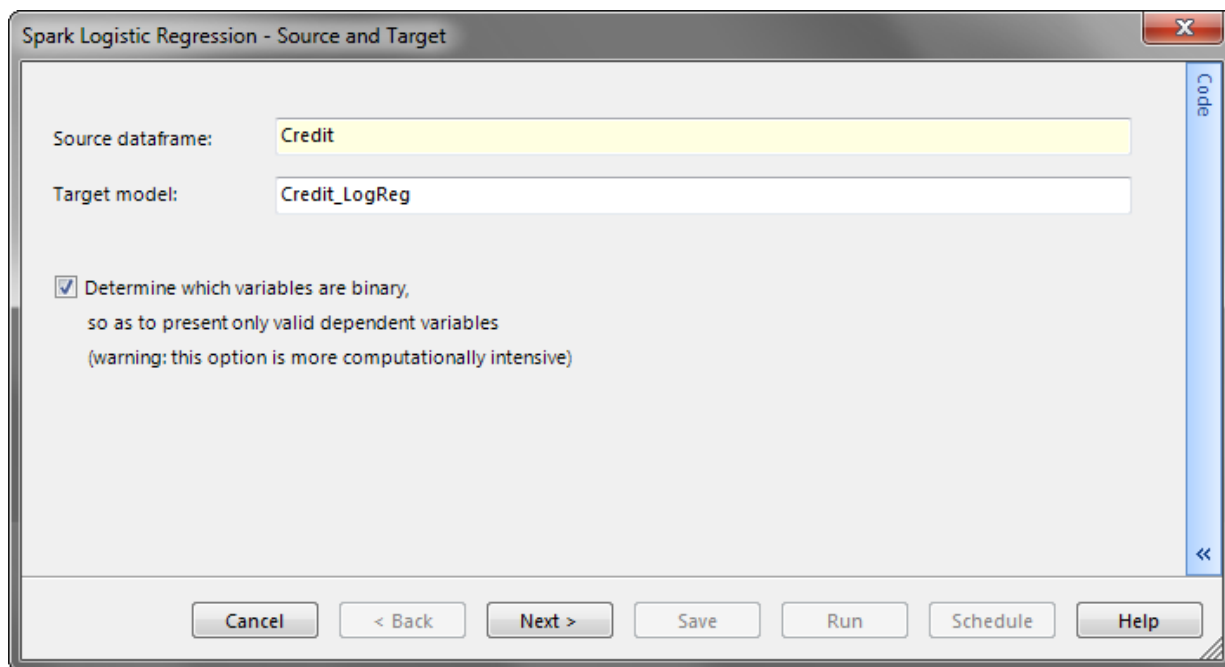
- Drag the **Logistic Regression** node from the **Model** palette onto the canvas and connect the input dataframe to it.



If the wizard does not open automatically, double-click the **Logistic Regression** node or right-click on it and select **Modify** to open the wizard.

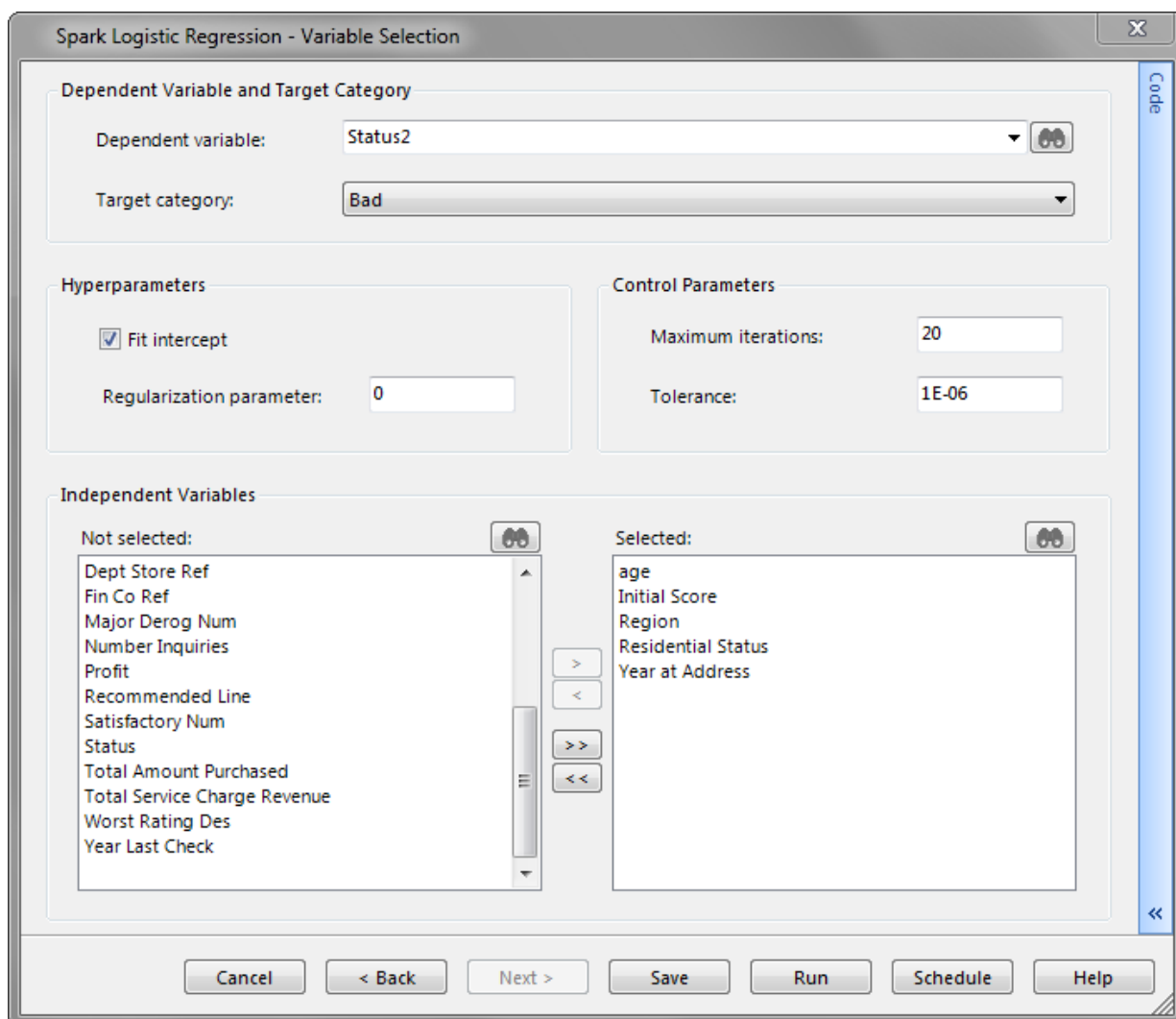
In the first page of the Logistic Regression wizard:

- Specify the name for the model
- If necessary, choose the option to exclude constant variables and identify binary variables. (If you select this option, note that proceeding to the next page may take some time for large and wide dataframes.)



Click Next. In the next page:

- Select the dependent variable and the target category. The dependent variable must be binary.
- Select the independent variables to include in the model.
- Specify the fitting and prediction parameters:
  - Regularization parameter
  - Prediction threshold (the probability threshold for the prediction of the target value to be considered correct)
  - Choose whether to fit intercept



The dialog box is titled "Spark Logistic Regression - Variable Selection". It contains several sections:

- Dependent Variable and Target Category:**
  - Dependent variable: Status2
  - Target category: Bad
- Hyperparameters:**
  - ☒ Fit intercept
  - Regularization parameter: 0
- Control Parameters:**
  - Maximum iterations: 20
  - Tolerance: 1E-06
- Independent Variables:**
  - Not selected:**
    - Dept Store Ref
    - Fin Co Ref
    - Major Derog Num
    - Number Inquiries
    - Profit
    - Recommended Line
    - Satisfactory Num
    - Status
    - Total Amount Purchased
    - Total Service Charge Revenue
    - Worst Rating Des
    - Year Last Check
  - Selected:**
    - age
    - Initial Score
    - Region
    - Residential Status
    - Year at Address

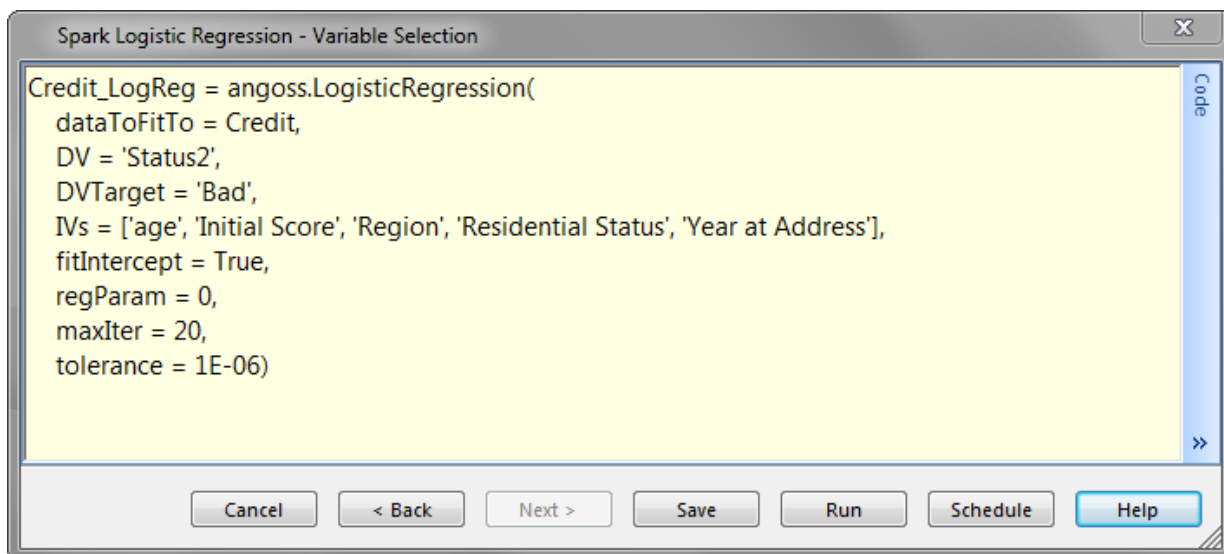
At the bottom are buttons: Cancel, < Back, Next >, Save, Run, Schedule, and Help. On the right side, there is a vertical blue bar labeled "Code" with arrows at the top and bottom.

**Tip:**

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

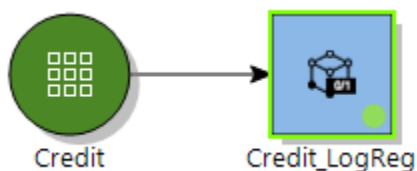
The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output model is created in the workflow after running the node.



Double-click the output model node to view the results.

## Random Forest Classification

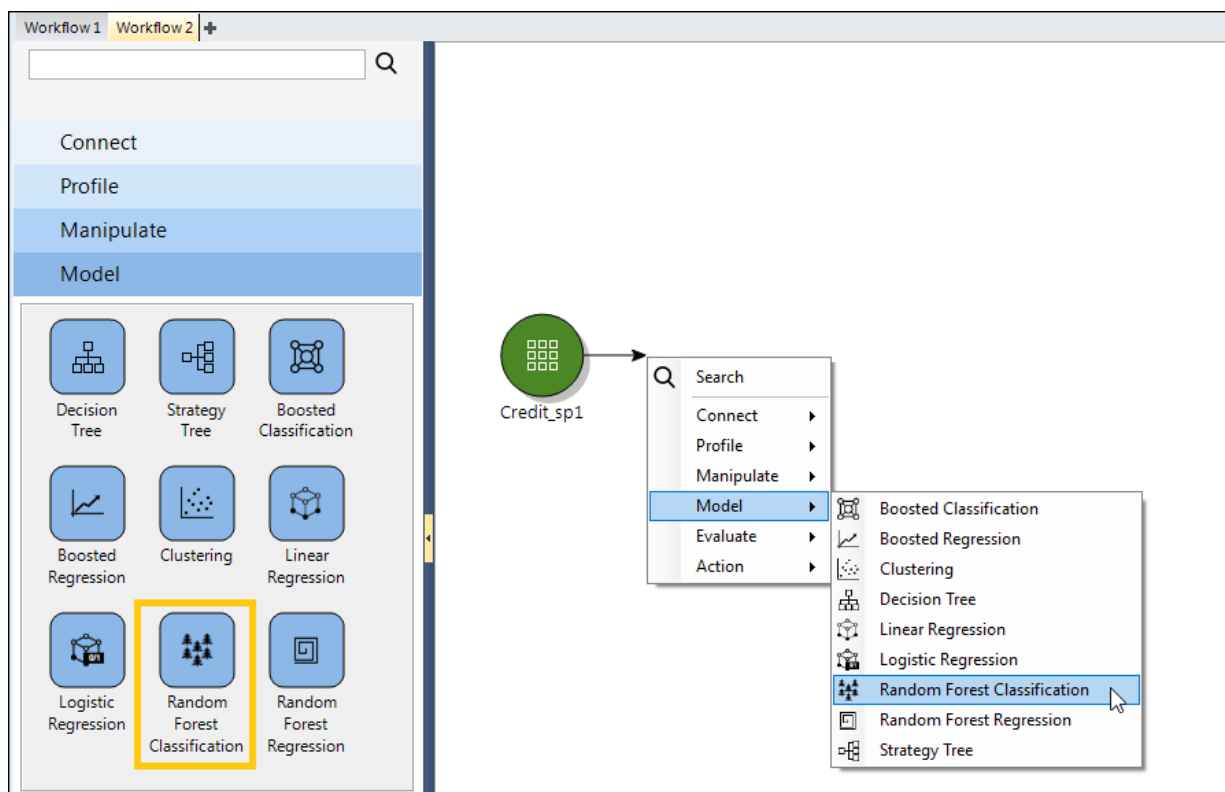
The Random Forest technique is an evolution of the Bootstrap Aggregating algorithm. The algorithm creates many different trees using changing features and different samples of the data, grows a tree in each sample, scores the probabilities and averages the results to generate the final probabilities. Knowledge Studio models on Spark include Random Forest Classification and Random Forest Regression. This topic describes the Random Forest Classification node.

To build a Random Forest Classification model on a Spark dataframe in your workflow:

- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the context menu, select **Model --> Random Forest Classification** as in the example screenshot below. A new **Random Forest Classification** process node will be created.

OR:

- Drag the **Random Forest Classification** node from the **Model** palette onto the canvas and connect the input dataframe to it.



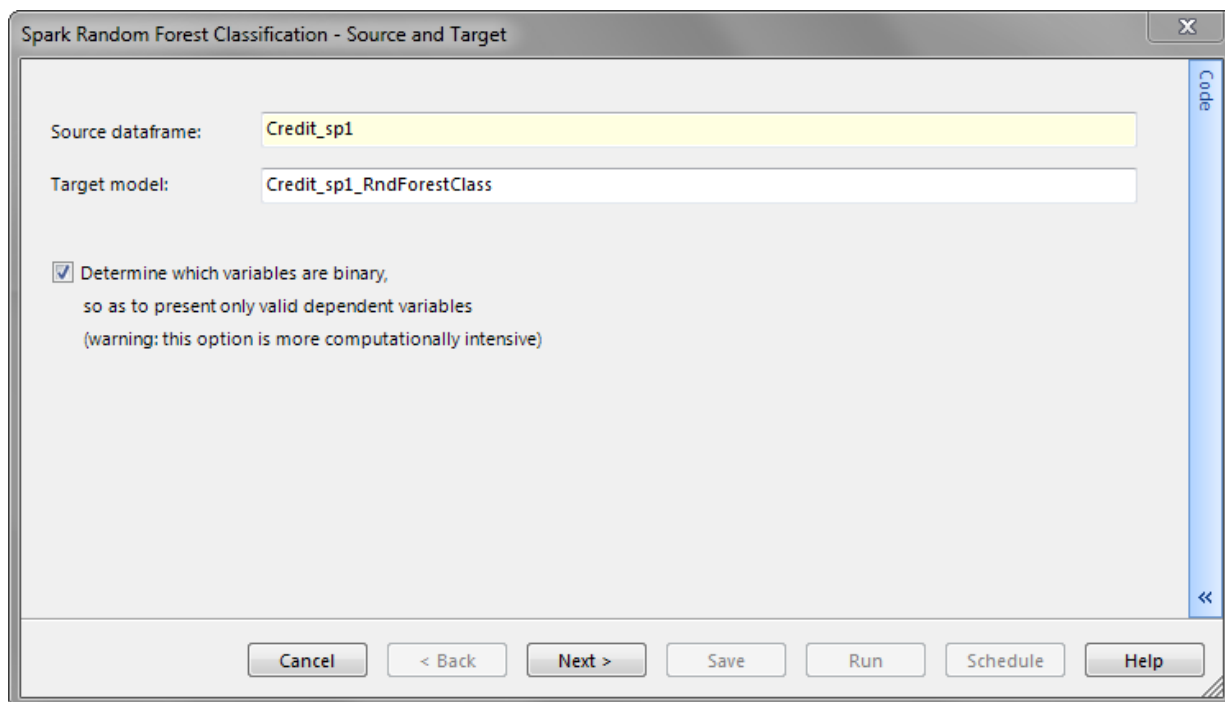
Page | 514



If the wizard does not open automatically, double-click the **Random Forest Classification** node or right-click on it and select **Modify** to open the wizard.

In the first page of the Random Forest Classification wizard:

- Specify the name for the model
- If necessary, choose the option to exclude constant variables and identify binary variables. (If you select this option, note that proceeding to the next page may take some time for large and wide dataframes.)



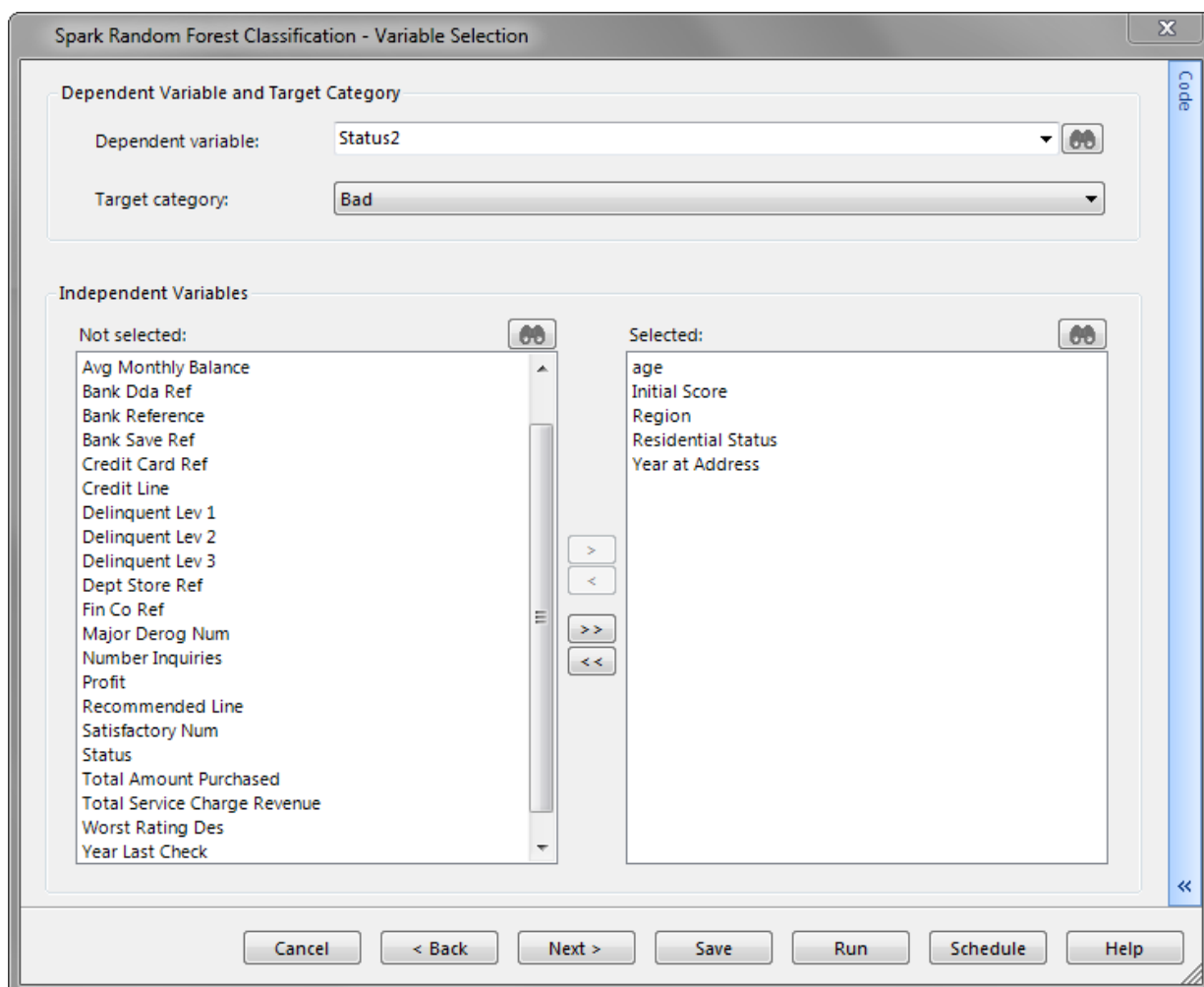
Click Next. In the next page:

- Select the dependent variable and the target category. The dependent variable must be binary.
- Select the independent variables to include in the model.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.



Click Next. In the next page, specify the model parameters:

Hyperparameters:

- Number of trees to build. (The default is 5.)
- Maximum tree depth, which is the number of tree levels including the root node. (The default is 5.)
- Maximum number of bins in each split. (The default is 5.)
- Minimum number of instances (records) per node. (The default is 50.)
- Minimum information gain. (The default is 0.)
- Subsampling rate. (The default is 1.)
- Impurity (the available options are: *entropy* and *Gini*)

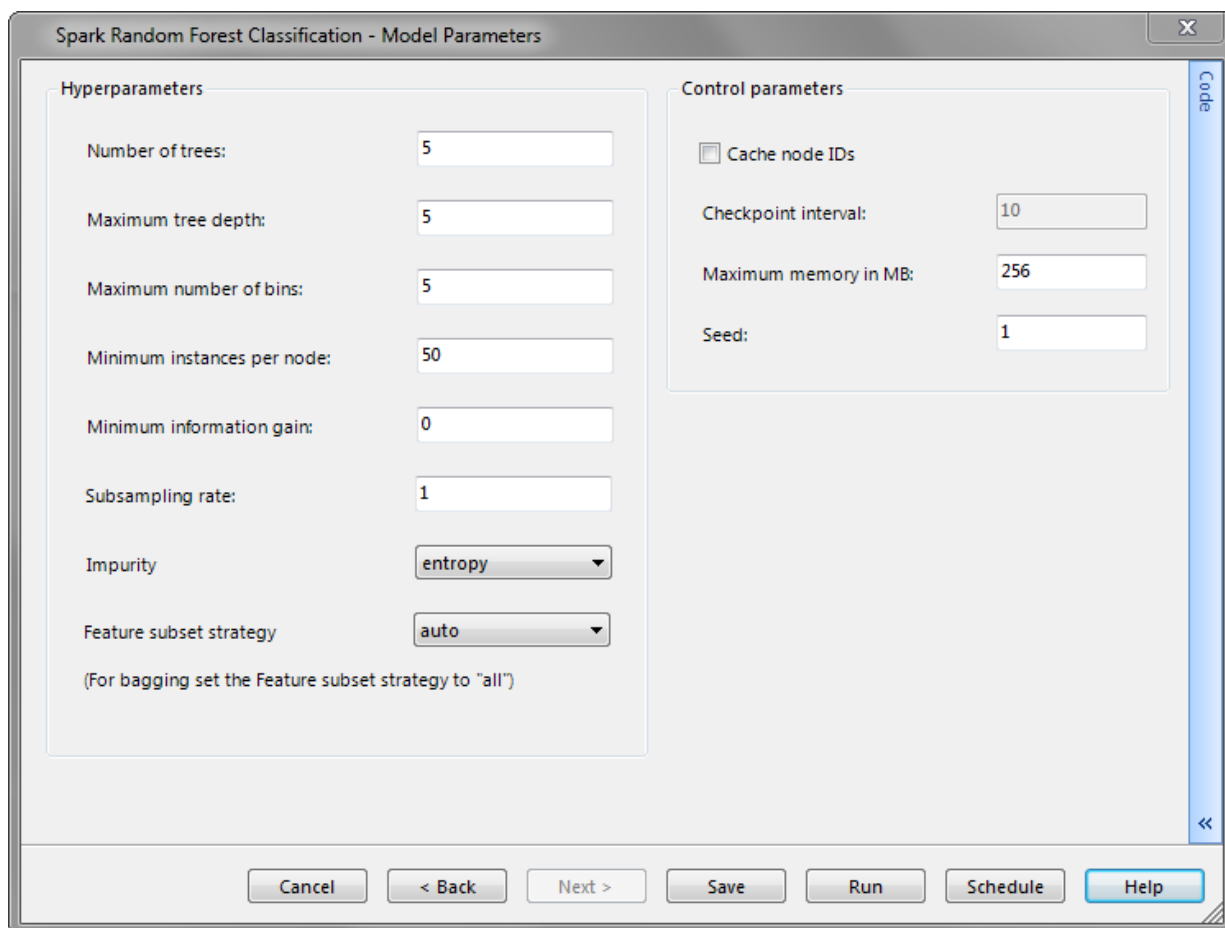


- Feature subset strategy (the available options are: *auto*, *all*, *one-third*, *square root*, *binary logarithm*, and user-defined *custom proportion*). The option "**all**" is equivalent to *Bagging* (Bootstrap Aggregating).

Control parameters:

- Cache node IDs (turned off by default)
- Checkpoint interval (enabled if the *Cache node ID* option is selected; the default is 10)
- Maximum memory in Megabytes (the default is 256 MB)
- Random seed

Page | 517



The image shows a dialog box titled "Spark Random Forest Classification - Model Parameters". It is divided into two main sections: "Hyperparameters" and "Control parameters".

**Hyperparameters:**

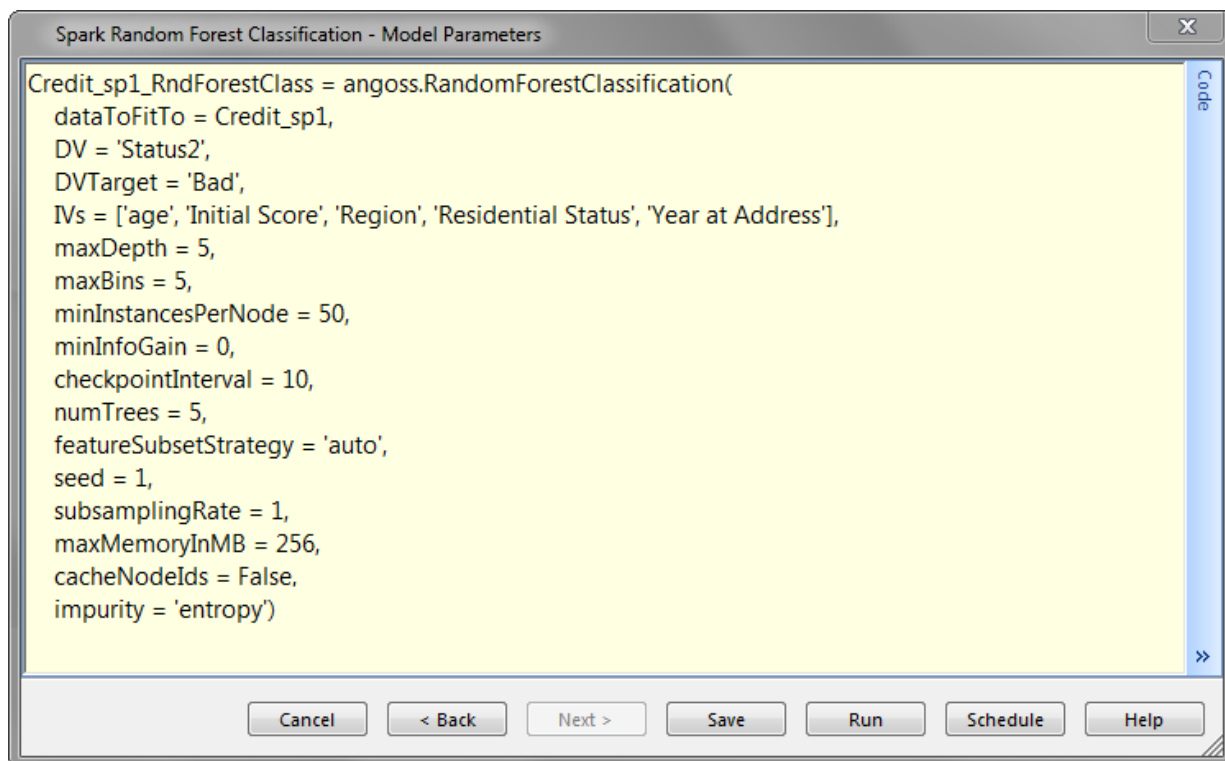
- Number of trees: 5
- Maximum tree depth: 5
- Maximum number of bins: 5
- Minimum instances per node: 50
- Minimum information gain: 0
- Subsampling rate: 1
- Impurity: entropy (dropdown menu)
- Feature subset strategy: auto (dropdown menu)
- (For bagging set the Feature subset strategy to "all")

**Control parameters:**

- ☐ Cache node IDs
- Checkpoint interval: 10
- Maximum memory in MB: 256
- Seed: 1

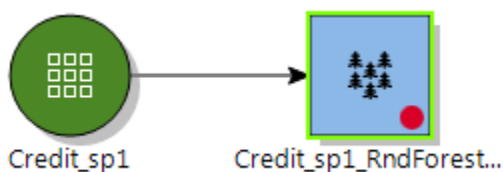
At the bottom of the dialog box, there are several buttons: Cancel, < Back, Next >, Save, Run, Schedule, and Help. On the right side, there is a blue vertical bar labeled "Code" with a double arrow icon at the bottom.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output model is created in the workflow after running the node.



Double-click the output model node to view the results.

## Random Forest Regression

To build a Random Forest Regression model on a Spark dataframe in your workflow:

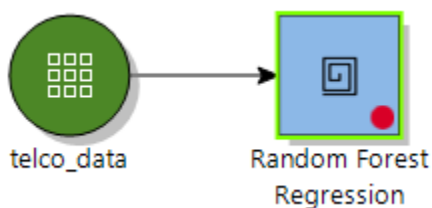
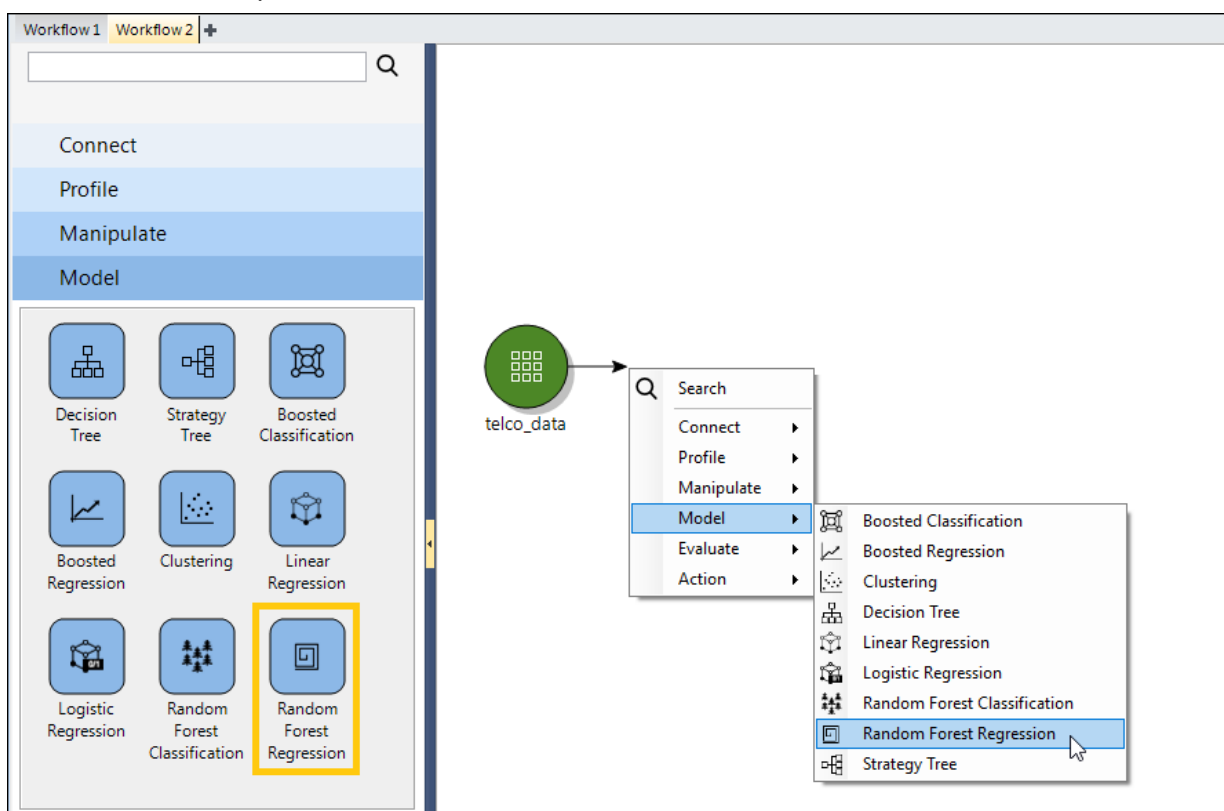
- Drag a connection arrow from a Spark dataframe node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the

context menu, select **Model** → **Random Forest Regression** as in the example screenshot below. A new **Random Forest Regression** process node will be created.

OR:

- Drag the **Random Forest Regression** node from the **Model** palette onto the canvas and connect the input dataframe to it.

Page | 519

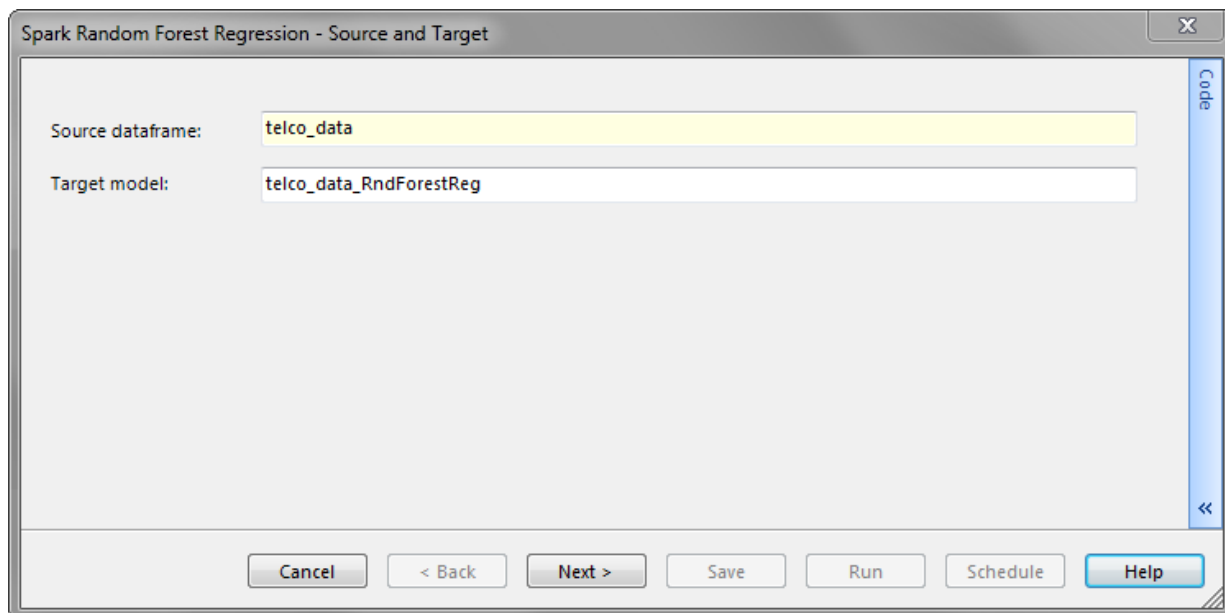


If the wizard does not open automatically, double-click the **Random Forest Regression** node or right-click on it and select **Modify** to open the wizard.

In the first page of the Random Forest Regression wizard:

- Specify the name for the model

- If necessary, choose the option to exclude constant variables. (If you select this option, note that proceeding to the next page may take some time for large and wide dataframes.)



Page | 520

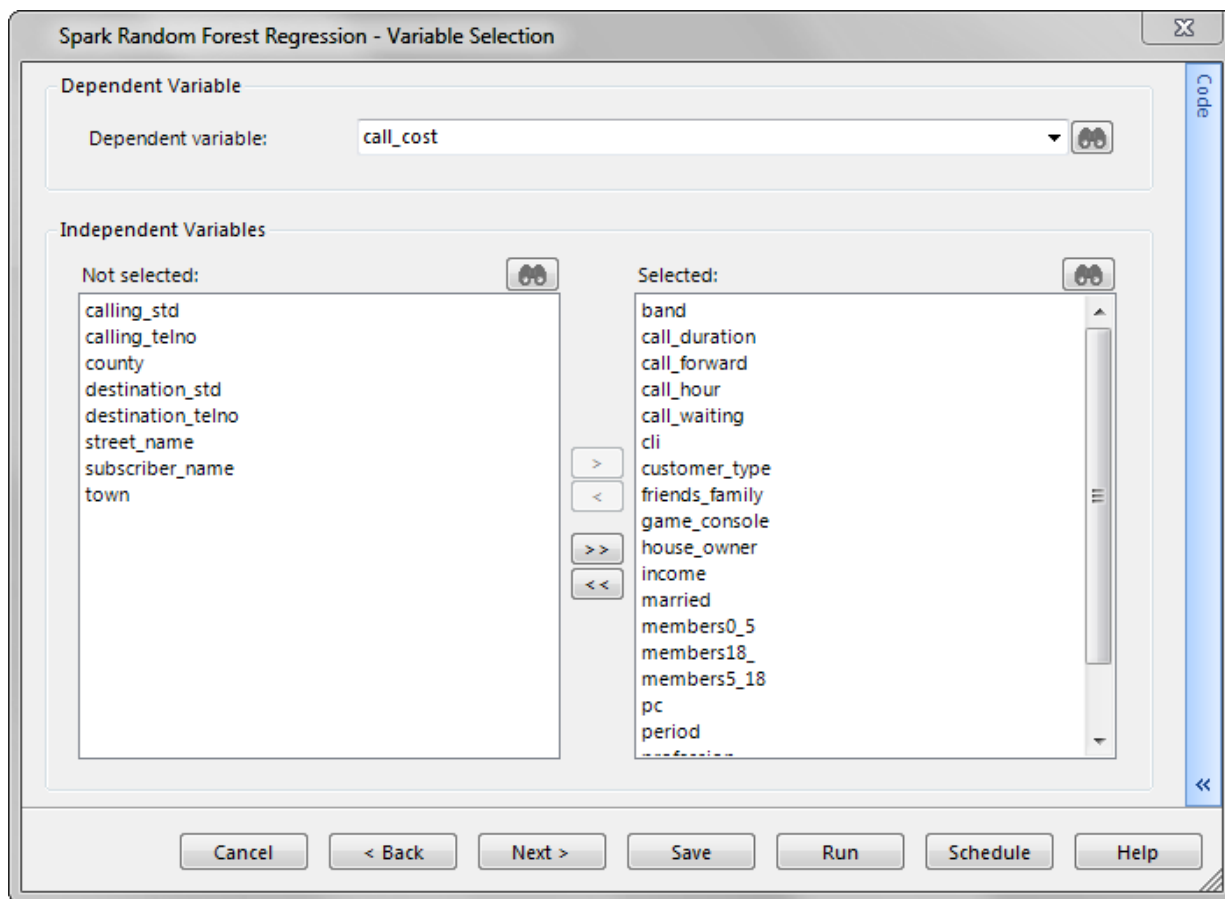
Click Next. In the next page:

- Select the dependent variable. Only numeric variables are allowed as dependent.
- Select the independent variables to include in the model.

*Tip:*

You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.



Click Next. In the next page, specify the model parameters:

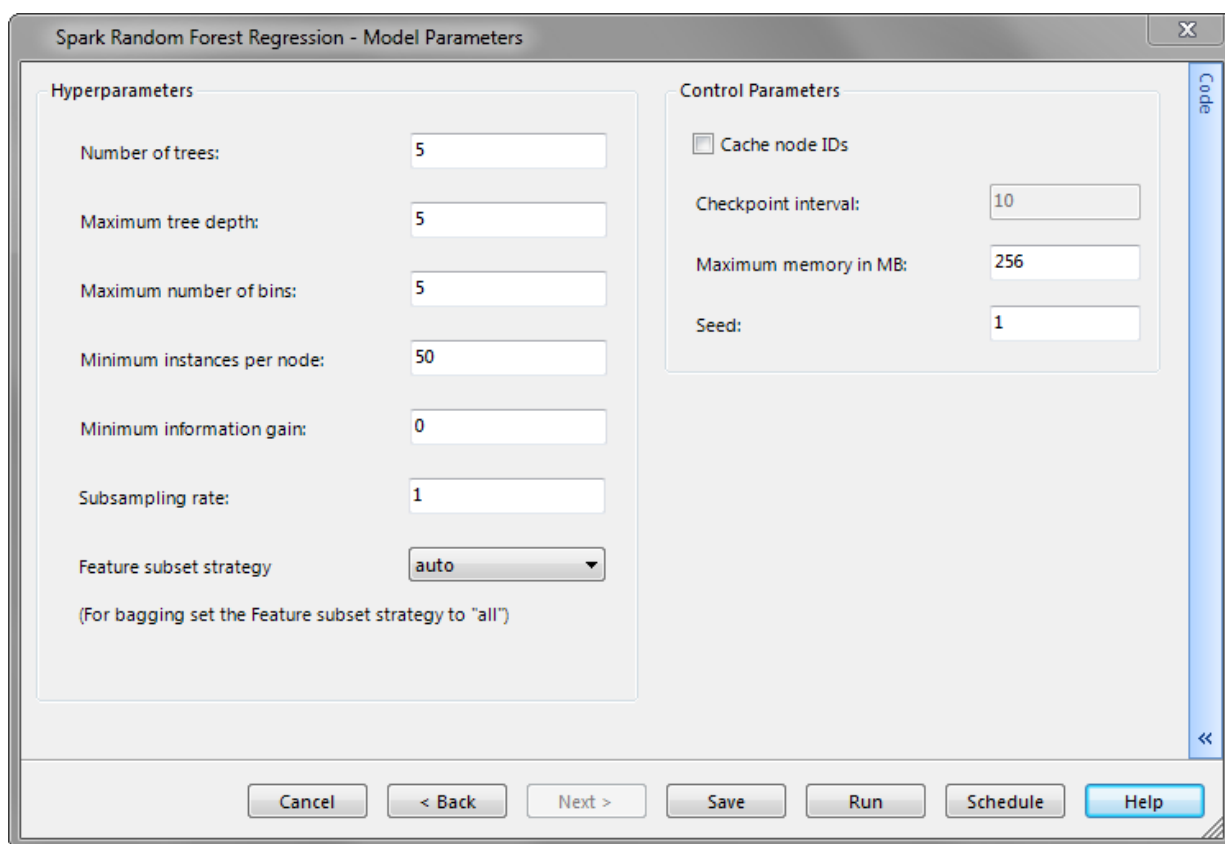
Hyperparameters:

- Number of trees to build. (The default is 5.)
- Maximum tree depth - the number of levels including the root node. (The default is 5.)
- Maximum number of bins in each split. (The default is 5.)
- Minimum number of instances (records) per node. (The default is 50.)
- Minimum information gain. (The default is 0.)
- Subsampling rate. (The default is 1.)
- Feature subset strategy (the available options are: *auto*, *all*, *one-third*, *square root*, *binary logarithm*, and user-defined *custom proportion*). The option "**all**" is equivalent to *Bagging* (Bootstrap Aggregating).

Control parameters:

- Cache node IDs (turned off by default)

- Checkpoint interval (enabled if the *Cache node ID* option is selected; the default is 10)
- Maximum memory in Megabytes (the default is 256 MB)
- Random seed



The dialog box is titled "Spark Random Forest Regression - Model Parameters". It is divided into two main sections: "Hyperparameters" and "Control Parameters".

