

Data Mining

Lab Exercises, SS 2014

Predictive Modelling

Purpose

The purpose of this study is to learn how data mining methods / tools (SAS System/ SAS Enterprise Miner) can be used to solve predictive modeling tasks, in particular building classification models based on real life data.

The study will illustrate stages of data mining process, as guided by the SEMMA (or CRISP-DM) methodology, required to build a successful DM solution:

- Sample data
- Explore data in order to understand characteristics of data,
- Modify data in order to prepare them for modeling,
- Model, i.e., fit a predictive model to data,
- Assess predictive performance of the model.

Input data

Data set: `spam`

This data set shows attributes of email messages and a target variable set to 'yes' for spam message and 'no' otherwise. The task consists in building a model to classify email messages as spam or no-spam. The attributes of email messages used here to train the classifier are based on the SpamAssassin (<http://spamassassin.apache.org>) project

Data set: `copper_wire`

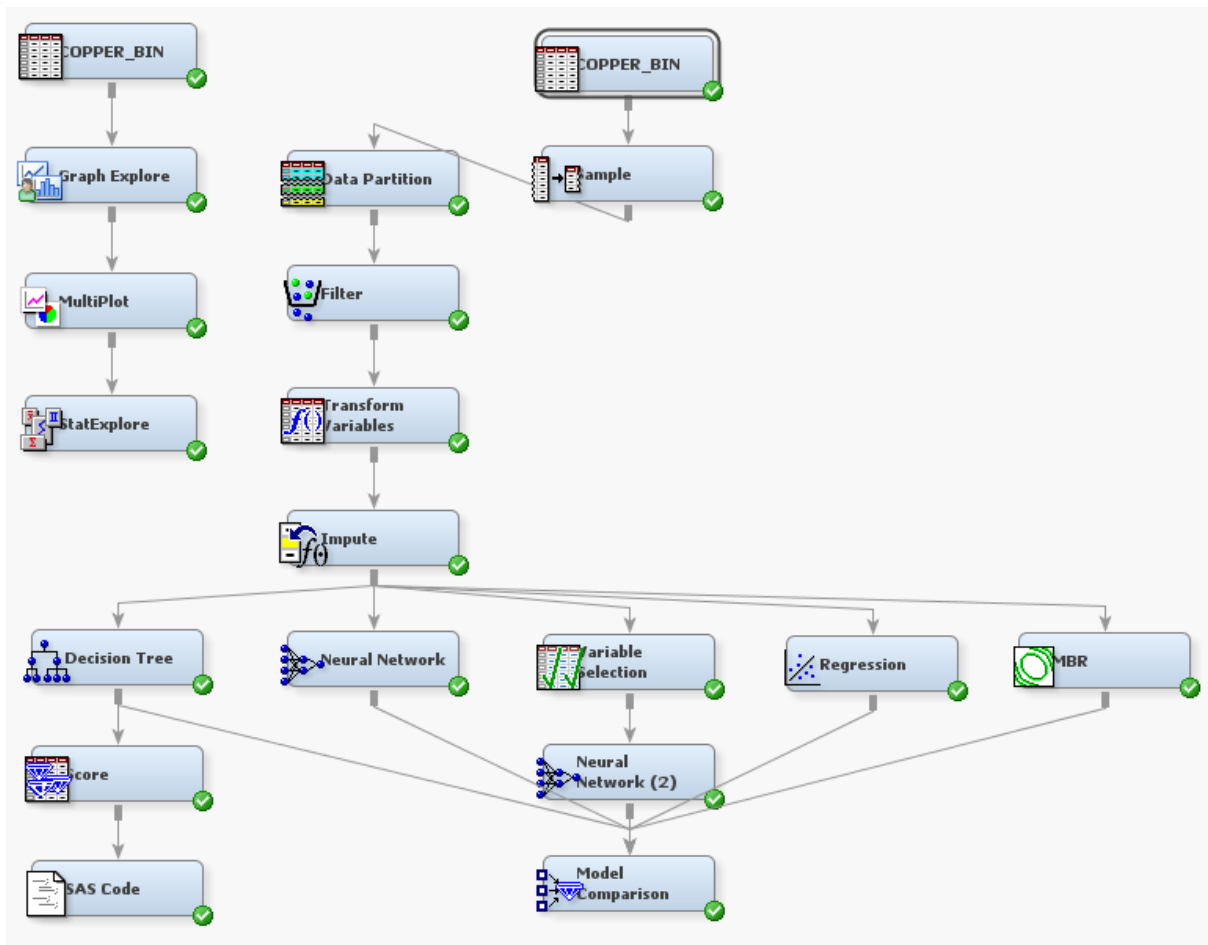
This data set is related to quality of manufacturing process of copper wire. The variable `quality` represents quality of a section (roll) of copper wire produced. Other variables represent data from the production process monitoring system (such as temperatures, levels of various impurities in copper etc.). The values of `quality` ≤ 6 denote section of good quality; quality levels above 6 denote poor quality.

Tasks – overview

The task involves

1. building a model for prediction of the target (i.e., quality (good / poor) of sections of copper wire based on parameters of production process, or spam / no-spam classification of email),
2. estimation of predictive performance of the model for new data,
3. fine-tuning the classifier.

Predictive models will be built in SAS Enterprise Miner. A sample Enterprise Miner diagram is shown in the following figure:



Output (deliverables):

Spam data

- Classification result for every email message in the spam data set,
- Classification error rates:
 - “no→yes” (good mail classified as spam) and
 - “yes→no” (spam classified as no spam).

The purpose of fine-tuning the classifier is to:

- minimize the “yes→no” error rate,
- while ensuring that “no→yes” error rate is < 1%.

Copper wire data

- Classification result for every manufactured item,
- Classification error rates:
 - “good→poor” (good quality misclassified as poor quality) and
 - “poor→good” (poor quality misclassified as good quality).

The purpose of fine-tuning the classifier is to:

- Maximize the total profit related to classification decisions calculated per production batch.

We assume the following cost/profit model associated with classifier's decisions:

Decision	Profit incurred
good → good	+10
good → poor	+5
poor → good	-100
poor → poor	+5

Report

Summarize predictive performance of your models as a function of:

- Model type
 - o logistic regression,
 - o tree,
 - o neural network,
 - o k-nn classifier,
 - o model boosting, etc.
- Feature selection method
 - o none,
 - o Variable selection node,
 - o classifier specific feature selection (e.g. Forward selection in Regression model).
- Cost of misclassification events (see Target profiling in Tutorial below).
- Specific settings of classification models (e.g., settings related to simplicity / complexity of models, such as the number of neurons in the hidden layer).
- Settings of metamodels / metalearning (e.g., number of individual trees in the *bagging* or *boosting* collection of models; settings of the Ensemble aggregate model).

Hints

Detailed procedure how to build a SEMMA diagram in Enterprise Miner is given in the Script **DATA MINING AND DATA WAREHOUSING – PRACTICAL GUIDE**, part I.

Most important part of this document are also given in the next section **Detailed description of SEMMA steps in Enterprise Miner**.

(The following part of this document is related to assessment of copper wire quality)

Detailed description of SEMMA steps in Enterprise Miner:

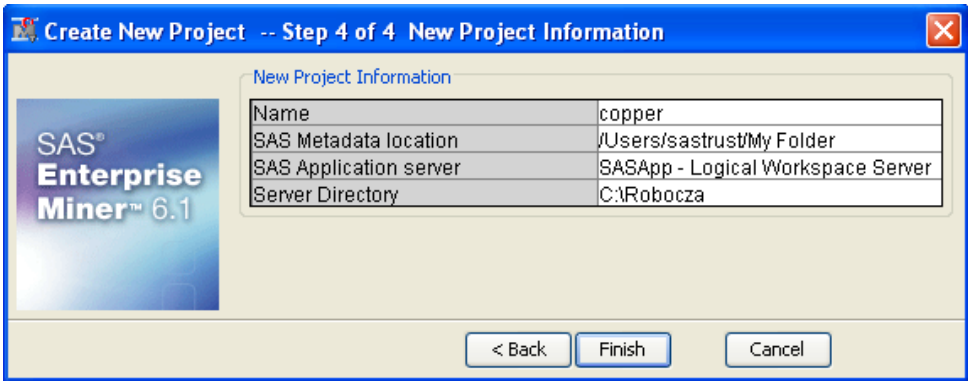
In this tutorial we describe the consecutive steps you need to follow to build a SEMMA diagram, such as:

- Preparatory tasks,
- Sample data,
- Explore data,
- Modify data,
- Model,
- Assessment of model performance,
- Scoring new data.

Preparatory tasks

Enterprise Miner is a project – oriented tool, so the first task is to create a new project for this study and to connect the source data to the project. Since all datasets are referenced in SAS through the means of libraries (i.e., LIBREFs which are references to folders / directories in the host operating system), a new library has to be created in the project to point to the location of the copper_bin source file.

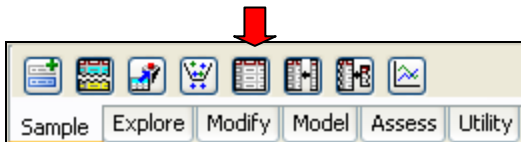
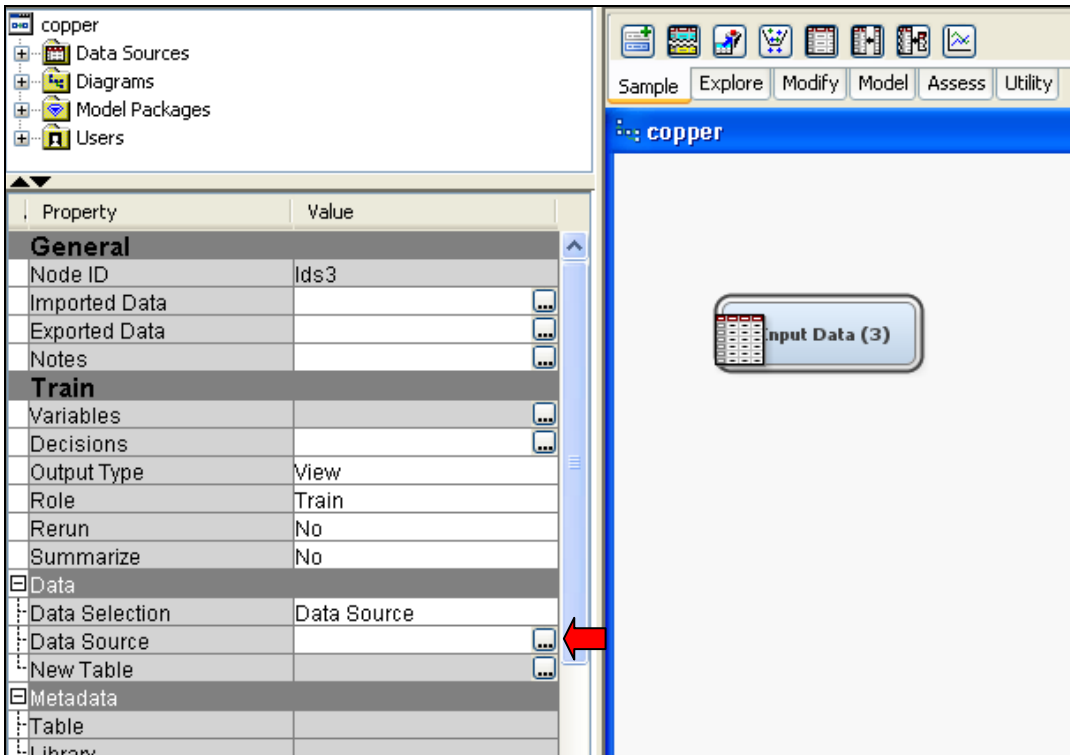
These tasks are realized in the following steps:

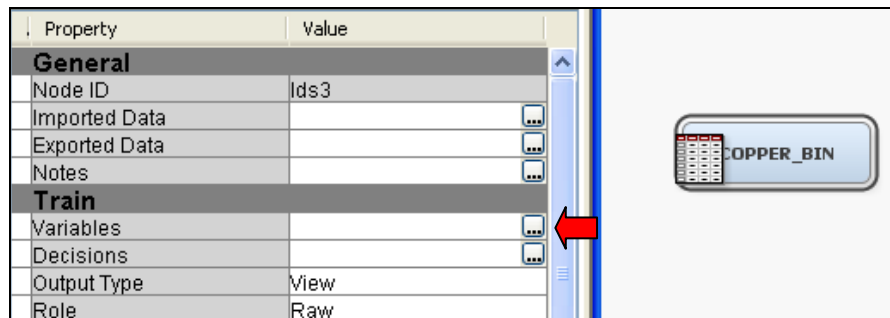
Task	Description
1	<p>Create the new Project</p> <p>The project is created using the Project Wizard launched by the File-New-Project menu. In the wizard, the following three elements are specified:</p> <ul style="list-style-type: none">• SAS server for the project• Project name and SAS server directory• SAS Folder location where project metadata will be stored <p>For the first and the third point the default values can be left. The project name and server directory in this example are: copper and C:\robocza, which gives the final setting for the new project as shown below.</p> 
2	<p>Connect the source data to the project</p> <ol style="list-style-type: none">1. Create the new Library The source data used in this example is the copper_bin dataset. First, a new library has to be created to point to the location (OS folder) of this dataset – the library is created using the Library Wizard launched by the File-New-Library menu.

	<p>2. Create the new Data Source</p> <p>data source is created using the Data Source Wizard (File-New-Data Source menu). In the wizard, the following decisions are made:</p> <ul style="list-style-type: none"> • A SAS table is selected that becomes the data source (in this example the dataset copper_bin is used, found in the library created in the previous step) • The column metadata are defined (such as the roles in the model and measurement levels of variables). In this step default values of metadata can be left, as proposed by the Metadata Advisor. Although the metadata could be corrected here, this will be done and explained in detail in the Sample data step. • All other settings should be accepted as proposed by the wizard.
3	<p>Create the new Diagram</p> <p>Using the File-New-Diagram menu, the new diagram called copper is created. The diagram will be populated with Enterprise Miner nodes to form the process flow.</p>

Sample data

In this step, we start building the process flow by connecting to and possibly sampling the data source, providing proper metadata for the variables and partitioning the data into training, validation and testing layers. These tasks are done in the following steps.

Task	Description
1	<p>Add to the diagram the Input Data node</p> <ol style="list-style-type: none"> 1. Drag and drop onto the diagram pane the Input Data node found on the Sample toolbar.  <ol style="list-style-type: none"> 2. Next, select the Input Data node and specify the data source that the node connects to (using the Data Source editor in the node's Property pane). Select the copper_bin data source. 
2	<p>Setup metadata for the data source</p> <p>Prior to building a predictive model, variables must be properly described by metadata to indicate (a) the intended role of a variable in the process (target or predictor variable), and (b) the measurement level of a variable (as quantitative or class variable).</p> <p>Metadata pertaining to variables are setup using the Variables editor (available in the Input Data property pane), as shown below.</p>



The role and measurement level of the copper_bin variables should be specified as follows. The qbin variable is used as the *binary target*, the part number and original value of quality variables are *rejected* from model building and all other variables are used as predictors with the *interval* measurement level (which indicates a quantitative variable).

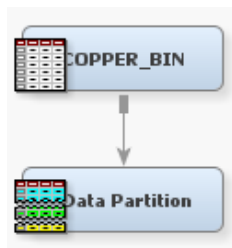
Name	Role	Level
level_o2	Input	Interval
part_no	Rejected	Ordinal
qbin	Target	Binary
quality	Rejected	Ordinal
size_max	Input	Interval
size_min	Input	Interval
sum_eddy_f1	Input	Interval
sum_eddy_f2	Input	Interval
sum_eddy_f3	Input	Interval
sum_ferro_f1	Input	Interval
sum_ferro_f2	Input	Interval
sum_ferro_f3	Input	Interval
temp	Input	Interval
vel_max	Input	Interval
vel_mean	Input	Interval
vel_min	Input	Interval

- 3 Partition data

Prior to training predictive models, input data must be divided into *training*, *validation* and *test* partitions. The training partition is used to fit a predictive model to the data, the validation partition is used to fine-tune parameters of the model in order to avoid model *overfitting*. Model fine-tuning consists in modification of such parameters of models as depth of a decision tree or the number of perceptrons in the hidden layer of a neural net, etc. The test partition can then be used to estimate the expected prediction error for new data.

The data is split by connecting the Data Partition node (available in the Sample toolbox) as shown below.

Data Set Allocations property of the Data Partition node provides the size of partitions (as percentages of original data); default size can be left as is. The Exported Data property provides names of the training, validation and test datasets produced by the node.



Property	Value
General	
Node ID	Part
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Output Type	Data
Partitioning Method	Default
Random Seed	12345
Data Set Allocations	
Training	40.0
Validation	30.0
Test	30.0

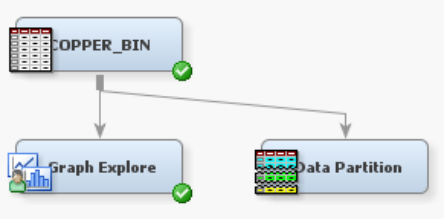
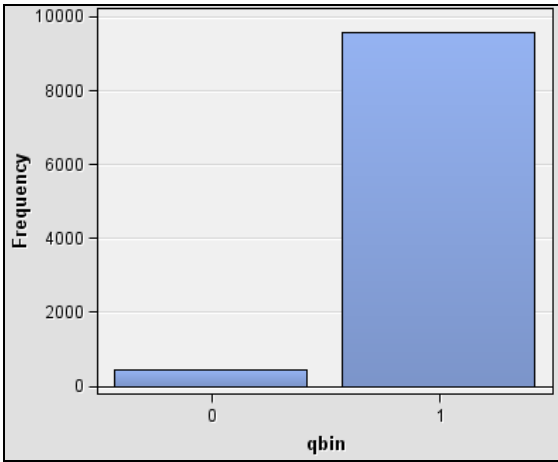
Explore data

The purpose of this step is to discover important characteristics of distributions of variables as well as inconsistencies, errors and outliers in data. Based on these findings, required data modifications can be designed in order to clean the data, impute missing values or transform variables to make some distributions less skewed or more normal-like (e.g., by the log transformation). These steps are mandatory to ensure robust predictive models.

In this guide we focus on some Enterprise Miner tools for graphical exploration and for statistical analysis of data, such as:

- Graph Explore,
- MultiPlot,
- StatExplore.

These tools can be used as shown and explained in the following steps.

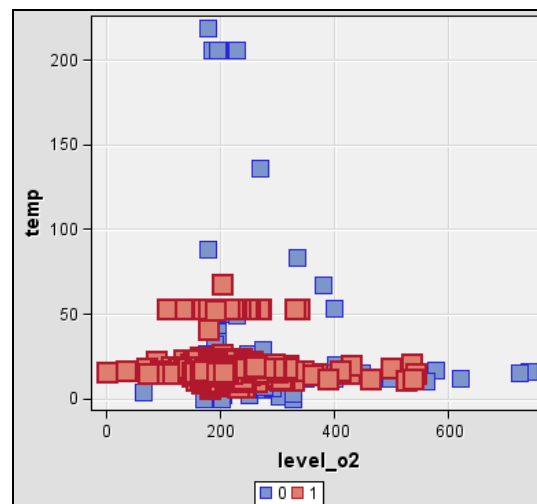
Task	Description						
1	<p>Perform graphical exploration of data</p> <p>Using the Graph Explore tool</p>  <ol style="list-style-type: none">1. Drag the Graph Explore icon into the diagram and connect to the Input Data node (the tool can be found in the Explore toolbar).2. Run the tool (right click – Run menu).3. Open Results window. Using View-Plot menu of Results window open the Chart wizard to configure a relevant graphical summary of data. <p>The first observation we make is that the data are heavily <i>unbalanced</i> in terms of distribution of the target variable. The proportion of observations is about 95% vs 5% of good quality (qbin=1) and poor quality (qbin=0) items, respectively.</p>  <table border="1"><caption>Frequency distribution of qbin</caption><thead><tr><th>qbin</th><th>Frequency</th></tr></thead><tbody><tr><td>0</td><td>~500</td></tr><tr><td>1</td><td>~9500</td></tr></tbody></table> <p>Building predictive models based on such data is generally difficult, as machine</p>	qbin	Frequency	0	~500	1	~9500
qbin	Frequency						
0	~500						
1	~9500						

learning algorithms tend to bias the models towards accurate prediction of the majority class (qbin=1), with poor predictive performance of the rare class (qbin=0). To avoid this, several techniques will be discussed later in this guide, including oversampling representatives of the rare class.

The Graph Explore tool provides features for graphical analysis of distributions of predictors and mutual relationships of predictors. These include several types of statistical plots (density plot, boxplots etc.) as well as many types of interactive explanatory plots (such as 2D and 3D plot scatter plots, contour plots etc.)

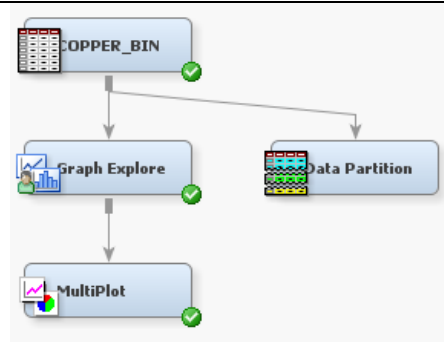
An example of a Scatter Chart is shown below, illustrating the relationship between the variables level_o2 and temp, grouped by the target variable (qbin).

