



SAS[®] Enterprise Miner[™]: Tutorials and Examples

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2017. *SAS® Enterprise Miner™: Tutorials and Examples*. Cary, NC: SAS Institute Inc.

SAS® Enterprise Miner™: Tutorials and Examples

Copyright © 2017, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

February 2018

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

14.2-P1:emex

Contents

Chapter 1 • About This Documentation	1
Other Resources	1
Chapter 2 • Initial Steps	3
Create a Project	3
Define a Data Source	5
Create a Process Flow Diagram	16
Chapter 3 • Sample Data	23
Sample Nodes: Overview	23
Partition Data	24
Filter Data	30
Chapter 4 • Explore Data	41
Explore Nodes: Overview	41
Chapter 5 • Modify Input Data	43
Modify Nodes: Overview	43
Transform Variables	44
Chapter 6 • Model Data	65
Model Nodes: Overview	65
Chapter 7 • Assess Model Performance	67
Assess Nodes: Overview	67
Chapter 8 • Utilities	69
Utility Nodes: Overview	70
Start Groups	71
End Groups	81
Metadata	85
SAS Code	90
Score Code Export	118
Chapter 9 • Applications	131
Survival Analysis	131
Time Series	169
Credit Scoring	170
Incremental Response	170
Chapter 10 • Advanced Topics	189
High-Performance Nodes: Overview	189
Data Source Roles	190

Chapter 1

About This Documentation

Other Resources	1
------------------------------	----------

Other Resources

The documentation included in Tutorials and Examples highlights some of the features of SAS Enterprise Miner. For more details about these tasks, or to access all of our documentation, see [SAS Enterprise Miner Documentation](#).

Chapter 2

Initial Steps

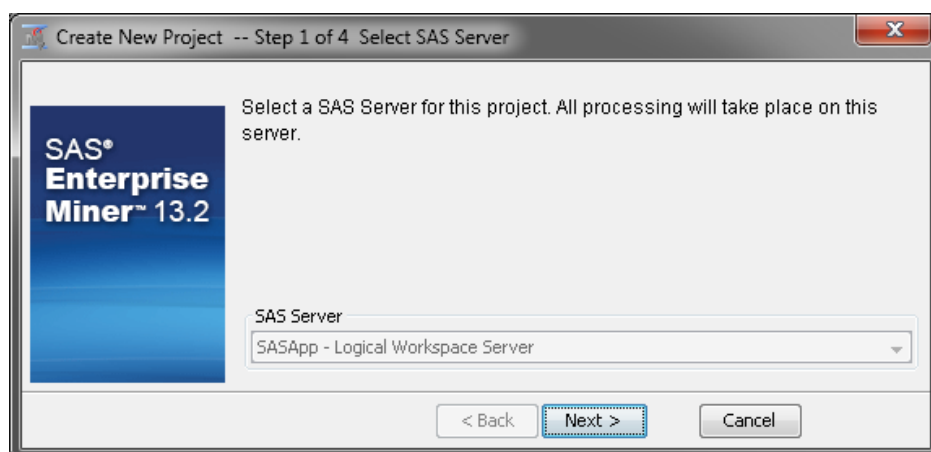
Create a Project	3
Define a Data Source	5
Create a Process Flow Diagram	16
Add a Node	16
Variables Table	16
Connect Nodes	17
Select Nodes	18
Disconnect Nodes	19
Delete Single Nodes	20
Delete Multiple Nodes	21
Move or Reposition Nodes	21
Copy and Paste Nodes	21

Create a Project

To create a new project, follow these steps:

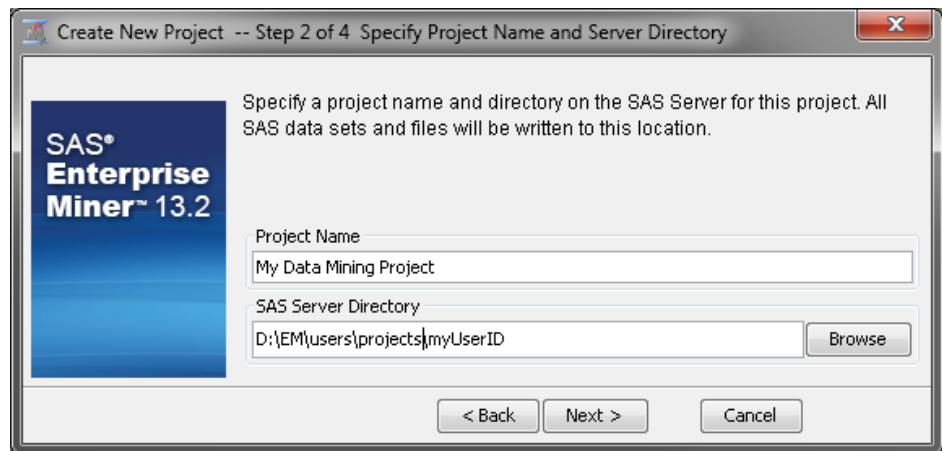
1. From the main menu, select **File** ⇒ **New** ⇒ **Project**.

The Create New Project window opens:



Select an available server for your project. Click **Next**.

2. Enter the name of your project in the **Project Name** field, and specify the server directory in the **SAS Server Directory** field.



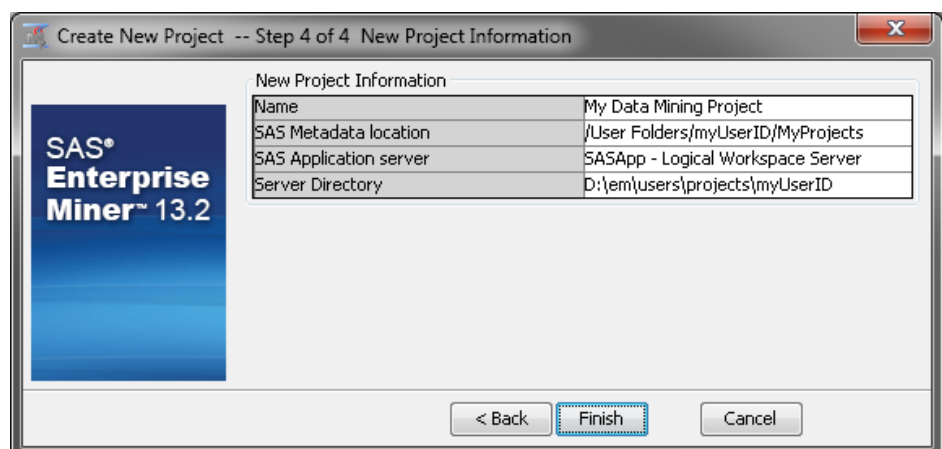
Click **Next**.

3. Use the **Browse** button to select the location for your project's SAS Folders on the server.



Click **Next**.

4. Review the project information you have just specified in the **New Project Information** table. Use the **Back** button if you need to make changes. When you are ready to create your SAS Enterprise Miner project, click **Finish**.

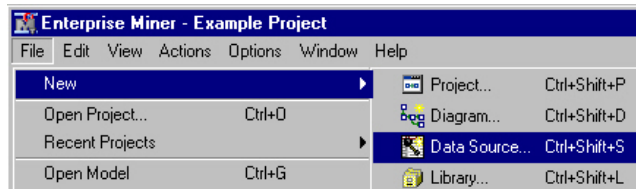


Define a Data Source

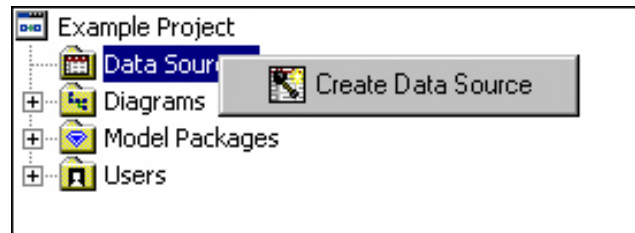
Follow these steps to use the **Data Source** wizard to create a data source.

1. Use one of the following methods to open the wizard:

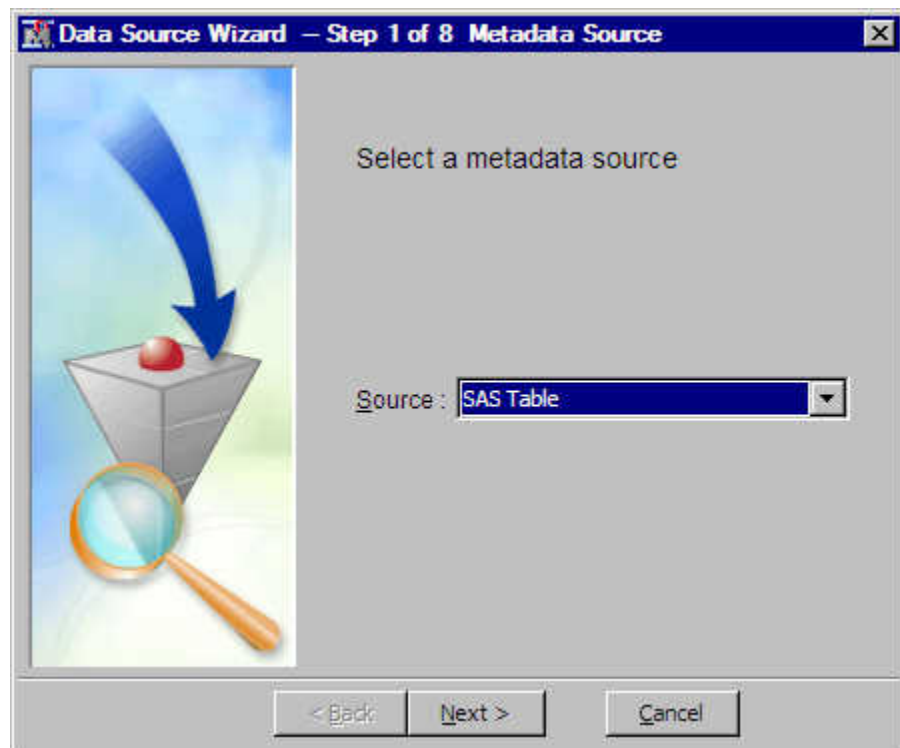
- Select **File** ⇒ **New** ⇒ **Data Source** from the SAS Enterprise Miner main menu.



- Right-click the Data Sources folder in the Project Panel and select **Create Data Source**.



2. In the Data Source Wizard — Metadata Source window, select the source of data that you want to access and click **Next**.



You can select from the following sources:

- **SAS Table** — This is an SAS library engine format table. The library must be created before the table is selected.

Note: If you have SAS tables only, you do not have to pre-assign libraries. The libraries are automatically available as a **Metadata Repository** from the Create Data Source wizard. If you have database or RDBM libraries (such as Oracle), you must pre-assign RDBMS libraries, and they are then available as a **SAS Table** in the Create Data Source wizard. If you have both SAS and RDBMS tables, you must choose the **Library is pre-assigned** option for both types of tables. With this option, all of the tables are made available via **SAS Table** in the Create Data Source wizard.

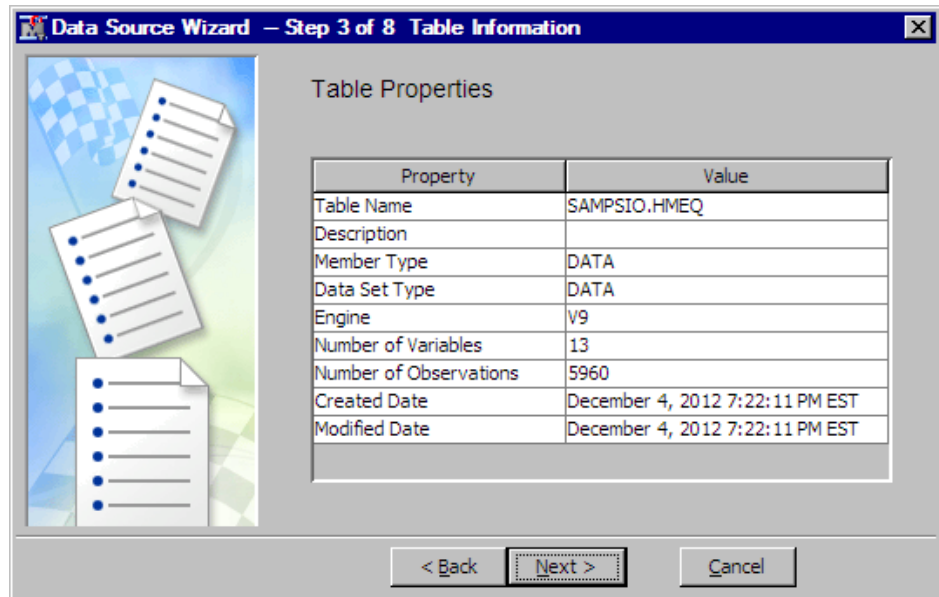
Password-protected tables are not valid for creating data sources.

3. If you select **SAS Table** as the source, the Data Source Wizard — Select a SAS Table window appears. Select a SAS data table by entering the data set name or clicking **Browse** to select from a list. Then, click **Next**.

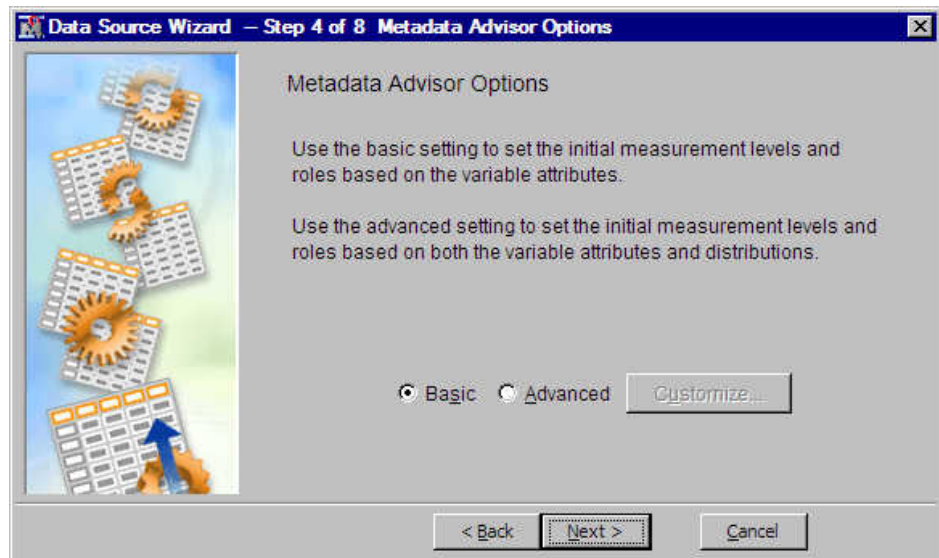


If you select **Metadata Repository** as the source, the Data Source Wizard — Import Metadata from Metadata Repository window appears. Click **Browse** to select an item from the list of registered tables.

4. In the Data Source Wizard – Table Information window, you can view the table attributes including the table name, the number of variables, and the number of observations. Then, click **Next**.



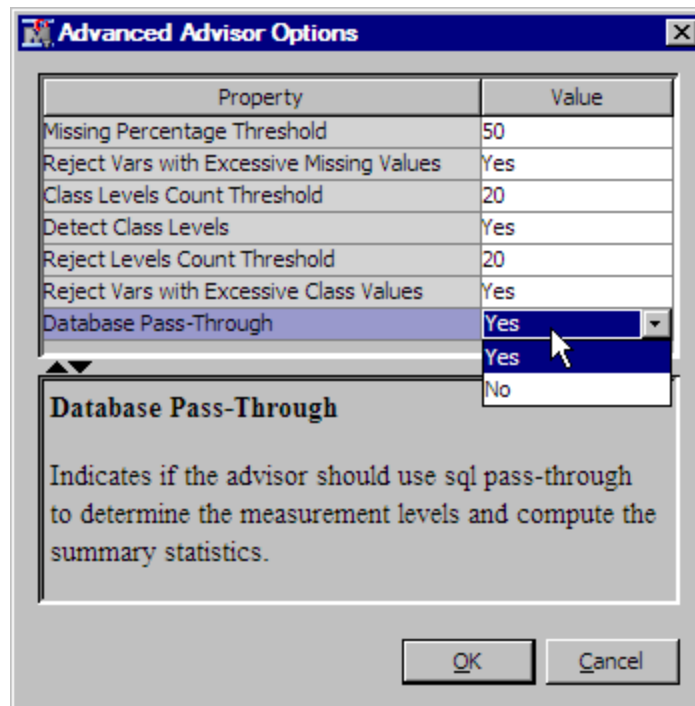
5. The Data Source Wizard – Metadata Advisor Options window enables you to select the type of advisor option used to create column attributes. Column attributes are also known as metadata.



Two options are available.

- **Basic** — Use the **Basic** option when you already know the variable roles and measurement levels. The initial role and level are based on the variable type and format values.
- **Advanced** — Use the **Advanced** option when you want SAS Enterprise Miner to automatically set the variable roles and measurement levels. Automatic initial roles and level values are based on the variable type, the variable format, and the number of distinct values contained in the variable. If this is selected, you will be able to display statistics in the Data Source Wizard – Column Metadata step. The Database pass-through option enables you to compute summary statistics in the database rather than in SAS Enterprise Miner.

If you select **Basic**, click **Next** to proceed. If you select **Advanced**, click **Customize** to view details of the options:

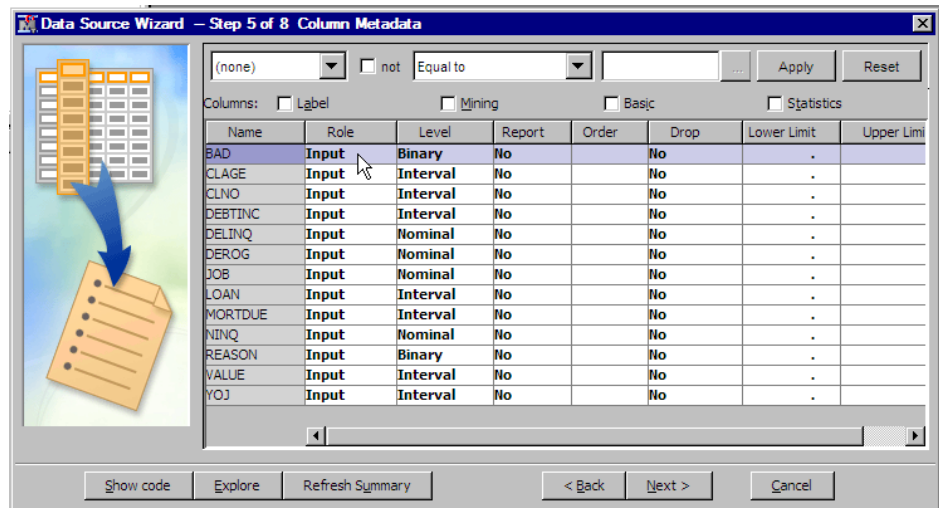


To customize the advanced options, enter a value or use the drop-down list in the Value column to change it. Click **OK** in the Advanced Advisor Options window to save your changes. Then, select **Next**.

The following options are available in the Advanced Options window:

- **Detect Class Levels** — specifies whether the number of class levels is determined for each variable.
- **Class Levels Count Threshold** — specifies the maximum number of class levels for each variable. When the **Detect Class Levels** property is set to **Yes**, if there are more class levels than the value specified here, the variable is considered an interval variable. Valid values are positive integers greater than or equal to 2.
- **Reject Vars with Excessive Missing Values** — specifies whether variables with a large percentage of missing values should be rejected.
- **Missing Percentage Threshold** — specifies the maximum percentage of missing values allowed for each variable. When the **Reject Vars with Excessive Missing Values** property is set to **Yes**, a variable is rejected when the percentage of missing values is greater than the value specified here. Valid values are integers between 0 and 100, inclusive.
- **Reject Vars with Excessive Class Values** — specifies whether variables with a large number of class variable levels should be rejected.
- **Reject Levels Count Threshold** — specifies the maximum number of class variable levels allowed for each variable. When the **Reject Vars with Excessive Class Values** property is **Yes**, class variables with more levels than the value specified here are rejected.
- **Identify Empty Columns** — specifies whether empty columns are assigned the value of MISSING.
- **Database Pass-Through** — specifies if the data source advisor should use in-database processing to calculate summary statistics and determine measurement levels.

6. The Data Source Wizard – Column Metadata window displays the default attributes that are defined for each variable.



On top of the table is a configurable WHERE clause filter that is helpful when configuring a long list of variables. To view extra columns, select the **Label**, **Mining**, **Basic**, or **Statistics** check boxes.

The following columns are displayed if **Mining** is checked:

- Creator
- Comment
- Format Type

The following columns are displayed if **Basic** is checked:

- Type
- Format
- Informat
- Length

To enable the calculation of summary statistics, check **Statistics**.

The following new columns are displayed:

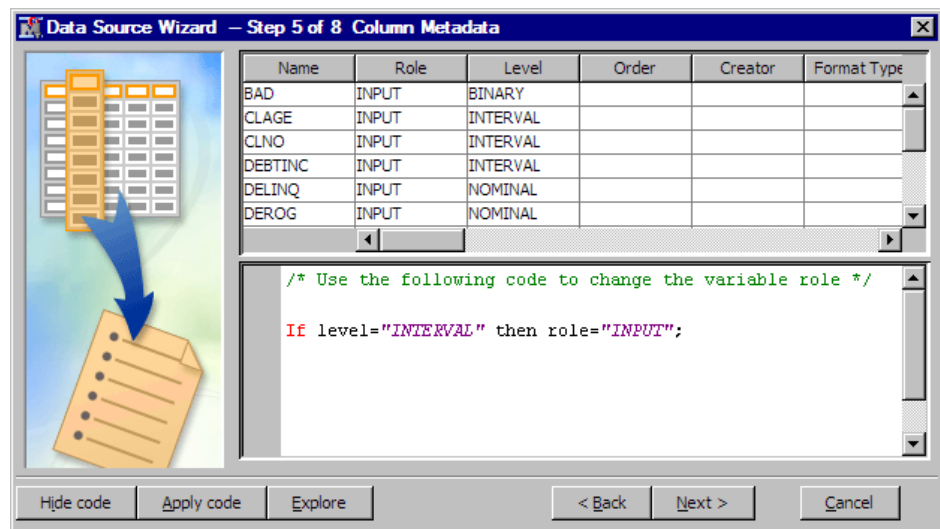
- **Number of Levels** — displays the number of levels for class variables only.
- **Percent Missing** — displays the percentage of missing values. For variables with no missing values, 0 is displayed.
- **Minimum** — displays the minimum values for interval variables only. For class variables, . is displayed.
- **Maximum** — displays the maximum values for interval variables only. For class variables, . is displayed.
- **Mean** — displays the mean values for interval variables only. For class variables, . is displayed.
- **Standard Deviation** — displays the standard deviation values for interval variables only. For class variables, . is displayed.
- **Skewness** — displays the skewness for interval variables only. For class variables, . is displayed.

- **Kurtosis** — displays the kurtosis values for interval variables only. For class variables, . is displayed.

Select a variable and click **Explore** to view the variable distribution. The following is a list of the attributes that you can change. To change an attribute, select the corresponding field of a variable and choose an item from the drop-down list.

- **Name** — is the name of the variable.
- **Role** — is the model role of the variable.
- **Level** — is the measurement level of the variable.
- **Report** — indicates whether a variable should be automatically included in reports such as the predictive model assessment decile and percentile tables.
- **Order** — is the formatted value sorting order for class variables.
- **Drop** — drops the variable.
- **Lower Limit** — is the lower limit of the variable.
- **Upper Limit** — is the upper limit of the variable.
- **Creator** — identifies the SAS Enterprise Miner node that created the variable, if any. For example, probability variables that are created by the **Regression** node have creator Reg.
- **Comment** — contains additional information. SAS Enterprise Miner nodes might set the value of this field.
- **Format Type** — is the format type of the variable.

Alternatively, you can write SAS code to automate the assignment of column attributes, particularly when you have a large number of variables. Clicking **Show Code** enables the Program Editor.



Note: Make sure the measurement levels and model roles in the SAS code are in UPPERCASE.

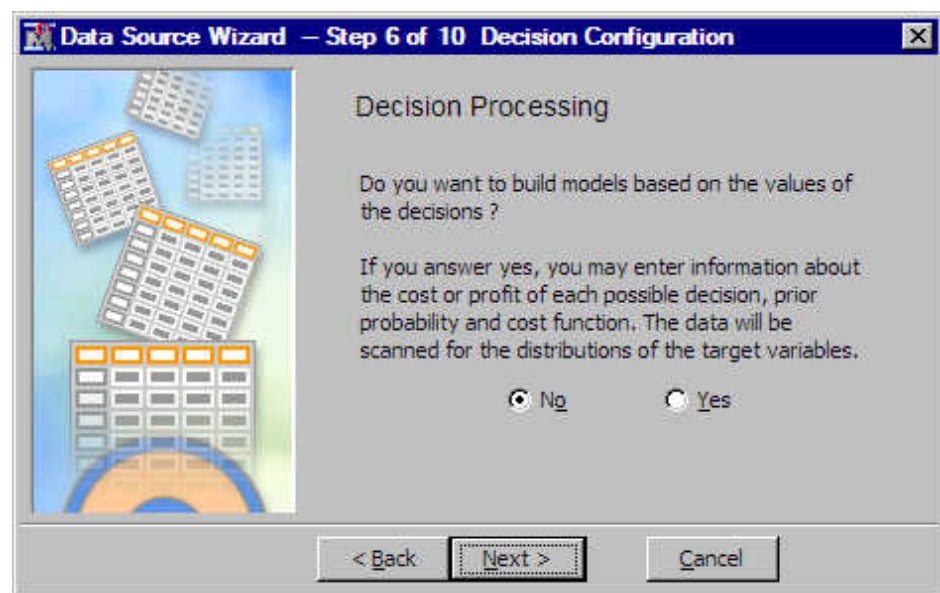
To submit the code, click **Apply Code**.

When you are using the Program Editor, variable names and attributes are displayed as raw values not translated for localization. This is because the program code runs and modifies the raw values. Once you have finished editing and submitting code and reviewing the results, select the Hide Code to view the translated display values.

Most supervised data mining models use a target variable. If you have not used code or other methods to specify a variable with the role of Target, do so now by selecting the variable in the table and choosing **Target** from the list. Use this method to specify other variable roles as well.

Click **Next**.

7. In the Data Source Wizard — Decision Configuration window, you choose whether to define a target profile that produces optimal decisions from a model. Notice that because a binary target variable was chosen, two more configuration windows are added to the Data Source Wizard. As a result, the numbering of the Data Source Wizard windows has changed. If you used Basic Metadata Advisor option, or if you did not configure a target variable role, decision processing is not enabled, and this window will not appear in your Data Source Wizard. If you used the Advanced Metadata Advisor role, and if you specified a binary target variable for your data source, the window depicted below appears, numbered Step 6 of 10.



- If you select **Yes** and click **Next**, the Data Source Wizard — Decision Configuration window appears.
 - If you select **No** and click **Next**, the Data Source Wizard — Data Source Attributes window appears.
8. The Data Source Wizard — Decision Configuration window enables you to set the decision values in the Target Profile. After you finish configuration of the decision configuration, click **Next**.

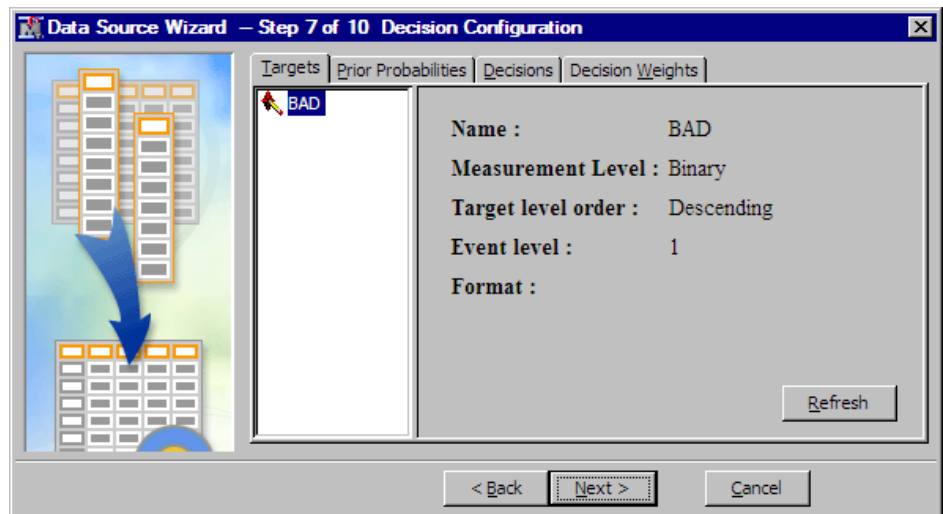
Note: For interval targets, decision configuration is not supported for this release.

The Data Source Wizard — Decision Configuration window has the following tabs:

- **Targets Tab**

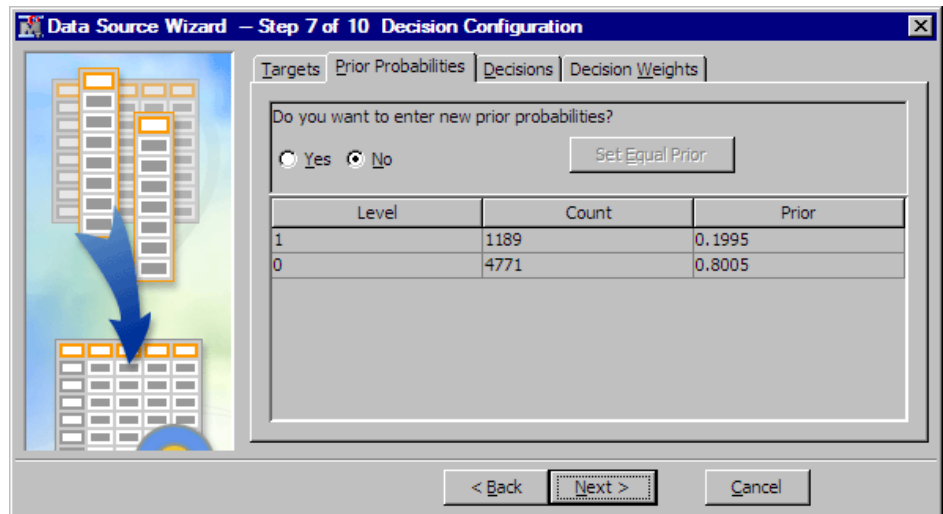
The left panel of the **Targets** tab displays the target variables that are defined in the data source.

For categorical targets, the right panel of the **Targets** tab lists the variable name, measurement level, order, and the event level.



- **Prior Probabilities Tab**

The **Prior Probabilities** tab enables you to specify prior probability values to implement prior class probabilities for categorical targets.



The **Prior Probabilities** tab has the following columns:

Level — displays the levels for target variables.

Count — displays the number of training observations for each target level.

Prior — displays the percentage of training observations for each target level.

Adjusted Prior — displays the prior probability values that you specify. The values that you specify in this column must sum to 1. This column is displayed only if you select **Yes**.

To use your prior probabilities, select **Yes** and enter your adjusted probabilities.

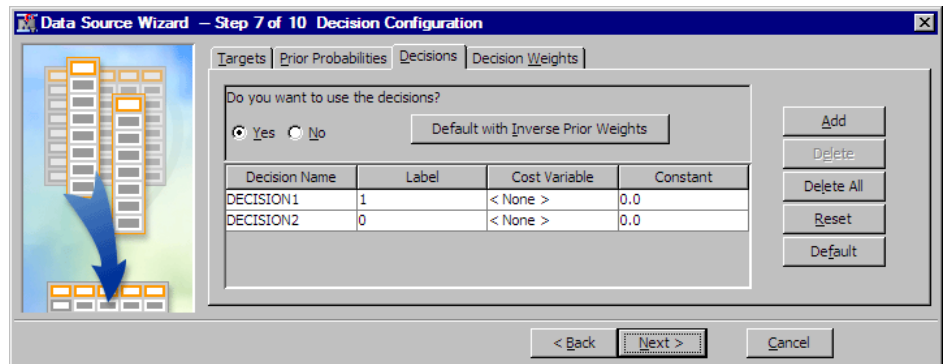
To set prior probabilities equal to each other, click **Set Equal Prior**.

- **Decisions Tab**

The **Decisions** tab enables you to specify decisions, a numeric constant cost or a cost variable for each decision.

The following display shows an example of the **Decisions** tab. To specify a cost variable, select the corresponding field for a decision and select the variable from

the drop-down list. In order to use a variable as the cost variable, the variable role must be set to **Cost** in the Data Source Wizard — Columns Meta window. To specify a constant cost, select **_Constant_** from the drop-down list and enter a value in the corresponding Constant field.



To add a decision, click **Add**.

To delete a decision, select a decision, and click **Delete**.

To delete all decisions, click **Delete All**.

To reset all the settings in the **Decisions** and **Decision Weights** tabs back to the values that you had when you opened the decision editor, click **Reset**.

To use the default values for all the settings in the **Decisions** and **Decision Weights** tabs, click **Default**. The default matrix contains a decision column for each corresponding target level. No cost variable or information is used by default.

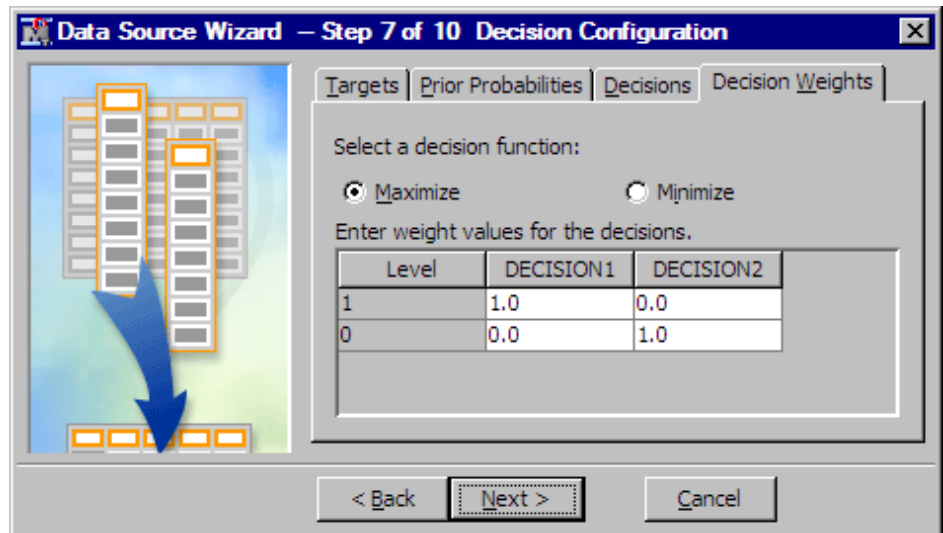
To set the inverse priors as decision weights, click **Default with Inverse Priors**. The **Decision Weights** tab reflects the decision weights that are calculated as the inverse of prior probabilities that you set in the **Prior Probabilities** tab.

- **Decision Weights Tab**

The **Decision Weights** tab enables you to specify the values in the decision matrix. The decision matrix contains columns that correspond to each decision, and rows that correspond to target values. The values of the decision variables represent target-specific consequence, which can be PROFIT, LOSS, or REVENUE. Based on the values of the decision variables, select either **Maximize** or **Minimize** to specify the assessment objective for the matrix. Select **Maximize** if the values in the decision matrix represent profit or revenue. Select **Minimize** if the values in the decision matrix represent loss. By default, **Maximize** is selected.

The target levels that are displayed in the default decision matrix depend on the order of target levels. By default, it is set to descending for categorical targets.

In the following example, there are two decisions because the target BAD contains values of 1 and 0. To change the values in the decision matrix, select a field in the matrix and enter a number.



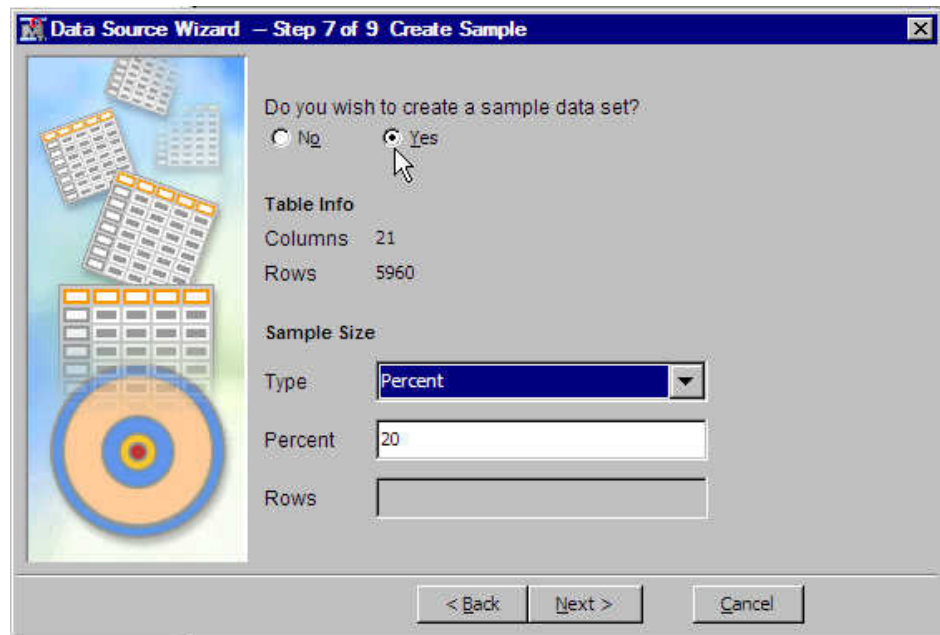
Select **Next**.

9. In the Data Source Wizard – Create Sample window, you specify information about the sample data set that Enterprise Miner automatically creates for data sources. If you did not create a target variable, this window will be labeled Data Source Wizard, Step 6 of 8. If you created a target variable, but did not create a decision matrix, this window will be labeled Data Source Wizard, Step 7 of 9. If you created a target variable and a decision matrix, then this window will be labeled Data Source Wizard, step 8 of 10. Regardless of the path that you took to this window, there are three remaining windows in the Data Source Wizard.

If you want to control the size of the sample data set that Enterprise Miner automatically creates for your data source, select **Yes**, then specify either the percentage of the data to sample, or the number of rows to sample.

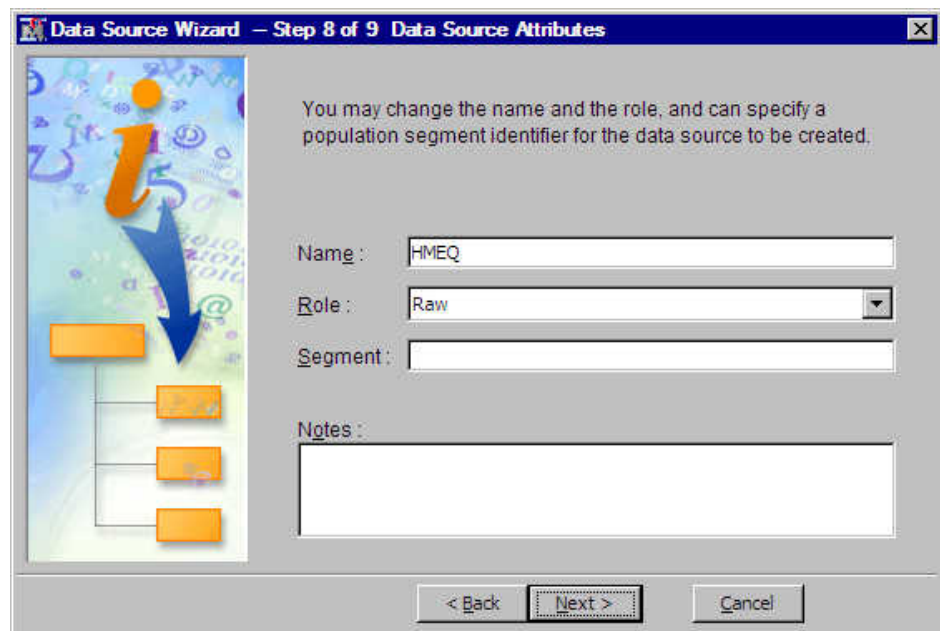
If one or more class targets are defined in the data source, and/or one or more segment variables are defined in the data source, then the automatically created data source sample will be stratified by those variables, up to a limit of two stratification variables.

Sampling is performed in the database where the data is stored, unless the Database pass-through option was set to **No** in the Advanced Advisor. Click **Next**.

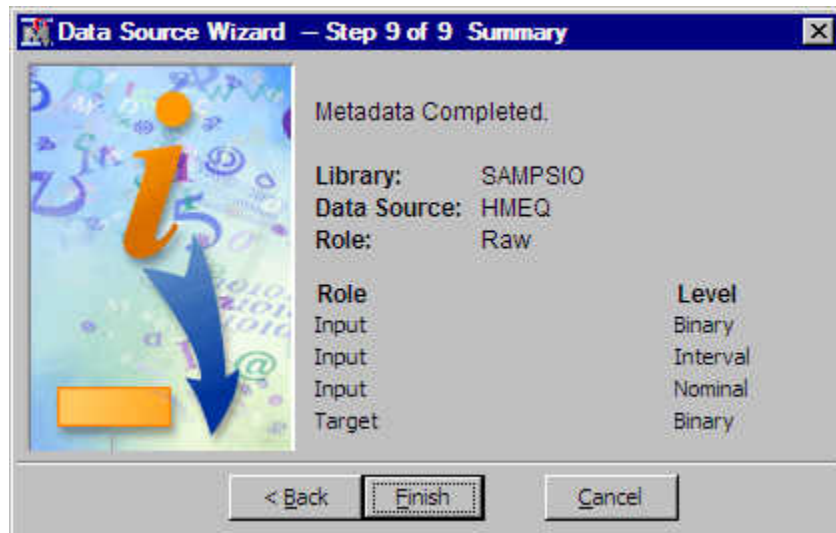


Note: When the input data set is distributed on an appliance, SAS Enterprise Miner automatically creates the sample, even if it is not specified. If the Data Source sample is specified in Step 7 of the Data Source Wizard, then the sample will be the size that you specify in the wizard. Otherwise, Enterprise Miner will create a 10,000 row sample of the data source. If one or more class targets are defined, and/or one or more segment variables are defined, then the sample will be stratified by those variables, up to a limit of two stratification variables.

10. In the Data Source Wizard — Data Source Attributes window, you can specify the name, the role, or a population segment identifier for the data source. Click **Next**.



11. In the Data Source Wizard – Summary window, click **Finish**. The data source appears in the Project Panel of the SAS Enterprise Miner main window.



Create a Process Flow Diagram

Use the diagram editor to add, connect, and move the various nodes in SAS Enterprise Miner. You can perform the following tasks with the diagram editor:

Add a Node

You create a process flow diagram by adding and connecting nodes in the Diagram Workspace. You connect nodes in a logical order that follows the SEMMA data mining methodology.

You can add nodes to a process flow diagram in the following ways:

- Drag node icons from the toolbar to the Diagram Workspace.
- From the main menu, select **Actions** ⇨ **Add node** to open a menu of nodes to add.
- Right-click the Diagram Workspace and select **Add node** from the menu.

Variables Table

If you select the **Edit Variables** option in the node pop-up menu, you will open the variables table for that node. For all nodes other than input data source nodes, the variables table contains a column named **Use**. This column determines whether the variable is used as an input variable.

- **Default** — If a variable has a role of **Input**, then this is treated as **Yes** and if the variable has a role of **Rejected**, then this is treated as **No**. If there is more than one target variable, then this option is not available.
- **Yes** — This is the default value when the role of the variables is **Frequency** or the variable is the first target variable. To use a rejected variable, set its **Use** status to **No**.
- **No** — This is the default value when there is more than one target variable, and the current variable is not the first target variable. To ignore an input variable, set its **Use** status to **No**.

Connect Nodes

Connect nodes in the Diagram Workspace to create a logical process flow. When you create a process flow, follow the SEMMA data mining methodology. Information flows from node to node following the connecting arrows.

The following example connects the Input Data node to the Sampling node.

Figure 2.1 Input Data Node to the Sampling Node



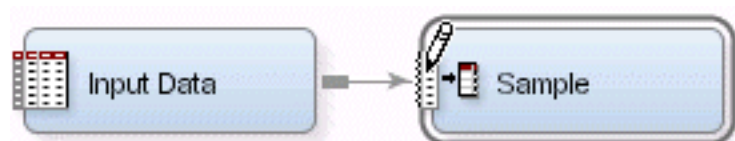
To connect the nodes, move the pointer to the right side of the Input Data node icon. The pointer icon changes to a pencil.

Figure 2.2 Connect Nodes

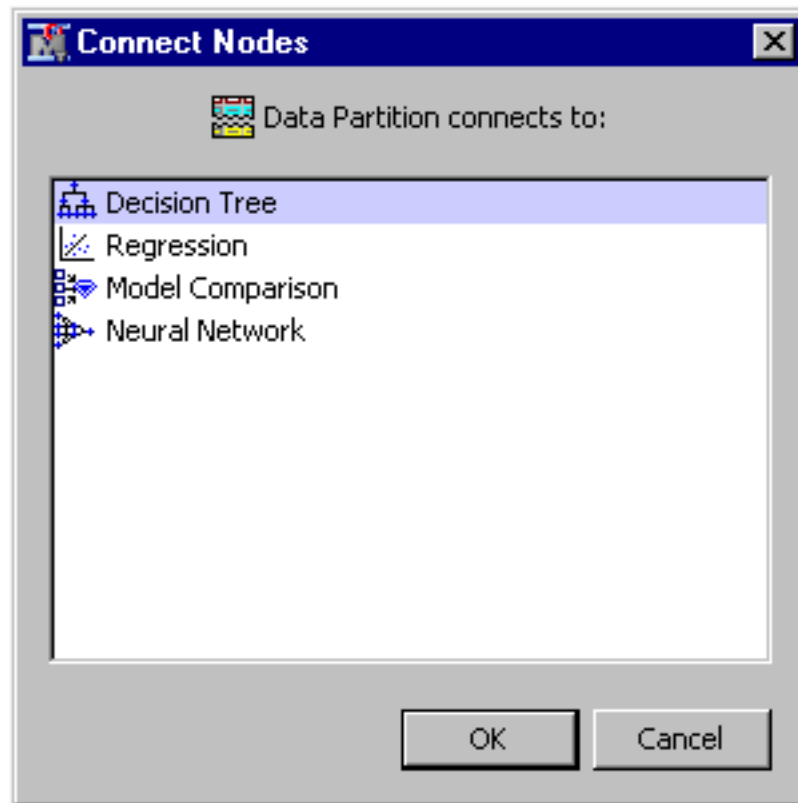


Drag the pointer to the Sample node.

Figure 2.3 Drag Pointer



Release the pointer, and the nodes are connected.

Figure 2.4 Connect Nodes Window

You can connect nodes by using the Connect Nodes window. Right-click the node that you want to connect to other nodes and select **Connect Nodes** from the pop-up menu. Note that the arrows in your process flow diagram point away from the node that you use to open the Connect Nodes window

Click the node name in the window to select the node. You can select multiple nodes by CTRL-clicking or SHIFT-clicking a range of nodes in the window.

You can also use the up and down arrows on your keyboard to select nodes that you want to connect.

Click **OK** to connect the nodes.

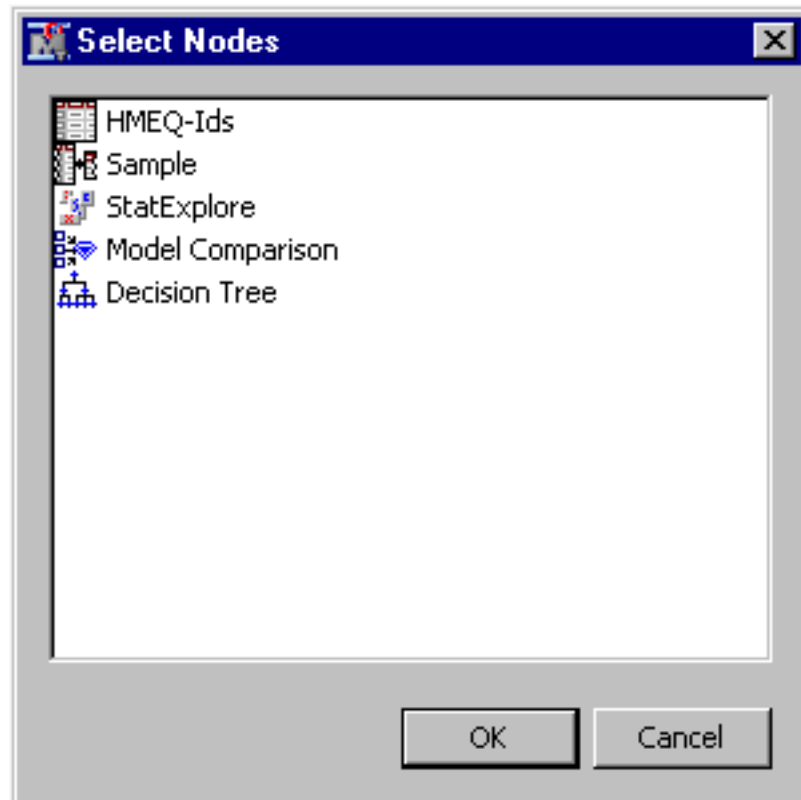
Select Nodes

You can select a node in the Diagram Workspace by clicking the node. Select multiple nodes by holding the CTRL key down and clicking the nodes that you want to select. You can select all of the nodes in the Diagram Workspace by right-clicking the Diagram Workspace and choosing **Select All** from the pop-up menu.

You can also select nodes by using the Select Nodes window. To open the **Select Nodes** window, right-click in the Diagram Workspace and choose **Select Nodes** from the pop-up menu.

You can select single or multiple nodes by using the Select Nodes window.

Figure 2.5 Select Nodes Window



Click the node name in the window to select the node. You can select multiple nodes by holding the CTRL key down and then clicking on the nodes that you want to select. Also, you can select one node, hold the SHIFT key down, then click any other node to select all nodes between and including the two you clicked.

You can also use the up and down arrows on your keyboard to choose a node that you want to select. To select multiple nodes with this method, highlight the first node that you want to select, hold down the CTRL key, use the up and down arrows to move to another node, and then select that node by pressing the space bar. If you hold the SHIFT key while using the up and down arrow keys, you can select all nodes between your first and last node.

Click **OK** to select the nodes.