**Hyperparameters:**

- Number of trees: 5
- Maximum tree depth: 5
- Maximum number of bins: 5
- Minimum instances per node: 50
- Minimum information gain: 0
- Subsampling rate: 1
- Feature subset strategy: auto (dropdown menu)

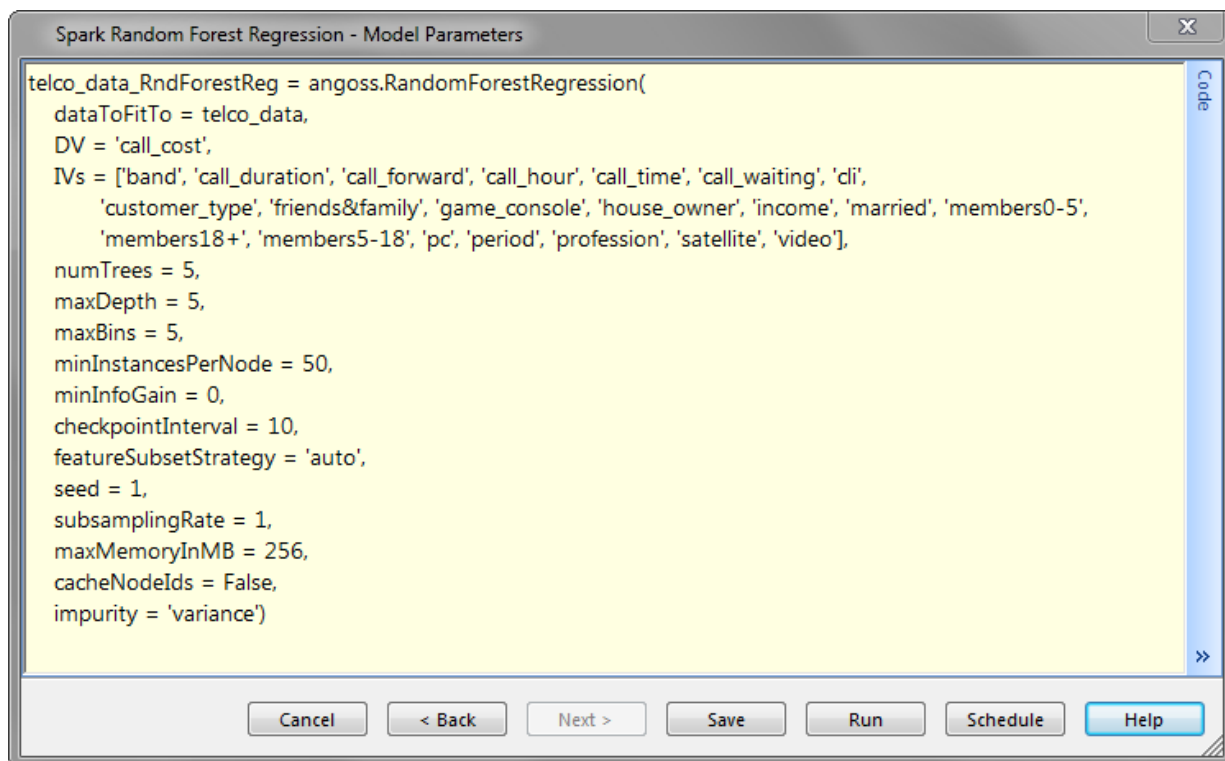
(For bagging set the Feature subset strategy to "all")

**Control Parameters:**

- ☐ Cache node IDs
- Checkpoint interval: 10
- Maximum memory in MB: 256
- Seed: 1

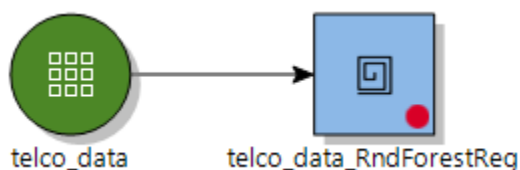
At the bottom of the dialog box, there are several buttons: Cancel, < Back, Next >, Save, Run, Schedule, and Help. On the right side of the dialog box, there is a blue vertical bar labeled "Code" with a double arrow icon at the bottom.

You can preview the corresponding code in Python at any time by clicking the arrows at the bottom of the **Code** bar - the blue vertical bar on the right side of the window. Click the arrows again to close the preview.



Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output model is created in the workflow after running the node.



Double-click the output model node to view the results.

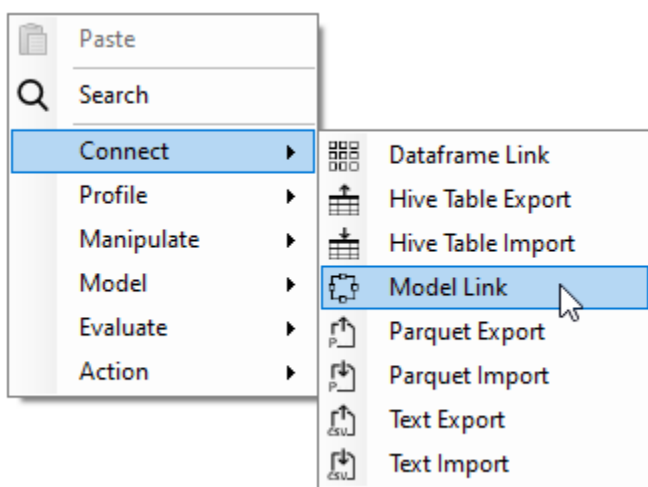
## Spark Model Link

Any Spark model available in memory during your current Spark session, whether it is present on your workflow canvas or not, can be put on the workflow canvas using the **Model Link** node in the **Connect** palette.

To link an existing Spark model to a new model node on the workflow canvas:

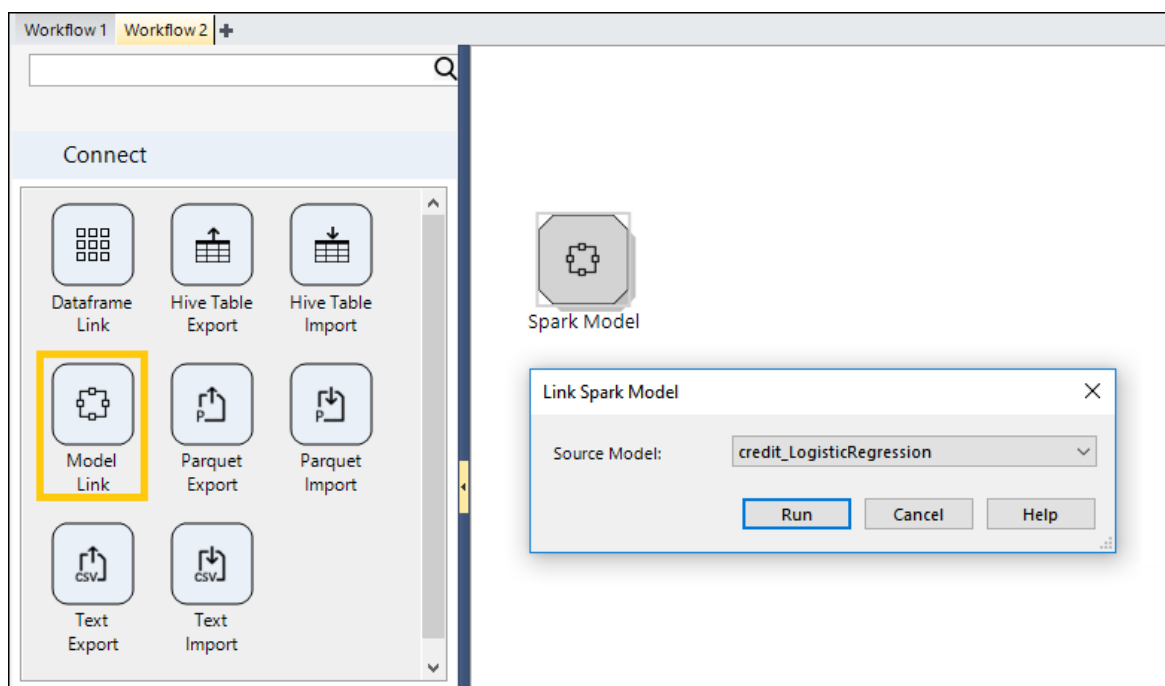
- Right-click anywhere on the Workflow canvas background. From the context menu, select **Connect --> Model Link** as shown in the example below. A new **Spark Model** node will be created and the Link Spark Model dialog will open.

Page | 524



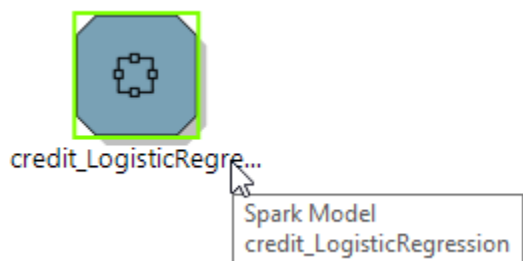
OR:

- Drag the Spark **Model Link** node from the **Connect** palette onto the canvas. Then double-click the new Spark Model node or right-click on it and select **Link** from the context menu.





Select the desired model from the Source Model list of the Link Spark Model dialog and click **Run**. The Spark Model node is now linked to the selected Spark model. Double-click on it to view the model.



The model node created this way is a model instance, which can be viewed and scored or validated. If it is a decision tree or a strategy tree, it can be modified in the interactive tree view, like any tree instance. If it is a regression model or any model other than a tree, it cannot be retrained, since it is not a process node.

Any model node placed on the canvas using the Model Link node can be re-linked by double-clicking on the node and selecting a different model.

## Model Results View

To view the model training results, double-click on the model node or right-click on it and select the **Open View** command from the context menu.

The default model results view displays a short summary report for the final model (model name, dependent variable and the target value (if applicable), and the date of model creation).

The Independent Variable Statistics table includes model coefficients and fitting statistics. For example, for Logistic Regression, it shows the coefficients, standard errors, T-Statistic, and p-value. See the example screenshot below.

**Spark Logistic Regression**

Model Name: credit\_LogisticRegression  
 Dependent Variable: Status2  
 Dependent Variable Target: Bad  
 Created: Tuesday, Dec 04, 2018 12:40:44 AM



| Independent Variable Statistics |          |             |                |             |             |
|---------------------------------|----------|-------------|----------------|-------------|-------------|
| Variable ▼                      | Category | Coefficient | Standard Error | T-Statistic | P-Value     |
| Year at Address                 |          | -0.002325   | 0.006015       | -0.386636   | 0.699026    |
| Residential Status              | Rent     | 1.159375    | 0.237499       | 4.881611    | 1.052225E-6 |
| Residential Status              | #NULL#   | 0.249585    | 0.237257       | 1.051964    | 0.292816    |
| Residential Status              | Own Home | 0.244803    | 0.245359       | 0.997732    | 0.318409    |
| Residential Status              | Other    | -0.174233   | 0.265204       | -0.656976   | 0.511196    |
| Region                          | Four     | 0.896675    | 0.747537       | 1.199506    | 0.230331    |
| Region                          | Two      | -0.723255   | 0.74552        | -0.970135   | 0.331979    |
| Region                          | Eleven   | -0.625187   | 0.746176       | -0.837855   | 0.402112    |
| Region                          | Nine     | -0.54743    | 0.744207       | -0.735588   | 0.461981    |
| Region                          | Three    | 0.628176    | 0.884283       | 0.710379    | 0.477469    |
| Region                          | One      | -0.314551   | 0.747208       | -0.420969   | 0.673778    |
| Region                          | Five     | -0.374231   | 1.269267       | -0.29484    | 0.768116    |
| Region                          | Seven    | 0.052968    | 0.766323       | 0.069119    | 0.944895    |
| Region                          | Twelve   | -0.041598   | 0.791741       | -0.05254    | 0.958098    |
| Initial Score                   |          | -0.001365   | 0.000302       | -4.521679   | 6.135091E-6 |
| age                             |          | -0.0274     | 0.002515       | -10.893549  | 0           |
| (Intercept)                     |          | -1.59332    | 0.786076       | -2.026928   | 0.04267     |

Created: Tuesday, Dec 04, 2018 12:40:44 AM

# Model Evaluation and Deployment

## Model Validation

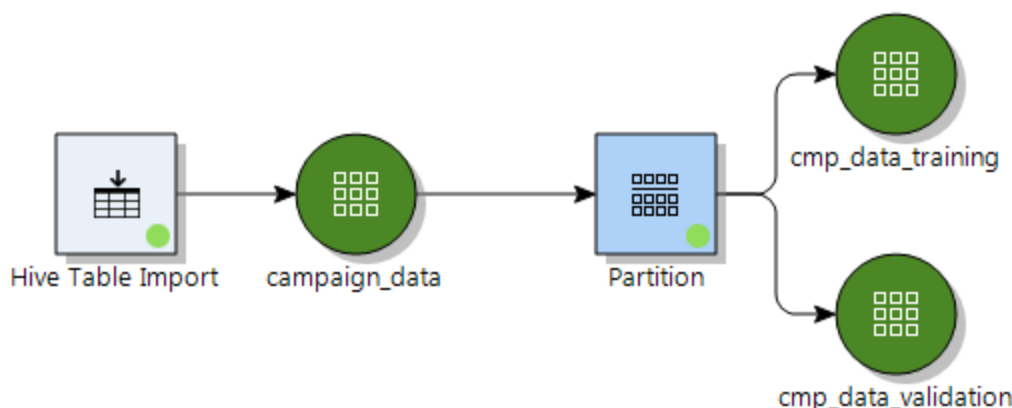
Page | 527

After creating a predictive model, it is reasonable to test the accuracy of its predictions before deploying it into production. This can be accomplished by applying the model to other datasets (different from the one the model was trained on) with known values of the dependent variable. Model predictions are then compared to these known values, and certain statistics are calculated to measure the model accuracy.

This process is called **validation**.

As a standard modeling practice, prior to creating models, the dataset is split into two partitions: **Training** and **Validation**. The model is trained on the **Training** partition and then validated on the **Validation** partition.

In the example project below, the Census dataset is split into Training and Validation partitions.



To learn how to partition a dataset, see the topic [About Partitioning](#).

## Validation Fields

The validation process creates a new dataset that contains model predictions and comparison fields. Creating some of these fields is optional. During the validation wizard setup, you choose which fields to include.

The output of the validation process depends on whether the input models have discrete or continuous dependent variable (DV).

### Validation Fields for Models with Discrete DV

In Figure 2, the *Income* field is a discrete DV with two possible values: **<=50K** and **>50K**. This field (also referred to as the *known outcome*) already exists in the dataset. Notice that the generated field names are prefixed with the DV name, *Income*.

Figure 2 - Fields Created for a Discrete DV

| -c | Income | Income Prediction | Income <=50 | Income >50K Prob  | Income Predict Prob | Income Correct                      |
|----|--------|-------------------|-------------|-------------------|---------------------|-------------------------------------|
|    | <=50K  | <=50K             | 1           | 0                 | 1                   | <input checked="" type="checkbox"/> |
|    | >50K   | >50K              | 0           | 1                 | 1                   | <input checked="" type="checkbox"/> |
|    | >50K   | >50K              | 0.090909090 | 0.909090909090909 | 0.909090909090909   | <input checked="" type="checkbox"/> |
|    | <=50K  | <=50K             | 1           | 0                 | 1                   | <input checked="" type="checkbox"/> |
|    | >50K   | <=50K             | 0.701492537 | 0.29850746268656  | 0.701492537313433   | <input type="checkbox"/>            |
|    | <=50K  | <=50K             | 0.713414634 | 0.28658536585365  | 0.713414634146341   | <input checked="" type="checkbox"/> |
|    | >50K   | >50K              | 0.271186440 | 0.72881355932203  | 0.728813559322034   | <input checked="" type="checkbox"/> |

Page | 528

In this example, the generated fields are:

|                                   |                                                                                                                                                                                                                                                                                                      |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Income Prediction</b>          | The predicted value for a given record.                                                                                                                                                                                                                                                              |
| <b>Income &lt;=50K Prob</b>       | The probability of the specified outcome (value). For example, the field <i>income &gt;50K Prob</i> contains the probability of the value ">50K". Usually, the probability of the value of interest is the only field necessary for the analysis. The probabilities of other DV values are optional. |
| <b>Income &gt;50K Prob</b>        | The probability of the predicted value (i.e., of the value in the <i>Income Prediction</i> field in this case).                                                                                                                                                                                      |
| <b>Income Predict Probability</b> | Indicates whether the predicted value coincides with the actual DV value. Provides a quick way to determine if the prediction is correct for a given record.                                                                                                                                         |

Note that the *Income Correct* column in Figure 2 shows one record that is not checked. This means the prediction does not match the actual *Income* value. You can verify this directly by comparing the values in the corresponding columns (*Income* and *Income Prediction*).

### Validation Fields for Models with Continuous DV

Because the "correctness" of a prediction on a continuous DV is not as discernible as it is for a discrete DV, a hit range is specified during the validation wizard. Predicted values within that range are assessed as correct.

In Figure 3, the *age* field is a continuous DV. As with the discrete DV example above, this field already exists in the dataset. Similarly, the generated field names are prefixed with the DV name, in this case, *age*.

Figure 3 - Fields Created for a Continuous DV

| age | age Prediction | age Correct                         | age Std Dev | age Error    | age Standard | ageAbsolute | age Squared |
|-----|----------------|-------------------------------------|-------------|--------------|--------------|-------------|-------------|
| 30  | 34.6918489066  | <input type="checkbox"/>            | 13.04705219 | -4.691848906 | 0            | 4.691848906 | 22.01344616 |
| 26  | 38.1526586621  | <input type="checkbox"/>            | 13.03777367 | -12.15265866 | -1           | 12.15265866 | 147.6871125 |
| 53  | 48.3490566038  | <input type="checkbox"/>            | 18.63485850 | 4.650943396  | 0            | 4.650943396 | 21.63127447 |
| 22  | 40.9773584906  | <input type="checkbox"/>            | 11.72860142 | -18.97735849 | -2           | 18.97735849 | 360.1401352 |
| 38  | 38.1526586621  | <input checked="" type="checkbox"/> | 13.03777367 | -0.152658662 | 0            | 0.152658662 | 0.023304667 |
| 18  | 40.9484873262  | <input type="checkbox"/>            | 14.47908737 | -22.94848732 | -2           | 22.94848732 | 526.6330705 |

In this example, the generated fields are:

|                               |                                                                                                                                                                                                                                                                                         |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>age Prediction</b>         | The predicted value for a given record.                                                                                                                                                                                                                                                 |
| <b>age Correct</b>            | Indicates whether the predicted value is close enough to the actual DV value to be considered a correct prediction. This depends on the user-defined <b>valid hit range</b> specified in the validation wizard. See the topic <a href="#">Validation on Spark</a> for more information. |
| <b>age Std Dev</b>            | The standard deviation of the DV based on the distribution in the entire dataset.                                                                                                                                                                                                       |
| <b>age Error</b>              | The difference between the actual value and the predicted value.                                                                                                                                                                                                                        |
| <b>age Standardized Error</b> | The standard deviation of the error.                                                                                                                                                                                                                                                    |
| <b>age Absolute Error</b>     | The absolute value of the error.                                                                                                                                                                                                                                                        |
| <b>age Squared Error</b>      | The squared value of the error.                                                                                                                                                                                                                                                         |
| <b>age Node ID</b>            | For Decision Tree models only, this is the node ID of the tree node to which the record is assigned.                                                                                                                                                                                    |

## Further Analysis

Once the validation is complete, there are a number of ways to analyze the results. Charts created in the model analyzer, such as cumulative lift, K-S and ROC charts, will help you determine how well the model performed during validation.

## Special Notes on Validating Decision Trees

Validating a decision tree is a little different from validating other models due to the interactive character of trees and the fact that the validation fields are calculated columns depending on the validated models.

If you validate a tree model, the calculated columns created with the validation operation refer to a read-only copy of the tree being validated, not to the original tree. This read-only version

of the original tree is created automatically by the validation process and is called a “snapshot tree”. This prevents a variety of problems that can occur if a referenced model is changed after validation (especially if validation fields were appended to the parent dataset) and allows the user to delete and modify trees without breaking any dependencies. In addition, the user always has a copy of a tree as it was at the time of validation.

Proceed to [Validation on Spark](#).

For general information about model validation, see the topic [About Validation](#).

This topic applies to models and dataframes on Apache Spark.

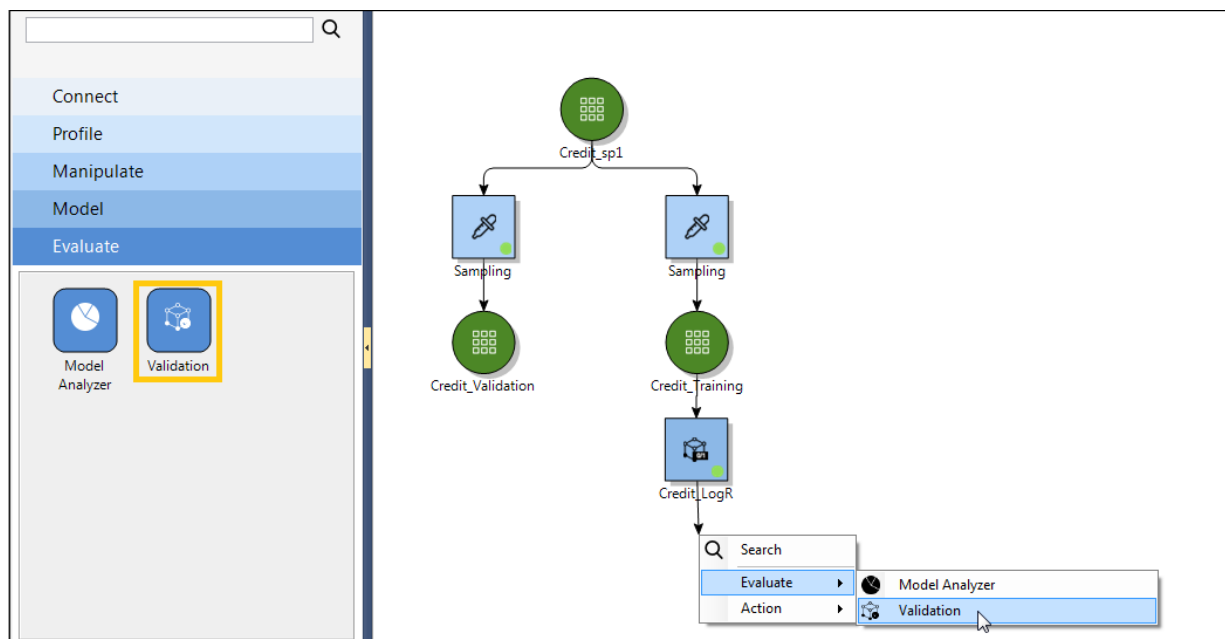
Validation can be performed for all predictive models, decision trees, and strategy trees. Validation is not applicable to Cluster Analysis. Model validation produces a dataframe with predictions and comparison metrics. This dataframe can then be used as an input for Model Analyzer to produce charts such as cumulative lift.

To add a validation operation to your workflow:

- Drag a connection arrow from a Spark model node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the context menu, select **Evaluate --> Validation** as in the example screenshot below. A new **Validation** process node will be created.

OR:

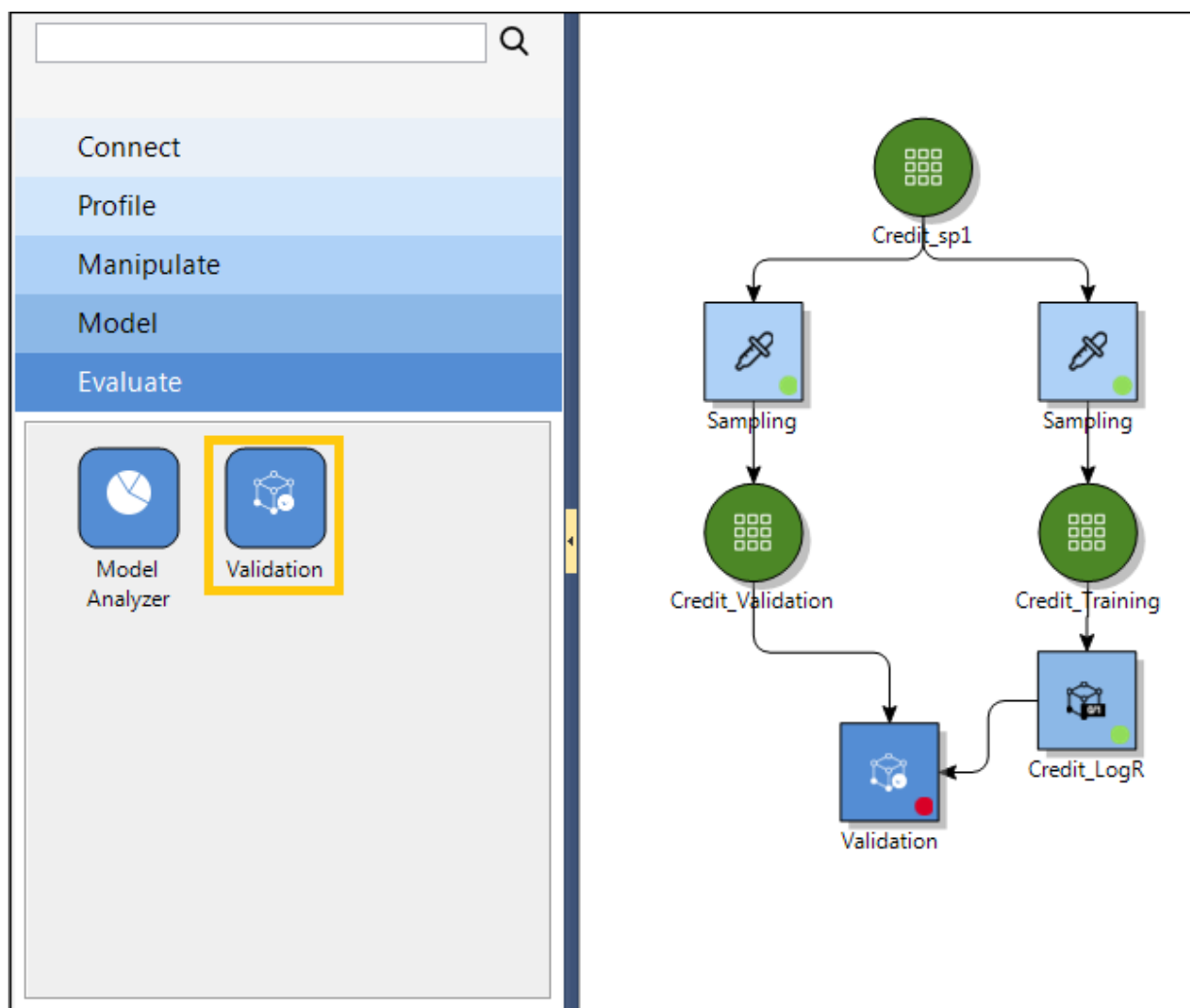
- Drag the **Validation** node from the **Evaluate** palette onto the canvas and connect the model and the input dataframe to it.



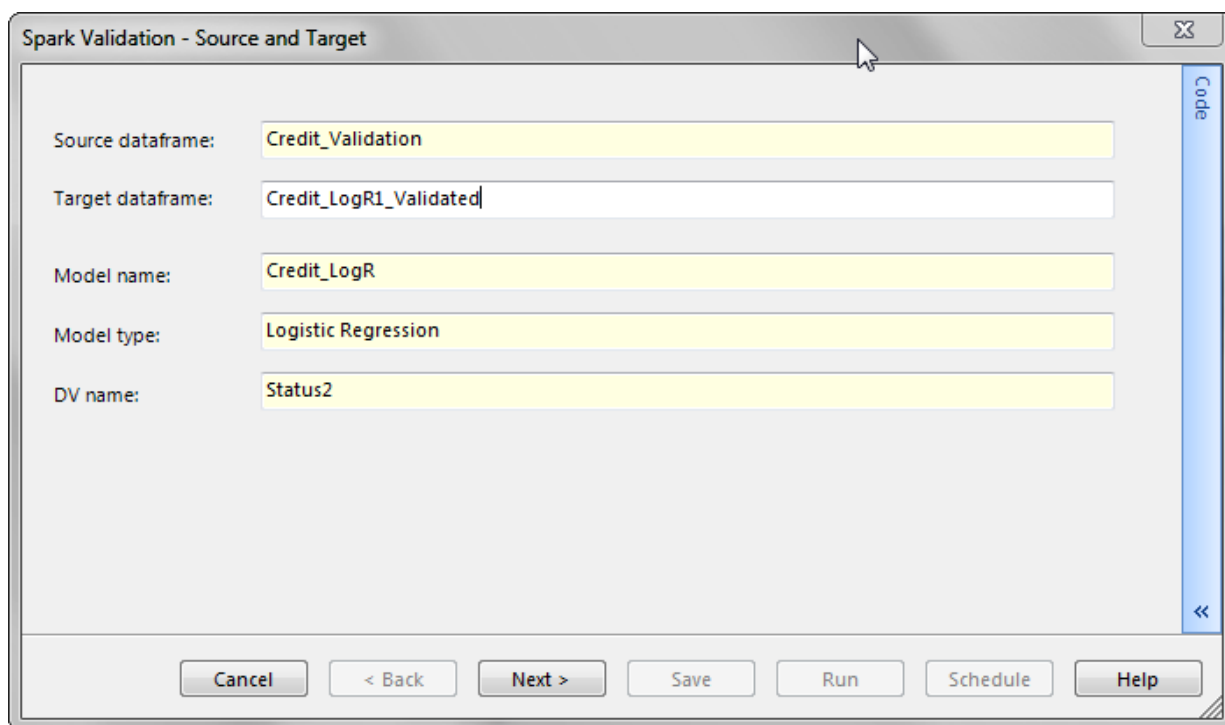
To define the validation parameters, double-click the **Validation** node or right-click on it and select **Modify**.

## Step 1

Specify the name of the output dataframe in the Target Dataset name field. The name of the input model, input dataframe, and the name of the dependent variable are also displayed in the first page.







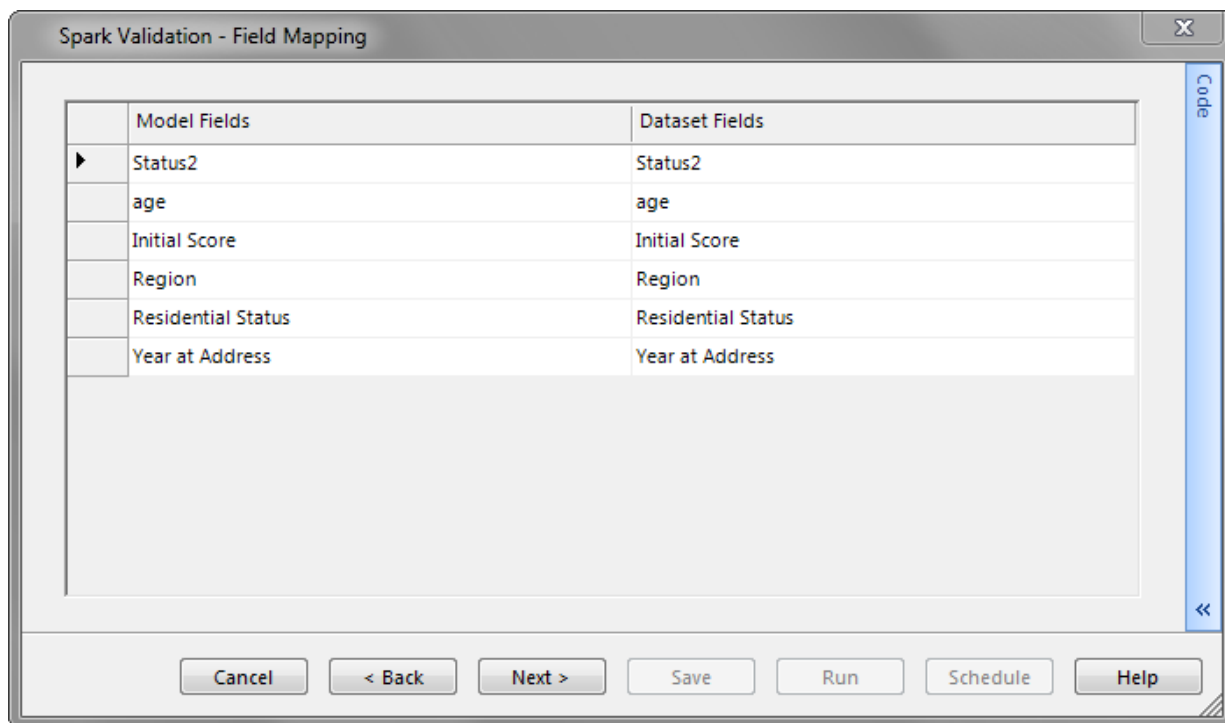
The dialog box titled "Spark Validation - Source and Target" contains the following fields:

| Field             | Value                  |
|-------------------|------------------------|
| Source dataframe: | Credit_Validation      |
| Target dataframe: | Credit_LogR1_Validated |
| Model name:       | Credit_LogR            |
| Model type:       | Logistic Regression    |
| DV name:          | Status2                |

At the bottom of the dialog are the following buttons: Cancel, < Back, Next >, Save, Run, Schedule, and Help. A vertical "Code" button is on the right side.

## Step 2

The next step provides the option of mapping the field names in the model to the ones in the validation input dataframe. In some situations, the names of model variables may not match the names of the corresponding fields in the input dataframe, so you will need to manually map fields by clicking in the right column (Dataset Field Names) and selecting the right fields to match the model variables in the left column:



### Step 3

At the next step, choose the new validation fields to be generated for the output dataframe. Generated field names are prefixed with the name of the dependent variable. In the case of a binary DV (a DV with two values), you can also specify a **cut off** value (see the explanation below).

The validation fields produced by validating models with discrete dependent variable (DV) are different from those produced by models with continuous DV. For example, for a Decision Tree model with discrete DV, the scoring fields are: DV Prediction, DV Probability of Prediction, DV Value Probability (for each DV value), DV Prediction Correct, Node ID, and Node Number.

For a Logistic Regression model with discrete DV, the validation fields are: DV Prediction, DV Probability of Prediction, and DV Value Probability for each DV value.

Spark Validation - Validation Fields

| Item                              | Field Name                        | Include                             | Cut Off |
|-----------------------------------|-----------------------------------|-------------------------------------|---------|
| Status2 Prediction                | Status2 Prediction                | <input checked="" type="checkbox"/> |         |
| Status2 Probability of Prediction | Status2 Probability of Prediction | <input type="checkbox"/>            |         |
| Status2 Bad Probability           | Status2 Bad Probability           | <input checked="" type="checkbox"/> | 0.5     |
| Status2 Good Probability          | Status2 Good Probability          | <input type="checkbox"/>            | 0.5     |
| Status2 Prediction Correct        | Status2 Prediction Correct        | <input checked="" type="checkbox"/> |         |

Code

Cancel < Back Next > Save Run Schedule Help

**Prediction**

The predicted value for a given record.

**DV Value Probability**

The probability of the specified outcome (value). For example, the field *Income >50K Prob* contains the probability with which the model predicts the value ">50K". Usually, the probability of the value of interest is the only field necessary for the analysis. The probabilities of other DV values are optional.

Here you can also specify the probability cut off point at which the prediction will be set in the DV Prediction field. For example, if your dependent variable is *Income*, with two values, ">50K" and "<=50K", and you set the cut-off for *Income <=50K Probability* to **0.7**, the *Income Prediction* field will be set to "<=50K" when the *Income <=50K Probability* value is **0.7** or greater. Note that the other dependent variable value probability cut-offs will automatically adjust to the total of 1 when you change the probability cut-off for one value.

**Probability of Prediction Prediction Correct**

The probability of the predicted value produced by the model. (This is obviously the highest probability for all possible predicted values.)

Indicates whether the predicted value coincides with the actual DV value. Provides a quick way to determine if the prediction is correct for a given record.

Validation fields for models with continuous DV are different:

Validation - Validation Fields

| Item                            | Field Name                      | Include                             | Cut Off | Hit Range |
|---------------------------------|---------------------------------|-------------------------------------|---------|-----------|
| capital-loss Prediction         | capital-loss Prediction         | <input checked="" type="checkbox"/> |         | 0         |
| capital-loss Prediction Correct | capital-loss Correct            | <input checked="" type="checkbox"/> |         |           |
| capital-loss Std Dev            | capital-loss Std Dev            | <input checked="" type="checkbox"/> |         |           |
| capital-loss Error              | capital-loss Error              | <input checked="" type="checkbox"/> |         |           |
| capital-loss Standardized Error | capital-loss Standardized Error | <input checked="" type="checkbox"/> |         |           |
| capital-loss Absolute Error     | capital-loss Absolute Error     | <input checked="" type="checkbox"/> |         |           |
| capital-loss Squared Error      | capital-loss Squared Error      | <input checked="" type="checkbox"/> |         |           |
| capital-loss Node ID            | capital-loss Node ID            | <input checked="" type="checkbox"/> |         |           |
| capital-loss Node Number        | capital-loss Node Number        | <input checked="" type="checkbox"/> |         |           |

**Prediction**

The predicted value for a given record.

**Correct**

Indicates whether the predicted value is close enough to the actual DV value to be considered a correct prediction. This depends on the user-defined valid hit range that can be specified in the Hit Range column on the right. By default, the Hit Range value equals the Valid Hit Range that was set in the model training wizard. You can change it at this step, if necessary. If you want a prediction to be considered correct whenever it falls into the interval  $[x-r, x+r]$ , where  $x$  is an actual DV value, specify  $r$  as your Valid Hit Range.

**Std Dev**

The standard deviation of the DV based on the distribution in the entire dataframe.

**Error**

The difference between the actual value and the predicted value.

**Standardized**

The standard deviation of the error.

**Error****Absolute**

The absolute value of the error.

**Error****Squared**

The squared value of the error.

**Error**

## Step 4

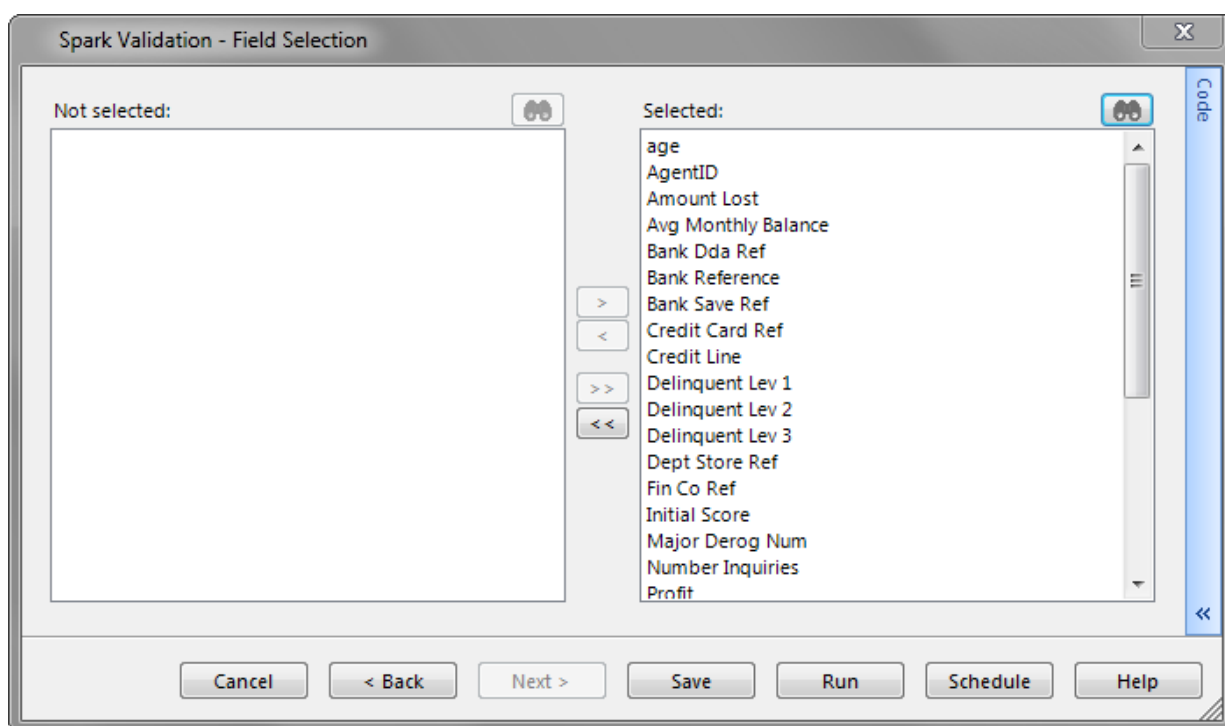
At the **Field Selection** step, choose the fields from the input dataframe that you would like to include in the validation output in addition to the new score / probability fields. The fields required by the model cannot be excluded. By default, all fields in the validation input dataframe are selected to be included in the output.

Page | 537

### Tip:

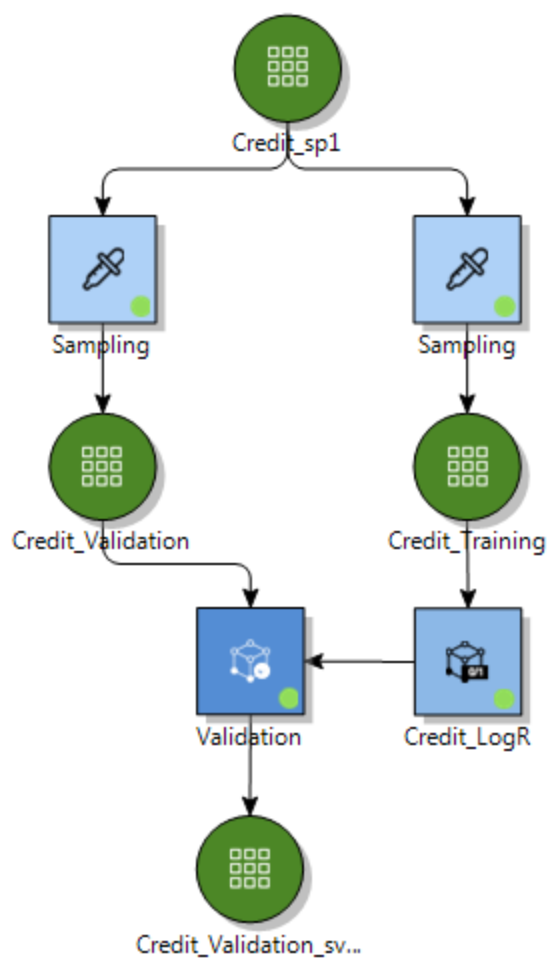
You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.

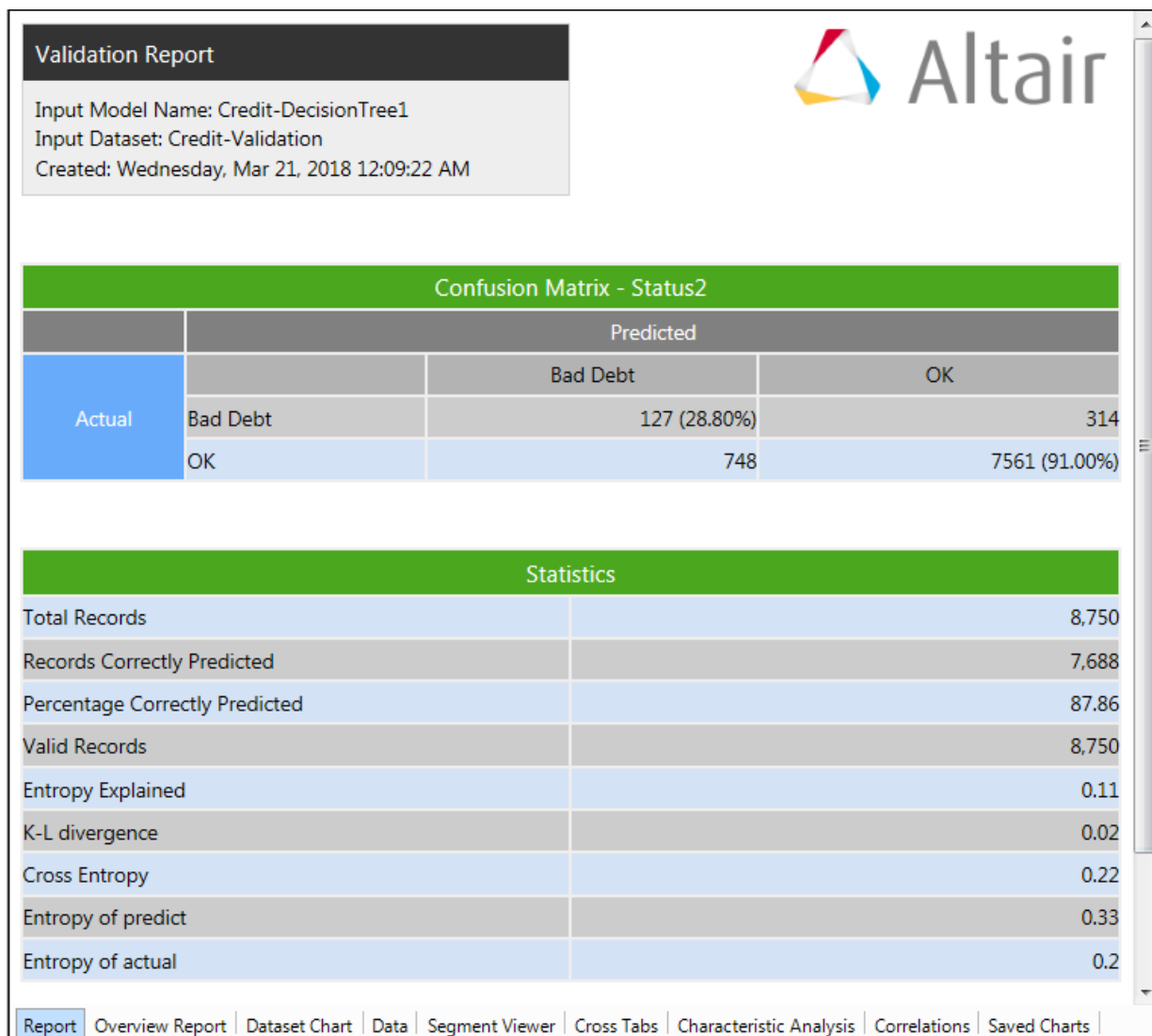


Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe node will be created in the workflow after running the validation node.