It can be observed from this plot that the upper tails of distributions of these two predictors correspond to poor quality items (qbin=0). This is an important finding, also confirmed for other predictors, as counter - outlier techniques usually tend to remove these tails. In our study, we will use these techniques cautiously so as not to lose some significant portion of the rare class representatives from the training data.



It should be noticed that all charts created in the Graph Explore tool are interactive, i.e., by selecting an element on the plot (such as a point or group of points in scatter plot, a bin in the histogram etc.), corresponding observations in the tabular view of raw data are also selected. This allows for easy and efficient analysis of observations which contribute to untypical values in distributions of some variables.

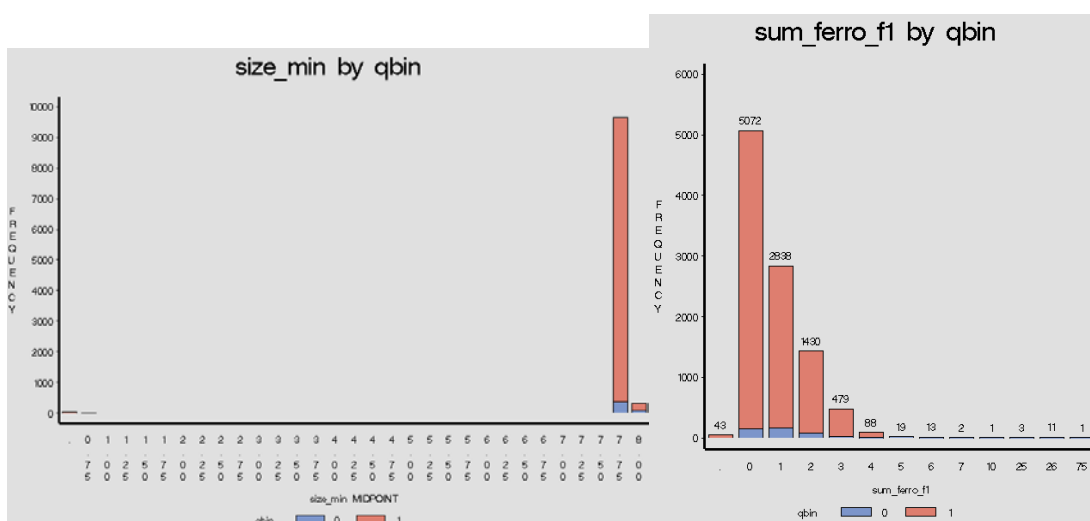
Graphical exploration using the Multiplot tool



To use this tool, connect its icon as shown in the diagram and Run the analysis (right click menu).

In the Results window, maximize the Train Graphs window to start quick, interactive inspection of distributions of subsequent variables. In this way we can efficiently scan through large data volumes to reveal illegal values / errors /outliers. An example of such analysis is shown below, where very small values of the size_min variable are immediately detected in the histogram below. Such erroneous observations will have to be filtered off in the Modify step.

We also observe that some variables have very skewed distributions (such as sum_ferro_f1, shown below). These variables can be later transformed to make the distribution more symmetric, which improves performance of regression or discriminant analysis methods.



Obs	NAME	NMISS	N	MIN	MAX	MEAN	STD	SKEWNESS	KURTOSIS
1	level_o2	62	9938	-10584.00	759.00	197.017	113.476	-86.1561	8201.64
2	size_max	13	9987	3.11	8.31	8.119	0.087	-37.8812	2174.95
3	size_min	13	9987	0.85	8.00	7.786	0.084	-56.2436	4652.24
4	sum_eddy_f1	43	9957	0.00	100.00	3.432	5.070	9.3763	118.97
5	sum_eddy_f2	43	9957	0.00	76.00	0.322	1.612	20.8708	658.93
6	sum_eddy_f3	43	9957	0.00	36.00	0.114	0.859	26.9124	881.06
7	sum_ferro_f1	43	9957	0.00	75.00	0.816	1.544	17.4251	630.02
8	sum_ferro_f2	43	9957	0.00	27.00	0.115	0.503	28.5527	1419.65
9	sum_ferro_f3	43	9957	0.00	25.00	0.055	0.349	38.7714	2647.28
10	temp	47	9953	0.00	219.00	19.526	11.094	5.2395	56.70
11	vel_max	62	9938	5.50	9999.90	12.488	100.196	99.6888	9937.90
12	vel_mean	62	9938	5.43	821.29	11.543	8.128	99.5300	9916.85
13	vel_min	62	9938	4.60	11.60	11.432	0.403	-8.8106	95.41

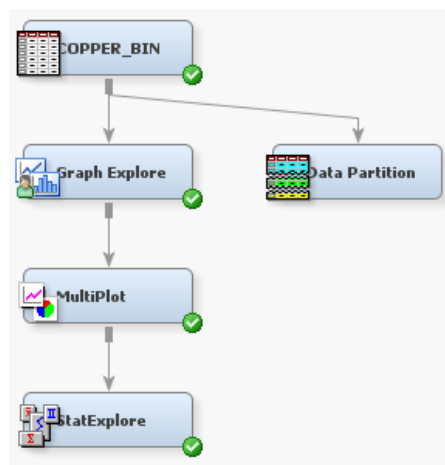
The following observations can be made:

- Some observations have very small or very high MIN or MAX values as compared to the mean value (e.g., level_o2 with erroneous MIN value, or vel_max and vel_mean with erroneous MAX value),
- Some observations have very large standard deviation as compared with the mean,
- All variables contain missing values (the number of missing values is given in the NMISS column).

Based on these observations, required data filtering and transformation rules will be implemented in the next SEMMA step.

2 Perform statistical analysis of data – using the StatExplore tool

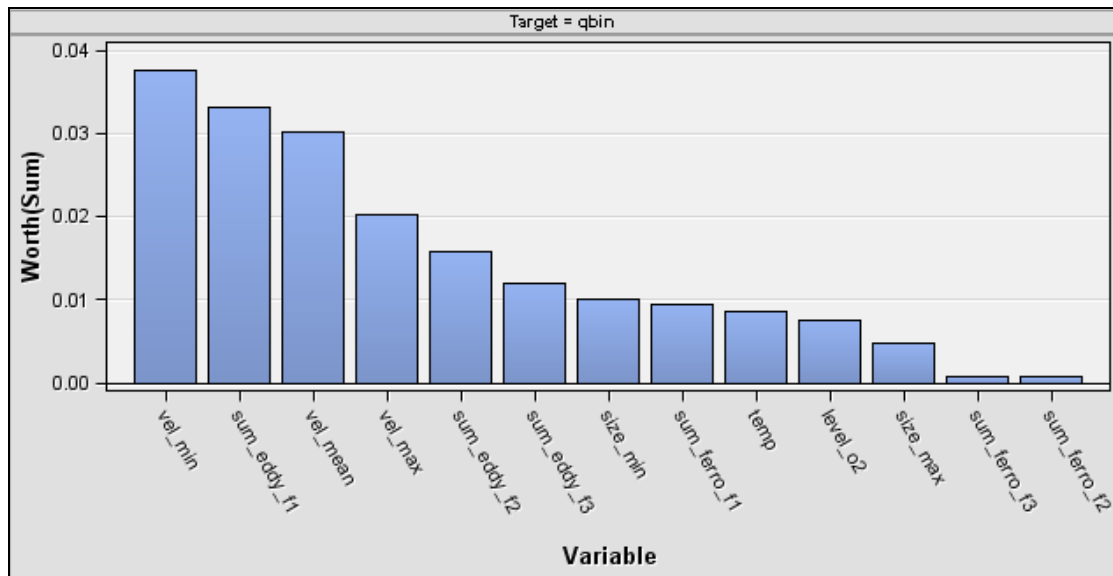
The StatExplore tool allows to examine distributions of variables and analyze association of inputs with the (interval or class) target. Based on these analyses, variable selection can be realized in order to reduce dimensionality of predictive (or clustering) models.



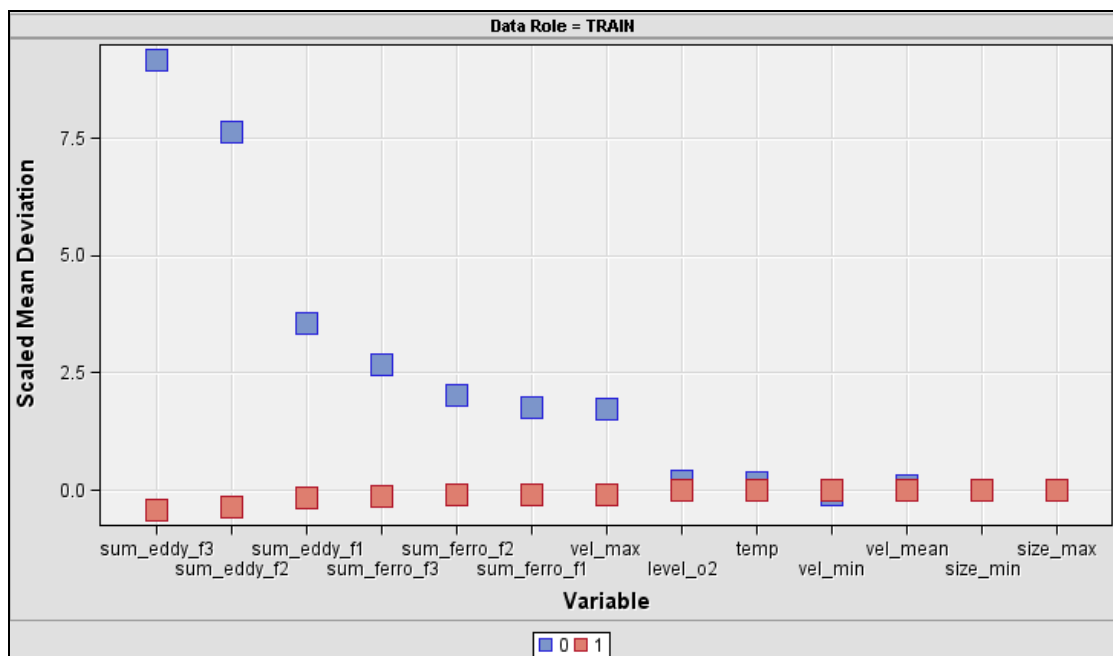
The StatExplore tool should be connected as shown in the diagram and executed (right click Run menu).

In the Results window, the Variable Worth and Interval Variables : qbin charts are available (through the View-Plot menu of the Results window).

The Variable Worth chart (shown below) ranks variables by their importance for target prediction, as calculated by the decision tree algorithm.