Disconnect Nodes

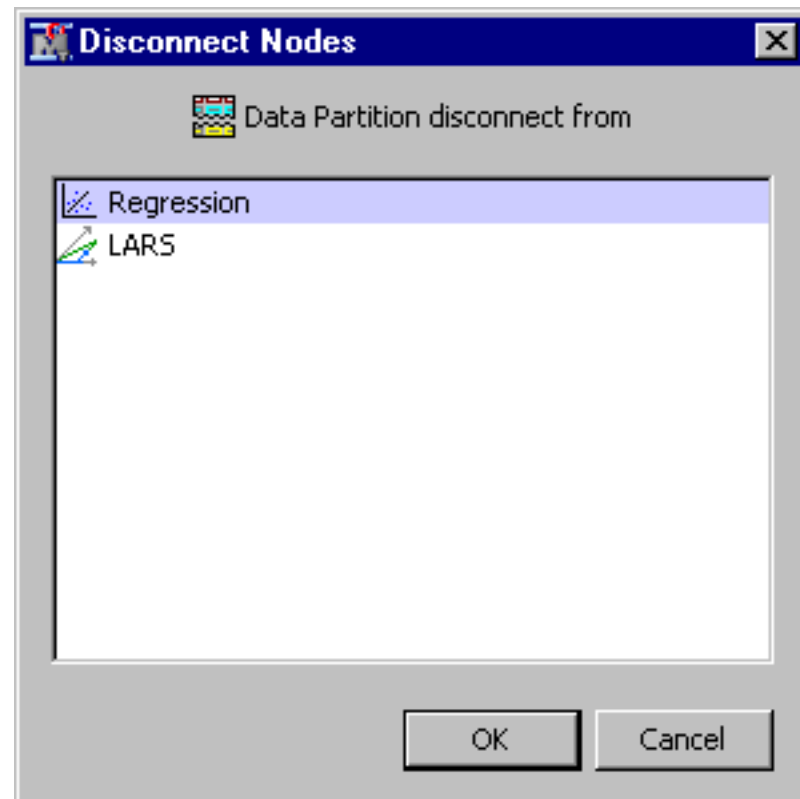
You can disconnect nodes in the Diagram Workspace by clicking the arrow between two nodes so that it is highlighted, and then pressing **Delete** on the keyboard. You can also disconnect two nodes by right-clicking an arrow that connects two nodes in the Diagram Workspace, and selecting Delete from the pop-up menu.

You can disconnect multiple nodes in the Diagram Workspace by holding CTRL down, selecting the arrows that you want to delete, and then either pressing **Delete** on the keyboard, or right-clicking a selected arrow, and choosing **Delete**.

You can also disconnect nodes by using the Disconnect Nodes window. To open the Disconnect Nodes window, right-click a node and select **Disconnect Nodes** from the pop-up menu.

You can select single or multiple nodes by using the Disconnect Nodes window.

Figure 2.6 *Disconnect Nodes Window*



Click the node name in the window to select the node. You can select multiple nodes by holding the CTRL key down and then clicking on the nodes that you want to select. Also, you can select one node, hold the SHIFT key down, then click any other node to select all nodes between and including the two you clicked.

You can also use the up and down arrows on your keyboard to choose a node that you want to select. To select multiple nodes with this method, highlight the first node that you want to select, hold down the CTRL key, use the up and down arrows to move to another node, and then select that node by pressing the space bar. If you hold the SHIFT key while using the up and down arrows that you can select all nodes between your first and last node.

Click **OK** to disconnect the nodes.

Delete Single Nodes

There are two ways in which you can delete a node from a process flow diagram. Here is the first:

1. Right-click the node icon to open a pop-up menu.
2. Select **Delete**. The Confirm Delete dialog box appears and asks you to verify your choice. Click **Yes** to remove the node from the process flow diagram. Click **No** to keep the node.

Here is the second:

1. Click the node to highlight it.
2. Select **Edit** ⇒ **Delete** to delete the node. You can also press the Delete key to delete the selected node.

CAUTION:

You cannot undelete a deleted node.

Delete Multiple Nodes

To delete multiple nodes, follow these steps:

1. Select a node icon that you want to delete, and then hold down the Control key and click to select the remaining node icons that you want to delete. The selected nodes become highlighted. You can also select multiple nodes by dragging your pointer around the nodes that you want to delete.
2. Right-click a selected node.
3. Select **Delete**. A Confirm Delete window appears that asks you to verify your choice. If you click **Yes**, then the nodes are removed from the process flow diagram. If you click **No**, then the nodes remain in the process flow diagram.

To delete all of the nodes in the Diagram Workspace, select **Edit** ⇒ **Select All** from the main menu, and then choose **Edit** ⇒ **Delete**.

Move or Reposition Nodes

To move a node, follow these steps:

1. Click the node to select it.
2. Drag the node icon to a new position in the Diagram Workspace. The connections stretch to accommodate the move.

Note: You can move multiple nodes by selecting the nodes that you want to move and dragging them to a new position.

You can arrange your diagram nodes by selecting **Edit** ⇒ **Layout** from the main menu, and then choosing **Horizontally** or **Vertically**.

Copy and Paste Nodes

You can copy an existing node to the clipboard and paste it to the Diagram Workspace. The same node properties of the node that you copy are used for the new node. You can copy a node and paste it to the same diagram or to a different diagram.

To copy an existing node:

1. Right-click the node that you want to copy in the Diagram Workspace and select **Copy**.
2. Right-click the location in the diagram where you want to paste the node and select **Paste**.

Alternatively, after you select a node in the Diagram Workspace, you can select **Copy** and then **Paste** from the main menu under **Edit**. To determine where the copied nodes will be pasted, left-click in the diagram where you want to paste the nodes.

Note that when you copy a node to a Diagram Workspace, you must ensure that all nodes have a unique name.

Chapter 3

Sample Data

Sample Nodes: Overview	23
Partition Data	24
Overview of the Data Partition Node	24
Data Partition Node Data Requirements	25
Data Partition Node Properties	25
Data Partition Node Variables Table	27
Data Partition Node Results	29
Data Partition Node Output Data Sources	30
Filter Data	30
Overview of the Filter Node	30
Filter Node Properties	30
Interactive Class Filter Window	35
Interactive Interval Filter Window	36
Filter Node Results	37
Filter Node Example	38

Sample Nodes: Overview

- The **Append** node enables you to append data sets that are exported by two or more paths in a single SAS Enterprise Miner process flow diagram. The **Append** node can append data according to the data role, such as joining training data to training data, transaction data to transaction data, score data to score data, and so on. The **Append** node can append data that was previously partitioned in train, test, and validate roles into one large training data set.
- The **Data Partition** node enables you to partition data sets into training, test, and validation data sets. The training data set is used for preliminary model fitting. The validation data set is used to monitor and tune the model weights during estimation and is also used for model assessment. The test data set is an additional hold-out data set that you can use for model assessment. This node uses simple random sampling, stratified random sampling, or user-defined partitions to create partitioned data sets.
- The **File Import** node enables you to import data that is stored in external formats into a data source that SAS Enterprise Miner can interpret. The **File Import** node currently can process CSV flat files, JMP tables, Microsoft Excel and Lotus spreadsheet files, and DBF, DLM, and DTA files.
- The **Filter** node enables you to apply a filter to the training data set in order to exclude outliers or other observations that you do not want to include in your data

mining analysis. Outliers can greatly affect modeling results and, subsequently, the accuracy and reliability of trained models.

- The **Input Data** node enables you to access SAS data sets and other types of data. The **Input Data** node represents the introduction of a predefined metadata into a Diagram Workspace for processing. You can view metadata information about your source data in the **Input Data** node, such as initial values for measurement levels and model roles of each variable.
- The **Merge** node enables you to merge observations from two or more data sets into a single observation in a new data set. The **Merge** node supports both one-to-one and match merging. In addition, you have the option to rename certain variables (for example, predicted values and posterior probabilities) depending on the settings of the node.
- The **Sample** node enables you to take random, stratified random, and cluster samples of data sets. Sampling is recommended for extremely large databases because it can significantly decrease model training time. If the sample is sufficiently representative, then relationships found in the sample can be expected to generalize to the complete data set.

Partition Data



Overview of the Data Partition Node

Most data mining projects utilize large volumes of sampled data. After sampling, the data is usually partitioned before modeling.

Use the Data Partition node to partition your input data into one of the following data sets:

Train	is used for preliminary model fitting. The analyst attempts to find the best model weights using this data set.
Validation	<p>is used to assess the adequacy of the model in the Model Comparison node.</p> <p>The validation data set is also used for model fine-tuning in the following nodes:</p> <ul style="list-style-type: none"> • Decision Tree node — to create the best subtree. • Neural Network node — to choose among network architectures or for the early-stopping of the training algorithm. • Regression node — to choose a final subset of predictors from all the subsets computed during stepwise regression.
Test	is used to obtain a final, unbiased estimate of the generalization error of the model.

Partitioning provides mutually exclusive data sets. Two or more mutually exclusive data sets share no observations with each other. Partitioning the input data reduces the computation time of preliminary modeling runs. However, you should be aware that for small data sets, partitioning may be inefficient as the reduced sample size can degrade the fit of the model and its ability to generalize.

What are the steps to partitioning a data set? First, you specify a sampling method: simple random sampling, stratified random sampling, or cluster sampling. Then you specify the proportion of sampled observations to write to each output data set (Train, Validation, and Test). It is not uncommon to omit the test data set; that is, assign 0% of the observations to the test data set.


Data Partition Node Data Requirements

The Data Partition node must be preceded by a node that exports at least one raw, train, or document data table, which is usually an Input Data node or a Sample node.


Data Partition Node Properties

Data Partition Node General Properties

The following general properties are associated with the Data Partition node:


- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Data Partition node that is added to a diagram will have a Node ID of Part. The second Data Partition node added to a diagram will have a Node ID of Part2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Data Partition window. The Imported Data — Data Partition window contains a list of the ports that provide data sources to the Data Partition node. Select the  button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data - Data Partition window. The Exported Data - Data Partition window contains a list of the output data ports that the Data Partition node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.


If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.

- **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Data Partition Node Train Properties

The following train properties are associated with the Data Partition node:

- **Variables** — Use the Variables property to specify the status for individual variables that are imported into the Data Partitioning node. Select the  button to open a window containing the variables table. You can specify the Partitioning Role for individual variables, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Output Type** — Use the Output Type property of the Data Partitioning node to specify the type of output the node should generate. The drop-down box offers choices of data set output (**Data**), or SAS data step views (**View**). The default setting for the Output Type property is **Data**.
- **Partitioning Method** — Use the Partitioning Method property to specify the sampling method that you want to use when you are partitioning your data.

You can choose from:

- **Default** — When Default is selected, if a class target variable or variables is specified, then the partitioning is stratified on the class target variables. Otherwise, simple random partitioning is performed. Note that if additional stratification variables are specified in the Variables Editor, they will be used in addition to the target variables.
- **Simple Random** — When Simple Random is selected, every observation in the data set has the same probability of being written to one of the partitioned data sets.
- **Cluster** — Using simple cluster partitioning, you allocate the distinct values of the cluster variable to the various partitions using simple random partitioning. Note that with this method, the partition percentages apply to the values of the cluster variable, and not to the number of observations that are in the partition data sets. If you select cluster partitioning as your method, you must use Variables property of the Data Partition node to set the Partition Role of the cluster variable (which may be the target variable) to Cluster.
- **Stratified** — Using stratified partitioning, you specify variables to form strata (or subgroups) of the total population. Within each stratum, all observations have an equal probability of being written to one of the partitioned data sets. You perform stratified partitioning to preserve the strata proportions of the population within each partition data set. This might improve the classification precision of fitted models. If you select Stratified partitioning as your method and no class target variables are defined, then you must use the Variables window to set the partition role and specify a stratification variable.
- **Random Seed** — Use the Random Seed property to specify an integer value to use as a random seed for the pseudorandom number generator that chooses observations for sampling. The default value for the Random Seed property is 12345.

Data Partition Node Train Properties: Data Set Allocations

You can specify the percentages, or proportions of the source data that you wish to allocate as Train, Validation, and Test data sets.

- **Training** — Use the Training property to specify the percentage of observations that you want to allocate to the training data set. Permissible values are real numbers between 0 and 100. The SAS default training percentage is 40%.
- **Validation** — Use the Validation property to specify the percentage of observations that you want to allocate to the validation data set. Permissible values are real numbers between 0 and 100. The SAS default validation percentage is 30%.
- **Test** — Use the Test property to specify the percentage of observations that you want to allocate to the test data set. Permissible values are real numbers between 0 and 100. The SAS default test percentage is 30%.

Data Partition Node Report Properties

The following report properties are associated with the Data Partition node:

- **Interval Targets** — Use the Interval Targets property to specify whether you want to generate reporting statistics on interval targets in your data set. The Interval Targets summary displays a table of descriptive statistics based on the partitioned data sets and original data for the interval target variables. The default setting for the Interval Targets report property is Yes.
- **Class Targets** — Use the Class Targets property to specify whether you want to generate reporting statistics on the class targets in your data set. The Class Targets summary displays bar charts that compare the distribution of the class target variables in the incoming data set with the partitioned data sets for the same variables. The charts display the percent of the total frequency for each level of the target for each data set.

Note: The Data Partition node allows users to disable partitioned data summaries of interval and class target variables which will speed processing when working with very large data sets.


Data Partition Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Data Partition Node Variables Table

Use the table in the Variables — Data Partition window to identify cluster and stratification variables and to specify data partitioning methods for individual variables

on a one-by-one basis. To open the Variables — Data Partition window, select the  button to the right of the Variables property in the Properties Panel while the Data Partition node is selected in the Diagram Workspace.

Use this table to specify the partitioning role of specific variables. You can choose the following roles: Cluster, Stratification, None, and Default. If a partition role is dimmed and unavailable, then that partition role would not be applicable for that particular variable.

The following are read-only attributes: Role, Level, Type, Order, Label, Format, Informat, Length, Lower Limit, Upper Limit, Comment, Report, Creator, Family, and Distribution.

You can highlight a variable in the table and click the Explore button to get sampling, distribution, and metadata information in the Explore Variable window.

The table in the Variables — Data Partition window contains the following columns:

- **Name** — displays the name of the variable.
- **Partition Role** — click the cell to specify the partitioning method that you want to use on an individual variable in your imported data set.
 - **Default** — Uses the partitioning method that is specified for all variables in the Data Partition node Properties panel, unless the Cluster partitioning method is chosen. If the partitioning method specified for all variables in the Properties Panel is **Default**, then class variables will be stratified and interval variables will use simple random sampling. If the **Cluster** partitioning method is specified, you must change the Partition Role for the cluster variable from **Default** to **Cluster**.
 - **None** — Do not partition the specified class or interval variable.
 - **Cluster** — If the Partitioning Method property of the Data Partition node is set to **Cluster**, you must use the Partition Role column of the Variables window to specify the cluster variable. Cluster partitioning samples from a cluster of observations are similar in some way.
 - **Stratification** — you specify variables from the input data set to form strata (or subsets) of the total population. Within each stratum, all observations have an equal probability of being selected for the sample. Across all strata, however, the observations in the input data set generally do not have equal probabilities of being selected for the sample. You perform stratified sampling to preserve the strata proportions of the population within the sample. This may improve the classification precision of fitted models.

The following are read-only columns in the Variables — Data Partition table. You must use a Metadata node if you want to modify any of the following information about Data Source variables imported into the Data Partition node.

- **Role** — displays the variable role
- **Level** — displays the level for class variables
- **Type** — displays the variable type
- **Order** — displays the variable order
- **Label** — displays the text label to be used for variable in graphs and output
- **Format** — displays the variable format
- **Informat** — displays the SAS Informat for variable
- **Length** — displays the variable length

- **Lower Limit** — displays the minimum value for variable
- **Upper Limit** — displays the maximum value for variable
- **Distribution** — the expected univariate distribution
- **Family** — identifies the general source of a variable such as demographic, financial, historic for record keeping. Such values may be useful in constructing the data.
- **Report** — displays the Boolean setting for creating reports
- **Creator** — displays the user who created the process flow diagram
- **Comment** — displays the user-supplied comments about a variable

Data Partition Node Results

After a successful node run, you can open the Results window of the Data Partition node by selecting the **Results** button in the Run Status window, or by going to the Diagram Workspace, right-clicking the Data Partition node, and selecting Results from the pop-up menu.

Select **View** from the main menu to view the following results in the Results window:

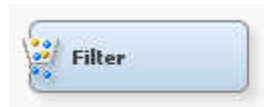
- **Properties**
 - **Settings** — displays a window that shows how the Data Partition node properties were configured when the node last ran. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
 - **Run Status** — indicates the status of the Data Partition node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — a table of the properties of the variables used in this node. Here, the nominal variable that is employed has been assigned as the clustering variable.
 - **Train Code** — the code that Enterprise Miner used to train the node.
 - **Notes** — allows users to read or create notes of interest.
- **SAS Results**
 - **Log** — the SAS log of the Data Partition node run.
 - **Output** — the SAS output of the Data Partition node run. The output displays a variable summary and summary statistics for class or interval targets or both in the original data and in the various partition data sets.
 - **Flow Code** — Flow Code is not available for the Data Partition node.
- **Scoring**
 - **SAS Code** — the Data Partition node does not generate SAS code.
 - **PMML Code** — the Data Partition node does not generate PMML code.
- **Summary Statistics**
 - **Interval Variables** — The Interval Variables summary displays a table of descriptive statistics based on the partition data sets and original data for the interval target variables. The statistics are as follows.
 - **Data** — the type of data
 - **Variable** — the interval target variable

- **Maximum** — maximum value
- **Mean** — arithmetic mean
- **Minimum** — minimum value
- **Number of Observations** — the number of non-missing observations
- **Missing** — number of missing observations
- **Standard Deviation** — the standard deviation.
- **Class Variables** — displays bar charts that compare the distribution of the class target variables in the incoming data set with the partitioned data sets for the same variables. The charts display the percent of the total frequency for each level of the target for each data set.
- **Table** — displays a table that contains the underlying data used to produce the selected chart. The **Table** menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — displays the Graph Wizard that you can use to create ad-hoc plots that are based on the underlying data that produced the selected table.

Data Partition Node Output Data Sources

The Data Partition node exports up to three data sets, depending on the settings that you make in the Properties panel. By default, the node exports a Train, Validate, and Test data set.

Filter Data



Overview of the Filter Node

The Filter node tool is located on the **Sample** tab of the Enterprise Miner tools bar. Use the Filter node to create and apply filters to your training data set. You can also use the Filter node to create and apply filters to the validation and test data sets. You can use filters to exclude certain observations, such as extreme outliers and errant data that you do not want to include in your mining analysis. Filtering extreme values from the training data tends to produce better models because the parameter estimates are more stable. By default, the Filter node ignores target and rejected variables.


Filter Node Properties

Filter Node General Properties


The following general properties are associated with the Filter Node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Filter node that


is added to a diagram will have a Node ID of Filter. The second Filter node added to a diagram will have a Node ID of Filter2, and so on.

- **Imported Data** — The Imported Data property provides access to the Imported Data — Filter window. The Imported Data — Filter window contains a list of the ports that provide data sources to the **Filter** node. Select the  button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Filter window. The Exported Data — Filter window contains a list of the output data ports that the **Filter** node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data set. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Filter Node Train Properties

The following train properties are associated with the Filter Node:


- **Export Table** — Use the Export Table property of the Filter node to indicate whether you want to export a data set containing filtered or excluded observations.
 - **Excluded** — Export the excluded observations.
 - **Filtered** — (default setting) Export the included observations.
 - **All** — Export all observations. A dummy indicator variable, `M_FILTER`, is created to identify the excluded observations. That is, `M_FILTER` is set to 1 for excluded observations and 0 otherwise.
- **Tables to Filter** — Use the Tables to Filter property of the Filter node to indicate whether you want to filter the training data set or all imported data sets.
 - **Training Data** — (default setting) Filter only the training data set.
 - **All Data Sets** — Filter all imported data sets.
- **Distribution Data Sets** — Use to create summary data sets to display histograms and bar charts when you set manual filters.

- **Yes** — (default setting) The VARDIST and CLASSDIST data sets will be created if there are interval and class variables.
- **No** — The data sets will not be created at training time.

If you run the node with this property set to **Yes** and then change it to **No**, the tables will not be deleted. Regardless of the value of this property, you should be able to create or refresh the VARDIST and CLASSDIST data sets using their respective Variables Editor.

Filter Node Train Properties: Class Variables

Use the Filter Node Class Variables properties to specify how you want to filter class variables. You can also use the Interactive Filter Class window to specify filter methods for individual variables.

- **Class Variables** — Click the  button to the right of the Class Variables property to open the “Interactive Class Filter Window” on page 35. You can use the columns in the table to specify Filtering Method, Keep Missing Values, Minimum Frequency Cutoff, Number of Levels Cutoff, and Report settings on a variable-by-variable basis, instead of applying a default filter to all class variables. The Role and Level values for a variable are displayed. You can see additional metadata information by selecting the Label, Mining, Basic, or Statistics check boxes.
- **Default Filtering Method** — Use the Default Filtering Method property of the Filter node to specify the method that you want to use to filter class variables.
 - **Rare Values (Count)** — Drop rare levels that have a count less than the level that you specify in the Minimum Frequency Cutoff property.
 - **Rare Values (Percentage)** — (default setting) Drop rare levels that occur in proportions lower than the percentage that you specify in the Minimum Cutoff for Percentage property.
 - **None** — No class variable filtering is performed.
- **Keep Missing Values** — Set the Keep Missing Values property of the Filter node to No if you want to filter out observations that contain missing values for class variables. The default setting for the Keep Missing Values property is Yes.
- **Normalized Values** — Set the Normalized Values property of the Filter node to No to suppress the normalization of class variable values before filtering. The default setting for the Normalize Values property is Yes.

Categorical values (class variable level values) in Enterprise Miner are normalized as follows:

- Left-justified
- Uppercase
- Truncated to a length of 32 (after left-justification)

This implies that values such as "YES", "yes", "Yes", and "yES" are all considered the same. Likewise, "purchasing a new big bright red ball" and "purchasing a new big bright red cap" are also considered identical, because they are identical under those rules for the first 32 characters.


- **Minimum Frequency Cutoff** — When you set the Default Filter Method for class variables to Rare Values (Count), you must use the Minimum Frequency Cutoff property of the Filter node to quantify the minimum count threshold. Class variable values that occur fewer times than the specified count are filtered. Permissible values are positive integers. The default value for the Minimum Frequency Cutoff property is 1.

- **Minimum Cutoff for Percentage** — When you set the Default Filter Method for class variables to Rare Values (Percentage), you must use the Minimum Percentage Cutoff property of the Filter node to specify the minimum percentage threshold. Observations with rare levels for a class variable are counted to calculate rare level proportions. Rare levels that do not meet the minimum percentage threshold are filtered. Permissible values are real numbers from 0 to 1. The default value is 1%, or 0.01.
- **Maximum Number of Levels Cutoff** — Use the Maximum Number of Levels Cutoff property to determine whether a class variable in a data set will be considered for filtering. Only class variables that have fewer levels than the cutoff value are considered for filtering.


Filter Node Train Properties: Interval Variables

Use the Filter Node Interval Variables properties to specify how you want to filter interval variables.

You can configure and modify the following properties for the Filter Node:

- **Interval Variables** — Click the  button to the right of the Interval Variables property to open the “Interactive Interval Filter Window” on page 36. The interactive window is especially useful if you want to specify different filter methods for individual variables instead of applying the default filtering method to all interval variables. You can use the columns in the table to specify Filtering Method, Keep Missing Values, Filter Upper and Lower Limit, and Report settings on a variable-by-variable basis. The Role and Level values for a variable are displayed. You can see additional metadata information by selecting the Label, Mining, Basic, or Statistics check boxes. You can use the Interactive Interval Filter table to filter an interval variable for a single value by setting both the Filter Upper Limit and Filter Lower Limit columns for that variable to the desired variable value.
- **Default Filtering Method** — Use the Default Filtering Method property of the Filter node to specify the method that you want to use to filter interval variables. The Default Filtering Method applies only to input variables.
 - **Mean Absolute Deviation (MAD)** — The Median Absolute Deviation method eliminates values that are more than n deviations from the median. You specify the threshold value for the number of deviations, n , in the Cutoff for MAD property.
 - **User-Specified Limits** — The User-Specified method specifies a filter for observations that is based on the interval values that are displayed in the Filter Lower Limit and Filter Upper Limit columns of your data table. You specify these limits in the Variables table.
 - **Metadata Limits** — Metadata Limits are the lower and upper limit attributes that you can specify when you create a data source or when you are modifying the Variables table of an Input Data node on the diagram workspace.
 - **Extreme Percentiles** — The extreme percentiles method filters values that are in the top and bottom p^{th} percentiles of an interval variable's distribution. You specify the upper and lower threshold value for p in the Cutoff Percentiles for Extreme Percentiles property.
 - **Modal Center** — The Modal Center method eliminates values that are more than n spacings from the modal center. You specify the threshold value for the number of spacings, n , in the Cutoff Percentiles for Modal Center property.
 - **Standard Deviations from the Mean** — (default setting) The Standard Deviations from Mean method filters values that are greater than or equal to n

standard deviations from the mean. You must use the Cutoff Percentiles for Standard Deviations property to specify the threshold value that you want to use for n .

- **None** — Do not filter interval variables.
- **Keep Missing Values** — Set the Keep Missing Values property of the Filter node to No if you want to filter out observations that contain missing values for interval variables. The default setting for the Keep Missing Values property is Yes.
- **Tuning Parameters** — Click the  button to the right of the Tuning Parameters property to open the Tuning Parameters window. You use the Tuning Parameters window to specify the tuning parameters, or cutoff values that correspond to the robust filter methods that are available in the Default Filtering Method property.
 - **Cutoff for MAD** — When you specify Mean Absolute Deviation as your Default Filtering Method, you must use the MAD Cutoff property of the Filter node to quantify n , the threshold value for the number of deviations from the median value. Observations with interval variable values that exceed the value of the Cutoff for MAD property are filtered. Permissible values are real numbers greater than or equal to zero. The default value is 9.0.
 - **Cutoff Percentiles for Extreme Percentiles** — When you specify Extreme Percentiles as your Default Filtering Method, you must use the Cutoff Percentiles for Extreme Percentiles property to specify p , the threshold value used to quantify the top and bottom p^{th} percentiles. Observations with interval variable values in the extreme p^{th} percentiles are filtered. Permissible values are percentages greater than or equal to 0 and less than 50. (P specifies upper and lower thresholds, $50\% + 50\% = 100\%$.) The default value is 0.5, or 0.5%.
 - **Cutoff for Modal Center** — When you specify Modal Center as your Default Filtering Method, you must use the Cutoff for Modal Center property to specify the threshold number of spaces n . Observations with interval values more than n spacings from the modal center are filtered. Permissible values are real numbers greater than or equal to zero. The default value is 9.0.
 - **Cutoff for Standard Deviation** — Use the Cutoff for Standard Deviation property to quantify n , the threshold for number of standard deviations from the mean. Observations with values more than n standard deviations from the mean are filtered. Permissible values are real numbers greater than or equal to zero. The default value is 3.0.

Filter Node Score Properties

The following score properties are associated with the Filter Node:

- **Create Score Code** — Set the Create Score Code property to No if you want to suppress creation of score code by the node. The score code contains the filters that were created during training. The default setting for the Create Score Code property is Yes.
- **Update Measurement Level** — specifies whether the measurement level of class variables is updated.

Filter Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.

- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Interactive Class Filter Window

Introduction

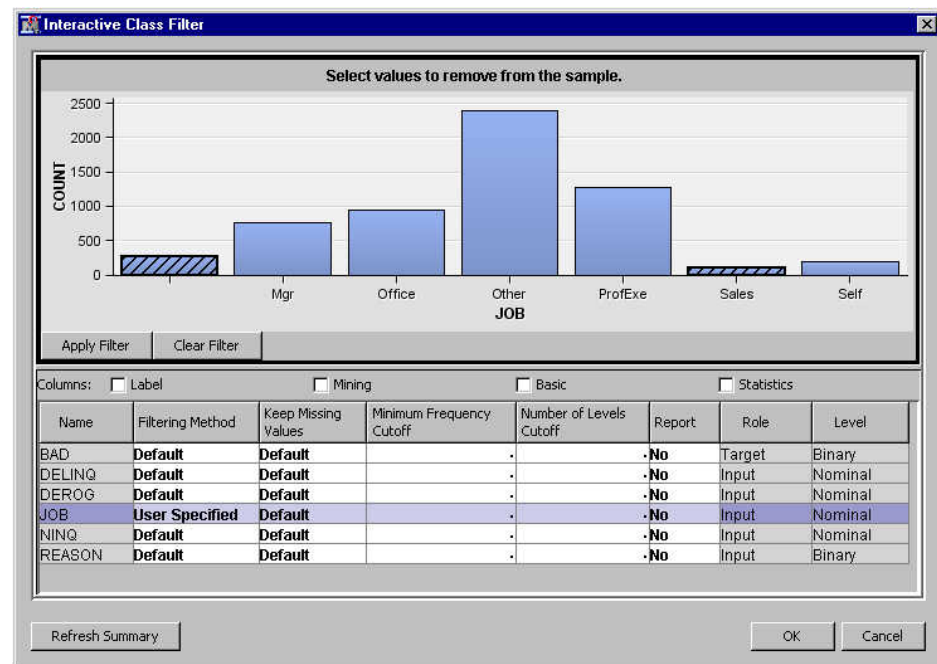
Use the Filter Variables Editor to specify filtering properties for class variables on a variable-by-variable basis. You can configure and modify the Filtering Method, Keep Missing Values, Minimum Frequency Cutoff, Number of Levels Cutoff, and Report attributes. For details about these filtering properties, see [“Filter Node Train Properties: Class Variables” on page 32](#).

Read-only attributes for the variable's Role and Level are also displayed. Additional read-only attributes can be displayed by selecting the Label, Mining, Basic, or Statistics check box. The Label option displays a Label read-only attribute. The Mining option displays the following read-only attributes: Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type. The Basic option displays the following read-only attributes: Type, Format, Informat, and Length. The Statistics option displays the following read-only attributes: Number of Levels, Percent Missing, Minimum, Maximum, Mean, Standard Deviation, Skewness, and Kurtosis.

Interactive Filtering

In addition to the filtering previously discussed, the Interactive Class Filter window provides the additional filtering method of **User Specified**. To interactively select values to exclude, select a variable in the table.

- The variable distribution displayed is based on a summary data set. To generate this data set, click **Refresh Summary**. Note that when the node runs, this data set is refreshed or created.
- To select the values to exclude, click on the various bars. The selected bars should be outlined and shaded. Click **Apply Filter** to confirm your choices, and the **Filtering Method** field should change to User Specified.
- Click **Clear Filter** to clear the specified filtered values.



Interactive Interval Filter Window

Introduction

Use the editor to specify filtering properties for interval variables on a variable-by-variable basis. You can configure and modify the Filtering Method, Keep Missing Values, Filter Lower Limit, Filter Upper Limit, and Report attributes.

- **Filter Lower Limit** — a numeric field that can be used to define the lower limit of a variable filter.
- **Filter Upper Limit** — a numeric field that can be used to define the upper limit of a variable filter.

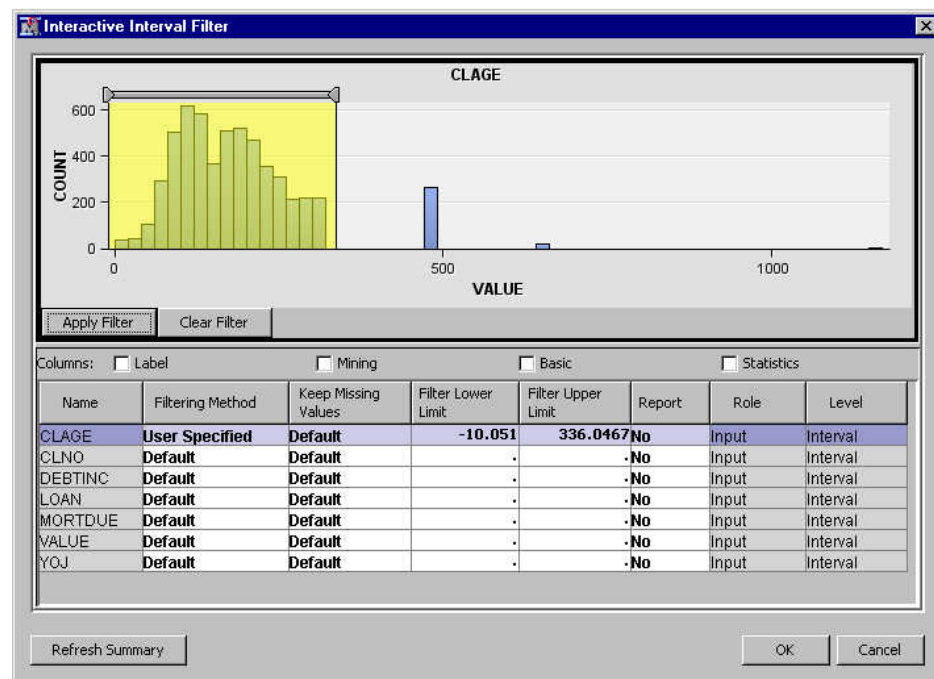
You can use the Interactive Interval Filter table to keep only records for only a specified single value of an interval variable. To keep only records for a specified value of an interval variable, set both the Filter Upper Limit and Filter Lower Limit columns for that variable to the desired variable value. For details about these filtering properties, see [“Filter Node Train Properties: Interval Variables” on page 33](#).

Read-only attributes for the variable's Role and Level are also displayed. Additional read-only attributes can be displayed by selecting the Label, Mining, Basic, or Statistics check box. The Label option displays a Label read-only attribute. The Mining option displays the following read-only attributes: Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type. The Basic option displays the following read-only attributes: Type, Format, Informat, and Length. The Statistics option displays the following read-only attributes: Number of Levels, Percent Missing, Minimum, Maximum, Mean, Standard Deviation, Skewness, and Kurtosis.

Interactive Filtering

In addition to the standard filtering methods, the Interactive Interval Filter window provides the additional filtering method of **User Specified**. To interactively select a filtering range for a variable, select a variable in the table.

- The variable distribution displayed is based on a summary data set. To generate this data set, click **Refresh Summary**. Note that when the node runs, this data set is refreshed or created.
- The shaded area identifies the range of values to be kept. To modify the filter limits, you can move the sliders at the top of the graph.
- You can also enter values directly in the Filter Lower Limit field and the Filter Upper Limit field. After entering a value, the shaded area in the plot is updated.
- Click **Apply Filter** to confirm your choices, the **Filtering Method** field should change to User Specified.
- Click **Clear Filter** to clear the filter limits.



Filter Node Results

You can open the Results window of the **Filter** node by right-clicking the node and selecting **Results** from the pop-up menu.

Select **View** from the main menu to view the following results in the Results window:

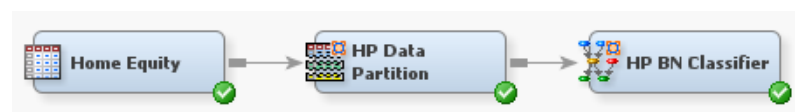
- **Properties**
 - **Settings** — displays a window that shows how the Filter node properties were configured when the node last ran. Use the **Show Advanced Properties** check box at the bottom of the window to see all of the available properties.
 - **Run Status** — indicates the status of the Filter node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — a table of the variables used in this node.
 - **Train Code** — the code that Enterprise Miner used to train the node. This code is not available when the Table to Filter property is set to All Data Sets.
 - **Notes** — allows users to read or create notes of interest.

- **SAS Results**
 - **Log** — the SAS log of the Filter node run.
 - **Output** — the SAS output of the Filter node run.
 - **Flow Code** — the SAS code used to produce the output that the Filter node passes on to the next node in the process flow diagram. This code is not available when the Table to Filter property is set to Training Data.
- **Scoring**
 - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
 - **PMML Code** — the Filter node does not generate PMML code.
- **Filters**
 - **Excluded Class Variables** — displays a table of all of the class variables that were filtered from the training data set. The table provides data for class variable role, level, train count, train percent, label, filter method, and keep missing values.
 - **Limits for Interval Variables** — displays a table that shows the upper and lower limits that were established by the filter method for interval variables. The table provides data for interval variable role, minimum, maximum, filter method, keep missing values, and label.
- **Graphs** — when filters are applied, the node generates distribution plots for filtered variables that are specified as report variables in the Class Variables or Interval Variables property tables.
 - **Class Variables** — display bar charts that compare the distribution of class variables in the training data set with the filtered levels of the same variables. Use the drop-down menu to select the variable that you want to view.
 - **Interval Variables** — display histograms that compare the distribution of interval variables in the training data set with the filtered variables. Use the drop-down menu to select the variable that you want to view.
- **Table** — open the data table that corresponds to the graph that you have in focus.
- **Plot** — opens the Select a Chart Type window so that you can plot the data in the table that you have in focus.

Filter Node Example

This example compares and contrasts the results of two different Filter nodes using the SAMPSIO.HMEQ data set.

This example assumes that you have already created and opened a project diagram. Your completed process flow diagram will resemble the following image.




1. This example uses the SAMPSIO.HMEQ data set. To add the SAMPSIO.HMEQ data set, select **Help** ⇒ **Generate Sample Data Sources**. In the Generate Sample Data Sources window, select only the **Home Equity** data set. Click **OK**.

2. Drag the **Home Equity** data source to the diagram workspace. Select the **Home Equity** node and ensure that the **Role** property is set to **Train**.
3. From the **Sample** tab, drag two **Filter** nodes to the diagram workspace. Connect the **Home Equity** node to both of the **Filter** nodes.

By default, the first filter node is named **Filter**, and the second filter node is named **Filter (2)**.


4. Right-click the first **Filter** node and select **Run**. In the Confirmation window, select **Yes**. After the **Filter** node has run successfully, click **OK** in the Run Status window.
5. Right-click the second **Filter (2)** node and select **Run**. In the Confirmation window, select **Yes**. Once the **Filter** node has run successfully, click **OK** in the Run Status window.

6. Select the **Filter** node in the diagram workspace. In the **Interval Variables** section of the Properties Panel, click the  for the **Interval Variables** property.

In the Interactive Interval Filter window, select **YOJ**. Set the value of the **Filter Lower Limit** column to **-10**. Set the value of the **Filter Upper Limit** column to **20**.

Click **OK**.

7. For the **Filter** node, set the value of the **Default Filtering Method** to **User-Specified Limits**.

8. Select the **Filter (2)** node in the diagram workspace. In the **Interval Variables** section of the Properties Panel, click the  for the **Interval Variables** property.

In the Interactive Interval Filter window, select **YOJ**. Set the value of the **Filter Lower Limit** column to **20.0001**. Set the value of the **Filter Upper Limit** column to **50**.

Click **OK**.

9. For the **Filter (2)** node, set the value of the **Default Filtering Method** to **User-Specified Limits**.
10. From the **Model** tab, drag a **Regression** node to the Diagram Workspace. Connect the **Filter** node to the **Regression** node.
11. From the **Model** tab, drag a second **Regression (2)** node to the Diagram Workspace. Connect the **Filter (2)** node to the **Regression (2)** node.
12. From the **Assess** tab, drag a **Model Comparison** node to the Diagram Workspace. Connect both the **Regression** and **Regression (2)** nodes to the **Model Comparison** node.
13. Right-click the second **Model Comparison** node and select **Run**. In the Confirmation window, select **Yes**. After the **Model Comparison** node has run successfully, click **Results** in the Run Status window.

Observe that the two regression nodes have created different models based on the filtering done to the Yoj variable.

Close the Results window.

Chapter 4

Explore Data

Explore Nodes: Overview	41
-------------------------------	----

Explore Nodes: Overview

- The **Association** node enables you to identify association relationships within the data. For example, if a customer buys a loaf of bread, how likely is the customer to also buy a gallon of milk? The node also enables you to perform sequence discovery if a sequence variable is present in the data set.
- The **Cluster** node enables you to segment your data by grouping observations that are statistically similar. Observations that are similar tend to be in the same cluster, and observations that are different tend to be in different clusters. The cluster identifier for each observation can be passed to other tools for use as an input, ID, or target variable. It can also be used as a group variable that enables automatic construction of separate models for each group.
- The **DMDB** node creates a data mining database that provides summary statistics and factor-level information for class and interval variables in the imported data set. The DMDB is a metadata catalog used to store valuable counts and statistics for model building.
- The **Graph Explore** node is an advanced visualization tool that enables you to explore large volumes of data graphically to uncover patterns and trends and to reveal extreme values in the database. For example, you can analyze univariate distributions, investigate multivariate distributions, and create scatter and box plots and constellation and 3-D charts. Graph Explore plots are fully interactive and are dynamically linked to highlight data selections in multiple views.
- The **Link Analysis** node transforms unstructured transactional or relational data into a model that can be graphed. Such models can be used to discover fraud detection, criminal network conspiracies, telephone traffic patterns, website structure and usage, database visualization, and social network analysis. Also, the node can be used to recommend new products to existing customers.
- The **Market Basket** node performs association rule mining over transaction data in conjunction with item taxonomy. This node is useful in retail marketing scenarios that involve tens of thousands of distinct items, where the items are grouped into subcategories, categories, departments, and so on. This is called item taxonomy. The **Market Basket** node uses the taxonomy data and generates rules at multiple levels in the taxonomy.

- The **MultiPlot** node is a visualization tool that enables you to explore larger volumes of data graphically. The **MultiPlot** node automatically creates bar charts and scatter plots for the input and target variables without making several menu or window item selections. The code created by this node can be used to create graphs in a batch environment.
- The **Path Analysis** node enables you to analyze Web log data to determine the paths that visitors take as they navigate through a website. You can also use the node to perform sequence analysis.
- The **SOM/Kohonen** node enables you to perform unsupervised learning by using Kohonen vector quantization (VQ), Kohonen self-organizing maps (SOMs), or batch SOMs with Nadaraya-Watson or local-linear smoothing. Kohonen VQ is a clustering method, whereas SOMs are primarily dimension-reduction methods.
- The **StatExplore** node is a multipurpose node that you use to examine variable distributions and statistics in your data sets. Use the **StatExplore** node to compute standard univariate statistics, to compute standard bivariate statistics by class target and class segment, and to compute correlation statistics for interval variables by interval input and target. You can also use the **StatExplore** node to reject variables based on target correlation.
- The **Variable Clustering** node is a useful tool for selecting variables or cluster components for analysis. Variable clustering removes collinearity, decreases variable redundancy, and helps reveal the underlying structure of the input variables in a data set. Large numbers of variables can complicate the task of determining the relationships that might exist between the independent variables and the target variable in a model. Models that are built with too many redundant variables can destabilize parameter estimates, confound variable interpretation, and increase the computing time that is required to run the model. Variable clustering can reduce the number of variables that are required to build reliable predictive or segmentation models.
- The **Variable Selection** node enables you to evaluate the importance of input variables in predicting or classifying the target variable. The node uses either an R-square or a Chi-square selection (tree based) criterion. The R-square criterion removes variables that have large percentages of missing values, and remove class variables that are based on the number of unique values. The variables that are not related to the target are set to a status of rejected. Although rejected variables are passed to subsequent tools in the process flow diagram, these variables are not used as model inputs by modeling nodes such as the Neural Network and Decision Tree tools.

Chapter 5

Modify Input Data

Modify Nodes: Overview	43
Transform Variables	44
Overview of the Transform Variables Node	44
Transform Variables Node Properties	44
Transformation Methods	50
Setting the Transformation Method	55
Using the Formula Builder	57
Using the Expression Builder	60
Creating Interaction Variables	62
Transform Variables Node Results	63

Modify Nodes: Overview

- The **Drop** node enables you to drop selected variables from your scored SAS Enterprise Miner data sets. You can drop variables that have roles of Assess, Classification, Frequency, Hidden, Input, Rejected, Residual, and Target from your scored data sets. Use the **Drop** node to trim the size of data sets and metadata during tree analysis.
- The **Impute** node enables you to replace missing values for interval variables with the mean, median, midrange, mid-minimum spacing, distribution-based replacement, or use a replacement M-estimator such as Tukey's biweight, Hubers, Andrew's Wave, or by using a tree-based imputation method. Missing values for class variables can be replaced with the most frequently occurring value, distribution-based replacement, tree-based imputation, or a constant.
- The **Interactive Binning** node is used to model nonlinear functions of multiple modes of continuous distributions. The interactive tool computes initial bins by quintiles, and then you can split and combine the initial quintile-based bins into custom final bins.
- The **Principal Components** node enables you to perform a principal components analysis for data interpretation and dimension reduction. The node generates principal components that are uncorrelated linear combinations of the original input variables and that depend on the covariance matrix or correlation matrix of the input variables. In data mining, principal components are usually used as the new set of input variables for subsequent analysis by modeling nodes.
- The **Replacement** node enables you to replace selected values for class variables. The **Replacement** node summarizes all values of class variables and provides you

with an editable variables list. You can also select a replacement value for future unknown values.

- The **Rules Builder** node enables you to create ad hoc sets of rules for your data that result in user-definable outcomes. For example, you might use the **Rules Builder** node to define outcomes named Deny and Review based on rules such as the following:

```
IF P_Default_Yes > 0.4 then do
    EM_OUTCOME="Deny";
    IF AGE > 60 then
        EM_OUTCOME="Review";
    END;
```

- The **Transform Variables** node enables you to create new variables that are transformations of existing variables in your data. Transformations can be used to stabilize variances, remove nonlinearity, improve additivity, and correct nonnormality in variables. You can also use the **Transform Variables** node to transform class variables and to create interaction variables.

Transform Variables



Overview of the Transform Variables Node

The Transform Variables node enables you to create new variables or new variables that are transformations of existing variables in your data. The Transform Variables node also enables you to transform class variables and to create interaction variables.

Transformations are useful when you want to improve the fit of a model to the data. For example, transformations can be used to stabilize variances, remove nonlinearity, improve additivity, and correct nonnormality in variables.

The Transform Variables node can be connected to any predecessor node that exports a Raw or Train data set. To transform a transformed variable, you should add another Transform Variables node to the process flow. For example, you might want to transform the target and then transform selected input variables to maximize their relationship with the transformed target.

If you have several missing values for one or more variables, you might want to impute missing values with either the Impute node or the Cluster node, before you transform variables.


Transform Variables Node Properties

Transform Variables Node General Properties


The following general properties are associated with the Transform Variables node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Transform Variable


node added to a diagram will have a Node ID of Trans. The second Transform Variable node added to the diagram will have a Node ID of Trans2.

- **Imported Data** — The Imported Data property provides access to the Imported Data — Transform Variables window. The Imported Data — Transform Variables window contains a list of the ports that provide data sources to the Transform node. Select the  button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click:





- **Browse** to open a window where you can browse the data set.
 - **Explore** to open the Explore window, where you can sample and plot the data.
 - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Transform Variables window. The Exported Data — Transform Variables window contains a list of the output data ports that the Transform node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an imported data source, you can select the row in the imported data table and click:

- **Browse** to open a window where you can browse the data set.
 - **Explore** to open the Explore window, where you can sample and plot the data.
 - **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Transform Variables Node Train Properties

The following train properties are associated with the Transform Variables node:

- **Variables** — Use the Variables property to specify the properties of each variable that you want to use in the data source. Select the  button to open a variables table. You set the transformation method in the Variables table. For more information, see [“Setting the Transformation Method” on page 55](#).
- **Formulas** — Use the Formulas property to create customized transformations. Select the  button to open the [Formula Builder on page 57](#).
- **Interactions** — Use the Interactions property to [create interaction variables on page 62](#). Select the  button to open the Terms window.
- **SAS Code** — Select the  button to open the SAS Code window. You enter SAS code statements to create your own custom variable transformation in the SAS Code window. If you want to use code from a SAS catalog or external file, use the SAS Code window to submit a filename statement and a %include statement.

Transform Variables Node Train Properties: Default Methods

The following default method properties are associated with the Transform Variables node. See “Transformation Methods” on page 50 for more information about the transformation methods.

- **Interval Inputs** — Use the Interval Inputs property to specify the default transformation method that you want to apply to interval input variables. Each variable in the Variables table that uses the Default method will use the method that you specify in this property. The available methods are
 - **Best** — performs several transformations and uses the transformation that has the best Chi Squared test for the target.
 - **Multiple** — creates several transformations intended for use in successor Variable Selection nodes.
 - **Log** — transformed using the logarithm of the variable.
 - **Log 10** — transformed using the base-10 logarithm of the variable.
 - **Square Root** — transformed using the square root of the variable.
 - **Inverse** — transformed using the inverse of the variable.
 - **Square** — transformed using the square of the variable.
 - **Exponential** — transformed using the exponential logarithm of the variable.
 - **Centering** — centers variable values by subtracting the mean from each variable.
 - **Standardize** — standardizes the variable by subtracting the mean and dividing by the standard deviation.
 - **Range** — transformed using a scaled value of a variable equal to $(x - \min) / (\max - \min)$, where x is current variable value, \min is the minimum value for that variable, and \max is the maximum value for that variable.
 - **Bucket** — buckets are created by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.
 - **Quantile** — data is divided into groups with approximately the same frequency in groups.
 - **Optimal Binning** — data is binned in order to maximize the relationship to the target.
 - **Maximum Normal** — a best power transformation to maximize normality.
 - **Maximum Correlation** — a best power transformation to maximize correlation to the target. No transformation occurs if used on data sets with non-interval targets.
 - **Equalize** — a best power transformation to equalize spread with target levels.
 - **Optimal Max. Equalize** — an optimized best power transformation to equalize spread with target levels.
 - **None** — (default setting) No transformation is performed.
- **Interval Targets** — Use the Interval Targets property to specify the default transformation method that you want to use for interval target variables. Each interval target variable in the Variables table that uses the Default method will use the method that you specify in this property. The available methods are
 - **Best** — tries several transformations and selects the one that has the R-square value with the target.

- **Log** — transformed using the logarithm of the variable.
- **Log 10** — transformed using the base-10 logarithm of the variable.
- **Square Root** — transformed using the square root of the variable.
- **Inverse** — transformed using the inverse of the variable.
- **Square** — transformed using the square of the variable.
- **Exponential** — transformed using the exponential logarithm of the variable.
- **Centering** — centers variable values by subtracting the mean from each variable.
- **Standardize** — standardizes the variable by subtracting the mean and dividing by the standard deviation.
- **Range** — transformed using a scaled value of a variable equal to $(x - \min) / (\max - \min)$, where x is current variable value, \min is the minimum value for that variable, and \max is the maximum value for that variable.
- **Bucket** — buckets are created by dividing the data into evenly spaced intervals based on the difference between the maximum and minimum values.
- **Quantile** — data is divided into groups with approximately the same frequency in groups.
- **Optimal Binning** — data is binned in order to maximize the relationship to the target.
- **None** — (default setting) No transformation is performed.
- **Class Inputs** — Use the Class Inputs property to specify the default transformation method that you want to use for class input variables. The available methods are
 - **Group rare levels** — transformed using rare levels.
 - **Dummy Indicators** — transformed using dummy variables. Dummy variable coding is applied to the categorical variables from highest class value to lowest class value. For a variable x that contains values from 1 to 7, the transformation dummy variables that correspond with $x=1$ are named ti_x7 , transformation dummy variables that correspond with $x=2$ are named ti_x6 , and transformation dummy variables that correspond with $x=7$ are named ti_x1 .
 - **None** — (default) No transformation is performed.
- **Class Targets** — Use the Class Targets property to specify the default transformation method that you want to use for class target variables.
 - **Group rare levels** — transformed using rare levels.
 - **Dummy Indicators** — transformed using dummy variables.
 - **None** — (default) no transformation is performed
- **Treat Missing as Level** — Use the Treat Missing as Level property to indicate whether missing values should be treated as levels when creating groupings. If the Treat Missing as Level property is set to Yes, a group value will be assigned to missing values of the variable. If the Treat Missing as Level property is set to No, the group value will be set to missing for each variable.

Transform Variables Node Train Properties: Sample Properties

The Transform Variables node uses a sample to display the graphical distribution of variables in the Formula Builder. Use the following properties to specify how the sample is created.

- **Method** — Use the Method property to specify the sampling method that is used to create a sample.