The top section of the Validation Report displays the name of the input dataset and the input model, as well as the validation date and time. The Confusion Matrix and the Statistics table show parameters characterizing the prediction accuracy of the model.



The **Confusion Matrix** shows how many of the predictions produced by model validation were correct and how many were wrong, i.e., did and did not coincide with actual DV values.

In the Validation Report example shown above, the validated model has the dependent variable *Status2* with values 'Bad Debt' and 'OK'. The Confusion Matrix shows that the actual 'Bad Debt' value was correctly predicted in 127 cases (28.8% of all Bad Debt values) and wrongly predicted as 'OK' in 314 cases. The 'OK' value was correctly predicted in 7561 cases (91% of all OK values) and wrongly predicted as 'Bad Debt' in 748 cases.

Note that for models with binary dependent variable, all the numeric results in the report depend on the **cut-off** value specified in the Validation Fields page of the Validation Wizard, since the predictions, i.e., the way the probabilities are interpreted, will be different depending on the cut-off. If we re-validate the model with the reduced 'Bad Debt' probability, the results will be slightly better

for 'Bad Debt' prediction accuracy. See the description of the Validation Fields page in the [Validation Wizard](#) for details on the cut-off value.

The **Statistics** table provides the following information:

- **Total Records:** The total number of input records.
- **Correctly Predicted:** The number of records with correctly predicted target outcome. Note: This depends on the cut-off value for discrete models and on the valid hit range for continuous models.
- **Percentage:** The Percentage of records with correctly predicted target outcome.
- **Valid Records:** This is the number of records for which the probabilities could be successfully calculated and therefore predictions were produced. Invalid records are the records in the input dataset for which the probabilities could not be calculated because the values in some fields were unknown to the model (for example, this can happen if the training dataset has values Male and Female and the input dataset for validation has Male, Female, and Unspecified) .
- **K-L Divergence:** The Kullback-Leibler (K-L) divergence, or relative entropy, is a quantity which measures the difference between two probability distributions.
- **Cross Entropy:** Cross entropy between two probability distributions measures the overall difference between the two distributions. This measures how well a distribution approximates another distribution.

Page | 540

## See Also:

[About Scoring](#)

## Model Analyzer

The **Model Analyzer** node is used to evaluate and compare model performance using a set of charts and statistics. Model performance is analyzed based on dataframes obtained as a result of model validation. In addition to evaluating how well a model performs on different dataframes, you can compare the performance of different models on a dataframe.

If you have a validated dataframe (a dataframe that is a result of validating a model), Model Analyzer can estimate the performance of the model on the basis of this dataframe. The necessary condition is that the dataframe must have a field with actual values of the variable of interest and the field with values predicted by the model (DV prediction field or Scorecard field). In Knowledge Studio for Apache Spark, dataframes with these fields are generated as a result of validating a model (see the Model Validation topic). However, the origin of these dataframes does not really matter (you can even use dataframes with prediction fields generated in other applications). The most important thing is the presence of the fields with predicted values to compare to actual values.

To add a Model Analyzer to your workflow:

- Drag a connection arrow from a Spark model node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the

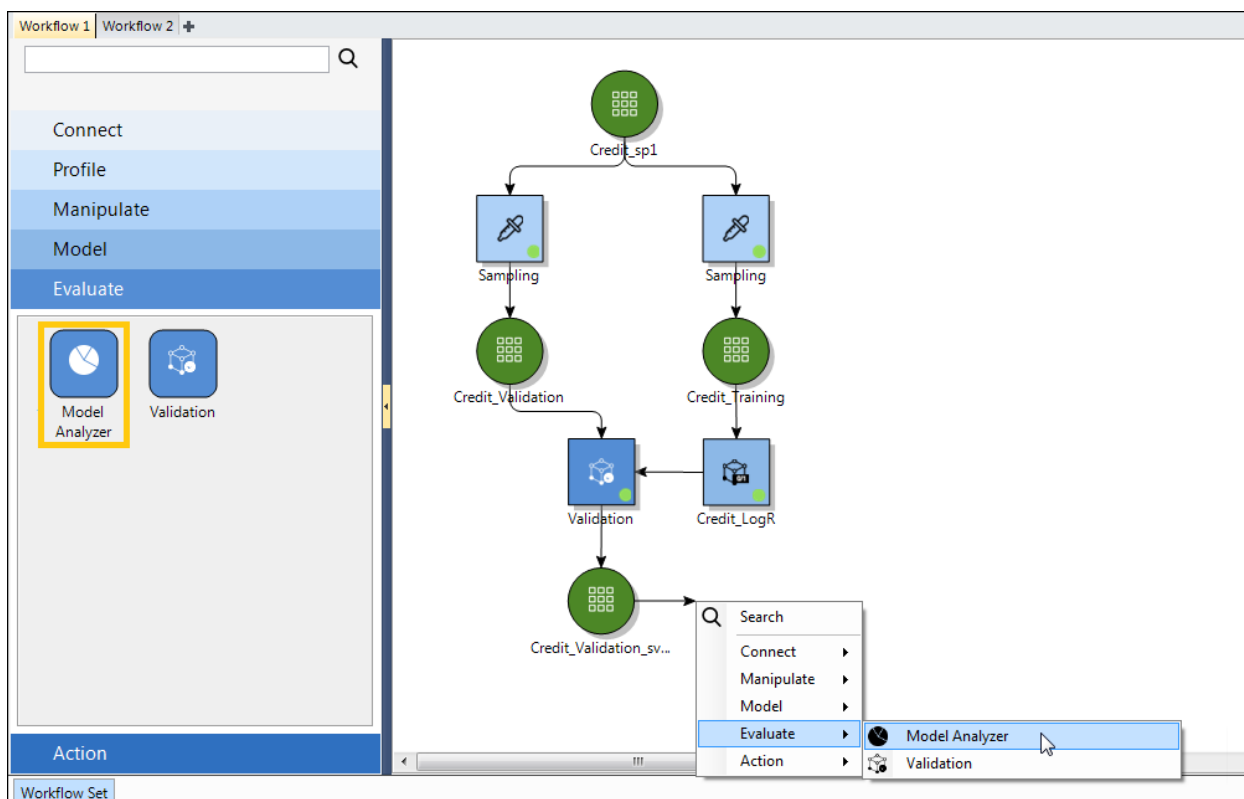


context menu, select **Evaluate --> Model Analyzer** as in the example screenshot below. A new **Model Analyzer** process node will be created.

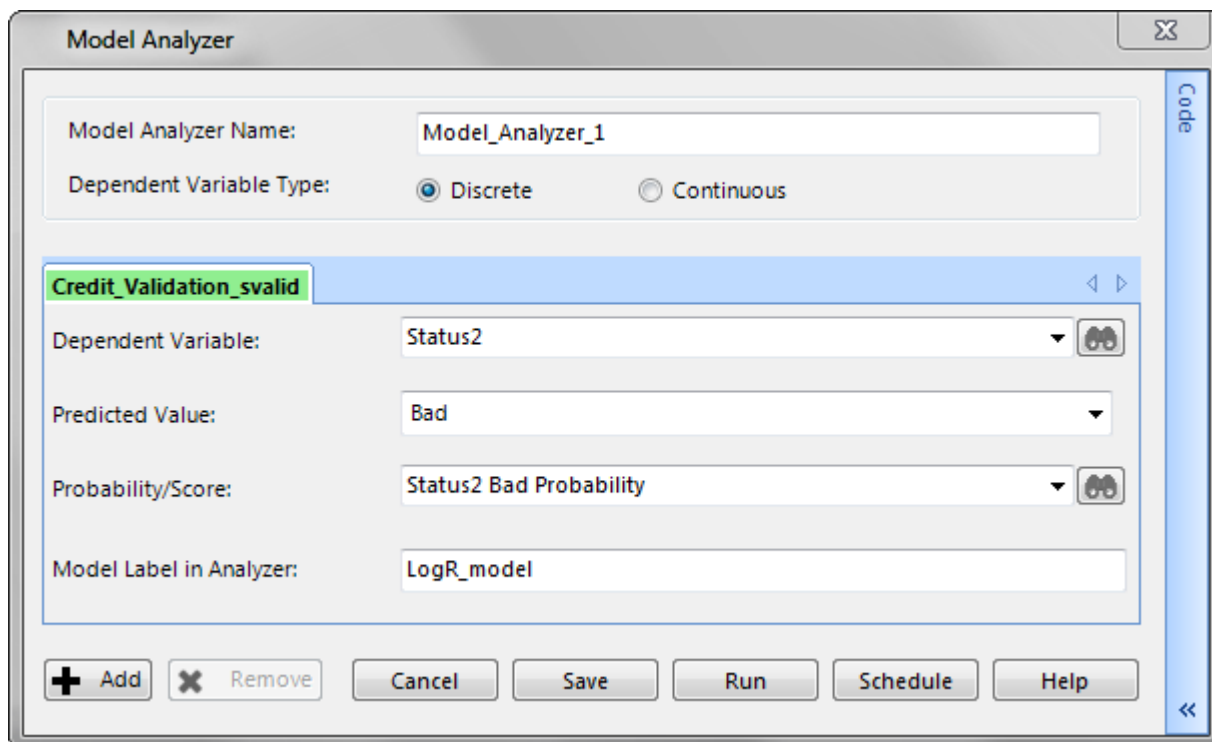
OR:

- Drag the **Model Analyzer** node from the **Evaluate** palette onto the canvas and connect the model and the input dataframe to it.

Page | 541



If the wizard does not open automatically, double-click on **Model Analyzer** node or right-click on it and select Modify from the context menu.


**Dependent Variable**

The field containing the values of the dependent variable of the validated model.

**Predicted Value**

The target value predicted by the model that produced the input validation dataframe.

**Probability/Score**

The field that contains the probability of the predicted value or the scores of a scorerecord.

Choose the appropriate sorting depending on how the scores are sorted in your input dataframe with respect to the target value (selected Predicted Value).

The typical structure of a dataframe produced by the Validation node is as follows (only the fields relevant to the model prediction are shown):

Figure 1 - Validated model

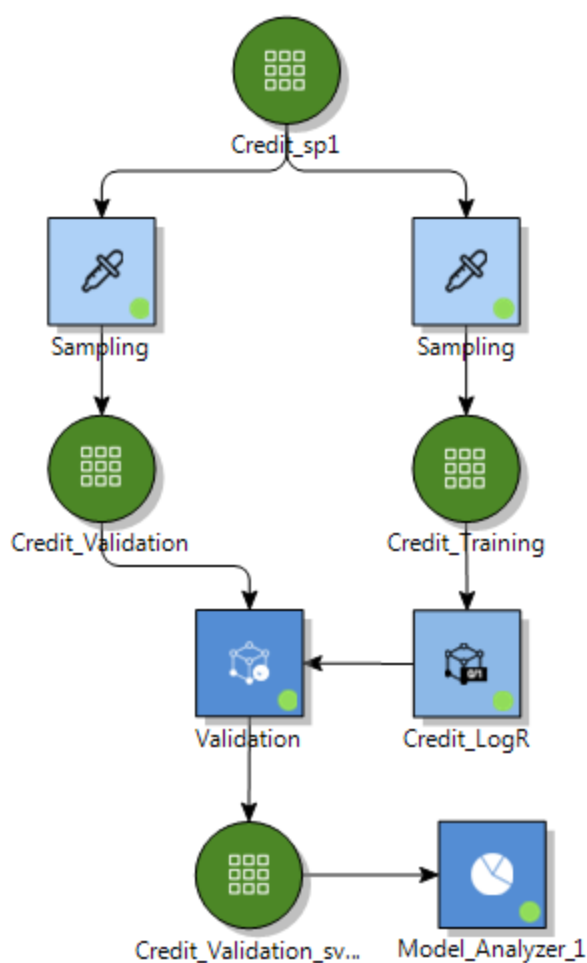
|  | Income | Income Prediction | Income Predict Prob | Income <=50K Prob | Income >50K Prob | Income Correct                      |
|--|--------|-------------------|---------------------|-------------------|------------------|-------------------------------------|
|  | <=50K  | <=50K             | 0.99752662935081    | 0.99752662935081  | 0.00247337064919 | <input checked="" type="checkbox"/> |
|  | <=50K  | <=50K             | 0.85896117393076    | 0.85896117393076  | 0.14103882606924 | <input checked="" type="checkbox"/> |
|  | >50K   | <=50K             | 0.59785440122996    | 0.59785440122996  | 0.40214559877004 | <input type="checkbox"/>            |
|  | >50K   | >50K              | 0.816749257712649   | 0.18325074228735  | 0.81674925771264 | <input checked="" type="checkbox"/> |
|  | <=50K  | <=50K             | 0.997994121929173   | 0.99799412192917  | 0.00200587807082 | <input checked="" type="checkbox"/> |

In this example, the DV is **Income**. It has two categories, "**<=50K**" and "**>50K**". The **Income Prediction** field contains values predicted by the model (either "**<=50K**" or "**>50K**"). The probability of the model predicting the value "**<=50**" or "**>50**" is displayed in the fields **Income <=50 Prob** and **Income >50 Prob**, respectively. The **Income Predict Prob** field contains the probability of the predicted value (i.e., the probability of the value in the **Income Prediction** field).

Page | 543

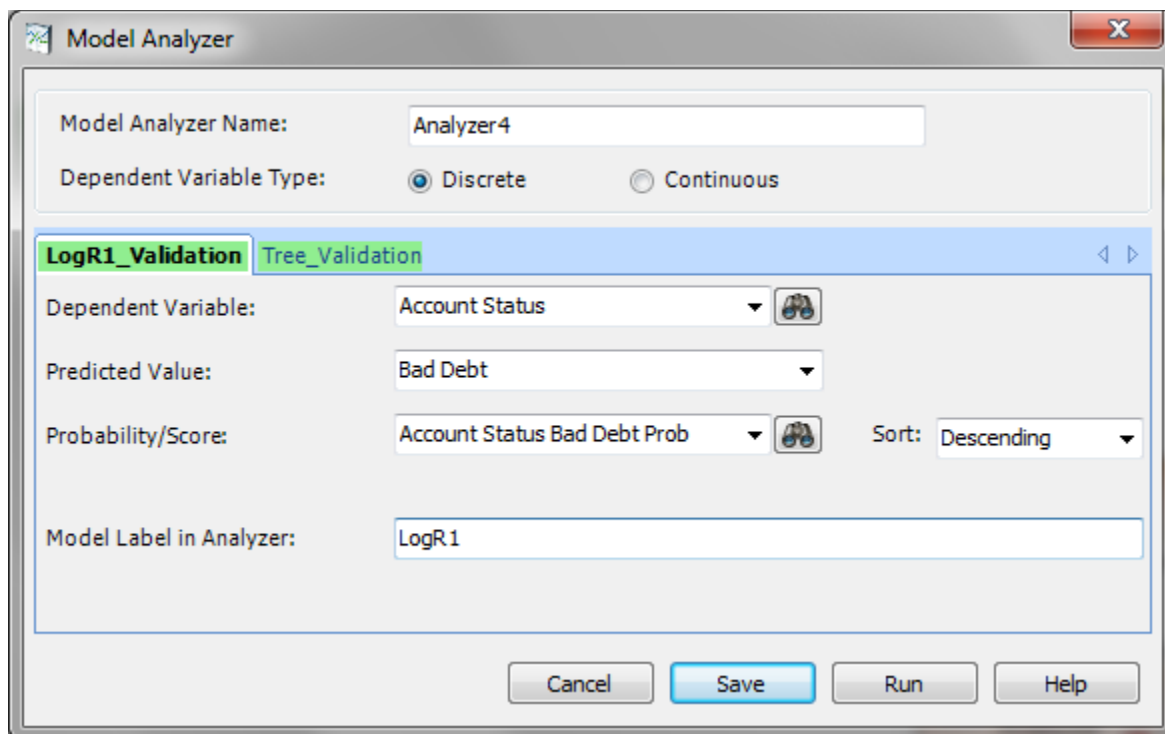
Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output Model Analyzer node will be created in the workflow after running the validation node. Double-click on it to see the model evaluation charts and reports.



You can add more than one dataset with model validation results to the model analyzer to compare their performance. To add another input dataset, connect it to the Model Analyzer node. Each input dataset is represented by a separate tab. Click on each tab to define the settings for each input..

In the example below, two input datasets - LogR1\_Validation and Tree\_Validation - resulting from the validation of two different models are represented by two tabs in the Model Analyzer.



Page | 544

To remove an input dataset from a Model Analyzer, just remove the connection from the dataset node to the analyzer node in the workflow by clicking on the connection arrow and pressing the Delete key.

## Models Analyzer Views

The Analysis Results view shows the following charts, reports, and statistics for model performance evaluation and comparison:

- Cumulative lift chart
- Area under the ROC curve
- Area under the PR curve (precision-recall curve)
- Lift Report (by deciles)

A **cumulative lift chart** illustrates how effective the model is in isolating the target group among the total population. The scored records are sorted in the descending order of the probabilities of the target category, and the percentages of target are plotted on the vertical axis against the percentage of the total population. Thus, the chart measures the effectiveness of the model compared to random selection.

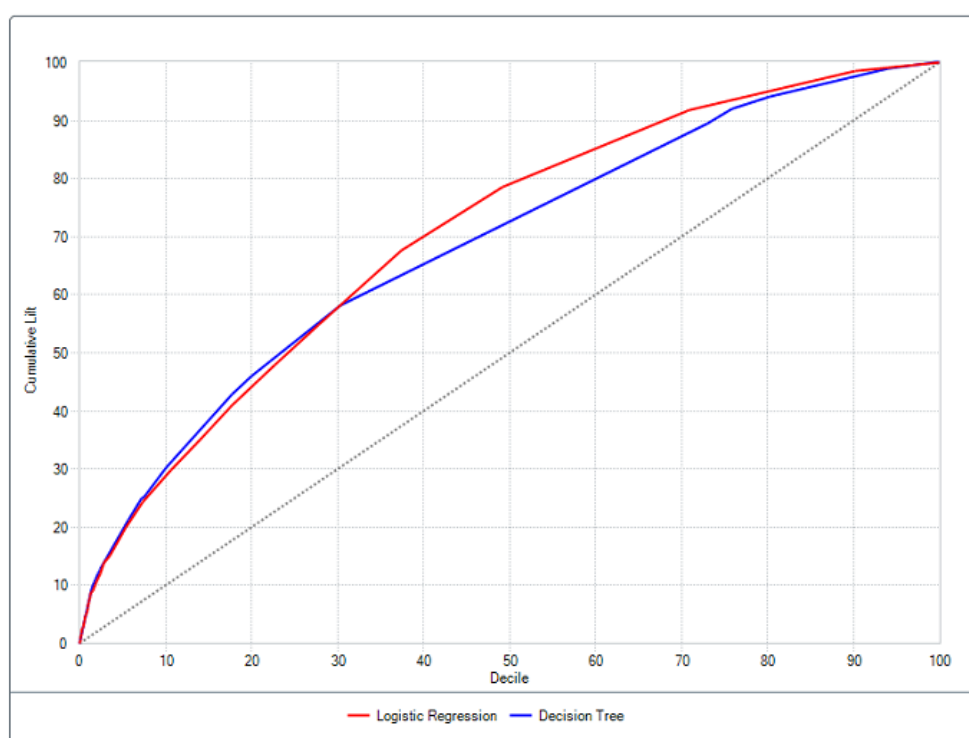
Below is an example of the Cumulative Lift chart with two models - a Decision Tree and a Logistic Regression model. The chart allows the user to easily determine which model is more effective. For details, see the topic [Calculating Cumulative Lift](#).

Model Analyzer Name: Model\_Comparison\_LR\_T

Model Type: Discrete

Page | 545

### Cumulative Lift



The *Cumulative Lift Report* shows the values of lift and cumulative lift at the decile level for each model in the Model Analyzer. If more than one model was added to the Model Analyzer, the Analysis Results view shows the report for each model. See the topic [Calculating Cumulative Lift](#) for the detailed information on how to calculate lift and cumulative lift.

The *Number\_of\_Records* column shows the number of records in each decile of the validation dataset for the selected model. The *Actual\_Target* column shows the number of records with target values of the DV in each decile. The values in the *Cumulative\_Actual\_Target* are obtained by adding up the values in the *Actual\_Target* column as you go down the deciles. The Cumulative Lift is the ratio of *Cumulative\_Actual\_Target* and the total number of target records.

For each model, a separate table shows the values of the Area under the ROC curve (ROC AUC) and the Area under the precision-recall curve (PR AUC). For details, see the ROC Chart topic.

| Logistic Regression - Area |               |
|----------------------------|---------------|
| area_under_ROC             | area_under_PR |
| 0.714165                   | 0.126832      |

Page | 546

| Logistic Regression - Lift |                   |               |                          |                 |
|----------------------------|-------------------|---------------|--------------------------|-----------------|
| Decile                     | Number_of_Records | Actual_Target | Cumulative_Actual_Target | Cumulative_Lift |
| 1                          | 2916              | 404           | 404                      | 0.283708        |
| 2                          | 2916              | 230           | 634                      | 0.445225        |
| 3                          | 2916              | 194           | 828                      | 0.581461        |
| 4                          | 2916              | 173           | 1001                     | 0.702949        |
| 5                          | 2916              | 127           | 1128                     | 0.792135        |
| 6                          | 2916              | 92            | 1220                     | 0.856742        |
| 7                          | 2916              | 77            | 1297                     | 0.910815        |
| 8                          | 2916              | 60            | 1357                     | 0.952949        |
| 9                          | 2916              | 44            | 1401                     | 0.983848        |
| 10                         | 2923              | 23            | 1424                     | 1               |

| Decision Tree - Area |               |
|----------------------|---------------|
| area_under_ROC       | area_under_PR |
| 0.694633             | 0.124408      |

| Decision Tree - Lift |                   |               |                          |                 |
|----------------------|-------------------|---------------|--------------------------|-----------------|
| Decile               | Number_of_Records | Actual_Target | Cumulative_Actual_Target | Cumulative_Lift |
| 1                    | 2916              | 427           | 427                      | 0.29986         |
| 2                    | 2916              | 229           | 656                      | 0.460674        |
| 3                    | 2916              | 166           | 822                      | 0.577247        |
| 4                    | 2916              | 116           | 938                      | 0.658708        |
| 5                    | 2916              | 110           | 1048                     | 0.735955        |
| 6                    | 2916              | 105           | 1153                     | 0.809691        |
| 7                    | 2916              | 85            | 1238                     | 0.869382        |
| 8                    | 2916              | 100           | 1338                     | 0.939607        |
| 9                    | 2916              | 45            | 1383                     | 0.971208        |
| 10                   | 2923              | 41            | 1424                     | 1               |

## Calculating Cumulative Lift

The cumulative lift is based on the probabilities of the category of interest plotted against the percentage of the total population.

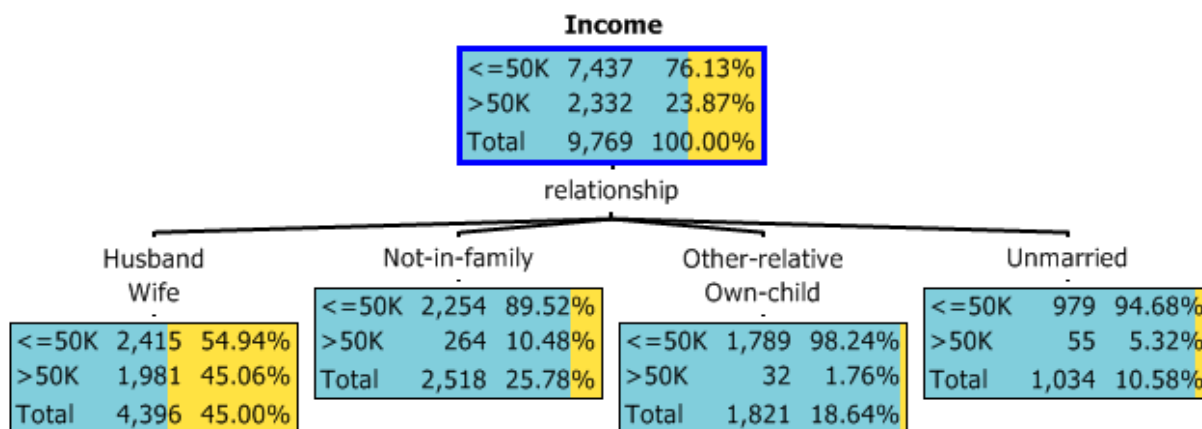
In our example, we insert a decision tree with DV=Income.

- Income has two categories, <=50K and >50K.
- We are interested in the category >50K.
- The total population is the total number of records used in the tree. In this example, we inserted a tree with 9769 records so our total population=9769.

Page | 547

Decision trees typically have more than one level and many more terminal nodes. To keep the example simple, we only grew the tree to one level beyond the root node. This yielded four terminal nodes.

Figure 1 - Decision Tree



The table below shows the normal and cumulative frequency and percentage of the category of interest (Income is ">50K") at each child node. The table is sorted by the probability of Income > 50K, which is the second column. Therefore, in terms of the node color, the child nodes are ordered by the size of their yellow portion, which indicates the percentage of >50K.

Table 1 - Cumulative Lift Calculation

| Node rule                    | Probability | Total ">50" in Node | Cumulative Total >50K | Cumulative Lift (Y-axis) | Total Records in Node | Cumulative Total of Nodes | Cumulative % of Population axis) |
|------------------------------|-------------|---------------------|-----------------------|--------------------------|-----------------------|---------------------------|----------------------------------|
| relationship = Husband, Wife | 45.06       | 1981                | 1981                  | 84.95%                   | 4396                  | 4396                      | 45.0                             |
| relationship = Not-in-family | 10.48       | 264                 | 2245                  | 96.27%                   | 2518                  | 6914                      | 70.7                             |
| relationship = Unmarried     | 5.32        | 55                  | 2300                  | 98.63%                   | 1034                  | 7948                      | 81.3                             |
| relationship = Other-        | 1.76        | 32                  | 2332                  | 100.00%                  | 1821                  | 9769                      | 100.0                            |

|                        |  |  |  |  |  |  |  |
|------------------------|--|--|--|--|--|--|--|
| relative,<br>Own-child |  |  |  |  |  |  |  |
|------------------------|--|--|--|--|--|--|--|

## Calculating the coordinates for the X-axis

Page | 548

The X-axis represents the cumulative percentage of the total population. Our total population is 9769 records. For each node, the calculation is as follows:

$$(\text{Cumulative Total of Nodes} / \text{Total Population}) * 100$$

For the first point representing the node [relationship = Husband, Wife], this is  $(4396 / 9769) * 100 = 0.4499949 * 100 = 45$  (after rounding)

For the second point representing the first two nodes, this is  $(6914 / 9769) * 100 = 0.7077490 * 100 = 70.77$  (after rounding)

Similar calculations yield **81.36%** for the first three child nodes, and obviously 100% for all four.

## Calculating the coordinates for the Y-axis

The **Y-axis** represents the cumulative percentage of the records where the DV category of interest is **>50K**. The total number of records with Income >50K is 2332, as displayed in the root node. For each node, the calculation is as follows:

$$(\text{Cumulative Total } >50K / \text{Total records with Income } >50K) * 100$$

For the first point representing the node [relationship = Husband, Wife], the cumulative lift is  $(1981 / 2332) * 100 = 0.8494854 * 100 = 84.95$  (after rounding)

For the second point representing the first two nodes, the cumulative lift is  $(2245 / 2332) * 100 = 0.9626929 * 100 = 96.27$  (after rounding)

Similar calculations yield **98.63%** for the first three child nodes, and finally 100% for all four.

## The Chart

From the above calculations, we have four points for the cumulative lift chart for the decision tree in Figure 1 above:

(45.00, 84.95)

(70.77, 96.27)

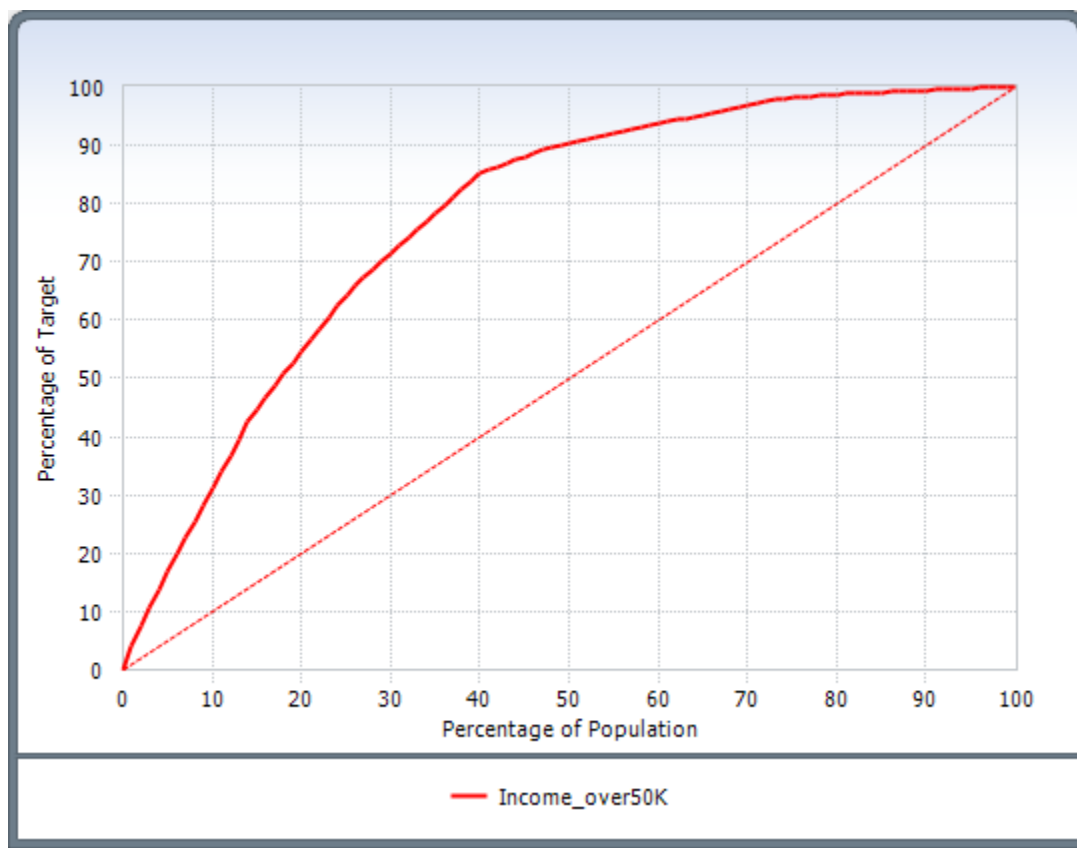
(81.36, 98.63)

(100, 100).



Plotting these points, we obtain the following chart:

Figure 2 - Cumulative Lift Chart



Page | 549

Suppose our business objective is to create a mail campaign to target people who have an annual income >50K.

The chart shows that if we target approximately 45% of the population, we will reach 84.95% of the people with Income >50K.

By default, the y-axis shows the percentage of target values. You can choose to display absolute values instead of percentages by selecting **View | Show Absolute** from the main menu.

## Scoring

For Decision Trees and Predictive Models, scoring is the process of predicting the values of the dependent variable (DV) for records in a new dataset. Scoring is usually done after completing the [validation](#) process, in which you assess how well your model performs on test data.

The **score** is the probability that the predicted outcome will occur. In contrast to validation, in scoring the target dataset is assumed to have no DV field, the correct outcome is not known, so no comparison of actual vs. predicted outcome takes place.

For Cluster Analysis models, scoring is assigning a record to a cluster, computing the probability for each record to belong to a cluster, and computing distances to the corresponding cluster center.

## Scoring Fields

During the scoring process, additional fields are created. These fields can be either part of a newly generated dataset, appended to the target dataset, or exported to an external file or database table. Not all of the generated fields are mandatory. During the scoring wizard setup, you choose which additional fields you want to include.

Models with discrete DV produce fields different from those produced by models with continuous DV. The scoring of Cluster Analysis models produces fields specific for clustering (Cluster Index, Cluster Probability, Distance to Center).

### Scoring Fields for Models with Discrete DV

Figure 1

| Income Prediction | Income <=50 | Income >50K Prob  | Income Predict Prob |
|-------------------|-------------|-------------------|---------------------|
| <=50K             | 1           | 0                 | 1                   |
| >50K              | 0           | 1                 | 1                   |
| >50K              | 0.090909090 | 0.909090909090909 | 0.909090909090909   |
| <=50K             | 1           | 0                 | 1                   |
| <=50K             | 0.701492537 | 0.29850746268656  | 0.701492537313433   |
| <=50K             | 0.713414634 | 0.28658536585365  | 0.713414634146341   |
| >50K              | 0.271186440 | 0.72881355932203  | 0.728813559322034   |

The score is the probability of the DV value of interest for a given record. For example, if you are interested in the records where the value ">50K" is predicted for your dependent variable *Income*, the values in the field **Income >50K Prob** represent the scores.

The scores are compared with the probability **cut off** value (specified in the scoring wizard) which determines the predicted value in the **DV Prediction** field (**Income Prediction** in this example). If the cut off was defined as **0.5** (default), then any score value on or above **0.5** will result in the predicted value ">50K" and a score value below **0.5** will result in the predicted value "<=50K".

#### **Income Prediction**

The predicted value for a given record.

#### **Income Prob and**

The probability of the specified outcome (value). For example, the field *Income >50K Prob* contains the probability of the value ">50K". Usually, the probability of the value of

|                    |                |                                                                                                           |
|--------------------|----------------|-----------------------------------------------------------------------------------------------------------|
| <b>Income</b>      | <b>&gt;50K</b> | interest is the only field necessary for the analysis. The probabilities of other DV values are optional. |
| <b>Prob</b>        |                |                                                                                                           |
| <b>Income</b>      | <b>Predict</b> | The probability of the predicted value (i.e., of the value in the Income Prediction field in this case).  |
| <b>Probability</b> |                |                                                                                                           |

## Scoring Fields for Models with Continuous DV

Page | 551

In Figure 2, the continuous DV **age** is predicted. In the continuous case, the **score** is the value in the **DV Prediction** field (**age Prediction** in this example).

Figure 2

| age Prediction | age Std Dev  | age Node ID |
|----------------|--------------|-------------|
| 42.569767441   | 13.975222357 | x.5/0.12/3  |
| 54.755208333   | 18.364188744 | x.5/2       |
| 54.755208333   | 18.364188744 | x.5/2       |
| 54.755208333   | 18.364188744 | x.5/2       |
| 42.569767441   | 13.975222357 | x.5/0.12/3  |
| 42.569767441   | 13.975222357 | x.5/0.12/3  |

|                       |                                                                                                                      |
|-----------------------|----------------------------------------------------------------------------------------------------------------------|
| <b>age Prediction</b> | The predicted value for a given record.                                                                              |
| <b>age Std Dev</b>    | The standard deviation of the DV can be used as the estimated precision of the prediction as estimated by the model. |
| <b>age Node ID</b>    | For Decision Tree models only, this is the node ID of the tree node to which the record is assigned.                 |

## Special Notes on Scoring Decision Trees

Scoring a decision tree is a little different from scoring other models due to the interactive character of trees and the fact that the scoring fields are calculated columns depending on the validated models.

If you validate a tree model, the calculated columns created with the scoring operation refer to a read-only copy of the tree being validated, not to the original tree. This read-only version of the original tree is created automatically by the scoring process and is called a “*snapshot tree*”. This prevents a variety of problems that can occur if a referenced model is changed after scoring (especially if scoring fields were appended to the parent dataset) and allows the user to delete and modify trees without breaking any dependencies. In addition, the user always has a copy of a tree as it was at the time of scoring.

Proceed to [Scoring Wizard](#).

For general information about model scoring, see the topic [About Scoring](#).

Scoring can be performed on all types of models - Decision Trees, regression, clustering, etc.

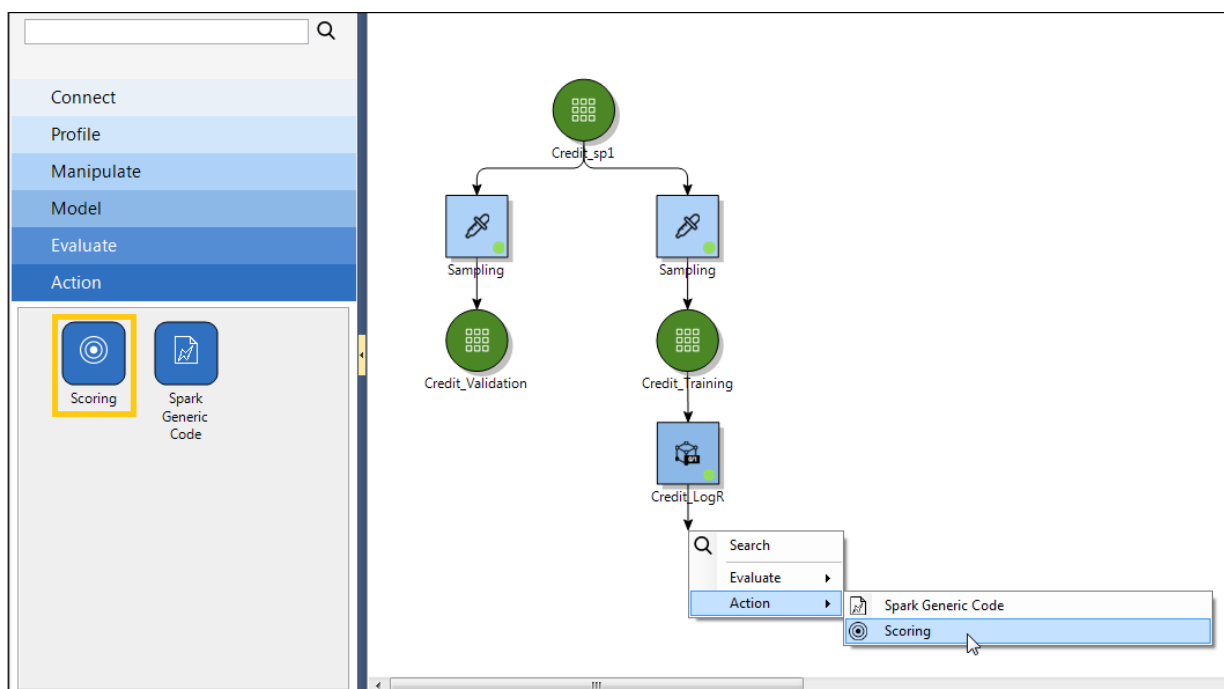
To add a Scoring operation to your workflow:

- Drag a connection arrow from a Spark model node to any area on the Workflow canvas background. When you release the mouse button, a context menu will open. From the context menu, select **Action --> Scoring** as in the example screenshot below. A new **Scoring** process node will be created.

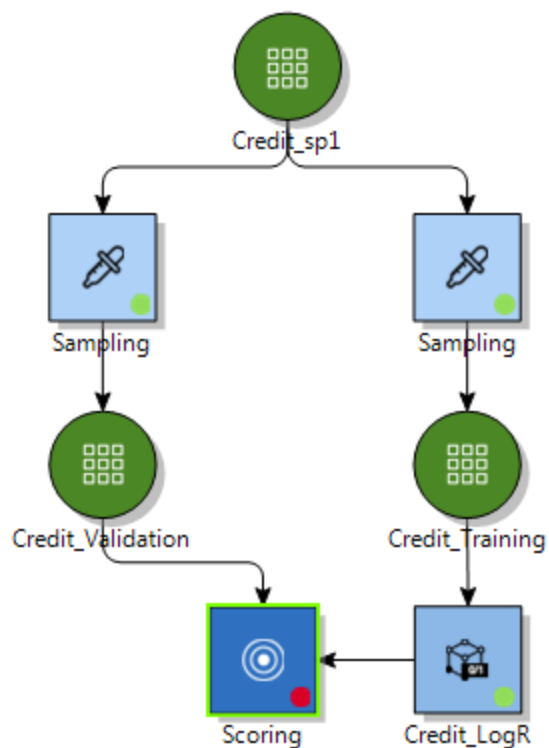
OR:

- Drag the **Scoring** node from the **Action** palette onto the canvas and connect the model and the input dataframe to it.

Page | 552

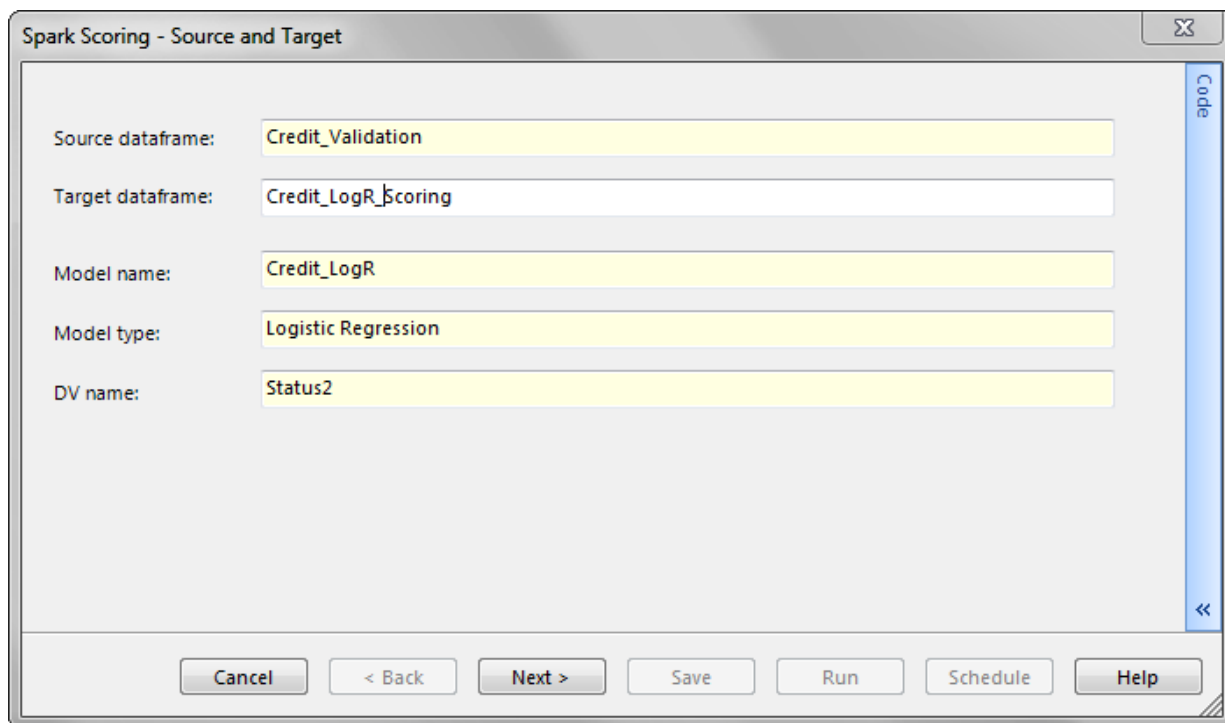


To define the scoring parameters, double-click the **Scoring** node or right-click on it and select **Modify**.



## Step 1

Specify the name of the output dataframe in the Target Dataset name field. The name of the input model, input dataframe, and the name of the dependent variable are also displayed in the first page.