Association of variables with the target is also shown in the following Scaled Mean Deviation plot. Here the (scaled) means in the “target=0” and “target=1” groups are compared with the overall mean of the variable (represented by the level of zero). This plot corresponds with the per-group descriptive statistics shown next. For instance, the variable size_max realizes roughly the same mean value in the groups “qbin=1”, “qbin=0” and overall, hence its position on the right of the plot. On the other hand, the variable sum_eddy_f1 realizes similar mean value in the group “qbin=1” and overall, with much bigger mean in the “qbin=0” groups, which is clearly seen in the Mean Deviation chart.



Data Role=TRAIN Variable=size_max							
Target	Target Level	Median	Missing	Non Missing	Minimum	Maximum	Mean
OVERALL		8.12	13	9987	3.11	8.31	8.119098
qbin	0	8.11	7	442	8	8.31	8.11414
qbin	1	8.12	6	9545	3.11	8.28	8.119327
Data Role=TRAIN Variable=sum_eddy_fl							
Target	Target Level	Median	Missing	Non Missing	Minimum	Maximum	Mean
OVERALL		3	43	9957	0	100	3.43246
qbin	0	4	0	449	0	100	15.61693
qbin	1	3	43	9508	0	20	2.857068

Summarizing, the following conclusions can be drawn from the Explore step:

- The data has heavily imbalanced distribution of target with roughly 5% of the rare class (poor quality items),
- Some predictors have clearly erroneous observations (such as the negative value of the level_o2), most of predictors have skewed distributions,
- Most of predictors include outlying observations, these however should be removed with caution (i.e., only when the value of predictor is beyond physical range of the variable), as outliers generally correspond to the rare class (poor quality) observations.

These issues will be tackled in the following Modify and Model steps.

Modify data

The purpose of this step is to:

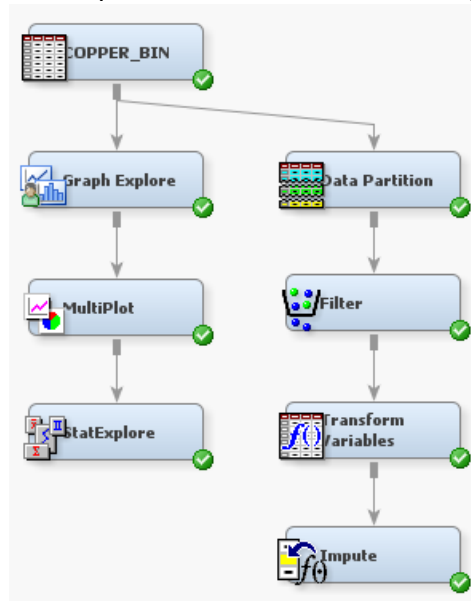
- Remove observations with wrong or outlying values of input variables,
- Transform variables to reduce skewness in distribution,
- Impute missing values.

Transformations of data to reduce skew in distribution bring variables closer to the normal distribution, which improves performance of predictive models based on the assumption on normality of features (such as the LDA). Filling in missing data (e.g., based on values in similar observations or using more sophisticated approach, such as prediction based methods) may improve performance of regression methods or neural classifiers which otherwise ignore observations in which missing values occur. (Note that decision tree algorithms accept missing values as legitimate values of predictors).

In this guide, we demonstrate several features of Enterprise Miner used to modify data, such as features implemented in the following nodes:









- Filter,
- Transform Variables,
- Impose.

The nodes should be connected to the Input Data as shown in the diagram below.



Data transformations implemented with these nodes are outlined in the following procedure.




Task	Description
1	<p>Filter observations containing erroneous inputs</p> <ol style="list-style-type: none">1. Select the Filer node, which activates the node's Property window as shown below.2. In the Property window, setup conditions for filtering out observations, based on values of class variables and interval variables. In this example, we change the Default Filtering Method for both class and interval variables to None (as these apply to all variables), and define specific filtering conditions for individual variables.

Property	Value
General	
Node ID	Filter
Imported Data	
Exported Data	
Notes	
Train	
Export Table	Filtered
Tables to Filter	Training Data
Class Variables	
Class Variables	
Default Filtering Method	Rare Values (Percentage) 
Keep Missing Values	Yes
Normalized Values	Yes
Minimum Frequency Cutoff	1
Minimum Cutoff for Percent	0.01
Maximum Number of Levels	25
Interval Variables	
Interval Variables	
Default Filtering Method	Standard Deviations from the Mean 
Keep Missing Values	Yes
Tuning Parameters	

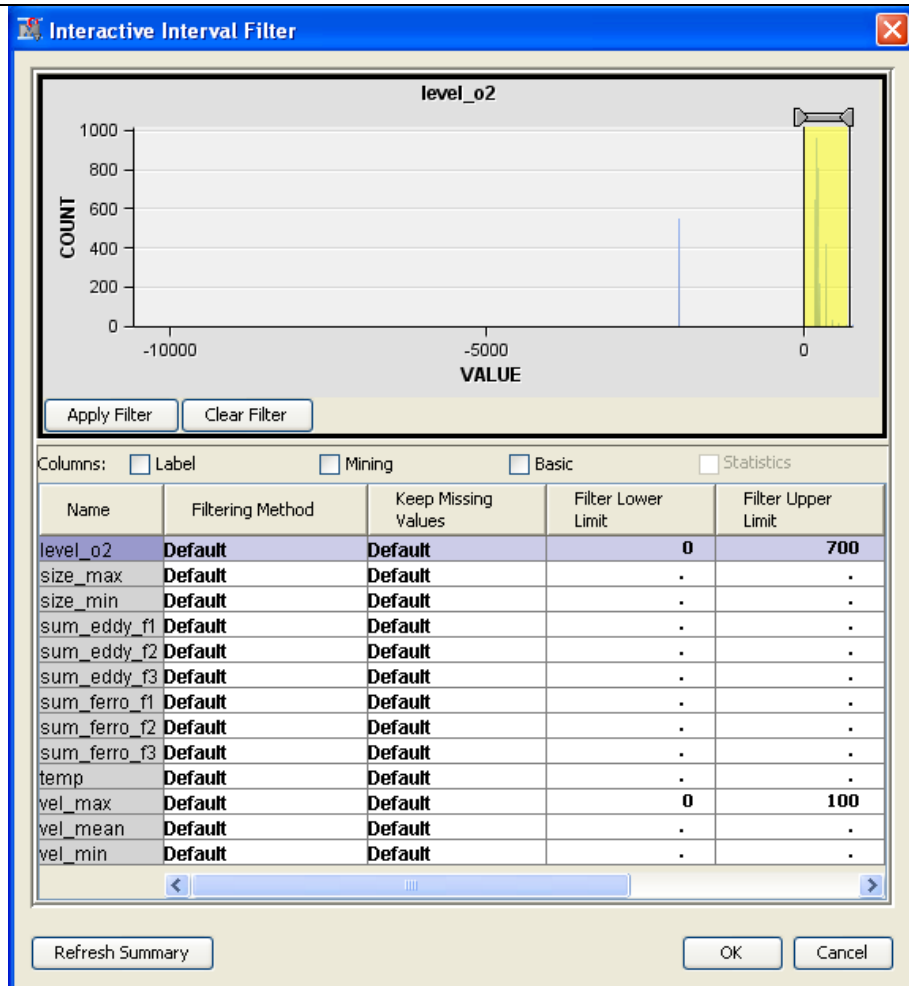
The tool offers several filtering methods, e.g., Standard Deviations from the Mean for interval variables. This method removes outliers, where outliers are defined by the 3 standard deviations from the mean condition.

In this example, we switch the Default Filtering Method to None. The default Standard Deviations from the Mean method would remove too many rare case (qbin=0) observations, as these observations are generally overrepresented in the far ends of distributions of predictors, as shown in the Explore stage. Thus we manually setup filtering conditions to remove wrong and preserve feasible albeit outlying data.

We do this using the editor of filtering conditions for interval inputs (the editor is invoked as indicated by the arrow below).

Interval Variables	
Interval Variables	
Default Filtering Method	None 
Keep Missing Values	Yes
Tuning Parameters	

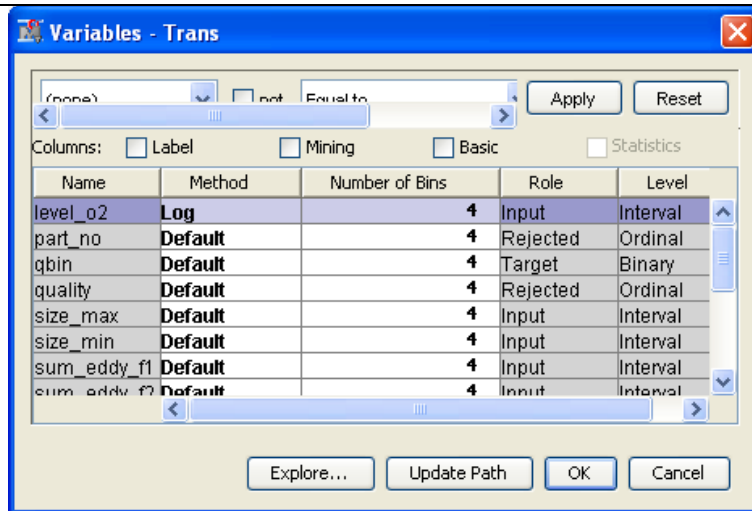
Next, we specify lower and upper limits for the inputs: level_o2 and vel_max as shown below (this removes observations with erroneous values in these variables).



- 2 Transform input variables to reduce skewness in distribution
 1. Select the Transform Variables node, this activates the node's Property window as shown below.
 2. Open the transformations editor (as indicated by the arrow).

Property	Value
General	
Node ID	Trans
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Formulas	
Interactions	
SAS Code	
Default Methods	
Interval Inputs	None
Interval Targets	None
Class Inputs	None
Class Targets	None
Treat Missing as Level	No

In this example, we will apply the log transformation to the level_o2 input, as shown in the Method column of the transformations editor.



- 3 Impute missing values
1. Select the Impute node, which activates the node's Property window as shown below.
 2. Change the imputation method for class variables to None and for interval variables to Tree Surrogate, as shown below.

Property	Value
General	
Node ID	Impt
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Non Missing Variables	No
Missing Cutoff	50.0
Class Variables	
Default Input Method	None
Default Target Method	None
Normalize Values	Yes
Interval Variables	
Default Input Method	Tree Surrogate
Default Target Method	None

There are several methods of missing value imputation, such as simple methods which fill in the mean or median of the variable's distribution, or more sophisticated methods such as the ones based on robust M-estimators of distribution location parameter, etc.

In this example, we use the tree based imputation method which calculates the missing value as predicted by the remaining inputs (for the purpose of this analysis, the variable with missing values is regarded as the target).