The following methods are available:

- **First N** — When First N is selected, the top n observations are selected from the input data set for the sample. (Default)
- **Random** — When random sampling is selected, each observation in the data set (population) has the same probability of being selected for the sample, independently of the other observations that happen to fall into the sample.
- **Size** — Use the Size property to specify the number of observations that are used to generate the plot.
 - **Default** — The default value depends on the record length.
 - **Max** — The Max value uses the maximum number of observations that are downloadable. To change the Size value from Default to Max, the Method property must be set to Random.
- **Random Seed** — Use the Random Seed property to specify the seed that is used to generate the random sample. The default value is 12345.

Transform Variables Node Train Properties: Optimal Binning

- **Number of Bins** — Use the Number of Bins property to specify the number of bins to use when performing optimal binning transformations.
- **Missing Values** — Use the Missing property to specify how to handle missing values when you use an optimal binning transformation. Select from any of the available missing value policies.
 - **Separate Branch** — assigns missing values to its own separate branch.
 - **Use in Search** — uses missing values during the split search.
 - **Largest Branch** — assigns the observations that contain missing values to the branch that has the largest number of training observations.
 - **Branch with Smallest Residual** — assigns the observations that contain missing values to the branch that has the smallest value of residual sum of squares.

Transform Variables Node Train Properties: Grouping Method

- **Cutoff Value** — Group the levels with a very small occurrence (less than the specified cutoff percentage) in an `_OTHER_` category. The default setting is 0.1, meaning all levels that occurs less than 10% will be grouped into the `_OTHER_` category.
- **Group Missing** — Indicates whether missing values should be grouped into the `_OTHER_` category with rare levels. The default setting is **No**.
- **Number of Bins** — Use the Number of Bins property to specify the number of bins to use when performing bucket or quantile transformations.
- **Add Minimum Value to Offset Value** — The Add Minimum Value to Offset Value property is useful when transforming data sets that contain negative variable values. Negative variable values are incompatible with some transformation functions and can result in missing value entries in the output transformed data. If the data set to be transformed contains negative values, use the Add Minimum Value to Offset Value property to shift input data values by the absolute value of the most negative variable value.

In its default configuration of **Yes**, the Add Minimum Value to Offset Value property shifts the most negative variable value in a pre-transformed data set from a negative number to 0.0. When the Offset Value property is configured in its default setting of 1.0, the two properties together will shift variable values so that the smallest variable value in the data to be transformed is 1.0. This results in transformations with the smallest possible number of preventable missing variable entries. When the Add Minimum Value to Offset Value property is set to **No**, the pre-transformation data is shifted only by the value that is specified in the Offset Value property.

- **Offset Value** — The Offset Value property is useful when transforming data sets that contain negative variable values. Negative variable values are incompatible with some transformation functions and can result in missing value entries in the output transformed data. If the data set to be transformed contains negative values, use the Offset Value property to shift input data values. The Offset Value property and the Add Minimum Value to Offset Value property are additive when used together.

If Add Minimum Value to Offset Value property is set to **Yes** and the Offset Value property is set to 1.0, the pre-transform data will be shifted so that the smallest variable value to be transformed is 1.0. If Add Minimum Value to Offset Value property is set to **Yes** and the Offset Value property is set to 2.0, the pre-transform data will be shifted so that the smallest variable value to be transformed is 2.0. If the Add Minimum Value to Offset Value property is set to **No**, the pre-transform data is shifted only by the amount specified in the Offset Value property.

Transform Variables Node Score Properties

The following score properties are associated with the Transform Variables node:

- **Use Meta Transformation** — Use the Use Meta Transformation property to specify whether to use transformations that are defined in the Enterprise Miner Meta library.
- **Hide** — Use the Hide property to specify how to handle the original variables after a transformation is performed. Setting the Hide property to **No** if you want to keep the original variables in the exported metadata from your transformed data set. The default setting for the Hide property is **Yes**. When this property is set to **Yes**, the original variables are removed only from the exported metadata, and not removed from the exported data sets and data views.
- **Reject** — Use the Reject property to specify whether the model role of the original variables should be changed to Rejected or not. The default value for this property is **Yes**. To change the Reject value from **Yes** to **No**, you must set the Hide property value to **No**.

Transform Variables Node Report Properties

The following report property is associated with the Transform Variables node:

- **Summary Statistics** — Use the Summary Variables property to specify which variables have summary statistics computed. The default setting of **Yes** generates summary statistics on the transformed and new variables in the data set. The **No** setting does not generate any summary statistics.

Transform Variables Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.

- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Transformation Methods

Overview

The Transform Variables node supports various computed transformation methods. The available methods depend on the type and the role of a variable.

- For interval variables:
 - [“Simple Transformations” on page 50](#)
 - [“Binning Transformations” on page 51](#)
 - [“Best Power Transformations” on page 51](#)
- For class variables:
 - [“Group Rare Levels Transformations” on page 55](#)
 - [“Dummy Indicators Transformation” on page 55](#)

Simple Transformations

You can choose from the following simple transformations in the Transform Variables node:

- **Log** — Variable is transformed by taking the natural log of the variable.
- **Square Root** — Variable is transformed by taking the square root of the variable.
- **Inverse** — Variable is transformed by using the inverse of the variable.
- **Square** — Variable is transformed by using the square of the variable.
- **Exponential** — Variable is transformed by using the exponential logarithm of the variable.
- **Standardize** — Variable is standardized by subtracting the mean and dividing by the standard deviation.

Note: If the minimum value of a variable to be transformed is less than or equal to zero, then you should configure the node to add an offset value to the variable before transforming it. When you add an offset value to a variable, you specify a constant value that is added to every observation for that variable. The offset shift prevents invalid mathematic operations during transformation, such as dividing by zero or taking the square root of a negative number. When the node attempts to transform a variable with invalid values, the operation creates a missing variable value in your output transformed data set. Use a properly configured offset value to avoid preventable missing values in your transformed data output.

In most cases, you should offset variable values in an untransformed data set so that the smallest value of a variable is equal to 1. If the data set to be transformed contains negative variable values, you can use the Offset Value and Add Minimum Value properties to configure the Transform node for the range of values in your data. Use the

default Offset Value property setting of 1.0 and the default Add Minimum Value setting of Yes to minimize instances of preventable "missing value" data in your transformed data set.

Binning Transformations

Binning transformations enable you to collapse an interval variable, such as debt to income ratio, into an ordinal grouping variable. There are three types of binning transformations.

- [Bucket on page 52](#) — Buckets are created by dividing the data values into equally spaced intervals based on the difference between the maximum and the minimum values.
- [Quantile on page 52](#) — Data is divided into groups that have approximately the same frequency in each group.
- [“Optimal Binning for Relationship to Target Transformation” on page 52](#) — Data is binned in order to optimize the relationship to the target.

Best Power Transformations

The best power transformations are a subset of the general class of transformations that are known as Box-Cox transformations.

The Transform Variables node supports the following best power transformations:

- [“Maximize Normality Power Transformation” on page 52](#) — This method chooses the transformation that yields sample quantiles that are closest to the theoretical quantiles of a normal distribution.
- [“Maximize Correlation with Target Power Transformation” on page 53](#) — This method chooses the transformation that has the best squared correlation with the target. This method requires an interval target. If the maximize correlation transformation is attempted with a non-interval target, the data will not be transformed.
- [“Equalize Spread with Target Levels Power Transformation” on page 54](#) — This method chooses the transformation that has the smallest variance of the variances between the target levels. This method requires a class target.
- [“Equalize Spread with Target Levels Power Transformation” on page 54](#) — This method chooses the transformation that equalizes spread with target levels. This method requires a class target.

All of the Best Power transformations evaluate the transformation subset listed below, and choose the transformation that has the best results for the specified criterion. In the list below, x represents the transformed variable.

- x
- $\log(x)$
- \sqrt{x}
- e^x
- $x^{1/4}$
- x^2
- x^4

Note: Variables are scaled before they are transformed by the power transformations. Variables are scaled so that their values range from 0 to 1. The scaled value of the variable is equal to $(x - \min) / (\max - \min)$.

Bucket and Quantile Transformations

You have the flexibility to divide the data values into n equally spaced classes. The quantile transformation is useful when you want to create groups with approximately the same frequency in each group. For example, you might want to divide the variable values into 10 uniform groups (10th, 20th, 30th,.... percentiles).

Buckets are created by dividing the data values into n equally spaced intervals based on the difference between the minimum and maximum values. Unlike a quantile transformation, the number of observations in each bucket is typically unequal.

Optimal Binning for Relationship to Target Transformation

The **Optimal Binning for Relationship to Target** transformation optimally splits a variable into n groups with regard to a target variable. This binning transformation is useful when there is a nonlinear relationship between the input variable and the target. A nominal measurement level is assigned to the transformed variable.

To create the n optimal groups, the node uses a decision tree of depth 1. The Missing Value property specifies how missing values are handled. Missing values are either used as a value assuring the split search or are assigned to a node based on the selected transformation. For example, if Largest Branch is selected, all of the observations that have missing values for the splitting rule are placed in the branch with the largest number of training observations.

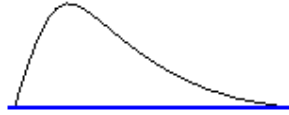
Maximize Normality Power Transformation

The **Maximize Normality** power transformation is useful when you want to normalize a variable that has a skewed distribution or an overly peaked or flat distribution. Skewness measures the deviation of the distribution from symmetry. A skewed distribution has a heavy tail or extreme values located on one side of the distribution. If the skewness statistic for a variable is clearly different from 0, then the distribution is asymmetrical. Normal distributions are perfectly symmetrical (bell shaped). After you run the Transform Variables node, you can see the skewness and kurtosis values in the Results - Transform Variables window. The skewness values for each variable are listed in the Skewness column of the Transformations Statistics table. The degree of flatness is measured by the value in the Kurtosis column, where a normal distribution has a value of 1.

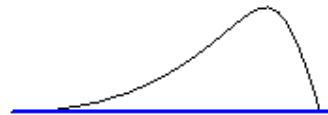
Note: Click Results from the Transform Variables pop-up menu to see the window.

The degree of normality obtained by applying the Maximize Normality transformation to a skewed variable is naturally data dependent.

Skewed to the Right
Positive Skewness

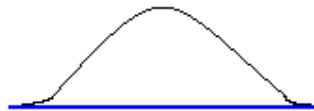


Skewed to the Left
Negative Skewness



Apply the Maximize Normality Transformation

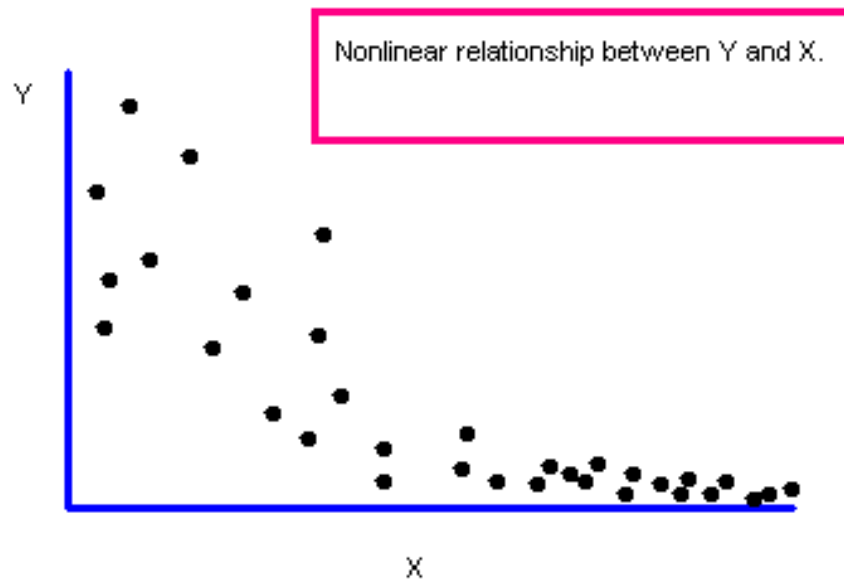
Approximately Symmetric
Skewness Closer to 0



Maximize Correlation with Target Power Transformation

The **Maximize Correlation with Target** power transformation enables you to straighten (linearize) the relationship between an interval target and an interval input variable. If the Maximize correlation with target setting is used with a non-interval target, the data will not be transformed when the node runs. Although neural networks and decision tree methods are quite capable of predicting nonlinear relationships, increasing the correlation with the target tends to simplify the relationship between the target and the input variable, making it more easily understood and communicable to others. The overall model fit is also often improved by linearizing the data.

In the following scatter plot, notice that the relationship between Y and X is nonlinear. An additional problem is also evident in that the variance in Y also decreases as the value of X increases. The Maximize correlation with target transformation might help straighten the relationship between Y and X and in turn stabilize the variance in Y across the different levels of X.



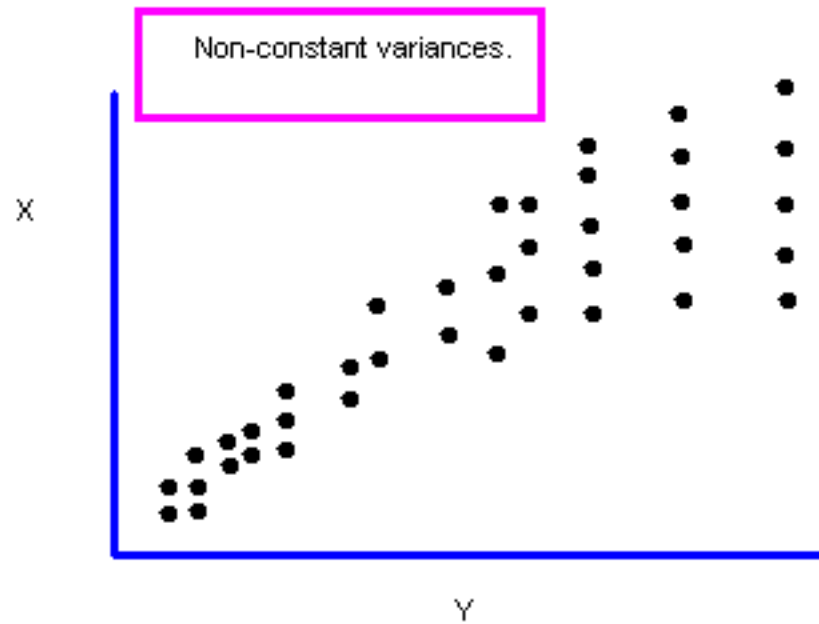
Equalize Spread with Target Levels Power Transformation

The **Equalize Spread with Target Levels** transformation helps stabilize the variance in the interval input across the different levels of a binary, nominal, or ordinal target.

Ordinary least squares estimation assumes that the errors are additive, and that they are normally independent random variables with a common variance. When the assumption of independence and common variance hold, least squares estimators have the desirable property of being the best (minimum variance) among all possible linear unbiased estimators.

The assumption of common variance implies that every observation on the target contains the same amount of information. Consequently, all observations in ordinary least squares receive the same weight. Unfortunately, equal weighting does not give the minimum variance estimates of the parameters if the variances are not equal. The direct impact of heterogeneous variances is a loss of precision in the parameter estimates compared to the precision that would have been realized if the heterogeneous variances had been stabilized using an appropriate transformation.

In the following scatter plot, although there is a linear relationship between Y and X, the variance increases as the values of the target Y increase. The variance is not equal across the different levels of the target Y. The Equalize Spread with Target Levels transformation might help reduce the spread among the target levels and in turn provide more stable parameter estimates.



Group Rare Levels Transformations


The Group Rare Levels transformation is available for class variables only. This method combines the rare levels (if any) into a separate group, `_OTHER_`. You use the **Cutoff Value** property to specify the cutoff percentage. Rare levels are the variable values that occur less than the specified cutoff value.

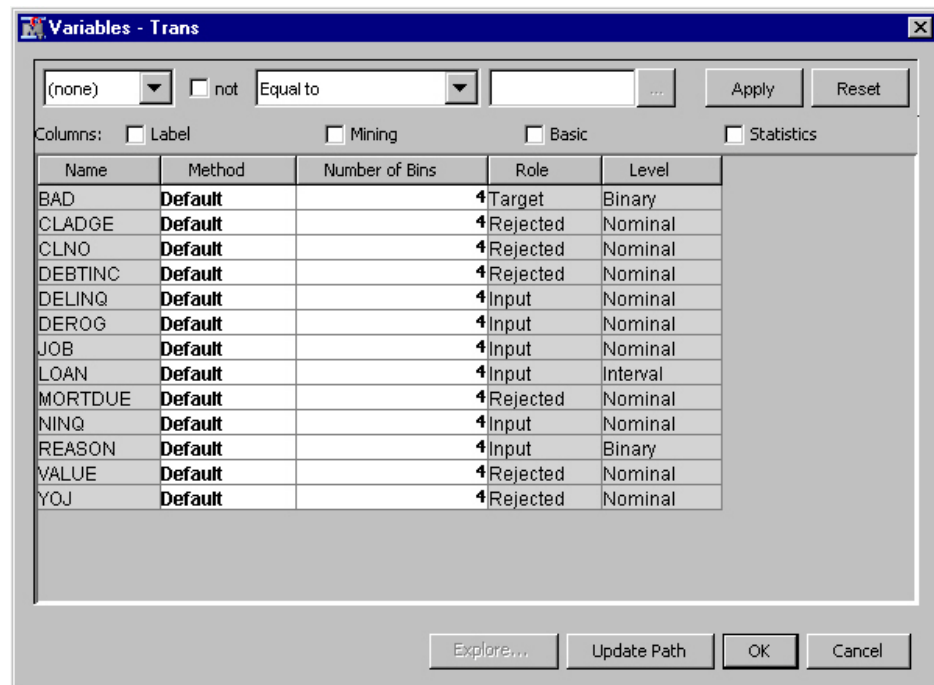
Dummy Indicators Transformation

The Dummy Indicators Transformation is available for class variables only. This method creates a dummy indicator variable (0 or 1) for each level of the class variable.

Setting the Transformation Method

Configuring the Variables Window

You configure the transformation method that you want to use for your data set using the table in the Variables window. You open the Variables window by selecting the  button by the Variables property in the Enterprise Miner properties panel. The following is a display of the Variables window.



The table in the Variables window has the following columns. You can edit only the Method and Number of Bins columns.

- **Name** — name of the variable
- **Method** — transformation method assigned to variable
- **Number of Bins** — number of bins for binning interval variables when the use methods such as Bucket or Quantile. For more information, see [“Setting the Number of Bins” on page 57](#).
- **Role** — variable role
- **Level** — level for class variables

You can add additional columns by selecting one or more check boxes.

Select the **Label** check box to add the following columns:

- **Label** — adds a column for a label for each variable.

Select the **Mining** check box to add the following columns:

- **Order** — variable order
- **Report** — Boolean setting for creating reports.
- **Lower Limit** — minimum value for variable
- **Upper Limit** — maximum value for variable
- **Creator** — user who created the process flow diagram
- **Comment** — user-supplied comments about a variable
- **Format Type** — variable format

Select the **Basic** check box to add the following columns:

- **Type** — variable type
- **Format** — SAS Format for variable

- **Informat** — SAS Informat for variable
- **Length** — Length of the variable

Select the **Statistics** check box to add the following columns:


- **Number of Levels** — displays the number of levels for class variables only.
- **Percent Missing** — displays the percent of missing values. For variables with no missing values, 0 is displayed.
- **Minimum** — displays the minimum values for numeric variables only. For character variables, . is displayed.
- **Maximum** — displays the maximum values for numeric variables only. For character variables, . is displayed.
- **Mean** — displays the arithmetic mean values for numeric variables only. For character variables, . is displayed.
- **Standard Deviation** — displays the standard deviation values for numeric variables only. For character variables, . is displayed.
- **Skewness** — displays the skewness for numeric variables only. For character variables, . is displayed.
- **Kurtosis** — displays the kurtosis values for numeric variables only. For character variables, . is displayed.

Setting the Number of Bins

If you select a binning transformation such as **bucket**, **quantile**, or **optimal**, then you can set the maximum number of bins in the Number of Bins column. To set the maximum number of bins, select the field in the Number of Bins column that corresponds to the variable, and enter the number of bins that you want to use . You must hit the Enter key after changing the number of bins in a cell for Enterprise Miner to accept the new value. The default value for the maximum number of bins is 4. With some data sets, Enterprise Miner might find less than the specified maximum number of bins.

Using the Formula Builder

Overview

The Formula Builder enables you to create customized transformations from the input variables. To open the Formulas window, select the  button for the Formulas property in the properties panel.

The Formulas window contains the following tabs:

- “Inputs Tab” on page 58
- “Outputs Tab” on page 59
- “Sample Tab” on page 60
- “Log Tab” on page 60

Note: The term, a *new variable*, refers to a customized transformation of the original variables in the input data source.

You can customize the layout of these tabs. Select the right arrow on the tab to view the contents of a tab in its own pane to the right of other tabs. Select the down arrow on the tab to view the contents of a tab in its own pane below other tabs.

Inputs Tab

The top portion of the Formulas window displays the distribution of the variable that is selected in the table below.

The bottom portion of the window displays the following information about the original variables:

- **Name** — name of the variable
- **Method** — transformation method assigned to variable
- **Number of Bins** — number of bins for binning interval variables when the use methods such as Bucket or Quantile.
- **Role** — variable role
- **Level** — level for class variables

You can add additional columns by selecting one or more check boxes. Select the Label check box to add a text label to be used for the variable in graphs and output.

Select the **Mining** check box to add the following columns:

- **Order** — variable order
- **Report** — Boolean setting for creating reports.
- **Lower Limit** — minimum value for variable
- **Upper Limit** — maximum value for variable
- **Creator** — user who created the process flow diagram
- **Comment** — user-supplied comments about a variable
- **Format Type** — variable format

Select the **Basic** check box to add the following columns:

- **Type** — variable type
- **Format** — SAS Format for variable
- **Informat** — SAS Informat for variable
- **Length** — Length of the variable

Select the **Statistics** check box to add the following columns:

- **Number of Levels** — displays the number of levels for class variables only.
- **Percent Missing** — displays the percent of missing values. For variables with no missing values, 0 is displayed.
- **Minimum** — displays the minimum values for numeric variables only. For character variables, . is displayed.
- **Maximum** — displays the maximum values for numeric variables only. For character variables, . is displayed.
- **Mean** — displays the arithmetic mean values for numeric variables only. For character variables, . is displayed.
- **Standard Deviation** — displays the standard deviation values for numeric variables only. For character variables, . is displayed.
- **Skewness** — displays the skewness for numeric variables only. For character variables, . is displayed.

- **Kurtosis** — displays the kurtosis values for numeric variables only. For character variables, . is displayed.


Outputs Tab

Here is an example display of the Outputs tab.

- To view the distribution of a new variable, select a variable in the table and click **Preview**.

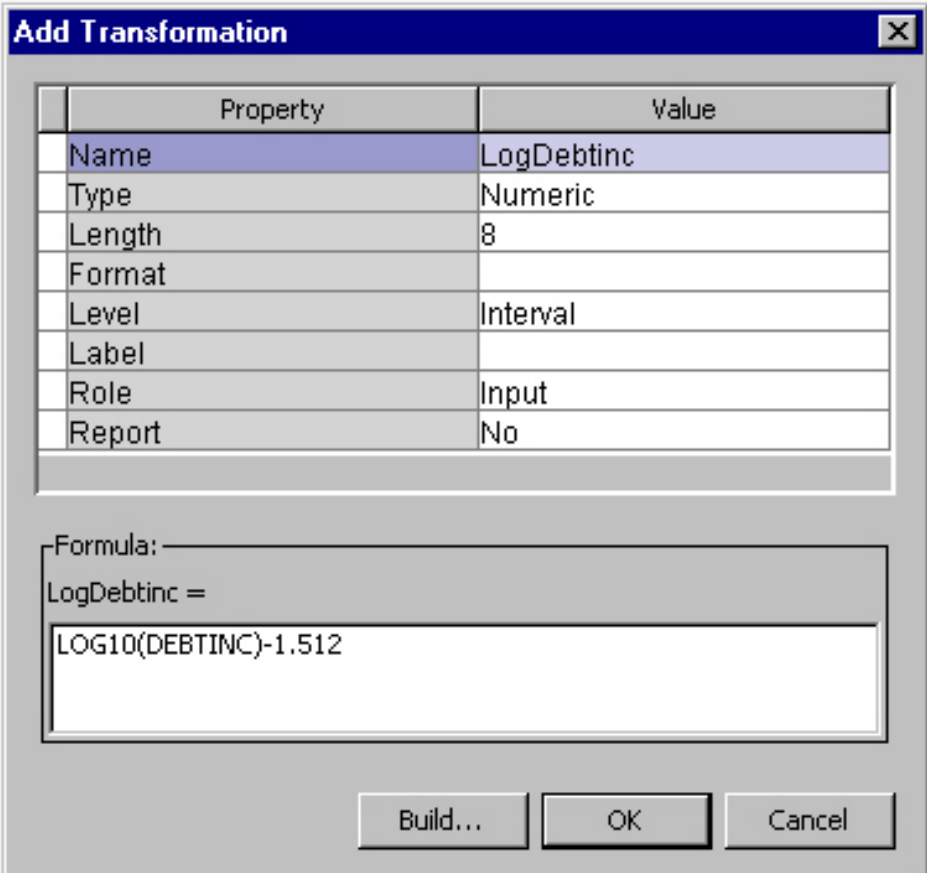
Note: A sample of the input data source is used to generate the distribution chart.

Use the “[Transform Variables Node Train Properties](#)” on page 45 to specify the settings that are used to generate the sample.

- To create a customized transformation, click the Create icon () at the top of the

Formulas window and the Add Transformation window appears. Specify a variable name, type (Numeric or Character with drop-down menu), variable length, format, level, label, role, and report setting. Specify an expression for the new variable either by entering the formula in the Formula box or by “[Using the Expression Builder](#)” on page 60. To open the Expression Builder, click Build.

In the following example, the variable that is called LogDebtinc is created from the existing variable DEBTINC.




Property	Value
Name	LogDebtinc
Type	Numeric
Length	8
Format	
Level	Interval
Label	
Role	Input
Report	No

Formula:




LogDebtinc =

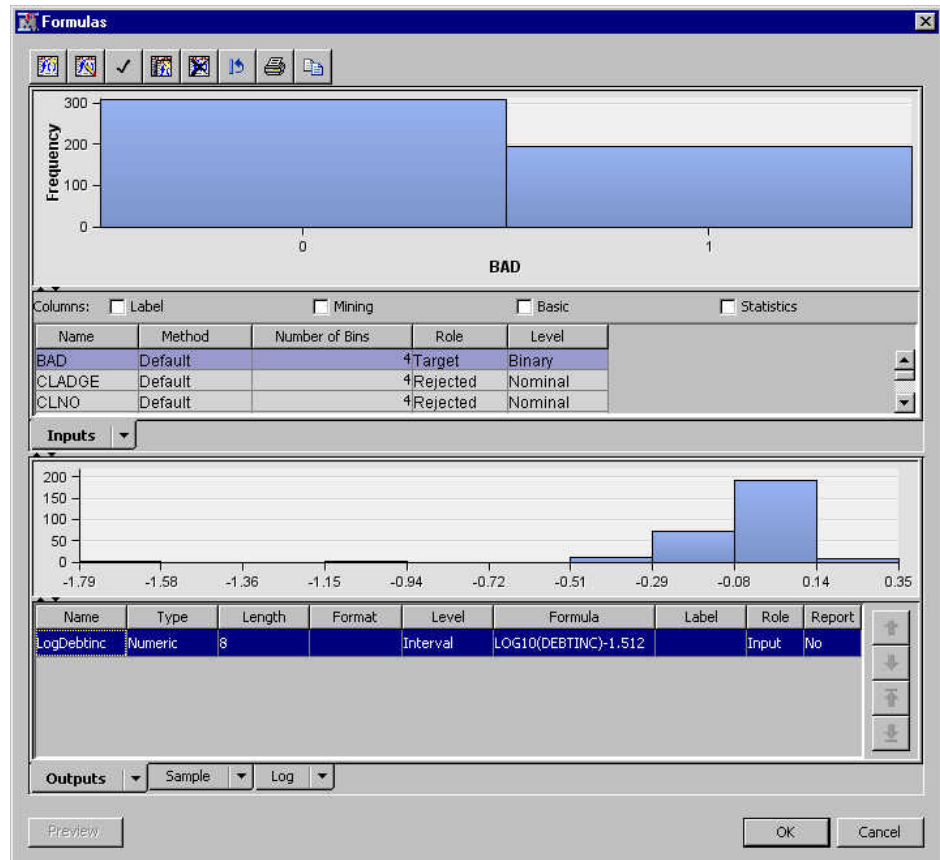
LOG10(DEBTINC)-1.512

Build... OK Cancel

- To modify the properties of a customized transformation, click the Edit properties icon () at the top of the Formulas window and the Edit Transformation

window appears. You can change the variable name, type (Numeric or Character with drop-down menu), variable length, format, level, label, role, and report setting.

- To modify the definition of a customized transformation, click the Edit expression icon () at the top of the Formulas window and the Expression Builder window appears. The expression can be modified as needed.
- To create a customized transformation from an existing customized transformation, select the existing variable in the table and click the Duplicate icon (). The variable name will be generated automatically.
- To delete a variable, select it in the table and click the Delete icon ().



Sample Tab

The Sample tab displays a sample of the observations in the data set, including transformed variables.

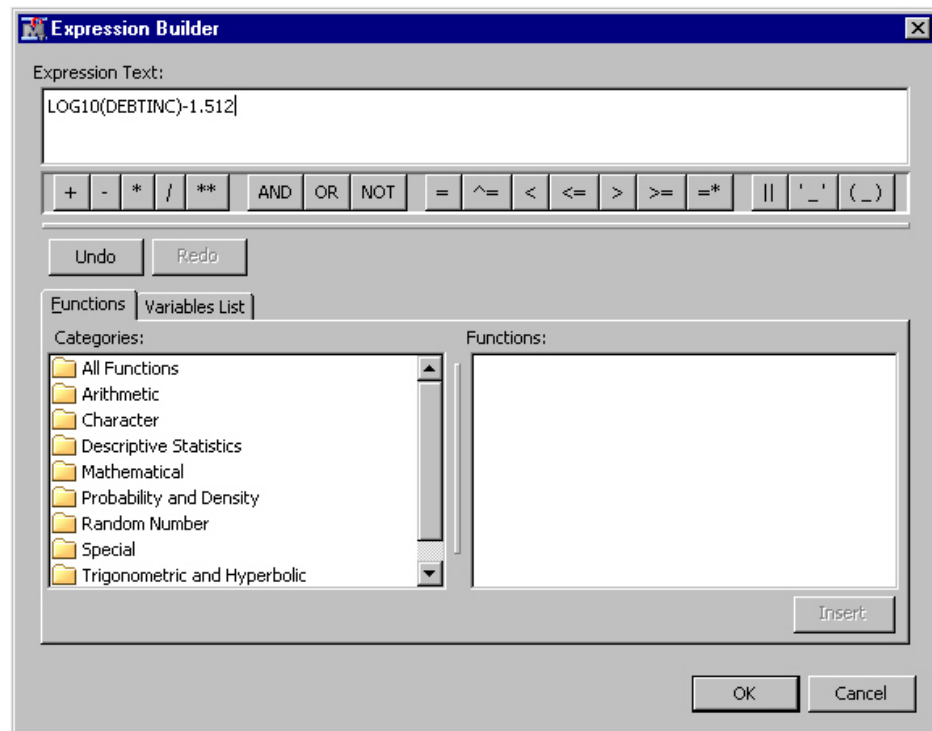
Log Tab

The Log tab displays the contents of the log that is generated when you create customized transformation.

Using the Expression Builder

Overview

You use the Expression Builder to define an expression.



The Expression Builder window contains the following elements:

- **Expression text** — specifies the combination of functions and mathematical operations that are used to derive a value. To enter an expression, you can type directly in the field or use the operator toolbar, the Functions tab, and the Variables List tabs to add operators, functions, and data to your expression. To clear the expression, use the Backspace or Delete keys.
- **Operator toolbar** — contains symbols that you can add to the expression including arithmetic operators, comparison operators, and logical operators.
- **Functions tab** — displays the list of available functions that you can use to create expressions. The functions are categorized by type. To add a function to the expression, double-click an item in the Functions tab, or select the item and click Insert.
- **Variables List tab** — displays a list of variables in the input data source that you can add to the expression. To add a variable to the expression, select the item and click Insert.
- **Insert** — inserts selected functions or data elements into the expression. This button is enabled only when you select a function or data source that can be inserted into the expression.

Functions Tab

Use the Functions tab to add functions to the expression. The Functions tab displays the list of available functions that you can use to create expressions. The functions are categorized by type. The list of available functions is determined by the underlying structure of your physical data.

- If you are building an expression that is based on relational data, then the functions that are supported by the SAS SQL procedure are displayed. For information about these functions, see "Functions and CALL Routines" in the SAS Language

Reference: Dictionary and "Summarizing Data" in the SAS SQL Procedure User's Guide.

- If you are building an expression that is based on OLAP data, then the MDX functions that are supported by SAS are displayed. For information about these functions, see "MDX Functions" in the SAS OLAP Server: MDX Guide.
- To specify a function that is specific to a different database system, manually enter the function name in the Expression Text field.

The Functions tab contains the following items:

- **Categories** — displays the different types of functions that you can select to add to the expression. When you select a category, a list of functions for that category appears in the **Functions** list box.
- **Functions** — displays the specific functions that you can add to the expression. You can add a function to the expression by double-clicking the selected function or by selecting it and then clicking **Insert**.

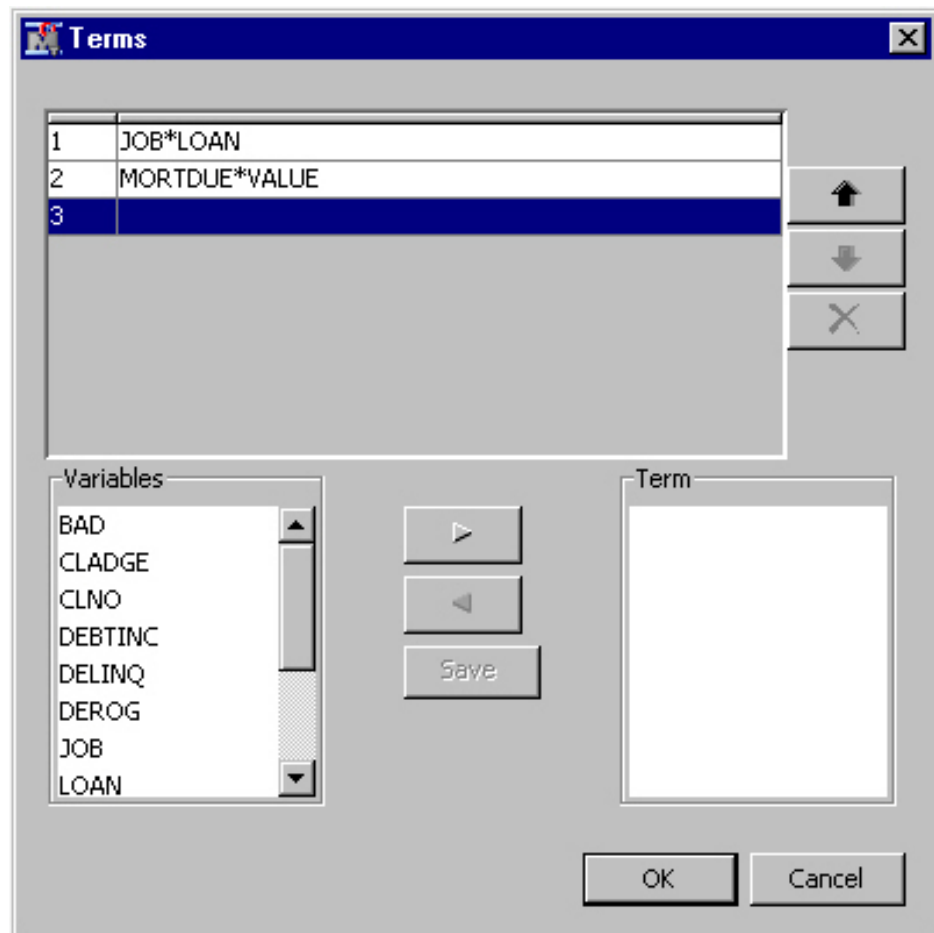
Variables List Tab

Use the Variables List tab to add variables to the expression. To add a variable, select the variable and click **Insert**.

Creating Interaction Variables

One of the enhancements of the Transform Variables node since Enterprise Miner 4.3 is the ability to create interaction variables in the Terms window. To open the Terms

window, select the  button for the Interactions property in the properties panel.



An interaction term is a product of existing explanatory inputs. For example, the interaction of the variables JOB and LOAN is JOB*LOAN.

The **Variables** list contains the variables in the input data source. The interaction variable is the product of the variables in the **Term** list. Follow these steps to create an interaction variable.

1. Select the variables in the Variables list.
2. Use the right arrow to move the selected variables to the Term list.
3. Click Save and the interaction variable will be displayed at the top portion of the Terms window.

You use the up and down arrows to move the interaction variables. To delete an

interaction variable, select an interaction variable and click



Note: When you have a Merge node before the Transform Variables node, you must run the Merge node first before you can invoke the Interactions editor.

Transform Variables Node Results

You can open the Results window of the Transform Variables node by right-clicking the node and selecting **Results** from the pop-up menu.

Select **View** from the main menu to view the following results.

- **Properties**
 - **Settings** — displays a window with a read-only table of the Transform Variables node properties configuration when the node was last run.
 - **Run Status** — indicates the status of the Transform Variables node run. Information about whether the run completed successfully, the Run Start Time, Run Duration, and the Run ID are displayed in this window.
 - **Variables** — a table of the variables property of the node. (Not all variables appear in this table.) You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
 - **Train Code** — Training is not available in the Transform Variables node.
 - **Notes** — displays any user-created notes of interest.
- **SAS Results**
 - **Log** — the SAS log of the Transform Variables node run.
 - **Output** — the SAS output of the Transform Variables node run. The SAS output displays a variable summary table, a transformations table by input variable, summary statistics for input interval and class variables, as well as summary statistics for output interval and class variables.
 - **Flow Code** — the SAS code used to produce the output that the Transform Variables node passes on to the next node in the process flow diagram.
- **Scoring**
 - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the Enterprise Miner environment in custom user applications. If no transformed values were generated, the SAS Code menu item is dimmed and unavailable.
 - **PMML Code** — The Transform Variables code does not generate PMML code.
- **Transformations**
 - **Formula** — Opens a read-only table that displays user-defined transformations and their basic statistics.
 - **Computed** — Opens a read-only table that displays computed transformations and their basic statistics.
 - **EM Meta** — Opens a read-only table that displays the transformations that were created using the Enterprise Miner Meta library.
 - **SAS Code** — Opens a read-only table that displays the transformations and their basic statistics that were created using SAS Code.
 - **Transformations Statistics** — displays the Transformations Statistics table.
- **Table** — open the data table that corresponds to the graph that you have in focus.
- **Plot** — opens the Select a Chart Type window that you can use to create a custom plot of the data in the table that you have in focus.

Chapter 6

Model Data

Model Nodes: Overview	65
-----------------------------	----

Model Nodes: Overview

- The **AutoNeural** node can be used to automatically configure a neural network. The **AutoNeural** node implements a search algorithm to incrementally select activation functions for a variety of multilayer networks.
- The **Decision Tree** node enables you to fit decision tree models to your data. The implementation includes features found in a variety of popular decision tree algorithms (for example, CHAID, CART, and C4.5). The node supports both automatic and interactive training. When you run the Decision Tree node in automatic mode, it automatically ranks the input variables based on the strength of their contribution to the tree. This ranking can be used to select variables for use in subsequent modeling. You can override any automatic step with the option to define a splitting rule and prune explicit tools or subtrees. Interactive training enables you to explore and evaluate data splits as you develop them.
- The **Dmine Regression** node enables you to compute a forward stepwise least squares regression model. In each step, the independent variable that contributes maximally to the model R-square value is selected. The tool can also automatically bin continuous terms.
- The **DMNeural** node is another modeling node that you can use to fit an additive nonlinear model. The additive nonlinear model uses bucketed principal components as inputs to predict a binary or an interval target variable with automatic selection of an activation function.
- The **Ensemble** node enables you to create new models by combining the posterior probabilities (for class targets) or the predicted values (for interval targets) from multiple predecessor models.
- The **Gradient Boosting** node uses tree boosting to create a series of decision trees that together form a single predictive model. Each tree in the series is fit to the residual of the prediction from the earlier trees in the series. The residual is defined in terms of the derivative of a loss function. For squared error loss with an interval target, the residual is simply the target value minus the predicted value. Boosting is defined for binary, nominal, and interval targets.
- The **LARS** node enables you to use Least Angle Regression algorithms to perform variable selection and model fitting tasks. The **LARS** node can produce models that range from simple intercept models to complex multivariate models that have many

inputs. When using the **LARs** node to perform model fitting, the node uses criteria from either least angle regression or the LASSO regression to choose the optimal model.

- The **MBR** (Memory-Based Reasoning) node enables you to identify similar cases and to apply information that is obtained from these cases to a new record. The **MBR** node uses k-nearest neighbor algorithms to categorize or predict observations.
- The **Model Import** node enables you to import models into the SAS Enterprise Miner environment that were not created by SAS Enterprise Miner. Models that were created by using SAS PROC LOGISTIC (for example) can now be run, assessed, and modified in SAS Enterprise Miner.
- The **Neural Network** node enables you to construct, train, and validate multilayer feedforward neural networks. Users can select from several predefined architectures or manually select input, hidden, and target layer functions and options.
- The **Partial Least Squares** node is a tool for modeling continuous and binary targets based on SAS/STAT PROC PLS. The **Partial Least Squares** node produces DATA step score code and standard predictive model assessment results.
- The **Regression** node enables you to fit both linear and logistic regression models to your data. You can use continuous, ordinal, and binary target variables. You can use both continuous and discrete variables as inputs. The node supports the stepwise, forward, and backward selection methods. A point-and-click interaction builder enables you to create higher-order modeling terms.
- The **Rule Induction** node enables you to improve the classification of rare events in your modeling data. The **Rule Induction** node creates a Rule Induction model that uses split techniques to remove the largest pure split node from the data. Rule Induction also creates binary models for each level of a target variable and ranks the levels from the most rare event to the most common. After all levels of the target variable are modeled, the score code is combined into a SAS DATA step.
- The **TwoStage** node enables you to compute a two-stage model for predicting a class and an interval target variable at the same time. The interval target variable is usually a value that is associated with a level of the class target.

Chapter 7

Assess Model Performance

Assess Nodes: Overview	67
------------------------------	----

Assess Nodes: Overview

- The **Cutoff** node provides tabular and graphical information to help you determine the best cutoff point or points for decision making models that have binary target variables.
- The **Decisions** node enables you to define target profiles to produce optimal decisions. You can define fixed and variable costs, prior probabilities, and profit or loss matrices. These values are used in model selection steps.
- The **Model Comparison** node provides a common framework for comparing models and predictions from any of the modeling tools (such as Regression, Decision Tree, and Neural Network tools). The comparison is based on standard model fits statistics as well as potential expected and actual profits or losses that would result from implementing the model. The node produces the following charts that help describe the usefulness of the model: lift, profit, return on investment, receiver operating curves, diagnostic charts, and threshold-based charts.
- The **Score** node enables you to manage, edit, export, and execute scoring code that is generated from a trained model. Scoring is the generation of predicted values for a data set that cannot contain a target variable. The **Score** node generates and manages scoring formulas in the form of a single SAS DATA step, which can be used in most SAS environments even without the presence of SAS Enterprise Miner.
- The **Segment Profile** node enables you to assess and explore segmented data sets. Segmented data is created from data BY-values, clustering, or applied business rules. The **Segment Profile** node facilitates data exploration to identify factors that differentiate individual segments from the population, and to compare the distribution of key factors between individual segments and the population. The **Segment Profile** node creates a Profile plot of variable distributions across segments and population, a Segment Size pie chart, a Variable Worth plot that ranks factor importance within each segment, and summary statistics for the segmentation results. The **Segment Profile** node does not generate score code or modify metadata.

Chapter 8

Utilities

Utility Nodes: Overview	70
Start Groups	71
Overview of the Start Groups Node	71
Start Groups Node and Missing Data Set Values	71
Using the Start Groups Node	72
Start Groups Node and the Ensemble Node	72
Start Groups Node Looping	73
Start Groups Node and Bag Bite Sampling	74
Start Groups Node Properties	75
Start Groups Node Results	78
Start Groups Node Example	79
End Groups	81
Overview of the End Groups Node	81
End Groups Node Algorithm	81
Using the End Groups Node	82
End Groups Node Properties	82
End Groups Node Results	83
End Groups Node Example	84
Metadata	85
Overview of the Metadata Node	85
Metadata Node Properties	86
Metadata Node Variables Table	88
Metadata Node Results	90
SAS Code	90
Overview of the SAS Code Node	90
SAS Code Node Properties	91
Code Editor Overview	94
Code Editor User Interface	94
Code Editor Macros	100
Macro Variables	108
Code Pane	115
SAS Code Node Results	117
Score Code Export	118
Overview of the Score Code Export Node	118
Data Requirements of the Score Code Export Node	119
Properties of the Score Code Export Node	119
Score Code Export Node Results	121
Score Code Node Output	122
SAS Enterprise Miner Tools Production of Score Code	124

Utility Nodes: Overview

- The **Control Point** node establishes a nonfunctional connection point to clarify and simplify process flow diagrams. For example, suppose three Input Data nodes are to be connected to three modeling nodes. If no Control Point node is used, then nine connections are required to connect all of the Input Data nodes to all of the modeling nodes. However, if a Control Point node is used, only six connections are required.
- The **End Groups** node terminates a group processing segment in the process flow diagram. If the group processing function is stratified, bagging, or boosting, the **Ends Groups** node will function as a model node and present the final aggregated model. (Ensemble nodes are not required as in SAS Enterprise Miner 4.3.) Nodes that follow the **Ends Groups** node continue data mining processes normally.
- The **ExtDemo** node illustrates the various UI elements that can be used by SAS Enterprise Miner extension nodes.
- The **Metadata** node enables you to modify the columns metadata information at some point in your process flow diagram. You can modify attributes such as roles, measurement levels, and order.
- The **Open Source Integration** node enables you to write code in the R language inside SAS Enterprise Miner. The node makes SAS Enterprise Miner data and metadata available to your R code, and returns R results to SAS Enterprise Miner. In addition to training and scoring supervised and unsupervised R models, the **Open Source Integration** node allows for data transformation and data exploration.
- The **Register Model** node enables you to register segmentation, classification, or prediction models to the SAS Metadata Server. Models registered in metadata can be submitted to SAS Model Manager, used to score data in SAS Enterprise Guide, or used to score data in SAS Enterprise Miner. Information about input variables, output variables, target variables, target levels, mining function, training data, and SAS score code is registered to the metadata.
- The **Reporter** node tool uses SAS Output Delivery System (ODS) capabilities to create a single document for the given analysis in PDF or RTF format. The document includes important SAS Enterprise Miner results, such as variable selection, model diagnostic tables, and model results plots. The document can be viewed and saved directly and will be included in SAS Enterprise Miner report package files.
- The **SAS Code** node tool enables you to incorporate SAS code into process flows that you develop using SAS Enterprise Miner. The **SAS Code** node extends the functionality of SAS Enterprise Miner by making other SAS System procedures available in your data mining analysis. You can also write a SAS DATA step to create customized scoring code, to conditionally process data, and to concatenate or to merge existing data sets.
- The **SAS Viya Code** node enables you to incorporate SAS Viya code into process flow diagrams that were developed using SAS Enterprise Miner. The **SAS Viya Code** node is a modified **SAS Code** node for SAS Viya and SAS CAS environments. The node is template-based and is populated with basic training code that you can modify. The templates include utility macros that can simplify integrating your code with SAS Enterprise Miner.
- The **Save Data** node enables you to save training, validation, test, score, or transaction data from a node to either a previously defined SAS library or a specified file path. The node can export JMP, Excel 2010, CSV, and tab-delimited files. The

default options are designed so that the node can be deployed in SAS Enterprise Miner batch programs without user input.

- The **Score Code Export** node tool enables you to extract score code and score metadata to an external folder. The **Score Code Export** node must be preceded by a Score node.
- The **Start Groups** node initiates a group processing segment in the process flow diagram. The **Start Groups** node performs the following types of group processing:

Stratified group processing that repeats processes for values of a class variable, such as GENDER=M and GENDER=F.

Bagging, or bootstrap aggregation via repeated resampling.

Boosting, or boosted bootstrap aggregation, using repeated resampling with residual-based weights.

Index processing, which repeats processes a fixed number of times. Index processing is normally used with a Sampling node or with user's code for a sample selection.

Start Groups



Overview of the Start Groups Node

The **Start Groups** node is located on the **Utility** tab of the SAS Enterprise Miner tools bar. The **Start Groups** node is a descendant of the SAS Enterprise Miner 4.3 **Group Processing** node. The **Start Groups** node is useful when your data can be segmented or grouped, and you want to process the grouped data in different ways. The **Start Groups** node uses BY-group processing as a method to process observations from one or more data sources that are grouped or ordered by values of one or more common variables. BY variables identify the variable or variables by which the data source is indexed, and BY statements process data and order output according to the BY group values.

You can use the SAS Enterprise Miner **Start Groups** node to do the following tasks:

- define group variables such as GENDER or JOB, in order to obtain separate analyses for each level of group variable
- analyze more than one target variable in the same process flow
- specify index looping, or how many times the flow that follows the node should loop
- resample the data set and use unweighted sampling to create bagging models
- resample the training data set and use reweighted sampling to create boosting models

Start Groups Node and Missing Data Set Values

The **Start Groups** node processes observations that have missing values as valid group values. If the input data set that you want to perform group processing on contains a significant number of observations with missing values, it might be beneficial to use the

Replacement or Impute nodes to replace or impute missing variable values before submitting the data set to the **Start Groups** node.

Using the Start Groups Node

The **Start Groups** node must be used in a process flow diagram that has one or more data sources. If you import more than one data set, the data sets must be compatible with each other. For example, if you import a training data set into the **Start Groups** node that uses a group variable named REGION, then the group variable REGION must also be in your score data set. When you import more than one data set (for example, a training, a validation, and a score data set), then group processing is performed on each data set when you run the process flow.

The **Start Groups** node requires at least one target variable to run. The **Start Groups** node can process more than one target variable. The **Start Groups** node processes all variables that have a role of target as targets.

In a process flow diagram, the group processing portion of the process flow diagram is defined by the **Start Groups** node, the **End Groups** node, and the data mining node tools that you place between the **Start Groups** node and the **End Groups** node. The **Start Groups** node is followed by one or more successor nodes that perform some data mining operation. The **End Groups** node defines the scope of the group processing operations. The **Start Groups** node will not run without an accompanying **End Groups** node. The **End Groups** node also accumulates data mining results for group processing reporting.

You cannot perform group processing on Transaction data sets in SAS Enterprise Miner.

You can connect any node to the **Start Groups** node as a successor, but only modeling nodes and the **Score** node accumulate results during individual loop iterations. If you want to accumulate the results of each loop iteration, you can use a successor **SAS Code** node to capture and record the intermediate values.