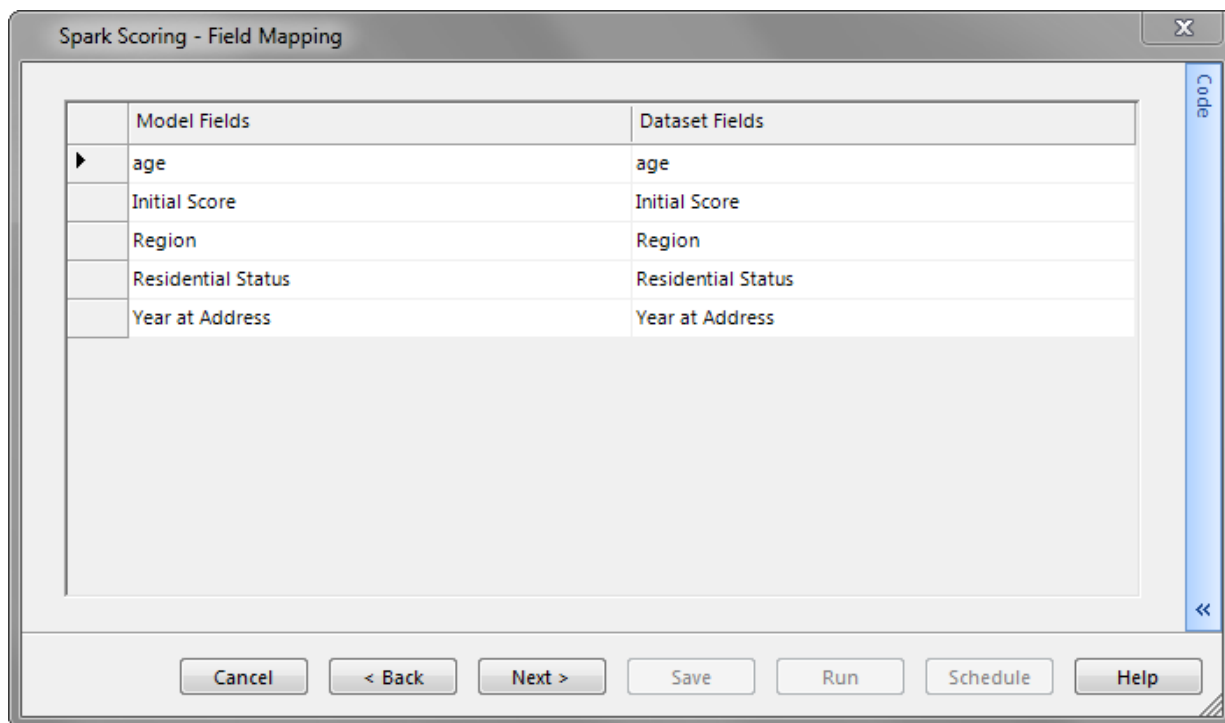
The dialog box titled "Spark Scoring - Source and Target" contains the following fields:

| Field             | Value               |
|-------------------|---------------------|
| Source dataframe: | Credit_Validation   |
| Target dataframe: | Credit_LogR_Scoring |
| Model name:       | Credit_LogR         |
| Model type:       | Logistic Regression |
| DV name:          | Status2             |

At the bottom of the dialog are the following buttons: Cancel, < Back, Next >, Save, Run, Schedule, and Help. A vertical "Code" button is on the right side.

## Step 2

The next step provides the option of mapping the field names in the model to the ones in the scoring input dataframe. In some situations, the names of model variables may not match the names of the corresponding fields in the input dataframe, so you will need to manually map fields by clicking in the right column (Dataset Field Names) and selecting the right fields to match the model variables in the left column:



### Step 3

At the next step, choose the new scoring fields to be generated for the output dataframe. Generated field names are prefixed with the name of the dependent variable. In the case of a binary DV (a DV with two values), you can also specify a **cut off** value (see the explanation below).

The scoring fields produced by validating models with discrete dependent variable (DV) are different from those produced by models with continuous DV. For example, for a Decision Tree model with discrete DV, the scoring fields are: DV Prediction, DV Probability of Prediction, DV Value Probability (for each DV value), Node ID, and Node Number.

For a Logistic Regression model with discrete DV, the scoring fields are: DV Prediction, DV Probability of Prediction, and DV Value Probability for each DV value.

Spark Scoring - Scoring Fields

| Item                              | Field Name                        | Include                             | Cut Off |
|-----------------------------------|-----------------------------------|-------------------------------------|---------|
| Status2 Prediction                | Status2 Prediction                | <input checked="" type="checkbox"/> |         |
| Status2 Probability of Prediction | Status2 Probability of Prediction | <input type="checkbox"/>            |         |
| Status2 Bad Probability           | Status2 Bad Probability           | <input checked="" type="checkbox"/> | 0.5     |
| Status2 Good Probability          | Status2 Good Probability          | <input type="checkbox"/>            | 0.5     |

Code

Cancel < Back Next > Save Run Schedule Help

**Prediction**

The predicted value for a given record.

**DV Value Probability**

The probability of the specified outcome (value). For example, the field *Income >50K Prob* contains the probability with which the model predicts the value ">50K". Usually, the probability of the value of interest is the only field necessary for the analysis. The probabilities of other DV values are optional.

Here you can also specify the probability cut off point at which the prediction will be set in the DV Prediction field. For example, if your dependent variable is *Income*, with two values, ">50K" and "<=50K", and you set the cut-off for *Income <=50K Probability* to **0.7**, the *Income Prediction* field will be set to "<=50K" when the *Income <=50K Probability* value is **0.7** or greater. Note that the other dependent variable value probability cut-offs will automatically adjust to the total of 1 when you change the probability cut-off for one value.

**Probability of Prediction**

The probability of the predicted value produced by the model. (This is obviously the highest probability for all possible predicted values.)

Scoring fields for models with continuous DV are different:



Validation - Validation Fields

| Item                            | Field Name                      | Include                             | Cut Off | Hit Range |
|---------------------------------|---------------------------------|-------------------------------------|---------|-----------|
| capital-loss Prediction         | capital-loss Prediction         | <input checked="" type="checkbox"/> |         | 0         |
| capital-loss Prediction Correct | capital-loss Correct            | <input checked="" type="checkbox"/> |         |           |
| capital-loss Std Dev            | capital-loss Std Dev            | <input checked="" type="checkbox"/> |         |           |
| capital-loss Error              | capital-loss Error              | <input checked="" type="checkbox"/> |         |           |
| capital-loss Standardized Error | capital-loss Standardized Error | <input checked="" type="checkbox"/> |         |           |
| capital-loss Absolute Error     | capital-loss Absolute Error     | <input checked="" type="checkbox"/> |         |           |
| capital-loss Squared Error      | capital-loss Squared Error      | <input checked="" type="checkbox"/> |         |           |
| capital-loss Node ID            | capital-loss Node ID            | <input checked="" type="checkbox"/> |         |           |
| capital-loss Node Number        | capital-loss Node Number        | <input checked="" type="checkbox"/> |         |           |

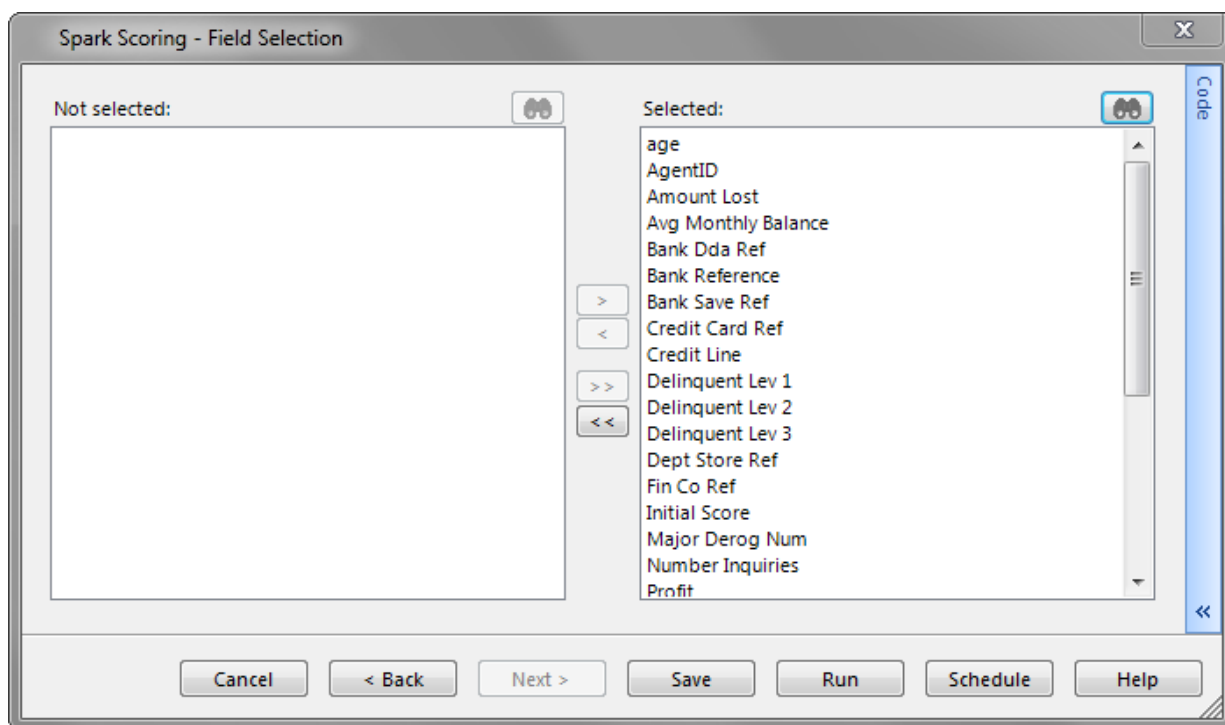
## Step 4

At the **Field Selection** step, choose the fields from the input dataframe that you would like to include in the scoring output in addition to the new score / probability fields. The fields required by the model cannot be excluded. By default, all fields in the scoring input dataframe are selected to be included in the output.

### Tip:

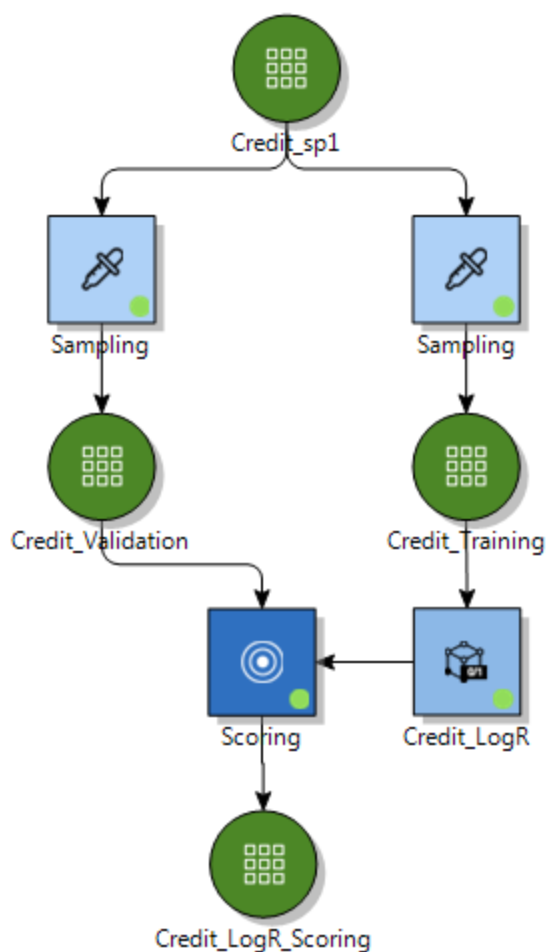
You can move the variables to and from the 'Selected' list by using the arrow buttons or double-clicking on their names. To select more than one variable in the lists, use mouse dragging, or mouse clicks while holding down the SHIFT or CTRL key, then click the appropriate arrow button to move the highlighted fields.

The [Field Finder](#) button  is especially useful for quick variable selection when there are lots of variables (hundreds or thousands), and it also allows you to apply a previously saved [Variable List](#) to make your selection. See the corresponding topics for details.



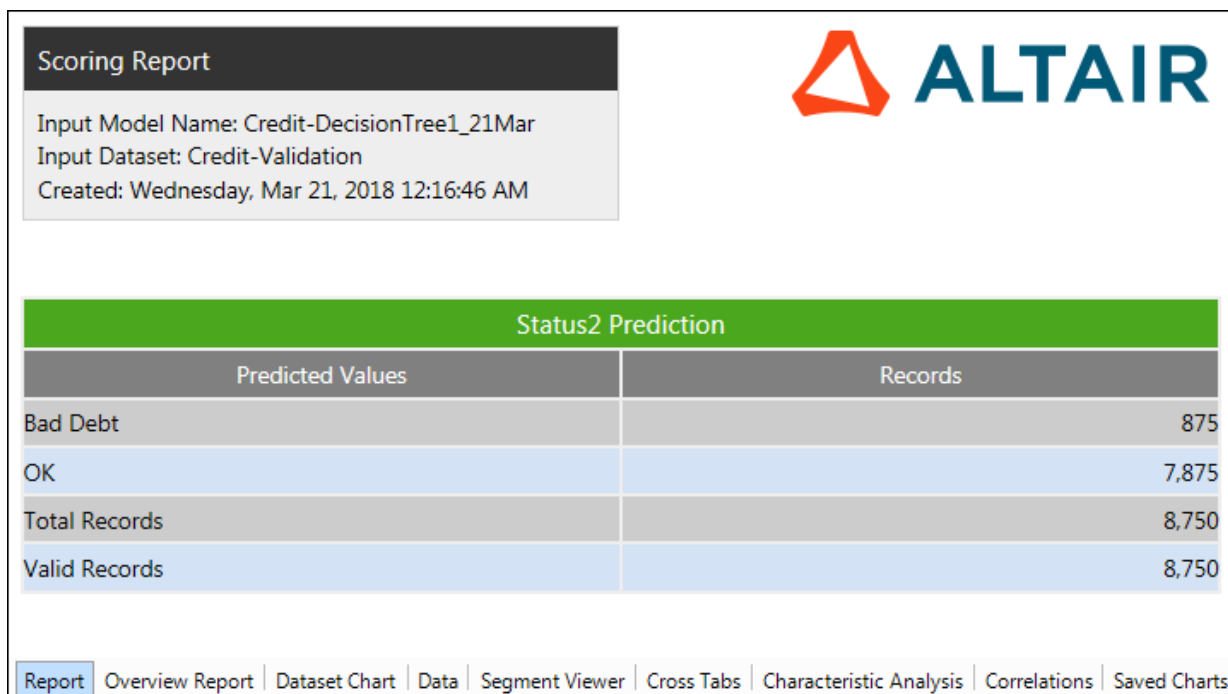
Click **Save** to save the settings, or click **Run** to save the settings and run the append process immediately, or click **Schedule** to schedule the task for execution at a later time.

The output dataframe node will be created in the workflow after running the scoring node.



The top section of the Score Report displays basic information such as the name of the input dataset, the name of the input model, and the date scoring was performed.

It also shows the total number of records provided as the scoring input ("**Total Number of Scored Records**") and the number of records for which a valid score was produced ("**Valid Records**"). The latter may be less than the former in the case where a required field in the scoring dataset has values that did not occur in the corresponding field of the training dataset (in this case, the produced score is a missing value).



## Code Generation

Decision Trees built on Spark dataframes in Knowledge Studio for Apache Spark can be automatically translated into deployment code in the Python language and deployed outside of Altair applications to produce model scores.

To generate Python code for a Decision Tree, open the tree and select the **Code Generation** command from the **Insert** menu. Then choose Python from the drop-down list. The code will be displayed in a separate frame.

Within the frame that displays the code, you can use the context menu as well as common keyboard shortcuts CTRL+F, CTRL+A, and CTRL+C to find, select and copy text. The font size can be increased using CTRL + mouse wheel zoom (scrolling the mouse wheel while holding down the CTRL key).

To save the code in a file, select the menu command **File | Save As...**

```

class TreeModel:

    DV = 'Status2'
    IVs = ['Initial Score', 'Year at Address', 'age', 'Residential Status']
    DVCategories = ['Bad Debt', 'OK']
    Weight = 1
    @staticmethod
    def predict(Initial_Score, Year_at_Address, age, Residential_Status):

        if Residential_Status is None \
        or Residential_Status == 'Live with Parents' \
        or Residential_Status == 'Other':

            if (Initial_Score >= 0 and Initial_Score < 255):

                if age is None \
                or (age >= 17 and age < 43):

                    return ['x.26/0.1/0.25/0', 17, 0.048943270300333706, 0.9510567296996663, 899]

                if (age >= 43 and age <= 98):

                    return ['x.26/0.1/0.25/1', 18, 0.012711864406779662, 0.9872881355932204, 472]

                return ['x.26/0.1/0', 5, 0.03646973012399708, 0.9635302698760029, 1371]

            if (Initial_Score >= 255 and Initial_Score < 284):

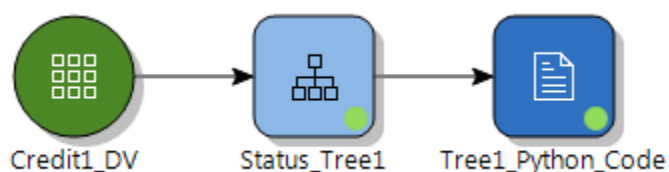
                if Year_at_Address is None \
                or (Year_at_Address >= 0 and Year_at_Address < 9):

                    return ['x.26/0.1/1.9/0', 19, 0.06564245810055866, 0.9343575418994413, 1432]

```

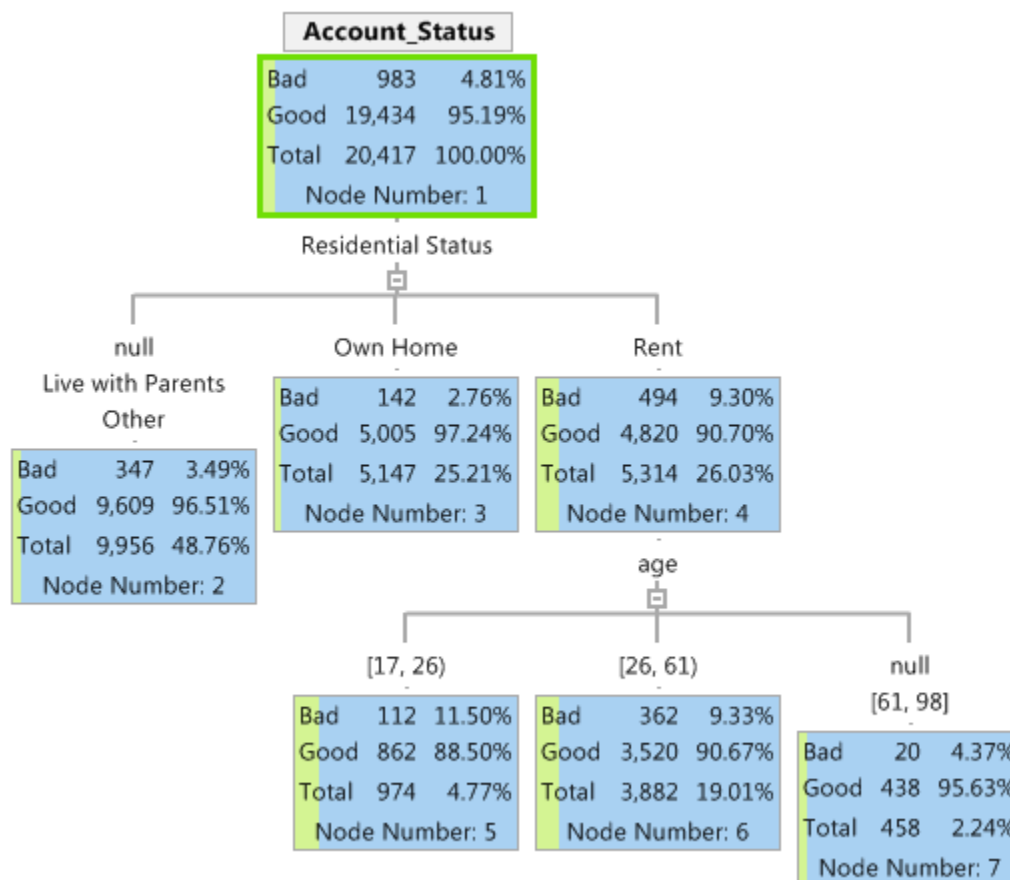
Code Generation

In addition, the Python code node will be automatically created on the Workflow canvas next to the decision tree node.



Double-click the code node to view the generated code or right-click on it and select **Open View**.

## Python Code Example and Deployment Instructions



The Python code for the first split in this example tree is given below.

- The first part of the code is the TreeModel class that traverses the tree branches and returns probabilities of target values of the DV along with the node numbers / IDs and node sizes.
- The second part of the code - starting with `colNodeNumber = []` - is the actual deployment (scoring) code that scans the records of the input dataframe over the columns representing the model variables and produces the output scores (model predictions) along with node numbers. In the example below, the scoring input dataframe is called "Credit\_DF". When using the code for deployment, this string is to be replaced with the actual input dataframe name.

```
class TreeModel:
    DV = 'Status2'
    IVs = ['age', 'Residential Status']
    DVCategories = ['Bad Debt', 'OK']
    Weight = 1
    @staticmethod
    def predict(age, Residential_Status):
        if Residential_Status is None \
        or Residential_Status == 'Live with Parents' \
        or Residential_Status == 'Other':
            return ['x.26/0', 2, 0.035039, 0.964960325, 9956]
```

```

        if Residential_Status == 'Own Home':
            return ['x.26/1', 3, 0.0290032, 0.9709967, 5147]
        if Residential_Status == 'Rent':
            if age is None \
            or (age >= 61 and age <= 98):
                return ['x.26/2.25/2', 7, 0.046757164, 0.953242835, 458]
            if (age >= 26 and age < 61):
                return ['x.26/2.25/1', 6, 0.0893233, 0.91067669, 3882]
            if (age >= 17 and age < 26):
                return ['x.26/2.25/0', 5, 0.10684474, 0.8931552, 974]
            return ['x.26/2', 4, 0.093906, 0.906093379, 5314]
    return ['x', 1, 0.0488223, 0.9511777, 20417]

colNodeNumber = []
colValues = []
for n in range(len(TreeModel.DVCategories)):
    colValues.append([])
colPrediction = []
for index, row in Credit_DF.iterrows():
    values = TreeModel.predict(*[row[column] for column in TreeModel.IVs])
    colNodeNumber.append(values[1])
    probability = 0
    prediction = None
    for n in range(len(TreeModel.DVCategories)):
        colValues[n].append(values[n + 2])
        if values[n + 2] > probability:
            probability = values[n + 2]
            prediction = TreeModel.DVCategories[n]
    colPrediction.append(prediction)
Credit_DF[TreeModel.DV + ' Node Number'] = colNodeNumber
for n in range(len(TreeModel.DVCategories)):
    Credit_DF[TreeModel.DV + ' ' + TreeModel.DVCategories[n] + ' Probability'] = colValues[n]
Credit_DF[TreeModel.DV + ' Prediction'] = colPrediction

```

**See also:**[Scoring on Spark](#)[Validation on Spark](#)[Spark Generic Code](#)

## Spark Generic Code Node

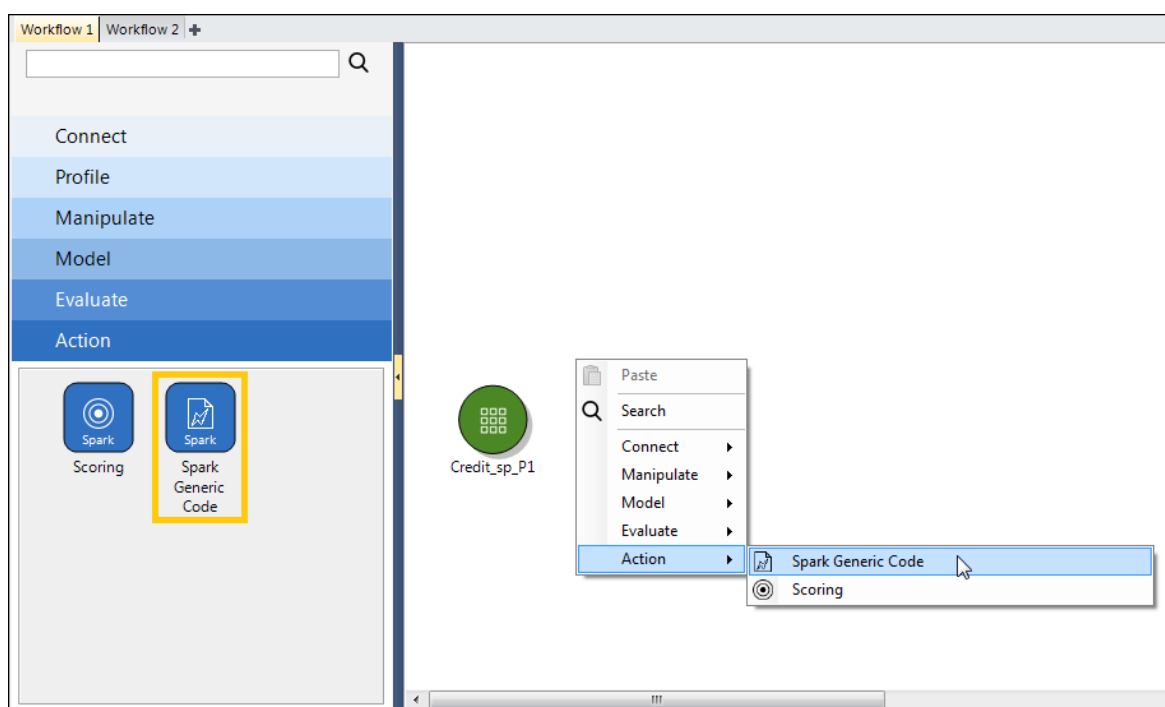
The **Spark Generic Code** node can be used to write and execute custom programs and macros in Python with PySpark on your Apache Spark server.

Page | 564

- Right-click anywhere on the Workflow canvas background. From the context menu, select **Action --> Spark Generic Code** as shown in the example below. A new **Spark Generic Code** node will be created.

OR:

- Drag the **Spark Generic Code** node from the **Action** node group onto the canvas. Then double-click the link node or right-click on it and select **Modify** from the context menu.



This opens the Spark Generic Code window, where you can write and execute any Python program with PySpark API.

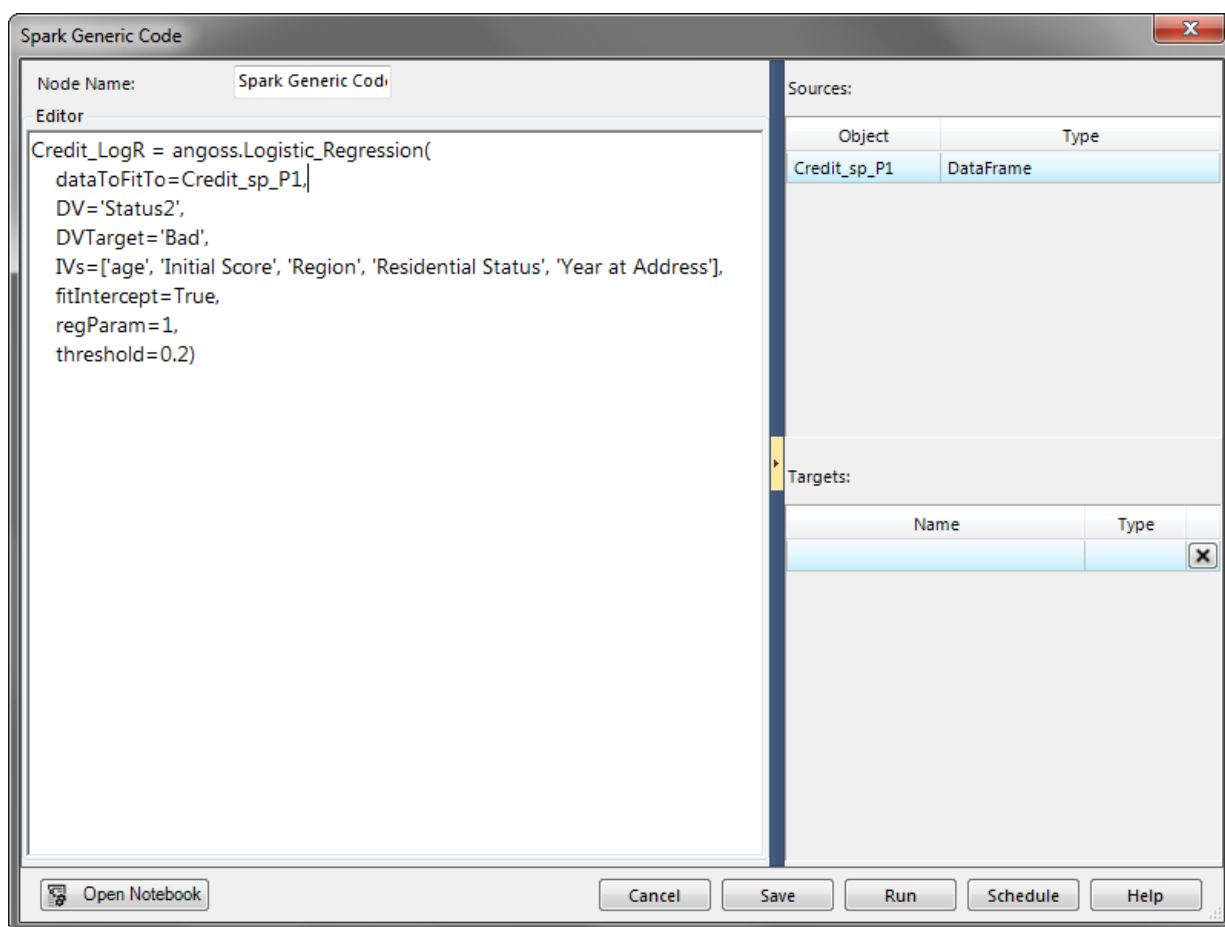
- The **Sources** section on the right-hand side lists the input dataframes currently connected to the Spark Generic Code node in the workflow.
- The **Targets** section on the right-hand side lists the output datasets to be produced by the Spark Generic Code node in the workflow. Specify the names and types of the output datasets here, if necessary.
- Clicking the button **Open Notebook** launches the web page with the *Jupyter Notebook* on the Spark Server you are currently connected to. It is a powerful development environment with rich functionality for additional code editing and validation. **Note:** The URL for Jupyter



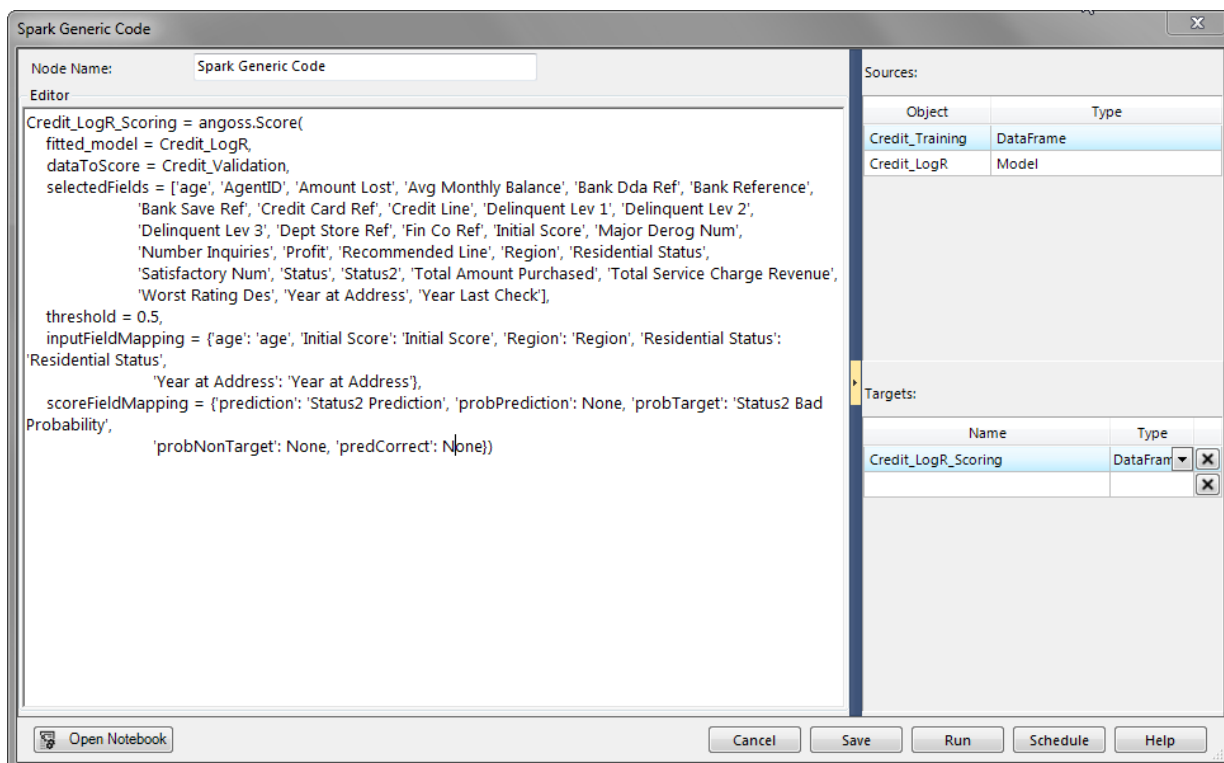
notebook server must be accessible from the local machine where KS Workstation is installed. If you are using the client/server mode, it must be accessible from both the client and server machines.

**Example 1:** Spark Generic Code node with Python code for Logistic Regression. The dataframe Credit\_sp\_P1 is connected to the code node and therefore is displayed in the Sources section in the right-hand side panel.

Page | 565

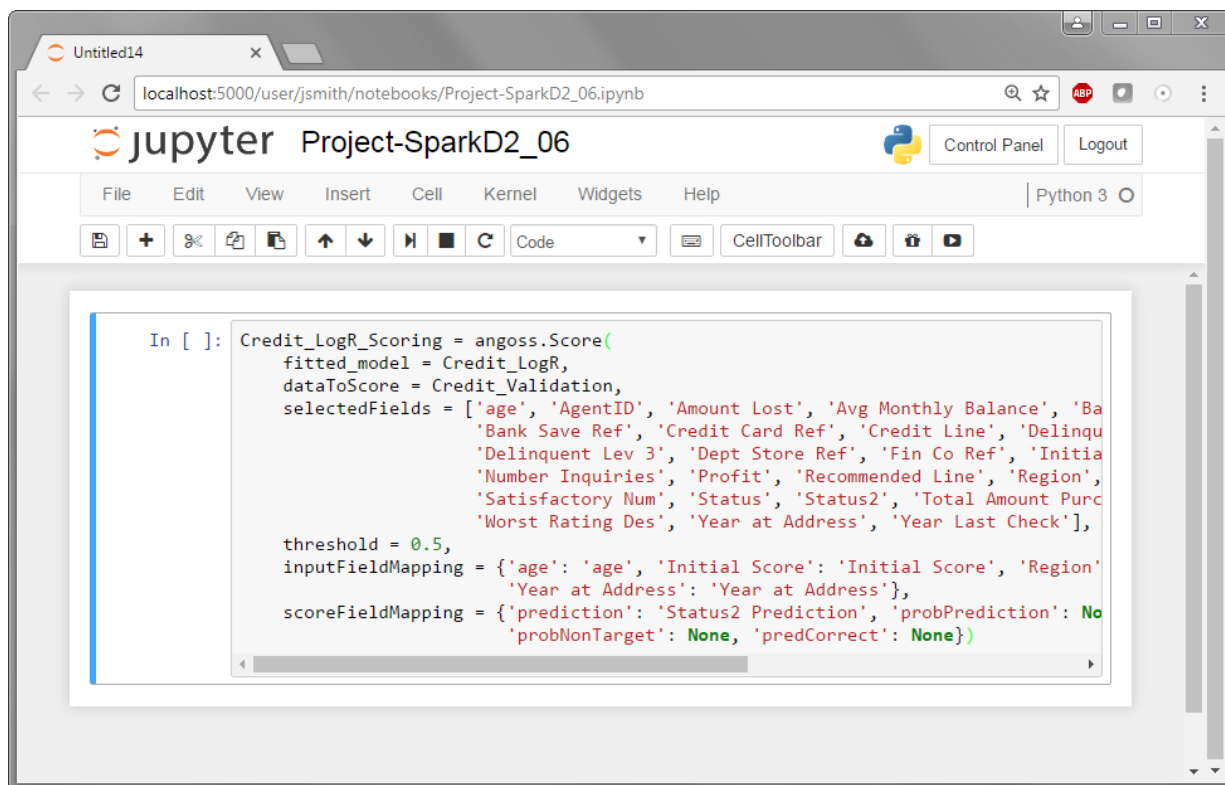


**Example 2:** Spark Generic Code node with Python code for Logistic Regression scoring. Both the dataframe Credit\_Training and the model Credit\_LogR are connected to the code node and therefore are displayed in the **Sources** section in the right-hand side panel. If you need to use the output dataframe in the workflow canvas, specify its name in the **Targets** section in the right-hand side panel.



**Example 3:** If you click **Open Notebook**, you are prompted to log in to the Jupyter Notebook of the Spark Server you are currently connected to. You can code in the notebook and then return to your Spark Generic Code node to continue.

**Note:** The URL for Jupyter notebook server must be accessible from the local machine where KS Workstation is installed. If you are using the client/server mode, it must be accessible from both the client and server machines.



---

# Selecting and Reusing Variables with Variable Lists

## Variable List

Page | 568

**Variable List** (often used in combination with the [Field Finder](#)) is a convenient tool that saves time and gives you more control in the variable selection process in many areas of the application, including data import, selecting independent variables for your models, etc.

Using variable lists, you can quickly select variables in datasets, dataframes, models, etc., and then use the saved variable list in other objects and processes. Variable Lists can be exported and imported, which also makes it possible to share them between several projects.

Variable Lists also provide a way to quickly repeat tasks that involve variable selection and can save you a lot of time and effort especially in the case of datasets with a lot of variables (hundreds or even thousands). For example, after importing and analyzing a dataset, you can save in a variable list a subset of variables selected for modelling. Then this list can be used in the future for quick selection of this subset when creating or modifying models, importing and exporting data, etc.

Variable Lists also allow you to do advanced variable selection based on various statistics available in the object you are working with. For example, based on the [Overview Report](#) of a dataset, you can quickly select variables depending on the number or percentage of missing values, their mean values, etc. If you are using a Decision Tree for variable reduction, you can rank all your variables by performing a [Find Split](#) and then quickly generate a variable list that would pick only variables with high information value.

- To insert a Variable List, select the menu command **Insert | Variable List**
- To save a copy of an existing Variable List in the same project, open it and select the menu command **File | Save Variable List As...**

The interface of the Variable List dialog depends on its source object (the object that is open when you are inserting the list).

A Variable List can be exported to an XML file or a plain text file. See [Exporting Variable Lists](#) for details.

### 1. Variable List based on the Overview Report of a Dataset or a Dataframe

An example of a variable list based on [Overview Report](#) is shown below. It includes univariate statistics shown and calculated in the Overview Report at the time the list is inserted.

**Note:** Not all of the univariate statistics are shown in the Overview Report by default. If you need more statistics, such as Sum, Variance, etc., add them to Overview Report using **Tools | Options** and calculate them using **Calculate All** *before inserting the variable list*.

Insert Variable List

General Information

List Name:

Credit\_VariableList1

Source:

Overview report of dataset Credit

Comments:

Credit data: Fields with no missing values.

Variables and Statistics

| #  | Variable           | Include <input type="checkbox"/>    | Label              | Cardinality | Unique Count | # of Missing Value | % of Missing Values |
|----|--------------------|-------------------------------------|--------------------|-------------|--------------|--------------------|---------------------|
| 6  | Number Inquiries   | <input checked="" type="checkbox"/> | Number Inquiries   | 29          | 5            | 16834              | 57.72%              |
| 7  | Worst Rating Des   | <input checked="" type="checkbox"/> | Worst Rating Des   | 10          | 0            | 16695              | 57.24%              |
| 8  | Credit Card Ref    | <input checked="" type="checkbox"/> | Credit Card Ref    | 4           | 1            | 15411              | 52.84%              |
| 9  | Dept Store Ref     | <input checked="" type="checkbox"/> | Dept Store Ref     | 4           | 1            | 14812              | 50.78%              |
| 10 | Bank Dda Ref       | <input checked="" type="checkbox"/> | Bank Dda Ref       | 4           | 1            | 14314              | 49.08%              |
| 11 | Year at Address    | <input checked="" type="checkbox"/> | Year at Address    | 70          | 10           | 12677              | 43.46%              |
| 12 | Residential Status | <input checked="" type="checkbox"/> | Residential Status | 5           | 0            | 11924              | 40.88%              |
| 13 | Bank Reference     | <input checked="" type="checkbox"/> | Bank Reference     | 5           | 2            | 11761              | 40.32%              |
| 14 | age                | <input checked="" type="checkbox"/> | age                | 83          | 2            | 622                | 2.13%               |
| 15 | Credit Line        | <input type="checkbox"/>            | Credit Line        | 48          | 18           | 0                  | 0.00%               |
| 16 | Initial Score      | <input type="checkbox"/>            | Initial Score      | 363         | 27           | 0                  | 0.00%               |
| 17 | Recommended Line   | <input type="checkbox"/>            | Recommended Line   | 24          | 9            | 0                  | 0.00%               |

Define Statistics...

Tree Variable Options

Select the source of variables:

Model Variable Options

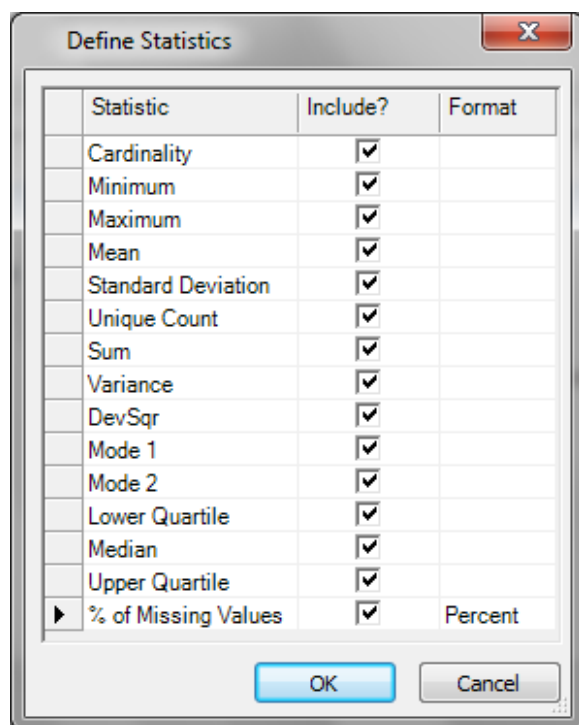
Select the source of variables:

OK

Cancel

Help

The **Define Statistics** button allows you to show or hide columns in the Variable List.



## 2. Variable List based on the Correlations View

An example of a variable list based on the [Correlations View](#) of a dataset is shown below. In addition the list of variables, it also shows the correlation matrix and the highlighting that was applied at the time of inserting the list.