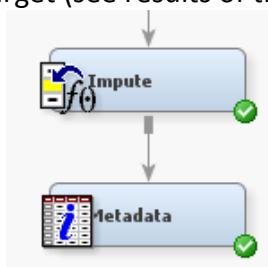
We note that data transformation and missing value imputation nodes in Enterprise Miner do not override the original variables. Instead they add new variables to the dataset, with names based on the type of transformation, as shown in the metadata listing below.

E.g., the level_o2 variable was first logged to create the new LOG_level_o2 variable, which was later transformed with the Impute node to create the IMP_LOG_level_o2 variable. The 'old' variables are

rejected from analysis and the modified variables labeled as inputs, i.e., will be used as predictors (this is done by the Role metadata column).

Name /	Hidden	Role	Level
IMP_LOG_level_o2	N	Input	Interval
IMP_size_max	N	Input	Interval
IMP_size_min	N	Input	Interval
IMP_sum_eddy_f1	N	Input	Interval
IMP_sum_eddy_f2	N	Input	Interval
IMP_sum_eddy_f3	N	Input	Interval
IMP_sum_ferro_f1	N	Input	Interval
IMP_sum_ferro_f2	N	Input	Interval
IMP_sum_ferro_f3	N	Input	Interval
IMP_temp	N	Input	Interval
IMP_vel_max	N	Input	Interval
IMP_vel_mean	N	Input	Interval
IMP_vel_min	N	Input	Interval
LOG_level_o2	Y	Rejected	Interval
WARN_	N	Assessment	Nominal
level_o2	Y	Rejected	Interval
part_no	N	Rejected	Interval
qbin	N	Target	Binary
quality	N	Rejected	Ordinal
size_max	Y	Rejected	Interval
size_min	Y	Rejected	Interval
sum_eddy_f1	Y	Rejected	Interval
sum_eddy_f2	Y	Rejected	Nominal
sum_eddy_f3	Y	Rejected	Nominal
sum_ferro_f1	Y	Rejected	Nominal
sum_ferro_f2	Y	Rejected	Nominal
sum_ferro_f3	Y	Rejected	Nominal
temp	Y	Rejected	Interval
vel_max	Y	Rejected	Interval
vel_mean	Y	Rejected	Interval
vel_min	Y	Rejected	Interval

The metadata can be inspected and modified if necessary using the Metadata node as shown in the diagram below. This node could be used for manual feature selection, i.e., to include or reject variables based on association of inputs with the target (see results of the StatExplore node).



In this example we use the Metadata node only for illustration on how transformation nodes work and hence this node will not be shown in the following diagrams.

Model

In this step, we will build different predictive models to estimate the target, i.e., to classify the manufactured items as good or poor quality. We will try:

- Decision trees,
- Neural networks,
- Logistic regression,
- Memory based reasoning method (i.e., the nonparametric k nearest neighbours classifier).

We will also demonstrate how feature selection can be realized in Enterprise Miner. Strictly speaking, this step is not crucial in our study, since the number of variables is relatively small. However in many real life problems with hundreds or more features, feature selection reducing dimensionality of data is mandatory, since many noisy features lead to deterioration in model performance and increase processing time and memory requirements.

Another important issue to consider prior to fitting a predictive model is definition of the criterion for model selection / comparison. By default, predictive models attempt to minimize the overall misclassification rate. This does not necessarily guarantee the optimal performance, especially if the consequences (costs) of the $0 \rightarrow 1$ and $1 \rightarrow 0$ misclassification decisions are different. In such studies, minimization of misclassification costs (or alternatively maximization of profits) might be the right criterion for model selection. This issue is discussed in the next section on Target profiling.

In this study, we also have to consider the problem of *highly imbalanced* data (as the poor quality class is represented by only ca 5% of observations). Predictive models usually demonstrate poor performance for the rare class. The reasons for this and the methods to tackle the problem are discussed in the Working with imbalanced data section.

Target profiling

The target profile is used to specify costs of $0 \rightarrow 1$ and $1 \rightarrow 0$ misclassifications. Target profiles are also applied in non binary classification problems, when costs of $c_i \rightarrow c_j$ decisions are provided in the form of the cost matrix, with c_i, c_j denoting the class labels.

Once defined for the target variable, the target profile is used by the model fitting algorithms to attempt to minimize the misclassification costs or maximize the overall profit.

The target profile is setup using the procedure outlined below.

Task	Description
1	<p>Associate the Target profile with the qbin variable</p> <ol style="list-style-type: none">1. Select the Input Data node, which activates the node's Property window.2. In the Property window, open the Decisions editor as shown below. In the Decision Processing window, click the Build button, which creates the default target profile for the qbin variable.

Property	Value
General	
Node ID	Ids
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Decisions	...
Output Type	View
Role	Raw
Rerun	No
Summarize	No



3. The target *event level* is selected based on the target level order. The target event level is later used to define the meaning of *sensitivity* of the classifier; also the (logit of) probability modelled by the logistic regression is related to the target event. In our case, we accept the event level of 1 (which translates into the decision of the classifier that an item is of good quality; also sensitivity of the model, reported later in the “Assessment of performance of the models” section will denote probability of correct recognition of the good quality item).
(If event level of 0 makes interpretation of classifier’s decisions easier, the event level can be changed by setting the Target level order to Ascending in the metadata associated with the target variable. Generally, the event level is selected as the first value in the list of sorted values of the target).

- 2 Define decision weights
- In Decision Weights editor (shown in the following picture), we specify costs (or profits) associated with particular decisions by the classifier, where:
- DECISION1 means classify as good quality (qbin=1),
 - DECISION2 means classify as poor quality (qbin=0),
- as indicated by the Decision tab (see the second picture).
- In the weights (or profits) matrix shown below, we reflect the following scenario (this scenario is based on the actual business perspective as seen by the copper company, albeit the values of profits/costs are fictitious):
- If an actually good quality item (level=1) is classified as good quality (DECISION1), then the company makes the profit of 10.
 - If an actually poor quality item (level=0) is classified as good quality (DECISION1),

then the company makes the profit of -100 (i.e., makes a loss, due to having to pay high warranty costs to its customer, exceeding prior profits).

- If an item is classified as poor quality (DECISION2), then the company sells the product as second quality, thus cheaper, and makes the profit of 5, irrespective of the actual quality.

Decision Processing - COPPER_BIN

Targets Prior Probabilities Decisions Decision Weights

Select a decision function:

☒ Maximize ☐ Minimize

Enter weight values for the decisions.

Level	DECISION1	DECISION2
1	10.0	5.0
0	-100.0	5.0

OK Cancel

We select the *maximize* decision function which is consistent with our interpretation of the values entered as profits (if costs were entered, then the *minimize* decision function would have to be selected).

Decision Processing - COPPER_BIN

Targets Prior Probabilities Decisions Decision Weights

Do you want to use the decisions?

☒ Yes ☐ No Default with Inverse Prior Weights

Decision Name	Label	Cost Variable	Constant
DECISION1	1	< None >	0.0
DECISION2	0	< None >	0.0

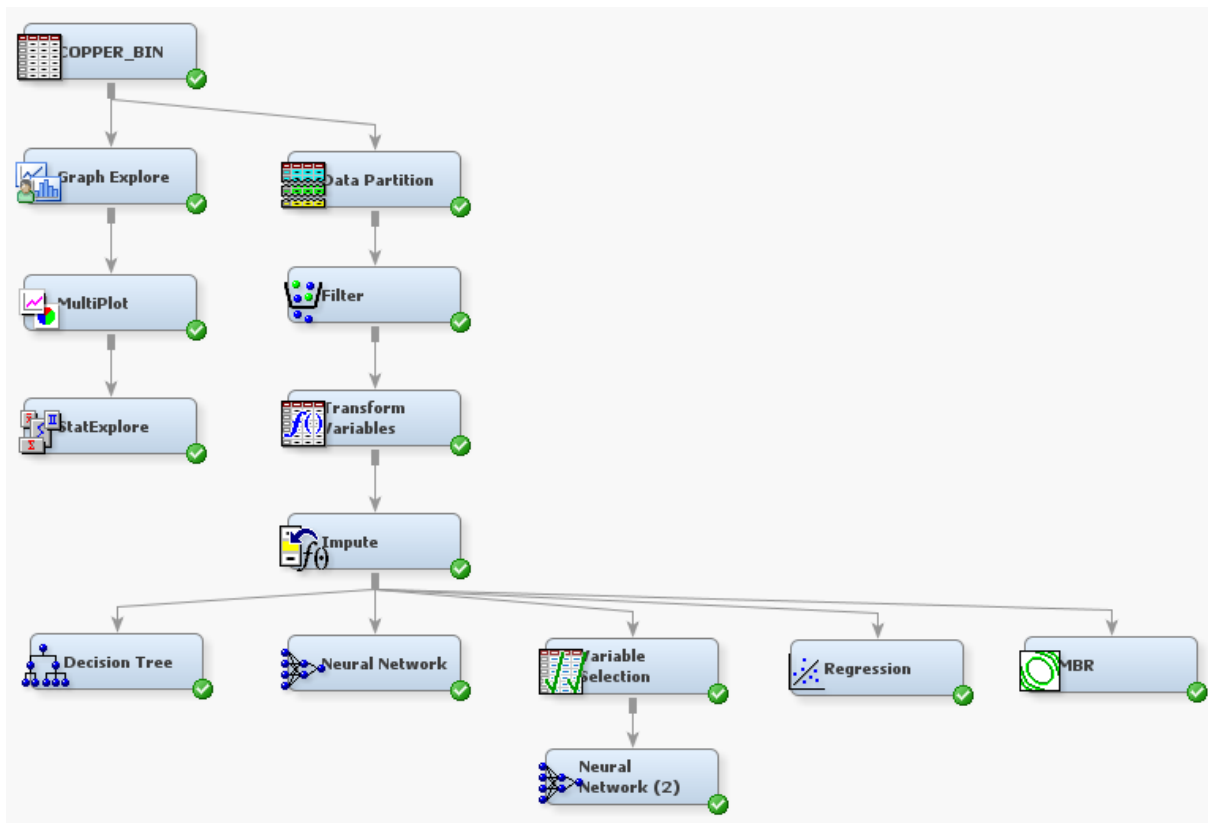
Once the target profile and the weights matrix is associated with the target, subsequent builds of predictive models will attempt to maximize the profit (or minimize costs) as the criterion for classifier selection.

Using predictive modelling nodes

We will build and compare five classifiers:

- Decision tree,
- Neural network (multilayer perceptron),
- Neural network preceded by a variable selection node,
- Logistic regression using forward feature selection method,
- Memory Based Reasoning (MBR), which is a simple nonparametric nearest neighbours classifier.