It is possible to have more than one **Start Groups** and **End Groups** node pair in a process flow, as long as each group processing operation (within a **Start Groups** and **End Groups** node pair) is performed serially. In other words, each group processing operation must complete before the subsequent group processing operation begins. SAS Enterprise Miner cannot run more than one group processing operation at a time. As a result, you cannot nest a group processing operation within a group processing operation. The restriction also means that you also should not design process flow diagrams that perform group processing operations in competing parallel branches.

Start Groups Node and the Ensemble Node

When using the **Start Groups** node to perform bagging or boosting in SAS Enterprise Miner 4.3, it was necessary to place an **Ensemble** node at the end of the group processing portion of the process flow diagram in order to produce the final bagging or boosting model or models.

In SAS Enterprise Miner 14.2, the **Start Groups** node does not require an **Ensemble** node to perform bagging or boosting. The **End Groups** node that terminates the group processing portion of all SAS Enterprise Miner 14.2 group processing diagrams also contains the code that is required to produce the final bagging or boosting model or models.

Start Groups Node Looping

The **Start Groups** node processes data using looping modes that cause the group processing portion of the process flow diagram to repeat a number of times. The looping modes for the **Start Groups** node are as follows:

- **Index** — the index mode setting specifies the number of times to loop through the group processing portion of the process flow diagram. No additional sampling or model averaging is performed.
- **Bagging** — the bootstrap aggregation, or bagging mode creates unweighted samples of the active training data set for bagging. The bagging method uses random sampling with replacement to create the samples. Unweighted resampling does not apply extra weight to the cases that were misclassified in the previous samples. When you specify the Bagging mode, you also need to use the Bagging section of the Train properties panel to specify your settings for the Type, Observations, Percentage, and Random Seed properties.

Unweighted resampling for bagging is the most straightforward method of resampling. The Bagging method uses random sampling with replacement to create the n sample replicates. You set the number of samples (and, in turn, models) that you want to create in the Index Count property in the General section of the Train properties.

You can specify the sample size as a percentage of the total observations in the data set, or as an absolute number of observations to be sampled. The default percentage is set to 100% of the observations in the input data set. The default number of observations is set to the total number of observations in the input data set. The actual sample size is an approximate percentage or number. For example, a 10% random sample of 100 observations might contain 9, 10, or 11 observations.

If the data set contains a frequency variable, then the frequency variable is used to determine the percentage of observations sampled instead of the number of observations. For example, assume that you have the following data set:

Obs	Freq	X1
1	5	a
2	3	b
3	2	c

In this case, the percentage of observations sampled is based on the total frequency count ($5+3+2=10$) instead of the total number of observations in the data set (3).

By default, the random seed that is used to generate the initial sample is 12345. To create a new random seed, enter a new seed value in the property field. The seed values used to generate your samples are saved by the node, so you can replicate the samples in another run of the process flow. You must use the same initial seed value to replicate samples from one run to another. To automatically use a different seed value each time, set the seed to 0. When the seed value is set to 0, the computer clock creates a random seed at run time to initialize the seed stream. Using this method, your samples are always different when you rerun the flow.

In bagging mode, after the group processing portion of the process flow diagram repeats the specified number of times, the final model is created by averaging the probabilities that were generated in each model iteration. Each iteration creates a new sample of the data that introduces variability. Bagging is most often performed with decision tree models.

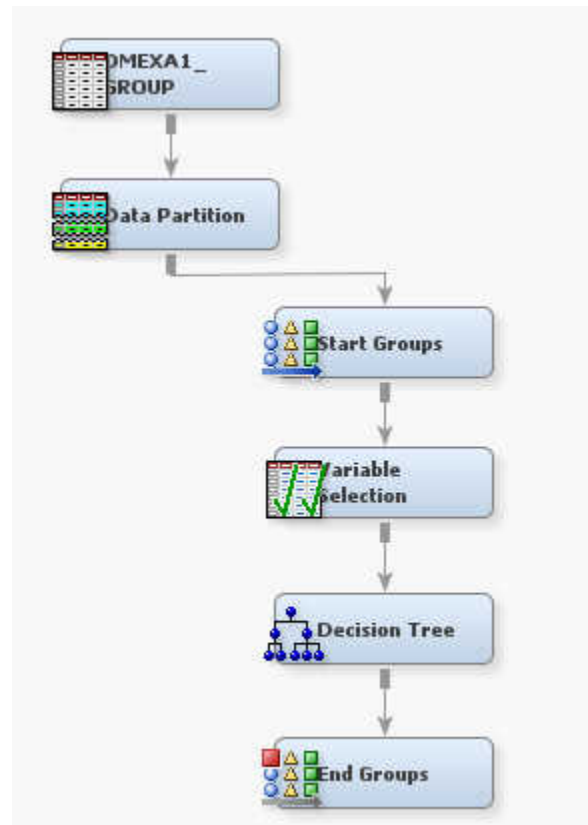
- **Boosting** — Reweighted resampling was developed as a way of boosting the performance of a weak learning algorithm. Use the boosting mode if you want the sampling method to reweight each training observation. The weights in the resampling are increased for the observations that are most often misclassified in the previous models. Therefore, the distribution of the observation weights is based on the model performance of the previous samples.
- **Target** — the target mode loops for each target variable that is identified in the training data. The target looping mode creates similar models for a number of targets, such as modeling a competitive cross-sell scenario.
- **Stratify** — Use the Stratify mode to perform standard group processing. When you use the Stratify mode, the **Start Groups** node loops through each level of group variable when you run the process flow diagram. When you select Stratify, the Minimum Group Size and Target Group properties are enabled.
- **Cross-Validation** — the cross validation looping mode is a modification of the stratified mode. You use the cross validation mode when you want to export the complement of the groups specified, as opposed to the groups themselves. For example, assume you have two group variables, GENDER [M, F] and REGION [N, S, E, W]. Using Stratify-mode looping to perform standard group processing, the groups passed would be as follows:
 - Loop 1 — Gender M and Region N
 - Loop 2 — Gender F and Region N
 - Loop 3 — Gender M and Region S
 - ...
 - Loop 8 — Gender F and Region W

When using Cross-Validation mode looping, all the data except the group is passed on. Using Cross-Validation mode looping with our example group variables GENDER [M, F] and REGION [N, S, E, W] we get the following:

- Loop 1 — not(Gender M and Region N), that is, everything that is not M and N
- Loop 2 — not(Gender F and Region N), that is, everything that is not F and N
- Loop 3 — not(Gender M and Region S), that is, everything that is not M and S
- ...
- Loop 8 — not(Gender F and Region W), that is, everything that is not F and W
- **No Grouping** — no looping is performed. You might want to suppress the **Start Groups** node from looping to reduce the run time during preliminary model building.

Start Groups Node and Bag Bite Sampling

You can use the index looping mode of the **Start Groups** node to perform Bag Bite sampling. To perform Bag Bite sampling, you typically write customized SAS code to create the *n* samples of the input data set that are required for subsequent modeling.





Each time the flow loops, the **Start Groups** node passes the input data to the **Variable Selection** node. The data then passes to the **Decision Tree** node for modeling. The **End Groups** node signifies the end of the Group Processing portion of the process flow diagram, and calculates the posterior probabilities (for class targets) or the predicted values (for interval targets) from the individual tree models to form the final classifier.

Start Groups Node Properties


Start Groups Node General Properties

The following general properties are associated with the **Start Groups** node:


- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **Start Groups** node that is added to a diagram will have a Node ID of Grp. The second **Start Groups** node added to a diagram will have a Node ID of Grp2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Start Groups window. The Imported Data — Start Groups window contains a list of the ports that provide data sources to the **Start Groups** node. Select the  button to the right of the Imported Data property to open a table of the imported data. If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:
 - **Browse** to open a window where you can browse the data set.
 - **Explore** to open the Explore window, where you can sample and plot the data.

- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Start Groups window. The Exported Data — Start Groups window contains a list of the output data ports that the **Start Groups** node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Start Groups Node Train Properties

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the **Start Groups** node. Select the  button to open a window that contains the variables table. You can set the Use, Report, and Grouping Role values for a variable. In the variables table, you can also view the columns metadata and open an Explore window to view a variable's sampling information, observation values, or view a plot of variable distribution.
- **Rerun** — Use the Rerun property to specify whether the portion of the process flow diagram between the **Start Groups** node and the **End Groups** node should rerun each time the process flow is executed, regardless of whether the node or the process flow diagram has run before or not. Setting the Rerun value to **Yes** enables group processing loops to be performed even if there are configuration errors in subsequent nodes downstream in the process flow diagram. Valid values are **Yes** and **No**. The default setting for the Rerun property is **No**.

Start Groups Node Train Properties: General

- **Mode** — specifies the looping mode that you want the **Start Groups** node to use. Exceptions for the High Performance Data Mining nodes are noted.
 - **Index** — the index mode setting specifies the number of times to loop through the group processing portion of the process flow diagram. No additional sampling or model averaging is performed.
All High Performance Data Mining nodes work in **Index** mode.
 - **Bagging** — Unweighted resampling for bagging is the most straightforward method of resampling. Unweighted resampling uses random sampling with replacement to create the *n* sample replicates. You set the number of samples (and in turn, models) that you want to create in the Index Count property in the General section of the Train properties. Unweighted resampling does not apply extra weight to the cases that were misclassified in the previous samples. When you specify the Bagging mode, you should specify settings for the Type, Percentage, and Random Seed properties in the Bagging properties section.

Most High Performance Data Mining nodes support **Bagging** in single-machine mode, but not with distributed data. The HP Forest node and HP SVM node are not supported in either single-machine or distributed mode.

- **Boosting**— the boosting mode performs weighted resampling to create boosting models. Boosting models are a modification of bagging models. Boosting models use a frequency variable that has a value proportional to the model residual. Rows that are incorrectly sampled are given higher frequency values, so the next model will consider them more significantly.

Most High Performance Data Mining nodes support **Boosting** in single-machine mode, but not with distributed data. The HP Forest node and HP SVM node are not supported in either single-machine or distributed mode.

- **Target** — the target mode loops for each target variable that is identified in the training data. The target looping mode creates similar models for a number of targets, such as modeling a competitive cross-sell scenario.

The HP Forest node and the HP SVM node with the **Optimization Method** property set to **Active** are not supported. All other High Performance Data Mining nodes are supported.

- **Stratify** — Use the Stratify mode to perform standard group processing. When you use the Stratify mode, the **Start Groups** node loops through each level of group variable when you run the process flow diagram. When you select the Stratify mode, the Minimum Group Size and Target Group properties are enabled.

The HP Forest node and the HP SVM node with the **Optimization Method** property set to **Active** are not supported. All other High Performance Data Mining nodes are supported.

- **Cross-Validation** — the cross validation looping mode is a modification of the stratified mode. You use the cross validation mode when you want to export the complement of the groups specified, as opposed to the groups themselves.

The HP Forest node and the HP SVM node with the **Optimization Method** property set to **Active** are not supported. All other High Performance Data Mining nodes are supported.

- **No Grouping** — no looping is performed. You might want to suppress the node from looping to reduce the run time during preliminary model building.
- **Target Group** — Use the Target Group property to specify whether to process the target as a separate group. The default setting for the Target Group property is **No**.
- **Index Count** — When the Mode property is set to Index, Bagging, or Boosting, use the Index Count property to specify the number of loops to be performed. The Index Count property accepts positive integers as inputs.
- **Minimum Group Size** — When the Mode property is set to Stratify or Cross-Validation, use the Minimum Group Size to define the minimum number of observations that must be in a group for looping to apply to that group. By default, if a group level has fewer than 10 observations, then that group level is not processed. You can change the Minimum Group Size by entering a different value in the input field. The Minimum Group Size property accepts positive integers as inputs.

Start Groups Node Train Properties: Bagging

The properties in the Bagging group are available only when the Mode property is set to Bagging. Otherwise, the properties in the Bagging group are dimmed and unavailable.

- **Type** — Use the type property to specify the method that you want to use to determine the sample size.
 - **Percentage** — a percentage of the population. When the Type property is set to Percentage, a value must be entered in the Percentage property that specifies the desired proportion of the population to extract as a sample.
 - **Number of Observations** — a discrete number of observations. When the Type property is set to Number of Observations, a value must be entered in the Observations property that specifies the number of observations to extract as a sample.
- **Observations** — Specifies the number of observations to include in the sample when the Type property is set to Number of Observations.
- **Percentage** — Specifies the proportion of observations to include in the sample when the Type property is set to Percentage.
- **Random Seed** — Specifies the random seed value used to randomly sample observations from the input data.

Start Groups Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Start Groups Node Results

Open the Results window by right-clicking the node and selecting **Results** from the pop-up menu.

Select **View** from the main menu to view the following results in the Group Processing Results window:

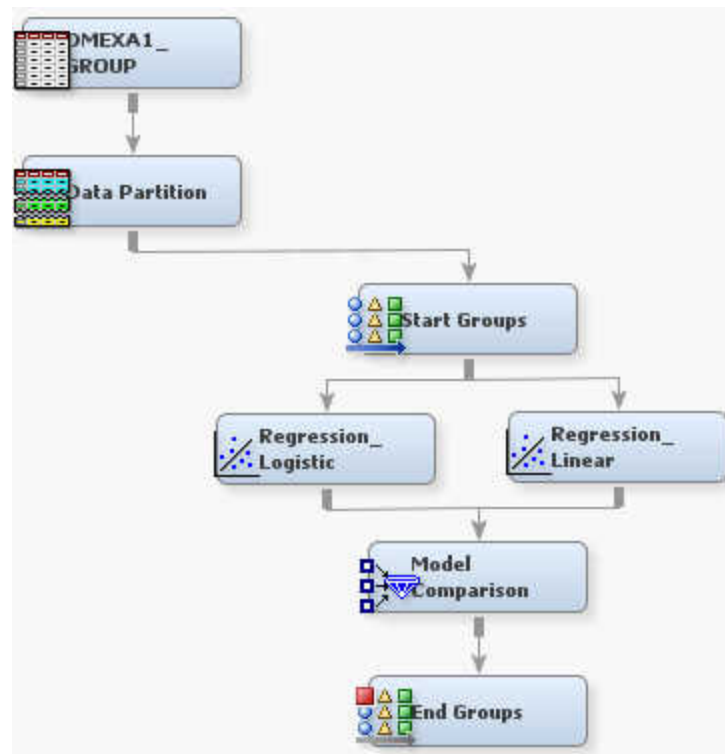
- **Properties**
 - **Settings** — displays a window with a read-only table of the configuration information in the **Start Groups** node properties panel. The information was captured when the node was last run.
 - **Run Status** — indicates the status of the **Start Groups** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — a read-only table of variable meta information about the data set that was submitted to the **Start Groups** node. The table includes columns to see

the variable name, the variable role, the variable level, the model used, and the group role used (Name, Role, Level, Use, Group Role).

- **Train Code** — the code that SAS Enterprise Miner used to train the node.
- **Notes** — displays any user-created notes of interest.
- **SAS Results**
 - **Log** — the SAS log of the **Start Groups** node run.
 - **Output** — the SAS output of the **Start Groups** node run. Typical output includes information such as a Variable Summary by Role, Level and Count, R-Squares, Chosen Effects, and Analysis of Variance (ANOVA) tables for the target variable, Estimating Logistic and Cutoff Classification tables, Node Split History, Split Effect Summary, and a Summary of Variable Selection. The Variable Clustering example contains additional information about the contents of the SAS Output window.
 - **Flow Code** — the SAS code used to produce the output that the **Start Groups** node passes on to the next node in the process flow diagram.
- **Scoring**
 - **SAS Code** — the **Start Groups** node does not generate SAS Code. The SAS Code menu item is dimmed and unavailable in the Group Processing Results window.
 - **PMML Code** — the **Start Groups** node does not generate PMML code.
- **Group Processing**
 - **Summary** — a table that contains the Group Index, Group, and Frequency Count.

Start Groups Node Example

As a marketing analyst at a catalog company, you would like to build a logistic regression model for a binary target that indicates whether a purchase is made. You would also like to run a linear regression on an interval target that indicates the purchase amount. Separate models are required for males and females.




Complete the following steps to create the DMEXA1 data source:

1. From the main menu, select **File** ⇒ **New** ⇒ **Data Source**.
2. In the Metadata Source window, select **SAS Table** as the **Source**, and click **Next**.
3. In the Select a SAS Table window, type **SAMPSTIO.DMEXA1** in the **Table** field. Click **Next**.
4. Click **Next** in the Table Information window.
5. In the Metadata Advisor Options window, select the **Advanced** button. Click **Next**.
6. Set the Role of **PURCHASED** and **AMOUNT** to **Target**. Click **Next**.
7. Select **No** in the Decision Configuration window, and click **Next**.
8. Select **No** in the Create Sample window. Click **Next**.
9. In the Data Source Attributes window, set the data set Role to **Train**. Click **Next**.
10. In the Summary window, click **Finish**.

You are now ready to create a diagram and add the nodes to your diagram workspace.

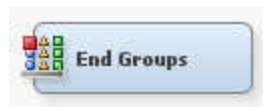
1. To create a diagram, right-click **Diagrams** in the Project Panel and select **Create Diagram**. Enter **Start Groups Example** in the **Diagram Name** field. Click **OK**.
2. From the Project Panel, drag the **DMEXA1** data source onto your diagram workspace.
3. From the **Sample** tab, drag a **Data Partition** node to the diagram workspace. Connect the **DMEXA1** data source to the **Data Partition** node. In the **Data Partition** node properties, set the value of **Training** to **70**, **Validation** to **30**, and **Test** to **0**.

- From the **Utility** tab, drag a **Start Groups** node to the diagram workspace. Connect the **Data Partition** node to the **Start Groups** node. Ensure that the **Mode** property is set to **Stratify**.

Select the **Start Groups** node in your diagram workspace. Click the  next to the **Variables** property to open the Variables window. Assign the variable GENDER the **Grouping Role** of **Stratification**. Click **OK**.

- From the **Model** tab, drag a **Regression** node to the diagram workspace. Connect the **Start Groups** node to the **Regression** node. Set the **Regression Type** property to **Logistic Regression**.
- Add a second **Regression** node to the diagram. Connect the **Start Groups** node to this node as well. Set the **Regression Type** property to **Linear Regression**.
- From the **Assess** tab, drag a **Model Comparison** node to your diagram workspace. Connect both of your **Regression** nodes to the **Model Comparison** node.
- From the **Utility** tab, drag an **End Groups** node onto your diagram workspace. Connect the **Model Comparison** node to the **End Groups** node.
- Right-click the **End Groups** node and select **Run**. In the Confirmation window click **Yes**. In the Run Status window click **OK**.

End Groups



Overview of the End Groups Node

The **End Groups** node is located on the **Utility** tab of the SAS Enterprise Miner tools bar. The **End Groups** node is used only in conjunction with the **Start Groups** node. The **End Groups** node acts as a boundary marker that defines the end of group processing operations in a process flow. Group processing operations are performed on the portion of the process flow that exists between the **Start Groups** node and the **End Groups** node.

If the group processing function that is specified in the **Start Groups** node is stratified, bagging, or boosting, the **End Groups** node functions as a model node and presents the final aggregated model. (It is no longer necessary to use an **Ensemble** node to present the aggregated group processing model, as was required in SAS Enterprise Miner 4.3.) SAS Enterprise Miner tools that follow the **End Groups** node continue data mining processes normally.

End Groups Node Algorithm

The **End Groups** node identifies the boundaries of the group processing portion of a process flow and aggregates results for stratified, bagging, or boosting models. The **End Groups** node does not heuristically process group processing data.

Using the End Groups Node


The **End Groups** node can be used only in conjunction with the **Start Groups** node. The **End Groups** node must be preceded in the process flow diagram by the **Start Groups** node and at least one successor node to the **Start Groups** node.

It is possible to have more than one **Start Groups** and **End Groups** node pair in a process flow, as long as each group processing operation (within a **Start Groups** and **End Groups** node pair) is performed serially. In other words, each group processing operation must complete before the subsequent group processing operation begins. SAS Enterprise Miner cannot run more than one group processing operation at a time. As a result, you cannot nest a group processing operation within a group processing operation. The restriction also means that you also should not design process flows that perform group processing operations in competing parallel branches.


End Groups Node Properties

End Groups Node General Properties

The following general properties are associated with the **End Groups** node:


- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first **End Groups** node that is added to a diagram will have a Node ID of EndGrp. The second **End Groups** node added to a diagram will have a Node ID of EndGrp2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — End Groups window. The Imported Data — End Groups window contains a list of the ports that provide data sources to the **End Groups** node. Select the  button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — End Groups window. The Exported Data — End Groups window contains a list of the output data ports that the **End Groups** node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

End Groups Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

End Groups Node Results

You can open the Results window of the **End Groups** node by right-clicking the node and selecting **Results** from the pop-up menu.

Select **View** from the main menu to view the following results in the Results Package:

- **Properties**
 - **Settings** — displays a window with a read-only table of the configuration information in the **End Groups** node properties panel. The information was captured when the node was last run.
 - **Run Status** — indicates the status of the **End Groups** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — a read-only table of variable meta information about the data set submitted to the **End Groups** node. The table includes columns to see the variable name, the variable role, the variable level, and the model used.
 - **Notes** — displays any user-created notes of interest.
- **SAS Results**
 - **Log** — the SAS log of the **End Groups** node run.
 - **Output** — the SAS output of the **End Groups** node run.
 - **Flow Code** — the SAS code used to produce the output that the **End Groups** node passes on to the next node in the process flow diagram.
 - **Group Log\Output** — the log for the group processing portion of the process flow diagram.
- **Scoring**
 - **SAS Code** — the SAS score code that was generated by the node.
- **Assessment**
 - **Classification Chart** — displays a histogram by classification and data role.

- **Score Rankings Matrix** — displays the Score Rankings matrix plot. The score rankings matrix plot overlays the selected statistics for standard, baseline, and best models in a lattice that is defined by the various models and the various data sets.

The curves for the **Best** measures represent a model that predicts the target correctly for all observations.

Available assessment options in drop-down menu include the following:

- Cumulative Lift
- Lift
- Gain
- % Response
- Cumulative % Response
- % Captured Response
- Cumulative % Captured Response
- **Score Distribution** — displays the Score Distribution chart. The Score Distribution chart plots the proportions of events (by default), nonevents, and other values on the vertical axis across the various bins in a lattice that is defined by the various models and the various data sets.
- **Fit Statistics** — a table that contains fit statistics for predicted variables.
- **Group Processing**
 - **Summary** — displays a table that contains the Group Index, Group, and Frequency Count.

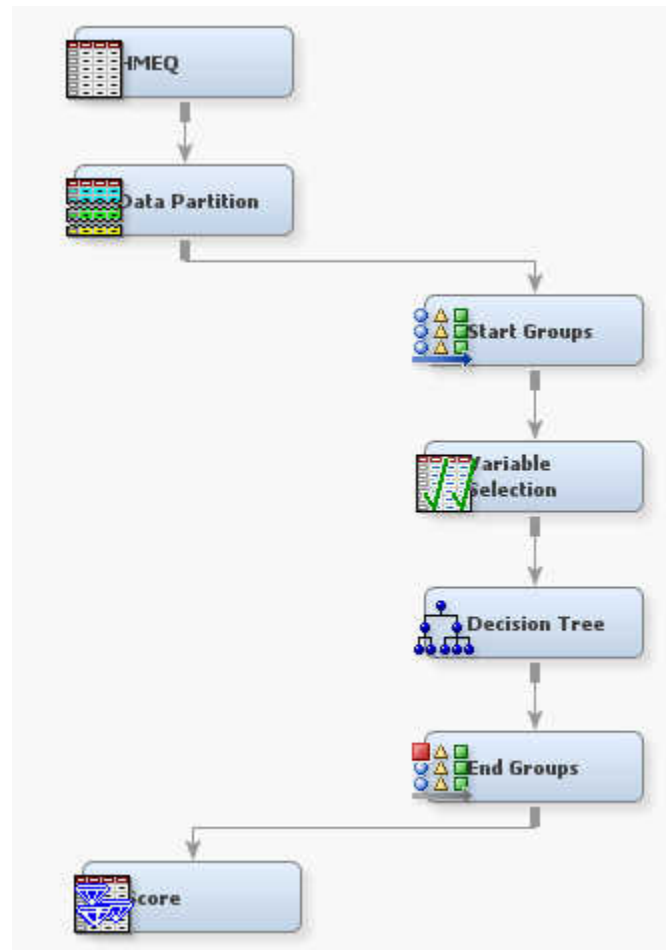
End Groups Node Example

The example below illustrates how the **End Groups** node indicates the end of the group processing portion of the process flow diagram.

The **Start Groups** node begins the group processing portion. The group processing portion of this process flow diagram is: **Start Groups** ⇒ **Variable Selection** ⇒ **Decision Tree** ⇒ **End Groups**.

The **Variable Selection** and **Decision Tree** nodes constitute the group processing portion of the process flow diagram below.

If the **Start Groups** node is configured to perform stratified, bagging, or boosting functions, then the **End Groups** node Results presents the aggregated model when the process flow diagram runs in its entirety.



Metadata



Overview of the Metadata Node


The Metadata node belongs to the Utility category in the SAS data mining process of Sample, Explore, Modify, Model, and Assess (SEMMA). Use the Metadata node to modify metadata information in your process flow diagram. You can modify attributes such as variable roles, measurement levels, and order. You can also merge predecessor variables and modify data role and multiple roles in the Metadata node.

For example, it is common to generate a data set in the SAS Code node and then modify its metadata with the Metadata node. You cannot follow the SAS Code node with an Input Data node in your process flow diagram.


Metadata Node Properties

Metadata Node General Properties


The following general properties are associated with the Metadata Node:

- **Node ID** — The Node ID property displays the ID that Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Metadata node added to a diagram will have a Node ID of Meta. The second Metadata node added to a diagram will have a Node ID of Meta2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Metadata window. The Imported Data — Metadata window contains a list of the ports that provide data sources to the Metadata node. Select the  button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:


- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Metadata window. The Exported Data — Metadata window contains a list of the output data ports that the Metadata node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Metadata Node Train Properties

The following train properties are associated with the Metadata node:

- **Import Selection** — Use the Import Selection property to specify the source to populate the import ports of the node. Select the  button to the right of the Import Selection property to open the Import Selection Editor. The Import Selection editor displays two columns: Port and Data Set. There are five data port types: Train, Validate, Test, Score, and Transaction. The Data Set field is a drop-down list that contains the IDS designation for each data source that has a particular port type.

- **Summarize** — Use the Summarize property to specify to compute statistics for the active metadata. If the statistics already exist, then the statistics will be refreshed.
- **Advanced Advisor** — Indicates whether the node should refresh the exported metadata using the Advanced Advisor. When set to **Yes**, the Advanced Advisor is used to determine the level and role attributes of the variables in the data.

Metadata Node Train Properties: Rejected Variables



- **Hide Rejected Variables** — Set the Hide Rejected property to **Yes** if you want to drop rejected variables from your exported metadata. Rejected variables are "hidden" and will not appear in the exported metadata, but they are not dropped from exported data sets and views. The default setting for the Hide Rejected property is **No**.
- **Combine Rule** — Use the Combine Rule property to specify the rule for rejecting input variables based on multiple sources of metadata.
 - **None** — The role of input and rejected variables is based on the active metadata.
 - **Any** — A variable is set to Rejected if it is rejected in at least one of the incoming metadata sources.
 - **All** — A variable is rejected only if it is rejected in all of the incoming metadata sources.
 - **Majority** — A variable is rejected if it is rejected in the majority of the incoming metadata sources. If there is a tie, the rejection is based on the active metadata source.

Metadata Node Train Properties: Variables

- **Variables** — Use the Variables property to specify how to use the variables in your data sources. The Metadata node recognizes five different types of data sources. The different types of data sources are determined by the Role property of each data source node.

The five types are as follows:

- **Train**
- **Transaction**
- **Validate**
- **Test**
- **Score**

There are five corresponding properties in the Metadata node's Variables property group. Select the  button to the right of any of the five properties to open a variables table for that data source type. For example, if you select the  button to the right of the Train property, a variables table opens for the input data source that has a role of Train. When there are multiple data sources of the same type, the data set that is displayed in the variables table is determined by the value that you designate in the Metadata node's Import Selection property. In the table, you can perform the following metadata changes:

- Set the Hide status for each variable to either Default, Yes, or No. The Default setting maintains the variable's previous Hide specification.
- Set the New Role setting to change the role of a selected variable.
- Specify a New Level setting if you want to change a variable's level. The available settings for the New Level column are Default, Unary, Binary,

Nominal, Ordinal, and Interval. The Default setting maintains the variable's previous Level specification.

- Specify a New Order setting if you want to change a variable's sort order. The available settings for the New Order column are Default, Ascending, Descending, Formatted Ascending, and Formatted Descending. The Default setting maintains the variable's previous Order specification.
- Specify a New Report setting if you want to change a variable's Report specification. The available values for the New Report column are Default, Yes, and No. The Default setting maintains the variable's previous Report specification.


Metadata Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Metadata Node Variables Table

Use the table in the Variables — Meta window to change or specify metadata that pertains to data that has been exported by an Enterprise Miner node. You can specify new metadata information on a variable-by-variable basis if you wish. To open the

Variables — Meta window, select the  button to the right of the Variables property in the Properties Panel while the Metadata node is selected in the diagram workspace. The example below uses the SAMPSIO.HMEQ data set.

Name	Hidden	Hide	Role	New Role	Level	New Level	New Order	New Report
BAD	N	Default	Target	Default	Binary	Default	Default	Default
CLAGE	N	Default	Input	Default	Interval	Default	Default	Default
CLNO	N	Default	Input	Default	Interval	Default	Default	Default
DEBTINC	N	Default	Input	Default	Interval	Default	Default	Default
DELINQ	N	Default	Input	Default	Interval	Default	Default	Default
DEROG	N	Default	Input	Default	Interval	Default	Default	Default
JOB	N	Default	Input	Default	Nominal	Default	Default	Default
LOAN	N	Default	Input	Default	Interval	Default	Default	Default
MORTDUE	N	Default	Input	Default	Interval	Default	Default	Default
NINQ	N	Default	Input	Default	Interval	Default	Default	Default
REASON	N	Default	Input	Default	Nominal	Default	Default	Default
VALUE	N	Default	Input	Default	Interval	Default	Default	Default
YOJ	N	Default	Input	Default	Interval	Default	Default	Default

You can resize the columns in the Variables — Meta window to enhance readability. Click a column heading to toggle between ascending and descending column and table sorts. Information in cells that have a white background can be configured. Cells that have gray backgrounds are read-only. To modify the following properties by individual variable, you can use the selections that are in the white columns. Selections that are dimmed or unavailable are not suitable for the type of variable that has been selected.

- **Hide** — Use the Hide column to specify whether to hide the variable when exporting output data to successor nodes. The New Hide column permits values of Default, Yes, and No.
- **New Role** — Use the New Role column to specify a new variable role. If you select Default, you continue to use the variable role that was assigned to the variable in the predecessor node. For a detailed list of roles, see Model Roles and Levels of Variables.
- **New Level** — Use the New Level column to specify a new measurement level for class variables. If you select Default, you continue to maintain the variable Level setting that was used in the predecessor node. For a detailed list of levels, see Model Roles and Levels of Variables.
- **New Order** — Use the New Order column to specify the sort order for class variables.
 - **Ascending**
 - **Descending**
 - **Default**
 - **Formatted Ascending**
 - **Formatted Descending**
- **New Report** — Use the New Report column to specify whether a variable should be automatically included in reports such as the predictive model assessment decile and percentile tables.
 - **Default** — Use the default variable report settings that are passed to this node.
 - **Yes** — Override any previous report settings for this variable and set them to Yes.
 - **No** — Override any previous report settings for this variable and set them to No.

Metadata Node Results

After a successful node run, you can open the Results window of the Metadata node by right-clicking the node and selecting Results from the pop-up menu.

Select **View** from the main menu to view the following results in the Results window:

- **Properties**
 - **Settings** — displays a window with a read-only table of the Metadata node properties configuration when the node was last run.
 - **Run Status** — indicates the status of the Metadata node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — a table of the variables in the training data set.
 - **Train Code** — the code that Enterprise Miner used to train the node. The Train Code property is, by default, dimmed and unavailable for the Metadata node.
 - **Notes** — displays the information typed by the user for the node.
- **SAS Results**
 - **Log** — the SAS log of the Metadata node run.
 - **Output** — The Output listing of the Metadata node contains a table of the variables that have modified attributes and a table of the metadata exported by the node for some of the attributes (name, role, level, creator, and label).
 - **Flow Code** — The Metadata node does not produce SAS flow code.
 - **Statistics Table** — The Metadata node does not produce a statistics table.
- **Scoring**
 - **SAS Code** — the Metadata node does not generate SAS score code.
 - **PMML Code** — the Metadata node does not generate PMML code.
- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard to modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

SAS Code



Overview of the SAS Code Node


The SAS Code node enables you to incorporate new or existing SAS code into process flow diagrams that were developed using SAS Enterprise Miner. The SAS Code node

extends the functionality of SAS Enterprise Miner by making other SAS System procedures available for use in your data mining analysis. You can also write SAS DATA steps to create customized scoring code, conditionally process data, or manipulate existing data sets. The **SAS Code** node is also useful for building predictive models, formatting SAS output, defining table and plot views in the user interface, and for modifying variables metadata. The **SAS Code** node can be placed at any location within a SAS Enterprise Miner process flow diagram. By default, the **SAS Code** node does not require data. The exported data that is produced by a successful **SAS Code** node run can be used by subsequent nodes in a process flow diagram.


SAS Code Node Properties

SAS Code Node General Properties


The following general properties are associated with the **SAS Code** node:

- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type.
- **Imported Data** — Select the  button to open a table of SAS data sets that are imported into the **SAS Code** node.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:


- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — Select the  button to open a table of SAS data sets that are exported data by the **SAS Code** node.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:


- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

SAS Code Node Train Properties

The following train properties are associated with the **SAS Code** node:

- **Variables** — Use the Variables table to specify the status for individual variables that are imported into the **SAS Code** node. Select the  button to open a window that contains the variables table. You can set the Use and Report status for individual variables, view the columns metadata, or open an Explore window to view a variable's sampling information, observation values, or a plot of variable

distributions. You can apply a filter based on the variable metadata column values so that only a subset of the variables is displayed in the table.

- **Code Editor** — Click the  button to open the Code Editor. You can use the Code Editor to edit and submit code interactively at the same time that you view the SAS log and output listings. You can also run a process flow diagram path up to and including the **SAS Code** node and view the Results window without closing the programming interface. For more details, see the Code Editor section below.
- **Tool Type** — specifies the node type using the SAS Enterprise Miner SEMMA framework. Valid values are as follows:
 - **Sample**
 - **Explore**
 - **Modify**
 - **Model**
 - **Assess**
 - **Utility**

The default setting for the Tool Type property is Utility. When the Tool Type is set to Model, SAS Enterprise Miner creates a target profile for the node if none exists. It also creates a report data model that is appropriate for a modeling node. Doing so enables SAS Enterprise Miner to automatically generate assessment results as long as certain variables are found in the scored data set (P_, I_, F_, R_ (depending on the target level)). See Predictive Modeling for more details about these variables and other essential information about modeling nodes.

- **Data Needed** — specifies whether the node needs at least one predecessor node. Valid values are **Yes** and **No**. The default setting for the Data Needed property is **No**.
- **Rerun** — specifies whether the node should rerun each time the process flow is executed, regardless of whether the node has run before or not. Valid values are **Yes** and **No**. The default setting for the Rerun property of the SAS Code node is **No**.
- **Use Priors** — specifies whether the posterior probability values are adjusted by the prior probability values. Valid values for the Use Priors property are **Yes** and **No**. The default setting for the Use Priors property is **Yes**.

SAS Code Node Score Properties

The following score properties are associated with the **SAS Code** node.

- **Advisor Type** — specifies the type of SAS Enterprise Miner input data advisor to be used to set the initial input variable measurement levels and roles. Valid values are as follows:
 - **Basic** — Any new variables created by the node will inherit Basic metadata attributes. These attributes include the following
 - character variables are assigned a Level of Nominal
 - numeric variables are assigned a Level of Interval
 - variables are assigned a Role of Input
 - **Advanced** — Variable distributions and variable attributes are used to determine the variable level and role attributes of newly created variables.

The default setting for the Advisor Type property is Basic. You can also control the metadata programmatically by writing SAS code to the file CDELTA_TRAIN.sas. There is also a feature that permits a user to create a data set that predefines metadata

for specific variable names. This data set must be named COLUMNMETA, and it must be stored in the EMMETA library.

- **Publish Code** — specifies the file that should be used when collecting the scoring code to be exported. Valid values are as follows:
 - **Flow** — Flow scoring code is used to score SAS data tables inside the process flow diagram. The scoring code is written to EMFLOWSCORE.sas.
 - **Publish** — Publish scoring code is used to publish the SAS Enterprise Miner model to a scoring system outside the process flow diagram. The scoring code is written to EMPUBLISHSCORE.sas.

The default setting of the Publish Code property is Publish. It is possible to have scoring code that is used within the process flow (Flow code) and different code that is used to score external data (Publish code). For example, when generating Flow code for modeling nodes, the scoring code can reference the observed target variable, and you can generate residuals from a statistical model. Since Publish code is destined to be used to score external data where the target variable is unobserved, residuals from a statistical model cannot be generated.

- **Code Format** — specifies the format of the score code to be generated. Valid values are as follows:
 - **DATA step** — The score code contains only the DATA step statement that is accumulated when the flow score code is generated.
 - **Other** — The score code contains statements other than DATA step statements, such as PROC step statements.

The default setting for the Code Format property is DATA step. It is necessary to make the distinction because nodes such as the **Ensemble** node and the **Score** node collect score code from every predecessor node in the process flow diagram. If all of the predecessor nodes generate only DATA step score code, then the score code from all of the nodes in the process flow diagram can be appended together. However, if PROC step statements are intermixed in the score code in any of the predecessor nodes, a different algorithm must be used.

SAS Code Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Code Editor Overview

You use the Code Editor to enter SAS code that executes when you run the node. The editor provides separate panes for Train, Score, and Report code. You can edit and submit code interactively in all three panes at the same time that you view the SAS log and output listings. You can also run the process flow diagram path up to and including the **SAS Code** node and view the Results window without closing the programming interface.

The Code Editor provides tables of macros and macro variables that you can use to integrate your SAS code with the SAS Enterprise Miner environment. You use the macro variables and the variables macros to reference information about the imported data sets, the target and input variables, the exported data sets, the files that store the scoring code, the decision metadata, and so on. You use the utility macros, which typically accept arguments, to manage data and format output. You can insert a macro variable, a variables macro, or a utility macro into your code without having to enter its name. You simply select an item from the macro variables list or macros table and drag it to the active code pane.

If an imported data set exists, you can access the variables table from the Code Editor. The variables table has the same functionality regardless of whether it is accessed from the Code Editor or the **SAS Code** node properties panel.

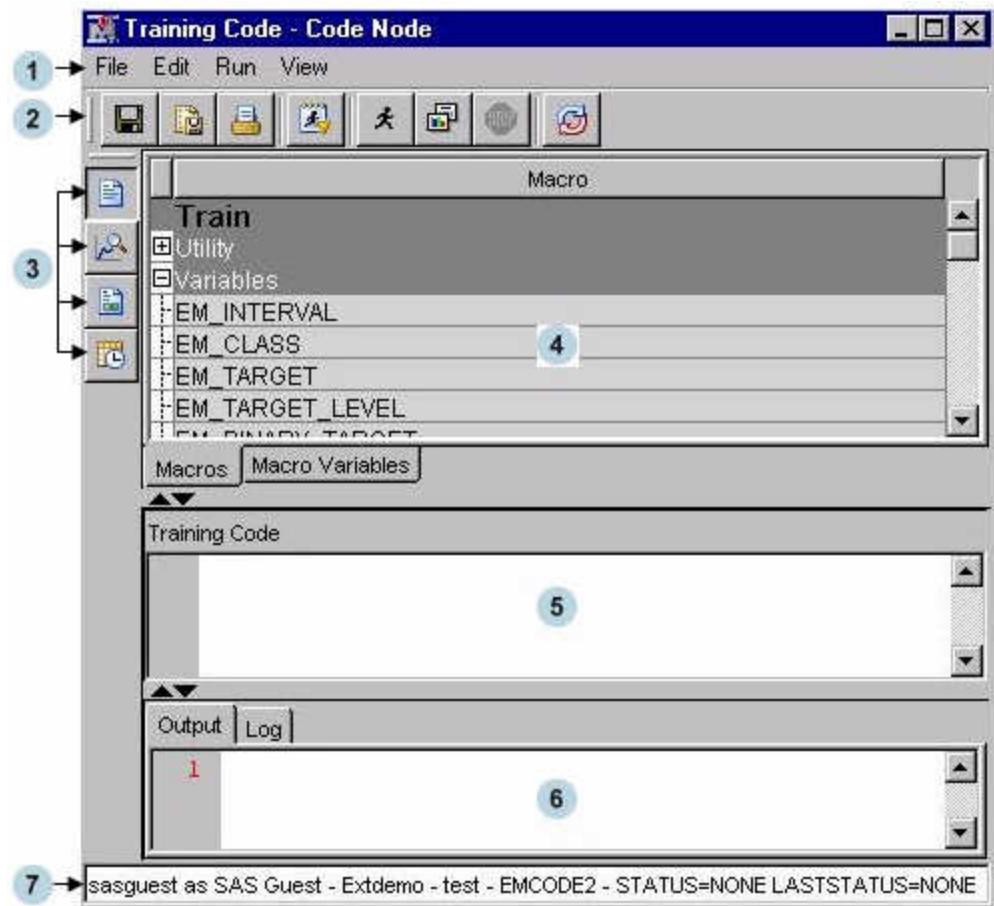
You can also access the **SAS Code** node properties panel from the Code Editor. You can specify values for any of the node properties in the Code Editor properties interface the same way you would in the **SAS Code** node properties panel.

Note: You cannot use the **SAS Code** node or SAS Enterprise Miner project start code to create an MS Excel library. If you want to use Excel data in SAS Enterprise Miner, you must use the **File Import** node to do so.

Code Editor User Interface

Introduction

The Code Editor consists of seven components. Some components serve multiple functions:







1. “Menu” on page 96
2. “Toolbar” on page 96
3. “Content Selector Buttons” on page 97
4. “Tables Pane” on page 97
 - Macros table
 - Macro variables table
 - Variables table
5. “Code Pane” on page 99
 - Training Code
 - Score Code
 - Report Code
6. “Results Pane” on page 99
 - Output
 - Log
7. “Status Bar” on page 100





Menu

The Code Editor menu consists of the following items:





- **File**
 - **Save** — saves the contents in the current view of the code pane.
 - **Save As** — saves any combination of the code, output, or log.
 - **Save All** — saves the code, output, and log.
 - **Print** — prints the contents of the pane that currently has the focus.
 - **Exit** — closes the Code Editor window and returns to the SAS Enterprise Miner main workspace.
- **Edit**
 - **Cut** — deletes the selected item and copies it to the clipboard.
 - **Copy** — copies the selected item to the clipboard.
 - **Paste** — pastes a copied item from the clipboard.
 - **Select All** — selects all of the text from the code pane.
 - **Clear All** — clears all of the text from the current code pane.
 - **Find and Replace** — opens the Find/Replace dialog box, which enables you to search for and replace text in the code, output, and log.
- **Run**
 - **Run Code** — runs the code in the active code pane. This does not affect the status of the node or the process flow. It is simply a way to validate your code.
 - **Run Node** — runs the **SAS Code** node and any predecessor nodes in the process flow that have not been executed.
 - **Results** — opens the **SAS Code** node Results window.
 - **Stop Node** — interrupts a currently running process flow.
- **View**
 - **Training Code** — views the Training Code pane.
 - **Score Code** — views the Score Code pane.
 - **Report Code** — views the Report Code pane.
 - **Properties** — open the **SAS Code** node properties panel.

Toolbar

-  — saves the contents in the current view of the code pane
-  — saves the contents of the code pane, the output, and the SAS log
-  — prints the contents of the code pane, the output, or the SAS log
-  — runs the code in the active code pane

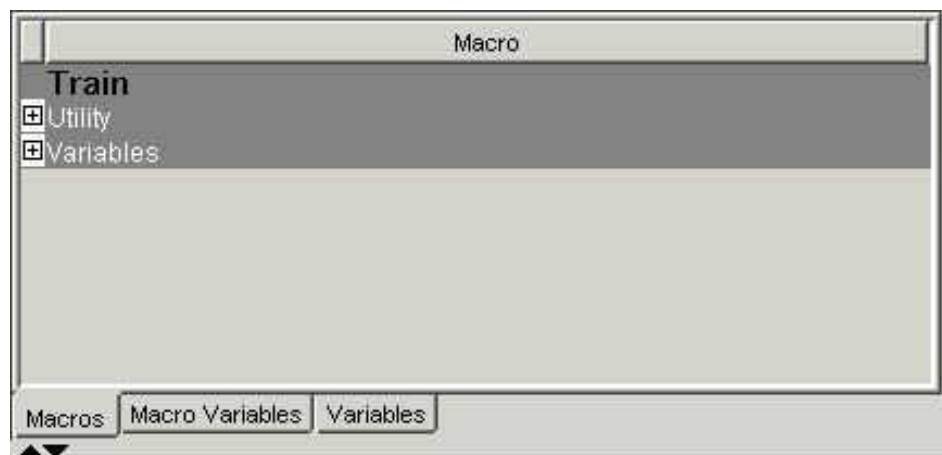
-  — runs the **SAS Code** node and any predecessor nodes in the process flow that have not been executed
-  — opens the **SAS Code** node Results window
-  — stops a currently running process flow diagram
-  — resets the workspace

Content Selector Buttons

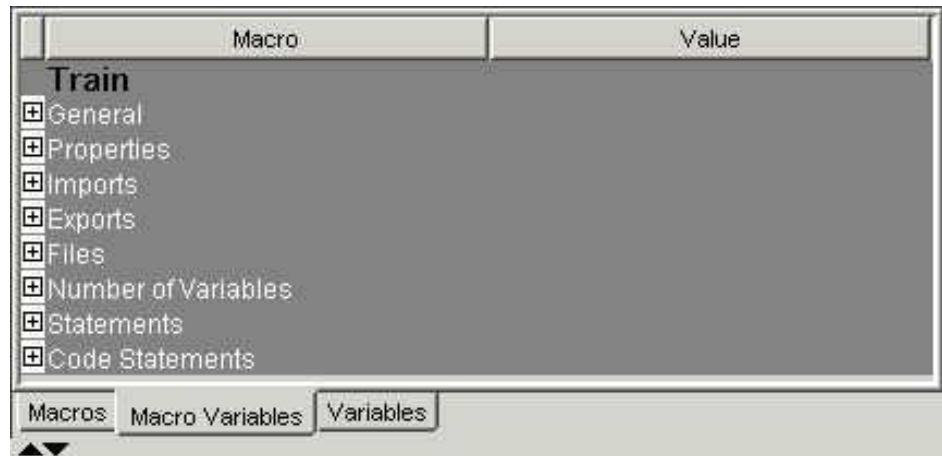
-  — displays the **SAS Code** node Training Code
-  — displays the **SAS Code** node Score Code
-  — displays the **SAS Code** node Report Code
-  — opens the property settings for the **SAS Code** node

Tables Pane

- **Macros** — Click the **Macros** tab to view a table of macros in the Tables pane. The macro variables are arranged in two groups: Utility and Variables. Click on the plus or minus sign on the left of the group name to expand or collapse the list, respectively. You can insert a macro into your code without entering its name by selecting an item from the macros table and dragging it to the code pane.



- **Macro Variables** — Click the **Macro Variables** tab to view a table of macro variables in the Tables pane. You can use the split bar to adjust the width of the columns in the table. For many of the macro variables, you see the value to which it resolves in the Value column. But in some cases, the value cannot be displayed in the table because those macro variables are populated at run time.



The macro variables are arranged in groups according to function:

- **General** — Use general macro variables to retrieve system information.
- **Properties** — Use properties macro variables to retrieve information about the nodes.
- **Imports** — Use imports macro variables to identify the SAS tables that are imported from predecessor nodes at run time.
- **Exports** — Use exports macro variables to identify the SAS tables that are exported to successor nodes at run time.
- **Files** — Use files macro variables to identify external files that are managed by SAS Enterprise Miner, such as log and output listings.
- **Number of Variables** — Use the number of variables macro variables for a given combination of the measurement levels and model roles.
- **Statements** — Use statements macro variables to identify SAS program statements that are frequently used by SAS Enterprise Miner, such as the decision statement in the modeling procedures.
- **Code Statements** — Use the Code Statements macro variable to identify the file containing the Code statement.

You can insert a macro variable into your code without entering its name by selecting an item from the macro variables table and dragging it to the code pane.

- **Variables** — Click the **Variables** tab to view the variables table in the Tables pane. The variables table has the same functionality regardless of whether it is accessed from the Code Editor or the **SAS Code** node properties panel.

Variables

(none) ☐ not Equal to

Name	Use	Report	Role	Level	Type	Order
BAD	Yes	No	Target	Binary	Numeric	
CLAGE	Yes	No	Input	Interval	Numeric	
CLNO	Yes	No	Input	Interval	Numeric	
DEBTINC	Yes	Yes	Input	Interval	Numeric	
DELINQ	Yes	No	Input	Interval	Numeric	
DEROG	Yes	No	Input	Interval	Numeric	
JOB	Yes	No	Input	Nominal	Character	
LOAN	Yes	No	Input	Interval	Numeric	
MORTDUE	Yes	No	Input	Interval	Numeric	
NINQ	Yes	No	Input	Interval	Numeric	
REASON	Yes	No	Input	Nominal	Character	

Macros Macro Variables Variables

Code Pane

The Code pane has three views: Training Code, Score Code, and Report Code.




Training Code

```


PROC MEANS data=%EM_IMPORT_DATA noprint;
  var %EM_INTERVAL;
  output out=t;
run;


data t;
  set t;
  drop _freq_ _type_;
  where _stat_ ne 'N';
run;

```

Click on the  (Training),  (Score), or  (Report) icons on the toolbar to choose the pane in which you want to work.

The code from the three panes is executed sequentially when you select **Run** node



(). Training code is executed first, followed by Score code, and then Report code.

If you select **Run Code** (), only the code in the visible code pane is executed. For more details, see the “Code Pane” on page 115 section.


Use the  controls to either expand () or collapse () the code pane.




Results Pane

The Results pane has two tabs: Output and Log. Click the **Output** tab to view the output generated by your code or click the **Log** tab to view the SAS log that was generated by

your code. If you run the node () , rather than just your code () , the output



and log must be viewed from the SAS Code node's Results window () and not from the Code Editor's Results pane.

Use the  controls to either expand () or collapse () the Results pane.