Insert Variable List

General Information

List Name:

telco\_VariableList1

Source:

Correlation diagnostics: Dataset = telco, Coefficient = Pearson, View = Matrix

Comments:

Variables

| Include                             |                 | customer_type | profession | income   | married  | members0-5 | members5-18 | members18+ | house_owner | video    | satellite | game_console |
|-------------------------------------|-----------------|---------------|------------|----------|----------|------------|-------------|------------|-------------|----------|-----------|--------------|
| <input checked="" type="checkbox"/> | customer_type   | 1             | 0.26779    | 0.22362  | 0.08956  | 0.04565    | 0.05918     | 0.19516    | 0.11156     | 0.00799  | 0.09086   | 0.02093      |
| <input type="checkbox"/>            | profession      | 0.26779       | 1          | 0.59851  | 0.28513  | -0.14853   | 0.1495      | -0.03692   | 0.30581     | -0.00371 | 0.11946   | 0.01456      |
| <input type="checkbox"/>            | income          | 0.22362       | 0.59851    | 1        | 0.43457  | -0.209     | 0.19358     | 0.26271    | 0.54041     | -0.00652 | 0.07216   | 0.05182      |
| <input checked="" type="checkbox"/> | married         | 0.08956       | 0.28513    | 0.43457  | 1        | -0.02686   | 0.19044     | 0.41811    | 0.3268      | -0.0152  | 0.05049   | 0.14817      |
| <input checked="" type="checkbox"/> | members0-5      | 0.04565       | -0.14853   | -0.209   | -0.02686 | 1          | 0.00211     | -0.08012   | -0.08792    | 0.0005   | 0.1151    | 0.02232      |
| <input checked="" type="checkbox"/> | members5-18     | 0.05918       | 0.1495     | 0.19358  | 0.19044  | 0.00211    | 1           | 0.06145    | 0.16172     | 0.00606  | 0.15657   | 0.29261      |
| <input checked="" type="checkbox"/> | members18+      | 0.19516       | -0.03692   | 0.26271  | 0.41811  | -0.08012   | 0.06145     | 1          | 0.08955     | -0.00593 | 0.00338   | 0.10528      |
| <input type="checkbox"/>            | house_owner     | 0.11156       | 0.30581    | 0.54041  | 0.3268   | -0.08792   | 0.16172     | 0.08955    | 1           | 0.00292  | 0.07499   | 0.07466      |
| <input checked="" type="checkbox"/> | video           | 0.00799       | -0.00371   | -0.00652 | -0.0152  | 0.0005     | 0.00606     | -0.00593   | 0.00292     | 1        | 0.00442   | 0.00292      |
| <input checked="" type="checkbox"/> | satellite       | 0.09086       | 0.11946    | 0.07216  | 0.05049  | 0.1151     | 0.15657     | 0.00338    | 0.07499     | 0.00442  | 1         | 0.15485      |
| <input checked="" type="checkbox"/> | game_console    | 0.02093       | 0.01456    | 0.05182  | 0.14817  | 0.02232    | 0.29261     | 0.10528    | 0.07466     | 0.00292  | 0.15485   | 1            |
| <input checked="" type="checkbox"/> | pc              | 0.04452       | 0.14743    | 0.1919   | 0.04518  | 0.01742    | 0.07179     | 0.11764    | 0.01847     | -0.00255 | -0.09531  | 0.04223      |
| <input checked="" type="checkbox"/> | friends&family  | 0.07403       | 0.34162    | 0.23738  | -0.0142  | -0.13429   | -0.10445    | 0.07606    | 0.10803     | -0.00503 | -0.04366  | -0.11223     |
| <input checked="" type="checkbox"/> | call_waiting    | -0.22748      | 0.28829    | 0.25446  | 0.09265  | -0.19465   | 0.1676      | 0.04311    | 0.13806     | 0.00637  | 0.07635   | -0.01528     |
| <input checked="" type="checkbox"/> | call_forward    | -0.29966      | 0.17094    | 0.22911  | 0.16521  | -0.14561   | -0.00574    | 0.16462    | 0.12797     | -0.00716 | 0.09577   | -0.02816     |
| <input checked="" type="checkbox"/> | cli             | -0.29816      | 0.13224    | -0.03877 | 0.08613  | -0.07071   | 0.05885     | -0.12214   | -0.0356     | -8E-05   | -0.02895  | 0.06438      |
| <input checked="" type="checkbox"/> | call_hour       | 0.09266       | 0.19522    | 0.15362  | 0.03954  | -0.10548   | -0.00061    | 0.11993    | 0.04516     | 0.01291  | -0.02569  | -0.02616     |
| <input checked="" type="checkbox"/> | call_duration   | -0.01717      | 0.01562    | 0.04881  | -0.02597 | -0.02126   | -0.02238    | -0.02784   | 0.00975     | -0.00185 | 0.00398   | -0.00616     |
| <input checked="" type="checkbox"/> | destination_std | -0.0612       | 0.01983    | 0.00253  | -0.04798 | -0.06181   | -0.0329     | -0.00019   | -0.01733    | -0.01793 | -0.05824  | -0.04616     |
| <input checked="" type="checkbox"/> | call_cost       | -0.1444       | -0.05366   | -0.03383 | -0.06542 | -0.03558   | -0.04484    | -0.05047   | -0.03389    | -0.01421 | -0.04282  | -0.04046     |

OK

Cancel

Help

Page | 571

### 3. Variable List based on a Decision Tree or a Strategy Tree

An example of a variable list based on a [Decision Tree](#) is shown below. The IsDV column (if shown) indicates whether a variable is the dependent variable of the tree. You can hide this column by unchecking the box "Show the IsDV column".

Altair Engineering, Inc. Copyright © 2022. All rights reserved.

**Insert Variable List**

**General Information**

List Name:

Source:

Comments:

---

**Variables and Statistics**

| #  | Variable           | Include <input checked="" type="checkbox"/> | infoValue | inputEntropy | outputEntropy | ratioEntropy |
|----|--------------------|---------------------------------------------|-----------|--------------|---------------|--------------|
| 1  | Residential Status | <input checked="" type="checkbox"/>         | 0.28566   | 0.28137      | 0.27146       | 0.03523      |
| 2  | Initial Score      | <input checked="" type="checkbox"/>         | 0.31026   | 0.28137      | 0.27193       | 0.03356      |
| 3  | Region             | <input checked="" type="checkbox"/>         | 0.21660   | 0.28137      | 0.27303       | 0.02966      |
| 4  | Number Inquiries   | <input checked="" type="checkbox"/>         | 0.21393   | 0.28137      | 0.27348       | 0.02804      |
| 5  | Credit Line        | <input checked="" type="checkbox"/>         | 0.26323   | 0.28137      | 0.27431       | 0.02509      |
| 6  | age                | <input checked="" type="checkbox"/>         | 0.18833   | 0.28137      | 0.27564       | 0.02038      |
| 7  | Worst Rating Des   | <input checked="" type="checkbox"/>         | 0.12857   | 0.28137      | 0.27670       | 0.01662      |
| 8  | Year Last Check    | <input checked="" type="checkbox"/>         | 0.12894   | 0.28137      | 0.27695       | 0.01573      |
| 9  | Year at Address    | <input type="checkbox"/>                    | 0.11238   | 0.28137      | 0.27765       | 0.01322      |
| 10 | Satisfactory Num   | <input type="checkbox"/>                    | 0.10298   | 0.28137      | 0.27773       | 0.01296      |
| 11 | Credit Card Ref    | <input type="checkbox"/>                    | 0.09706   | 0.28137      | 0.27804       | 0.01186      |
| 12 | Recommended Line   | <input type="checkbox"/>                    | 0.08857   | 0.28137      | 0.27846       | 0.01035      |
| 13 | Bank Reference     | <input type="checkbox"/>                    | 0.06606   | 0.28137      | 0.27922       | 0.00766      |
| 14 | Dent Store Ref     | <input type="checkbox"/>                    | 0.03493   | 0.28137      | 0.28019       | 0.00421      |

Define Statistics... ☐ Show the "IsDV" column

---

**Tree Variable Options**

Select the source of variables:

---

**Model Variable Options**

Select the source of variables:

OK Cancel Help

## Tree Variable Options

**Split from**

If this option is selected, the Variable List includes the DV and the variables of all splits **at the currently**



|                                       |                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>current split</b>                  | <b>selected node.</b> It is the same as the list of variables in the <a href="#">Split Report</a> tab or the list displayed by the <a href="#">Go To Split</a> command for the currently selected node. It will also show the <a href="#">split parameters</a> for each split variable (note that split parameters depend on the <a href="#">measure</a> ). |
| <b>Variables used in visible tree</b> | If this option is selected, the Variable List includes the DV and the split variables of <b>all visible splits in the tree</b> . It does <b>not</b> include the variables of the splits that are not shown (but accessible via the <a href="#">Go To Split</a> command). Split parameters are not shown in this case.                                       |
| <b>Split list from all splits</b>     | Variable List includes all variables from all splits in the tree. It does not show split parameters in this case.                                                                                                                                                                                                                                           |

#### 4. Variable List based on a Predictive Model or a Cluster Analysis Model

This is an example of a variable list based on a Predictive Model or a Cluster Analysis Model.

In this case, the source is a Logistic Regression model with stepwise variable selection. In the Model Variable Options drop-down list, you can select a particular sequence as a source of variables. By default, all input variables of the model are selected.

**Insert Variable List**

**General Information**

List Name:

Source:

Comments:

---

**Variables and Statistics**

| #  | Variable           | Include <input checked="" type="checkbox"/> | IsDV | Expression                                                |
|----|--------------------|---------------------------------------------|------|-----------------------------------------------------------|
| 1  | Recommended Line   | <input checked="" type="checkbox"/>         | No   |                                                           |
| 2  | Number Inquiries   | <input checked="" type="checkbox"/>         | No   |                                                           |
| 3  | Year Last Check    | <input checked="" type="checkbox"/>         | No   |                                                           |
| 4  | Satisfactory Num   | <input checked="" type="checkbox"/>         | No   |                                                           |
| 5  | Year at Address    | <input checked="" type="checkbox"/>         | No   |                                                           |
| 6  | Region             | <input checked="" type="checkbox"/>         | No   |                                                           |
| 7  | age                | <input checked="" type="checkbox"/>         | No   |                                                           |
| 8  | Residential Status | <input checked="" type="checkbox"/>         | No   |                                                           |
| 9  | Status2            | <input checked="" type="checkbox"/>         | Yes  | CASE WHEN ([Status] = 'Bad Debt') THEN 'Bad Debt' WHEN    |
| 10 | ScoreBand          | <input checked="" type="checkbox"/>         | No   | CASE WHEN ([Initial Score] >= 0 and [Initial Score] < 346 |

Define Statistics... ☒ Show the "IsDV" column

---

**Tree Variable Options**

Select the source of variables:

---

**Model Variable Options**

Select the source of variables:  The currently selected sequence is 1

OK Cancel Help

## 5. Variable List based on a Custom Variable Sequence in Predictive Model Wizard

In the Variable Selection Methods of the Predictive Model Wizard, there is an option to define a variable sequence of your choice. If you use this option and define a custom sequence, you can save it in a Variable List by clicking the Save Variable List button in the wizard page.

**Insert Variable List**

**General Information**

List Name:

Source:

Comments:

---

**Variables and Statistics**

| # | Variable             | Include <input checked="" type="checkbox"/> | IsDV | Information Value | Significance | Expression          |
|---|----------------------|---------------------------------------------|------|-------------------|--------------|---------------------|
| 1 | Recommended Line     | <input checked="" type="checkbox"/>         | No   |                   | 0.67667      |                     |
| 2 | Number Inquiries     | <input checked="" type="checkbox"/>         | No   |                   | 0            |                     |
| 3 | Year Last Check      | <input checked="" type="checkbox"/>         | No   |                   | 0.00079      |                     |
| 4 | Satisfactory Num     | <input checked="" type="checkbox"/>         | No   | 0.10242           | 0.0047       |                     |
| 5 | Year at Address      | <input checked="" type="checkbox"/>         | No   | 0.11357           | 0            |                     |
| 6 | age                  | <input checked="" type="checkbox"/>         | No   | 0.19174           | 0            |                     |
| 7 | Residential Status   | <input checked="" type="checkbox"/>         | No   |                   |              |                     |
|   | Own Home             |                                             |      |                   | 0            |                     |
|   | Other                |                                             |      |                   | 0.00102      |                     |
|   | Live with Parents    |                                             |      |                   | 0.23608      |                     |
|   | Rent                 |                                             |      |                   | 0            |                     |
| 8 | ScoreBand            | <input checked="" type="checkbox"/>         | No   |                   |              | CASE WHEN ([Initial |
|   | 003:[433.50to867.00] |                                             |      |                   | 0            |                     |
|   | 004:[346.80to433.50] |                                             |      |                   | 0.13128      |                     |

Define Statistics... ☒ Show the "IsDV" column

---

**Tree Variable Options**

Select the source of variables:

---

**Model Variable Options**

Select the source of variables:

OK Cancel Help

You can save a copy of an existing variable list using the command **File | Save Variable List As....**. The list can be modified before saving, if necessary.

See also:

[Field Finder](#)

---

[Exporting Variable Lists](#)

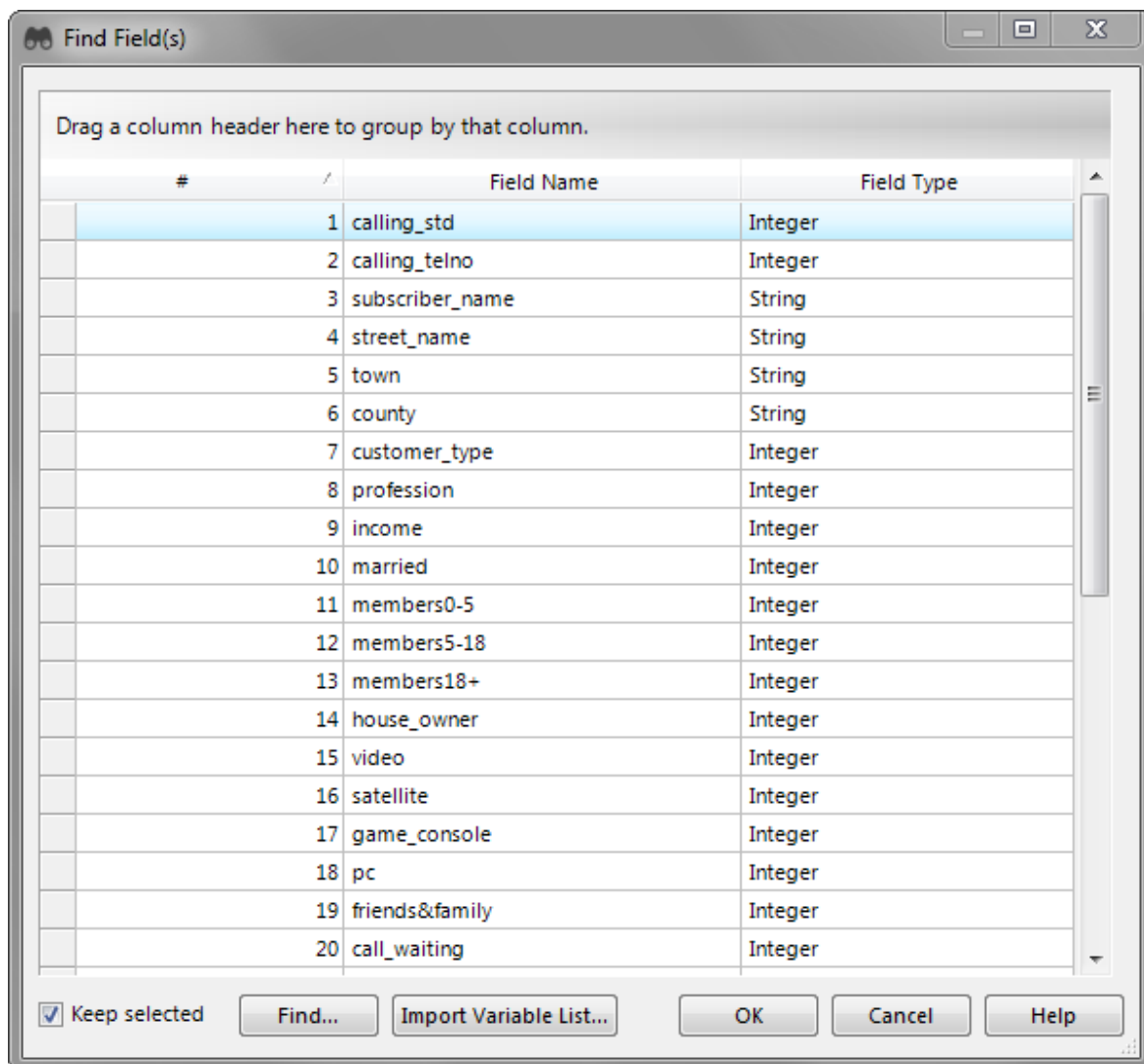
## Field Finder

**Field Finder** provides an easy way to find and select the fields of interest in any area of the application, for example, when importing a dataset, selecting variables for correlation analysis, predictive model, segment viewer charts, etc. It is especially helpful when there are a lot of variables in your dataset (hundreds or even thousands).

Page | 576

Field Finder is launched by the binoculars button , which can be found in any view or dialog that involves selecting or highlighting a field (variable).

The Field Finder for Spark dataframes shows the data type for each field.



### Field Finder features:

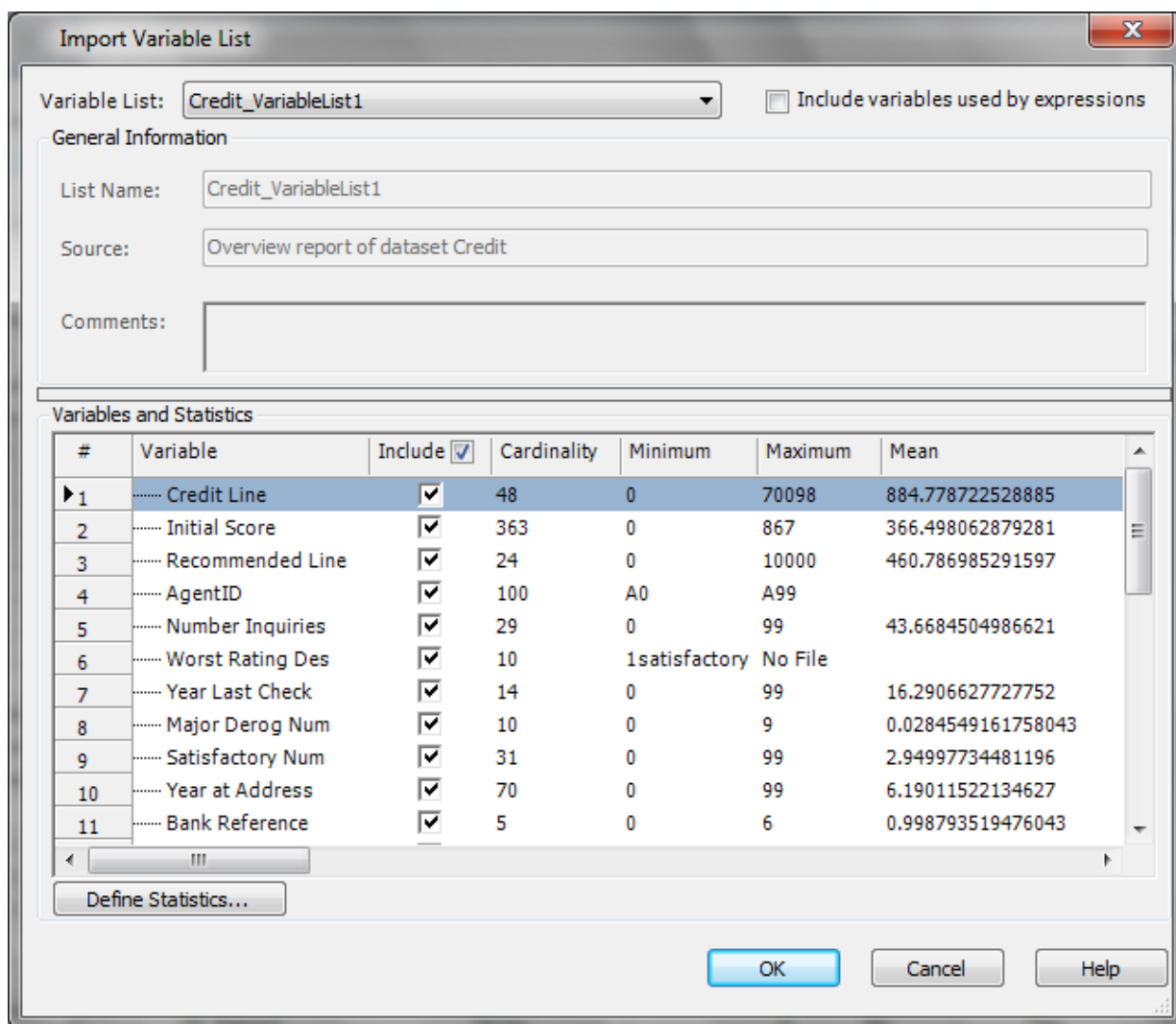
- The field list can be sorted by the field index, name, or type.
- Multiple fields can be selected by holding the SHIFT or CTRL buttons while clicking on the rows in the grid.
- If the "**Keep Selected**" box is checked, then a single mouse-click adds fields to your selection without cancelling the existing selection. This option may be especially useful if the number of fields is very large.
- Click the **Find** button at the bottom to search for a field (or fields) whose name contains a particular substring.
- Click the **Import Variable List** button at the bottom to apply the variable selection from a previously saved [Variable List](#).

## Importing Variable Lists

Field Finder can use [Variable Lists](#) for advanced field selection to apply the selection of fields from any Variable List saved previously. (The Variable List option is not applicable in the case of Spark dataframes). For example, suppose that you saved the list of all variables of your dataset that have no missing values. If at some point you need the same selection of variables for your correlation analysis or principal component analysis, you can use the Field Finder in the respective wizards and apply the selection by importing your variable list.

Page | 578

Using the **Import Variable List** button, you can import any variable list inserted previously and apply its selection to the list in the Field Finder. You can import a variable list from any project or even an [external file](#).



The dialog box titled "Import Variable List" contains the following fields and sections:

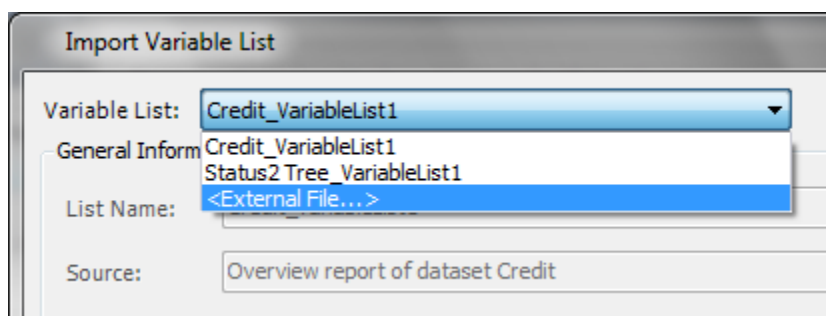
- Variable List:** A dropdown menu showing "Credit\_VariableList1".
- ☐ Include variables used by expressions
- General Information** section:
  - List Name:** Credit\_VariableList1
  - Source:** Overview report of dataset Credit
  - Comments:** (Empty text area)
- Variables and Statistics** section:
 

| #  | Variable         | Include <input checked="" type="checkbox"/> | Cardinality | Minimum       | Maximum | Mean               |
|----|------------------|---------------------------------------------|-------------|---------------|---------|--------------------|
| 1  | Credit Line      | <input checked="" type="checkbox"/>         | 48          | 0             | 70098   | 884.778722528885   |
| 2  | Initial Score    | <input checked="" type="checkbox"/>         | 363         | 0             | 867     | 366.498062879281   |
| 3  | Recommended Line | <input checked="" type="checkbox"/>         | 24          | 0             | 10000   | 460.786985291597   |
| 4  | AgentID          | <input checked="" type="checkbox"/>         | 100         | A0            | A99     |                    |
| 5  | Number Inquiries | <input checked="" type="checkbox"/>         | 29          | 0             | 99      | 43.6684504986621   |
| 6  | Worst Rating Des | <input checked="" type="checkbox"/>         | 10          | 1satisfactory | No File |                    |
| 7  | Year Last Check  | <input checked="" type="checkbox"/>         | 14          | 0             | 99      | 16.2906627727752   |
| 8  | Major Derog Num  | <input checked="" type="checkbox"/>         | 10          | 0             | 9       | 0.0284549161758043 |
| 9  | Satisfactory Num | <input checked="" type="checkbox"/>         | 31          | 0             | 99      | 2.94997734481196   |
| 10 | Year at Address  | <input checked="" type="checkbox"/>         | 70          | 0             | 99      | 6.19011522134627   |
| 11 | Bank Reference   | <input checked="" type="checkbox"/>         | 5           | 0             | 6       | 0.998793519476043  |
- Buttons:** Define Statistics..., OK, Cancel, Help.

If the variable list you are importing contains calculated columns, then you can choose to include all variables occurring in the expressions for these calculated columns. For example, in the figure above, the *ScoreBand* variable is a calculated field that depends on the *Initial Score* variable. If the checkbox "Include variables used by expressions" is checked, then *Initial Score* will be imported together with *ScoreBand*.

Page | 579

If you wish to import a variable list from another project or elsewhere, use the External File option in the drop-down list:



Then browse for the desired variable list file. (Variable lists are represented by files with \*.kvl extension in project directories).

If you are importing a variable list from an XML file (see [Exporting Variable Lists](#)), change the File Type from \*.kvl to \*.xml and browse for the desired file.

See also:

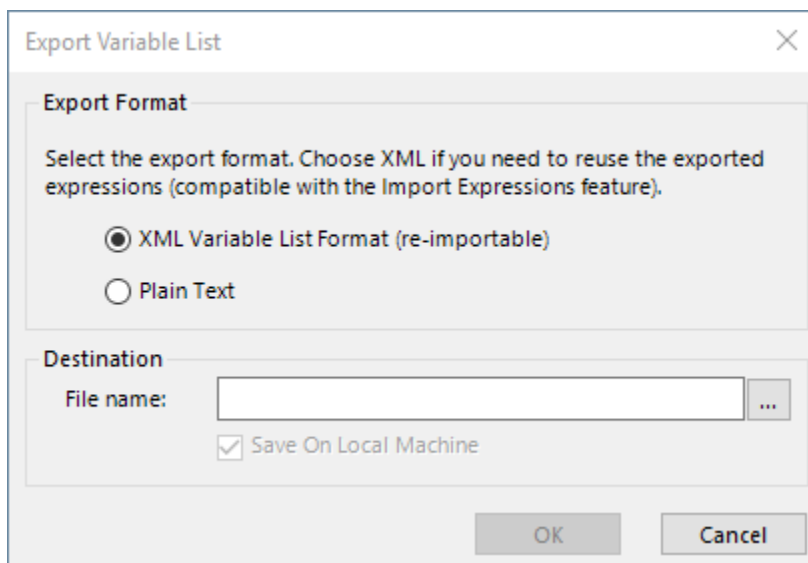
[Variable List](#)

[Exporting Variable Lists](#)

## Exporting Variable Lists

A [Variable List](#) can be exported to an XML file for future use in [Field Finder](#). Although in Field Finder you can import variable lists *directly* from the current project or other projects, you may want to save a variable list in an external file to store it in case your project may get deleted or even for the purpose of sharing it with other users.

To export a variable list, open it and select the command **File | Export ...** from the main menu.



The default format for the output file is **XML**. Variable List files can have extensions KVL, KVLC, and XML files (\*.kvl, \*.kvlc, \*.xml).

The resulting variable list file can be imported into Field Finder whenever necessary in order to apply the same field selection. See the [Field Finder](#) topic for details on importing variable lists.

Choosing the **Plain Text** format will produce a text file with the variable list in a tabular form. A variable list file in the plain text format cannot be reimported.

If you are working in the client/server mode, you can save the exported file either on the local machine or on the server.

See also:

[Variable List](#)

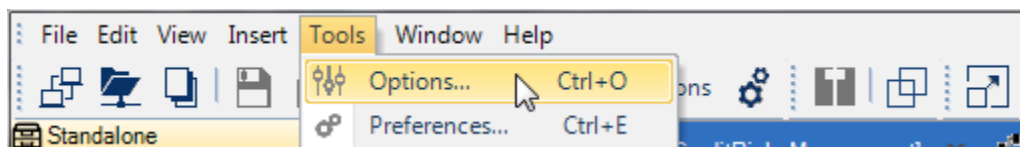
[Field Finder](#)



## Options and Preferences

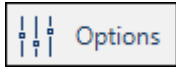
Various attributes, parameters and display settings for all types of objects and views can be changed using the **Tools | Options** and **Tools | Preferences** menu commands. The **Options** command is only available when applicable to the object that is currently active.

Page | 581



### The Scope of Options and Preferences

- Changes made through **Tools | Options** are applied locally - only to the **object that is currently active**. They take effect immediately. Aside from the **Tools** menu, Options can be

accessed by clicking the Options toolbar button  or, in some cases, using the right-click context menu (for instance, in the Decision Tree view).

- The settings and properties that can be changed through **Tools | Preferences** are global preferences determining the properties of objects to be created (models, etc.) or styles and display settings. If you change any preferences for a certain type of objects (for example, decision trees), this will affect all objects of this type created in the future in any project. The changes in Preferences do not affect existing models. Aside from the **Tools** menu,

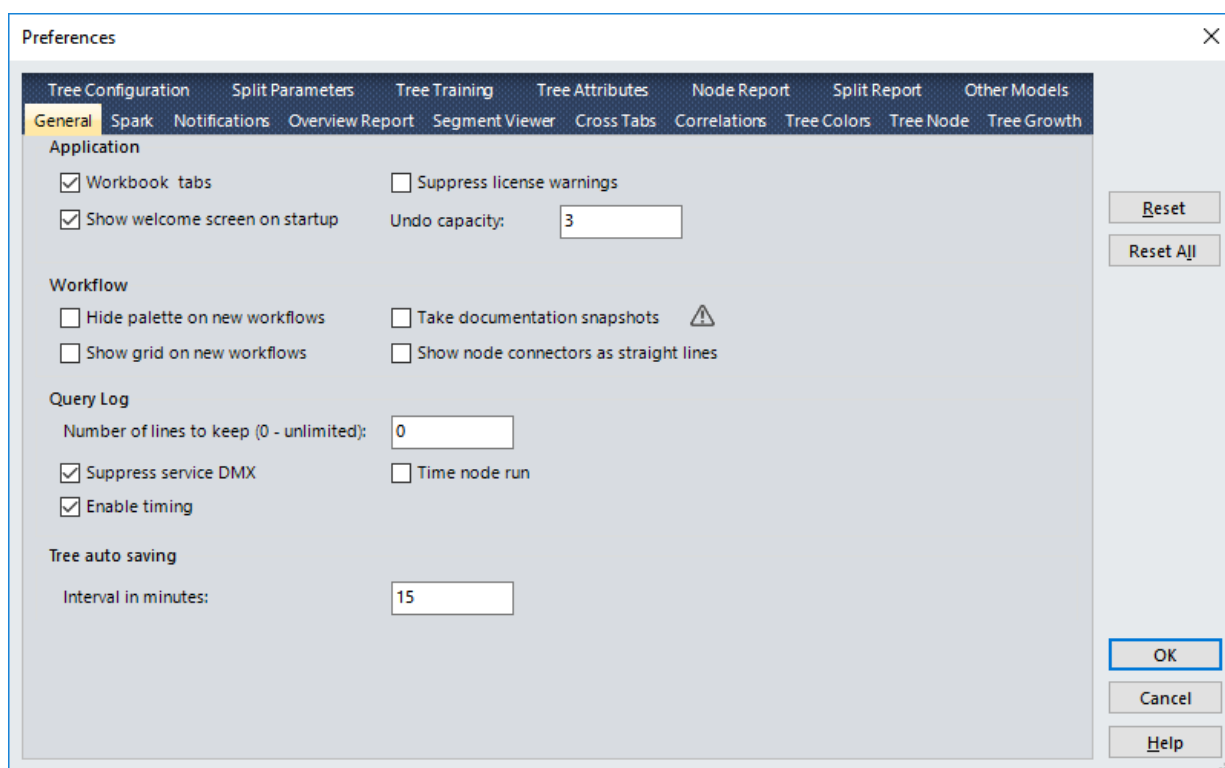
Preferences can be accessed by clicking the toolbar button .

For the description of each category of options and preferences, click on the appropriate link:

- [Covariance/Correlation](#)
- [Cross Tabs](#)
- [General](#)
- [Node Report](#)
- [Overview Report](#)
- [Segment Viewer](#)
- [Split Report](#)
- [Spark](#)
- [Tree Colors](#)
- [Tree Configuration](#)
- [Tree Growth](#)
- [Tree Node Settings](#)
- [Tree Split Parameters](#)
- [Tree Training](#)
- [Tree Attributes](#)

- [Display Format in Reports](#)
- Text Analysis
- [Notifications](#)

## General application preferences



### **Workbook tabs**

Shows a separate tab for every object that is currently open. This makes it easier to navigate between the currently open objects.

### **Show welcome screen on startup**

Shows the Welcome screen every time the application is started.

### **Suppress license warnings**

License warnings may appear for various reasons, for example, if the license is about to expire. Selecting this option will suppress all license warnings.

### **Undo capacity**

The maximum number of actions that can be rolled back using the Undo command in the toolbar. This applies to Decision Trees and Strategy Trees.

## Workflow preferences

### ***Hide palette on new workflows***

If this option is selected, the Palette panel is automatically docked when a new workflow is created.

### ***Show grid on new workflows***

Shows the grid by default for any new workflow created in the project.

### ***Take documentation snapshots***

Snapshots are taken automatically when opening any view of any object. The snapshots are used by the Self-Documentation and Model Documentation nodes. **Note:** Since this feature can slow down the performance, it is not recommended to turn this option on unless you plan to document your workflow using the Documentation nodes.

### ***Show node connectors as straight lines***

If this option is selected, the node connectors on the workflow are shown as straight lines. This may be useful especially in busy workflows with many nodes.

Page | 583

## Query Log preferences

Query Log provides the history of your data mining activities in the form of SQL-like commands (DMX commands). It is updated in real time as you perform your actions in the project.

For more information about the Query Log, see the topic [Query Log](#) in the Logging Capabilities section.

### ***Number of lines to keep***

Specifies the number of the most recent DMX commands to preserve in the Query Log (the default is 0, which stands for "unlimited", i.e., all commands processed during the current session will be kept in the log).

### ***Suppress service DMX***

Suppresses the auxiliary DMX commands that are not directly related to the data mining tasks being performed (for example, commands that update the project hierarchy, or commands that create temporary objects).

### ***Enable timing***

Displays the starting time and duration of each DMX command.

### ***Time node run***

Measure the time it took for each node to run and include these times in the log.

Example:

```
# =====> Node run started at 2018-12-03  2:24
AM
# 'Decision Tree: Census_DecisionTree' started run
at 2018-12-03  2:24:21 AM
# 'Decision Tree: Census_DecisionTree' finished
run at 2018-12-03  2:24:21 AM in 0.62 sec (0.616 sec)
# =====> Node run finished in 1.12 sec (1.116 sec)
```

Page | 584

## Tree Auto-Saving preferences

Using this option, you can set the preferred time interval for auto-saving trees. The default interval is 15 minutes.

## Cross Tabs

The settings for the Cross Tabs view of the Dataset are specified in this tab.



Settings

Number of bins

☒ Equal Width ☐ Equal Height

Data Presentation Settings

Display data as ☒ Chart ☐ Report

- |                              |                                                                                                                        |
|------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <b><i>Number of bins</i></b> | Maximum number of bins in which to distribute the data. The default value is 10.                                       |
| <b><i>Equal Height</i></b>   | When this is checked, the data is distributed so that each bin contains approximately the same number of observations. |
| <b><i>Equal Width</i></b>    | When this is checked, the width of the intervals on the x-axis will be the same for all bins (default).                |

**See also:** [Cross Tabs Chart](#)

## Correlations

Correlation and covariance coefficients are displayed in the Correlations view in the dataset. The general settings for the Correlations view are specified in this tab.

Initial View

Correlations Coefficients:

☐ Pearson
☒ Covariance Matrix
☐ Spearman's rho

Discretize integer variables when cardinality less than:

Maximum number of indicator variables:

Maximum correlation between IV's:

Minimum correlation between DV and IV:

### Correlation

#### Covariance

**Discretize integer variables when cardinality less than**

**Maximum fan-out (Maximum # of indicator variables)**

Displays correlation coefficients (default).

Displays covariance coefficients.

Determines which integer-valued variables are discretized when calculating correlation coefficients or covariance. It also determines which variables are highlighted by the Highlight Continuous button. Integer variables are considered continuous if their cardinality is higher than this threshold. Otherwise, integer variables are considered discrete and are represented as a set of indicator variables corresponding to the unique values of the variable. The default cardinality threshold is 30. All non-integer numeric variables are always treated as continuous regardless of their cardinality.

The maximum number of categories to display for discrete variables. If this parameter is set to N, for all discrete variables whose cardinality is greater than N, the first N-1 unique values will be displayed as separate categories and the rest will be mapped to one category denoted by ###. The default value is 10.

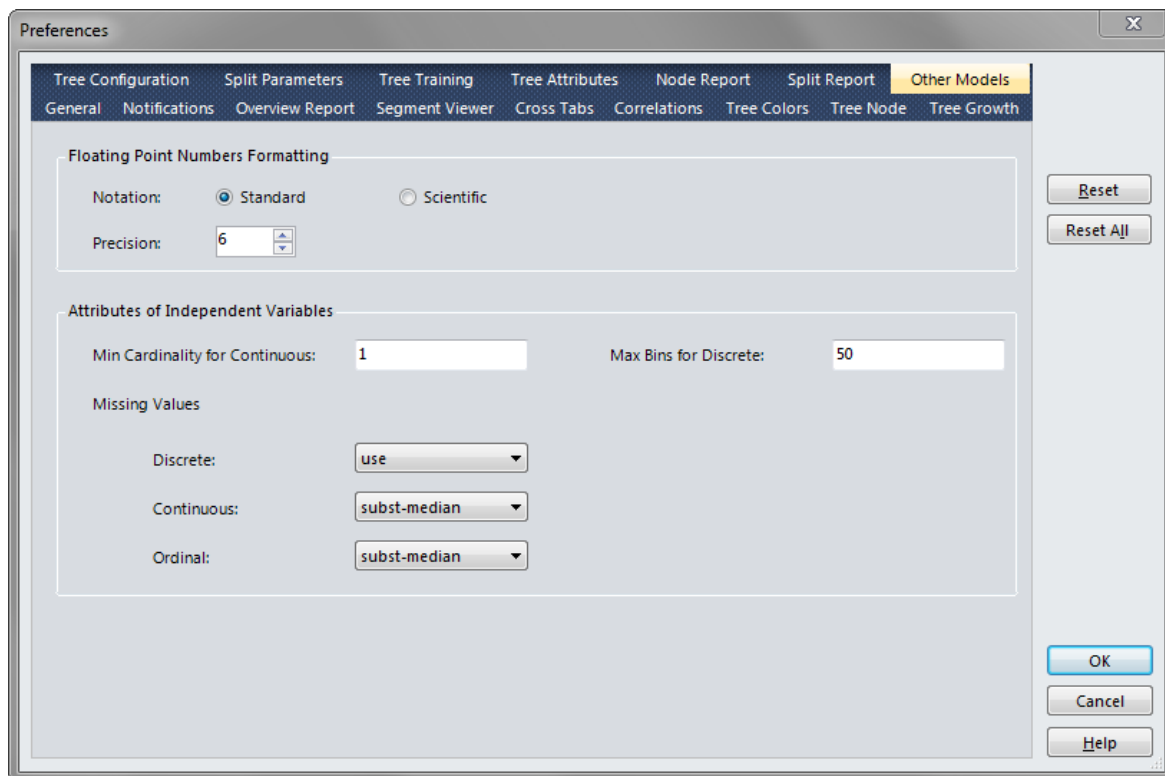
See also: [Correlations Tab](#).

## Model Attributes

The settings in the **Other Models** tab of the Preferences dialog determine the following properties:

- Attributes of independent variables of all models other than decision trees and strategy trees. (The tree attribute defaults are defined in the [Tree Attributes](#) tab of Preferences)

Page | 586



### ***Min Cardinality for Continuous***

Numeric independent variables with cardinality higher than this value are considered to be continuous. The default threshold is **1** (all numeric variables are continuous by default). If you set this threshold to **x**, then all numeric variables with cardinality  $\geq x$  will have Usage = continuous, and all numeric variables with cardinality less than **x** will have Usage = ordinal (i.e., ordered discrete).

### ***Max Bins for Discrete***

The maximum number of categories for discrete variables. The default is 50. If the specified maximum number of categories (bins) **N** is larger or equal to the cardinality of the variable, then each of its values is represented as a separate category. If **N** is less than the cardinality, the following mapping is applied: the first **N-1** unique values are represented as separate categories and the rest are mapped into one category denoted by triple number sign ("###") or "OTHER".

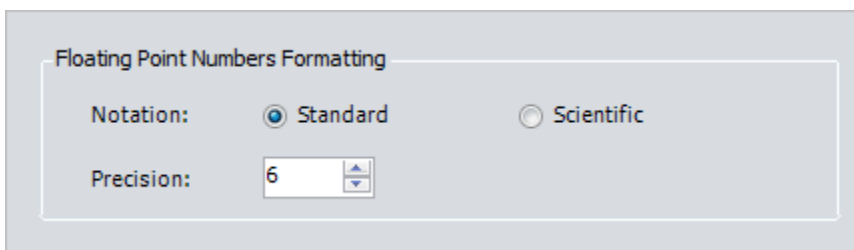
**Missing Values**

The way missing values are treated for different types of independent variables (discrete, continuous, and ordinal). By default, the missing value in a discrete variable is considered as a valid category ("use"). The available options are "use", "ignore", and "substitute mode". Missing values in continuous and ordinal variables are substituted by the median value by default. The available options for continuous are "ignore" (drop records with missing values) and "substitute median / mean / minimum / maximum / mode / zero". The available options for continuous are "use", "ignore" (drop records with missing values), and "substitute median / minimum / maximum / mode".

Page | 587

2. The format and precision for the numbers displayed in the following reports:

- the [Parameters and Attributes](#) tab of trees and models
- the Results tab of Predictive Models (including the Sequence Statistics views that show model parameters and detailed model output)
- the Results tab of Cluster Analysis Models



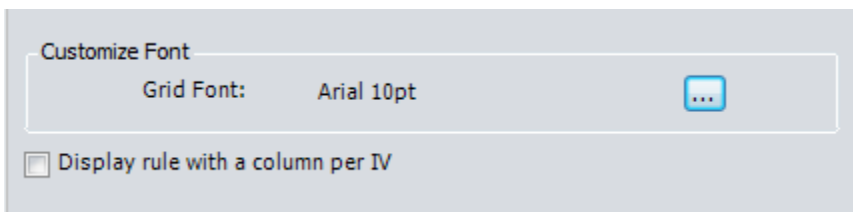
The dialog box titled "Floating Point Numbers Formatting" contains two settings. The "Notation" section has two radio buttons: "Standard" (which is selected) and "Scientific". The "Precision" section has a numeric input field showing the value "6" with up and down arrow buttons.

The default precision for numbers is 6 digits after the decimal point.

In some cases, for example, if a model parameter is displayed as 0.000000, it's actual value may be too small for the default display precision, so you may want to use this dialog to increase the precision or switch to the Scientific notation.

## Node Report

The Node Report view in the Decision Tree provides you with information about the terminal nodes in your tree. The only general preference for the Node Report view is the font setting, which is specified in this tab. Click the browse button to choose the font.



The dialog box titled "Customize Font" has a "Grid Font:" label followed by the text "Arial 10pt" and a small blue button with three dots. Below this is a checkbox labeled "Display rule with a column per IV" which is currently unchecked.

See also: [Node Report](#).

## Overview Report

The Overview Report view in datasets displays the dataset structure and univariate statistics such as the mean, maximum, and minimum values, the number of missing values, etc. See the topic [Overview Report](#) for detailed information about the available statistics.

Page | 588

To customize the report for the current dataset, select the **Tools | Options...** command from the menu or click the **Options** button in the toolbar.

To set the columns to be displayed by default in the Overview Report for all *new* datasets, select **Tools | Preferences** instead of **Options** and choose the Overview Report tab. Changes in Preferences do not affect the datasets already created.

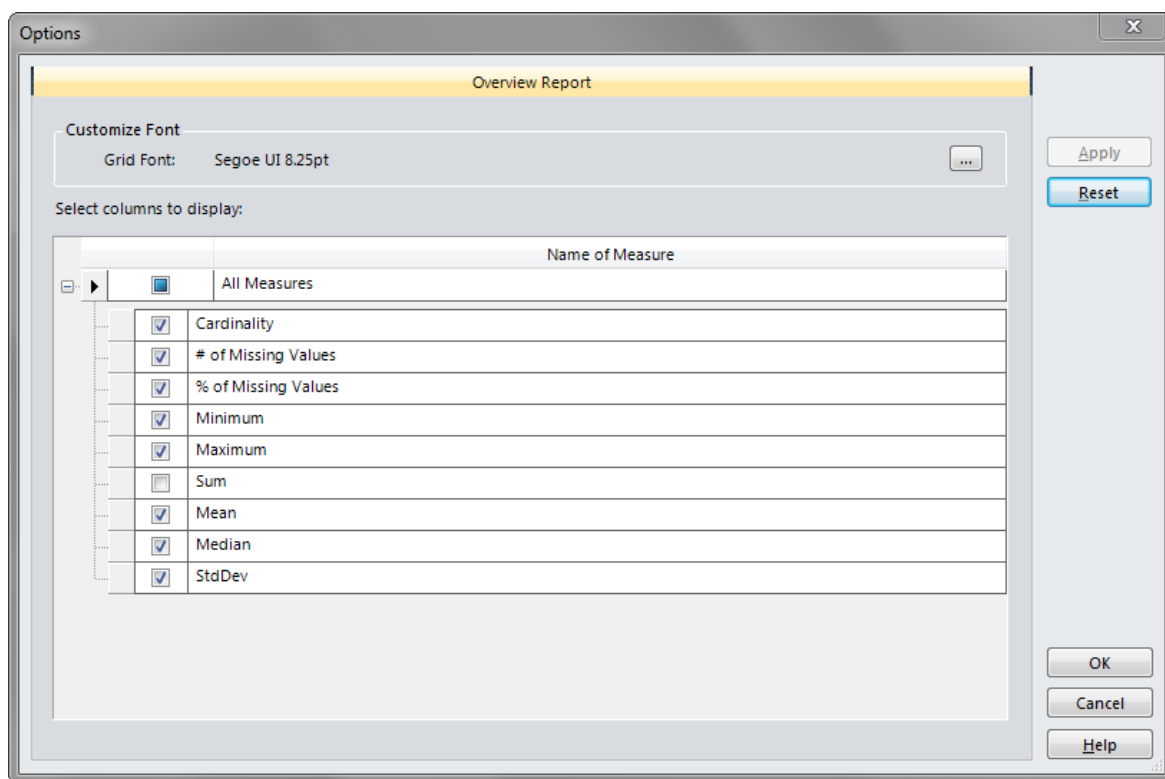
Select the statistics to be displayed in Overview Report. To hide unnecessary statistics, deselect the corresponding boxes. To change the default font for Overview Report, click the button to the right of the Grid Font name and select the desired font.