To build these classifiers, predictive modelling nodes should be added to the diagram as shown below.



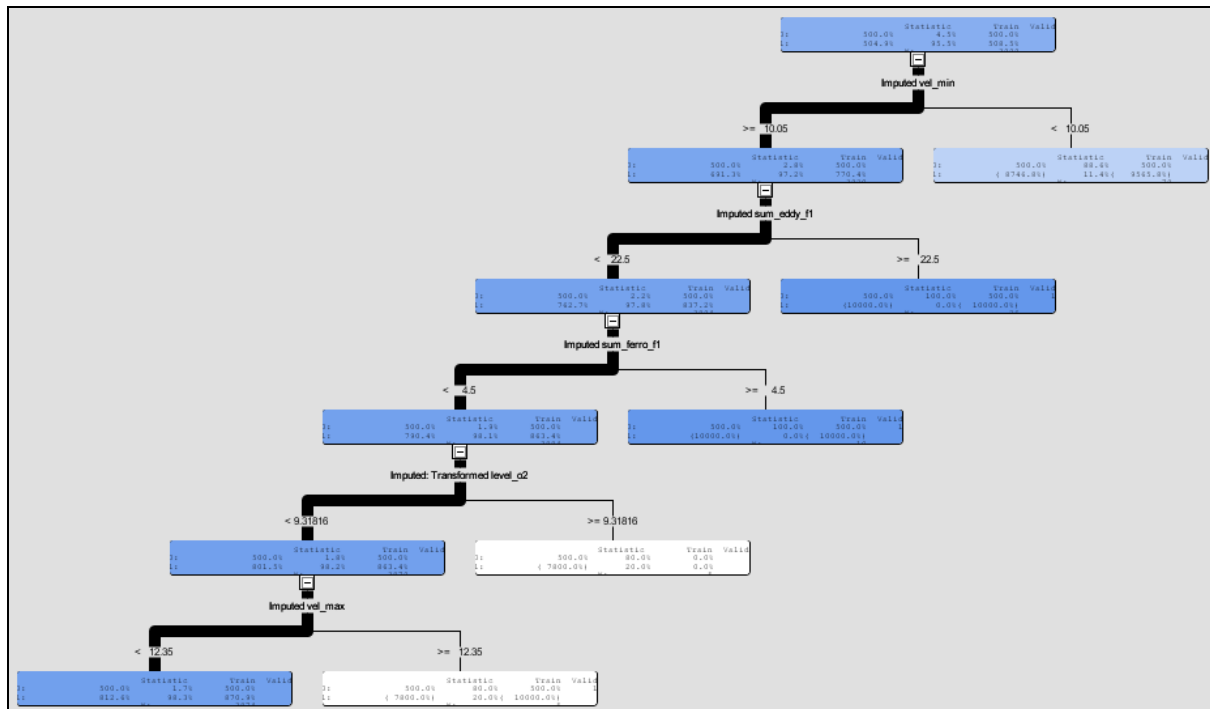
Once the classifiers are fitted to data, the modelling nodes provide the following technical details to the user:

- detailed information about performance of the models measured for the training, validation and test partitions in terms of misclassification rate, total profit etc.,
- detailed information about the fitting process (such as details on subsequent iterations of the forward feature selection process in logistic regression, error rates for subsequent iterations of a neural network, etc.),
- access to the model code in the form of a standalone SAS 4GL program.

Systematic analysis and comparison of performance of the fitted models follows in the section devoted to the Assessment step.

Based on the decision tree model, we will now explain how a model can be analyzed using its Results screen.

The fitted tree is presented graphically, as schematically shown below. Based on performance on validation data, the algorithm fine-tuned the tree to have the depth of 5.



We can examine the tree in the equivalent form of English language rules, where subsequent nodes correspond to the leaf nodes in the graphical representation of the tree. The nodes are related to classification decisions of the tree model, with the majority class in a particular node indicating the tree's answer.

```

IF Imputed vel_min < 10.050000191
THEN
  NODE      :      2
  N         :      79
  1         :    11.4%
  0         :    88.6%

```

```

IF          22.5 <= Imputed sum_eddy_f1
AND 10.050000191 <= Imputed vel_min
THEN
  NODE      :      7
  N         :     26
  1         :     0.0%
  0         :   100.0%

```

```

IF          4.5 <= Imputed sum_ferro_f1
AND Imputed sum_eddy_f1 <          22.5
AND 10.050000191 <= Imputed vel_min
THEN
  NODE      :     11
  N         :     10
  1         :     0.0%
  0         :   100.0%

```

```

IF 9.3181616203 <= Imputed: Transformed
level_o2
AND Imputed sum_ferro_f1 <          4.5
AND Imputed sum_eddy_f1 <          22.5
AND 10.050000191 <= Imputed vel_min
THEN
  NODE      :     13
  N         :      5
  1         :    20.0%
  0         :    80.0%

```

```

IF Imputed vel_max < 12.349999905
AND Imputed: Transformed level_o2 <
9.3181616203
AND Imputed sum_ferro_f1 <          4.5
AND Imputed sum_eddy_f1 <          22.5
AND 10.050000191 <= Imputed vel_min
THEN
  NODE      :     14
  N         :   3874
  1         :   98.3%
  0         :     1.7%

```

```

IF 12.349999905 <= Imputed vel_max
AND Imputed: Transformed level_o2 <
9.3181616203
AND Imputed sum_ferro_f1 <          4.5
AND Imputed sum_eddy_f1 <          22.5
AND 10.050000191 <= Imputed vel_min
THEN
  NODE      :     15
  N         :      5
  1         :    20.0%
  0         :    80.0%

```

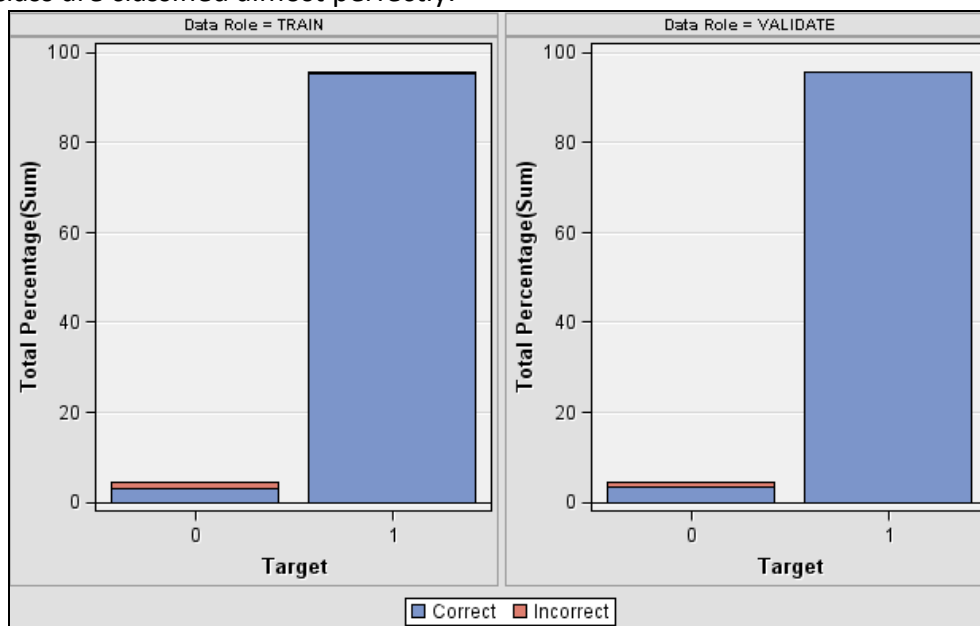
Analyzing the tree model, we can also observe various fit statistics calculated for the training, validate and test partitions:

Target	Fit Statistics	Statistics Label	Train	Validation	Test
qbin	_NOBS_	Sum of Frequencies	3999	2999	3002
qbin	_SUMW_	Sum of Case Weights Times Freq	7998	5998	6004
qbin	_MISC_	Misclassification Rate	0.019255	0.012337	0.015323
qbin	_MAX_	Maximum Absolute Error	0.982963	0.982963	0.982963
qbin	_SSE_	Sum of Squared Errors	148.9005	74.1298	90.28365
qbin	_ASE_	Average Squared Error	0.018617	0.012359	0.015037
qbin	_RASE_	Root Average Squared Error	0.136445	0.111171	0.122626
qbin	_DIV_	Divisor for ASE	7998	5998	6004
qbin	_DFT_	Total Degrees of Freedom	3999	.	.
qbin	_APROF_	Average Profit for qbin	8.028257	8.581194	8.332778
qbin	_PROF_	Total Profit for qbin	32105	25735	25015

We observe that the misclassification rate for the test data (i.e., expected error rate for new data) equals about 1.5%, and the total profit expected from 3002 items in the test partition equals 25015, which translates into average profit per item of 8.33. Note that if all the items were actually good quality and all classification decisions were correct, then the total profit would amount to about 30 thousand. The difference (of ca 5 thousand) is due to

- some poor quality items in the batch (this accounts for ca 0.75K difference),
- the classifier's errors (these account for majority of the difference, i.e., over 4K).

Another interesting perspective in analysis of the tree model is based on the Classification chart, where we observe that the rare class (qbin=0) is indeed much more difficult to properly classify, while items of the frequent class are classified almost perfectly.



Analysis of the tree model also provides information about importance of inputs for prediction of target, as shown below. This information may be useful for implementation of feature selection rules. It can be observed that the tree selects only the first five variables on top of the list below as predictors for estimation of the target.

Variable Name	Number of Splitting Rules	Importance	Validation Importance	Ratio of Validation to Training Importance
MP_vel_min	1	1	1	1
MP_sum_eddy_f1	1	0.65856	0.517785	0.786239
MP_sum_ferro_f1	1	0.41031	0.324253	0.790265
MP_vel_max	1	0.231724	0.170396	0.73534
MP_LOG_level_o2	1	0.231425	0.023481	0.101464
MP_size_max	0	0	0	.
MP_size_min	0	0	0	.
MP_temp	0	0	0	.
MP_sum_eddy_f2	0	0	0	.
MP_sum_ferro_f3	0	0	0	.
MP_sum_eddy_f3	0	0	0	.
MP_sum_ferro_f2	0	0	0	.
MP_vel_mean	0	0	0	.