Status Bar

The status bar displays the following:

- SAS User ID — the SAS User ID of the current SAS Enterprise Miner session owner.
- User name — the User name that is associated with the current SAS Enterprise Miner session owner.
- Project name — the name of the currently open SAS Enterprise Miner project.
- Diagram name — the name of the currently open SAS Enterprise Miner diagram.
- Node name — the name of the selected node in the current SAS Enterprise Miner diagram workspace.
- Current status — the current status of the selected node in the current SAS Enterprise Miner diagram workspace.
- Last status — the last known status of the selected node in the current SAS Enterprise Miner diagram workspace.

sasuserID as UserName - ProjectName - DiagramName - NodeName - STATUS=NONE LASTSTATUS=None

Code Editor Macros

Overview

The Macros table lists the SAS macros that are used to encode multiple values, such as a list of variables, and functions that are already programmed in SAS Enterprise Miner. The macro variables are arranged in two groups: Utility and Variables. Utility macros are used to manage data and format output and Variables macros are used to identify variable definitions at run time. The macros discussion below is organized as follows:

- [“Utility Macros” on page 101](#)
 - %EM_REGISTER
 - %EM_REPORT
 - %EM_MODEL
 - %EM_DATA2CODE
 - %EM_DECDATA
 - %EM_CHECKMACRO
 - %EM_CHECKSETINIT
 - %EM_ODSLISTON
 - %EM_ODSLISTOFF
 - %EM_METACHANGE
 - %EM_GETNAME

- %EM_CHECKERROR
- %EM_PROPERTY
- [“Variables Macros” on page 106](#)

Utility Macros

Use utility macros to manage data and format output.

The following utility macros are available:

- **%EM_REGISTER** — Use the %EM_REGISTER macro to register a unique file key. When you register a key, SAS Enterprise Miner generates a macro variable named &EM_USER_key. You then use &EM_USER_key in your code to associate a file with the key. Registering a file enables SAS Enterprise Miner to track the state of the file, avoid name conflicts, and ensure that the registered file is deleted when the node is deleted from a process flow diagram.

%EM_REGISTER allows the following arguments:

- ACTION = < TRAIN | SCORE | REPORT > — associates the registered CATALOG, DATA, FILE, or FOLDER with an action. If the registered object is modified, the associated action is triggered to execute whenever the node is run subsequently. The default value is TRAIN. This is an optional argument. The argument has little use in the **SAS Code** node but can be of significant value to extension node developers.
- AUTODELETE = <Y|N> — Request the delete status of the file prior to the run. This argument is optional.
- EXTENSION = <file-extension> — an optional parameter to identify nonstandard file extensions (.sas or .txt, for example).
- FOLDER = <folder-key> — the folder key where a registered file resides (optional).
- KEY = <data-key> — an alias for a filename.
- PROPERTY = <Y|N> — an optional argument that indicates that the file is a node property and that when the node or the process flow diagram is exported, the content of the registered file will also be exported with the rest of the properties.
- TYPE = <CATALOG | DATA | FILE | FOLDER> — the type of file that is to be registered.

For example, if you want to use the data set Class from the SASHELP library, register the key Class as follows:

```
%em_register(key=Class, type=data);
```

Later, in your code, you can use statements like the following:

```
data &em_user_Class;
set Sashelp.Class;
```

References to &EM_USER_Class then resolve to the permanent data set Sashelp.Class.

- **%EM_REPORT** — Use the %EM_REPORT macro to specify the contents of a results window display that is created using a registered data set. The display contents, or view, can be a data table view or a plot view. Examples of plot types are histograms, bar charts, and line plots. The views (both tables and plots) appear in the results window of the **SAS Code** node and in any results package files (SPK files).

%EM_REPORT allows the following arguments:

- AUTODISPLAY = <Y | N> — specifies whether the report displays automatically when the results viewer is opened.
- BLOCK = <group-name> — specifies the group that the report belongs to when the results viewer is opened. The default setting is CUSTOM.
- COLOR = <variable-name> — specifies a variable that contains color value.
- COMPARE = <Y | N> — specifies whether data in the generated report can be used to compare registered models. The default setting is N.
- DESCRIPTION = <window-title-description> — specifies a text string or report description that appears in the window title.
- DISCRETEX = <Y | N> — specifies whether the values on the x-axis are discrete when the VIEWTYPE is HISTOGRAM.
- DISCRETEY = <Y | N> — specifies whether the values on the y-axis are discrete when the VIEWTYPE is HISTOGRAM.
- EQUALIZECOLX = <Y | N> — specifies if the x-axis should be equalized (that is, use a shared common scale and tick marks) across columns of the lattice. The default setting is N.
- EQUALIZECOLY = <Y | N> — specifies if the y-axis should be equalized across columns of the lattice. The default setting is N.
- EQUALIZEROWX = <Y | N> — specifies if the x-axis should be equalized (that is, it should use a shared common scale and tick marks) across rows of the lattice. The default setting is N.
- EQUALIZEROWY = <Y | N> — specifies if the y-axis should be equalized across rows of the lattice. The default setting is N.
- FREQ = <frequency-variable-name> — specifies a frequency variable.
- GROUP = <group-variable-name(s)> — specifies one or more grouping variables.
- IDVALUE = <data-set-name> — specifies a data set. When a corresponding variable name is specified using the REPORTID argument, a report is generated for each value of the specified variable in the named data set. A report window is created for each unique value.
- KEY = <data-key> (required) — specifies the data key. Because this is a required argument, you must assign the data key using %EM_REGISTER before using %EM_REPORT.
- LATTICETYPE = <viewtype> — valid viewtypes are Data, Scatter, Lineplot, Bar, Histogram, Pie, Profileview, Gainsplot.
- LATTICEX = <lattice-row-variable-name> — specifies variables to be used as rows in a lattice.
- LATTICEY = <lattice-column-variable-name> — specifies variables to be used as columns in a lattice.
- LOCALIZE = <Y | N> — specifies whether the description should be localized or used as-is. The default setting is N.
- REPORTID = <variable-name> — specifies a variable name. When a corresponding data set name is specified using the IDVALUE argument, a report is generated for each value of the specified variable in the named data set. A report window is created for each unique value.

- **SLIDER** = <Y | N> — specifies whether the threshold slider tool is visible for constellation plots.
- **SORTORDER** = < ASC | ASCENDING | DATA | DESC | DESCENDING > — specifies the order in which the X-axis data is sorted.
- **SORTORDERY** = < ASC | ASCENDING | DATA | DESC | DESCENDING > — specifies the order in which the Y-axis data is sorted.
- **SPK** = <Y | N> — specifies whether to include the report and data in an SPK package. The default setting is Y.
- **SUBGROUP** = <subgroup-variable-name(s)> — specifies one or more subgrouping variables.
- **TIPTEXT** = <variable-name> — specifies a variable that contains tooltip text.
- **TOOLTIP** = <variable-name> — specifies a variable containing tooltip text for the Constellation application.
- **VIEWS** = <numeric-value> — assigns a numeric ID to the generated report.
- **VIEWTYPE** = < plot-type > — specifies the type of plot that you want to display. Valid plot types include Data, Bar, Histogram, Lineplot, Pie, Profileview, Scatter, Gainsplot, Lattice, Dendrogram, and Constellation. Data is the default value.
- **WHERE** = — specifies an explicit SQL WHERE clause.
- **X** = <x-variable-name> — specifies the x-axis variable.
- **XREF** = <numeric-value> — specifies a reference line on the x-axis.
- **Y** = <y-variable-name> — specifies the y-axis variable.
- **Yn** = <Yn-variable-name> — where n is an integer ranging from 1 to 16. Y1, Y2, ..., Y16 specify variables that are to be plotted and overlaid on the y-axis.
- **YREF** = <numeric-name> — specifies a reference line on the y-axis.
- **Z** = <z-variable-name> — specifies the z-axis variable of a 3-dimensional plot.

Examples using %EM_REPORT are provided below.

- **%EM_MODEL** — The %EM_MODEL macro enables you to control the computations that are performed and the score code that is generated by the SAS Enterprise Miner environment for modeling nodes.

The macro supports the following arguments:

- **TARGET** = <target-variable-name> — name of the target (required).
- **ASSESS** = <Y|N> — assess the target. The default is Y.
- **DECSCORECODE** = <Y|N> — generate decision score code. The default is N.
- **FITSTATISTICS** = <Y|N> — compute fit statistics. The default is N.
- **CLASSIFICATION** = <Y|N> — generate score code to generate classification variables (I_, F_, U_). The default is N.
- **RESIDUALS** = <Y|N> — generate score code to compute residuals. The default is N.
- **PREDICTED** = <Y|N> — indicate whether the node generates predicted values. The default is Y.

For example, suppose you have a binary target variable named BAD and your code only generates posterior variables. You can use the %EM_MODEL macro to indicate

that you want SAS Enterprise Miner to generate fit statistics, assessment statistics, and to generate score code that computes classification, residual, and decision variables.

```
%em_model (
    target=BAD,
    assess=Y,
    decscorecode=Y,
    fitstatistics=Y,
    classification=Y,
    residuals=Y,
    predicted=Y);
```

Note: %EM_MODEL is available for use in your code, but it does not currently appear in the Code Editor's table of macros.

- **%EM_DATA2CODE** — The %EM_DATA2CODE macro converts a SAS data set to SAS program statements. For example, it can be used to embed the parameter estimates that PROC REG creates directly into scoring code. The resulting scoring code can be deployed without need for an EST data set. You must provide the code to use the parameter estimates to produce a model score.

%EM_DATA2CODE accepts the following arguments:

- APPEND= <Y | N> — specifies whether to append or overwrite code if the specified file already exists.
- DATA= <source-data-name> — specifies the source data.
- OUTDATA= <output-data-set-name> — specifies the name of the output data set that is created when the DATA step code runs.
- OUTFILE= <output-data-step-code-filename> — specifies the name of the output file that contains the generated SAS DATA step code.
- **%EM_DECDATA** — The %EM_DECDATA macro uses information that you entered to create the decision data set that is used by SAS Enterprise Miner modeling procedures. %EM_DECDATA copies the information to the WORK library and assigns the proper type (profit, loss, or revenue) for modeling procedures.
 - DECDATA = <decision-data-set> — specifies the data set that contains the decision data set.
 - DECMETA = <decision-metadata> — specifies the data set that contains decision metadata.
 - NODEID = <node-identifier> — specifies the unique node identifier.
- **%EM_CHECKMACRO** — Use the EM_CHECKMACRO macro to check for the existence of a macro variable. Assigning a value is optional.

%EM_CHECKMACRO accepts the following arguments:

- NAME = <macro-variable-name> — specifies the name of the macro variable for which you want to check.
- GLOBAL = <Y | N> — specifies whether the named macro variable is a global macro variable.
- VALUE = <variable-value> — specifies a value for the macro variable if it has not been previously defined.
- **%EM_CHECKSETINIT** — Use the %EM_CHECKSETINIT macro to validate and view your SAS product licensing information.

%EM_CHECKSETINIT has the following required argument:

- **PRODUCTID** = <product id number> — specifies the product identification number. If the product specified is not licensed, SAS Enterprise Miner issues an error and halts execution of the program.
- **%EM_ODSLISTON** — Use the %EM_ODSLISTON macro to turn on the SAS Output Delivery System (ODS) listing and to specify a name for the destination HTML file.

%EM_ODSLISTON accepts the following arguments:

- **FILE** = <destination-file> — specifies the name of an HTML output file that contains the generated ODS listing.
- **%EM_ODSLISTOFF** — Use the %EM_ODSLISTOFF utility macro to turn SAS ODS listing off. No argument is needed for this macro.
- **%EM_METACHANGE** — Use the %EM_METACHANGE macro to modify the columns metadata data set that is exported by a node. The macro should be called during either the TRAIN or SCORE actions.

%EM_METACHANGE allows the following arguments:

- **NAME** = <variable-name> — the name of the variable that you want to modify (required).
- **ROLE** = <ASSESS | CENSOR | CLASSIFICATION | COST | CROSSID | DECISION | FREQ | ID | INPUT | KEY | LABEL | PREDICT | REFERRER | REJECTED | RESIDUAL | SEGMENT | SEQUENCE | TARGET | TEXT | TEXTLOC | TIMEID | TREATMENT | URL > — assign a new role to the variable (optional).

Note: You cannot specify more than one frequency variable in a SAS Enterprise Miner data source.

- **LEVEL** = <UNARY | BINARY | ORDINAL | NOMINAL | INTERVAL> — assign a new measurement level to the variable (optional).
- **ORDER** = <ASC | DESC | FMTASC | FMTDESC> — new level ordering for a class variable (optional).
- **COMMENT** = <string> — a string that can be attached to a variable (optional).
- **LOWERLIMIT** = <number> — the lower limit of a numeric variable's valid range (optional).
- **UPPERLIMIT** = <number> — the upper limit of a numeric variable's valid range.
- **DELETE** = <Y | N> — indicates whether the variable should be removed from the metadata (optional).
- **KEY** = <data-key> — specifies which data set will receive the delta code, or metadata changes. If KEY= is not specified, then the default setting for KEY= is CDELTA_TRAIN, the exported training data set. The generated code is stored in the CDELTA.TRAIN.sas file in the node folder.

If there are multiple source of data and metadata feeding in the node, then a user can use other keys to modify the metadata. Other valid values for KEY= are CDELTA_VALIDATE (an exported validation data set), CDELTA_TEST (an exported test data set), CDELTA_SCORE (an exported score data set), or CDELTA_TRANSACTION (an exported transaction data set).

- **%EM_GETNAME** — Use %EM_GETNAME to retrieve the name of a file or data set that is registered to a given key. The macro initializes the EM_USER_key macro variable. This macro should be called in actions other than CREATE, rather than call the EM_REGISTER macro.

%EM_GETNAME allows the following arguments:

- KEY = <data-key> — the registered data key
- TYPE = <CATALOG | DATA | FILE | FOLDER | GRAPH> — the type of file that is registered.
- EXTENSION = <file-extension> — an optional parameter to identify nonstandard file extensions.
- FOLDER = <folder-key> — the folder key where a registered file resides (optional).
- **%EM_CHECKERROR** — This macro checks the return code and initializes the &EMEXCEPTIONSTRING macro variable. %EM_CHECKERROR has no arguments.
- **%EM_PROPERTY** — Use %EM_PROPERTY in the CREATE action to initialize the &EM_PROPERTY_name macro variable for the specified property. The macro enables you to specify the initial value to which &EM_PROPERTY_name resolves. You can also associate the property with a specific action (TRAIN, SCORE, or REPORT).

%EM_PROPERTY allows the following arguments:

- NAME = <property name> — specify the name of the property that is to be initialized (required). This is case sensitive and must match the property name that is specified in the XML properties file.
- VALUE = <initial value> — specify the initial value for the property (required). The value should match the initial attribute that is specified for the property in the XML properties file.
- ACTION = <TRAIN | SCORE | REPORT> — specify the action that is associated with the property (optional).
- **%EM_DATASET_VERIFY** — Use the %EM_DATASET_VERIFY macro to verify that a data set exists, can be open, and has the correct variables before further processing. Errors are generated, printed, and returned, by default using the sashelp.dmine data set error messages. The **Text Rule Builder** node uses this macro.

The following are examples of how to use the %EM_DATASET_VERIFY macro along with possible output from running the macro:

```
%em_dataset_verify(sashelp.dmine,vars=locale KEY Lineno text key);
    *no error;
%em_dataset_verify(sashelp.dminemine,vars=locale KEY Lineno text key);
    ----> The data set "sashelp.dminemine" does not exist.
%em_dataset_verify(sashelp.dmine,vars=locale target TARGET);
    ---->ERROR: The following columns were not found in the Table
           "sashelp.dmine: target TARGET":
```

Variables Macros

Use the variables macros to identify variable definitions at run time. Variables appear in these macros only if the variable's Use or Report status is set to Yes.

- **%EM_INTERVAL** — resolves to the input variables that have an interval measurement level. Interval variables are continuous variables that contain values across a range.
- **%EM_CLASS** — resolves to the categorical input variables, including all inputs that have a binary, nominal, or ordinal measurement level.
- **%EM_TARGET** — resolves to the variables that have a model role of target. The target variable is the dependent or the response variable.
- **%EM_TARGET_LEVEL** — resolves to the measurement level of the target variable.
- **%EM_BINARY_TARGET** — resolves to the binary variables that have a model role of target.
- **%EM_ORDINAL_TARGET** — resolves to the ordinal variables that have a model role of ordinal.
- **%EM_NOMINAL_TARGET** — resolves to the nominal variables that have a model role of nominal.
- **%EM_INTERVAL_TARGET** — resolves to the interval variables that have a model role of target.
- **%EM_INPUT** — resolves to the variables that have a model role of input. The input variables are the independent or predictor variables.
- **%EM_BINARY_INPUT** — resolves to the binary variables that have a model role of input.
- **%EM_ORDINAL_INPUT** — resolves to the ordinal variables that have a model role of input.
- **%EM_NOMINAL_INPUT** — resolves to the nominal variables that have a model role of input.
- **%EM_INTERVAL_INPUT** — resolves to the interval variables that have a model role of input.
- **%EM_REJECTED** — resolves to the variables that have a model role of REJECTED and a USE value of Yes.
- **%EM_BINARY_REJECTED** — resolves to the binary variables that have a model role of rejected.
- **%EM_ORDINAL_REJECTED** — resolves to the ordinal variables that have a model role of rejected.
- **%EM_NOMINAL_REJECTED** — resolves to the nominal variables that have a model role of rejected.
- **%EM_INTERVAL_REJECTED** — resolves to the interval variables that have a model role of rejected.
- **%EM_ASSESS** — resolves to the variables that have a model role of assessment.
- **%EM_CENSOR** — resolves to the variables that have a model role of censor.
- **%EM_CLASSIFICATION** — resolves to the variables that have a model role of classification.
- **%EM_COST** — resolves to the variables that have a model role of cost.
- **%EM_CROSSID** — resolves to the variables that have a model role of Cross ID.
- **%EM_DECISION** — resolves to the variables that have a model role of decision.

- **%EM_FREQ** — resolves to the variables that have a model role of freq.
- **%EM_ID** — resolves to the variables that have a model role of ID.
- **%EM_LABEL** — resolves to the variables that have a model role of label.
- **%EM_PREDICT** — resolves to the variables that have a model role of prediction.
- **%EM_REFERRER** — resolves to the variables that have a model role of referrer.
- **%EM_REJECTS** — resolves to the variables that have a model role of REJECTED.
- **%EM_REPORT_VARS** — resolves to the variables that have a model role of report.
- **%EM_CLASS_REPORT** — resolves to the class variables that have a model role of report.
- **%EM_INTERVAL_REPORT** — resolves to the interval variables that have a model role of report.
- **%EM_RESIDUAL** — resolves to the variables that have a model role of residual.
- **%EM_SEGMENT** — resolves to the variables that have a model role of segment.
- **%EM_SEQUENCE** — resolves to the variables that have a model role of sequence.
- **%EM_TEXT** — resolves to the variables that have a model role of text.
- **%EM_TIMEID** — resolves to the variables that have a model role of Time ID.

Macro Variables

Overview

The Macro Variables table lists the macro variables that are used to encode single values such as the names of the input data sets. The macro variables are arranged in groups according to function:

- “General” on page 108
- “Properties” on page 109
- “Imports” on page 110
- “Exports” on page 111
- “Files” on page 112
- “Number of Variables” on page 113
- “Statements” on page 114
- “Code Statements” on page 115

General

Use general macro variables to retrieve system information.

- **&EM_USERID** — resolves to the user name.
- **&EM_MTAHOST** — resolves to the host name of SAS Metadata Repository.
- **&EM_MTAHOST** — resolves to the port number of SAS Metadata Repository.

- **&EM_LIB** — resolves to the numbered EMWS SAS library containing the data sets and SAS catalogs related to the current process flow diagram. This is the same as the value of the process flow diagram's ID property.
- **&EM_DSEP** — resolves to the operating system file delimiter (for example, backslash (\) for Windows and slash (/) for UNIX).
- **&EM_CODEBAR** — resolves to the macro variable that identifies a code separator.
- **&EM_VERSION** — resolves to the version of SAS Enterprise Miner.
- **&EM_TOOLTYPE** — resolves to the node type (Sample | Explore | Modify | Model | Assess | Utility).
- **&EM_NODEID** — resolves to the node ID.
- **&EM_NODEDIR** — resolves to the path to the node folder.
- **&EM_SCORECODEFORMAT** — resolves to the format of the score code (DATASTEP | OTHER).
- **&EM_PUBLISHCODE** — resolves to the Publish Code property (FLOW | PUBLISH).
- **&EM_META_ADVISOR** — resolves to the Advisor Type property (BASIC | ADVANCED). This is equivalent to &EM_PROPERTY_MetaAdvisor.
- **&EM_MININGFUNCTION** — resolves to a description of the function of the node.

Properties

Use properties macro variables to retrieve information about the nodes.

- **&EM_PROPERTY_ScoreCodeFormat** — resolves to the value of the Code Format property.
- **&EM_PROPERTY_MetaAdvisor** — resolves to the value of the Advisor Type property. This is equivalent to &EM_Meta_Advisor.
- **&EM_PROPERTY_ForceRun** — resolves to Y or N. When set to Y, the node and its successors will rerun even though no properties, variables or imports have changed.
- **&EM_PROPERTY_UsePriors** — resolves to the value of the Use Priors property.
- **&EM_PROPERTY_ToolType** — resolves to the value of the Tool Type property.
- **&EM_PROPERTY_DataNeeded** — resolves to the value of the Data Needed property.
- **&EM_PROPERTY_VariableSet** — resolves to the name of the catalog containing the VariableSet.
- **&EM_PROPERTY_PublishCode** — resolves to the value of the Publish Code property.
- **&EM_PROPERTY_NotesFile** — resolves to the name of the file containing the contents of the Notes Editor.
- **&EM_PROPERTY_Component** — resolves to the SAS Enterprise Miner node name.
- **&EM_PROPERTY_RunID** — resolves to the value of the Run ID property. Each time the node is run a new ID is generated.

Imports

Use imports macro variables to identify the SAS tables that are imported from predecessor nodes at run time.

- **&EM_IMPORT_DATA** — resolves to the name of the training data set.
- **&EM_IMPORT_DATA_CMETA** — resolves to the name of the column metadata data set that corresponds to the training data set.
- **&EM_IMPORT_VALIDATE** — resolves to the name of the validation data set.
- **&EM_IMPORT_VALIDATE_CMETA** — resolves to the name of the column metadata data set that corresponds to the validation data set.
- **&EM_IMPORT_TEST** — resolves to the name of the test data set.
- **&EM_IMPORT_TEST_CMETA** — resolves to the name of the column metadata data set that corresponds to the test data set.
- **&EM_IMPORT_SCORE** — resolves to the name of the score data set.
- **&EM_IMPORT_SCORE_CMETA** — resolves to the name of the column metadata data set that corresponds to the score data set.
- **&EM_IMPORT_TRANSACTION** — resolves to the name of the transaction data set.
- **&EM_IMPORT_TRANSACTION_CMETA** — resolves to the name of the column metadata data set that corresponds to the transaction data set.
- **&EM_IMPORT_DOCUMENT** — resolves to the name of the document data set.
- **&EM_IMPORT_DOCUMENT_CMETA** — resolves to the name of the column metadata data set that corresponds to the document data set.
- **&EM_IMPORT_RULES** — resolves to the name of the rules data set that is exported from a predecessor Association or Path Analysis node.
- **&EM_IMPORT_REPORTFIT** — resolves to the name of the fit statistics data set.
- **&EM_IMPORT_RANK** — resolves to the name of the rank data set.
- **&EM_IMPORT_SCOREDIST** — resolves to the name of the score distribution data set.
- **&EM_IMPORT_ESTIMATE** — resolves to the name of the parameter estimates data set.
- **&EM_IMPORT_TREE** — resolves to the name of the tree data set from a predecessor modeling node.
- **&EM_IMPORT_CLUSSTAT** — resolves to the name of the cluster statistics data set from a predecessor Cluster node.
- **&EM_IMPORT_CLUSMEAN** — resolves to the name of the cluster mean data set from a predecessor Cluster node.
- **&EM_IMPORT_VARMAP** — resolves to the name of the data set of variable mapping from a predecessor Cluster node.
- **&EM_METASOURCE_NODEID** — resolves to the node ID that is providing the variables metadata.
- **&EM_METASOURCE_CLASS** — resolves to the class of the node.
- **&EM_METASOURCE_CHANGED** — resolves to Y or N, indicating whether the source of the metadata has changed.

Exports

Use exports macro variables to identify the SAS tables that are exported to successor nodes at run time.

- **&EM_EXPORT_TRAIN** — resolves to the name of the export training data set.
- **&EM_TRAIN_SCORE** — resolves to Y or N, indicating whether SAS Enterprise Miner should score the training data set.
- **&EM_TRAIN_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the training column metadata data set.
- **&EM_EXPORT_TRAIN_CMETA** — resolves to the name of the column metadata data set that corresponds to the export training data set.
- **&EM_EXPORT_VALIDATE** — resolves to the name of the export validation data set.
- **&EM_VALIDATE_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output validation data set.
- **&EM_VALIDATE_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the validation column metadata data set.
- **&EM_EXPORT_VALIDATE_CMETA** — resolves to the name of the column metadata data set that corresponds to the export validation data set.
- **&EM_EXPORT_TEST** — resolves to the name of the export test data set.
- **&EM_TEST_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output test data set.
- **&EM_TEST_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the test column metadata data set.
- **&EM_EXPORT_TEST_CMETA** — resolves to the name of the column metadata data set that corresponds to the export test data set.
- **&EM_EXPORT_SCORE** — resolves to the name of the export score data set.
- **&EM_SCORE_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output score data set.
- **&EM_SCORE_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the score column metadata data set.
- **&EM_EXPORT_SCORE_CMETA** — resolves to the name of the column metadata data set that corresponds to the export score data set.
- **&EM_EXPORT_TRANSACTION** — resolves to the name of the export transaction data set.
- **&EM_TRANSACTION_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output transaction data set.
- **&EM_TRANSACTION_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the transaction column metadata data set.
- **&EM_EXPORT_TRANSACTION_CMETA** — resolves to the name of the column metadata data set that corresponds to the export transaction data set.
- **&EM_EXPORT_DOCUMENT** — resolves to the name of the export document data set.
- **&EM_DOCUMENT_SCORE** — resolves to Y or N, indicating whether the score code will be used to create the output document data set.

- **&EM_DOCUMENT_DELTA** — resolves to Y or N, indicating whether the metadata DATA step code will be used to modify the document column metadata data set.
- **&EM_EXPORT_DOCUMENT_CMETA** — resolves to the name of the column metadata data set that corresponds to the export document data set.

Files

Use files macro variables to identify external files that are managed by SAS Enterprise Miner, such as log and output listings. Not all nodes create or manage all external files.

- **&EM_DATA_IMPORTSET** — resolves to the name of the data set containing metadata for the imported data sets.
- **&EM_DATA_EXPORTSET** — resolves to the name of the data set containing metadata for the exported data sets.
- **&EM_DATA_VARIABLESET** — resolves to the data set containing metadata for the variables that are available for use with the node.
- **&EM_DATA_ESTIMATE** — resolves to the name of the parameter estimates data set.
- **&EM_DATA_EMTREE** — resolves to the name of the tree data set.
- **&EM_DATA_EMREPORTFIT** — resolves to the name of the fit statistics data set in columns format.
- **&EM_DATA_EMOUTFIT** — resolves to the name of the fit statistics data set.
- **&EM_DATA_EMCLASSIFICATION** — resolves to the name of the data set that contains classification statistics for categorical targets.
- **&EM_DATA_EMRESIDUAL** — resolves to the name of the data set that contains summary statistics for residuals for interval targets.
- **&EM_DATA_EMRANK** — resolves to the name of the data set that contains assessment statistics such as lift, cumulative lift, and profit.
- **&EM_DATA_EMSCOREDIST** — resolves to the name of the data set that contains assessment statistics such as mean, minimum, and maximum.
- **&EM_DATA_INTERACTION** — resolves to the name of the interaction data set.
- **&EM_DATA_EMTRAINVARIABLE** — resolves to the name of the training variable data set.
- **&EM_CATALOG_EMNODELABEL** — resolves to the name of the node catalog.
- **&EM_FILE_EMNOTES** — resolves to the name of the file containing your notes.
- **&EM_FILE_EMLOG** — resolves to the name of the SAS Enterprise Miner output log file.
- **&EM_FILE_EMOUTPUT** — resolves to the name of the SAS Enterprise Miner output data file.
- **&EM_FILE_EMTRAINCODE** — resolves to the name of the file that contains the training code.
- **&EM_FILE_EMFLOWSCORECODE** — resolves to the name of the file that contains the flow score code.
- **&EM_FILE_EMPUBLISHSCORECODE** — resolves to the name of the file that contains the publish score code.

- **&EM_FILE_EMPMML** — resolves to the name of the PMML file.
- **&EM_FILE_CDELTA_TRAIN** — resolves to the name of the file that contains the DATA step code that is used to modify the column metadata associated with the training data set that is exported by a node (if one exists).
- **&EM_FILE_CDELTA_TRANSACTION** — resolves to the name of the file that contains the DATA step code that is used to modify the column metadata associated with the transaction data set that is exported by a node (if one exists).
- **&EM_FILE_CDELTA_DOCUMENT** — resolves to the name of the file that contains the DATA step code that is used to modify the column metadata associated with the document data set that is exported by a node (if one exists).

Number of Variables

Use the number of variables macro variables for a given combination of Level and Role. These macro variables only count variables that have a Use or Report status of Yes.

- **&EM_NUM_VARS** — resolves to the number of variables.
- **&EM_NUM_INTERVAL** — resolves to the number of interval variables.
- **&EM_NUM_CLASS** — resolves to the number of class variables.
- **&EM_NUM_TARGET** — resolves to the number of target variables.
- **&EM_NUM_BINARY_TARGET** — resolves to the number of binary target variables.
- **&EM_NUM_ORDINAL_TARGET** — resolves to the number of ordinal target variables.
- **&EM_NUM_NOMINAL_TARGET** — resolves to the number of nominal target variables.
- **&EM_NUM_INTERVAL_TARGET** — resolves to the number of interval target variables.
- **&EM_NUM_BINARY_INPUT** — resolves to the number of binary input variables.
- **&EM_NUM_ORDINAL_INPUT** — resolves to the number of ordinal input variables.
- **&EM_NUM_NOMINAL_INPUT** — resolves to the number of nominal input variables.
- **&EM_NUM_INTERVAL_INPUT** — resolves to the number of interval input variables.
- **&EM_NUM_BINARY_REJECTED** — resolves to the number of rejected binary input variables.
- **&EM_NUM_ORDINAL_REJECTED** — resolves to the number of rejected ordinal input variables.
- **&EM_NUM_NOMINAL_REJECTED** — resolves to the number of rejected nominal input variables.
- **&EM_NUM_INTERVAL_REJECTED** — resolves to the number of rejected interval input variables.
- **&EM_NUM_ASSESS** — resolves to the number of variables that have the model role of Assess.

- **&EM_NUM_CENSOR** — resolves to the number of variables that have the model role of Censor.
- **&EM_NUM_CLASSIFICATION** — resolves to the number of variables that have the model role of Classification.
- **&EM_NUM_COST** — resolves to the number of variables that have the model role of Cost.
- **&EM_NUM_CROSSID** — resolves to the number of variables that have the model role of Cross ID.
- **&EM_NUM_DECISION** — resolves to the number of variables that have the model role of Decision.
- **&EM_NUM_FREQ** — resolves to the number of variables that have the model role of Freq.
- **&EM_NUM_ID** — resolves to the number of variables that have the model role of ID.
- **&EM_NUM_LABEL** — resolves to the number of variables that have the model role of Label.
- **&EM_NUM_PREDICT** — resolves to the number of variables that have the model role of Predict.
- **&EM_NUM_REFERRER** — resolves to the number of variables that have the model role of Referrer.
- **&EM_NUM_REJECTS** — resolves to the number of variables that have the model role of Rejected.
- **&EM_NUM_REPORT_VAR** — resolves to the number of variables that have the model role of Report.
- **&EM_NUM_CLASS_REPORT** — resolves to the number of class variables that have the model role of Report.
- **&EM_NUM_INTERVAL_REPORT** — resolves to the number of interval variables that have the model role of Report.
- **&EM_NUM_RESIDUAL** — resolves to the number of variables that have the model role of Residual.
- **&EM_NUM_SEGMENT** — resolves to the number of variables that have the model role of Segment.
- **&EM_NUM_SEQUENCE** — resolves to the number of variables that have the model role of Sequence.
- **&EM_NUM_TEXT** — resolves to the number of variables that have the model role of Text.
- **&EM_NUM_TIMEID** — resolves to the number of variables that have the model role of Time ID.

Statements

Statements macro variables resolve to values that refer to information about decision variables and decision information. These macro variables are empty when there is more than one target variable.

- **&EM_DEC_TARGET** — resolves to the name of the target variable.
- **&EM_DEC_LEVEL** — resolves to the event level.

- **&EM_DEC_ORDER** — resolves to the sorting order of the target levels (ASCENDING | DESCENDING).
- **&EM_DEC_FORMAT** — resolves to the format of the decision target variable.
- **&EM_DEC_DECMETA** — resolves to the decision metadata data set of the target variable.
- **&EM_DEC_DECDATA** — resolves to the decision data set of the target variable.
- **&EM_DEC_STATEMENT** — resolves to the decision statement.

Code Statements


Use the Code Statements macro variable to identify the file containing the CODE statement.

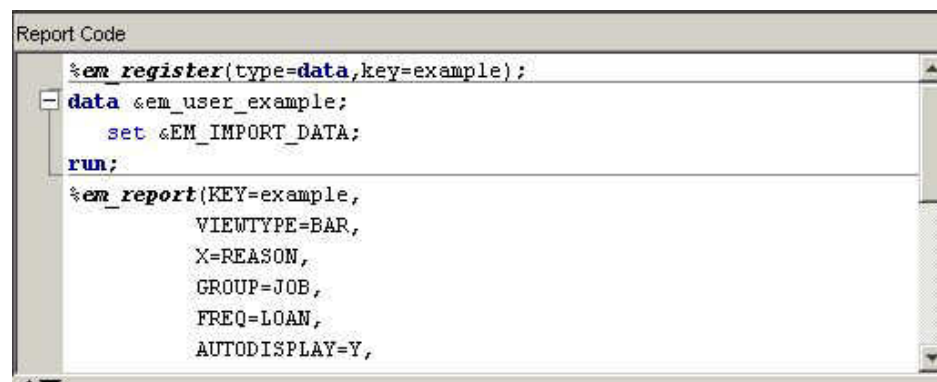
- **&EM_STATEMENT_RESCODE** — resolves to the file containing a CODE statement with a residuals option. In effect, this resolves to the file containing FLOW scoring code (&EM_FILE_EMFLOWSCORECODE).
- **&EM_STATEMENT_CODE** — resolves to the file for containing a CODE statement does not have a residuals option. In effect, this resolves to the file containing PUBLISH scoring code (&EM_FILE_EMPUBLISHSCORECODE).

Code Pane

The code pane is where you write new SAS code or where you import existing code from an external source. Any valid SAS language program statement is valid for use in the **SAS Code** node with the exception that you cannot issue statements that generate a SAS windowing environment. The SAS windowing environment from Base SAS is not compatible with SAS Enterprise Miner. For example, you cannot execute SAS/LAB from within a SAS Enterprise Miner **SAS Code** node.

The code pane has three views: Training Code, Score Code, and Report Code. You can use either the icons on the toolbar or the **View** menu to select the editor in which you want to work.

When you enter SAS code in the code pane, DATA steps and PROC steps are presented as collapsible and expandable blocks of code. The code pane itself can be expanded or contracted using the  icons located at the bottom left-side of the pane.



```
Report Code
%em_register(type=data,key=example);
data em_user_example;
  set &EM_IMPORT_DATA;
run;
%em_report(KEY=example,
  VIEWTYPE=BAR,
  X=REASON,
  GROUP=JOB,
  FREQ=LOAN,
  AUTODISPLAY=Y,
```

You can drag macros and macro variables from their respective tables into the code pane. This speeds up the coding process and prevents spelling errors. You can import SAS code that is stored as a text file or a source entry in a SAS catalog. If your code is in an external text file, then follow this example:

```
filename fref "path-name\mycode.sas";
%inc fref;
filename fref;
```

If your code is in a catalog, follow this example:

```
filename fref catalog "libref.mycatalog.myentry.source";
%inc fref;
filename fref;
```

The code in the three views is executed sequentially when the node is run. Training code is executed first, followed by Score code, and finally, Report code. Suppose, for example, that you make changes to your Report code but do not change your Training and Score code. When you run your node from within the Code Editor, SAS Enterprise Miner does not have to rerun the Training and Score code; it just reruns the Report code. This can save considerable time if you have complex code or very large data sets.

The three views are designed to be used in the following manner:

- **Training Code** — Write code that passes over the input training or transaction data to produce some result in the Training Code pane. Here is an example:

```
proc means data=&em_import_data;
output out=m;
run;
```

You should also write dynamic scoring code in the training code pane. Scoring code is code that generates new variables or transforms existing variables. Dynamic scoring code, as opposed to static scoring code, is written so that no prior knowledge of the properties of any particular data set is assumed. That is, the code is not unique to a particular process flow diagram. For example, suppose that you begin your process flow diagram with a particular data source and it is followed by a **SAS Code** node that contains dynamic scoring code. If you changed the data source in the diagram, the dynamic scoring code should still execute properly. Dynamic scoring code can use SAS PROC statements and macros, whereas static scoring code cannot.

- **Score Code** — Write code that modifies the train, validate, test, or transaction data sets for the successor nodes. However, the Score view is reserved for static scoring code. Static scoring code makes references to properties of a specific data set, such as variable names, so the code is unique for a particular process flow diagram. Here is an example:

```
logage= log(age);
```

If you write dynamic scoring code in the Score Code pane, it will not execute. Scoring code that is included in the Score Code pane must be in the form of pure DATA steps. SAS PROC statements and macros will not execute in the Score Code pane.

- **Report Code** — code that generates output that is displayed to the user. The output can be in the form of graphs, tables, or the output from SAS procedures. An example is a statement such as the following:

```
proc print data=m;
run;
```


Calls to the macro %EM_REPORT are the most common form of Report code.

You can execute your code in two modes.

- Run Code () — Code is executed immediately in the current SAS session.

Only the code in the active code pane is executed. The log and output appear in the

Code Editor's Results pane. If a block of code is highlighted, only that code is executed. No pre-processing or post-processing occurs. Use this mode to test and debug blocks of code during development.

- Run Node () — The code node and all predecessor nodes are executed in a separate SAS session, exactly as if the user has closed the editor and run the path. All normal pre-processing and post-processing occurs. Use the Results window to view the log, output, and other results generated by your code.

Most nodes generate permanent data sets and files. However, before you can reference a file in your code, you must first register a unique file key using the %EM_REGISTER macro and then associate a file with that key. When you register a key, SAS Enterprise Miner generates a macro variable named &EM_USER_key. You use that macro variable in your code to associate the file with the key. Registering a file enables SAS Enterprise Miner to track the state of the file and avoid name conflicts.

Use the %EM_GETNAME macro to reinitialize the macro variable &EM_USER_key when referring to a file's key in a code pane other than the one in which it was registered. Using Run Code causes the code in the active code pane to execute in a separate SAS session. If the key was registered in a different pane, &EM_USER_key will not be initialized. The registered information is stored on the server, so you don't have to register the key again, but you must reinitialize &EM_USER_key.

SAS Code Node Results

To view the **SAS Code** node Results window from within the Code Editor, click the



icon. Alternatively, you can view the Results window from the main SAS Enterprise Miner workspace by right-clicking the **SAS Code** node in the diagram and selecting **Results**.

Select **View** from the main menu in the Results window to view the following results:

- **Properties**
 - **Settings** — displays a window with a read-only table of the **SAS Code** node properties configuration when the node was last run.
 - **Run Status** — displays the status of the **SAS Code** node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — display a table of the variables in the training data set.
 - **Train Code** — displays the code that SAS Enterprise Miner used to train the node.
 - **Notes** — displays (in Read-Only mode) any notes that were previously entered in the Notes editor.
- **SAS Results**
 - **Log** — the SAS log of the **SAS Code** node run.
 - **Output** — The **SAS Code** node output report, like all other nodes, includes Training Output, Score Output, and Report Output. The specific contents are determined by the results of the code that you write in the **SAS Code** node.
 - **Flow Code** — the SAS code used to produce the output that the **SAS Code** node passes on to the next node in the process flow diagram.

- **Train Graphs** — displays graphs that are generated by SAS\GRAPH commands from within the Train code pane.
- **Report Graphs** — displays graphs that are generated by SAS\GRAPH commands from within the Report code pane.
- **Scoring**
 - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside of the SAS Enterprise Miner environment in custom user applications.
 - **PMML Code** — the **SAS Code** node does not generate PMML.
- **Assessment** — appears only if the Tool Type property is set to MODEL. By default, it contains a submenu item for Fit Statistics. However, you can generate other items by including the appropriate type code in the node.
- **Custom Reports** — appears as an item in the menu when you generate custom reports with %EM_REPORT. The title in the menu, by default, is Custom Reports, but that can be changed by specifying the BLOCK argument of the macro %EM_REPORT.
- **Table** — displays a table that contains the underlying data that is used to produce a chart.
- **Plot** — use the Graph wizard to modify an existing Results plot or create a Results plot of your own.

Score Code Export



Overview of the Score Code Export Node

The Score Code Export node is located on the Utilities tab of the Enterprise Miner workspace.

SAS Enterprise Miner creates SAS language score code for the purpose of scoring new data. Users run this code in production systems to make business decisions for each record of new data.

The Score Code Export node is an extension for SAS Enterprise Miner that exports files that are necessary for score code deployment. Extensions are programmable add-ins for the SAS Enterprise Miner environment.

The following files are exported by the Score Code Export node:

- the SAS scoring model program.
- an XML file containing scoring variables and other properties.
- an XML file containing descriptions of the final variables created by the scoring code. This file can be kept for decision-making processes.
- a ten-row sample of the scored data set showing typical cases of the input attributes, intermediate variables, and final output variables. This data set can be used to test and debug new scoring processes.

- a ten-row sample of the training data set showing the typical cases of the input attributes.
- a format catalog, if the scoring program contains user-defined formats.

For more information about the exported files, see [“Score Code Node Output” on page 122](#).

SAS Enterprise Miner can register models directly in the SAS Metadata Server. Models registered in the SAS Metadata Server are used by SAS Data Integration Studio, SAS Enterprise Guide, and SAS Model Manager for creating, managing, and monitoring production and analytical scoring processes.

The Score Code Export node exports score code created by SAS Enterprise Miner into a format that can be used by the SAS Scoring Accelerator. The SAS Scoring Accelerator supports many databases, including Teradata, Greenplum, DB2, Netezza, Oracle, and Hadoop. The exported files are stored in a directory, not the SAS Metadata Server.

The Score Code Export node does not replace the functionality of registering models in the SAS Metadata Server to be used by SAS Data Integration Studio, SAS Enterprise Guide, and SAS Model Manager.


Data Requirements of the Score Code Export Node

The Score Code Export node requires a process flow diagram that contains both a Score node and another node that creates score code. The Score node aggregates the score code for the entire process flow diagram and transfers it the Score Code Export node. Therefore, the Score node must directly precede the Score Code Export node.

Properties of the Score Code Export Node


Score Code Node General Properties

The following general properties are associated with the Score Code Export node:


- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Score Code Export node that is added to a diagram will have a Node ID of CodeXpt. The second Score Code Export node added to a diagram will have a Node ID of CodeXpt2, and so on.
- **Imported Data** — accesses the Imported Data — Regression window. The Imported Data — Regression window contains a list of the ports that provide data sources to the Regression node. Select the  button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Exported Data** — accesses the Exported Data — Regression window. The Exported Data — Regression window contains a list of the output data ports that the

Regression node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an imported data source, you can select the row in the imported data table and click:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a Table tab and a Variables tab. The tabs contains summary information (metadata) about the table and the variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Score Code Node Train Properties

- **Rerun** — Use this property to force the node to run again. This property is useful if the macro variable controlling the target directory and folder name have changed.
- **Output Directory** — Enter a fully qualified name for the location of an output directory to contain the score code files. If no directory is entered, a default directory named Score is created in the SAS Enterprise Miner project directory. You can change the value of the default directory by setting the `&EM_SCOREDIR=directory` macro variable in the SAS Enterprise Miner project start code or server start code.
- **Folder Name** — Enter the name of the model that you are creating. The name is used to create a new subdirectory in the output directory that contains the exported score files. If no name is entered, a default name is generated as a combination of the `&SYSUSERID` automatic macro variable and an incremental index (for example, `userID, userID_2, userID_3`).

You can replace the `&SYSUSERID` automatic macro variable with a custom name by setting the `&EM_SCOREFOLDER=score-folder-name` macro variable in the SAS Enterprise Miner project start code or server start code. An incremental index preceded by an underscore is added to *score-folder-name*.

Score Code Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time that the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Score Code Export Node Results

You can open the Results window of the Scorecard node by right-clicking the node and selecting **Results**. For general information about the Results window, see Using the Results Window.

Select **View** from the main menu to view the following results in the Scorecard Results window:

- **Properties**
 - **Settings** — displays a window with a read-only table of the Score Code Export node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
 - **Run Status** — indicates the status of the Score Code Export node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings, and you can toggle column sorts between descending and ascending by clicking on the column headings.
 - **Train Code** — the code that Enterprise Miner used to train the node.
 - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
 - **Log** — the SAS log of the Score Code Export node run.
 - **Output** — the SAS output of the Score Code Export node run. The Scorecard SAS output includes the following:
 - variables summary
 - output variables
 - files exported
 - **Flow Code** — the SAS code used to produce the output that the Score Code Export node passes on to the next node in the process flow diagram.
- **Scoring**
 - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the Enterprise Miner environment in custom user applications.
 - **PMML Code** — The Score Code export node does not generate PMML code.
 - **Input Variables** — a read-only table of the input variables for the Score Code Export node.
 - **EM Output Variables** — a read-only table of the output variables for the Score Code Export node. This table indicates the role, create, type, label, and length of each output variable.
 - **Summary** — A table that indicates who ran the node, when it was run, and where the output files are located.
- **Table** — is unavailable for this node.

- **Plot** — opens the Graph Wizard for the table that you select.

Score Code Node Output

Score Code Export Node Output Files

The Score Code Export node writes the following output files, and possibly a formats catalog, to the location specified by the Output Directory property. These files are used as input to the %INDTD_PUBLISH_MODEL macro that creates the scoring functions.

- **score.sas** — SAS language score code that is created by Enterprise Miner. This code can be used directly in a SAS program.
- **score.xml** — A description of the variables that are used and created by the scoring code. XML files are created by a machine process for the use of machine process. Do not edit the XML file.

Note: The maximum number of input variables for a UDF function is 128.

- **emoutput.xml** — A description of the final variables that are created by the scoring code. This file can be kept for decision-making processes. These variables include the primary classification, prediction, probability, segment, profit, and loss variables that are created by a data mining process. The list does not include intermediate variables that are created by the analysis. For more information, see Fixed Variable Names.
- **scoredata.sas7bdat** — A ten-row sample of the score data set that shows typical cases of the input attributes, intermediate variables, and final output variables. Use this data set to test and debug new scoring processes.
- **traindata.sas7bdat** — A ten-row sample table of the training data set that shows typical cases of the input attributes, intermediate variables, and final output variables.
- **Formats Catalog** — If the training data contains SAS user-defined formats, the Score Code Export node creates a formats catalog. The catalog contains the user-defined formats in the form of a look-up table. This file has an extension of .sas7bcat.

Score Code Export Node Output Variables

The Score Code Export node creates score code with both intermediate variables and output variables. Intermediate variables include imputed values of missing variables, transformation variables, and encoding variables. Output variables include predicted values and probabilities. Any of the intermediate or output variables can be used in a scoring process.

The number of input parameters on a scoring function has a direct impact on performance. The more parameters there are, the more time it takes to score a row. A recommended best practice is to ensure that only variables that are involved in a model score evaluation are exported by the Score Code Export node.

The most important output variables use a prefix and follow the naming conventions outlined in the following table:

Role	Type	Prefix	Key	Suffix	Example
Prediction	N	P_	Target Variable Name		P_amount

Role	Type	Prefix	Key	Suffix	Example
Probability	N	P_	Target Variable Name	Predicted event value	P_purchaseYES P_purchaseNO
Classification	\$	I_	Target Variable Name		I_purchase
Expected Profit	N	EP_	Target Variable Name		EP_conversion
Expected Loss	N	EL_	Target Variable Name		EL_conversion
Return on Investment	N	ROI_	Target Variable Name		ROI_conversion
Decision	\$	D_	Target Variable Name		D_conversion
Decision Tree Leaf	N	_NODE_			_NODE_
Cluster Number or SOM cell ID	N	_SEGMENT_			_SEGMENT_

Fixed Variable Names

The Score node maps the output variable names to fixed variable names. This mapping is appropriate in the most common case that there is only one prediction target or one classification target. The table below will help make sense of other cases.

Using the fixed variable names enables scoring users to build processes that can be reused for different models without changing the code that processes the outputs. These fixed names are listed in the **emoutput.xml** file and are described in the table below. Most scoring processes return one or more of these variables.

Role	Type	Fixed Name	Description
Prediction	N	EM_PREDICTION	The prediction value for an interval target variable.
Probability	N	EM_PROBABILITY	The probability of the predicted classification, which can be any one of the target variable values.

Role	Type	Fixed Name	Description
Probability	N	EM_EVENTPROBABILITY	The probability of the target event. By default, this is the first value in descending order, but you can alter the ordering scheme. This is often the event of interest.
Classification	\$	EM_CLASSIFICATION	The predicted target class value.
Expected Profit	N	EM_PROFIT	Based on the selected decision.
Expected Loss	N	EM_LOSS	Based on the selected decision.
Return on Investment	N	EM_ROI	Based on the selected decision.
Decision	\$	EM_DECISION	Optimal decision based on a function of probability, cost, and profit or loss weights.
Decision Tree Leaf, Cluster number, or SOM cell ID	N	EM_SEGMENT	Analytical customer segmentation.

SAS Enterprise Miner Tools Production of Score Code

The table below shows the types of score code that is created by each node in SAS Enterprise Miner. In addition, users can develop extension nodes, which can create either SAS DATA step or SAS program score code. However, this code is not converted to PMML, C, or Java.