Nine univariate statistics are available in Options:

- cardinality
- number and percentage of missing values
- minimum and maximum
- sum
- mean
- median
- standard deviation

All these statistics except *Sum* are shown by default for Spark dataframes.





## Segment Viewer

The Segment Viewer tab in the Dataset displays the distributions of the independent variables under each category of the selected segmentation variable. The general settings for the dataset Segment Viewer are specified in this tab.

Discretize integer variables when cardinality less than:

Histogram Settings

Number of bins:

☐ Equal Height ☒ Equal Width

Pie Settings

Number of slices:

Data Presentation Settings

Display data as ☒ Charts ☐ Text-Based Report

***Discretize integer variables when cardinality less than***

The distribution of integer variables whose cardinality is less than or equal to this number will be displayed in the form of bar charts (histograms) by default. Otherwise, it will be displayed as pie charts. The default is 30.

***Number of bins***

Number of bins in which to distribute the data (for histograms).

***Equal Height***

When this is checked, the data is distributed so that each bin contains approximately the same number of observations.

***Equal Width***

When this is checked, the width of the intervals on the x-axis will be the same for all bins.

***Number of slices***

Specifies the maximum number of categories to display on the pie chart.

See also: [Segment Viewer](#).

## Split Report

The **Split Report** view of the Decision Tree displays the parameters of all splits at the currently selected node of the decision tree, provided that their significance (i.e., the value of the *Info* parameter) exceeds the Significance Threshold specified in the Split Report preferences or options. The default value of Significance Threshold is zero. Adjust the value of this parameter as desired if you want to filter out the splits for the least significant variables in Split Report.

For more details on the informativeness of splits and the meaning of the *Info* parameter, see Section 1.2.1 and the subsequent sections of Mathematical Formulations: [Decision Tree](#).

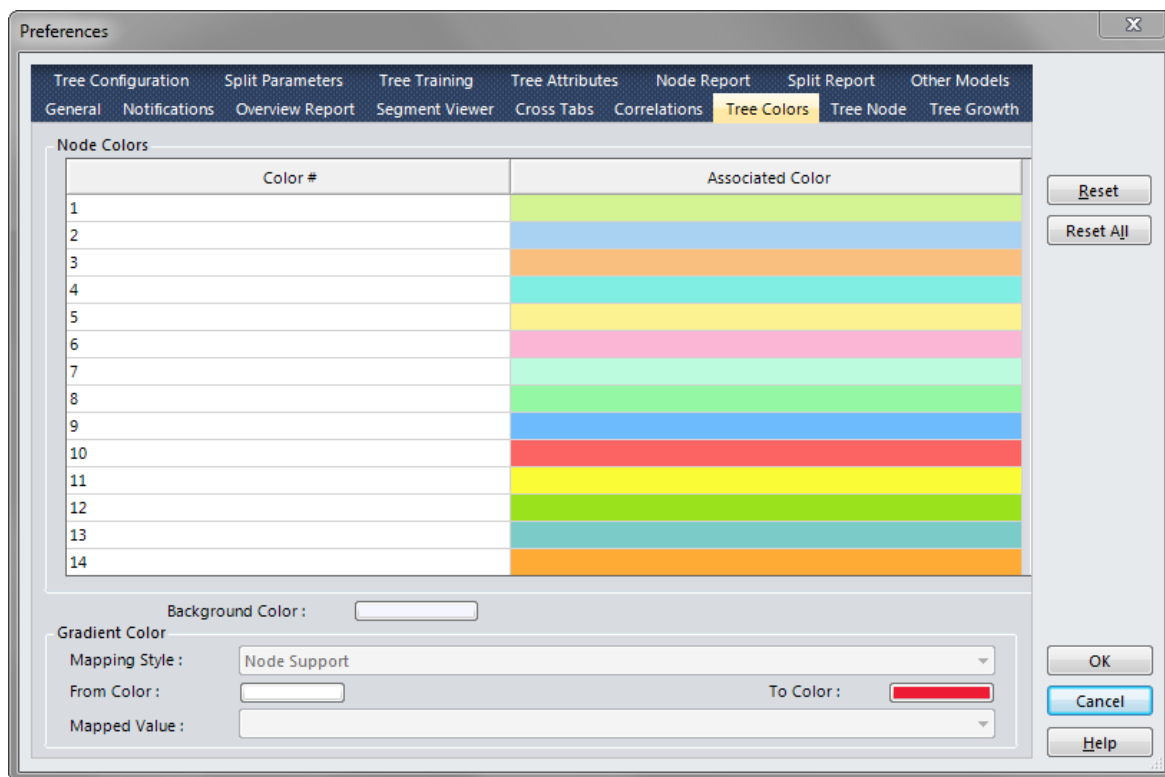
See also: [Split Report](#).

## Tree Colors

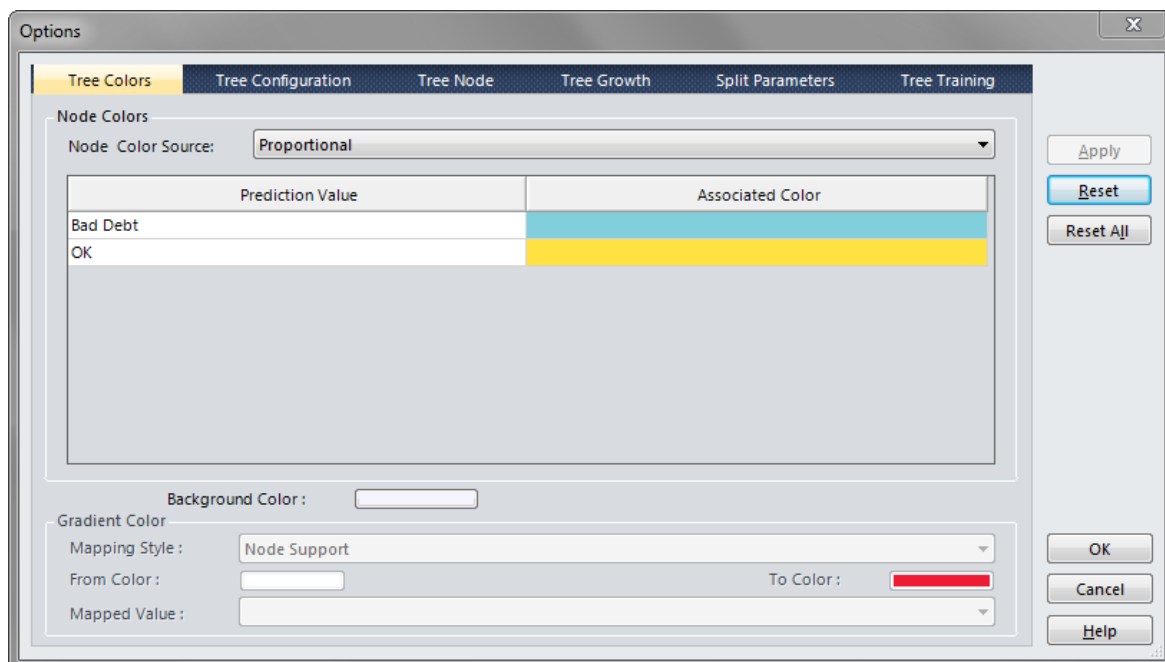
Color options and preferences depend on the tree type. The first subsection of this topic deals with Decision Tree color options and preferences. Strategy Tree colors are described in Subsection 2.

### 1. Decision Tree Colors

The default color settings for the Decision Tree and its background are specified in the **Tree Colors** tab of the Options dialog (menu: **Tools | Options**) and Preferences dialog (menu: **Tools | Preferences**). These settings affect the Tree tab of the Decision Tree object.



- For trees with **discrete** dependent variable (DV), the Node Colors list in this dialog box assigns a color to each value of the dependent variable. The values are listed in alphabetical order. This setting applies if the Node Color Source is set to *Proportional* in **Tools | Options | Tree Colors**.



- The default color settings for trees with **continuous** DV are defined by the Gradient Color options (Mapping Style and From/To Color).
  - Mapping Style determines which parameter is the Gradient Color is based on. The options are: *Node Support* (the total number of records in a node), *Average* (the Average value of the DV in a node), and *StdDev* (the Standard Deviation of the DV in a node).
  - From/To Color: determines the color boundaries on the spectrum. The tree nodes will be assigned the colors within these color boundaries. For example, if the Mapping Style is *Average*, the color chosen as the From Color will be assigned to the node with the minimum value of Average, and the color chosen as the To Color will be assigned to the node with the maximum value of Average.

### Changing the colors:

To assign a custom color to a DV category, click in the corresponding row of the Associated Color column. To specify a background color for the tree, click in the Background Color field. To specify the color boundaries for the Gradient Color style, click in the From Color or To Color fields.

The color palette will appear. Select the desired color and click OK to apply the change.

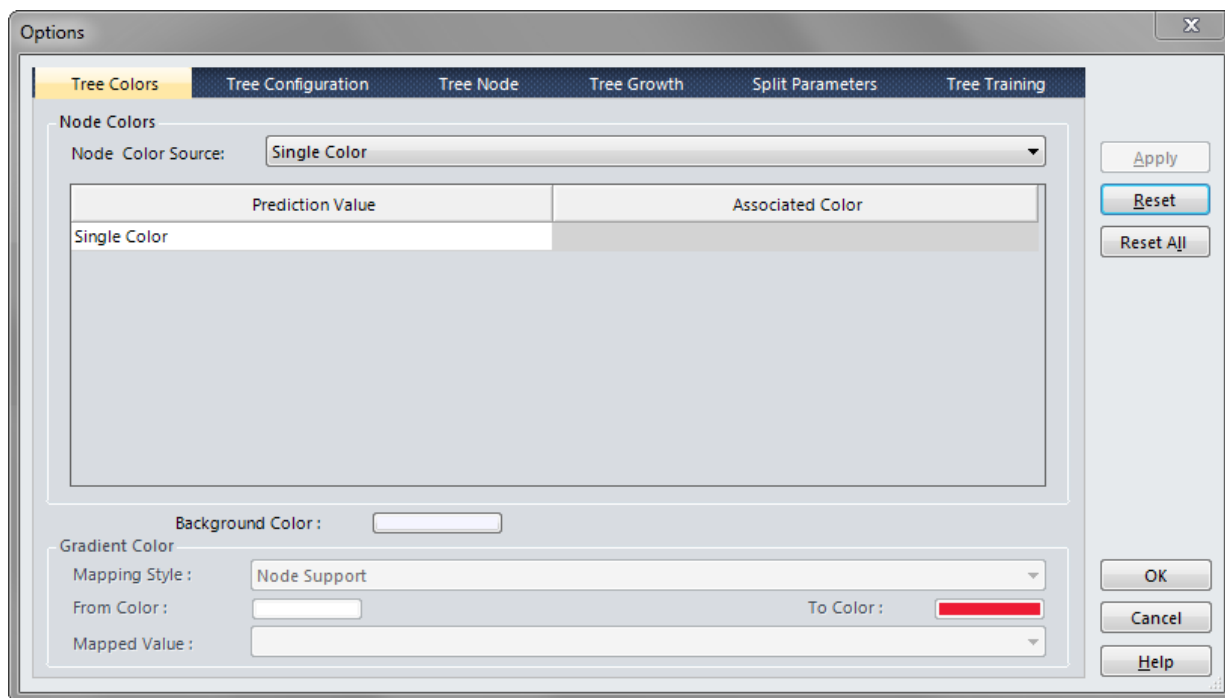
## 2. Strategy Tree Color Options

There are five Strategy Tree color schemes, depending on the selected *Node Color Source*.

The *Proportional Calculations* and *Gradient* schemes are based on the values of existing node calculations. The *Expression* color scheme allows you to enter a custom expression to assign node colors based on a specific condition. The *Treatments* scheme allows you to color nodes based on the [treatments](#) assigned to them.

### a) Single Color

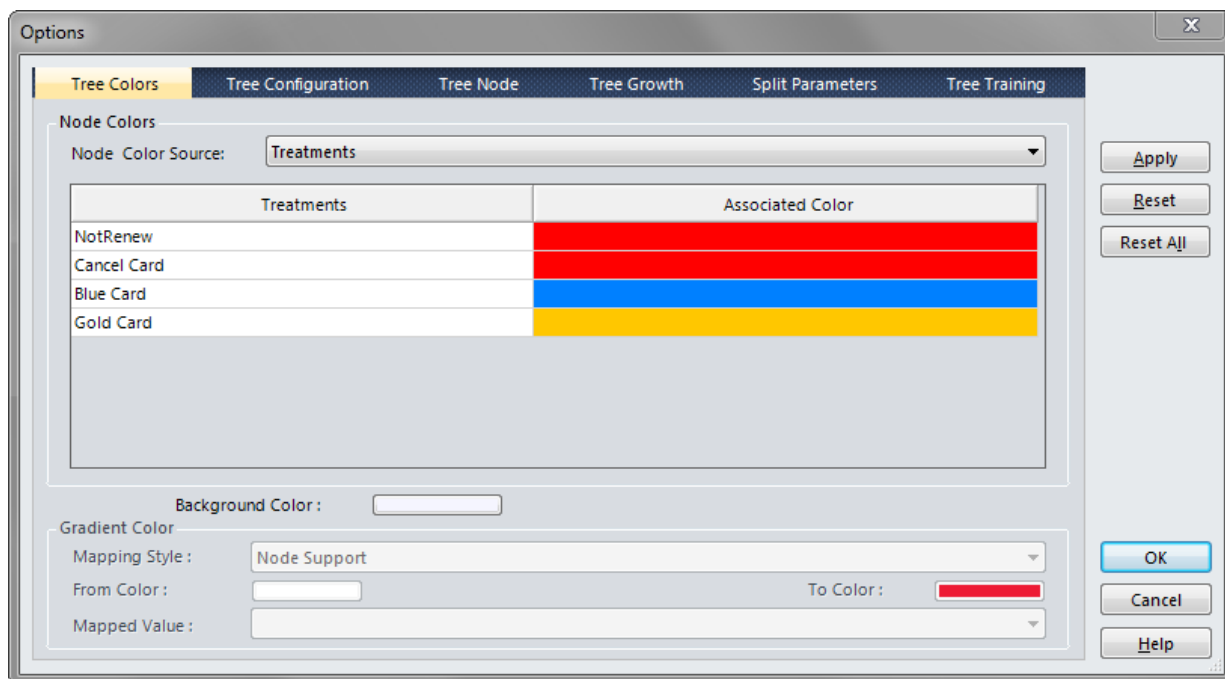
The first, and the simplest scheme is **Single Color**, where all nodes have the same color. The default is gray. You can change it by clicking in the Associated Color column.



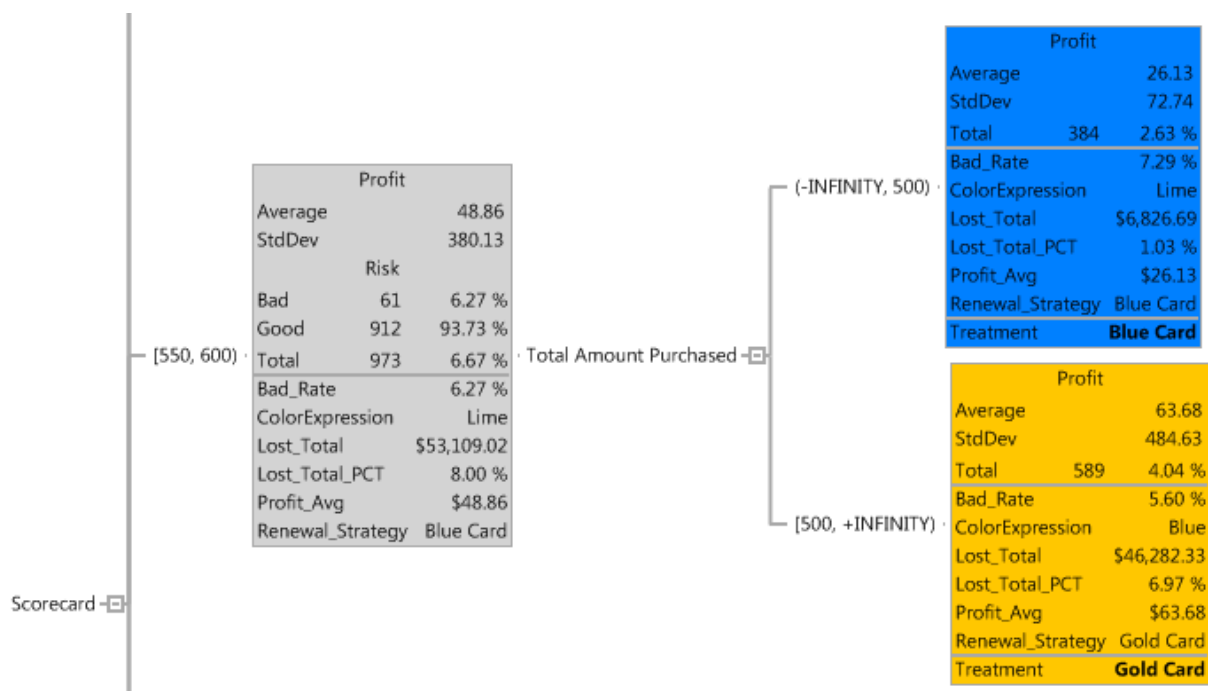
Page | 593

### b) Treatments

The **Treatments** option is available in the Node Color Source list if there is at least one treatment defined in the Strategy Tree. If you select Treatments as Node Color Source, all available treatments are listed in the left column and the associated colors are shown in the right column. Click in the Associated Colors column if you want to change the default colors for the treatments. In the example below, there are three treatments in a Strategy Tree defining a renewal strategy for credit card accounts: 'Cancel Card', 'Blue Card' offer, and 'Gold Card' offer.



A fragment of the Strategy Tree with this color scheme is shown below.



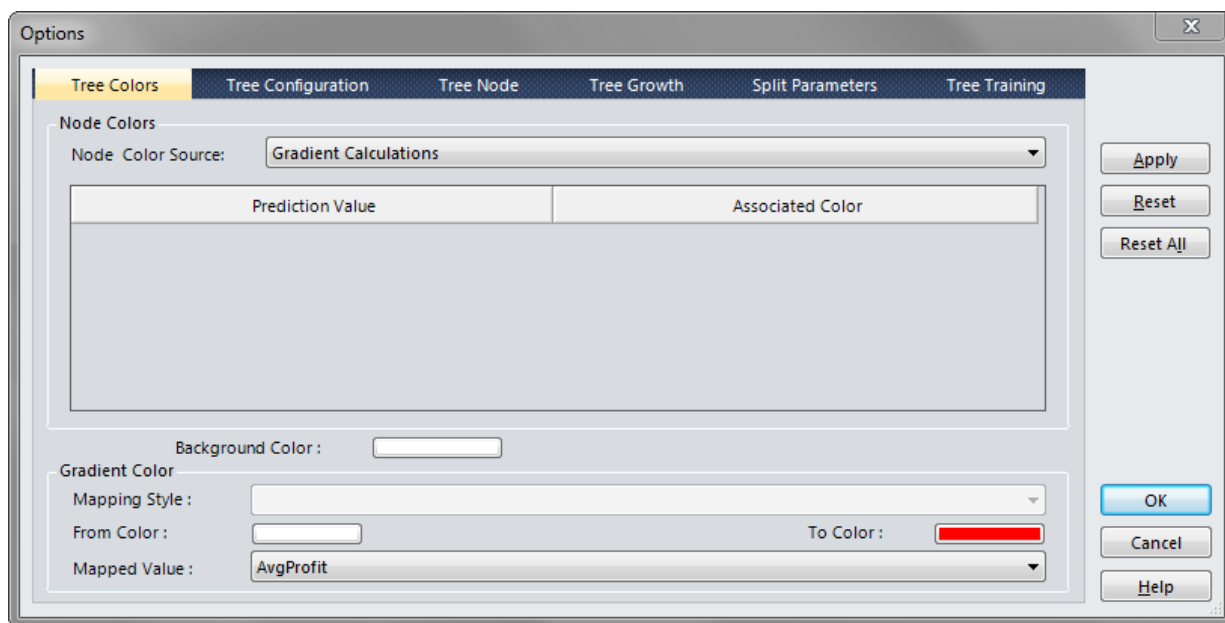
### c) Gradient Calculations

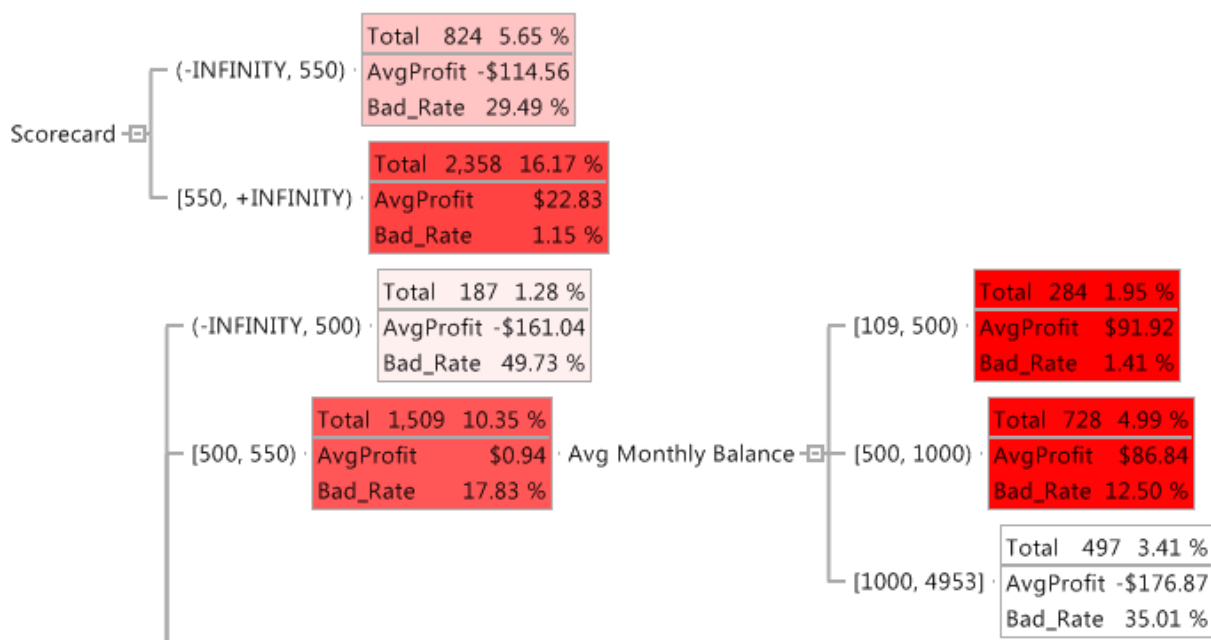
The **Gradient Calculations** color scheme uses gradations of color to express the numerical value of a node calculation. The color range (*From Color*, *To Color*) corresponds to the range of values of the calculation. See the Options dialog below. The higher the calculation value at the node, the closer its color is to the color on the right-hand side (*To Color*). The lower the calculation value at the node, the closer its color is to the color on the left-hand side (*From Color*). The node calculation can be selected from the **Mapped Value** list.

Page | 595

The default color range is from white to red, so, by default, the calculation value is expressed with the gradation of red. Click on the From and To Color fields if you want to change the default color range.

In the example below, the calculation called *AvgProfit* (average Profit) is selected as Mapped Value. A fragment of a Strategy Tree with this color scheme for *AvgProfit* is shown.





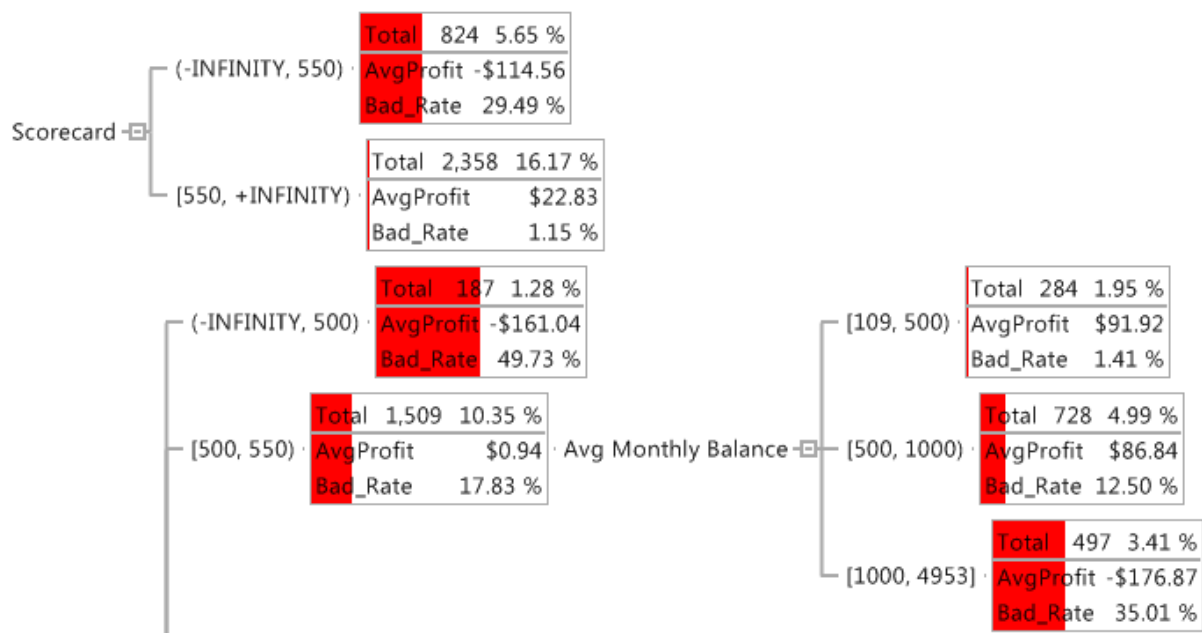
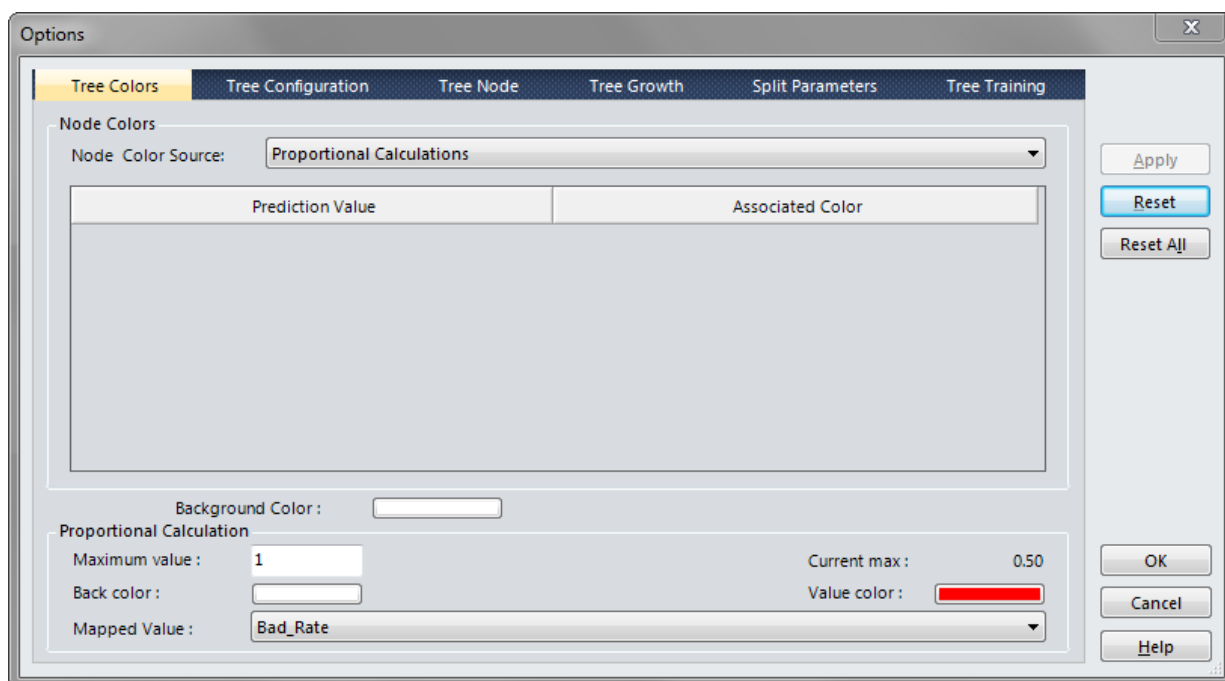
The higher the value of AvgProfit, the more red the node appears.

#### d) Proportional Calculations

In the **Proportional Calculations** color scheme, portions of nodes are colored to express the numerical value of a calculation. This color scheme is suitable mainly for positive-valued calculations, especially discrete distributions such as percentage of a specific value.

Select the desired node calculation from the **Mapped Value** list. The higher the calculation value at the node, the larger is its portion colored in *Value Color*, which is red by default. The rest of the node has color selected as *Back Color*, which is white by default. See the Options dialog below. As a result, nodes at which the calculation value equals the *Maximum value* specified in the Options dialog are entirely colored in *Value Color*. Nodes with zero value of the calculation are entirely colored in *Back Color*.



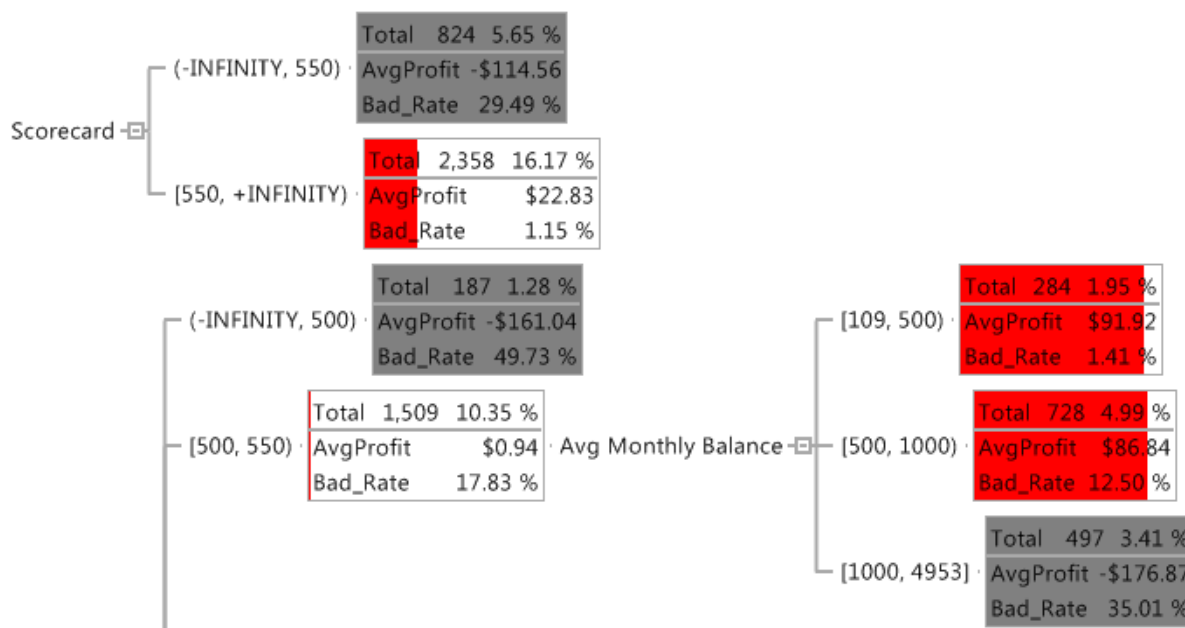


In the example above, approximately one third of the bottom right child node is colored in red because  $Bad\_Rate=35.01\%$  ( $0.3501$ ), and  $0.3501$  is approximately one third of  $Maximum\ value = 1.00$  specified in the Options dialog.

Note that you can set the *Maximum value* to be as high or low as desired. Therefore, at some tree nodes the calculation value may be higher than the *Maximum value*. In this case the *Current max* shown in the Options dialog exceeds *Maximum value*. Nodes at which the value of the calculation is either negative or exceeds the *Maximum value* are entirely grey.

For example, if we choose *AvgProfit* as Mapped Value and set the *Maximum value* to 100, then the same tree fragment will look as follows:

Page | 598

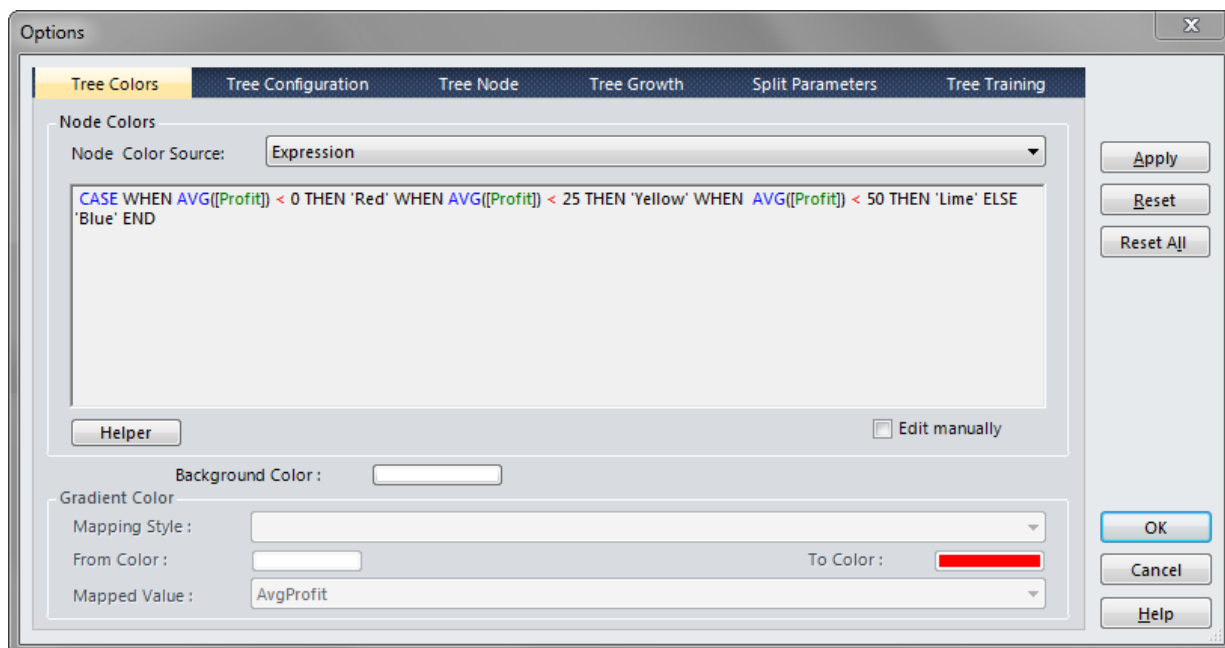


*AvgProfit* = 91.92 at the top-right node, so approximately 9/10 of it this node is colored in red because 100 is the *Maximum value* specified in the Options dialog. The bottom-right child node is grey because the value of *AvgProfit* is negative (-176.87). Any child node with *AvgProfit* > 100 would also be grey.

Unlike the predefined *Maximum value*, **Current max** changes as you grow the tree. *Maximum value* is a fixed, user-defined reference point and the threshold beyond which the nodes will be colored in grey.

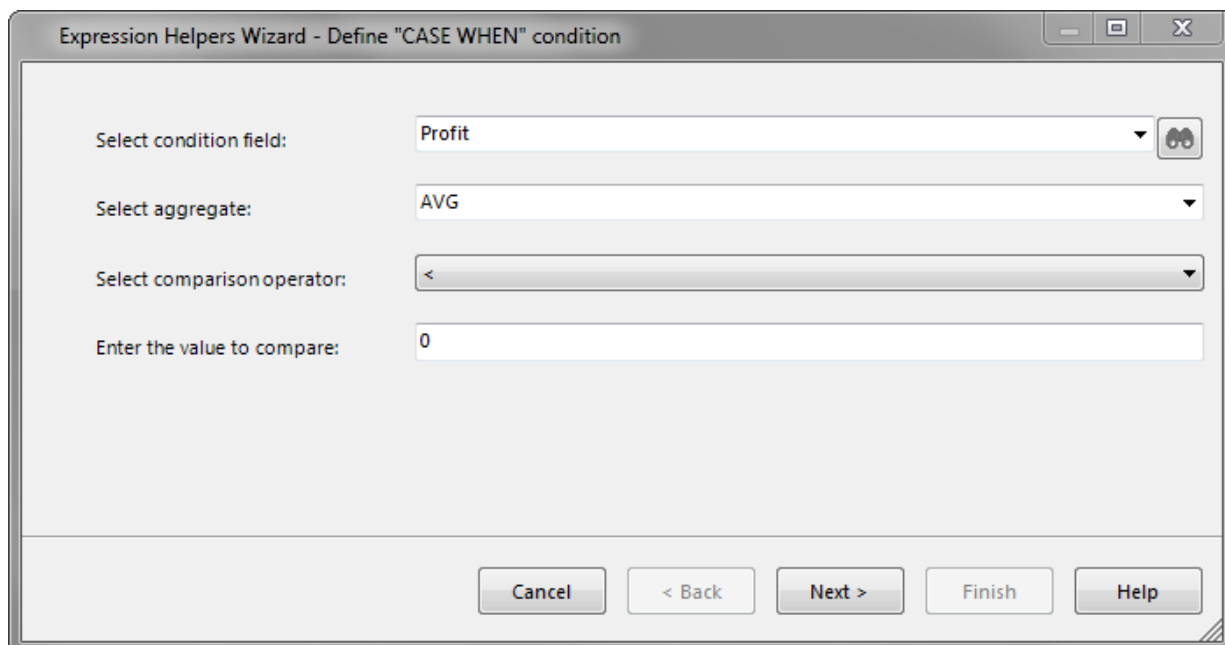
### e) Expression

This option allows you to specify an arbitrary expression defining node colors. For example, this way you can assign different colors to nodes depending on the interval that the  $AVG([Profit])$  calculation belongs to. See an example below.

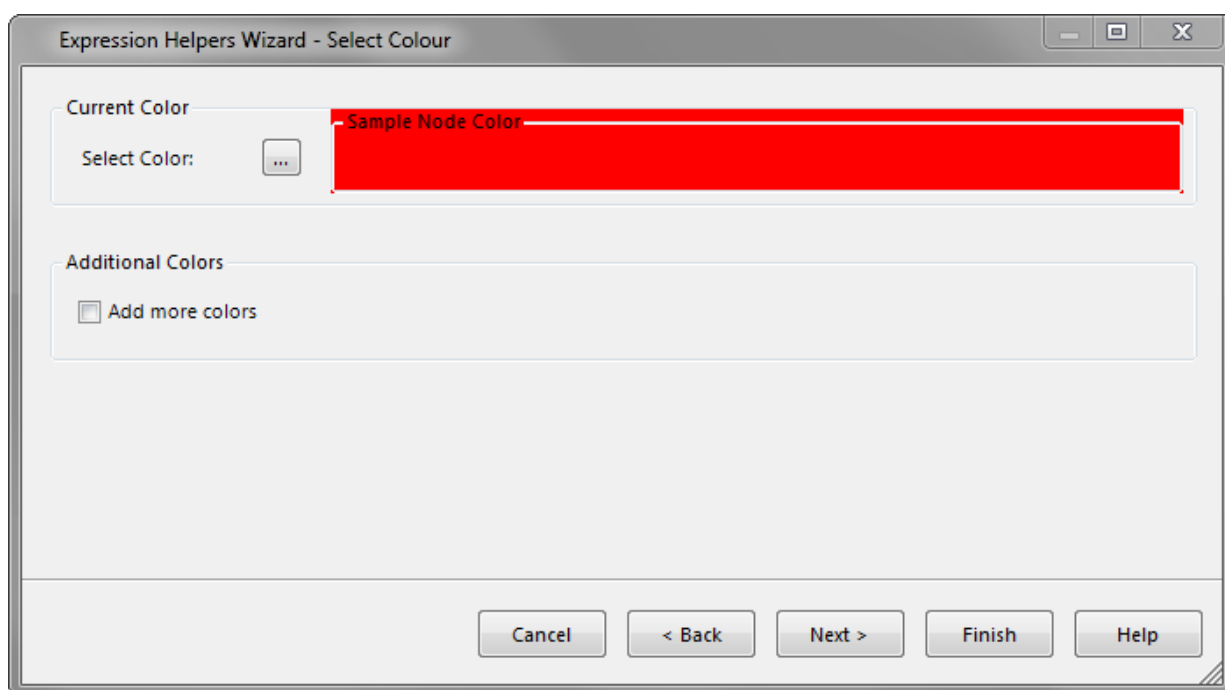


To enter an expression manually, check the *Edit manually* box.

Alternatively, you can click the Helper button to run the wizard that lets you specify conditions and the corresponding colors. Specify the CASE WHEN condition. In the example below, the resulting condition will be CASE WHEN AVG([Profit]) < 0



Click Next and specify the color to assign to nodes with  $\text{AVG}([\text{Profit}]) < 0$ .



Page | 600

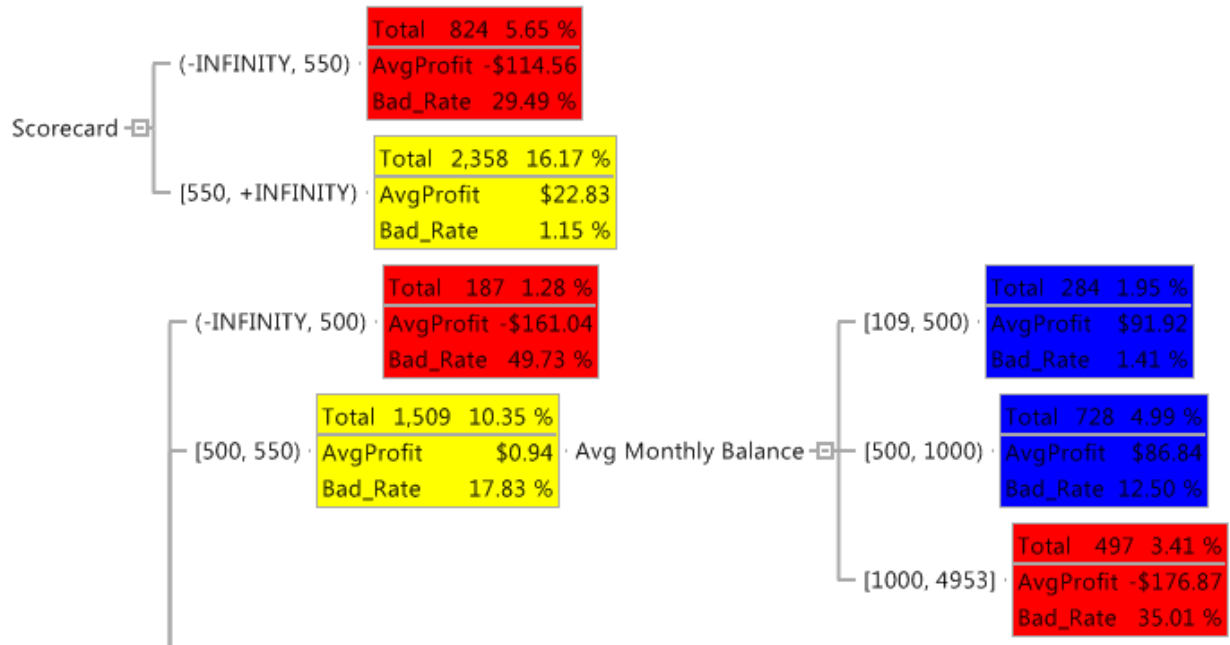
At this stage, the resulting expression will be `CASE WHEN AVG([Profit]) < 0 THEN 'Red'`.

Select the *Add more colors* option if you need to add more conditions, for example specify a different color for  $\text{AVG}([\text{Profit}]) > 100$ .