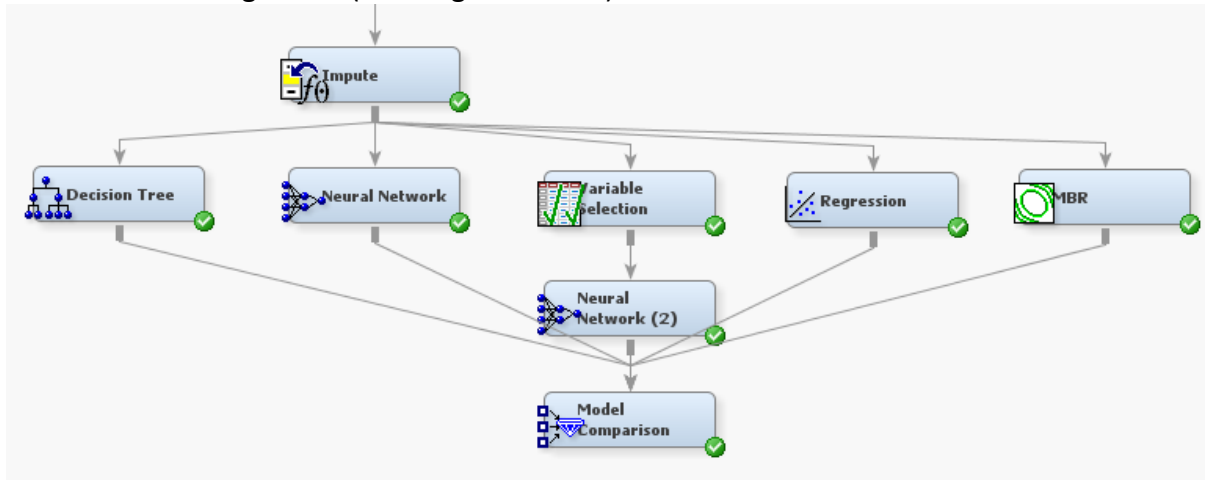
The tree node offers several other interesting methods for analysis of the model, such as e.g., lift analysis. These methods are however more appropriate for problems where the task consists in selection of items with the highest probability of event. In our case, the problem consists in quality prediction of *all* the items with the criterion to minimize the cost of wrong decisions (or maximize overall profit).

Similar in-depth analysis of the fitted models is available with other nodes (Neural Network, Regression, MBR). However, in the next section we will concentrate on comparison of the models in terms of some simple practical criteria.

Assessment of performance of the models

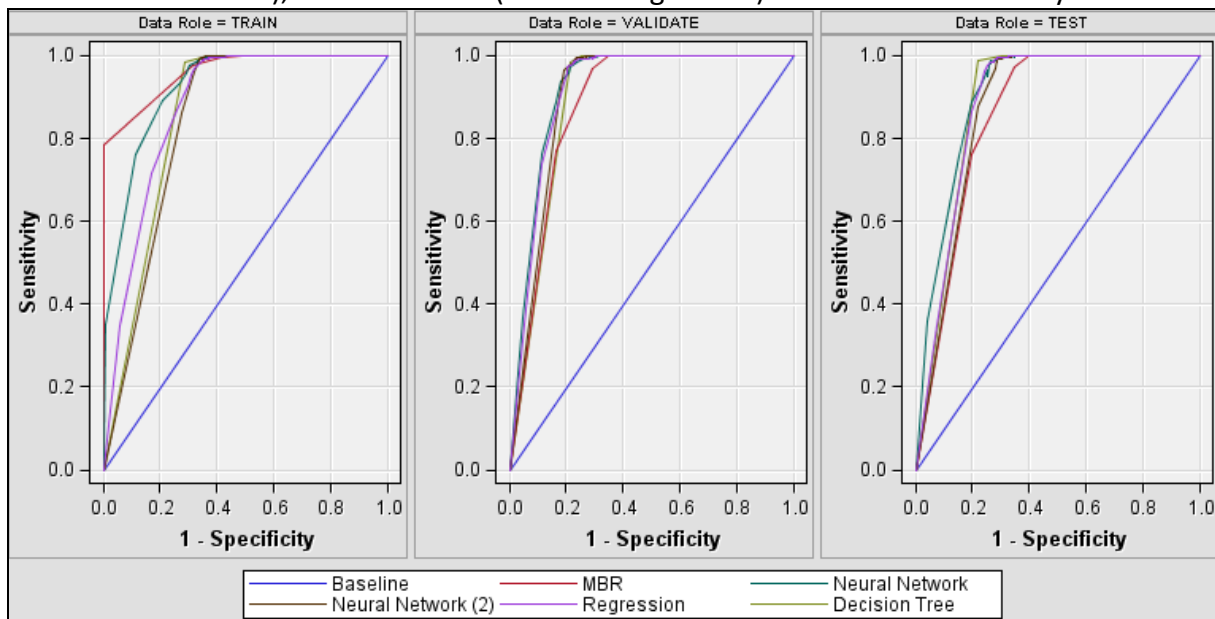
The purpose of this SEMMA step is to evaluate and compare models in terms of their practical usefulness. The models can be compared using several criteria such as the profit/loss, misclassification rate, or using ROC curves or cut-off analysis.

Overall assessment of fitted models is realized with the Model Comparison node, added to the diagram after the modelling nodes (see diagram below).



The Model Comparison node provides several tools for analysis of the models, such as the ROC curves as shown below. The ROC analysis indicates that the models demonstrate similar performance, where sensitivity of ca 100% inevitably translates into about 20% (=1-specificity) error rate in recognition of the other (poor quality) class. Note that sensitivity is related to the target event of 1, i.e., recognition of good quality (frequent) class. The ROC analysis is consistent with the rare class recognition problem, described previously.

In terms of selection of the winning model, no firm conclusion can be made based on the ROC curves. The tree, neural network models and regression perform similarly (with slightly better results of the tree expected for the test data), while the MBR (nearest neighbours) classifier is remarkably weaker.



The qualitative conclusions from the ROC analysis can be quantitatively confirmed through a number of criteria summarized in the table below. These measures are calculated for the test partition, i.e., similar performance can be expected for new data. Whereas all the models misclassify roughly 2% of cases, the tree model slightly outperforms other models in terms of the total (and average per decision) profit, as well as in terms of the total number of wrong classifications.

	Tree	Reg	Neural	Neural2	MBR
Test: KS Statistic	0.76	0.72	0.72	0.70	0.62
Test: Average Profit	8.66	8.41	8.38	8.34	7.68
Test: Average Squared Error	0.01	0.02	0.02	0.02	0.02
Test: Roc Index	0.89	0.89	0.91	0.86	0.86
Test: Average Error Function	0.07	0.08	0.08	0.08	0.26
Test: Bin 2Way KS Prob Cutoff	0.99	0.94	0.94	0.97	0.91
Test: Cum % Captured Response	10.39	10.43	10.46	10.32	10.37
Test: Percent Captured Response	5.18	5.16	5.20	5.13	5.17
Test: Freq of Classified Cases	3002.00	3002.00	3002.00	3002.00	3002.00
Test: Divisor for ASE	6004.00	6004.00	6004.00	6004.00	6004.00
Test: Error Function	423.79	464.20	454.30	483.11	1552.93
Test: Gain	3.61	4.01	4.36	2.97	3.43
Test: Gini Coefficient	0.77	0.78	0.82	0.73	0.71
Test: Bin-Based 2Way KS Statistic	0.76	0.72	0.72	0.69	0.62
Test: KS Probability Cutoff	0.80	0.94	0.89	0.92	0.88
Test: Cumulative Lift	1.04	1.04	1.04	1.03	1.03
Test: Lift	1.04	1.03	1.04	1.03	1.03
Test: Maximum Absolute Error	0.99	0.99	1.00	0.99	1.00
Test: Misclassification Rate	0.02	0.02	0.02	0.02	0.03
Test: Sum of Frequencies	3002.00	3002.00	3002.00	3002.00	3002.00
Test: Total Profit	25985.00	25240.00	25155.00	25035.00	23060.00
Test: Root Average Squared Error	0.12	0.13	0.13	0.13	0.15
Test: Cumulative Percent Response	98.95	99.34	99.67	98.34	98.77
Test: Percent Response	98.95	98.67	99.33	98.00	98.77
Test: Sum of Squared Errors	85.85	95.11	95.20	100.87	133.87
Test: # of Wrong Classifications	46.00	51.00	54.00	57.00	79.00

Again, the Model Comparison node provides several other tools for assessment of models, such as lift analysis; these however do not bring useful interpretation for the problem of quality prediction, and hence are omitted in this guide.

More in-depth analysis of the winning model will be presented in the next section on Scoring new data.

Scoring new data

The purpose of this section is to explain

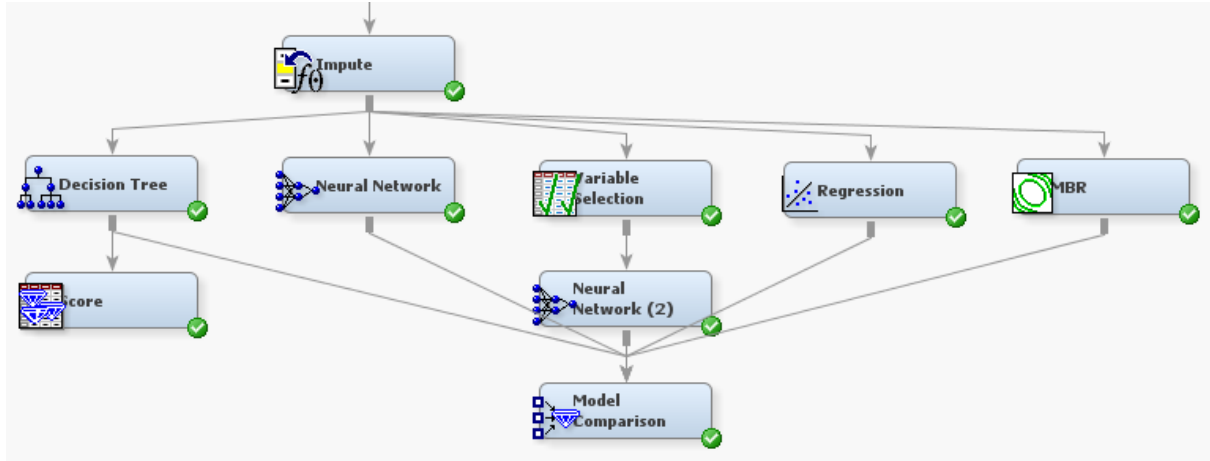
- how the predictive model (e.g., the winning model selected by the Assessment step) can be used to classify (score) new data, and
- how the scoring code can be maintained.

We will also examine contents of the dataset produced by the scoring code and explain how these scored datasets can be the basis for more in-depth analyses using custom SAS coding.

The tool used for scoring new data and for management of scoring codes is the Score node. The node can be connected to:

- any node that produces the scoring code (such as the predictive modelling nodes used in this study),
- the Model Comparison node – in this case the Score node will obtain the winning model from the preceding node (the winning model is selected based on a single criterion specified in the properties of the Model Comparison node).

In this example, we connect the Score node directly to the Decision Tree node, as shown below.