Node	SAS DATA Step	SAS Program	PMML	C	Java	Database UDF (Teradata, Greenplum, DB2, Netezza, Oracle, Hadoop)
Sample						
Input Data	*	*	*	*	*	*
Sample	*	*	*	*	*	*

Node	SAS DATA Step	SAS Program	PMML	C	Java	Database UDF (Teradata, Greenplum, DB2, Netezza, Oracle, Hadoop)
Data Partition	*	*	*	*	*	*
File Import	*	*	*	*	*	*
Append	N	Y	N	N	N	N
Merge	N	Y	N	N	N	N
Time Series	N	Y	N	N	N	N
Filter	Y When the user keeps the created filter variable.	*	N	Y	Y	Y
Explore						
Association	N	Y	Y	N	N	N
Cluster	Y	N	Y	Y	Y	Y
DMDB	*	*	*	*	*	*
Graph Explore	*	*	*	*	*	*
Link Analysis	N	Y	N	N	N	N
Market Basket	N	N	N	N	N	N
Multiplot	*	*	*	*	*	*
Path Analysis	N	Y	Y	N	N	N
SOM/Kohonen	Y	N	N	Y	Y	Y
Stat Explore	*	*	*	*	*	*

Node	SAS DATA Step	SAS Program	PMML	C	Java	Database UDF (Teradata, Greenplu m, DB2, Netezza, Oracle, Hadoop)
Variable Clustering	Y	N	N	Y	Y	Y
Variable Selection	Y	N	N	Y	Y	Y
Modify						
Drop	*	*	*	*	*	*
Impute	Y	N	Y	Y	Y	Y
Interactive Binning	Y	N	N	Y	Y	Y
Principal Componen ts	Y	N	N	Y	Y	Y
Replaceme nt	Y	N	N	Y	Y	Y
Rules Builder	Y	N	N	Y	Y	Y
Transform Variables	Y	N	N	Y	Y	Y
Model						
AutoNeura l	Y	N	Y	Y	Y	Y
Decision Tree	Y	N	Y	Y	Y	Y
Dmine Regression	Y	N	Y	Y	Y	Y
DMNeural	Y	N	N	Y	Y	Y
Ensemble	Y	N	N	Y	Y	Y
Gradient Boosting	Y	N	N	Y	Y	Y
LARS	Y	N	N	Y	Y	Y
MBR	N	Y	N	N	N	N

Node	SAS DATA Step	SAS Program	PMML	C	Java	Database UDF (Teradata, Greenplum, DB2, Netezza, Oracle, Hadoop)
Model Import	*	*	*	*	*	*
Neural Network	Y	N	Y	Y	Y	Y
Partial Least Squares	Y	N	N	Y	Y	Y
Regression (linear, logistic, and multinomial)	Y	N	Y	Y	Y	Y
Rule Induction	Y	N	N	Y	Y	Y
Two Stage	Y	N	N	Y	Y	Y
Assess						
Cutoff	Y	N	N	Y	Y	Y
Decisions	Y	N	N	Y	Y	Y
Model Comparison	Y	N	N	Y	Y	Y
Score	Y	N	N	Y	Y	Y
Segment Profile	*	*	*	*	*	*
Utility						
Control Point	*	*	*	*	*	*
End Groups	Y	N	N	Y	Y	Y
Ext Demo	*	*	*	*	*	*
Metadata	*	*	*	*	*	*

Node	SAS DATA Step	SAS Program	PMML	C	Java	Database UDF (Teradata, Greenplum, DB2, Netezza, Oracle, Hadoop)
Open Source Integration	Y	N	Y	N	N	N
Register Mode	N	N	N	N	N	N
Reporter	*	*	*	*	*	*
SAS Code You can enter either SAS DATA step code or SAS program code.	Y	Y	N	N	N	N
Save Data	*	*	*	*	*	*
Score Code Export	N	N	N	N	N	N
Start Groups	Y	N	N	Y	Y	Y
Credit Scoring						
Credit Exchange	*	*	*	*	*	*
Interactive Grouping	Y	N	N	Y	Y	Y
Reject Inference	Y	N	N	Y	Y	Y
Scorecard	Y	N	N	Y	Y	Y
HPDM Nodes						
HP BN Classifier	Y	N	N	N	N	Y
HP Cluster	Y	N	N	N	N	Y
HP Data Partition	*	*	*	*	*	*

[illegible]

Chapter 9

Applications

Survival Analysis	131
Overview of the Survival Node	131
Input Data Requirements for the Survival Node	133
Overview of Discrete Time Logistic Hazard Modeling Using the Survival Node	134
Prepare the Data for Survival Analysis	134
Expand the Data for Survival Analysis	136
Sample the Expanded Data	137
Configure and Run Survival Model	138
Validate the Survival Model	138
Specify Reporting Options	140
Model Scoring	141
Survival Modeling with Time-Dependent Covariates	143
Survival Node Properties	150
Survival Node Results	155
Survival Node Example	159
Time Series	169
Credit Scoring	170
Incremental Response	170
Overview	170
Input Data Requirements for the Incremental Response Node	171
Missing Values in Incremental Response Node Input Data	172
Incremental Response Node Properties	172
Incremental Response Node Results	176
Incremental Response Node Output Variables	178
Incremental Response Node Example	179
References	187

Survival Analysis

**Overview of the Survival Node**

Survival data mining is the application of survival analysis to data mining problems that concern customers.. The application to the business problem changes the nature of the

statistical techniques. The issue in survival data mining is not *whether* an event will occur in a certain time interval, but *when* the next event will occur.

The SAS Enterprise Miner **Survival** node is located on the Applications tab of the SAS Enterprise Miner tool bar. The **Survival** node performs survival analysis on mining customer databases when there are time-dependent outcomes. Some examples of time-dependent outcomes are as follows:

- customer churn
- cancellation of all products and services
- unprofitable behavior
- server downgrade or extreme inactivity

The time-dependent outcomes are modeled using multinomial logistic regression. The discrete event time and competing risks control the occurrence of the time-dependent outcomes.

The discrete event time represents the duration from the inception (start) time until the censoring date. Discrete event times are represented by nonnegative integer values. Functions of the discrete event time (in the form of cubic spline basis functions) are used as predictors in the multinomial regression. The hazard and subhazard functions that the **Survival** node generates are based on the multinomial logistic regression. The hazard function is a conditional probability of an event at time t . The hazard and subhazard functions depend on the discrete event time. Many times the discrete event time function is nonlinear in nature. Transforming the event time function with cubic spline basis functions allows the hazard and subhazard functions to be more flexible. This results in a greater ability to detect and model customer behavior patterns.

A key element of survival data mining is the concept of *competing risks*. In a traditional medical survival analysis, a patient is considered “lost to follow up” if the patient dies, or moves out of state. The same concept also applies to survival data mining.

An example of competing risks in survival data mining is found in voluntary and involuntary churn. For example, some customers are forced to leave when their account remains delinquent too long. Other customers leave voluntarily by transferring selected services or by changing service providers altogether. Any action that terminates the customer’s relationship with a provider is a competing risk. After a customer has experienced one of the competing risk events, that customer is excluded from the at-risk population for any of the other competing risks. In other words, competing risks are mutually exclusive and exhaustive.

Censoring is a nearly universal feature of survival data. Censoring occurs in many forms for many different reasons. When the data is extracted, usually not all customers have experienced the event. An observation is considered to be censored if the event has not yet occurred by the end date that is chosen.

All of the modeled events are time-dependent because the probability distribution of the time until the event controls their occurrence.

The **Survival** node includes functional modules that prepare data for mining, that expand data to one record per time unit, and perform sampling to reduce the size of the expanded data without information loss. The **Survival** node also performs survival model training, validation, scoring, and reporting.

Note: The **Survival** node is an application node that does not utilize frequency variables. Frequency variables are not recommended for use in **Survival** node models.

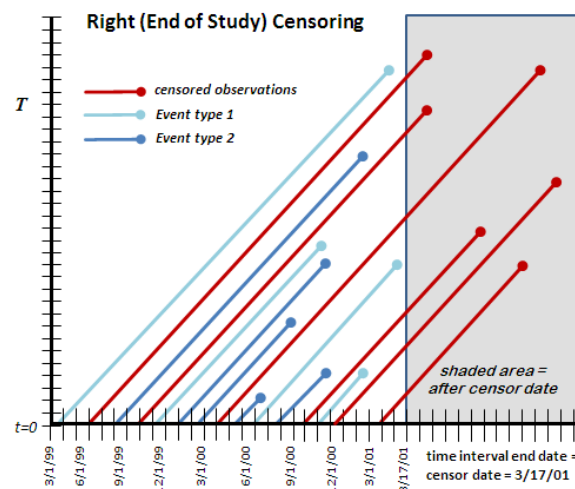
Input Data Requirements for the Survival Node

The **Survival** node requires that the input data submitted for analysis meet the following requirements:

- The input data must have a unique ID variable (such as customer ID) for observations.
- At least two TIMEID variables are required. The first TIMEID variable maps to the inception, or start date. The second TIMEID variable maps to the event date.
- The event TIMEID variable must be generated before data is submitted to the **Survival** node for analysis. The event TIMEID variable cannot be the result of a function that uses other columns in the data set as inputs.
- At least one input variable is required for predictive hazard modeling using the **Survival** node.

There must be one target variable that represents the type of event that occurs on the event date. The target variable is subject to the following constraints:

- Target variable values must be numeric.
- The target variable must be a class variable. Interval values are not permitted.
- The target variable must represent the different outcome levels that are associated with an event. Event outcome levels are discrete values that belong to a mutually exclusive and exhaustive set. For example, if you are modeling churn, you might use three target variable levels. Voluntary churn could be represented by a target variable value of 1. Involuntary churn could be represented by a target variable value of 2. Observations that do not contain a churn event within the defined analysis interval could be represented by a target variable value of 0. Observations that have a target variable value of 0 are said to be *censored* because no churn event occurred during the analysis interval. Censoring is a term that describes observations for which the exact event time is unknown. Churn events that might occur after the analysis interval are not relevant to the analysis. The target variable value for all censored observations must be 0.



Overview of Discrete Time Logistic Hazard Modeling Using the Survival Node

Use of the SAS Enterprise Miner **Survival** node in data mining follows a specific process:


- **Prepare the Data for Survival Analysis**
- **Expand the Data for Survival Analysis**
- **Sample the Expanded Data**
- **Configure and Run the Survival Model**
- **Validate the Survival Model**
- **Specify Reporting Options**
- **Model Scoring**

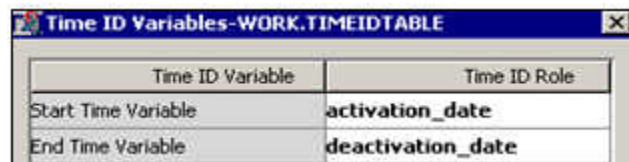
Prepare the Data for Survival Analysis

The **Survival** node supports three types of input data formats: **Standard**, **Change-Time**, or **Fully Expanded**. The following narrative uses the **Standard** data format. You use the **Change-Time** and **Fully Expanded** data formats when your model uses time-dependent covariates, a structure requiring multiple rows per ID variable showing changing covariable values. For more information about Survival analysis with time-dependent covariates, see [“Survival Modeling with Time-Dependent Covariates” on page 143](#).

The data to be mined for survival modeling must be configured for survival analysis in the **Survival** node properties. You must identify the time ID variables, as well as the basic unit of time that will be the basis for censoring the data. You use the SAS Enterprise Miner **Survival** node properties **Time ID variables** and **Time Interval** to specify the required information.

Specify Time ID variables: You must configure your survival analysis data set so that SAS Enterprise Miner can identify the variable roles that are required to perform the time interval heuristics.

The **Time ID variables** and **Time Interval** properties specify how the input data to be analyzed is censored. Selecting the  button to the right of the **Time ID variables** property enables you to specify the mapping for the start and end date variables, as follows:



The screenshot shows a dialog box titled "Time ID Variables-WORK.TIMEIDTABLE". It contains a table with two columns: "Time ID Variable" and "Time ID Role".

Time ID Variable	Time ID Role
Start Time Variable	activation_date
End Time Variable	deactivation_date

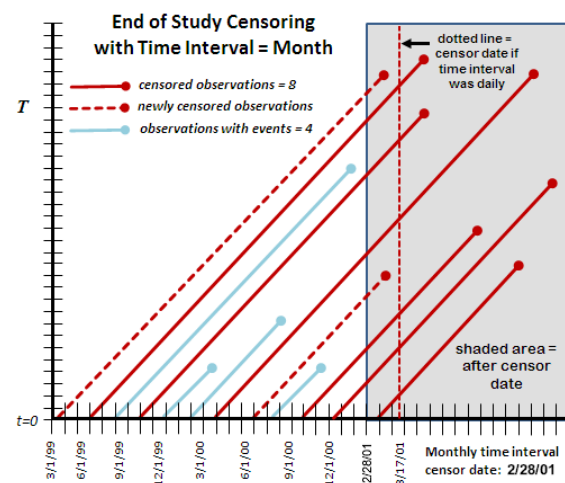
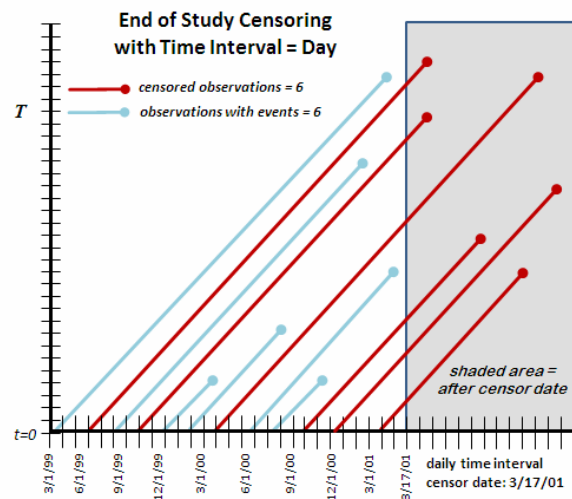
SAS Enterprise Miner scans the input data and chooses the start date for modeling. It does so by finding the minimum date value across all observations for the variable specified as the **Start Time Variable**. SAS Enterprise Miner chooses the modeling end date by finding the maximum date value across all observations for the variable that you specify in the **End Time Variable** field. The censoring date is based on the maximum date for the **End Time Variable** and the **Time Interval** that you selected.

Note that for **Change Time** format (for which there are three required Time ID variables), SAS Enterprise Miner cannot automatically select these variable roles. You must manually specify the proper roles for the Time ID variables.

You use the **Time Interval** property to specify the time interval that you would like to use for censoring and reporting. The available time intervals are as follows:

- Day
- Week
- Month
- Quarter
- Semi-year
- Year

To see how the **Time ID variables** and **Time Interval** properties work in conjunction to censor the data, consider the following examples:



Note that the two data plots have different censor dates due to the chosen time period. Remember that the **Survival** node chooses the modeling end date by finding the maximum date value across all observations for the variable that you specify in the **End Time Variable** field. When the **Time Interval** property is configured as *Day*, the

maximum date value for the day time unit is March 17, 2001. When the **Time Interval** property is configured as *Month*, the maximum date value for the month time unit is February 28, 2001.

The difference in censor dates (related to the choice of time interval units) can affect the data to be analyzed and modeled. Note that the example graph for the daily time interval contains multiple event types. There are 9 voluntary churn events, 6 involuntary churn events, and 6 instances where no event is observed before the censor date. In the example graph for the monthly time interval, the interval numbers vary: there are 9 voluntary churn events, 4 involuntary churn events, and 8 instances where no event is observed before the censor date. When months are specified as the time interval, the change in the censor date due to the larger time units affects the event type classifications. Two customers experienced involuntary churn events on March 15, 2001 when the time interval is specified in days and the censor date is March 17, 2001. However, if the time interval is specified in months, the two customer events on March 15 occur *after* the monthly censor date of February 28, 2001. This changes the event type classification for those two customers from involuntary churn events to censored.

Expand the Data for Survival Analysis

The training data set for the **Survival** node requires one observation for each unique customer (account number in the example below) when using the standard format. The time span for the training data set ranges from the start time to the end time (censoring time). After you configure your data for survival analysis by specifying the **Time ID variables** and **Time Interval** properties, the **Survival** node expands the training data set. The training data set is expanded so that each customer has one record for each incremental time interval in which the customer was observed. For example, the following shows the first ten observations in the training data with one observation per ID value (account number):

	Account Number	Event Time	Event Type	Good Bad Credit Indicator	Disable Reason	Type of Rate Plan	Activation Date	Deactivation Date	Provider activation Date	Provider Type
1	180437080184	16	0	1		3	09/28/1999		PROV1	PROV1
2	180437283474	0	0	1		1	01/09/2001		PROV1	PROV1
3	180437340410	13	0	0		1	12/31/1999		PROV1	PROV1
4	18043736568	6	2	0	DUE	1	12/22/1999	05/28/2000	PROV2	PROV2
5	18043736837	9	0	1		1	04/17/2000		PROV3	PROV3
6	180437375280	12	1	1	TRANSFER	2	08/15/1999	08/21/2000	PROV1	PROV1
7	180437382909	18	0	1		1	07/26/1999		PROV3	PROV3
8	180437420657	13	0	0		1	12/15/1999		PROV2	PROV2
9	18043743673	2	0	0		3	11/21/2000		PROV1	PROV1
10	180437452331	1	0	0		2	12/28/2000		PROV3	PROV3

The 0 value in the **Event Type** column of the first observation means that the customer did not experience an event between the start time and the censoring date. The **Event Time** column for the customer in observation 1 shows that that customer has been active for 16 time periods (months). In the expanded data set shown below, the single observation for customer 1 becomes 17 observations, with one row per observed time period (month).

	Account Number	Event Time	Discrete Event Time	Event Type	Good Bad Credit Indicator	Disable Reason	Type of Rate Plan	Activation Date	Deactivation Date	Provider Type
1	180437080184	16	0	0	1		3	09/28/1999		PROV1
2	180437080184	16	1	0	1		3	09/28/1999		PROV1
3	180437080184	16	2	0	1		3	09/28/1999		PROV1
4	180437080184	16	3	0	1		3	09/28/1999		PROV1
5	180437080184	16	4	0	1		3	09/28/1999		PROV1
6	180437080184	16	5	0	1		3	09/28/1999		PROV1
7	180437080184	16	6	0	1		3	09/28/1999		PROV1
8	180437080184	16	7	0	1		3	09/28/1999		PROV1
9	180437080184	16	8	0	1		3	09/28/1999		PROV1
10	180437080184	16	9	0	1		3	09/28/1999		PROV1
11	180437080184	16	10	0	1		3	09/28/1999		PROV1
12	180437080184	16	11	0	1		3	09/28/1999		PROV1
13	180437080184	16	12	0	1		3	09/28/1999		PROV1
14	180437080184	16	13	0	1		3	09/28/1999		PROV1
15	180437080184	16	14	0	1		3	09/28/1999		PROV1
16	180437080184	16	15	0	1		3	09/28/1999		PROV1
17	180437080184	16	16	0	1		3	09/28/1999		PROV1

Creating an expanded data set can easily result in very large training data sets that are impractical to use for modeling. The **Survival** node provides a sampling method that you can use to reduce the size of the training data set while minimizing the loss of information.

Sample the Expanded Data

Expanding the modeling event data to represent one customer record per unit time can quickly create very large input data tables that are impractical to use for modeling. Use the **Survival** node sampling properties to specify a sampling method that will reduce the size of the time series training table and minimize the loss of information.

To enable sampling in the **Survival** node, set the **Sampling** property to **Yes**. Use the **Event Proportion** property to specify the proportion of events that you want to include in your sample. The default setting for the **Event Proportion** property is 0.20, or 20%.

For example, suppose you have a 10% event rate in your training data, but you would like a 20% event rate in your sample. The **Survival** node can increase the relative occurrence of events in a sample data set by eliminating observations from the expanded data that do not have an event. The **Survival** node sampling algorithm does not remove any observations that have event outcomes. Only observations with no event are removed during sampling. Here is an overview of how the **Survival** node sampling algorithm operates:

1. Let π represent the training data event rate, which is the proportion of observations in the fully expanded training data. In our example, $\pi = 0.10$. Let ρ represent the desired proportion of events in the sample. In our example, $\rho = 0.20$.
2. The **Survival** node performs heuristic checks to ensure that the expanded training data set contains enough events to support the desired proportion of events in the sample to be created. If $\rho < \pi$, then the algorithm sets $\rho = \pi$. Otherwise, ρ remains equal to the proportion specified in the properties panel.
3. All observations with events are retained. The remaining observations not associated with an event are randomly sampled with selection probability $P(\text{non-event observations}) = [((1 - \rho)\pi)/((1 - \pi)\rho)]$. Because all observations with events are retained, the probability that a given observation in the training data set is selected for the sample is $P(\text{event observations}) = 1$.

For a training data set with event rate $\pi = 0.10$ and a desired sampled data set with event rate $\rho = 0.20$, the probability for a given non-event observation to be selected from the training data as part of the sample data set is $[((1 - 0.20)*0.10)/((1 - 0.10)*0.20)] = 0.44 = 44\%$. The sample data set will be created from the training data by selecting all observations with an event, and 44% of the observations with no event selected at random. The model is then built with the biased sample data set as the training data.

Using the biased sample to train the model allows the predicted subhazard functions to be more precisely estimated than compared to a random sample. To correct the bias caused by sampling, the subhazard functions are adjusted after the model is built. The logit function for each of the subhazard functions is adjusted by adding the log of the selection function: $\ln[((1 - \rho)\pi)/((1 - \pi)\rho)]$. The adjustment factor for the subhazard functions can be viewed in the output scoring code of the **Survival** node.

This sampling technique is a well-known method that is used to handle rare categorical outcomes, and is known as *case-control* or *choice-based* sampling.

Configure and Run Survival Model

The basis of survival data mining is the hazard and subhazard functions. These functions represent the chance that a customer account that has existed for a given span of time is going to cancel service before the next unit of time. The subhazard function simply represents the conditional probability of an event occurrence of type n at time t , given that no event has occurred before time t . The **Survival** node uses observations that have a target value of 0 (observations with no observed event, or censored observations) as the reference level. The hazard and subhazard functions can be suitably modeled using multinomial regression models.

The discrete time variable determines the shape of the hazard function. The multinomial regression model should include an appropriate parameterization of time. The discrete time variable is often non-linear in nature. One effective approach that allows for more flexibility in the time effect is to use regression spline terms as transformations of the discrete time effect. The transformations are included in the multinomial regression. The specific discrete time transformations that the **Survival** node uses are cubic spline basis functions. The cubic spline basis functions are of the following form:

$$csb(t, k_j) = \begin{cases} -t^3 + 3k_j t^2 - 3k_j^2 t & \text{if } t \leq k_j \\ -k_j^3 & \text{if } t > k_j \end{cases}$$

The preceding assumes that j is the number of knots and k is the value of the knot.

The cubic spline basis functions are segmented functions that consist of polynomials. The join points between the segments are called knots. The knots are points where the function makes a transformation. For example, a knot is the point at which one of the cubic spline basis functions changes from a cubic function to a constant function. Use the **Number of Knots** property in the Regression Spline Model section of the **Survival** node properties panel to specify how many knots the cubic spline basis functions will have.

The Regression Spline Model section of the **Survival** node properties panel also contains a **Stepwise Regression** property. When you set the **Stepwise Regression** property to Yes, the multinomial regression model that is created will use stepwise regression. Use the **Entry Significance Level** and **Stay Significance Level** properties to specify the p-value settings for model effects. When you configure the **Survival** node to perform stepwise regression, the discrete time effect is not part of the stepwise variable selection procedure. You can choose to include cubic spline basis functions in the stepwise regression procedure if you set the **Knot Selection** property to **Yes**. Otherwise, discrete time effects and cubic spline basis functions are always used as inputs in the multinomial regression model.

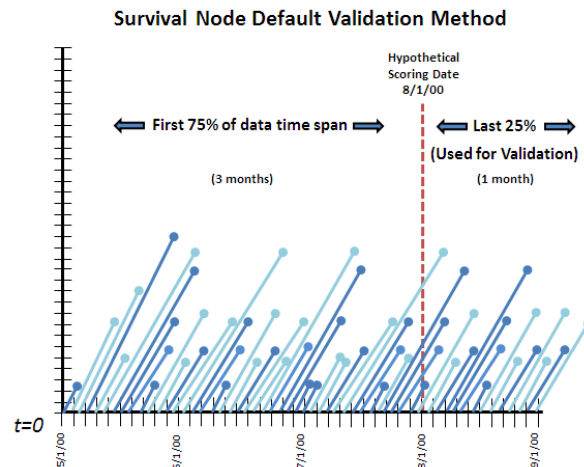
Validate the Survival Model

The measure of validity for models that you create is based on the model's performance while using a subset of the data. Survival model validation can be challenging, because event outcomes are time dependent. It is normal for some of the survival data in your validation range to be censored. This is because some observations in the validation data will not have outcomes within the allotted time interval. Data for the **Survival** node is organized chronologically, so the **Survival** node validates survival models by using a subset of the time interval. For each of the imported data sets for the **Survival** node

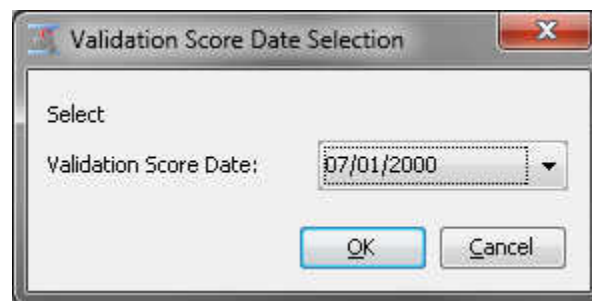
(Train, Validate, and Test), a subset is taken based on the Model Validation properties that define the hypothetical scoring date and scoring interval as described below. Model validation (based on the model that was built using the entire training data set) is performed with the subset of observations in the scoring interval from the train data set, and when available, subsets from the validation and test data sets are also used.

The **Survival Validation Method** property has two settings: **Default** and **User Specified**. The **Default** method automatically assigns values to the **Validation Score Date** and **Interval Length** properties for you. The **User-Specified** method enables you to specify your own values for **Validation Score Date** and **Interval Length**.

The **Validation Score Date** property represents a hypothetical scoring date within the time interval that is represented by the data used for model validation. If the **Survival Validation Method** is set to **Default**, then the time interval used for model validation is divided into quarters. The date most closely associated with the beginning of the last quarter of the interval automatically becomes the hypothetical scoring date (**Validation Score Date**) for the data used for model validation.



If the **Survival Validation Method** is set to **User Specified**, you can use the **Validation Score Date** property to open the Score Date Selection window. Here you can manually specify your own hypothetical scoring date from the time interval that is represented by the data used for validation.

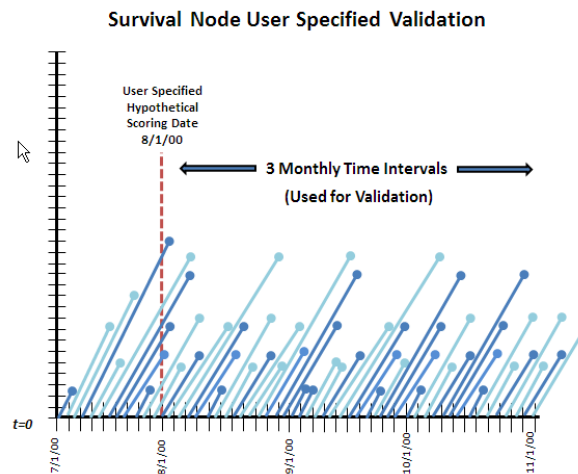


The **Interval Length** property represents the span of time that follows the hypothetical scoring date to be used for model validation. If the **Survival Validation Method** is set to **Default**, then the **Interval Length** property is automatically set to the last quarter of the model validation data interval. This interval is the time remaining between the automatically assigned **Validation Score Date** and the end of the time interval that is represented by the data used for validation. If the **Survival Validation Method** is set to **User Specified**, you must use the **Interval Length** property to define the time interval to

be used for model validation. You do so by entering a scalar multiple of the time unit that is defined in the Survival Train **Time Interval** property.

Survival Validation	
Survival Validation Method	User Specified
Validation Score Date	...
Interval Length	2

For example, if the Train **Time Interval** property is set to *Month*, then entering a value of 3 for the **Interval Length** property would result in a validation interval of three months. The three-month validation interval begins immediately after the hypothetical scoring date, or **Validation Score Date**.



CAUTION:

If you manually specify values for the **Validation Score Date** and **Interval Length** properties, the **Interval Length** end point must fall within the validation data time interval.

Specify Reporting Options

Use the Reporting properties for the **Survival** node to specify reporting options for your model output tables. Report tables provide information about the customers with the highest event probabilities for training, validation, and test data sets.

The **Survival** node report tables identify the customers with the highest likelihood for occurrence of churn, or the given event of interest. You can use the **High Risk Account Tables** option to specify how to report on high-risk customers. The **High Risk Account Tables** settings are as follows:

None

No report tables are generated. When **High Risk Account Tables** is set to **None**, the remaining properties in the **Survival Report** section become dimmed and unavailable.

Event Probability

The report table sorts customers by descending probability of experiencing the event of interest.

Survival Probability

The report table sorts customers by descending probability of survival.

Hazard Probability

The report table sorts customers by descending overall hazard probability.

All

All three report tables (Event Probability, Survival Probability, and Hazard Probability) are generated.

The reporting tables are useful for studying customer retention. For example, the top 100 customers with the highest survival probabilities could be sent a promotion to retain them as customers.

If you want to report the top number of customers, use the **Fixed Numbers** setting for the **Risk Account Selection** property. This enables the **Number** property, which you use to specify the number of customers that you would like summarized in your report table.

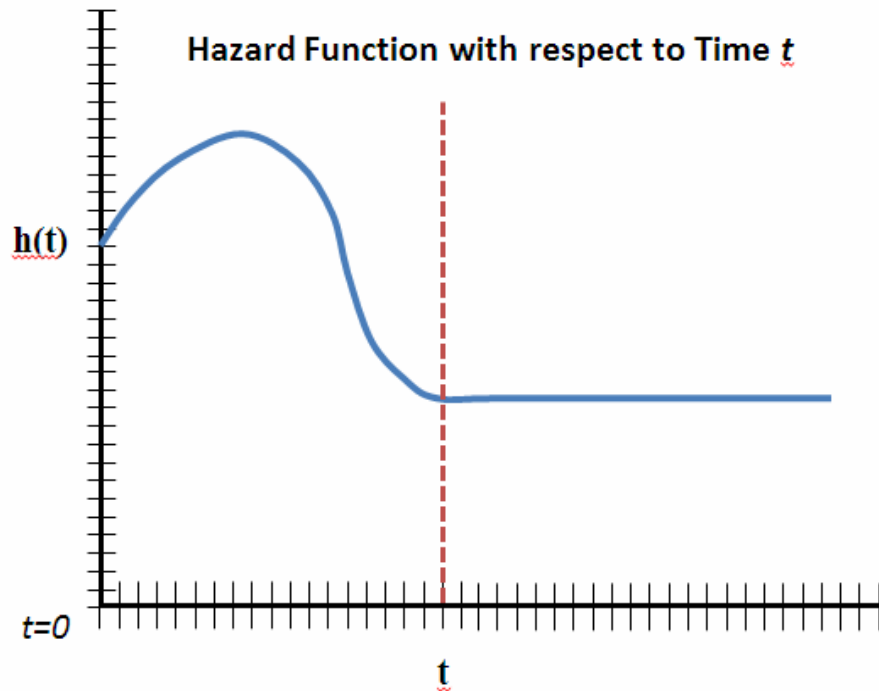
If you would like to report on high-risk customers using percentiles, use the **Percentiles** setting for the **Risk Account Selection** property. This enables the **Percentile** property, where you can choose the top n^{th} percentile of customers that you would like included in the report.

Model Scoring

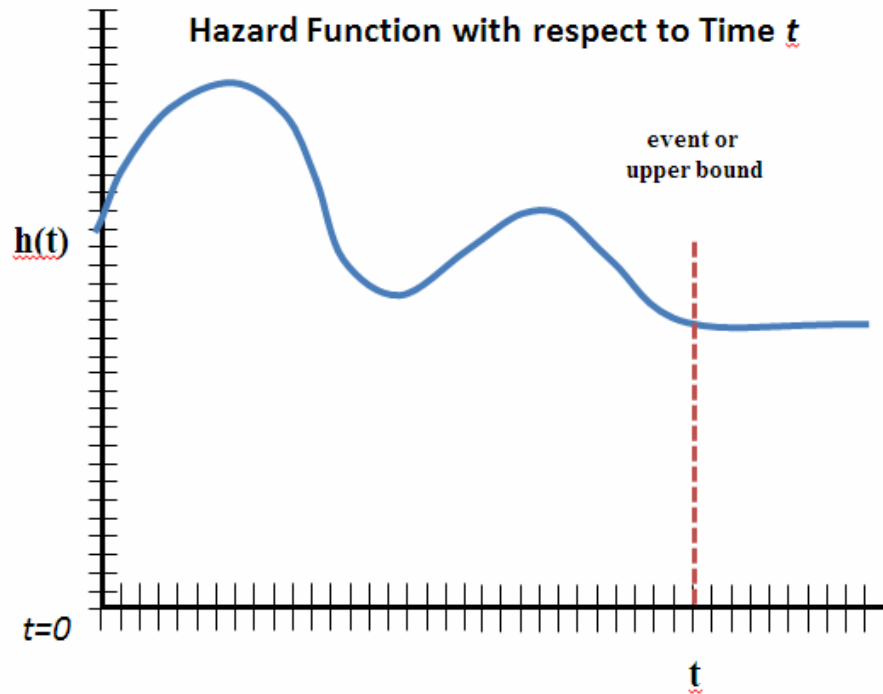
The last step in the Survival data mining model process is to score the data. Use the **Survival** node Score properties to configure your model's scoring output.

You use the first three Score properties for the **Survival** node (**Mean Residual Life**, **Default Maximum MRL**, and **User-Specified Maximum MRL**) to specify options for Mean Residual Life (MRL). In survival data mining, *mean residual life* is the time that remains until an event occurs, such as an existing customer's terminating a business relationship.

The SAS Enterprise Miner **Survival** node computes two types of mean residual life, constant hazard extrapolation and the restricted mean. The constant hazard extrapolation MRL assumes that at some point in time t , the hazard function becomes constant, as shown in the graph below:



For many business survival modeling problems, it makes more sense to use a restricted mean MRL. The restricted mean MRL goes to a constant hazard function when either the event of interest occurs, or when some upper limit is reached. The maximum value (or upper bound) often represents the limit on a meaningful remaining lifetime, such as a cable TV or cell phone contract duration. For example, a housing loan might be bounded by the terms of a mortgage. The mortgage might be 30 years in duration, but some customers pay off the loan in 15 years. All customers who paid off their loan in 15 years would have a constant hazard function value from the 15-year point going forward. This allows customers whose residual life (time remaining until the event) exceeds the time horizon to be considered equal. The following plot illustrates a restricted mean MRL:



If you use the restricted mean MRL, you can specify a maximum (upper bound) by using the **Default Maximum MRL** and **User Specified MRL** properties.

The future value of a customer depends on the remaining lifetime. Mean residual life is concerned with scoring future time intervals with the hazard model. If you compute mean residual life, the **Default Forecast Intervals** and **Number of Forecasted Intervals** properties specify the number of time units into the future that the mean residual life function will use for calculations.

When you specify how many time units into the future you would like the mean residual life to be computed, you will also obtain a future survival function as well. For example, if you select *Yes* in the **Default Forecast Interval** property, and set the **Time Interval** property to *Year*, then the survival probabilities are calculated for every customer in the training data set. The survival probability indicates the likelihood that a given current customer will still be a customer one year from the time that the model was trained.

Survival Modeling with Time-Dependent Covariates

Overview

Some survival models need to measure how the values of certain variables related to an ID change over time. For example, Internet, phone, and cable service providers might track the type and number of complaints customers register in order to predict churn, or medical care providers might track the number of symptoms a patient might exhibit prior to death or incapacitation. Problems such as these require data that is formatted with multiple rows per ID, in order to show changing covariate values over time. The **Survival** node uses the **Fully Expanded** and **Change-Time** data formats to perform time-dependent covariate survival modeling.

All inputs in the **Change-Time** and **Fully Expanded** data formats are treated as time-dependent. It is not necessary to distinguish any of the variables that do not change over time. For interval inputs, the node uses the weighted average over the time interval. For class inputs, the last recorded value for a time interval is used. When scoring, the current

value at censor time is used. Only the last observation for an ID (with the change date before the score or censor time) is used for scoring, validation, or for empirical function evaluation.

Choosing Your Time Dependent Covariate Data Format

You must consider several criteria when choosing the data format for a time-varying covariate survival data mining problem.

Generally, the **Change-Time** format allows for smaller input data sets, and provides greater flexibility results calculations. **Change-Time** data requires records only for the initial start date, and whenever covariate variable values change, as opposed to the standard approach, which requires a record for every successive time interval. When you use **Change-Time** formatting, you can use the **Survival** node property settings to manipulate the discrete time interval at which model results are calculated, as well as left-truncation dates and right-censoring dates.

By contrast, data sets in the **Fully Expanded** format must be altered with SAS DATA step code or SQL queries in order to affect different discrete time intervals, left-truncation dates, and right-censoring dates. The length of the recorded time interval for data in **Fully Expanded** format will also influence modeling results. If you want your modeling results to contain the most detail, you should submit the most granular input data via the **Fully Expanded** data format.

For example, weekly data would typically capture more information than monthly data, monthly data would typically capture more information than quarterly data, and so on. To change an input data set to **Change-Time** or **Fully Expanded**, see the following detailed instructions.

The INTNX() DATA step function and the EM_SURV_SVEXP macro variable are two tools that you can use to prepare or alter **Change-Time** or **Fully Expanded** data sets.

SAS Enterprise Miner includes three example Survival data sets:

SAMPSIO.churn_changetime, SAMPSIO.churn_fullyexpanded_weekly, and SAMPSIO.churn_fullyexpanded_monthly. You can use these three example Survival data sets to experiment with the two data formats using different time intervals.

Beginning with SAS Enterprise Miner 12.3, the EM_SURV_SVEXP macro variable can be used in project start code. It will create the expanded data set that is sent to the **Workspaces** folder beneath the SAS Enterprise Miner project folder directory.

The submitted code

```
%let EM_SURV_SVEXP=1;
```

creates the expanded data set SURV_EXPCENDATA.SAS7BDAT that the Survival node uses in the Workspaces folder.

Note: If you are using **Change-Time** or **Fully Expanded** data formats, and if your process flow diagram contains a Data Partition node that precedes the **Survival** node, your ID variable must be configured in the **Data Partition** node as a cluster variable, with the partitioning method set to **Cluster**.

Fully Expanded Format Data Requirements

Data in the **Fully Expanded** data must meet the following requirements:

- An ID variable is required. Generally, an ID variable represents a unique customer or patient identifier. When possible, ID variables should be configured as nominal variables.
- Two TimeID variables are required, a start time and an end time. An time interval index variable (positive integers only) named `_t_` is also required. Start and end time

variables must share the same date or datetime format. For each TimeID variable in the data set, there must be accompanying rows numbered from 0 to n for the time interval index variable, where n represents the time interval count from the start date to the end or censor date, whichever comes first.

CAUTION:

In general, SAS Enterprise Miner will not process any data set with the role TRAIN that contain variables named with an initial underscore character (_). However, certain data sets with the role SCORE processed by the SAS Enterprise Miner **Survival** node require a variable named `_t_`. If you are not scoring survival data, creating SAS Enterprise Miner variable names with an initial underscore (_) is generally not recommended, because the software reserves the use of variable names with an initial underscore (_) in all training data sets.

- The time index variable must reflect start and end dates in consistent time index variable units. For example, in a model that uses a monthly time interval, if the start date is March 15 and the end date is April 2, the time index variable must have a row for `_t_`=0 that corresponds to March 1, and a row for `_t_`=1 that corresponds to April 1, with the event occurring at `_t_`=1.
- All observations that have the same ID variable must share the same target variable value.
- Right-censored records in **Fully Expanded** data should have the end Time ID variable set to “.”, the value for numerical missing.
- Use caution when switching to **Fully Expanded** data that is less granular than the original, more detailed data source. For example, if a customer churns within a short period of time, valuable data regarding time-varying covariates might be compressed into a single, less informative record.
- When using the **Fully Expanded** data format, the time interval, left-truncation date and training time ranges cannot simply be changed by changing Survival node properties. The input data itself must be manipulated to make any changes to the time frame and time interval that the model will use as a basis for calculations and results.

Time Intervals and Fully Expanded Data Format

Choosing the right time interval for your time-varying covariate survival analysis is profoundly important. Let us look at some fully expanded example churn data sets, recorded at different time intervals, and examine the sample data to see how survival model results might be impacted.

The following example fully expanded data charts come from two data sets that are included in your SAS Enterprise Miner sample data library, SAMPSIO. One data set contains summarized weekly data, `SAMPSIO.churn_fullyexpanded_weekly`, and the other data set contains summarized monthly data, `SAMPSIO.churn_fullyexpanded_monthly`. You can locate the expanded-data format tables by browsing the SAMPSIO library using the SAS Enterprise Miner Data Source Wizard.

The weekly data contains more detailed figures for the time varying covariate `num_complaints`. The monthly data contains the same basic trends, but at a less granular level. However, the monthly data will also require less disk and memory resources to process, a potential advantage for modelers that are using extremely large data sets.

Figure 9.1 View of Expanded Weekly Data Table *SAMPSIO.churn_fullyexpanded_weekly*

	customer_id	_t_	promotions	num_complaints	churn	start	end
1	1	0	1	0	1	20MAY1988	10JUL1988
2	1	1	1	5	1	20MAY1988	10JUL1988
3	1	2	1	6	1	20MAY1988	10JUL1988
4	1	3	1	8	1	20MAY1988	10JUL1988
5	1	4	1	10	1	20MAY1988	10JUL1988
6	1	5	1	10	1	20MAY1988	10JUL1988
7	1	6	1	10	1	20MAY1988	10JUL1988
8	1	7	1	10	1	20MAY1988	10JUL1988
9	1	8	1	10	1	20MAY1988	10JUL1988
10	2	0	1	0	1	10NOV1987	21FEB1988
11	2	1	1	0	1	10NOV1987	21FEB1988
12	2	2	1	0	1	10NOV1987	21FEB1988
13	2	3	1	0	1	10NOV1987	21FEB1988
14	2	4	1	0	1	10NOV1987	21FEB1988
15	2	5	1	0	1	10NOV1987	21FEB1988
16	2	6	1	0	1	10NOV1987	21FEB1988
17	2	7	1	0	1	10NOV1987	21FEB1988
18	2	8	1	1	1	10NOV1987	21FEB1988
19	2	9	1	1	1	10NOV1987	21FEB1988
20	2	10	1	1	1	10NOV1987	21FEB1988
21	2	11	1	1	1	10NOV1987	21FEB1988
22	2	12	1	1	1	10NOV1987	21FEB1988
23	2	13	1	1	1	10NOV1987	21FEB1988
24	2	14	1	1	1	10NOV1987	21FEB1988
25	2	15	1	1	1	10NOV1987	21FEB1988
26	3	0	1	0	0	27JUL1987	.
27	3	1	1	1	0	27JUL1987	.

Figure 9.2 View of Expanded Monthly Data Table *SAMPSIO.churn_fullyexpanded_monthly*

	customer_id	_t_	promotions	num_complaints	churn	start	end
1	1	0	1	0	1	20MAY1988	10JUL1988
2	1	1	1	5	1	20MAY1988	10JUL1988
3	1	2	1	10	1	20MAY1988	10JUL1988
4	2	0	1	0	1	10NOV1987	21FEB1988
5	2	1	1	0	1	10NOV1987	21FEB1988
6	2	2	1	0	1	10NOV1987	21FEB1988
7	2	3	1	1	1	10NOV1987	21FEB1988
8	3	0	1	0	0	27JUL1987	.
9	3	1	1	0	0	27JUL1987	.

Less granular data might be advantageous from a processing resources perspective, but using less granular data can result in data loss that impacts the quality of the survival model outcomes. These situations can be hard to find and prevent without taking a look at the how different fully expanded data sets are created.

An example of such information loss can be found by browsing the data for Customer ID 27 in both of the weekly and monthly fully expanded sample data sets. By examining the weekly expanded data, we can see that Customer ID 27 opened an account mid-month, logged a net total of 48 complaints over the following two weeks, and then churned by month's end.

Figure 9.3 View of CustID 27 in Expanded Weekly Data

	customer_id	_t_	promotions	num_complaints	churn	start	end
458	26	10	2.5	6	1	03OCT1987	06DEC1987
459	27	0	2.5	3	1	14JUL1988	31JUL1988
460	27	1	2.5	15	1	14JUL1988	31JUL1988
461	27	2	2.5	15	1	14JUL1988	31JUL1988
462	27	3	2.5	15	1	14JUL1988	31JUL1988
463	28	0	2.5	2	1	17DEC1987	14FEB1988

Now, let us examine CustID 27 in the monthly data. CustID 27 is represented in the figure below by a single record. The time varying covariate num_complaints for CustID27 reads 0 (as opposed to a total of 48 complaints in the weekly data). This is because the monthly data table records the existing number of complaints on the first day of the month, the time at which monthly data is summarized. Discrepancies such as

these can negatively impact parameter estimates in the logistic survival model. You need to know the point in the time interval when your data are summarized and recorded. In this case, the monthly expanded data would reveal a more accurate summary if the data were summarized and recorded at the end of the month. Another option to avoid data compression errors like these is to use change-time data. For more information about change-time data, see [“Change Time Format Data Requirements” on page 149](#).

It is not hard to see that the larger discrete time scale for monthly data, as well as discrepancies between the time-varying covariates across the weekly- and monthly summarized data is very likely to produce two significantly different survival models.

In order to see how time intervals within expanded data affects the models, we can assign the following data roles to the models:

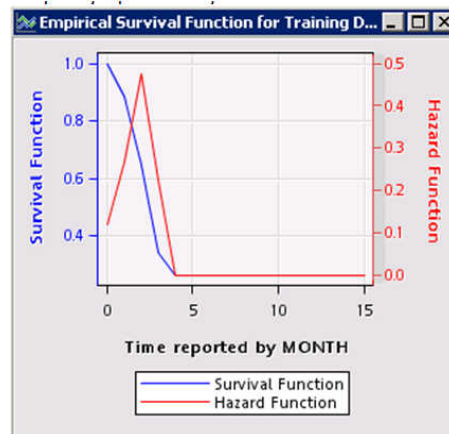
Figure 9.4 Variable Roles for Example Expanded Data Models

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
t_	Input	Interval	No		No	*	*
churn	Target	Binary	No		No	*	*
customer_id	ID	Nominal	No		No	*	*
end	Time ID	Interval	No		No	*	*
num_complaints	Input	Interval	No		No	*	*
promotions	Input	Nominal	No		No	*	*
start	Time ID	Interval	No		No	*	*

Using the variable roles shown above, Survival models were built and run using the expanded weekly and monthly data. Model results were generated. Plots for Empirical Survival Function and Hazard Function and regression coefficient statistics are provided for both models below.

Figure 9.5 Empirical Survival Function Using Expanded Weekly Data



Figure 9.6 Empirical Survival Function Using Expanded Monthly Data

Notice the increased detail in the Empirical Hazard Function plot that was constructed from the weekly data.

Now, let us examine the regression coefficients for the weekly expanded data and the monthly expanded data models.

Figure 9.7 Regression Coefficients for Survival Model Using Weekly Expanded Data

Parameter:	Estimate:
Intercept	-4.9412
t	-90.0439
_csb1	-0.1847
_csb2	0.6972
_csb3	-1.0924
_csb4	0.8209
_csb5	-0.2434
promotions(1)	-1.9007
promotions(2.5)	-0.3250
promotions(10)	0
num_complaints	0.1471

Figure 9.8 Regression Coefficients for Survival Model Using Monthly Expanded Data

Parameter:	Estimate:
Intercept	-1.3461
t	5.6155
_csb1	2.6030
_csb2	-2.4526
_csb3	1.9465
_csb4	-1.0729

_csb5	0.2920
promotions(1)	-2.5410
promotions(2.5)	-0.3056
promotions(10)	0
num_complaints	0.3013

Note the dissimilar values for regression coefficients between the expanded data weekly and expanded data monthly models. In the case of this particular example, one could make a strong argument that the model that was created from the weekly fully expanded data is more accurate.

Change Time Format Data Requirements

Change-time data format can be used to avoid data compression errors as illustrated above. You can set the **Survival** node Time Interval property to calculate survival models at several discrete time scales, while concurrently specifying properties for left-truncation and right-censoring dates. To do so requires data manipulation with the fully expanded format.

The SAS Enterprise Miner example SAMPSIO library provides a sample data set called SAMPSIO.churn_changetime. You can use the SAMPSIO.churn_changetime data set to visualize the extra functionality that the change-time data form provides. Using the fully expanded SAMPSIO.churn_changetime data set, you can generate weekly and monthly survival models by simply changing the Time Interval property. Empirical results would be expected to be exactly similar to those built from the expanded data. Regression parameters would be expected to differ only slightly, as the node uses a weighting scheme to expand change-time data before the modeling procedures are invoked.







Variable roles must be appropriately specified for change-time data. The following figure provides suggested variable role settings.

Figure 9.9 Variable Roles for Example Change-Time Data Survival Model

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
change_time	Time ID	Interval	No		No	-	-
churn	Target	Binary	No		No	-	-
customer_id	ID	Nominal	No		No	-	-
end	Time ID	Interval	No		No	-	-
num_complaints	Input	Interval	No		No	-	-
promotions	Input	Nominal	No		No	-	-
start	Time ID	Interval	No		No	-	-

Time interval, truncation settings, and training time range must also be specified for time-change data modeling. You use these property settings specify the time frame and time interval that the model will use without having to manipulate the input data.


Figure 9.10 Properties to Specify Time Frame and Time Interval for Change-Time Data Modelers

General	
Node ID	SURV
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Data Format	Standard
Time ID Variables	
Time Interval	Week
Left-Truncated Data	Yes
Training Time Range	


Survival Node Properties

Survival Node General Properties

The following general properties are associated with the **Survival** node:


- **Node ID** — The Node ID property displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Survival node that is added to a diagram will have a Node ID of Survival. The second Survival node added to a diagram will have a Node ID of Survival2, and so on.
- **Imported Data** — The Imported Data property provides access to the Imported Data — Survival window. The Imported Data — Survival window contains a list of the ports that provide data sources to the **Survival** node. Select the  button to the right of the Imported Data property to open a table of the imported data.

If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Exported Data** — The Exported Data property provides access to the Exported Data — Survival window. The Exported Data — Survival window contains a list of the output data ports that the **Survival** node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.


If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.

- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.

Survival Node Train Properties

The following train properties are associated with the **Survival** node:

- **Variables** — Use the Variables window to view variable information, and change variable values using the **Survival** node. Select the  button to open a window that contains the variables table. You can specify *Use* and *Report* values for a variable. You can view the columns metadata. You can also open an Explore window to view a variable's sampling information, observation values, or a variable distribution plot. By default, columns for the variables **Name**, **Use**, **Report**, **Role**, and **Level** are displayed.


To modify these values, and add additional columns, you can use the following options:

- **Apply** — Changes metadata based on the values that are supplied in the drop-down menus, check box, and selector field.
- **Reset** — Changes metadata back to its state before use of the **Apply** button.
- **Label** — Adds a column for a label for each variable.
- **Mining** — Adds columns for the Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
- **Basic** — Adds columns for the Type, Format, Informat, and Length of each variable.
- **Statistics** — Adds statistics metadata for each variable.
- **Explore** — Opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Data Format** — Specifies the data format that is used. Use **Standard** when you do not need to accommodate time-dependant covariates. Use **Change Time** and **Fully Expanded** to enable multiple observations per time ID, which accommodates time-dependent covariates. **Fully Expanded** requires a `_t_` variable with a row for each time interval. **Change Time** requires a row only for each time interval where a covariate changes value. An additional time ID variable contains the time at which this happens.

Note: If you specify **Standard** or **Change-Time** as the setting for your **Data Format** property, if your data set contains a variable named `_t_` with role=Input, the `_t_` variable will not be used.

If you specify either the **Change Time** or **Fully Expanded** data format, the exported data contains only columns from the score code for the last observation for each time ID. That is, every observation except the last observation contains missing values for any variables added to the import data set.