In the above example, the resulting expression is obtained in four steps of the wizard using the *Add more colors* option:

```
CASE WHEN AVG([Profit]) < 0 THEN 'Red' WHEN AVG([Profit]) < 25 THEN 'Yellow'
      WHEN AVG([Profit]) < 50 THEN 'Lime' ELSE 'Blue' END
```

A fragment of the resulting tree is shown below:



## Tree Configuration

The settings in the Tree Configuration tab of the Preferences dialog define the default font, orientation, and layout of the tree.

|                                                    |    |                                                                                                     |
|----------------------------------------------------|----|-----------------------------------------------------------------------------------------------------|
| <b>Customize Font</b><br>Main Font : Segoe UI 10pt |    | <b>Orientation</b><br><input checked="" type="radio"/> Vertical<br><input type="radio"/> Horizontal |
| <b>Layout</b>                                      |    |                                                                                                     |
| Tree Border                                        | 2  | <input checked="" type="checkbox"/> Show Plus/Minus Box                                             |
| Node Border                                        | 3  | Plus/Minus Box Size 8                                                                               |
| Parent Distance                                    | 15 | <input checked="" type="checkbox"/> Show Minimap                                                    |
| Line Thickness                                     | 2  | <input checked="" type="checkbox"/> Anti-Aliasing <input type="checkbox"/> Show Contour             |

**Main Font** The font used in the decision tree nodes and marginal rules. The default is Tahoma, 10pt.

**Vertical** The tree is displayed vertically. This is the default orientation for Decision Trees.

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Horizontal</b>          | The tree is displayed horizontally. This is the default orientation for Strategy Trees.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Show Plus/Minus Box</b> | Show or hide the plus/minus box beneath each node that has a split, which serves as a shortcut to the <b>Expand / Collapse</b> command under the <b>View</b> menu. If the plus is displayed, the tree branch under the node is collapsed. Clicking on it will expand the branch and the plus will turn into minus. Conversely, clicking on the minus sign collapses the branch. The plus/minus box is not shown by default.<br><br>The size of the plus/minus box can be customized by adjusting the value in the Plus/Minus Box Size field. The default size is 8. |
| <b>Tree Border</b>         | Determines the distance from the root node to the top of the Tree view window. The default is 2.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Node Border</b>         | Determines the distance between the adjacent tree nodes of the same level. The default is 3.                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Parent Distance</b>     | Determines the vertical distance from the child nodes to their parent node. The default is 10.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Line Thickness</b>      | Determines the thickness of the lines connecting child nodes with their parent node. The default is 2.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Anti-Aliasing</b>       | Smooths the lines connecting child nodes with their parent node. This option is turned on by default.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Show Contour</b>        | Show or hide the contour around currently selected node and the whole tree branch beneath this node. Contours are not shown by default.                                                                                                                                                                                                                                                                                                                                                                                                                             |

## Tree Growth

The settings in the Tree Growth tab of the Preferences dialog define three parameters that control the tree growth: the minimum node creation size, the Auto-Grow stop size, and the maximum tree depth.

Stopping Criteria

Node Size:

☒ Percentage of training data
☐ # Records

Non-Terminal Nodes:

3

%

Terminal Nodes:

1

%

Maximum Tree Depth:

5

☒ Allow consecutive splits on same variable

**Node Size Limits in Percentage or Number of Records**

If the **Percentage** option is selected, the node size limits must be specified in percentage of total. This option is selected by default and is recommended, since it is logical to set these limits relative to the size of the input dataset. Select the option **# Records** if you want to specify the node size limits in absolute numbers of records.

**Minimum Node Size for Non-Terminal Nodes**

The minimum number of records for a node to contain before the **Find Split** or **Automatic Grow** operations are allowed to split it. For the option "Percentage of training data", the default is 3%.

**Minimum Node Size for Terminal Nodes**

The minimum number of records allowed for terminal nodes. For the option "Percentage of training data", the default is 1%.

**Maximum Tree Depth for Automatic Grow**

The maximum number of levels allowed for the **Automatic Grow** command to grow the decision tree. The number of levels includes the root node, so the depth of a tree with a single split is 2. Enter the value of zero (0) for unlimited depth (in that case, the depth is limited only by the minimum node size constraints defined in this dialog). The default is 5.

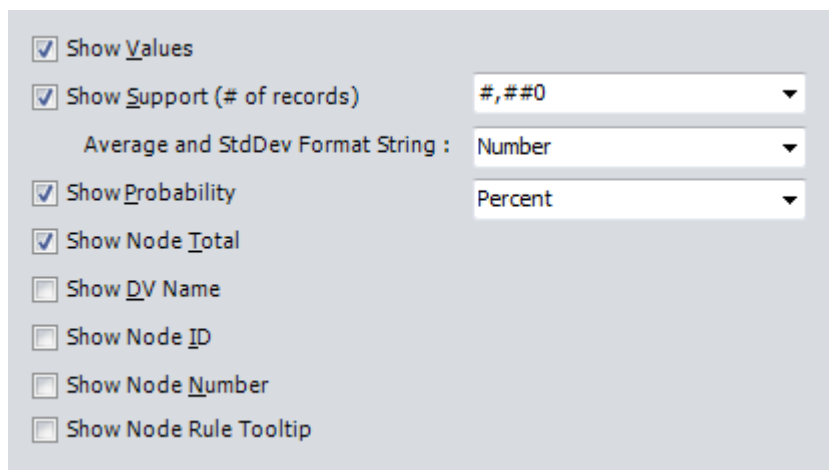
**Allow consecutive splits on same variable:** If this option is selected, splits on the same variable are allowed on two consecutive levels of the tree, on the same branch. This option is selected by default.

Note that for individual trees you can also use the Tree Growth tab under **Tools | Options** to limit your Automatic Grow operation depending on the number (or percentage) of specific DV values in the nodes. See the topic [Automatic Grow](#) for details.

## Tree Node Settings

The settings in the Tree Node Settings tab of the Preferences dialog determine the information to be displayed in the tree nodes by default.

Page | 604



☒ Show Values

☒ Show Support (# of records)    #,##0

Average and StdDev Format String :    Number

☒ Show Probability    Percent

☒ Show Node Total

☐ Show DV Name

☐ Show Node ID

☐ Show Node Number

☐ Show Node Rule Tooltip

**Show values**

Shows the values of the dependent variable (for a discrete DV) or Average and StdDev (for a continuous DV) in the node.

**Show Support (# of records)**

Shows the total number of records in a node and the number of records for each DV value (for a discrete DV) or the values of Average and StdDev (for a continuous DV).

**Show Probability**

Shows the percentage of the number or records for each DV value (for a discrete DV) and the percentage of total number of records (if the Show Node Total box is checked).

**Show Node Total**

Shows the total number of records and percentage of total in each node.

**Show Title**

Shows the name of the DV in each node as a node title.

**Show Node ID**

Shows the node ID in each node.

## Tree Split Parameters

The settings in the Tree Split Parameters tab of the Preferences dialog determine the information to be displayed for each split in the tree.



**Display Settings**

☐ Marginal Probability
 ☐ Marginal Support

☒ Split Label
 

#,##0

**Select Split Parameters**

| Parameter Name | Include                  | Format string |
|----------------|--------------------------|---------------|
| B              | <input type="checkbox"/> |               |
| C              | <input type="checkbox"/> |               |
| F              | <input type="checkbox"/> |               |
| P              | <input type="checkbox"/> |               |
| chi2           | <input type="checkbox"/> |               |

### **Marginal Probability**

Displays the probability of records being assigned to each node (in percent). For any node, the sum of the marginal probability values for all its child nodes must add up to 100%.

### **Split Label**

For each split in the tree, displays the name of the independent variable that defines the split rule.

### **Marginal Support**

If the split variable is discrete: For each value of the split variable, displays the number of all records in the node that contain this value.

If the split variable is continuous: Displays the number of records for each value range of the split variable in the marginal rules.

### **The Split Parameters list**

The list of all split parameters for all available measures. Each measure has a different set of split parameters. For any given tree, only the split parameters applicable to the measure of the current tree can be displayed.

You can specify the display format for each split parameter in the Format String column of this dialog. In the above example, the confidence level (C) will be displayed with four and P-value with two digits after the decimal point, and chi2 will be displayed in scientific notation. For more information on format strings, see the topic [Specifying Format](#) in the Expression Editor section.

For the explanation of parameters for each measure, see the topic [Supported Measures](#) in the section "Decision Trees --> Inserting a Decision Tree". Also see the [Decision Tree](#) section in Mathematical Formulations.

## **Tree Training**

The settings in the Tree Training tab of the Preferences dialog determine main parameters of the tree growth algorithm (the split search method, the measure, and the relevant thresholds).

Split Search Method

☒ Cluster
☐ Forward Exhaustive
☐ Exhaustive

Bisection Samples:

Measure

Bonferroni-adjusted p-value

Continuous Filter Threshold (percent error):

Discrete Filter Threshold (percent error):

Bonferroni Adjustment:

Bonferroni Adjustment Type:

☒ Kass
☐ Biggs

**Cluster**

The *Cluster* method finds groups that maximize similarity within the groups and dissimilarity between the groups. Merging stops when either there is only one group left or there is no pair of groups that is "similar" enough with respect to the "merging threshold". The Cluster method is faster and more efficient than Exhaustive, but may potentially stop at slightly less optimal solutions. This method is the application default. See the [Find Split](#) topic for more technical details.

**Exhaustive**

The *Exhaustive* method performs an exhaustive search to find groups that maximize statistical significance. Merging stops when there is only one group left. While merging, the sequence of merges is recorded in a special history buffer, noting the "best", global, solution found to date. When the merging process is finished, a re-initialization to the start point is performed, and the sequence of merges recorded in the aforementioned "history buffer" is performed. This process stops as soon as the "best" history entry (i.e., the noted best solution) is encountered. Splits derived from the Exhaustive method display the most significant relationships. See the [Find Split](#) topic for more technical details.

**Forward Exhaustive**

This split search method is optimized for ordered variables. It begins with no split points, and adds them iteratively. It combines binary split search applied to the bins of the current (k-th) iteration and ternary splits on the bins obtained at the previous iteration (k-1). In addition, at each iteration it uses a discrete mesh of potential split points to increase the chances of achieving the global minimum p-value, rather than stopping at a local minimum. The **Bisection Samples** parameter determines the number of nodes in this mesh. The default is 10. The admissible values are nonnegative integers. Zero means no sampling (i.e., a complete sweep is performed). The higher the value, the more likely the method is to find the optimal global solution – at the expense of performance. These optimizations affect only ordered variables –

**Measure**

the variables whose Usage is 'ordinal' or 'continuous'. For splits on nominal variables, the regular Exhaustive method is used.

Six measures are available: Entropy Variance; Gini Variance; Unadjusted p-value; Unadjusted log p-value; Bonferroni-adjusted p-value, Bonferroni-adjusted log p-value. For the description of the measures, see [Supported Measures](#) in the section "Decision Trees -> Inserting a Decision Tree". Also see the [Decision Tree](#) section in Mathematical Formulations.

Page | 607

**Continuous Filter Threshold (percent error)**

*This option is applicable only for the Bonerroni-adjusted measures.*

When analyzing data using the adjusted or unadjusted P-value measure, you run the risk that your data may display results obtained through chance, which can be potentially misleading. By adjusting the filter threshold value, you can lower the possibility that chance will play a part in your analysis - you determine how rigorously your data will be analyzed.

**Discrete Filter Threshold (percent error)**

Continuous and discrete filter thresholds are for trees with continuous and discrete dependent variable, respectively. These thresholds are based on the confidence level (C) of a split, which is described in the section "Mathematical Formulations" (Decision Trees). The **Info** parameter in Split Report (same as Rank in the Go To Split dialog) is the value of C expressed in percentage. For example, if C=0.95, then Info = Rank = 95%.

Note: If you want the parameter C to be displayed at every split of the current tree (if the tree uses a P-value measure), select **Tools | Options** from the main menu, go to the Tree Split Parameters tab, and select the checkbox for C.

By specifying the filter threshold, you define the minimum value of Info (or Rank) that is of interest to you, so that the Find Split and Automatic Grow operations would not show the splits on independent variables whose rank is lower than 100% minus the specified error. For example, if you are interested only in the independent variables whose Rank is higher than 98%, specify 2 as the filter threshold (percent error). In this case, variables with Rank less than 98% (= 100% - 2%) will not appear in the Go To Split dialog and no statistics will be displayed for them in Split Report..

**Bonferroni Adjustment**

*This option is applicable only for the Bonerroni-adjusted measures.*

To minimize the discovery of chance relationships, Bonferroni adjustments let you adjust the level of significance used when testing the validity of a relationship.

This type of adjustment compensates for amount of redundant variables in your dataset. Normally, if a concept is captured by more than one variable, it has a disproportionate opportunity to enter the analysis by chance. Use the Bonferroni adjustment to compensate for this possibility.

**Bonferroni  
Adjustment Type  
(Kass / Biggs)**

Bonferroni adjustments are quite conservative. Increasing the size of the adjustment increases the level of conservatism. It is not meaningful to set this value greater than the total number of variables in your dataset.

Two options are available for the Bonferroni adjustment type: **Kass** and **Biggs** (named after the authors of these methods, G.V. Kass and D. Biggs). The Kass type is the default. These options influence the number of child nodes in splits, and their effect depends on the type (nominal, ordinal, or continuous) and the cardinality of the split variable. For example, according to G. Ritschard, the Kass adjustment type tends to penalize high-cardinality predictors more than low-cardinality ones, and nominal predictors more than ordinal or continuous predictors. For details, see the articles by these authors listed in the [References](#) section of Mathematical Formulations.

Thus, for instance, if you increase the *Default number of bins* (the initial number of bins for the Find Split operation) in Attribute Editor before growing the tree with Find Split or Automatic Grow, the Kass adjustment type may be more preferable as the Find Split algorithm will be less likely to pick variables with lower relevance to the target. The Biggs type may be preferable in other contexts. For guidance, see the articles cited above.

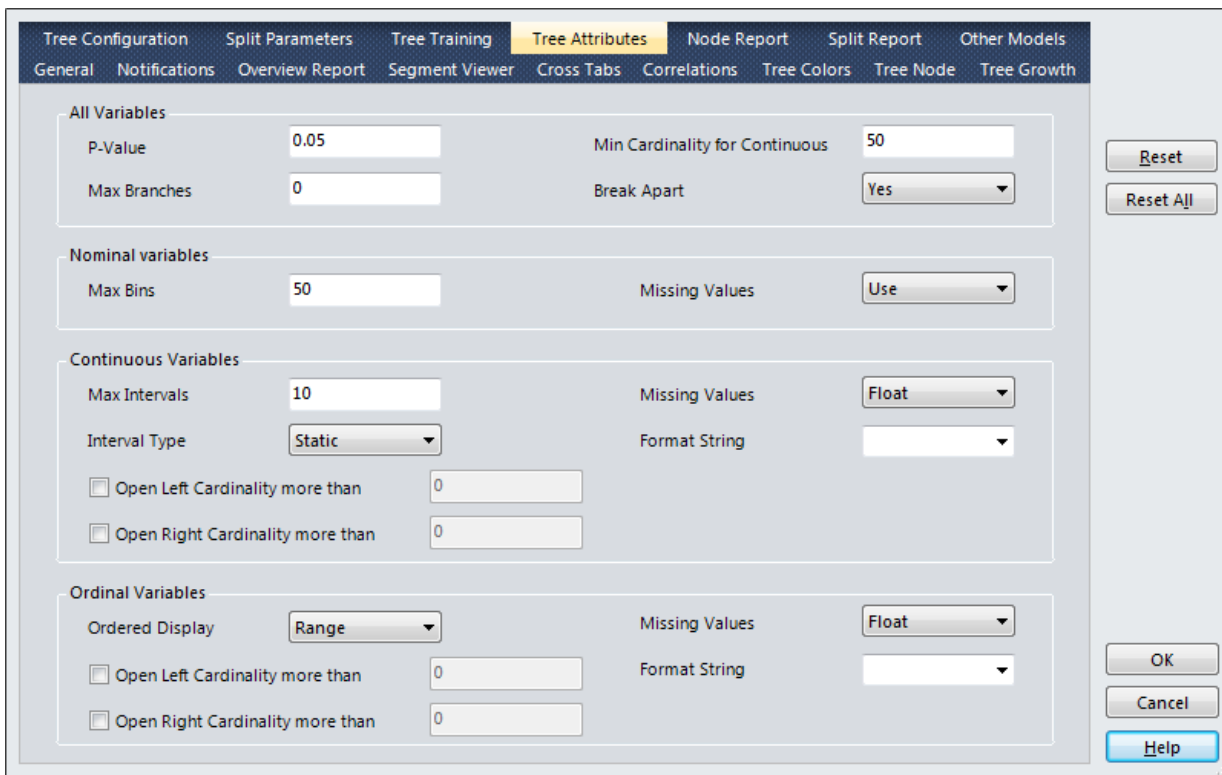
## Tree Attributes

Global preferences for Tree Attributes can be set using the **Tools | Preferences** menu.

The settings in the **Tree Attributes** tab of the **Preferences** dialog determine the defaults in the **Attribute Editor** for Decision Trees and Strategy Trees. These are the default attributes for any new decision tree or strategy tree to be created.

(The [Attribute Editor](#) for Decision Trees and Strategy Trees is available under the Tools menu and also in the context menu of any tree when you click anywhere on the tree background. For detailed description of the Attribute Editor options, see the topic [Using the Attribute Editor](#).)

NOTE: These settings do not apply to the dependent variable. They apply to independent variables only.


***P-Value***

The p-value threshold to merge child nodes during the Find Split procedure in trees. The default is 0.05. Increasing this value will make the splits more "bushy", i.e., values of any given the split variable will be less likely to be grouped.

***Max Branches***

The Max Branches value determines the maximum number of branches (leaves) produced by Find Split on the given variable. See the topic [Using the Attribute Editor](#) for details. The default is 0 (which means "no fixed limit", aside from the upper limit set in the column Default # Bins, if applicable).

***Min Cardinality for Continuous***

Numeric independent variables with cardinality higher than this value are considered to be continuous. The default is 50. If you set this threshold to **x**, then all numeric variables with cardinality  $\geq x$  will have Usage = continuous, and all numeric variables with cardinality less than **x** will have Usage = ordinal (i.e., ordered discrete).

***Max Bins (nominal variables)***

The maximum number of categories (bins) to display for nominal variables. The default is 50. If the maximum number of bins **N** is larger or equal to the cardinality of the variable, then each of its values is represented as a separate category. If **N** is less than the cardinality, the following mapping is applied: the first **N-1** unique values are represented as separate categories and the rest are mapped into one category denoted by triple number sign ("###").

**Max Intervals  
(continuous  
variables)**

The default for the initial number of intervals for continuous independent variables. (To form splits in a decision tree, continuous values are initially divided into discrete intervals where each interval has approximately the same number of observations. These discrete intervals are then treated as ordered and combined accordingly. The number specified here determines how many **initial** intervals are formed for this continuous variable prior to the ordered grouping. The resulting number of nodes (intervals) may be less than or equal to this initial value.) See the topic [Using the Attribute Editor](#) for more details.

**Open Left  
Cardinality more  
than...**

Applies to continuous variables or ordinal variables displayed in ranges. If this option is selected, the leftmost interval of any variable with cardinality higher than the specified value will be open from the left.

**Open Right  
Cardinality more  
than...**

Applies to continuous variables or ordinal variables displayed in ranges. If this option is selected, the rightmost interval of any variable with cardinality higher than the specified value will be open from the right.

**Format String**

This determines the default display format or precision for numeric independent variables. For the description of format strings that can be specified, see the topic [Specifying Format](#) in the Expression Editor section.

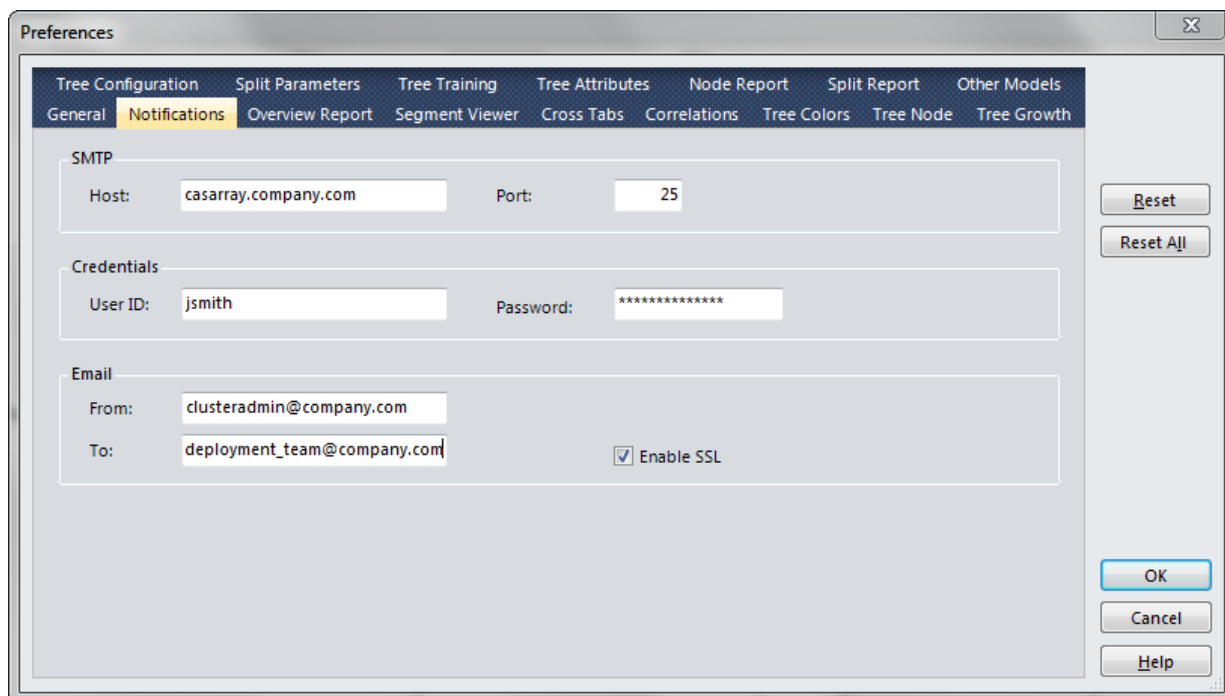
The Missing Values default is *Float* for both Ordinal and Continuous variables.

The default Ordered Display option for Ordinal variables is *Range*.

## Notifications

To configure the Email notifications for scheduled tasks, use the **Notifications** tab in the **Preferences** dialog accessible from the **Tools** menu. The notifications can be set for tasks scheduled in the Multiple Scenario Optimization node (if InsightOPTIMIZER features are enabled) and all [Spark-integrated nodes](#) (in Knowledge Studio for Apache Spark).

Specify the SMTP host name or IP address, SMTP port, User ID and password (optional), and the *From/To* Email addresses. The option **Enable SSL** is selected by default, but you may need to disable it (depending on the security settings in your Email client). If you are not getting notifications after configuring these settings, deselect the Enable SSL option and try again.



Preferences

Tree Configuration | Split Parameters | Tree Training | Tree Attributes | Node Report | Split Report | Other Models

General | **Notifications** | Overview Report | Segment Viewer | Cross Tabs | Correlations | Tree Colors | Tree Node | Tree Growth

SMTP

Host:  Port:

Credentials

User ID:  Password:

Email

From:

To:  ☒ Enable SSL

Reset  
Reset All  
OK  
Cancel  
Help

**See also:****[Scheduling Analytic Tasks on Spark](#)****Spark Settings**

The settings in the **Spark** tab of the Preferences dialog define memory management and other configuration parameters for Apache Spark processing in the Knowledge Studio for Apache Spark product for Big Data. They are applicable if you have a Knowledge Studio for Apache Spark license. Adjusting these parameters can help optimize Knowledge Studio for Apache Spark performance.

|                                                                                                               |                                          |
|---------------------------------------------------------------------------------------------------------------|------------------------------------------|
| Monitor jobs (seconds):                                                                                       | <input type="text" value="5"/>           |
| Maximum number of categories for models                                                                       | <input type="text" value="10"/>          |
| <input checked="" type="checkbox"/> Use Spark's OneHotEncoder (as opposed to custom Knowledge Studio Encoder) |                                          |
| <input checked="" type="checkbox"/> Cache dataframes in memory                                                |                                          |
| <input checked="" type="checkbox"/> Precalculate univariate statistics when dataframes are created            |                                          |
| Pandas threshold (MB)                                                                                         | <input type="text" value="10"/>          |
| Use Pandas on                                                                                                 | <input type="text" value="Worker node"/> |
| Reshape threshold (number of columns)                                                                         | <input type="text" value="200"/>         |
| Reshape threshold (MB)                                                                                        | <input type="text" value="10000"/>       |

**Monitor Jobs  
(seconds)**

The interval between checkpoints for monitoring Spark jobs (in seconds). This concerns the Spark Tasks log, which is available under the **View** menu. The default is 5 seconds.

**Maximum number of  
categories for  
models**

The maximum number of categories for discrete (nominal) variables in models built on Spark. If the cardinality of the variable does not exceed the number **N** specified here, then no mapping is applied (each value is a separate category). Otherwise, the first N-1 unique values are represented as separate categories and the rest are mapped into one category denoted by triple number sign ("###") or "OTHER". The default is 10.

**Use Spark's  
OneHotEncoder  
(as opposed to  
custom  
Knowledge  
Studio encoder)**

This option concerns the preprocessing of discrete variables in computations involved in modeling on Spark. It determines what method is used to encode discrete integer variables using the one-hot (one-of-K) schema. See the documentation for Spark (<https://spark.apache.org/docs/latest/ml-features.html#onehotencoderestimator>). The user may choose to use Spark's implementation or a custom Knowledge Studio implementation. Spark's implementation is used by default.



**Cache  
dataframes in  
memory**

When this option is turned on, dataframes are cached upon import. This saves time for the calculation of statistics for Spark dataframes, decision trees and strategy trees. The import process takes longer, but calculations are processed faster later.

**Precalculate  
univariate  
statistics when  
dataframes are  
created**

When this option is turned on, univariate statistics such as minimum, maximum, mean, sum, etc., are calculated when Spark dataframes are being created. It takes longer to process the dataframe import, but analysis tasks take less time later as they use precalculated statistics.

Note that different methods are used for calculating the statistics depending on the dataframe characteristics. If the dataframe size is smaller than the Pandas threshold specified below, the [Pandas](#) functions are used for calculations. Dataframes with more columns than the threshold specified below are reshaped to long format, and statistics are computed using a Spark SQL groupBy. Otherwise, the calculation loops over all columns one-at-a-time.

The cached calculation results will be transferred to derived dataframes where applicable (for example, a dataframe produced by the Variable Transformation node will inherit the cached calculation results for the original columns).

**Pandas  
threshold (MB)**

*[Applicable if dataframe caching is enabled]*. If the size of the cached dataframe is less than this threshold, then [Pandas](#) is used to calculate the statistics. For smaller dataframes, calculation using Pandas can be faster than with the usual Spark methods. Otherwise, the usual Spark methods are used. The default threshold is 10 MB.

**Use Pandas on  
Master or  
Worker Node**

*[Applicable if dataframe caching is enabled]*. This setting affects the dataframes smaller than the Pandas threshold specified above. It controls where on the Spark cluster [Pandas](#) is used to compute univariate statistics. If 'Master node' is chosen, statistics are computed on the cluster node running the driver program. If 'Worker node' is chosen, statistics are computed on a worker node selected by the resource manager. The default is 'Worker node'.

**Reshape  
threshold**

If the dataframe has more than the number of columns specified here AND its size in Megabytes is

---

|                            |                                                                                                                                                                                                                                                                                      |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>(number of columns)</b> | smaller than the threshold specified below, the dataframe is reshaped for calculations (wide-to-long reshape instead of calculations per column). The default threshold is 200 columns. <i>[Not applicable if the dataframe size is below the Pandas threshold specified above.]</i> |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                               |                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Reshape threshold (MB)</b> | If the cached dataframe is smaller than this threshold (in Megabytes) AND if it has more columns than the threshold specified above, the dataframe is reshaped for calculations (wide-to-long reshape instead of calculations per column). The default is 10000 MB. <i>[Not applicable if the dataframe size is below the Pandas threshold specified above.]</i> |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Logging and Command Line Tools

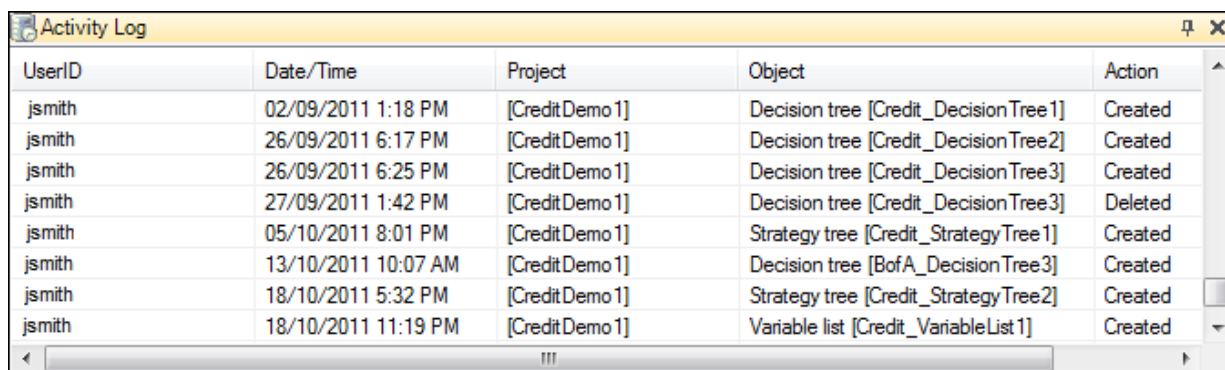
### Activity Log

Activity Log is a history of your activities which is stored with your projects. It provides a permanent record of major events in the current project (such as creating trees, predictive models, importing data, scoring or validation). It is a convenient way of tracking the changes you made in the project over time.

Page | 615

To view the Activity Log, select **View | Activity Log** from the application menu.

The Activity Log grid opens in the lower part of the program window. For any event, it displays the author's user ID, date and time, name and type of object created, and type of action.



| UserID | Date/Time           | Project       | Object                               | Action  |
|--------|---------------------|---------------|--------------------------------------|---------|
| jsmith | 02/09/2011 1:18 PM  | [CreditDemo1] | Decision tree [Credit_DecisionTree1] | Created |
| jsmith | 26/09/2011 6:17 PM  | [CreditDemo1] | Decision tree [Credit_DecisionTree2] | Created |
| jsmith | 26/09/2011 6:25 PM  | [CreditDemo1] | Decision tree [Credit_DecisionTree3] | Created |
| jsmith | 27/09/2011 1:42 PM  | [CreditDemo1] | Decision tree [Credit_DecisionTree3] | Deleted |
| jsmith | 05/10/2011 8:01 PM  | [CreditDemo1] | Strategy tree [Credit_StrategyTree1] | Created |
| jsmith | 13/10/2011 10:07 AM | [CreditDemo1] | Decision tree [BofA_DecisionTree3]   | Created |
| jsmith | 18/10/2011 5:32 PM  | [CreditDemo1] | Strategy tree [Credit_StrategyTree2] | Created |
| jsmith | 18/10/2011 11:19 PM | [CreditDemo1] | Variable list [Credit_VariableList1] | Created |

### Activity Log Window Properties

The Activity Log window is dockable. You can detach it and relocate within the application window by clicking its title bar and dragging to the desired place. To close the Activity Log, click the 'x' button in its top right corner or select **View | Activity Log** again.

You can make the Activity Log auto-hide when it is not active by right-clicking on its title bar and selecting Auto Hide from the context menu.

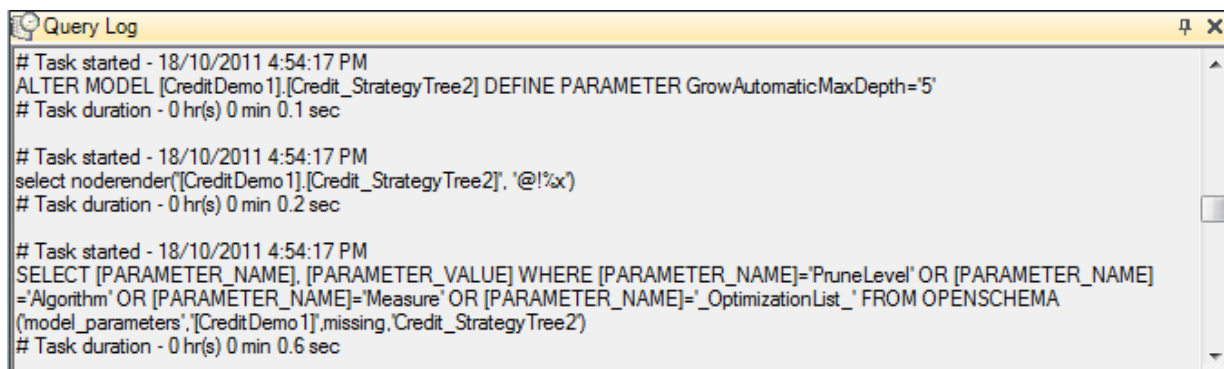
### Query Log

Query Log is a tool for advanced users that provides a record of your actions represented in the form of SQL-like commands sent to the Altair Data Mining Engine from the user interface when your data mining tasks are being performed.

The commands displayed in the Query Log are expressed in the **Data Mining Extensions query language (DMX)**, which is an extension of SQL for Data Mining.

To open the Query Log, select **View | Query Log** from the menu.

In the following example, the second command in the Query Log represents the Automatic Grow operation performed on the tree Credit\_P1\_Decision Tree1 based on the CreditRisk dataset.



```
# Task started - 18/10/2011 4:54:17 PM
ALTER MODEL [CreditDemo1].[Credit_StrategyTree2] DEFINE PARAMETER GrowAutomaticMaxDepth='5'
# Task duration - 0 hr(s) 0 min 0.1 sec

# Task started - 18/10/2011 4:54:17 PM
select nodereader([CreditDemo1].[Credit_StrategyTree2], '@!%x')
# Task duration - 0 hr(s) 0 min 0.2 sec

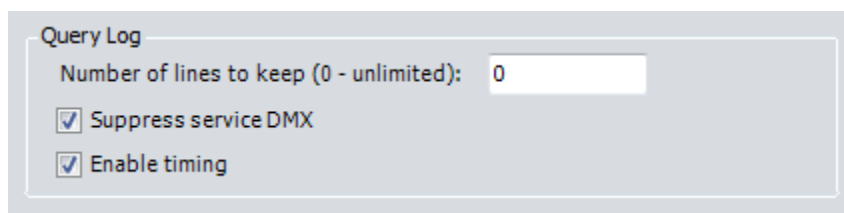
# Task started - 18/10/2011 4:54:17 PM
SELECT [PARAMETER_NAME], [PARAMETER_VALUE] WHERE [PARAMETER_NAME]='PruneLevel' OR [PARAMETER_NAME]
='Algorithm' OR [PARAMETER_NAME]='Measure' OR [PARAMETER_NAME]='_OptimizationList_' FROM OPENSHEMA
(model_parameters,[CreditDemo1],missing,[Credit_StrategyTree2])
# Task duration - 0 hr(s) 0 min 0.6 sec
```

Page | 616

## Query Log Display Options

You can choose the level of detail represented in the Query Log, specify how many recent commands to keep in the log, and choose to display the starting time for each command and the time it takes to execute.

To define the display options for the commands in Query Log, select **Tools | Preferences** from the menu and select the **General** tab.



### ***Number of lines to keep***

Specifies the number of the most recent DMX commands to preserve in the Query Log (the default is 0, which stands for "unlimited", i.e., all commands processed during the current session will be kept in the log).

### ***Suppress service DMX***

Suppresses the auxiliary DMX commands that are not directly related to the data mining tasks being performed (for example, commands that update the project hierarchy, or commands that create temporary objects).

### ***Enable timing***

Displays the starting time and duration of each DMX command.

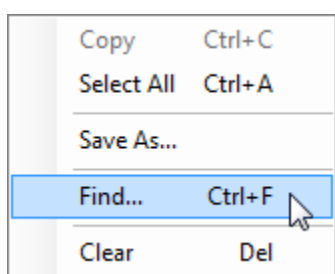
## Query Log Window Properties

The Query Log window is dockable. You can detach it and relocate within the application window by clicking its title bar and dragging to the desired place. To close the Query Log, click the 'x' button in its top right corner or select **View | Query Log** again.

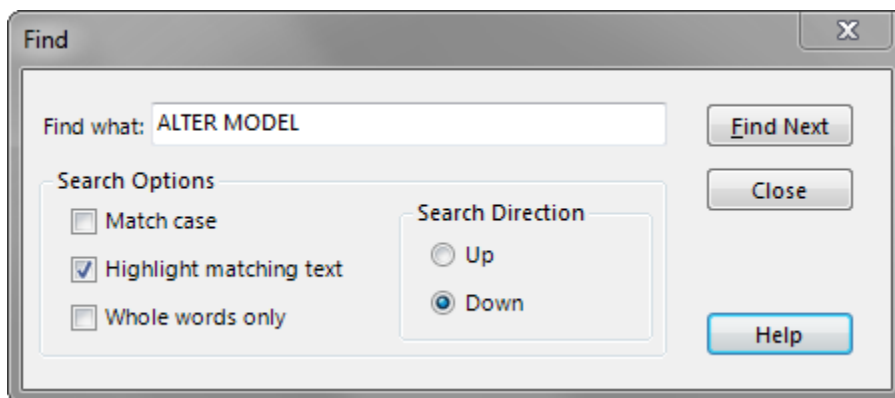
Page | 617

You can make the Query Log auto-hide when it is not active by right-clicking on its title bar and selecting Auto Hide from the context menu.

To find a particular entry in the Query Log, press CTRL F or right-click in the Query Log frame and select **Find**.

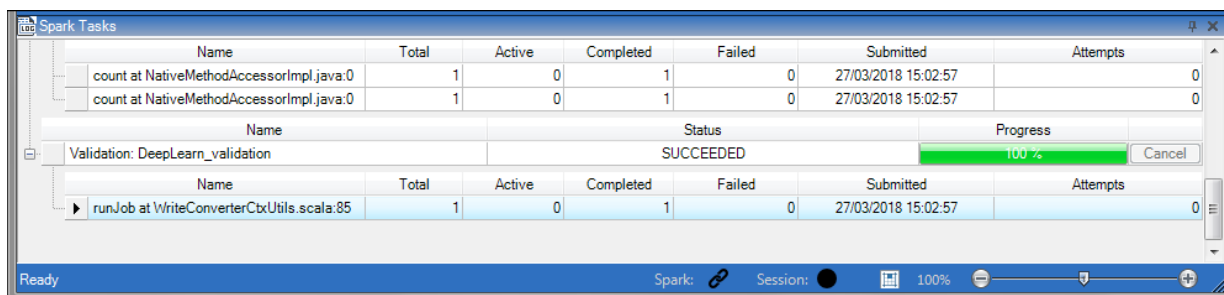


This brings up the Find dialog, where you can perform advanced search for a string.



## Spark Tasks

To check the task progress of the current Spark tasks, select the **Spark Tasks** command from the **View** menu. This opens the Spark Tasks frame in the lower part of the Knowledge Studio application window. It shows the name, status, and progress of your Spark tasks in this session. Here is an example of what the Spark Tasks frame may show during a model validation process:



| Name                                     | Total | Active | Completed | Failed | Submitted           | Attempts |
|------------------------------------------|-------|--------|-----------|--------|---------------------|----------|
| count at NativeMethodAccessorImpl.java:0 | 1     | 0      | 1         | 0      | 27/03/2018 15:02:57 | 0        |
| count at NativeMethodAccessorImpl.java:0 | 1     | 0      | 1         | 0      | 27/03/2018 15:02:57 | 0        |

| Name                             | Status    | Progress |
|----------------------------------|-----------|----------|
| Validation: DeepLearn_validation | SUCCEEDED | 100 %    |

| Name                                      | Total | Active | Completed | Failed | Submitted           | Attempts |
|-------------------------------------------|-------|--------|-----------|--------|---------------------|----------|
| runJob at WriteConverterCtxUtils.scala:85 | 1     | 0      | 1         | 0      | 27/03/2018 15:02:57 | 0        |

The process status is shown only while the task is being performed. The log is cleared once the process is completed.

See also: [Spark Server Connection](#), [Scheduling Analytic Tasks on Spark](#), [Managing Spark Objects](#).

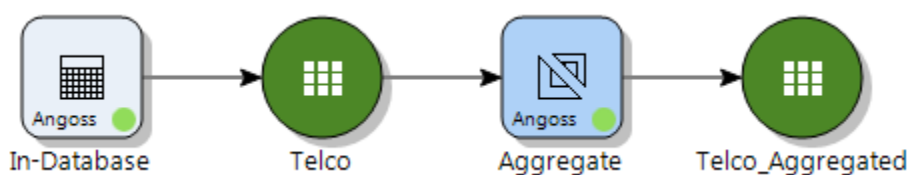
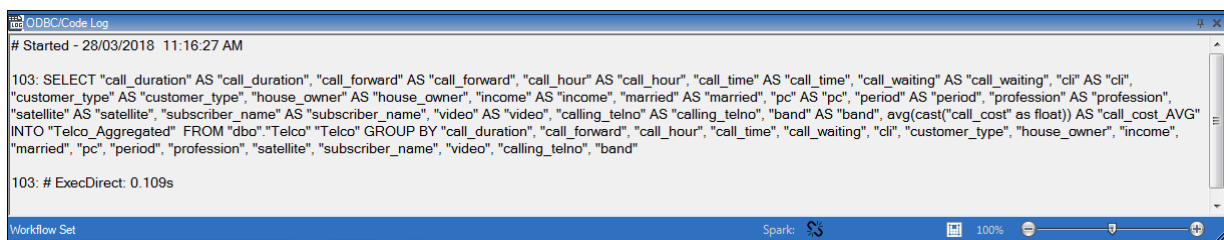
## ODBC/Code Log

ODBC/Code Log is a tool for advanced users that provides a record of your actions when you run In-Database Analytics tasks.

The commands displayed in the ODBC Log are the commands in the SQL dialect of the target database (translated from the [SQL DMX code](#) of the Altair Data Mining Engine).

To open the ODBC Log, select **View | ODBC/Code Log** from the menu.

In the following example, the second command in the log represents the aggregation query that was run in the database.

| Command                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| # Started - 28/03/2018 11:16:27 AM                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 103: SELECT "call_duration" AS "call_duration", "call_forward" AS "call_forward", "call_hour" AS "call_hour", "call_time" AS "call_time", "call_waiting" AS "call_waiting", "cli" AS "cli", "customer_type" AS "customer_type", "house_owner" AS "house_owner", "income" AS "income", "married" AS "married", "pc" AS "pc", "period" AS "period", "profession" AS "profession", "satellite" AS "satellite", "subscriber_name" AS "subscriber_name", "video" AS "video", "calling_telno" AS "calling_telno", "band" AS "band", avg(cast("call_cost" as float)) AS "call_cost_AVG" INTO "Telco_Aggregated" FROM "dbo"."Telco" GROUP BY "call_duration", "call_forward", "call_hour", "call_time", "call_waiting", "cli", "customer_type", "house_owner", "income", "married", "pc", "period", "profession", "satellite", "subscriber_name", "video", "calling_telno", "band" |
| 103: # ExecDirect: 0.109s                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

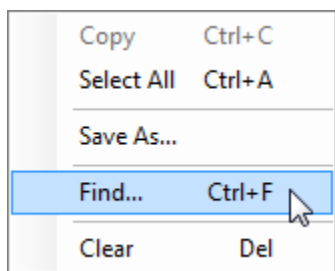
## ODBC/Code Log Window Properties

The ODBC/Code Log window is dockable. You can detach it and relocate within the application window by clicking its title bar and dragging to the desired place. To close the ODBC/Code Log, click the 'x' button in its top right corner or select **View | ODBC/Code Log** again.

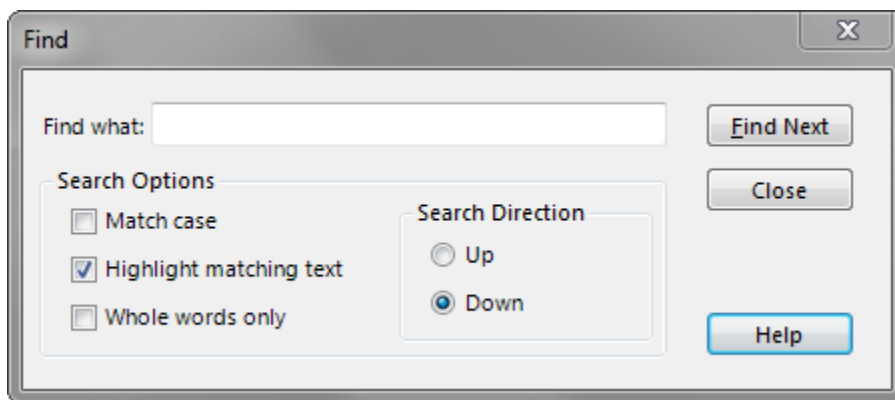
Page | 619

You can make the ODBC/Code Log frame auto-hide when it is not active by right-clicking on its title bar and selecting Auto Hide from the context menu.

To find a particular entry in the log, press CTRL F or right-click in the ODBC/Code Log frame and select **Find**.



This brings up the Find dialog, where you can perform advanced search for a string.