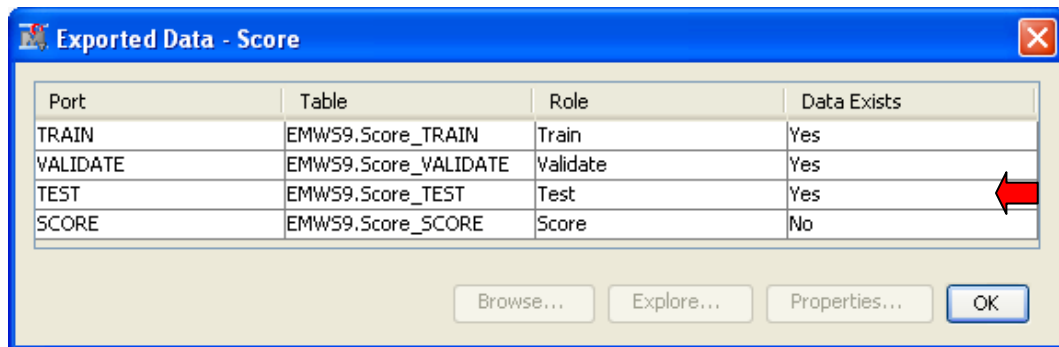


The functionality of the Score node allows to:

- Obtain the scoring code from the preceding node. The scoring code can be then managed outside the Enterprise Miner environment. The scoring code can be exported in the following languages: SAS 4GL, C, Java, PMML.
- Execute the scoring code against a dataset connected to the Score node. Normally, this dataset is connected to the Score node using the Input Data node with the metadata role of Score. Alternatively, the scoring code is applied for the train, validate and test partitions (these data sets are passed through to the Score node).

In this example, we will use the Score node to classify data in the *test* partition, since predictive performance for this partition is a reliable measure of expected performance for new data.

The Exported Data property of the Score node indicates where the results of scoring are placed by the node. In this example the scored test data is found in the EMWS9.Score_TEST dataset.

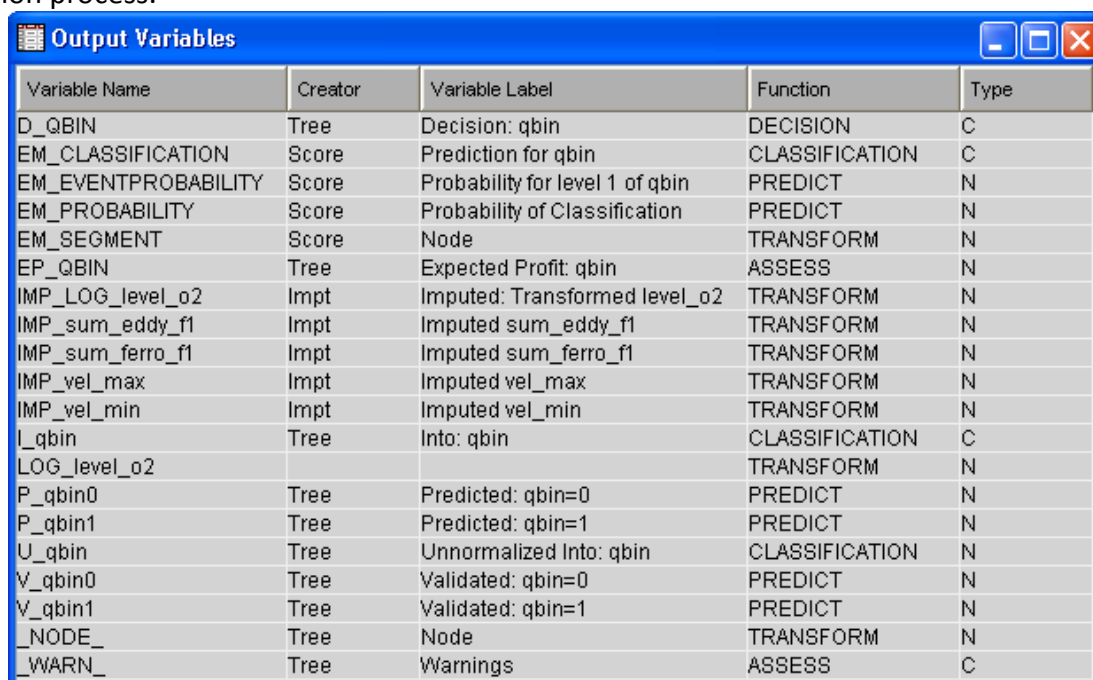


The dialog box titled "Exported Data - Score" contains a table with the following data:

Port	Table	Role	Data Exists
TRAIN	EMWS9.Score_TRAIN	Train	Yes
VALIDATE	EMWS9.Score_VALIDATE	Validate	Yes
TEST	EMWS9.Score_TEST	Test	Yes
SCORE	EMWS9.Score_SCORE	Score	No

At the bottom of the dialog are buttons for "Browse...", "Explore...", "Properties...", and "OK". A red arrow points to the "TEST" row in the table.

The variables in this dataset involved in the classification process are listed by the node in its Results screen as shown below. These include the predictors used by the tree model as well as variables produced by the tree node or the score node to provide detailed technical information pertaining to the classification process.



Variable Name	Creator	Variable Label	Function	Type
D_QBIN	Tree	Decision: qbin	DECISION	C
EM_CLASSIFICATION	Score	Prediction for qbin	CLASSIFICATION	C
EM_EVENTPROBABILITY	Score	Probability for level 1 of qbin	PREDICT	N
EM_PROBABILITY	Score	Probability of Classification	PREDICT	N
EM_SEGMENT	Score	Node	TRANSFORM	N
EP_QBIN	Tree	Expected Profit: qbin	ASSESS	N
IMP_LOG_level_o2	Impt	Imputed: Transformed level_o2	TRANSFORM	N
IMP_sum_eddy_f1	Impt	Imputed sum_eddy_f1	TRANSFORM	N
IMP_sum_ferro_f1	Impt	Imputed sum_ferro_f1	TRANSFORM	N
IMP_vel_max	Impt	Imputed vel_max	TRANSFORM	N
IMP_vel_min	Impt	Imputed vel_min	TRANSFORM	N
I_qbin	Tree	Into: qbin	CLASSIFICATION	C
LOG_level_o2			TRANSFORM	N
P_qbin0	Tree	Predicted: qbin=0	PREDICT	N
P_qbin1	Tree	Predicted: qbin=1	PREDICT	N
U_qbin	Tree	Unnormalized Into: qbin	CLASSIFICATION	N
V_qbin0	Tree	Validated: qbin=0	PREDICT	N
V_qbin1	Tree	Validated: qbin=1	PREDICT	N
NODE	Tree	Node	TRANSFORM	N
WARN	Tree	Warnings	ASSESS	C

The following variables provide interesting technical information about the process of classification:

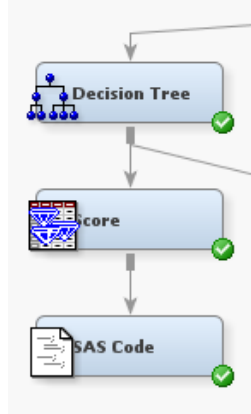
- D_QBIN Contains the predicted class level (quality of an item). The prediction is done by the classifier fine-tuned to maximize profit (i.e., built using the target profile Decision Weights matrix).
- I_QBIN Contains the predicted class label, produced by the classifier fine-tuned to minimize the overall number of misclassified items (i.e., built without using the target profile Decision Weights matrix). The name given to this variable by the scoring node is EM_CLASSIFICATION.
- EM_EVENTPROBABILITY The probability associated with the classifier's decision that an item is good quality.
- EM_PROBABILITY The probability of the decision finally made by the classifier. This probability is estimated as:

$$EM_PROBABILITY = \max(EM_EVENTPROBABILITY, 1 - EM_EVENTPROBABILITY)$$

Given this technical output appended to the results of scoring dataset (i.e., EMWS9.Score_TEST), further in-depth analysis of the model itself or of the scored data is possible using custom SAS coding.

To illustrate this, we will post-process results of scoring to calculate the *coincidence matrix* and the *sensitivity* and *specificity* parameters of the model.

To do this, the SAS Code node is connected to the Score node, as shown below.



In order to compute the coincidence matrix, the following SAS code is placed in the SAS Code node (the Code Editor is available through properties of this node):

```
proc freq data=emws9.score_test;
  tables qbin*d_qbin;
  tables qbin*i_qbin;
run;
```

PROC FREQ is the SAS/STAT procedure used to produce frequency or contingency tables to examine relationship between two classification variables.

In this example, we use the FREQ procedure to compare:

- the actual quality of copper (qbin) with the quality predicted using the profit maximization rule (this decision is coded in the d_qbin classifier's output variable), or
- the actual quality of copper (qbin) with the quality predicted using the misclassification rate minimization rule (this decision is coded in the i_qbin classifier's output variable).

The coincidence matrixes summarizing performance of these two classifiers are given below. We also calculate the total profit and misclassification rates.

The conclusions can be summarized as follows:

- The model based on decision weights indeed realizes higher total profit as compared to the original classifier (25985 vs. 25015), although the total number of misclassifications is higher (72 vs. 46).
- Improvement in the total profit is achieved by reducing the number of the costly 0→1 classification errors (from 41 to 30), at the expense of increased 1→0 error rate.

Classifier fine tuned to maximize the total profit	Classifier fine tuned to minimize the misclassification rate
Table of qbin by D_QBIN	Table of qbin by I_qbin

qbin				D_QBIN(Decision: qbin)				qbin				I_qbin(Into: qbin)											
Frequency				Percent				Frequency				Percent											
Row Pct				Col Pct				Row Pct				Col Pct											
0				1				Total				0				1				Total			
105				30				135				94				41				135			
3.50				1.00				4.50				3.13				1.37				4.50			
77.78				22.22								69.63				30.37							
71.43				1.05								94.95				1.41							
42				2825				2867				5				2862				2867			
1.40				94.10				95.50				0.17				95.34				95.50			
1.46				98.54								0.17				99.83							
28.57				98.95								5.05				98.59							
147				2855				3002				99				2903				3002			
4.90				95.10				100.00				3.30				96.70				100.00			
TOTAL PROFIT 25985								TOTAL PROFIT 25015															
TOTAL # OF MISCLASSIFICATIONS 72								TOTAL # OF MISCLASSIFICATIONS 46															

These models can also be compared in terms of sensitivity and specificity. These parameters compare as follows:

- model on the left: sensitivity=98.54% , specificity=77.78%
- model on the right: sensitivity=99.83% , specificity=69.63%

Observe that (1-specificity) is the misclassification rate for the rare class (poor quality items): this parameter was reduced from ca 30% to 22%. This analysis confirms that the decision weights matrix leads to improvement in recognition of the rare class.