Note: If you want to partition your data and use the **Change Time** or **Fully Expanded** data format, you must specify certain settings in the **Data Partition** node. The ID variable must be used as a cluster variable and the **Partitioning Method** must be set to **Cluster**.

- **Time ID Variables** — The time ID variables identify the variables that will indicate the start and end times for the discrete time span until the event of interest. You assign start and end time variables via the mapping table that appears when you select the ellipsis button  to the right of the Time ID Variables property in the **Survival** node properties panel. The **Survival** node scans the TIMEID column's meta-variable values to determine maximum and minimum TIMEID values in the table. The variable that has the earliest time unit is mapped to **Start Time Variable**, and the variable that has the latest time unit is mapped to **End Time Variable**.

If you specify **Change Time** as the **Data Format**, then the **Change Time Variable** selection becomes available.

- **Time Interval** — The time interval variable specifies the unit of time to be used for measurement and prediction. The results of the analysis are expressed in terms of the chosen time scale. They should represent the smallest units that are measurable and actionable. Typically, the values represent days, weeks, months, billing cycles, quarters, or years.

The smallest time interval found in most databases is a day. Many customers initiate, cancel, or change their products and services on the same day. After you specify the time interval in the **Survival** node properties panel, SAS Enterprise Miner performs error checking to ensure that your start time date format supports the corresponding time interval.

- **Left-Truncated Data** — Specifies whether the **Training Time Range** property enables you to specify a truncation date. All data that is observed before the truncation date is omitted from analysis.
- **Training Time Range** — Opens the Train Date Selection window that enables you specify the **Left-Truncation** date and the **Right-Censoring** date. Data outside of this range is omitted from analysis.

Survival Node Train Properties: Sampling Options

- **Sampling** — Specifies whether sampling of the expanded data should occur. If it should, simple random sampling of the non-events only is performed. All events remain in the table to be modeled.
- **Event Proportion** — Specifies the event proportion in the sample. Event proportion values are real numbers ranging between 0.0 and 0.9. The default setting is 0.2.
- **Seed** — Specifies the seed number to be used during sample selection. The default sampling seed value is 12345.

Survival Node Train Properties: Regression Spline Model

- **Covariate x Time Interactions** — Specifies the variables that interact with the **_t_** variable in the regression model. Select **Do not include** to suppress all interactions, **Include All** to enable interactions with all variables, and **Include Selected** to manually specify the interactions. You can manually specify the variables with the **New Covariates for Interactions** property.
- **New Covariates for Interactions** — Opens a window that enables you to specify that variables that interact with the **_t_** variable in the regression model.
- **Stepwise Regression** — Specifies whether stepwise regression should be used for variable selection during the modeling stage.
- **Entry Significance Level** — Specifies the significance level for adding variables in stepwise regression.

- **Stay Significance Level** — Specifies the significance level for removing variables in a stepwise regression.
- **Number of Knots** — Specifies the number of knots used in the cubic spline basis function.
- **Knot Selection** — Specifies whether to use automatic knot selection for the cubic spline basis functions when stepwise selection is being performed. When set to **Yes**, the cubic spline functions that are created are entered into the regression model and are part of the stepwise variable selection procedure. This enables statistical determination of whether a cubic spline basis function is significant or not. When **Knot Selection** is set to **Yes**, the **Number of Knots** property specifies the number of knots to be created for consideration as part of variable selection.


Survival Node Train Properties: Survival Validation

The Survival Validation section of the **Survival** node properties panel provides the following settings for configuring the validation of your Survival model:

- **Survival Validation Method** — Use the **Survival Validation Method** property to specify how the validation holdout sample is generated. By default, subsets are created from each of the training, validation, and test data sets that are passed to the **Survival** node, and those subsets are used for model validation. When the default survival validation method is selected, the last 25% of the total time interval is set aside for survival model validation and scoring. The scoring interval is used for validation calculations.

If you desire, you can choose to create a user-specified scoring interval. When you select the user-specified validation method, you can specify a specific hypothetical scoring date to define the beginning of the scoring interval as well as the scoring interval length.

All Survival model validation is performed with the model that was built using the entire Train data set.

- **Validation Score Date** — When you select **User-Specified** as the value for your **Survival Validation Method** setting, you can use the **Validation Score Date** property to specify a hypothetical scoring date. The hypothetical scoring date defines the beginning of the scoring interval that is used to subset the data for model evaluation. Select the  button to the right of the **Validation Score Date** property to open a table in which you specify the date value for scoring. The **Validation Score Date** date value that you specify must fall between your defined start date and censor date.
- **Interval Length** — When you select *User Specified* as the value for your **Survival Validation Method**, the **Interval Length** property specifies the length of the time interval that is used to perform survival validation. The interval will begin with the hypothetical scoring date that you specified in the **Validation Score Date** property setting, and then extend the specified number of time intervals forward. If the **Time Unit** property is set to *Day* and the **Interval Length** property is set to *15*, your scoring interval is the 15 days that follow the hypothetical scoring date.

Survival Node Score Properties

The Score section of the **Survival** node properties panel provides the following settings for configuring your Survival model scoring:

- **Mean Residual Life (MRL)** — This property specifies whether Mean Residual Life (MRL) should be calculated. MRL represents the remaining time until the event occurs. MRL can be processor-intensive to calculate, so the default setting for this property is **NONE**, which suppresses the MRL calculation. If you select the **Constant**

Hazard Extrapolation setting, you assume that from time t onward, the hazard function is constant from the final value. If you select the Restricted Mean Residual Life setting, the hazard function continues trending until an event occurs, or until the maximum value for MRL is reached, whichever comes first. After the maximum value for MRL is reached, the hazard is held constant from that point forward.

- **Default Maximum (MRL)** — This property is used in conjunction with Restricted Mean Residual Life. It specifies whether default values that are based on the specified Time Interval will be used to set the maximum MRL used in calculations. When **Default Maximum MRL** is set to Yes, the following default values will be used based on Time Interval: Day=108, Week=108, Month=60, Quarter=40, Semi-Year=50, and Year=50.
- **User-Specified Maximum MRL** — If the **Default Maximum MRL** property is set to No, the **User-Specified Maximum MRL** property enables you to specify the maximum MRL value. The value that you specify is used to calculate Restricted Mean Residual Life.
- **Default Forecast Intervals** — The **Default Forecast Intervals** property specifies the number of time units into the future that you want to use for score code generation, survival functions, and so on. The **Default Forecast Intervals** property indicates whether default values based on the Time Unit should be selected, or whether user-specified values will be used. If **Default Forecast Intervals** is set to Yes, the following time unit values will be used: Day=30, Week=4, Month=3, Quarter=4, Semi-Year=2, and Year=1.
- **Number of Forecasted Intervals** — If the **Default Forecast Intervals** property is set to No, the **Number of Forecasted Intervals** property specifies the number of time intervals into the future to use as the basis for your survival calculations.

Survival Node Report Properties

The Report section of the **Survival** node properties panel provides the following settings for configuring your Survival model reports:

- **Risk Account Selection** — This reporting property specifies whether the table should be generated based on percentile or count.
- **High Risk Account Tables** — Specifies whether High Risk tables should be generated and, if so, which tables to create. If **None** is selected, no additional tables are created. If **Event Probability** is selected, those IDs with the highest event occurrence probability within the forecast time period given survival until the current time are identified and displayed in the results. If **Survival Probability** is selected, those IDs with the lowest survival probability at the forecasted time are identified and displayed in the results. If **Hazard Rate** is selected, those IDs with the highest overall hazard rate at the forecasted time are identified and displayed in the results. If **All** is selected, all three tables are generated and displayed.
- **Percentile** — The **Percentile** reporting property specifies the corresponding percentage value to be used in creating the table.
- **Count** — The **Count** reporting property specifies the corresponding count value to be used in creating the table.

Survival Node Status Properties

The following status properties are associated with the **Survival** node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.

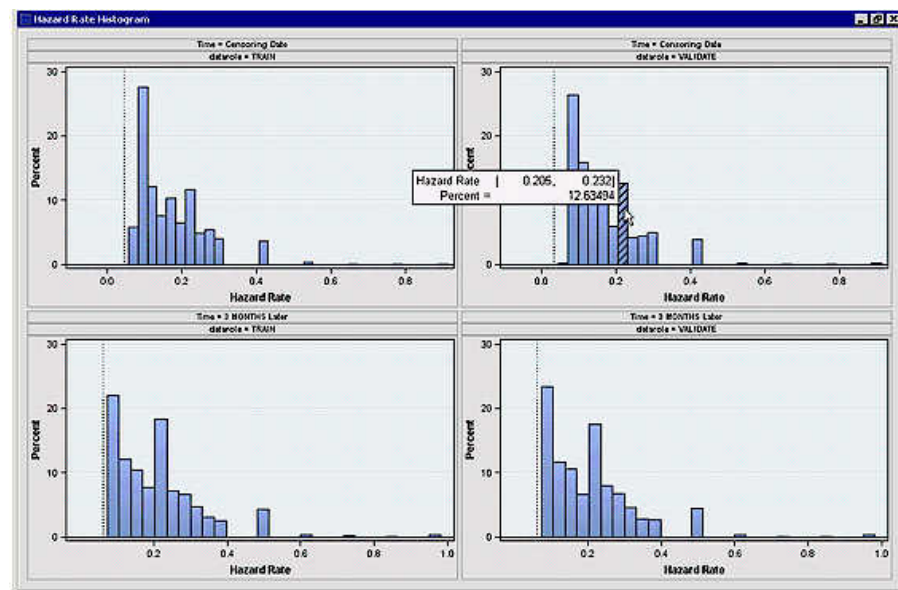
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.
- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Survival Node Results

The **Survival** node Results browser contains the following output plots and data tables:

- **Hazard Rate Histogram**
- **Event Occurrence Probability Histogram**
- **Survival Probability Histogram**
- **Empirical Subhazard Function for Training Data**
- **Empirical Survival Function for Training Data**
- **Model Validation Plot**
- **Model Validation Statistics**
- **SAS Output**

Hazard Rate Histogram (Time=Censoring Date and Time=3 Time Units Later):

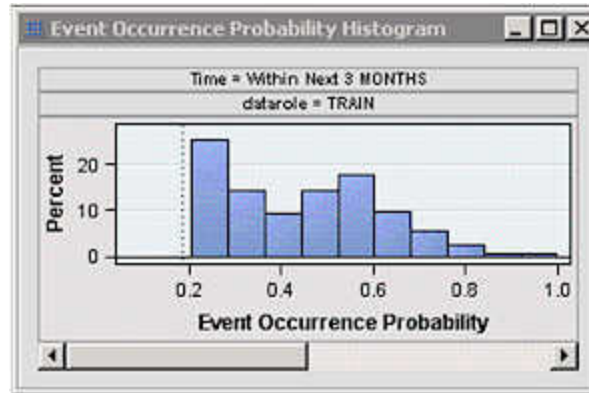


The Hazard Rate histogram displays the plots of the hazard rate distribution for the time interval (for example, month) that contains the censoring date. If f is the number of forecast intervals specified, then the hazard rate is also plotted for the f th interval that follows the censor date. For example, for an analysis that uses month as the time interval, and that uses the default setting of three forecast intervals ($f = 3$), the Hazard Rate histogram displays the hazard rate through the third month that follows the censor

date. Each bar in the histogram represents the percentage of hazard rates that are between the beginning and ending bins (where the bins are the hazard rates).

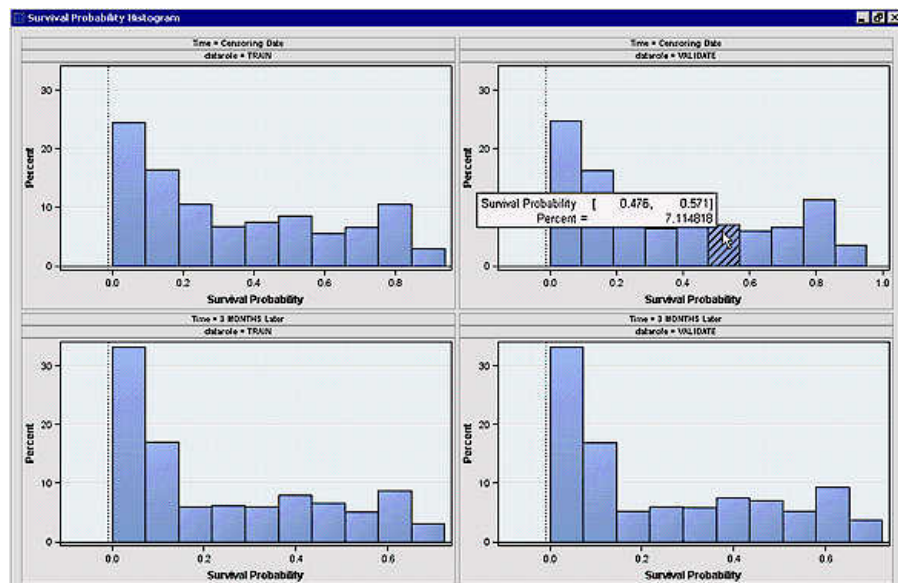
The hazard rate tells you the chance that someone is going to cancel or stop service. You can use the histogram that extrapolates three time units into the future to identify customers that have high hazard probabilities. You might intervene with high hazard probability customers by offering them some type of promotion.

Event Occurrence Probability Histogram (within next 3 time units):



The Event Occurrence Probability histogram displays the distribution of the probabilities of having an event of interest occur within the next f forecast time intervals, where f is the number of forecast intervals that the **Survival** node is configured to analyze (default setting is $f = 3$). You specify the time units for each forecast interval in the **Survival** node properties panel when you configure the node. Each bar in the plot represents the percentage of customers who will experience an event of interest during the next f forecast intervals.

Survival Probability Histogram (Time=Censoring Date and Time=3 Time Units Later):

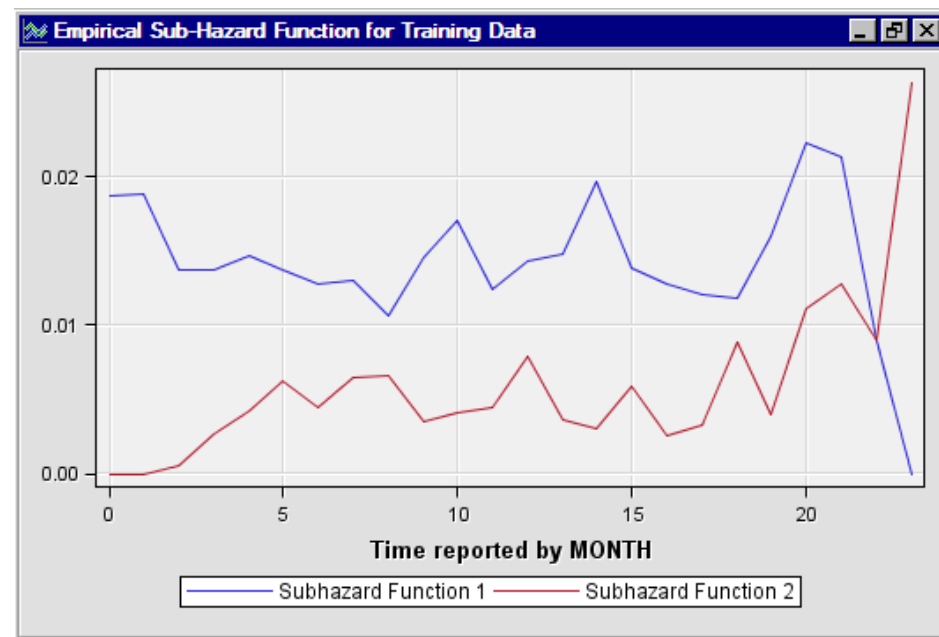


The Survival Probability Histogram displays the distribution of the survival probabilities for the time interval (for example, month) that contains the censoring date. If f is the number of forecast intervals that are specified, then the Survival Probability histogram is

also plotted for the f_{th} interval that follows the censor date. For example, for an analysis that uses month as the time interval, and that uses the default setting of three forecast intervals ($f = 3$), the Survival Probability histogram displays the survival probability through the third month that follows the censor date. Each bar in the histogram represents the survival probabilities that are between the beginning and ending bins (where the bins are the hazard rates)

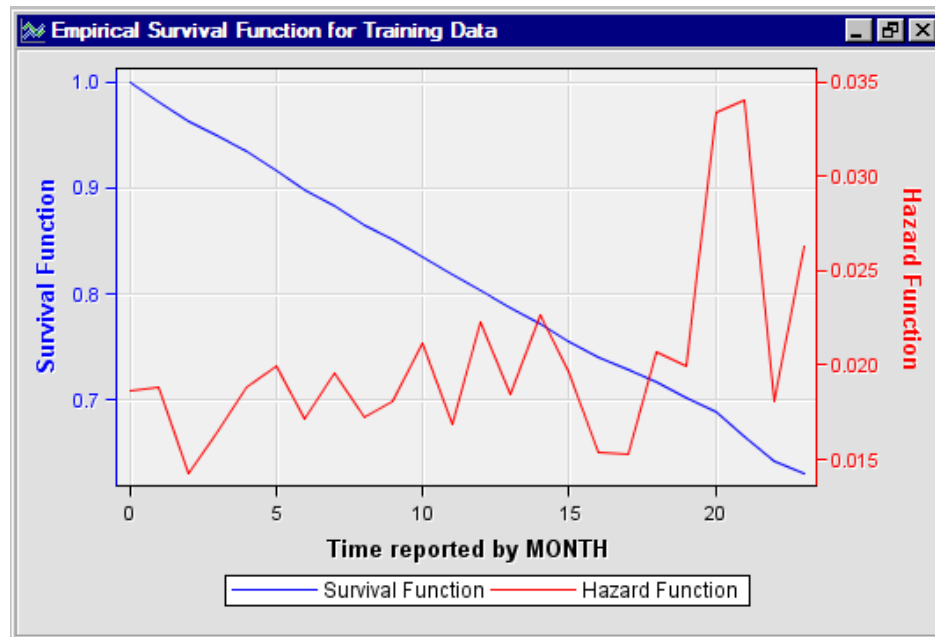
You could use the histogram to determine which customers would be suitable for a loyalty promotion by targeting customers that have high survival probabilities. The Survival Probability histogram for three time units later displays the probabilities that a customer account will remain active during the three-month interval that follows the censoring date.

Empirical Subhazard Function for Training Data:



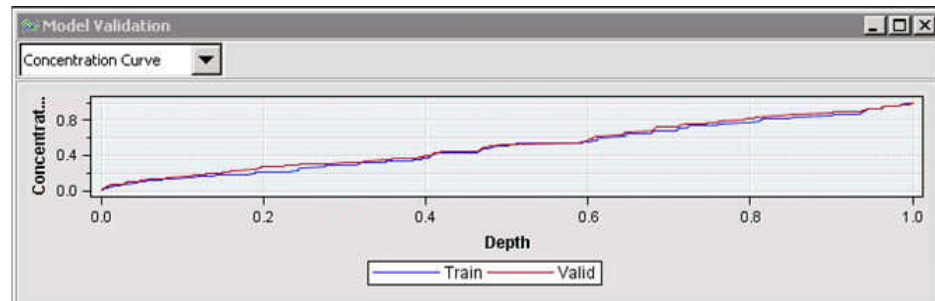
For each one of the competing risks, the subhazard function is computed and displayed in this graph. The vertical axis is the values of the subhazard functions, and the horizontal axis is time.

Empirical Survival Function for Training Data:



This plot overlays the Hazard function and the Survival function across the entire time continuum. The left vertical axis represents the Survival function, and the right vertical axis represents the Hazard function. The horizontal axis is the time unit.

Model Validation Plot:



The three Model Validation Plots are as follows:

- **Concentration Curve:**

In a concentration curve, the concentration is plotted on the vertical axis. The horizontal axis is the depth (deciles) of the data. For each decile, the concentration is computed. The concentration is the ratio of the number of events in the decile to the total number of events. The concentration curve is similar to a Cumulative Captured Response curve from the **Model Comparison** node.

- **Lift:**

The lift chart is a data mining measure that is used to choose among competing models. Assume a model that predicts the probability of some event, such that a higher calculated probability value implies a higher likelihood of an event.

Consider submitting a data set to the model to score the data, and then rank the scored data by descending posterior probability values. In a model with good discriminatory performance, the cases that are predicted to be events should be at the top of the sorted list. The cases that are predicted to be non-events should be at the bottom of the list.

- **Benefit:**

The Benefit is plotted on the vertical axis. The horizontal axis is the depth (deciles) of the data. The Benefit statistics represent the difference between the Concentration Curve and the random model (represented by a 45-degree diagonal line). The largest Benefit value indicates the depth at which the model is doing the best job at predicting the outcome, using the Average Hazard Function as the score. The Benefit curve can be used to establish an appropriate cutoff value.

Model Validation Statistics:

The Model Validation Statistics table displays the following statistics:

Benefit

the maximum benefit value

Average Hazard Function

the average hazard function at the maximum benefit value

Depth

the depth at the maximum benefit value

Lift

the lift at the maximum benefit value

Kolmogorov-Smirnov statistic

the maximum distance between the event and non-event distributions

Gini Concentration Ratio

twice the area between the concentration curve and the random model (represented by a 45-degree diagonal line). The Gini concentration ratio statistic is a measure of the separation between the probabilities of event and non-event..

SAS Output

The output window in the Survival Results browser contains the SAS output from running the LIFETEST and DMREG procedures.

Survival Node Example

Overview

The **Survival** node usage example proceeds sequentially through the following steps:

1. [“Create and Configure Survival Data Source” on page 159](#)
2. [“Place Survival Analysis Nodes in the Diagram Workspace” on page 161](#)
3. [“Configure the Survival Node Settings” on page 161](#)
4. [“Run the Survival Node” on page 161](#)
5. [“Examine Survival Node Results” on page 162](#)
6. [“Submit a Data Set for Scoring” on page 167](#)

Create and Configure Survival Data Source

This example uses the SAS sample data set SAMPSIO.CELLPHONE. The SAMPSIO.CELLPHONE data set is found in the SAMPSIO example data library that is included with your copy of SAS Enterprise Miner.

Note: You also use the SAMPSIO.CELLPHONE_SCORE data set from the SAMPSIO library in the last step of this example to use your trained model to score data.

Use the SAS Enterprise Miner Data Source Wizard to create a SAS Enterprise Miner Data Source from the example SAMPPIO.CELLPHONE data set. In the SAS Enterprise Miner Project Panel, right-click the **Data Sources** folder and select **Create Data Source** to launch the Data Source Wizard.

In the Data Source Wizard, do the following to create your SAS Enterprise Miner Data Source:

1. Choose **SAS Table** as your metadata source and click **Next**.
2. Enter SAMPPIO.CELLPHONE in the **Table** field and click **Next**.
3. Continue to the Metadata Advisor and choose the **Advanced Metadata Advisor**.
4. In the Column Metadata window, make the following variable role assignments:
 - Set the role of the variable ACCOUNT_NUM to **ID**.
 - Set the roles of the variables ACTIVATION_DATE and DEACTIVATION_DATE to **TimeID**.
 - Set the role of the variable DISABLE to **Rejected**.
 - Set the role of the variable TARGET to **Target**.
 - All of the remaining variables (GOOD_BAD, PLAN_TYPE, and PROVIDER_TYPE), should be set to the role of **Input**.
 - Click **Next**.
5. Accept the default settings for the remainder of the Data Source Wizard windows. Select **Next** for each window, and when you reach the last Data Source Wizard window, select **Finish**.
6. The SAMPPIO.CELLPHONE data set should appear in your Data Sources folder in the SAS Enterprise Miner Project Panel.

The SAMPPIO.CELLPHONE data source that you just created should have 10,185 observations. Each observation is a unique customer observation from a cell phone provider company. The data source that you created contains the cell phone data as follows:

Ob...	Target	disable	account_num	good_bad	plan_type	activation_date	deactivation_date	provider_type
1	0		180437090184	1	3	08/28/1999		PROV1
2	0		1804372283474	1	1	01/09/2001		PROV1
3	0		180437340410	0	1	12/31/1999		PROV1
4	2DUE		180437356568	0	1	12/22/1999	08/28/2000	PROV2
5	0		180437358937	1	1	04/17/2000		PROV3
6	1TRANSFER		180437375280	1	2	08/16/1999	08/21/2000	PROV1
7	0		180437392809	1	1	07/26/1999		PROV3
8	0		180437420857	0	1	12/15/1999		PROV2
9	0		180437433673	0	3	11/21/2000		PROV1
10	0		180437452331	0	2	12/28/2000		PROV3
11	0		180437468686	1	3	07/15/2000		PROV3
12	0		180437482423	1	1	11/20/2000		PROV1
13	0		180437494598	0	2	08/28/2000		PROV1
14	0		180437498978	0	2	08/18/2000		PROV1
15	0		180437499481	1	1	07/03/1999		PROV2
16	0		180437502892	1	3	03/22/2000		PROV1
17	0		180437507436	1	1	07/02/1999		PROV1
18	1PAY		180437512268	0	1	08/29/1999	07/13/2000	PROV2
19	1PAY		180437514956	1	1	12/04/1999	06/09/2000	PROV1
20	1PAY		180437519787	1	1	08/17/1999	03/08/2000	PROV1
21	0		180437535931	1	1	09/04/2000		PROV1
22	0		180437544749	1	2	12/27/2000		PROV2
23	0		180437547814	1	3	12/27/2000		PROV1
24	0		180437551602	1	1	10/14/2000		PROV1
25	0		180437558314	1	1	08/11/1999		PROV2
26	0		180437559000	0	2	08/26/1999		PROV1
27	0		180437566244	1	2	12/14/2000		PROV1
28	0		180437576804	1	3	11/16/1999		PROV1
29	0		180437576885	0	1	02/12/2000		PROV1
30	0		180437577676	1	2	01/26/1999		PROV1
31	0		180437584659	1	2	12/04/1999		PROV1
32	0		180437587313	0	3	11/01/2000		PROV4
33	0		180437589753	1	1	08/08/2000		PROV1
34	0		180437591341	1	1	03/26/1999		PROV4
35	0		180437593825	1	1	01/20/2001		PROV4
36	0		180437593522	1	3	04/01/1999		PROV1
37	0		180437599247	1	1	04/17/1999		PROV2
38	1ADDITIONAL		180437603277	1	2	08/18/1999	09/02/1999	PROV4

In this data set, TARGET is the target variable that is modeled. TARGET indicates what type of risk is being modeled. There are three risks that are being modeled:

- If TARGET=0, the customer still has an active account
- If TARGET=1, the customer has voluntarily ended the relationship with cell phone company
- If TARGET=2, the customer has involuntarily ended the relationship with the cell phone company. For example, a customer might be terminated for delinquent payments for services.

The TARGET variable value is computed as follows:

If TARGET=(DEACTIVATION_DATE ne .) and DISABLE=DUE, then the DEACTIVATION_DATE variable contains a date value, and the class variable DISABLE is set to DUE. On the date in DEACTIVATION_DATE, the customer relationship was ended. The class value for the DISABLE variable provides the reason that the customer relationship ended.

If the customer has a DEACTIVATION_DATE value and the customer was deactivated because account payments were past due, that would be an involuntary churn event (TARGET=2).

If a customer has a DEACTIVATION_DATE value and the DISABLE variable is anything other than DUE, then the customer relationship was ended by the customer, resulting in voluntary churn (TARGET=1).

If the DEACTIVATION_DATE variable is a missing value, and the DISABLE variable is a missing value, then the customer is still active (TARGET=0).

The data set also contains other explanatory variables, such as a good or bad credit indicator (GOOD_BAD), the customer's type of rate plan (PLAN_TYPE) and the customer's provider plan (PROVIDER_TYPE).

Place Survival Analysis Nodes in the Diagram Workspace

Drag the CELLPHONE data source that you just created from the Data Sources folder in the Project Panel onto a blank Diagram Workspace.

Next, drag a **Data Partition** node from **Sample** tab of the Nodes Toolbar onto the Diagram Workspace. Connect the CELLPHONE data source to the **Data Partition** node. Select the **Data Partition** node. Set the **Training** property to **70 . 0** the **Validation** property to **30 . 0**, and the **Test** property to **0 . 0**.

Drag a **Survival** node from the **Applications** tab of the Nodes Toolbar onto the Workspace and connect the **Data Partition** node to the **Survival** node.



Configure the Survival Node Settings

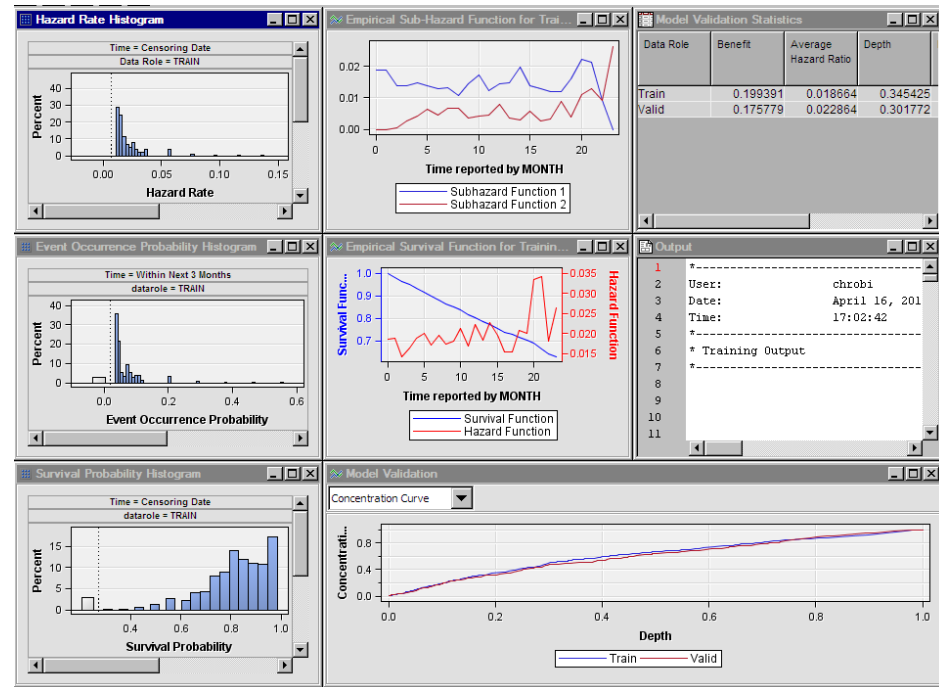
Select the Survival node in the Diagram Workspace, and then use the Properties Panel to set the **Survival** node **Stepwise Regression** property to **Yes**.

Run the Survival Node

Right-click the Survival node in the Diagram Workspace and select **Run**. After the Process Flow Diagram finishes running, click **Yes** to open the Results browser.

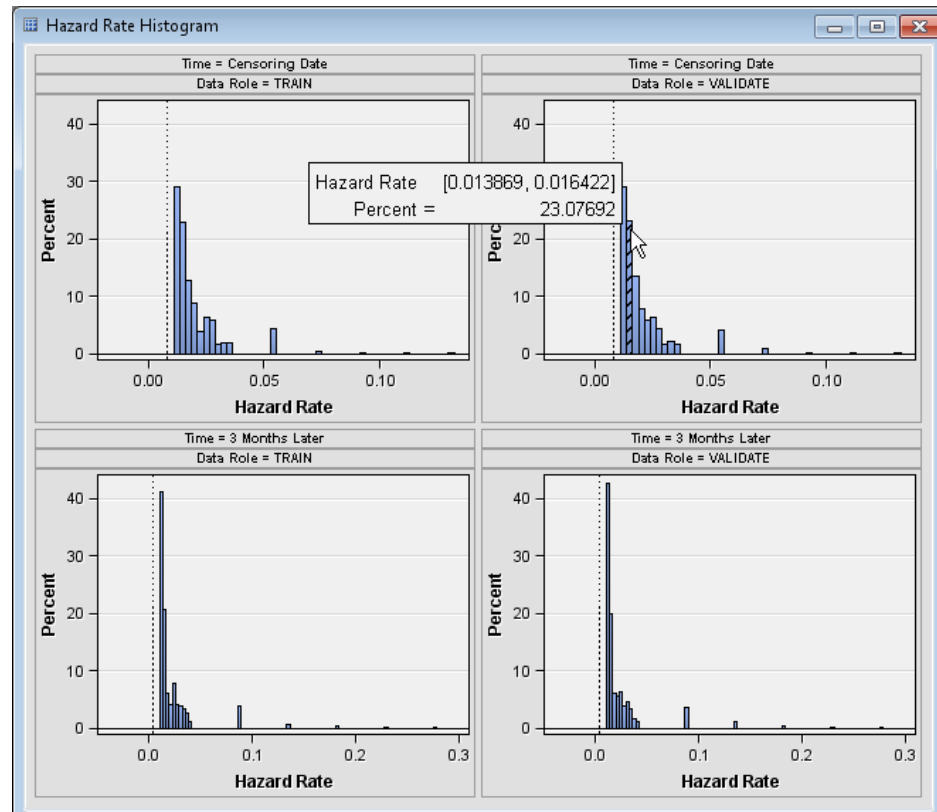
Examine Survival Node Results

The **Survival** node Results window shows a number of plots and tables. They include histograms of the hazard rates, event occurrence probabilities, and survival probabilities. There are overlay plots of the empirical sub-hazards and the survival and hazard functions. There is also a line plot of the model validation statistics, a table of model validation statistics, as well as the SAS output.



Hazard Rate histogram:

The Hazard Rate Histogram displays the distribution of the hazard rate for time unit that contains the censoring date, as well as the third time interval that follows the censoring date.

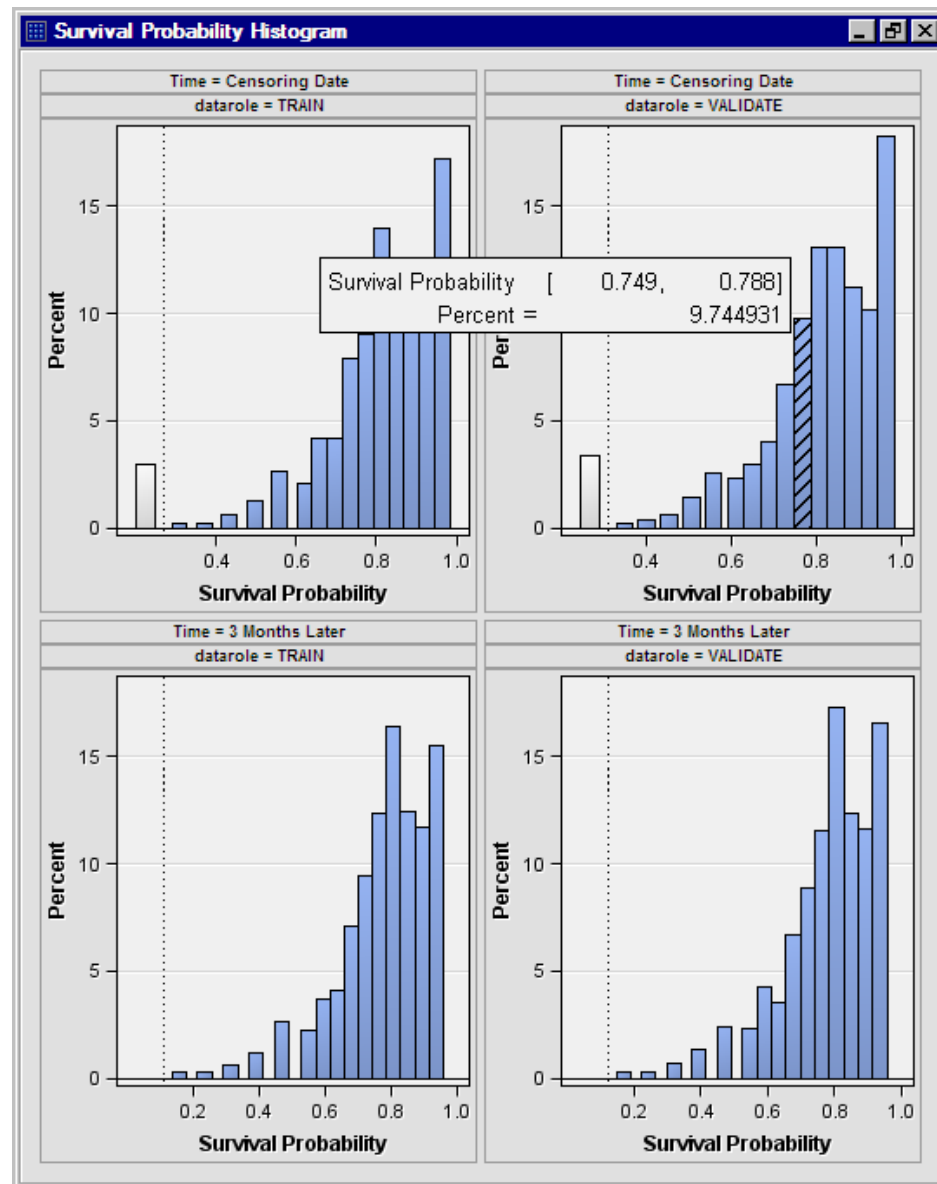


For example, the highlighted bar tells you that approximately 23% of the customers in the validation data set on the censoring date will have a hazard probability between 0.0139 and 0.0164. The hazard probability indicates the chance that someone is going to cancel or stop service.

You can use the histogram that is extrapolated for three time units into the future to identify customers that have high hazard probabilities. You might want to consider intervening with high hazard customers by offering them some sort of promotion or retention incentive. To identify these customers, specify **Hazard Rate** for the **High Risk Account Tables** property in the **Report** properties group.

Survival Probability Histogram:

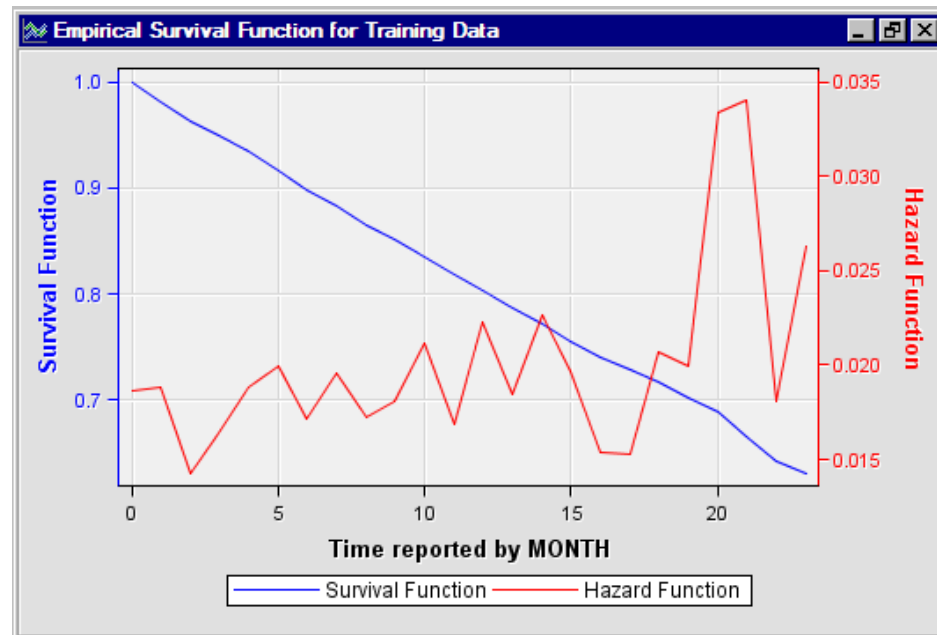
The Survival Probability histogram displays the distribution of the survival probabilities for the censoring time and for three time units into the future. For example, the highlighted bar tells you that 9.7% of the customers in the validation data set will have a survival probability between 0.749 and 0.788.



You can use the Survival Probability histogram for 3 Months later to determine which customers might be suitable for a loyalty promotion. The Survival Probability histogram for three months later displays the probabilities that a customer will still be a customer three months after the censor date. You can use this information to identify the customers that have high survival probabilities.

Empirical Survival Function for Training Data:

The Empirical Survival Function for Training Data plot overlays the hazard and survival functions.



The horizontal axis on the Empirical Survival Function for Training Data plot represents the tenure of customers measured in months. The hazard function highlights different important events in the customer's lifecycle.

The very first hazard probability at time zero is 0.0186. This might represent customers that do not start the service right away, or customers that experience buyer's remorse.

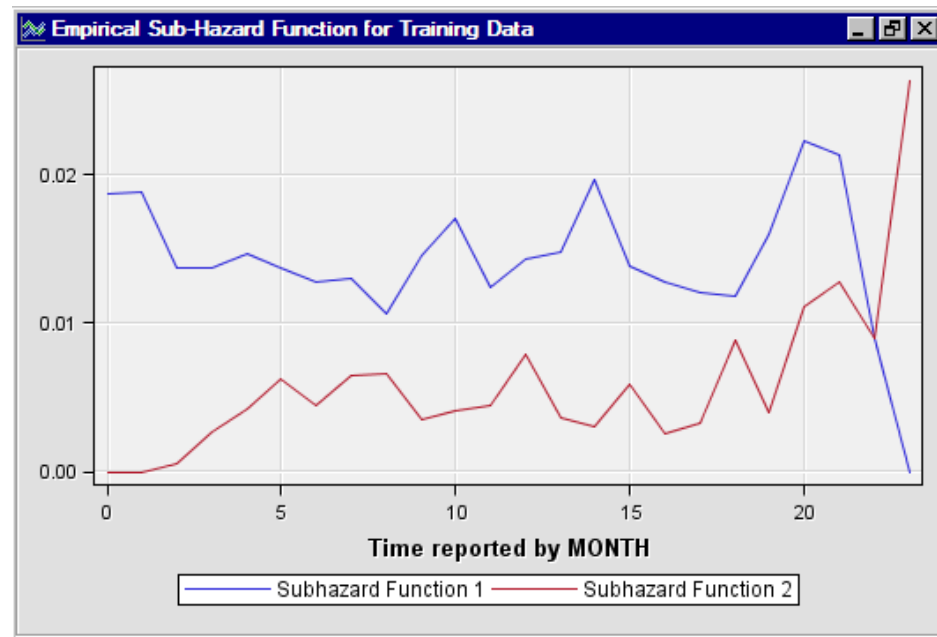
The first peak in the hazard function occurs in month 5. The peak might be a result of the cell phone company trying several customer payment incentives that did not work. Eventually, the cell phone service provider must force some churn due to non-payment.

The peaks that are shown in the hazard probability plot around months 10, 12, 14, and 20 might be due to the end of promotional periods. Customers who sign up for a service because of a highly discounted initial offer often discontinue service when the initial discount expires. On the bright side, the customers who tend to discontinue service when promotions expire are typically customers with no delinquent payments.

If hazard plots (in red) provide a snapshot of the customer lifecycle, then survival plots (in blue) provide a more complete picture. The survival probability at time t is the likelihood that a customer will survive to that point in time. The left vertical axis represents the survival likelihood of the customers. Notice that the curve starts at 1 (because at $t=0$, all customers are still active) and gradually declines to 0. The survival values will always be between 0 and 1. The survival probability curve can be used in conjunction with the hazard function. When a spike is observed in the hazard plot, a steep decline can be expected in the survival function, indicating that customers are not surviving beyond this point. In general, the smaller the hazards are, the flatter the survival curve is. When hazards are large, the survival slope is usually steeper.

Empirical Sub-Hazard Function:

The Empirical Sub-Hazard Function plot shows the subhazard functions for voluntary (Sub-Hazard Function 1 in blue) and involuntary churn (Sub-Hazard Function 2 in red).



This plot graphically describes competing risks in survival data mining. A good example of competing risks is the distinction between voluntary and involuntary churn. Some customers are forced to leave (typically due to non-payment) whereas others leave voluntarily. When modeling churn, sometimes models are built that ignore one or the other group of customers. Such practices could bias the model.

In the case of competing risks, there is a separate probability for each risk. After a customer experiences an at-risk event (such as voluntary churn), that customer is excluded from the remaining at-risk events. The plot above shows the competing risks for voluntary and involuntary churn. The top blue line shows that customers who succumb to voluntary churn are at greater risk.

Model Validation Statistics:

The Model Validation Statistics table shows at what depth the best benefit, lift, K-S statistic, and Gini concentration ratio statistics can be found.

Model Validation Statistics						
Data Role	Benefit	Average Hazard Ratio	Depth	Lift	Kolmogorov-Smirnov Statistic	Gini Concentration Ratio
Train	0.199391	0.018664	0.345425	1.577233	0.230021	0.215371
Valid	0.175779	0.022864	0.301772	1.58249	0.203368	0.186408

For example, in the validation data set, at a depth of 0.30 (about the top 30% of the data), the best lift value occurs.

The benefit value of 0.176 in the validation data set indicates this is the depth of the predicted probabilities at which the model best differentiates between the customers who experience a churn event, and customers who do not.

The K-S statistic and the Gini concentration ratio are both measures of how well the model is separating between the probabilities of churn and no churn. The K-S statistic measures the maximum vertical distance between the curves that represent the cumulative distributions of customers who experience a churn event and customers who do not churn.

Submit a Data Set for Scoring

After you build a good predictive model with the **Survival** node, you are ready to score a data set. The data set that contains the score data for this example is `SAMPSIO.CELLPHONE_SCORE`. Like the training data `SAMPSIO.CELLPHONE`, the score data set is included in the `SAMPSIO` sample data library that is included with your copy of SAS Enterprise Miner.

Before you can score a data set using score code from the **Survival** node, you must create a `_t_` variable. For this example, the `_t_` variable has already been created in the `SAMPSIO.CELLPHONE_SCORE` data set. When scoring data, the `_t_` variable represents the number of time intervals in a given time span. In our example, the `_t_` variable represents the elapsed time between the activation and the censoring date. If you look at the **Survival** node Results browser and view the flow code that was generated during Survival model training, you see the following:

```
format _currentdate MMDDYY10.0 ;
_currentdate=input("31DEC2000",anydtdte10.);
if (activation_date> _currentdate or deactivation_date< _currentdate)
    and deactivation_date^=.
    then _WARN_ ='U';
_T_=intck("MONTH",activation_date, _currentdate);
if _T_ ne . then do;
```

The `if _T_ ne . then do;` code statement indicates that if the `_t_` variable is not present in the score data, then the score values for sub-hazards, survival functions, and so on, are not computed.

To create the scoring data set variable `_t_` that the **Survival** node needs, the code below was submitted when `SAMPSIO.CELLPHONE_SCORE` was created. The purpose of the code is to create the `_t_` variable. You can write similar code of your own to create `_t_` variables for other data sets that you want to score.

```
data sampsio.cellphone_score;
set sampsio.cellphone;
format _currentdate MMDDYY10.0 ;
_currentdate=input("31DEC2000",anydtdte10.);
_T_=intck("MONTH",activation_date, _currentdate);
drop Target _currentdate;
run;
```

The SAS `intck` function returns the number of time intervals in a given time span. The first argument of the SAS `intck` function is the time unit that you want to use (MONTH in this example). The second and third arguments are the “from” and “to” values for the time interval. In this example, the “from” argument is the activation date of a customer’s cell account. The “to” argument is the censoring date that defines the end of the analysis interval.

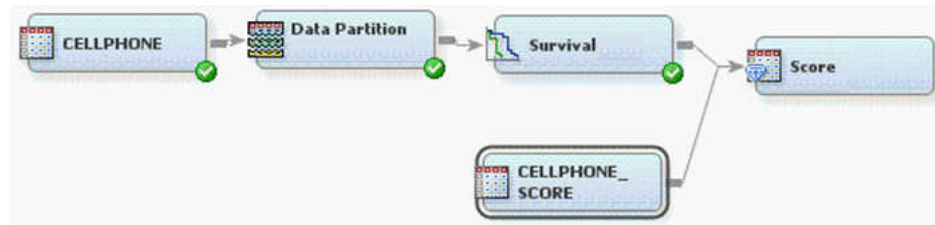
Right-click the Data Sources folder in the Project Panel, and then select **Create Data Source** to launch the Data Source Wizard.

In the Data Source Wizard, do the following to create your score data set as a SAS Enterprise Miner Data Source:

1. Choose **SAS Table** as your metadata source and click **Next**.
2. Enter `SAMPSIO.CELLPHONE_SCORE` in the **Table** field and click **Next**.
3. Continue to the Metadata Advisor and choose the **Advanced Metadata Advisor**.
4. In the Column Metadata window, make the following variable role assignments:

- a. Set the role of the variables ACCOUNT_NUM, ACTIVATION_DATE, DEACTIVATION_DATE, and DISABLE to **Rejected**.
 - b. Set the roles of GOOD_BAD, PLAN_TYPE, and PROVIDER_TYPE, and _t_ to **Input**.
 - c. Click **Next**.
5. Continue to the Data Source Attributes window and change the **Role** of the score data set to **Score**. Click **Next**.
6. Accept the default settings for the remainder of the Data Source Wizard windows. Select **Next** for each window, and when you reach the last Data Source Wizard window, select **Finish**.
7. The SAMPSIO.CELLPHONE_SCORE data set should appear in your Data Sources folder in the SAS Enterprise Miner Project Panel.

Drag the CELLPHONE_SCORE data source that you just created from the Data Sources folder in the Project Panel onto the Diagram Workspace. Next, drag a **Score** node from the **Assess** tab of the node Tools bar onto the Diagram Workspace. Then, connect the nodes as shown below:



Right-click the Score node and select **Run**. After the Process Flow Diagram runs, follow the prompts to open the Results browser.

The screenshot displays the SAS interface with three main windows:

- Optimized SAS Code:** Contains SAS code for variable definitions and labels.


```

34 length _warn_ $ 4;
35 label _warn_ = "Warnings";
36 label EM_SURVIVAL = "Survival Probability";
37 label EM_SURVFCST = "Survival Probability";
38 label EM_SURVEVENT = "Event Probability";
39 label EM_HAZARD = "Hazard Function at Center";
40 label EM_HZRDFCST = "Hazard Function at Center";
41 length _uname $ 32;
42 DROP _uname;
43
44 if _T_ ne . then do;
45
46 label T_FCST = "Number of Time Units in Future";
47 T_FCST=T+3 ;
48
49
50
51 /*-----Generate Cubic Spline Basis Functions-----*/

```
- Output:** Shows the results of the SAS code execution, including variable definitions and labels.


```

19
20 *-----
21 * Score Output
22 *-----
23
24
25
26
27 Score Input Variables
28
29
30
31 Variable Name    Role    Creator    Comment    Label
32
33 good_bad        INPUT
34 plan_type       INPUT
35 provider_type   INPUT
36

```
- SAS Code:** Contains the same SAS code as the Optimized SAS Code window.


```

31 /*-----
32 /*rpt_surv_score_title_begin*/
33 /*-----
34
35 length _warn_ $ 4;
36 label _warn_ = "Warnings";
37 label EM_SURVIVAL = "Survival Probability";
38 label EM_SURVFCST = "Survival Probability";
39 label EM_SURVEVENT = "Event Probability";
40 label EM_HAZARD = "Hazard Function at Center";
41 label EM_HZRDFCST = "Hazard Function at Center";
42 length _uname $ 32;
43 DROP _uname;
44
45 if _T_ ne . then do;
46
47 label T_FCST = "Number of Time Units in Future";
48 T_FCST=T+3 ;
49
50
51

```
- Output Variables:** A table showing the variables generated by the SAS code.