## KSQuery Command Line Interface

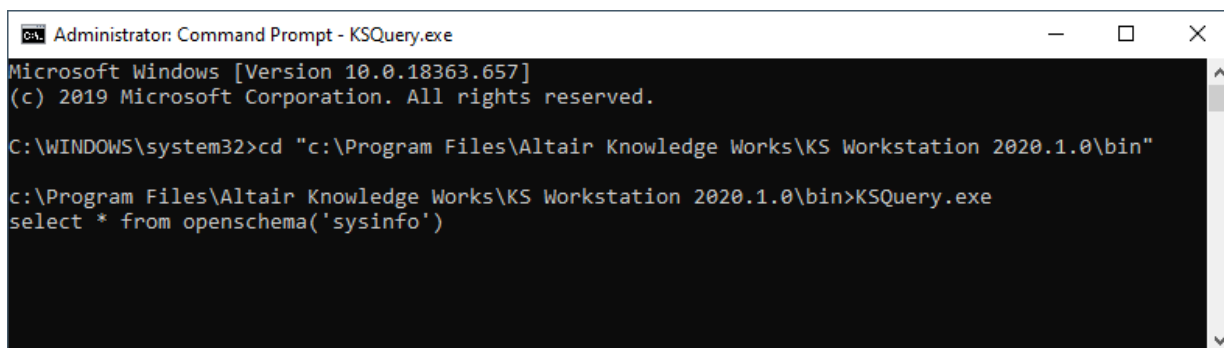
ksquery is a command line executable that allows to you issue data mining commands in the DMX language (SQL with Data Mining Extensions) directly in the Altair Data Mining Engine from the Windows command line or Linux shell.

On Windows, KSQuery.exe is located in the *bin* subdirectory of the application program folder (for example, the default location in version 2022.3 is C:\Program Files\Altair Data Analytics\KS Workstation 2022.3.0\bin). On Linux, ksquery is located in the *lib* subdirectory of the KS Server home (the default is /opt/Altair-Data-Analytics/KS-Server-2022.3/lib).

Ksquery allows you to test the Data Mining Engine without the use of the graphical user interface to diagnose technical issues, get more detailed, or even automate some tasks that for any reason you cannot or would not want to automate via the graphical user interface of the application. Use the command `ksquery --help` for details.

Page | 620

To start ksquery on Windows, run the Command Prompt, change to the *bin* subdirectory of the Altair Data Analytics program folder, and type `ksquery`. The ksquery command prompt opens. The ksquery prompt is an empty line with a cursor. To exit the ksquery prompt, type `:exit` (colon followed by "exit"). Alternatively, press CTRL C on the keyboard, or close the Command Prompt window.



Some examples:

To get the information on the current licenses, type

```
select * from openschema ('licenses')
```

To get the detailed system information, type

```
select * from openschema ('sysinfo')
```

You can specify statements directly after the ksquery command using the switch `--exec` and enclosing your statement in double quotes. For example, the following command lists all objects in the project `Project_CreditScoring`:

```
ksquery --exec "SELECT * FROM [Project_CreditScoring]"
```

You can also copy a sequence of commands in a query file and execute all commands in the file as follows:



```
ksquery --read queries.txt
```

In general, using ksquery you can execute any statements recorded in the [Query Log](#). You can copy any command from the Query Log and paste it into the ksquery prompt. They will be executed, provided that the Working Directory is set properly. To set the working directory in ksquery, use the --init switch with the Mining Location option:

Page | 621

```
ksquery --init "Mining Location=<path to the working
directory>"
```

To get syntax help for KSQuery, use the command

```
KSQuery.exe --help
```

### **General Usage:**

```
ksquery [--xml] [--provider ppp] [--init iii]
        [--read fff] [--exec qqg] [--echo] [--help]
        [{fff|qqg} ...]
```

--xml forces output to be an XML document

--dme enables DME protocol

ppp == name of provider

iii == connect string for provider

fff == file name containing queries

qqg == query

--echo turns on query echo

--help produces basic syntax help

### **KSQuery and License Location**

KSQuery looks for the license location in the file `RegistrationKeys.xml` (which is in the same folder as `KSQuery`). `KSQuery` reads the path specified in the line `<License directory="XXXXXXX"></License>`, then loads the license(s) found there.

The file `RegistrationKeys.xml` is created and updated only once - when `KS Workstation` is installed. If you change the license location via the [License Manager](#) dialog, this does not affect `RegistrationKeys.xml`. If the license is then moved to the new location, `KSQuery` will be unaware of

this unless you manually update the <License directory> entry in `RegistrationKeys.xml`. Also, make sure the license directory specified there has Read permissions for the user who runs KSQuery.

For more details on the DMX and the ksquery syntax, please contact [Technical Support](#).

## Version Compatibility

If more than one product version are installed side by side on the same Windows machine, running KSQuery.exe of the older version may return an error: *"The requested operation is not licensed"*. In that case the name of the license file for the older version has to be explicitly specified in the KSQuery command line within the **--init** argument. Here is an example:

```
ksquery.exe --init "Mining  
Location=C:\Projects;BaseLicenseFileName=xxxxxxx.li  
c" --read query_file.sql
```

where `xxxxxxx.lic` is the license file for the older version. To find out the name of your older license file, select License Manager from the Help menu and scroll to the right in the License Manager window until you see the FILE column.

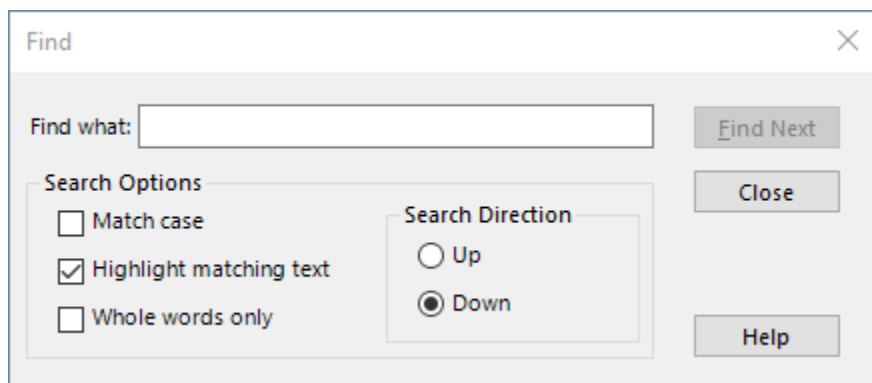
## Find and Replace Text

To find or replace text in various views and frames, use the common keyboard shortcuts **CTRL+F** and **CTRL+H**, respectively.

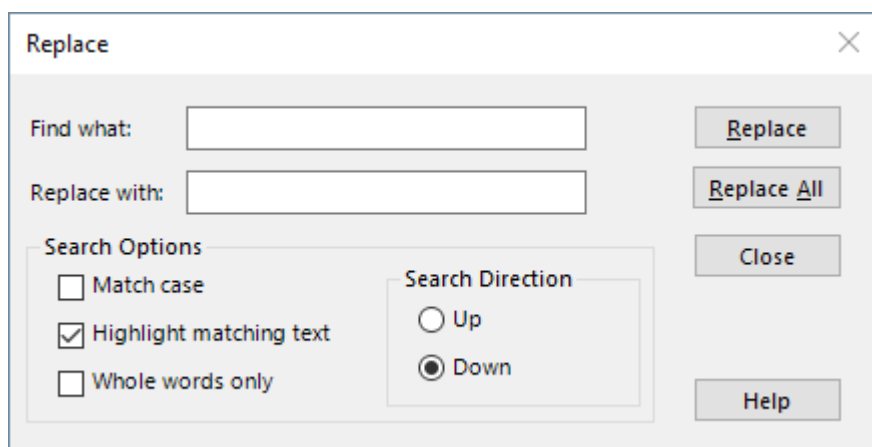
These shortcuts work, for example, in the following dialogs and frames:

- Query Log
- Expression Editor in Variable Transformations

Pressing **CTRL+F** invokes the Find dialog. Enter the text to find and select the desired search options. Then click *Find Next*.



Pressing **CTRL+H** invokes the Replace dialog. It is only applicable to editable frames such as Expression Editor. Enter the text to find, the text to replace it with, and select the desired search options. Then click *Replace* or *Replace All*.



See also: [Field Finder](#)

---

# Printing

## Printing a Tree

To print a tree, select **File | Print** from the menu or press **Ctrl P** on your keyboard when the tree is open.

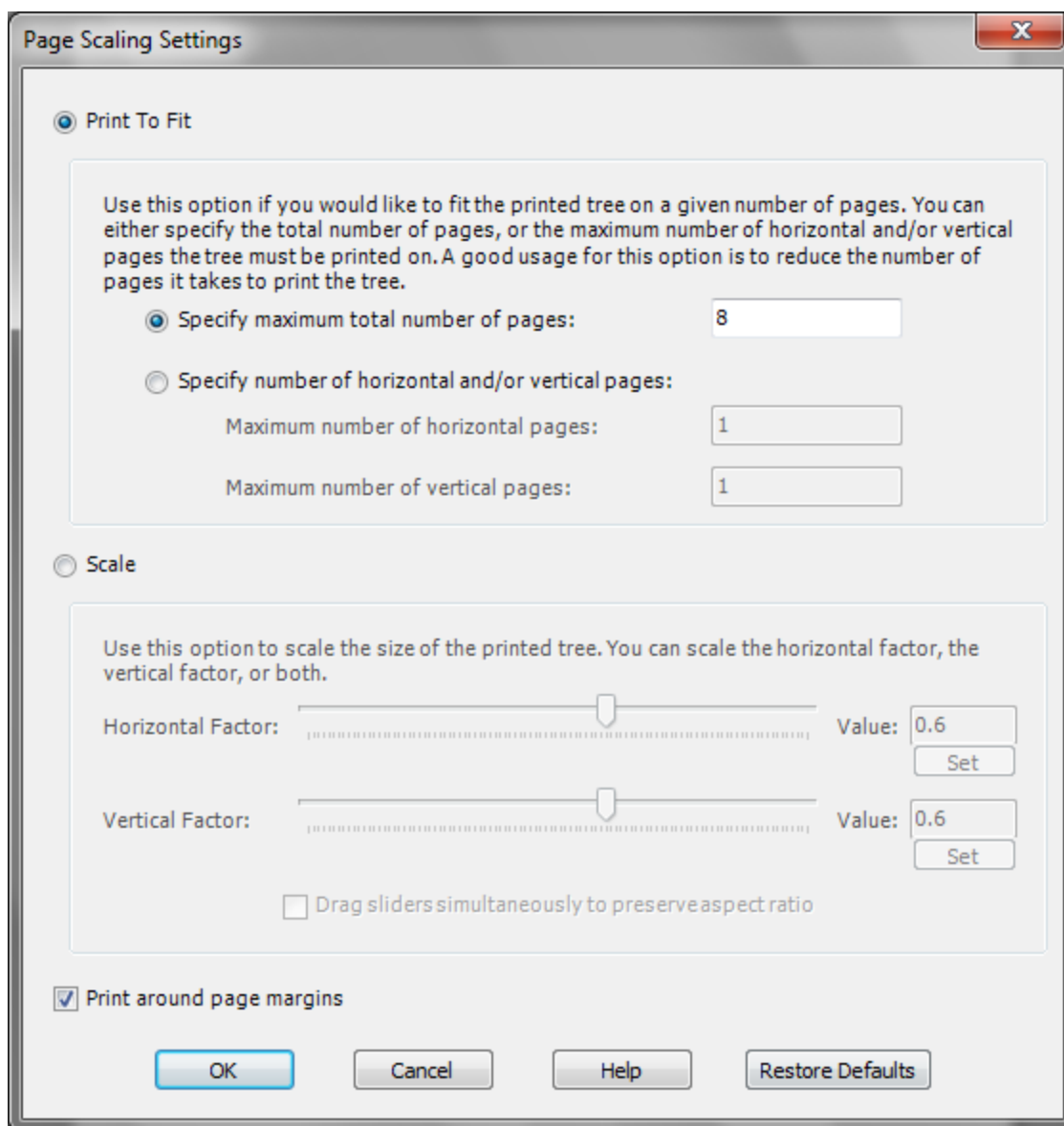
The **File | Print Preview** command provides advanced page scaling options that are especially useful when printing large trees.

The tree printing algorithm is designed to preserve the integrity of tree nodes, i.e., nodes are never split across adjacent pages. A tree that spans over two or more pages in the Print Preview window may not look exactly as it does in the original Tree view, since the length of its branches may be adjusted for printing purposes to prevent the splitting of nodes across adjacent pages.

The last but one page in Tree Preview shows the correct layout of the printed pages - the way you should arrange them to get the full image of the tree.

Open the tree you want to print and select **File | Print Preview** from the menu.

In the Print Preview window, select **Page Scaling...** from the menu.



In the Page Scaling Settings page, you can specify

- the maximum number of pages
- the maximum number of vertical and horizontal pages
- the horizontal and vertical scale factors for the tree
- whether to use page margins

Page Setup (settings for paper size, orientation, and margins) can be accessed either directly by selecting **File | Page Setup...** from the main menu, or from the Print Preview menu.

To print a tree to a PDF file, see the topic [Printing to PDF](#).

## Printing Charts, Grids, and Reports

Any chart, grid, or report in the application can be printed using the menu command **File | Print** or pressing **Ctrl P** on your keyboard.

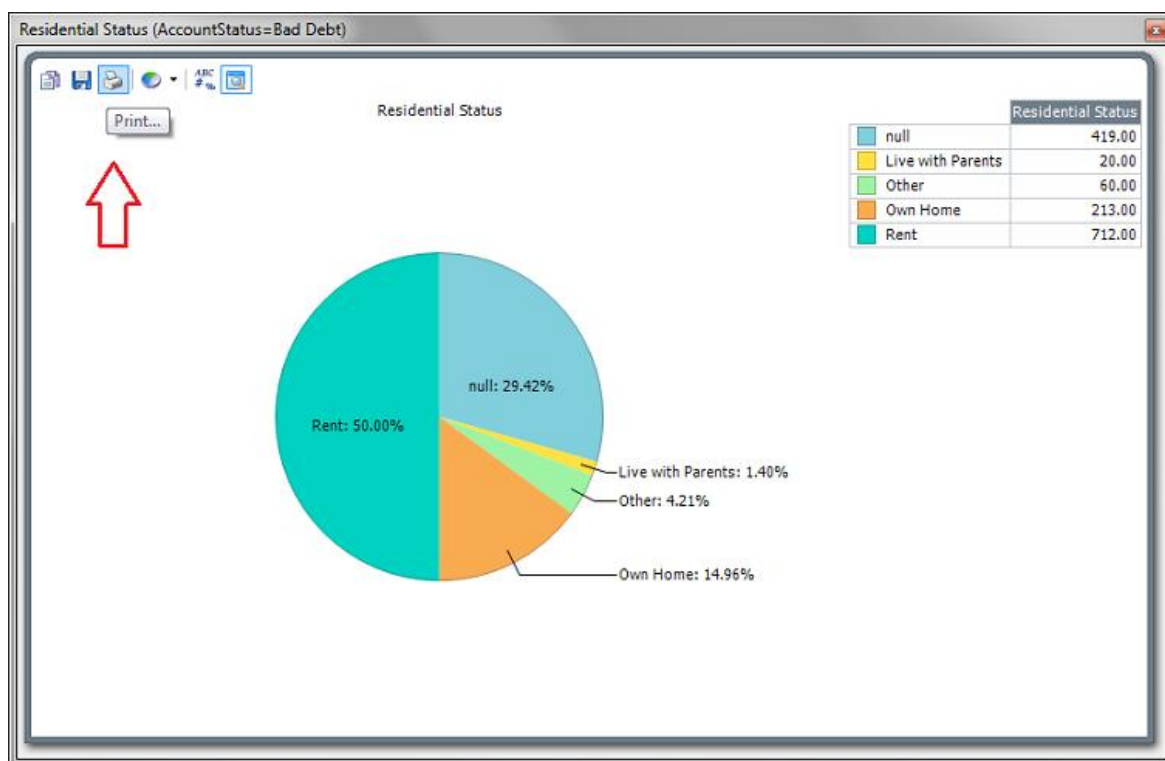
Page | 626

Use the **File | Print Preview** command to preview the object to be printed. As usual, Page Setup (settings for paper size, orientation, margins, etc.) can be accessed either directly by selecting **File | Page Setup...** from the main menu, or from the Print Preview menu.

### Charts:

Whether it is a single chart view (such as Dataset Chart) or a multi-chart view (such as Segment Viewer), the active view will be printed exactly as it is displayed.

An individual chart in a multi-chart view such as Segment Viewer can be printed by double-clicking the chart and then clicking the Print icon in the chart window:



If the chart toolbar is not displayed, right-click anywhere on the background (not on the graph itself) and select the Toolbar option.

---

To print the underlying data of a chart, select the menu command **Chart | Copy To Clipboard | As Text**. Then open Microsoft Excel or another MS Office application, paste the chart data and print from there.

### Reports

Reports in the HTML format such as Model Overview for Regression models, Parameters and Attributes, Validation and Scoring Results are printed as any other HTML pages.

Page | 627

### Grids

Any grid can be printed either directly from the Knowledge Studio application or copied (**Edit | Copy**) and pasted to Microsoft Excel or another MS Office application and printed from there.

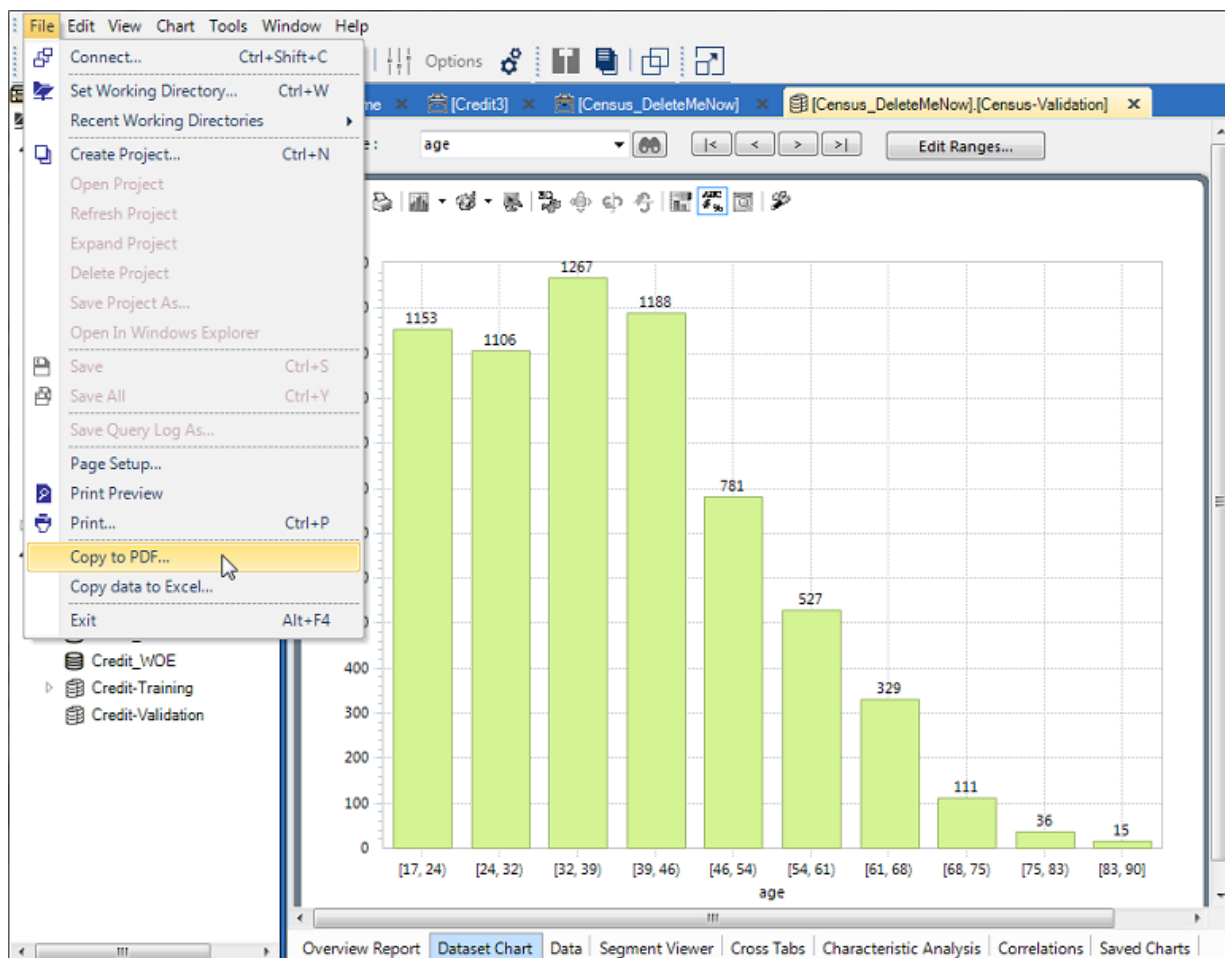
To print a grid, chart, or report to a PDF file, see the topic [Printing to PDF](#).

## Printing to PDF

Any view, including decision or strategy trees, charts, tables and reports can be saved in the PDF format (printed to PDF) using the menu command **File | Copy To PDF**.

Note that the advanced scaling options of the Print Preview feature for decision trees and strategy trees are not applicable when using the Copy to PDF function.

Open the view that you want to save to a PDF file and select **File | Copy To PDF** from the main menu. You will be prompted to specify the output file name and location. Enter the file name and click **Save**.



## Troubleshooting the KS PDF Printer Issues

If the **Copy to PDF** command is not available under the File menu, KS PDF Printer either failed to install or was uninstalled from your computer. To reinstall KS PDF Printer, do the following:

- open the Windows command line prompt as administrator (on your Windows desktop, select **Start --> All Programs --> Accessories**, right-click the Command Prompt item in the Accessories list, and select "Run as administrator").
- change to the directory C:\Program Files\Altair Data Analytics\KS Workstation 2022.3.0\bin
- run **AngossPDFInstall.cmd**

## Version compatibility



If you use KS Workstation 2022.3 side by side with an older version (for example, during an upgrade), you may get the error “*Printer not activated, error code - 41*” when using the command **File | Copy To PDF**. The workaround is to reinstall both versions of the KS PDF Printer as follows:

Run the Command Prompt as administrator and execute these commands in this exact order:

Page | 629

```
cd "C:\Program Files\Altair Data Analytics\KS Workstation
2022.3.0\bin"

Install -s -u "KS PDF Printer"

Install -s "KS PDF Printer"

cd "C:\Program Files\Altair Knowledge Works\KS Workstation
2020.3.0\bin"

Install -s -u "KS PDF Printer"

Install -s "KS PDF Printer"
```

See also:

[Printing Charts, Grids, and Reports](#)

[Printing a Tree](#)

---

# Mathematical Formulations

## Decision Tree

### 1.1 Notation

$Y$  - dependent variable (DV);

$N$  - number of records at the node.

For a given independent variable (IV):

Category - a group of values that a variable can take (ranging from a single value to all possible values);

$r$  - number of disjoint categories;

$\mathcal{V}_i$  -  $i$ -th category;

$n_i$  - number of records at the node with IV in  $\mathcal{V}_i$ .

#### 1.1.1 Numeric DV

$y_{i1}, \dots, y_{in_i}$  - values of DV for records with IV in  $\mathcal{V}_i$ ;

$$y_i = \sum_{j=1}^n y_{ij} ;$$

$$\bar{y}_i = \frac{y_i}{n_i} ;$$

$$y = \sum_{i=1}^r y_i = \sum_{i=1}^r \sum_{j=1}^n y_{ij} ;$$

$$\bar{y} = \frac{y}{N} ;$$

$$STD = \sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y})^2 ;$$

$$SSR = \sum_{i=1}^r n_i (\bar{y}_i - \bar{y})^2 ;$$

$$SSE = \sum_{i=1}^r \sum_{j=1}^n (y_{ij} - \bar{y}_i)^2 ;$$

$$MSTR = \frac{SSR}{r-1} ;$$

$$MSE = \frac{SSE}{N-r} ;$$

$$F^* = \frac{MSTR}{MSE} ;$$

$$PV = F(F^*; r-1, N-r) .$$

If  $i$  and  $k$  are two categories of the IV then:

$$MSTR_{(i,k)} = n_i \left( \bar{y}_i - \frac{y_i + y_k}{n_i + n_k} \right)^2 + n_k \left( \bar{y}_k - \frac{y_i + y_k}{n_i + n_k} \right)^2 ;$$

$$MSE_{(i,k)} = \frac{\sum_{j=1}^{n_i} (y_{ij} - \bar{y}_i)^2 + \sum_{j=1}^{n_k} (y_{kj} - \bar{y}_k)^2}{n_i + n_k - 2} ;$$

$$F_{(i,k)}^* = \frac{MSTR_{(i,k)}}{MSE_{(i,k)}} .$$

### 1.1.2 Categorical DV

$d$  - number of distinct values that DV can take;

$w_j$  -  $j$ th value of DV;

$q_{ij}$  - number of records with DV=  $w_j$  and IV in  $v_i$  ;

$m_j$  - number of records with DV=  $w_j$  ;

$$\bar{q}_{ij} = \frac{n_i}{N} m_j .$$

We have:

$$m_j = \sum_{i=1}^r q_{ij} \quad n_i = \sum_{j=1}^d q_{ij} \quad N = \sum_{j=1}^d m_j = \sum_{i=1}^r n_i = \sum_{i=1}^r \sum_{j=1}^d q_{ij};$$

$$X^2 = \sum_{i=1}^r \sum_{j=1}^d \frac{(q_{ij} - \bar{q}_{ij})^2}{\bar{q}_{ij}};$$

$$X^2_{(i,k)} = \sum_{j=1}^d \frac{(q_{ij} - \bar{q}_{ij})^2}{\bar{q}_{ij}} + \sum_{j=1}^d \frac{(q_{kj} - \bar{q}_{kj})^2}{\bar{q}_{kj}};$$

$$PV = \chi^2(X^2; df) \text{ where } df = (d-1)(r-1);$$

$$E_i = -\frac{1}{\ln d} \sum_{j=1}^d \frac{q_{ij}}{n_i} \ln \frac{q_{ij}}{n_i};$$

$$E = -\frac{1}{\ln d} \sum_{j=1}^d \frac{m_j}{N} \ln \frac{m_j}{N};$$

$$G = 1 - \frac{\sum_{j=1}^d m_j^2}{N^2};$$

$$G_i = 1 - \frac{\sum_{j=1}^d q_{ij}^2}{n_i^2}.$$

## 1.2 Algorithms

### Treatment of a Numeric Independent Variable

To use a numeric IV in a decision tree it must be made categorical. The range of values of the IV is subdivided into a user-controlled number of intervals (set by default to 10) so that the number of records for each interval is the same. Each interval is then considered a category.

### 1.2. Knowledge Seeker

#### Finding a Split for a given IV

The split is found by iteratively grouping categories of the IV into new categories. An eligible pair of categories is grouped into a single category if the two are “similar” with respect to the DV. For

an unordered IV all pairs are eligible. For an ordered IV only adjacent pairs are. There are two grouping methods: exhaustive and cluster.

### Exhaustive Method

This method starts with each value of the IV being a separate category. The following occurs at each iteration:

Page | 634

- For every eligible pair of categories  $(i,k)$  its similarity statistic is computed. For a numeric DV this statistic is  $F_{(i,k)}^*$ . For a categorical DV it is  $X_{(i,k)}^2$ ;
- The pair with the highest similarity statistic is merged into a single category thus decreasing the total number of categories  $r$  by 1. The new set of categories is the next candidate split and its p-value is computed. For a numeric DV the p-value is  $F(F^*; r-1, N-r)$ . For a categorical DV it is  $\chi^2(X^2; df)$ .

These two steps are repeated until  $r = 1$ . Of all candidate splits, the one with the highest *Info* (see below) is chosen. If *Info* is above the user-controlled threshold, the split is presented for that IV. Otherwise no significant splits for the IV can be found.

### Cluster Method

This method also starts with each value of the IV being a separate category. The following occurs at each iteration:

- For every eligible pair of categories  $(i,k)$  its similarity statistic is computed. For a numeric DV this statistic is  $F_{(i,k)}^*$ . For a categorical DV it is  $X_{(i,k)}^2$ ;
- The pair with the highest similarity statistic is merged into a single category thus decreasing the total number of categories  $r$  by 1. The new set of categories is the next candidate split and its p-value is computed. For a numeric DV the p-value is  $F(F^*; r-1, N-r)$ . For a categorical DV it is  $\chi^2(X^2; df)$ .

These two steps are repeated until the p-value of the pair with the highest similarity statistic found in step 1 exceeds the user-controlled threshold specified in the *P-Value* column of the Tree Attribute Editor (accessible from the Tools menu), where it is set by default to 0.05, or until  $r = 1$ . The above p-value is equal to  $F(F_{(i,k)}^*; 1, N-2)$  for a numeric DV and  $\chi^2(X_{(i,k)}^2; d-1)$  for a categorical DV. Of all candidate splits, the one with the highest *Info* is chosen. After this split is found the algorithm attempts to improve the significance by taking single DV values from categories to which they were assigned and grouping them with other categories. If improvement is achieved, the split that produced it becomes the candidate. This re-splitting process returns a possibly different split from

the one that merging produced. If its *Info* is above the user-controlled threshold the split is presented for that IV. Otherwise no significant splits for the IV can be found.

Note: The exact technical details of re-splitting are available on request.

### Measuring Informativeness of splits obtained for each IV

Page | 635

In the first part of the algorithm at most one split has been found for each IV. Informativeness of these splits is computed according to the split measure in effect. Split measures are described below. A split is informative if it passes the user-controlled threshold, set in the Measure tab of the Tree Configuration dialog - Tools menu - or in Step 3 of the Insert Tree wizard in the Maximum Attr. Info and Minimum Attr. Info fields. The user can choose any informative split. The most informative one is chosen by default.

### Growth Control Parameters

The tree growth is also constrained by a number of user-controlled parameters found in the *Tree Growth* tab of the **Options** and **Preferences** dialogs (see Options or Preferences under the **Tools** menu).

The following parameters affect the Find Split and Automatic Grow operations:

Minimum Node Creation Size - Nodes smaller than the specified size will not be created. More precisely: If, at the end of a splitting algorithm, a node in the found split contains fewer records than this number, that node is merged with an adjacent node (for an ordered IV) or with the most statistically similar node (for an unordered IV);

Auto-Grow Stop Size (min. records) - If a node contains fewer records than specified by this parameter, that node will not be split;

Auto-Grow Maximum Tree Depth - If the path from the root node to a given node contains as many nodes as specified by this parameter, that node will not be split. When this parameter is equal to 0 (as it is by default) it means that tree depth is not restricted.

Minimum DV Values - These parameters apply to trees with discrete dependent variables only. They define the minimum threshold values for the frequencies or percentages (if the As Percent option is selected) of the DV values in nodes to be created. Nodes with frequencies (or percentages) of DV values below these thresholds will not be created.

## 1.3 Measures for Splitting a Node

### Split Statistics

When evaluating the significance of a split, Knowledge Studio calculates a number of statistics (apart from record counts and proportions) that can be viewed in the Split Report tab of a decision

tree or displayed directly in the tree via settings in the Node Detail - Edit menu. Which statistics are reported is measure-dependent except *Info* which is given for every measure and DV type (it can only be viewed in the Split Report).

Below we provide the list of statistics with mathematical expressions for each measure and DV type.

### 1.3.1 Statistical Measures

A split is a partition of the node dataset into subsets according to the value of the independent variable. According to a statistical measure the best split is the one that generates subsets which are least likely to come from the same distribution. The likelihood is measured by standard statistical tests.

A split is presented if it passes a predetermined significance level. This level can be chosen in two ways:

1. With an *unadjusted* p-value measure the desired significance level is directly applied to the best split .
2. The *adjusted* measure takes into account the fact that a number of different splits is evaluated for every IV (see *Algorithms* above) and only one of them is required to be significant. This split has to pass a more stringent significance test than if only a single split was attempted. The choice of that stringent test is made within the Bonferroni framework by adjusting the p-value of the split. Exact technical details of this adjustment are beyond the current scope of this document and are available on request.

#### Unadjusted p-value

##### 1. Numeric DV

- $Info = 100 \times C$  ;
- $df1 = N - r$  ;
- $df2 = r - 1$  ;
- $F = F^* = \frac{MSTR}{MSE}$  ;
- $P = PV = F(F^*, df2, df1)$  ;
- $C = 1 - P$  .

##### 2. Categorical DV



- $Info = 100 \times C$ ;
- $df = (d - 1)(r - 1)$ ;
- $Chi2 = X^2 = \sum_{i=1}^r \sum_{j=1}^d \frac{(q_{ij} - \bar{q}_{ij})^2}{\bar{q}_{ij}}$ ;
- $P = PV = \chi^2(X^2, df)$ ;
- $C = 1 - P$ .

## Adjusted p-value

### 1. Numeric DV

- $Info = 100 \times C$ ;
- $df1 = N - r$ ;
- $df2 = r - 1$ ;
- $F = F^* = \frac{MSTR}{MSE}$ ;
- $uP = PV = F(F^*, df2, df1)$ ;
- $uC = 1 - uP$ ;
- $P = B \times uP$ ;
- $C = 1 - P$ ;
- $B = B_a B_m$ ;

•  $B_m$  is the additional adjustment factor that the user can specify in the Measure tab of the Tree Configuration dialog when adjusted measure is chosen. By default it is set to 1;

•  $B_a$  is the Bonferroni adjustment factor determined automatically by the algorithm (see above).

Note:  $B_a$  depends on the number of distinct values of IV( $r$ ), the number of categories formed by the algorithm (Cluster or Exhaustive), the algorithm itself, and the grouping type of IV.  $B_a$  is greater for unordered than for ordered IV, for Cluster than for Exhaustive and it increases with the cardinality of IV.

### 2. Categorical DV

- $Info = 100 \times C$ ;
- $df = (d - 1)(r - 1)$ ;
- $Chi2 = X^2 = \sum_{i=1}^r \sum_{j=1}^d \frac{(q_{ij} - \bar{q}_{ij})^2}{\bar{q}_{ij}}$ ;
- $uP = PV = \chi^2(X^2; df)$ ;
- $uC = 1 - uP$ ;
- $P = B \times uP$ ;
- $C = 1 - P$ ;
- $B = B_a B_m$ ;

- $B_m$  is the additional adjustment factor that the user can specify in the Measure tab of the Tree Configuration dialog when adjusted measure is chosen. By default it is set to 1;
- $B_a$  is the Bonferroni adjustment factor determined automatically by the algorithm (see above).

Note:  $B_a$  depends on the number of distinct values of IV( $r$ ), the number of categories formed by the algorithm (Cluster or Exhaustive), the algorithm itself, and the grouping type of IV.  $B_a$  is greater for unordered than for ordered IV, for Cluster than for Exhaustive and it increases with the cardinality of IV.

### 1.3.2 Non-statistical Measures

#### Entropy Variance

This quantity tells how much information about a DV value is contained in a random draw from the node dataset. If DV values were spread evenly within that set then, by drawing a record at random, one would gain no information because every DV value would be equally likely. If, on the contrary, all records had the same DV value, a random draw would provide perfect information about it. High entropy means low information and vice versa. Entropy E achieves a maximum of 1 for a uniform (evenly spread) distribution and a minimum of 0 for a degenerate (peaked) distribution.

##### 1. Numeric DV

- $Info = 100 \times RatioVariance$  ;
- $InputVariance = STD$  ;
- $OutputVariance = SSE$  ;
- $RatioVariance = 1 - \frac{OutputVariance}{InputVariance}$  ;

## 2. Categorical DV

- $Info = 100 \times RatioEntropy$  ;
- $InputEntropy = E$  ;
- $OutputEntropy = \sum_{i=1}^r \frac{n_i}{N} E_i$  ;
- $RatioEntropy = 1 - \frac{OutputEntropy}{InputEntropy}$  ;

## Gini Variance

Gini variance is analogous to entropy variance in that it also measures how evenly (or unevenly) DV values are spread within a dataset.

### 1. Numeric DV

- $Info = 100 \times RatioVariance$  ;
- $InputVariance = STD$  ;
- $OutputVariance = SSE$  ;
- $RatioVariance = 1 - \frac{OutputVariance}{InputVariance}$  ;

### 2. Categorical DV

- $Info = 100 \times RatioIndex$  ;
- $InputIndex = G$  ;
- $OutputIndex = \sum_{i=1}^r \frac{n_i}{N} G_i$  ;
- $RatioIndex = 1 - \frac{OutputIndex}{InputIndex}$  .

## 1.4 Thresholds for Informativeness of Splits

In the description of algorithms above it was mentioned that only informative splits (i.e. those with *Info* passing the threshold) are presented. This threshold is set differently for different measure types.

### 1.4.1 Statistical Measures

For statistical measures the *Info* threshold is derived from the significance level that the user sets in the Measure tab of the Tree Configuration dialog - Tools menu. Splits for which P is below that significance level (and thus *Info* is above the derived threshold) are presented. The default value of the significance level is 0.05.

### 1.4.2 Non-statistical Measures

For non-statistical measures the user can set maximum and minimum thresholds for *Info*. This can be done in Step 3 of the Insert Tree wizard in the Maximum Attr. Info and Minimum Attr. Info fields. Splits with *Info* value between these two numbers are highlighted even though all splits with positive *Info* are presented.

## 1.5 Tree Pruning

Tree Pruning allows one to simplify a given tree model by reducing the number of nodes on the tree while maintaining the lowest misclassification rate possible.

### 1.5.1 Error-Complexity Pruning

This algorithm only applies to trees with a discrete DV.

The Error-Complexity Pruning algorithm follows these steps:

1. Build a complete tree.

Calculate  $\alpha$  for each subtree (except the first) and select the sub-tree with the smallest value of  $\alpha$  for pruning. Repeat this process until there are no sub-trees left, to yield a series of increasingly pruned trees.

$$\alpha = \frac{R(t) - R(T_t)}{N_T - 1}$$

Page | 641

where  $t$  is a node,  $T$  is a sub-tree,  $N_T$  is the number of nodes for sub-tree  $T$ ,  $R(T_t)$  is error cost if a sub-tree is kept, and  $R(t)$  is the error cost if the sub-tree becomes a leaf through pruning,

$$R(T_t) = \sum R(i) \text{ for } i = \text{sub-tree leaves,}$$

and

$$R(i) = r(i)p(i)$$

$r(i)$  = Number of wrongly classified records in node  $i$  / Total population of node  $i$ ,

$p(i)$  = Total population of node  $i$  / Total population of the tree;

$$R(t) = r(t)p(t);$$

2. Run a test dataset through the sub-tree with the smallest  $\alpha$ , to get the misclassification rate. Repeat this process until you get the tree with the smallest misclassification rate.

### 1.5.2 Critical Value Pruning

The method estimates the important or strength of a node from calculations done in the tree creation stage.

1. Calculate the critical values for each node. Which critical value is used depends on the algorithm of decision tree.
2. Rank the critical values. The larger the critical value selected, the greater the degree of pruning and the smaller the resulting tree.
3. Validate the tree when it has been pruned, and then calculate the misclassification.
4. Repeat steps 2 and 3 steps until the misclassification rate stops decreasing.

### 1.5.3 Reduced-Error Pruning

This algorithm only applies to trees with a discrete DV.

This pruning method produces a series of pruned trees by using the test data directly. It ends with the smallest minimum-error tree on the test data.

Page | 642

1. Start with a complete tree and run the test data through it.
2. Count the error cost if a sub-tree is kept ( $R(T_t)$ ) and the error cost if the sub-tree becomes a leaf through pruning ( $R(t)$ ), where  $t$  is a node, and  $T$  is a sub-tree,

$$R(T_t) = \sum R(i) \quad \text{for } i = \text{sub-tree leaves, and}$$

$$R(i) = r(i)p(i),$$

$r(i)$  = Number of wrongly classified records of node  $i$  / Total population of node  $i$ ,

$p(i)$  = Total population of node  $i$  / Total population of the tree;

$$R(t) = r(t)p(t);$$

3. Use the difference between the error costs (if positive) as a measure of the gain from pruning the sub-tree, i.e.,

$$\text{A Measure} = R(T_t) - R(t);$$

4. Choose the one with largest difference as the sub-tree to prune;
5. Continue this process until further pruning would increase the misclassification rate of whole tree,  $R$ ,  
 $R = \text{Misclassified Rows} / \text{Total Rows}.$

---

## References

Breiman, L., Friedman, J., Stone, C. J., and Olshen R. A. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC.

Biggs, D., de Ville, B., and Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. *Journal of Applied Statistics*, **18**, 49-62.

Calinski, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, **3**, 1-27.

Dunn, O. J. (1961). Multiple comparisons among means. *J. Amer. Statist. Assoc.*, **56**, 54-64.

Efron, B. (1983). Estimating the error rate of a prediction rule. *J. Amer. Statist. Assoc.*, **78**, 316-333.

Feller, W. (1968). *An Introduction to Probability Theory and its Applications*. J. Wiley & Sons, New York.

Hawkins, D. M. and Kass, G. V. (1982). Automatic interaction detection. In: *Topics in Applied Multivariate Analysis*, Cambridge University Press.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, **29**, 119-127.

Kass, G. V. (1976). Significance testing in automatic interaction detection. *Applied Statistics*, **24**(2), 178-189.

Lattin, J. M., Carroll, J. D., and Green, P. E. (2003). *Analyzing Multivariate Data*. Brooks/Cole.

Michie, D. (1991). Methodologies from machine learning in data analysis and software. *The Computer Journal*, **34**(6), 559-565.

Miller, R. G. (1981). *Simultaneous Statistical Inference*. Springer, New York.

Quinlan, R. (1979). Discovering rules by induction from large collections of examples. In: *Expert Systems in the Micro-Electronic Age*, 168-201, Edinburgh.

Sonquist, J. A., Baker, E. and Morgan, J. (1973). *Searching For Structure*. Institute for Social Research, University of Michigan, Ann Arbor, MI.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika* **52** (3-4), 591-611. JSTOR 2333709. MR 205384.

---

de Ville, B. (1990). Applying statistical knowledge to database analysis and knowledge base construction. In: *Proceedings of the 6th IEEE Conference on Artificial Intelligence Applications*, IEEE Computer Society, Washington.



---

# Index

## A

accuracy 386

Attribute editor

attribute editor dialog 374, 456

break-apart 374, 456

cardinality 374, 456

grouping type 382

include 374, 456

intervals and bins 383

missing values 374, 456

ordered display 374, 456

significance 374, 456

usage 374, 456

weights 379

Automatic Grow 309

## C

Charts

profile chart 352

Client-server mode 48

Connection 48

constraint 437

Copy

data 373

screen 373

splits 366

tree 371

Correlations 262

Covariance 262

Cross tabs 248

Cumulative lift chart 544

about 544

calculating the cumulative lift 544

## D

Data profiling 226

Dataset

about 226

characteristic analysis 257

chart 237

correlations 262

covariance 262

cross tabs 248

data 240

overview report 230

segment viewer 242

**Decision tree**

about 285

Automatic Grow 309

inserting 288

measures 302

**E****Edit**

find and replace text 620

Edit split 320

Email options 608

Erase splits 319

Expression editor

function syntax 178

helpers (strategy tree) 419

Expressions

syntax 178

**F**

Field Finder 574

Find split 305

Force split 311

Format

Reports 584

**G**

Go to split 329

**I**

Ignore variable 318

Intervals Type 374, 456

**L**

License information

[acquire by email](#) 44

Altair Units Licensing 27

Managed Altair Units 27

trial license 17

**M**

Measure 302

misclassification rate 386

**N**

Next Split 328

Node

data 344

dataset 384

report 346

Node calculation

about 411

expression editor 414

insert 412

Node dataset

about 384

---

---

|                                     |                     |
|-------------------------------------|---------------------|
| inserting 384                       | Spark               |
| notifications 608                   | model link 103, 521 |
| <b>O</b>                            | Spark dataframe     |
| Optimization 437                    | filtering join 130  |
| solution constraint 437             | Splits              |
| Ordered Display 374, 456            | about 286           |
| Overview Report 230, 236, 586       | copy 366            |
| <b>P</b>                            | edit 320            |
| Paste                               | erase 319           |
| splits 366                          | find 305            |
| Predictive model                    | force 311           |
| model instance 282                  | go to 329           |
| Previous Split 329                  | ignore 318          |
| Profile chart 352                   | next 328            |
| Projects                            | paste 366           |
| about 60                            | previous 329        |
| <b><u>creating a project</u></b> 63 | report 345          |
| <b>R</b>                            | Standalone mode 48  |
| relative error 386                  | Strategy tree       |
| resubstitution 386                  | about 393           |
| <b>S</b>                            | inserting 395       |
| Score                               | score 479           |
| model 547                           | treatment 440       |
| strategy 479                        | validate 466        |

---

validation tree 475

## **T**

Technical information

copyright 11

system requirements 6

Technical support 10

Treatment 440

Tree views

node data 344

node report 346

profile chart 352

split report 345

tree map 343

tree view 330

Trees

strategy trees 393

## **U**

Usage 374, 456

## **V**

Validate

model 525

strategy 466

Variable list 566

export 577

import 574

## **W**

Weight 227, 374, 456

frequency 379

sample 379

Workflow 64

working directory 50, 60