Variable Name	Creator	Variable Label	Function	Type
EM_HAZARD	SURV	Hazard Function at Cen...	TRANSFORM	N
EM_HZRDF...	SURV	Hazard Function at Fut...	TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SUBHZ...	SURV		TRANSFORM	N
EM_SURVEV...	SURV	Event Probability before...	TRANSFORM	N
EM_SURVF...	SURV	Survival Probability at F...	TRANSFORM	N
EM_SURVIVAL	SURV	Survival Probability at C...	TRANSFORM	N
T_FCST	SURV	Number of Time Units i...	TRANSFORM	N
T	SURV		TRANSFORM	N
warn	SURV	Warnings	TRANSFORM	C

The SAS scoring code is shown in the left two panes. The Output Variables tables shows that the hazard, sub-hazards, and survival functions were all generated.

Time Series

- The **Time Series Correlation** node provides autocorrelation and crosscorrelation analysis for timestamped data. .
- The **Time Series Data Preparation** node provides time series data cleaning, summarization, transformation, transposition, and so on.
- The **Time Series Decomposition** node computes classical seasonal decomposition of time series data.
- The **Time Series Dimension Reduction** node currently supports five time series dimension reduction techniques. The goal is to approximate the original time series with a more compact time series that retains as much information from the original series as possible.
- The **Time Series Exponential Smoothing** node generates forecasts by using exponential smoothing models with optimized smoothing weights for many time series models.
- The **Time Series Similarity** node computes similarity measures associated with time-stamped data.

Credit Scoring

Note: The SAS Credit Scoring feature is not included with the base version of SAS Enterprise Miner. If your site has not licensed SAS Credit Scoring, the credit scoring node tools will not appear in your SAS Enterprise Miner software.

Analysts can use SAS Enterprise Miner and its credit scoring tools to build scorecard models that assign score points to customer attributes, to classify and select characteristics automatically or interactively using Weights of Evidence and Information Value measures, and to normalize score points to conform with company or industry standards.

- The **Credit Exchange** node enables you to exchange the data that is created in SAS Enterprise Miner with the SAS Credit Risk Solution. The **Credit Exchange** node creates a statistics table, a target table, flow score code, and a required input variables matrix.
- The **Interactive Grouping** node groups variable values into classes that you can use as inputs for predictive modeling, such as building a credit scorecard model. Use the **Interactive Grouping** node to interactively modify classes in order to create optimal binning and grouping.
- The **Reject Inference** node uses the scorecard model to score previously rejected applications. The observations in the rejected data set are classified as inferred "goods" and inferred "bads". The inferred observations are added to the Accepts data set that contains the actual "good" and "bad" records, forming an augmented data set. This augmented data set can then serve as the input data set of a second credit scoring modeling run.
- The **Scorecard** node fits a logistic regression model for the binary target variable. The regression coefficients and scaling parameters are used to calculate scorecard points. Scorecard points are organized in a table that assigns score points to customer attributes. You use the **Scorecard** node to calculate the scaling parameters of a credit scorecard, display the distribution of various score-related statistics, determine an optimal cutoff score, and generate a scored data set.

Incremental Response



Overview

The Incremental Response node models the incremental impact of a treatment (such as a marketing action or incentive) in order to optimize customer targeting for maximum return on investment. The Incremental Response node can determine the likelihood that a customer purchases a product, uses a coupon, or predicts the incremental revenue realized during a promotional period. The Incremental Response node is located on the **Applications** tab of the SAS Enterprise Miner Nodes toolbar.

To better understand incremental response modeling, suppose that potential customers are divided into the following groups:

- **Persuadables:** These are customers who respond only when they are targeted. Persuadables can also be thought of as true responders.
- **Sure Things:** These are customers who will respond anyway. Marketing outreaches have no measurable impact on this group.
- **Lost Causes:** These are customers who will not respond whether they are targeted or not. Marketing outreaches have no measurable impact on this group.
- **Sleeping Dogs, or Do Not Disturb:** These are customers who are *less* likely to respond if they are targeted.

Conventional response models target marketing actions on people who are most likely to buy. However, the cost of this approach is wasted on the **Sure Things** group of customers, who would buy anyway. The only potential customer group that provides true incremental responses is the **Persuadables** group. In order to maximize incremental sales at minimum cost, ideally only the **Persuadables** group should be targeted by a marketing action.

In order to identify the incremental impact that is associated with a specific direct marketing action, Incremental Response uses control groups to measure the difference in response rate between the treated group and the control group.

In cases where there is a binary outcome, such as purchase/no purchase, incremental response modeling considers the probability of purchase for each customer under two scenarios—treated and non-treated.

In cases where there are two target variables, such as a binary variable for purchase/no purchase, and an interval variable that represents the amount of revenue in the event that purchase occurs, incremental response modeling considers the difference in the amount of expected sales for each customer under treated and non-treated scenarios. This approach can be called an incremental sales model.

Before building the predictive model, the Incremental Response node enables you to perform predictive variable selection in order to identify the variables most likely to maximize the incremental response rate. Variable selection is important in incremental response models because it can improve model stability as well as predictive accuracy. The Incremental Response node uses the Net Information Value (NIV) technique to perform variable selection. NIV measures the differential in information values between the control group and the treatment group for each variable.

Input Data Requirements for the Incremental Response Node

The Incremental Response node requires a binary treatment variable. Because the Incremental Response node measures response differentials between control and treatment groups, the tool must be able to distinguish between data that represents control groups and data that represents treatment groups. The binary treatment variable (0 for control group, 1 for treatment group) indicates which group an observation belongs to.

The Incremental Response node also requires a binary target variable called the response variable. The response variable indicates the outcome of the marketing action on the customer. For example, a purchase response variable would have a value of 0 for no purchase and 1 for purchase.

The Incremental Response node enables you to define an optional interval target variable that is called the outcome variable. For example, if you wanted to construct an

incremental sales model, you would require a binary response variable (purchase /no purchase) and an interval outcome variable (purchase amount).

In SAS Enterprise Miner 12.3 and later, the Incremental Response node uses a penalized NIV (Net Information Value) calculation to perform variable selection when a validation data set exists. The penalty term is calculated as the differentiation between train data and validation data.

Note: The **Incremental Response** node is an application node that does not use frequency variables. Frequency variables are not recommended for use in **Incremental Response** node models.

Missing Values in Incremental Response Node Input Data


The Incremental Response node excludes observations that have missing values from its analytical analysis. If your input data contains significant missing values, consider using the Impute node to replace the missing values in your input data with imputed values.

Incremental Response Node Properties

Incremental Response Node General Properties


The following general properties are associated with the Incremental Response Node:

- **Node ID** — displays the ID that SAS Enterprise Miner assigns to a node in a process flow diagram. Node IDs are important when a process flow diagram contains two or more nodes of the same type. The first Incremental Response node that is added to a diagram will have a Node ID of IR. The second Incremental Response node that is added to a diagram will have a Node ID of IR2, and so on.

- **Imported Data** — provides access to the Imported Data — Incremental Response window. The Imported Data — Incremental Response window contains a list of the ports that provide data sources to the Incremental Response node. Select the  button to the right of the Imported Data property to open a table of the imported data.


If data exists for an imported data source, you can select the row in the imported data table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.
- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.

- **Exported Data** — provides access to the Exported Data — Incremental Response window. The Exported Data — Incremental Response window contains a list of the output data ports that the Incremental Response node creates data for when it runs. Select the  button to the right of the Exported Data property to open a table that lists the exported data sets.

If data exists for an exported data set, you can select the row in the table and click one of the following buttons:

- **Browse** to open a window where you can browse the data set.
- **Explore** to open the Explore window, where you can sample and plot the data.

- **Properties** to open the Properties window for the data source. The Properties window contains a **Table** tab and a **Variables** tab. The tabs contain summary information (metadata) about the table and variables.
- **Notes** — Select the  button to the right of the Notes property to open a window that you can use to store notes of interest, such as data or configuration information.


Incremental Response Node Train Properties

The following Train properties are associated with the Incremental Response Node:

- **Variables** — Use the Variables window to view variable information. Select the ellipsis button to the right of the Variable property to open a variables table. The Use column in the variables table can be used to change the Use status for individual variables in certain models.

You can view the columns metadata. You can also open an Explore window to view a variable's sampling information, observation values, or a variable distribution plot. By default, columns for variable Name, Use, Role, and Level are displayed.

You can modify these values, and add additional columns using the following options:

- **Apply** — changes metadata based on the values supplied in the drop-down menus, check box, and selector field.
- **Reset** — changes metadata back to its state before use of the **Apply** button.
- **Label** — adds a column that displays the label for each variable.
- **Mining** — adds columns for the Report, Order, Lower Limit, Upper Limit, Creator, Comment, and Format Type for each variable.
- **Basic** — adds columns for the Type, Format, Informat, and Length of each variable.
- **Explore** — opens an Explore window that enables you to view a variable's sampling information, observation values, or a plot of variable distribution.
- **Treatment Level Selection** — Select the  button to the right of the Treatment Level Selection property to open a window that enables you to set the treatment level of the treatment variable.
- **Prescreen Variables** — Set the Prescreen Variables property to **Yes** if you want to choose a subset of predictive variables before building the model. Variables selection is performed based on the net information value (NIV) for each variable. NIV is determined using the following formulas:

$$NWOE_i = \log \left(\frac{P(X = x_i | Y = 1)_T / P(X = x_i | Y = 0)_T}{P(X = x_i | Y = 1)_C / P(X = x_i | Y = 0)_C} \right)$$

$$NIV = \sum_{i=1}^n (P(X = x_i | Y = 1)_T \cdot P(X = x_i | Y = 0)_C - P(X = x_i | Y = 0)_T \cdot P(X = x_i | Y = 1)_C) \cdot NWOE_i$$

Here, $P(X | Y)_T$ and $P(X | Y)_C$ are the conditional probabilities for the treatment group and the control group, respectively.

Note that the NIV value is penalized when a validation data set exists. The penalty term is the difference between the training data and the validation data.

- **Rank Percentage Cutoff** — Specifies the percentage of the variables to select if the Prescreen Variables property is set to Yes. Ranks all predictive variables according to the net information value, and selects variables that have higher net information values. The default setting for the Rank Percentage Cutoff property is 50%.

Incremental Response Node Train Properties: Model Selection

- **Combined Model** — Set the **Combined Model** property to **Yes** if you want to include the treatment variable as one predictor in the model, instead of running separate models for treatment group and control group.

When set to **No**, the treatment variable is used to partition the data. Separate predictive models are implemented on the treatment and control groups. For each data set bin, a difference score is calculated as follows:

$$D = E(Y_T) - E(Y_C)$$

The net scores are then ranked and the top bins are selected as optimal targets.

The Combined Model is based on Lo's "The True Life Model — A Novel Data Mining Approach to Response Modeling in Database Marketing" (2002). In this model, the treatment variable T is a covariate indicator variable that interacts with the other effects. Therefore, the response variable is modeled as

$Y = X\beta + T\gamma + (XT)\phi + \epsilon$. The difference score is calculated as follows:

$$D = E(Y|X, T = 1) - E(Y|X, T = 0)$$

As with the other model, net scores are ranked and the top bins are selected as optimal targets.

- **Selection Method** — specifies the method for selecting effects for the selection process. The default value for the **Selection Method** property is **None**, which specifies no model selection and the complete model is used.
 - **Forward** — starts with no effects in the model and adds effects until the Entry Significance Level is met.
 - **Stepwise** — similar to the Forward method, except that the effects already in the model do not necessarily remain in the model.
 - **Backward** — starts with all effects in the model and incrementally removes effects.
- **Selection Criterion** — chooses from the list of models at each step of the selection process to find the model that yields the best value for the specified criterion. The default value of the **Selection Criterion** property is **None**, which chooses the model at the final step of the selection process.
 - **AIC** — chooses the model that has the smallest Akaike Information Criterion value.
 - **SBC** — chooses the model that has the smallest Schwarz's Bayesian Criterion value.
 - **Validation Error** — chooses the model that has the smallest misclassification rate or error rate for the validation data set. The misclassification rate is used for binary targets, and the error rate is used for interval targets. The error rate is measured using the sum of squared errors statistic.
 - **Cross-Validation Error** — chooses the model that has the smallest misclassification rate or error rate for the cross validation of the training data set. The misclassification rate is used for binary targets, and the error rate is used for

interval targets. The error rate is measured using the sum of squared errors statistic.

- **Significance Level for Entry** — specifies the significance level to be used as a threshold criterion for adding input variables. The default value for the Significance Level for Entry property is 0.0. This means that variables that have a p-value that is less than or equal to 0.05 are added as inputs.
- **Stay Significance Level** — specifies the significance level to be used as a threshold criterion for removal of input variables. The default value for the Significance Level for Entry property is 0.0. This means that variables that have a p-value that is less than or equal to 0.05 are removed as inputs.
- **Suppress Intercept** — Set the Suppress Intercept property to Yes if you want to suppress the intercept term in the model. The default setting for the Suppress Intercept property is No.
- **Two-Way Interactions** — Set the Two-Way Interactions property to Yes if you want to include all two-way interaction terms for class variables and all of the second-order polynomial terms of interval variables that have a status of Use. The default setting for the Two-Way Interactions property is No.

Incremental Response Node Train Properties: Revenue Calculation

- **Use Constant Revenue** — Set the Use Constant Revenue property to Yes if you want to specify a fixed revenue for each response. This option overrides the expected revenue for individual customers. The expected revenue for individual customers is the estimated outcome from the model. The default setting for the Use Constant Revenue property is No.
- **Revenue Per Response** — specifies the fixed revenue for each response if the Use Constant Revenue property is set to Yes. The default setting for the Revenue Per Response property is 10.
- **Use Constant Cost** — specifies whether to use a fixed cost for each observation. If you specify **Yes**, then the value that is specified in the **Cost** property will override the cost variable in the input data set.
- **Cost** — specifies the cost of the direct marketing action for each contact. The default value of the Cost property is 0. A customer is considered as profitable in the incremental response model only if the incremental revenue is greater than the cost.

Incremental Response Node Report Properties

- **Number of Bins** — specifies the number of bins, n . The width of each bin is $100\% / n$.

Incremental Response Node Status Properties

The following status properties are associated with this node:

- **Create Time** — displays the time at which the node was created.
- **Run ID** — displays the identifier of the node run. A new identifier is created every time the node runs.
- **Last Error** — displays the error message from the last run.
- **Last Status** — displays the last reported status of the node.
- **Last Run Time** — displays the time at which the node was last run.
- **Run Duration** — displays the length of time of the last node run.

- **Grid Host** — displays the grid server that was used during the node run.
- **User-Added Node** — specifies if the node was created by a user as a SAS Enterprise Miner extension node.

Incremental Response Node Results

You can open the Results browser of the Incremental Response node by right-clicking the node and selecting **Results**.

Select **View** on the main menu to view the following results in the Incremental Response — Results window:

- **Properties**
 - **Settings** — displays a window with a read-only table of the Incremental Response node properties configuration when the node was last run. Use the Show Advanced Properties check box at the bottom of the window to see all of the available properties.
 - **Run Status** — indicates the status of the Incremental Response node run. The Run Start Time, Run Duration, and information about whether the run completed successfully are displayed in this window.
 - **Variables** — a table of the variables in the training data set. You can resize and reposition columns by dragging borders or column headings. You can toggle column sorts between descending and ascending by clicking on the column headings.
 - **Train Code** — the code that SAS Enterprise Miner used to train the node.
 - **Notes** — enables users to read or create notes of interest.
- **SAS Results**
 - **Log** — the SAS log of the Incremental Response node run.
 - **Output** — the SAS output of the Incremental Response node run.
 - **Flow Code** — the SAS code used to produce the output that the Incremental Response node passes on to the next node in the process flow diagram.
- **Scoring**
 - **SAS Code** — the SAS score code that was created by the node. The SAS score code can be used outside the SAS Enterprise Miner environment in custom user applications.
 - **PMML Code** — The Incremental Response node does not generate PMML code.
- **Model**
 - **Selected Variables by NIV** — a table of the selected variables and their ranks in the original variable set according to the net information values.
 - **Response Outcome Summary Table** — a table of summary statistics of the data, such as number of observations, number of responses, response rate, average outcome, and total outcome.
 - **Response Outcome Summary** — displays bar charts of the statistics in the Response Outcome Summary Table for both treatment and control groups.

- **Parameter Estimates for Response Model** — a plot of parameter estimates for the treatment and control groups. Parameter estimates are provided for each input variable that is in the response model.
- **Parameter Estimates for Outcome Model** — a plot of parameter estimates for the treatment and control groups. Parameter estimates are provided for each input variable that is in the outcome model. The default outcome model is a two-stage model that uses the inverse Mills ratio. The combined model uses separate regressions for the binary and interval target. The interval target is zero unless there is a response.
- **Plots**
 - **Net Weight of Evidence** — displays the net weight of evidence for each explanatory variable.
 - **Response Outcome Summary** — plots various measures of success based on whether customers received treatment.
- **Revenue Plots**
 - **Average Revenue** — displays the average revenue within the percentile for the treatment group and the control group. The horizontal axis is the percentile based on the rank order of the predicted incremental revenue. The predicted incremental revenue is the difference between the treatment group and the control group.
 - **Average Incremental Revenue** — displays the average incremental revenue within the percentile. The increment is the difference of the expected revenue between the treatment group and the control group. The cost per contact is subtracted if it is specified in the Revenue Calculation options.

To determine the average incremental revenue, let R_T be the treatment group revenue, R_C be the control group revenue, and R_I be the incremental revenue. In addition, let Rev be either the revenue defined by the **Revenue per Response** property or the revenue value determined by the data. $Cost$ is the value determined by either the **Cost** property or determined by the data. Finally, let Y_T and Y_C be the response value for customers in the treatment and control groups, respectively. Average incremental response is defined as follows:

$$\hat{R}_T = \hat{Y}_T \cdot Rev - Cost$$

$$\hat{R}_C = \hat{Y}_C \cdot Rev$$

$$\hat{R}_I = \hat{R}_T - \hat{R}_C$$

Note that because observations are binned based on the **Number of Bins** property, these equations compute averages within each bin.

- **Response Plots**
 - **Treatment Response Model Diagnostics** — shows the response rate within the percentile for both the predicted treatment and observed treatment.
 - **Control Response Model Diagnostics** — shows the response rate within the percentile for both the predicted control and observed control.
 - **Incremental Response Model Diagnostics** — displays the predicted and observed change in the response rates.

- **Cumulative Incremental Response Diagnostics** — displays the cumulative incremental response rate for both predicted and observed data.
- **Cumulative Incremental Response** — displays the predicted and observed cumulative incremental response rate for the data.
- **Outcome Plots**
 - **Treatment Outcome Model Diagnostics** — shows the average outcome within the percentile for both the predicted treatment and observed treatment.
 - **Control Outcome Model Diagnostics** — shows the average outcome within the percentile for both the predicted control and observed control.
 - **Incremental Outcome Model Diagnostics** — displays the predicted and observed change in outcome values.
 - **Cumulative Incremental Outcome Diagnostics** — displays the predicted and observed incremental outcome for the predicted data.
 - **Cumulative Incremental Response** — displays the cumulative incremental outcome for both predicted and observed data.
- **Table** — displays a table that contains the underlying data that is used to produce a chart. The Table menu item is dimmed and unavailable unless a results chart is open and selected.
- **Plot** — opens the Graph Wizard for the selected table. You can modify an existing Results plot or create a Results plot of your own. The Plot menu item is dimmed and unavailable unless a Results chart or table is open and selected.

Incremental Response Node Output Variables

The score code of the Incremental Response node produces the following output variables, which can be used for business decisions based on the incremental response model:

- **EM_P_TREATMENT_RESPONSE**: the predicted response probability from the treatment model.
- **EM_P_CONTROL_RESPONSE**: the predicted response probability from the control model.
- **EM_P_INCREMENT_RESPONSE**: the incremental predicted response rate.
- **EM_P_ADJ_INCREMENT_RESPONSE**: the adjusted incremental predicted response rate, which makes the incremental predicted response rate positive.
- **EM_P_ABS_INCREMENT_RESPONSE**: the absolute value of the incremental predicted response rate, when an interval outcome target is used with the response variable.
- **EM_P_TREATMENT_OUTCOME**: the predicted value of the outcome variable from the treatment model.
- **EM_P_CONTROL_OUTCOME**: the predicted value of the outcome variable from the control model.
- **EM_P_INCREMENT_OUTCOME**: the predicted value of the incremental outcome.

When an interval outcome target or constant revenue is used with the response target variable, the following output variables are produced:

- **EM_REV_TREATMENT**: the estimated revenue for the treatment model.

- EM_REV_CONTROL: the estimated revenue for the control model.
- EM_REV_INCREMENT: the estimated incremental revenue.

Incremental Response Node Example

Overview

The following example uses incremental response modeling to examine opportunities for incremental sales as well as to enhance the profitability and effectiveness of a direct marketing campaign. A direct marketing team can use the modeling results to identify and target a reduced number of customers whose purchasing behavior can be changed positively by a treatment such as a sales incentive or a marketing promotion. The model measures the incremental sales that are generated by the promotion as well as examines the incremental customer responses. Incremental customer responses are examined and ordered from the customers who are most positively influenced by a treatment to those who are most negatively influenced by a treatment. The model also provides the estimated incremental revenue that could be realized because of the marketing action.

Create and Configure the Incremental Response Node Data Source

This example uses a data set from the SAMPSIO library called DMRETAIL. DMRETAIL is a training data set of 9,000 observations that documents the purchasing behaviors of customers. Each observation in the data set represents the purchasing history of a customer, including purchases made during the promotional incentive period.

Not all customers receive the promotional marketing treatment. The 9,000 customers are divided evenly into two treatment groups, as indicated by the binary treatment variable value in each observation. Half (4,500) of the customers in the data set are in treatment group 0. Treatment group 0 receives no marketing incentive promotion. But their purchase behaviors are recorded over the period of time that corresponds exactly with the duration of the marketing incentive promotion. The remaining 4,500 customers are in treatment group 1, which receives a treatment in the form of a promotional marketing incentive. Partitioning the data set into equal treatment and non-treatment populations allows the behavioral effect of the marketing promotion on customers to be statistically analyzed and measured.

The DMRETAIL table contains two variables that will be used as targets. A binary target variable Response indicates whether a purchase takes place during the promotional treatment period. An interval target variable Sales indicates the amount of the purchase, if a customer buys. The model results will indicate a customer's likelihood to make a purchase during the period of interest, as well as the expected amount of the purchase.

The Incremental Response node does not require two target variables. Only a binary target variable for the purchase/no purchase response is required. When you create an Incremental Response node model using only a binary response target variable, the model can predict only the likelihood of a customer to make a purchase during the period of interest.

Use the following steps to convert the SAS table SAMPSIO.DMRETAIL from a data set into a SAS Enterprise Miner data source:

1. In the SAS Enterprise Miner Project Navigator, right-click the Data Sources folder and select **Create Data Source**. This opens the Data Source Wizard. The default selection for metadata source is SAS Table. Accept the default setting by clicking **Next**.

2. In the field for **Select a SAS table**, enter SAMPSIO.DMRETAIL and then click **Next**.
3. The Table Properties display provides summary information about the SAMPSIO.DMRETAIL table. Select **Next** to advance to the **Metadata Advisor Options** portion of the Data Source Wizard. In the **Metadata Advisor Options**, select the **Advanced** button and click **Next**.
4. In the **Column Metadata** display of the Advanced Advisor, make the following variable role assignments:
 - Set the role of the variable **Promotion** to **Treatment**. Ensure that the Level for **Promotion** is binary.
 - Set the role of the variable **Response** to **Target**. Ensure that the Level for **Response** is binary.
 - Set the role of the variable **Sales** to **Target**. Ensure that the Level for **Sales** is interval.
 - Select **Next**.
5. Accept the default settings for the remainder of the windows in the Data Source Wizard. Select **Next** to advance through each window until you reach the last Data Source Wizard window, and select **Finish**.
6. The SAMPSIO.DMRETAIL data source should appear in your **Data Source** folder in the SAS Enterprise Miner Project Navigator.

Create and Configure the Incremental Response Process Flow Diagram

Right-click the Diagrams folder in the Project Navigator and select **Create Diagram**. You can name the diagram whatever you like. Drag the DMRETAIL data source that you just created to your newly created diagram workspace.

Next, from the **Sample** tab of the nodes toolbar, drag a Data Partition node to the workspace, and connect the DMRETAIL data source to the Data Partition node. Select the **Data Partition** node. Then use the Properties Panel to partition the DMRETAIL data into 60% training data and 40% validation data. This example does not use a holdout test data partition.

Drag an Incremental Response node from the **Applications** tab of the nodes toolbar to the diagram workspace. Connect the Data Partition node to the Incremental Response node.

The default property settings of the Incremental Response node are sufficient for this example. You do not need to make any changes.

Your process flow diagram should resemble the one shown below:

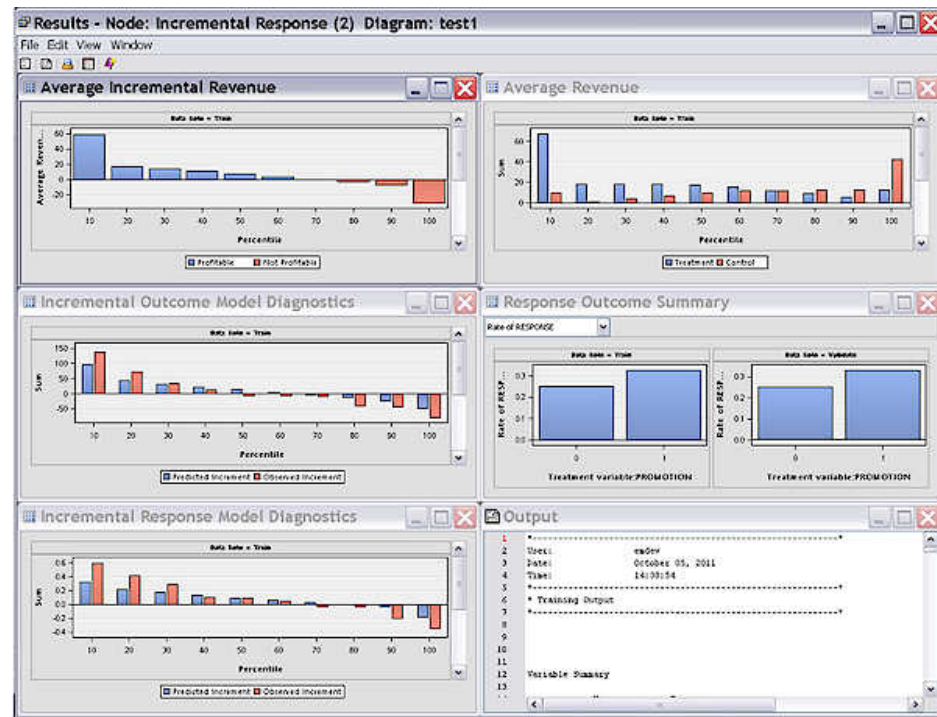


Run the Incremental Response Process Flow Diagram

Right-click the Incremental Response node in the diagram workspace and select **Run**. After the process flow diagram finishes running, click **Yes** to open the Incremental Response node Results browser.

Examine the Incremental Response Model Results

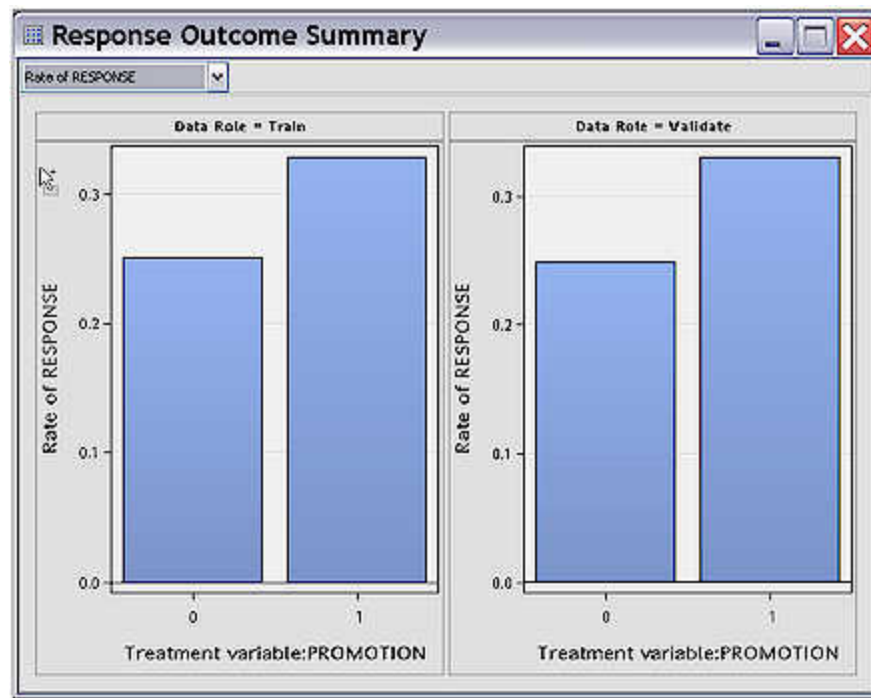
The Incremental Response node Results browser opens to display the Results plots and tables.



The plots and tables include the Response Outcome Summary plot, the Average Revenue plot, the Average Incremental Revenue plot, the Parameter Estimates for Response Model, the Parameter Estimates for Outcome Model, the Treatment Response Model Diagnostics, the Control Response Model Diagnostics, the Incremental Response Model Diagnostics, the Cumulative Incremental Response Diagnostics, the Treatment Outcome Model Diagnostics, the Control Outcome Model Diagnostics, the Incremental Outcome Model Diagnostics, the Cumulative Incremental Outcome Diagnostics, and the Selected Variables Table by NIV (Net Information Value).

The default display contains the Average Revenue plot, the Response Outcome Summary plot, the Average Incremental Revenue plot, the Incremental Outcome Model Diagnostics plot, and Incremental Response Model Diagnostics plot. To access other plots and tables, click the **View** tab and see the Model section, the Revenue Plots section, the Response Plots section, or the Outcome Plots section.

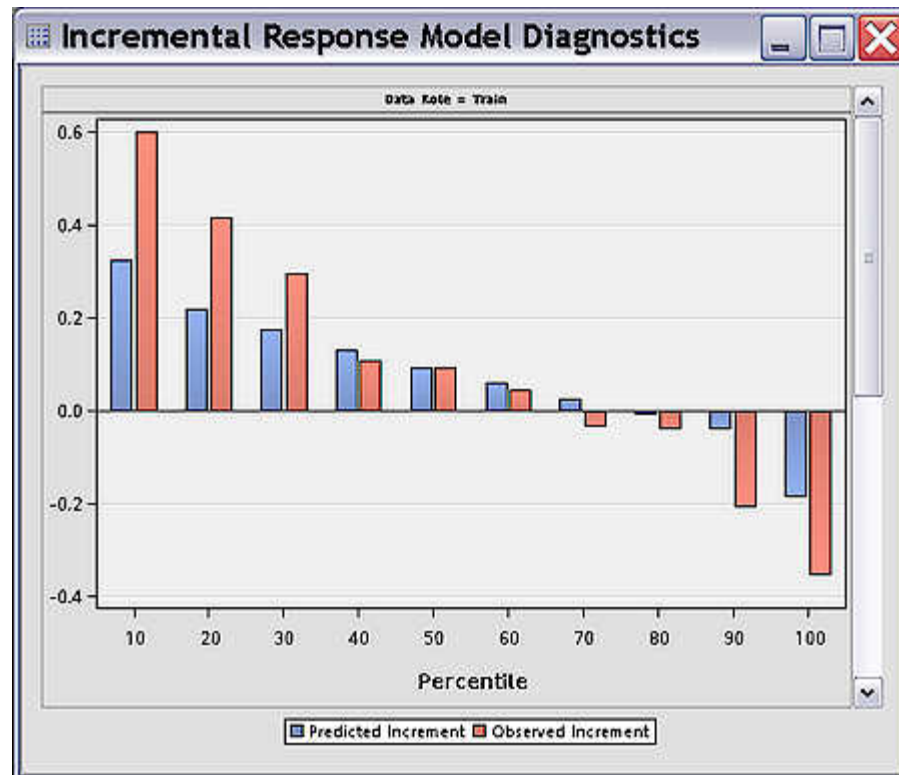
First, examine the Response Outcome Summary plot. By default, it opens to the Rate of Response selection.



This plot shows the rate of response (purchases) across the control and treatment groups. The responses across the training and validation data appear to be uniform, indicating little variation in the characteristics of the two partitioned data sets. The y-axis represents percent response.

The control group, which received no treatment, displays a 25.08% response. Approximately 1 in 4 customers made a purchase without any marketing treatment. The treatment group indicates a 32.91% response. that is, approximately 1 in 3 customers made a purchase in the group that received a marketing incentive promotion. The customers in the treatment group are not homogenous. As in the control group, some customers would purchase whether they received a marketing treatment. Other customers in the treatment group presumably purchased because of the marketing incentive that they received. Comparing the response rates of the control group to the treatment group, it would appear that approximately 7.83% ($32.91\% - 25.08\%$) of the customers in the treatment group were “true” responders to the marketing promotion. This 7.83% represents the average incremental response rate.

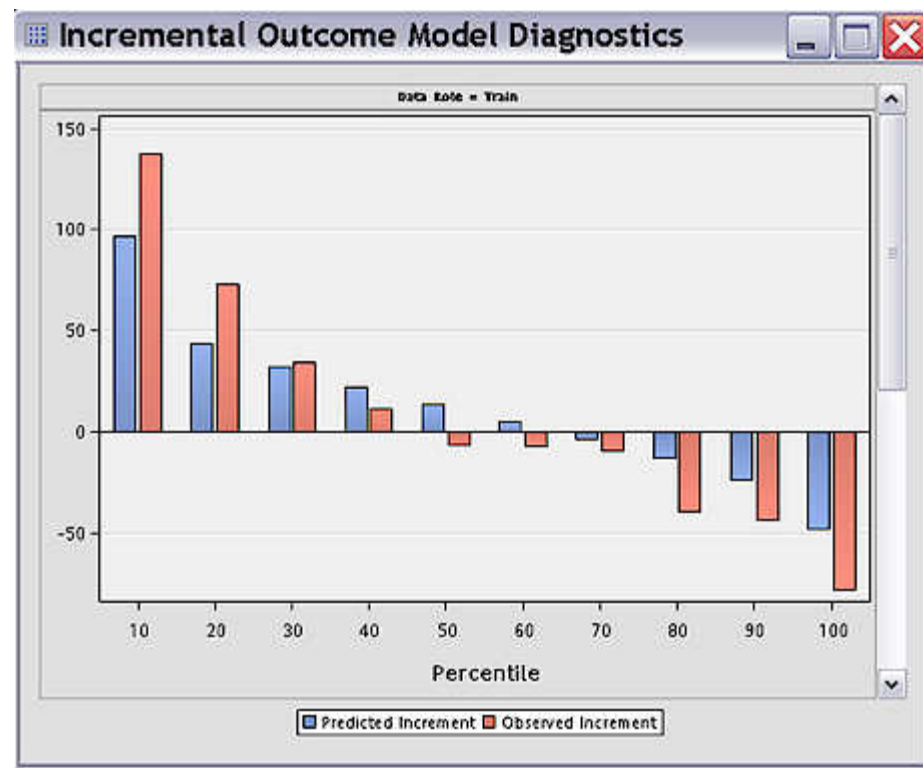
Next, examine the Incremental Response Model Diagnostics plot. The Incremental Response Model Diagnostics plot displays both the predicted and observed incremental response rate. The increment is calculated by subtracting the control response rate from the treatment response rate.



This plot displays incremental response by deciles. The strongest response decile is on the left (the top 10% of respondents), and the weakest response decile is on the right (the weakest 10% of respondents). The predicted incremental response rate by percentile in blue displays a declining pattern. The observed increment response rate in red displays a similar trend. The top 10% of customers have the observed incremental response rate of 59.77%. The expected incremental response rate is 32.29%. These rates are much higher than the average incremental rate of 7.83% that is indicated in the Response Outcome summary statistics mentioned above.

The results demonstrate that the model has the potential to pick up a significant portion of the customers that are most likely to be positively influenced by the promotion. It provides a guideline for targeting the Persuadables, helps justify the expense of the marketing promotion, and indicates the ability to minimize the risk of negative responses to the promotion.

Next, examine the Incremental Outcome Model Diagnostics plot. The Incremental Outcome Model Diagnostics plot displays both the predicted and observed incremental outcomes. The increment is calculated by subtracting the control outcome from the treatment outcome.

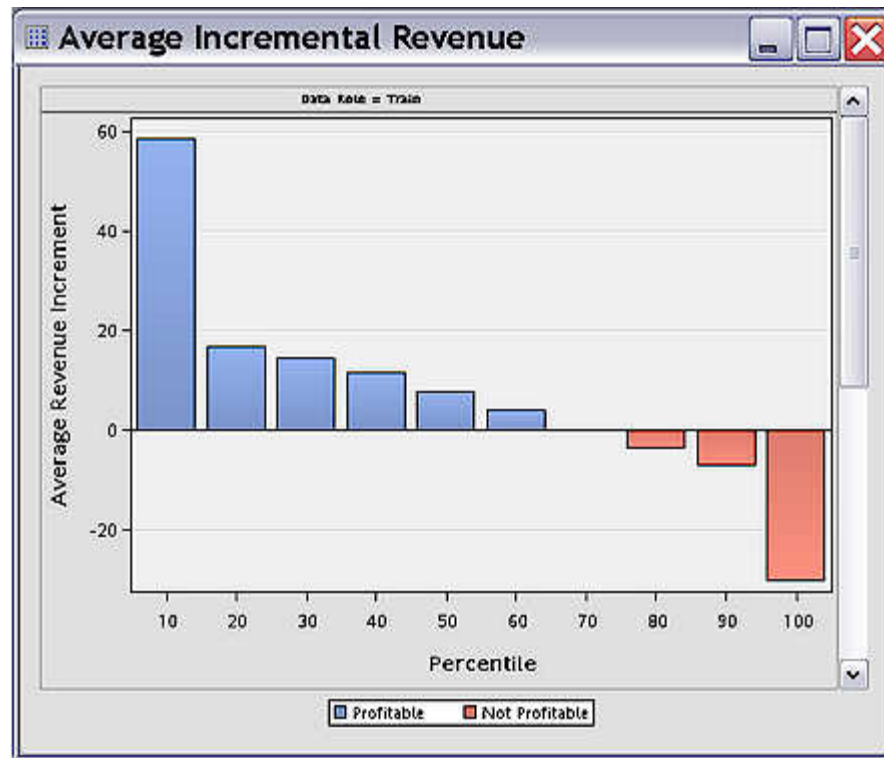


This plot evaluates the model performance from the sales perspective. If the customer makes any purchase during the promotion period, the purchase amount is included as a target variable. A two stage model is run to estimate the outcome as well. The top percentile contains the customers who are most likely to spend more if a marketing incentive is provided. The top 10% of customers indicate an observed incremental outcome of \$137.19, with an expected incremental response rate of \$96.76. These rates are both much higher than the average incremental rate of \$1.42.

Note: The average incremental rate of \$1.42 can be found in the Response Outcome Summary plot by setting the plot selector in the upper left corner to Average Sales. Examining the chart for Average Sales, you can see that the average sales for the treatment group (PROMOTION = 1) is \$158.95, and the average sales for the control group (PROMOTION = 0) is \$157.53. The resulting difference of \$1.42 represents the average incremental sales rate.

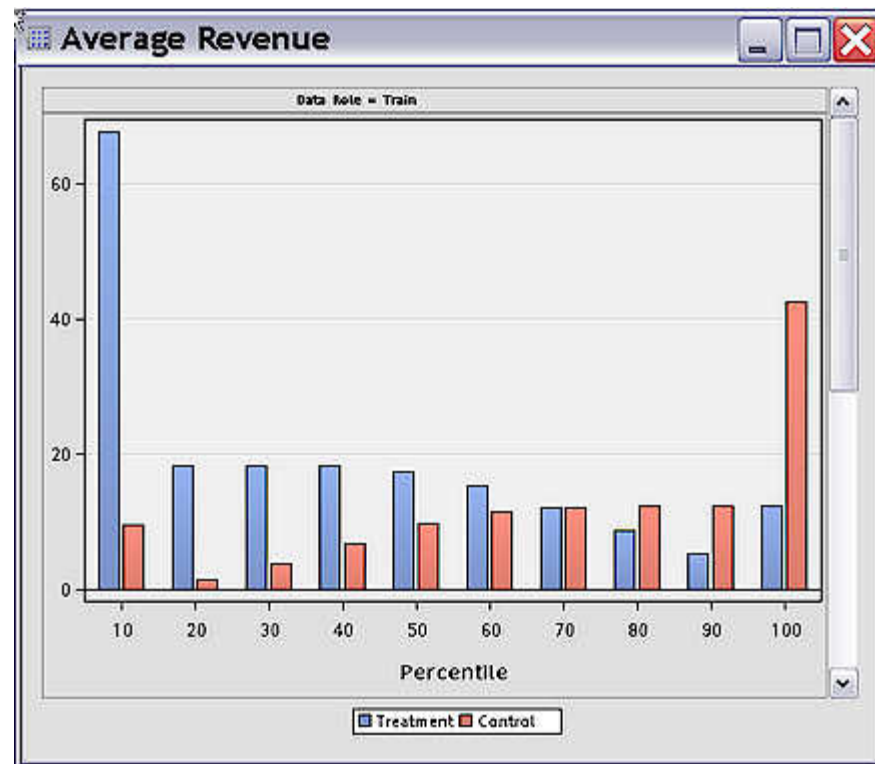
These results help us identify the customers who display the maximal increment in sales from a much larger group that includes the customers who will make purchases whether or not a promotion was offered.

Next, examine the Average Incremental Revenue plot. The Average Incremental Revenue plot visually represents the incremental revenue that was realized from the marketing action. It should be viewed across the customer deciles and after considering the cost of the promotional marketing treatment.



A customer is considered profitable only if the incremental gross revenue is higher than the contact cost. The first 60% of customers indicate positive net revenue increments, and are considered to be profitable. For example, the top 10% of customers have an expected revenue increment of \$58.28. A negative value indicates the negative effect of the promotion on individuals such as the sleeping dogs. So it is also important for the campaign or marketing action to target that class of customers for omission from promotional treatment. These results not only quantify expected revenue by customer segment, but also help identify the customer classes for whom incremental promotional costs only result in further income loss.

Next, examine the Average Revenue plot:



The Average Revenue plot provides a decile-by-decile breakdown of the average revenue for treatment and control groups. Customers are rank ordered the same way as in the Average Incremental Revenue plot, from the most profitable deciles on the left to the least profitable deciles on the right.

Next, examine the Selected Variables by NIV (net information value) table in the Incremental Response node Results.

Variable	Net Information Value	Rank Percentile ▲
CUST_TENURE	2056.967	3.125
MEMBERSHIP	477.9946	6.25
SPEND_CAT5	450.1365	9.375
SPEND_CAT2	357.7655	12.5
ORDER_ONLINE	336.199	15.625
ORDER_MAY	329.5455	18.75
SPEND_CAT4	297.8781	21.875
ITEM_TOTAL	297.8762	25
LAST_YEAR_SPEND	297.6157	28.125
SPEND_CAT3	291.5341	31.25
SPEND_CAT1	279.4105	34.375
ITEM_ONLINE	278.6421	37.5
FREQUENT_BUYER	276.9668	40.625
ORDER_TOTAL	230.2379	43.75
ORDER_APR	207.8365	46.875
REGENCY	136.1352	50

When you set the Prescreen Variables property of the Incremental Response node to Yes before running, the Selected Variables Table displays the top 50% of input variables ranked by NIV score. The NIV score indicates the variables that have the strongest correlation to model responses. The net information value is calculated as the difference in information values between the treatment group and control group for each input variable. The proportion of variables that is selected for inclusion in the table by NIV ranking can be specified in the Rank Percentage Cutoff property for the node.

References

- Larsen, K. 2010. "Net Lift Models: Optimizing the Impact of Your Marketing Efforts." SAS Course Notes, SAS Institute Inc., Cary, NC.
- Lo, Victory S.Y. 2002. "The True Lift Model — A Novel Data Mining Approach to Response Modeling in Database Marketing." *SIGKDD Explorations* 4 (2): 78–86.

Chapter 10

Advanced Topics

High-Performance Nodes: Overview	189
Data Source Roles	190

High-Performance Nodes: Overview

SAS Enterprise Miner High Performance (HP) data mining nodes can leverage cloud and grid environment resources to perform data mining tasks. This is useful with large data sets and data stores.

- The **HP Bayesian Network Classifier** node supports several Bayesian network structures that can be used to create effective predictive models based on conditional dependencies between variables.
- The **HP Cluster** node uses disjoint cluster analysis on the basis of Euclidean distances to perform observation clustering. Observation clustering can be used to segment databases.
- The **HP Data Partition** node partitions sampled input data into Train and Validation data partitions for modeling.
- The **HP Explore** node enables you to obtain descriptive information about a training data set. Statistics such as mean, standard deviation, minimum value, maximum value, and percentage of missing values are generated for each input variable and displayed in tabular and graphical form.
- The **HP Forest** node creates a predictive model called a forest. A forest consists of several decision trees that differ from each other in two ways. First, the training data for a tree is a sample without replacement from all available observations. Second, the input variables that are considered for splitting a node are randomly selected from all available inputs. In other respects, trees in a forest are trained like standard trees.
- The **HP GLM** (Generalized Linear Model) node is able to fit models for standard distributions in the exponential family. The HP GLM node fits zero-inflated Poisson and negative binomial models for count data.
- The **HP Impute** node replaces missing values for variables in data sets that are used for data mining.
- The **HP Neural** node creates multilayer neural networks that pass information from one layer to the next in order to map an input to a specific category or predicted value. The **HP Neural** node enables this mapping to take place in a distributed

computing environment, which enables you to build neural networks on massive data sets in a relatively short amount of time.

- The **HP Principal Components** node calculates eigenvalues and eigenvectors from the covariance matrix or the correlation matrix of input variables. Principal components are calculated from the eigenvectors and can be used as inputs for successor modeling nodes in the process flow diagram.
- The **HP Regression** node fits a linear regression or a logistic regression for an interval or binary target variable that is specified in the training data set. Linear regression attempts to predict the value of an interval target as a linear function of one or more independent inputs. Logistic regression attempts to predict the probability that a binary target will acquire the event of interest based on a specified link function of one or more independent inputs.
- The **HP SVM** node supports binary classification problems, including polynomial, radial basis function, and sigmoid nonlinear kernels. The **HP SVM** node does not perform multi-class problems or support vector regression.
- The **HP Text Miner** node enables you to build predictive models for a document collection in a distributed computing environment. Data is processed in two phases: text parsing and transformation. Text parsing processes textual data into a term-by-document frequency matrix. Transformations such as singular value decomposition (SVD) alter this matrix into a data set that is suitable for data mining purposes. A document collection of millions of documents and hundreds of thousands of terms can be represented in a compact and efficient form.
- The **HP Transform** node enables you to transform interval input and target variables. Interval input transformations are important for improving the fit of your model. Transformation of variables can be used to stabilize variances, remove nonlinearity, and correct non-normality in variables.
- The **HP Tree** node enables you to create and visualize a tree model and determine input variable importance in a high-performance data mining environment. The **HP Tree** node includes many of the tools and results found in the **Decision Tree** node. For interval targets, the **HP Tree** node supports the Variance and F Test splitting criteria, as well as the Average Square Error pruning criterion. For categorical targets, the **HP Tree** node supports the Entropy, Fast CHAID, Gini, and Chi-Square splitting criteria, as well as the Gini, Entropy, Average Square Error, and Misclassification rate pruning criteria. Also available are standard visualization and assessments plots, such as tree diagrams, treemaps, leaf statistics, subtree assessment plots, and node rules.
- The **HP Variable Selection** node reduces the number of modeling inputs by identifying the input variables that are not related to the target variable, and rejecting them. Although rejected variables are passed to subsequent nodes in the process flow, these variables are not used as model inputs by a successor modeling nodes.

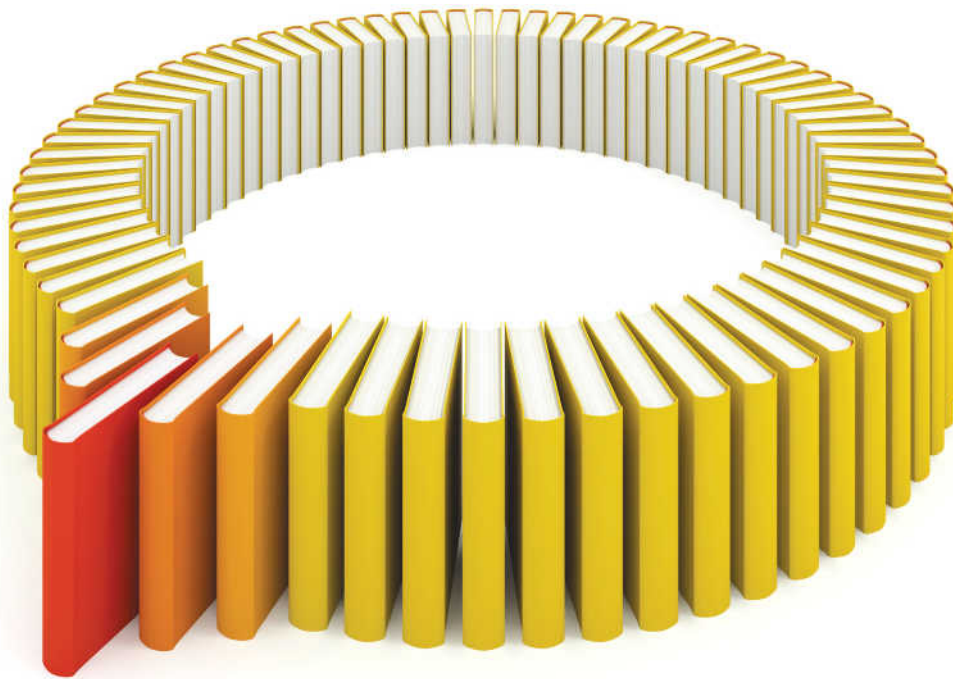
Data Source Roles

A data source appears in the Project Panel of the SAS Enterprise Miner window. When a data source is selected, properties are displayed in the Properties Panel. See Data Source Properties for detailed information.

When you select a data source, use the Role property to change the role of a data source. The role of a data source determines how the data source is used throughout the process flow diagram. Select one of the following values:

- **Raw** (Default) — is used as raw input to a node.
- **Train** — is used to fit initial models.
- **Validate** — is used by default for model comparison, if available. The validate data source is also used for fine-tuning the model. The **Decision Tree** and **Neural Network** nodes have the capacity of overfitting the TRAIN data set. To prevent these nodes from overfitting the train data set, the validate data set is automatically used to retreat to a simpler fit than the fit based on the train data alone. The validate data set can also be used by the **Regression** node for fine-tuning stepwise regression models.
- **Test** — is used to obtain a final, unbiased estimate of the generalization error of the model.
- **Score** — is used to score a new data set that might not contain the target.
- **Transaction** — is used in the **Time Series** and **Path Analysis** nodes. Transaction data is time-stamped data that is collected over time at no particular interval.

If you do not have training, validation, and test data sets, then you can create them with a successor **Data Partition** node.



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.



SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613

