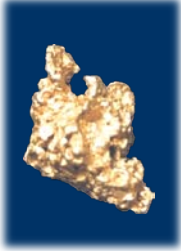


Chapter 3 Introduction to Predictive Modeling: Decision Trees

0.1	Introduction.....	Error! Bookmark not defined.
0.2	A Section Title.....	Error! Bookmark not defined.
	Demonstration: <Type title of demo here.>.....	Error! Bookmark not defined.
	Exercises	Error! Bookmark not defined.
0.3	Chapter Summary.....	Error! Bookmark not defined.
0.4	Solutions	Error! Bookmark not defined.
	Solutions to Exercises	Error! Bookmark not defined.
	Solutions to Student Activities (Polls/Quizzes)	Error! Bookmark not defined.

3.1 Introduction

Predictive Modeling



The Essence of Data Mining

“Most of the big payoff [in data mining] has been in predictive modeling.”

– Herb Edelstein

3

Predictive modeling has a long history of success in the field of data mining. Models are built from historical event records and are used to predict future occurrences of these events. The methods have great potential for monetary gain. Herb Edelstein states in a 1997 interview (Beck 1997):

"I also want to stress that data mining is successfully applied to a wide range of problems. The beer-and-diapers type of problem is known as association discovery or market basket analysis, that is, describing something that has happened in existing data. This should be differentiated from predictive techniques. *Most of the big payoff stuff has been in predictive modeling.*"

Predicting the future has obvious financial benefit, but, for several reasons, the ability of predictive models to do so should not be overstated.

First, the models depend on a property known statistically as *stationarity*, meaning that its statistical properties do not change over time. Unfortunately, many processes that generate events of interest are not stationary enough to enable meaningful predictions.

Second, predictive models are often directed at predicting events that occur rarely, and in the aggregate, often look somewhat random. Even the best predictive model can only extract rough trends from these inherently *noisy* processes.

Predictive Modeling Applications



Database marketing



Financial risk management



Fraud detection



Process monitoring



Pattern detection

4

Applications of predictive modeling are only limited by your imagination. However, the models created by SAS Enterprise Miner typically apply to one of the following categories.

- Database marketing applications include offer response, up-sell, cross-sell, and attrition models.
- Financial risk management models attempt to predict monetary events such as credit default, loan prepayment, and insurance claim.
- Fraud detection methods attempt to detect or impede illegal activity involving financial transactions.
- Process monitoring applications detect deviations from the norm in manufacturing, financial, and security processes.
- Pattern detection models are used in applications ranging from handwriting analysis to medical diagnostics.

Predictive Modeling Training Data

Training Data

	inputs			target

Training data case: categorical or numeric input and target measurements

5
...

To begin using SAS Enterprise Miner for predictive modeling, you should be familiar with standard terminology.

Predictive modeling (also known as *supervised prediction* or *supervised learning*) starts with a *training data set*. The observations in a training data set are known as *training cases* (also known as *examples*, *instances*, or *records*). The variables are called *inputs* (also known as *predictors*, *features*, *explanatory variables*, or *independent variables*) and *targets* (also known as a *response*, *outcome*, or *dependent variable*). For a given case, the inputs reflect your state of knowledge before measuring the target.

The measurement scale of the inputs and the target can be varied. The inputs and the target can be numeric variables, such as income. They can be nominal variables, such as occupation. They are often binary variables, such as a positive or negative response concerning home ownership.

Predictive Model

Training Data

	inputs			target

► Predictive model: a concise representation of the input and target association

7 ...

The purpose of the training data is to generate a predictive model. The *predictive model* is a concise representation of the association between the inputs and the target variables.

Predictive Model

	inputs		

►

predictions

Predictions: output of the predictive model given a set of input measurements

10 ...

The outputs of the predictive model are called *predictions*. Predictions represent your best guess for the target given a set of input measurements. The predictions are based on the associations **learned** from the training data by the predictive model.

Modeling Essentials

- ▶ Predict new cases.
- ▶ Select useful inputs.
- ▶ Optimize complexity.

12

...

Predictive models are widely used and come in many varieties. Any model, however, must perform three essential tasks:

- provide a rule to transform a measurement into a prediction
- have a means of choosing useful inputs from a potentially vast number of candidates
- be able to adjust its complexity to compensate for noisy training data

The following slides illustrate the general principles involved in each of these essentials. Later sections of the course detail the approach used by specific modeling tools.

Modeling Essentials

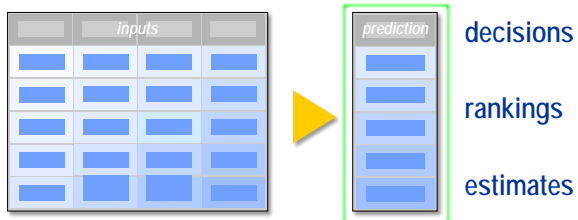
- ▶ Predict new cases.
- ▶ Select useful inputs.
- ▶ Optimize complexity.

13

...

The process of predicting new cases is examined first.

Three Prediction Types



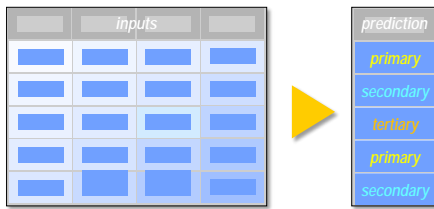
15

...

The training data is used to construct a model (rule) that relates the input variables to the original target variable. The predictions can be categorized into three distinct types:

- decisions
- rankings
- estimates

Decision Predictions



A predictive model uses input measurements to make the best decision for each case.

17

...

The simplest type of prediction is the *decision*. Decisions usually are associated with some type of action (such as classifying a case as a donor or a non-donor). For this reason, decisions are also known as *classifications*. Decision prediction examples include handwriting recognition, fraud detection, and direct mail solicitation.

Decision predictions usually relate to a categorical target variable. For this reason, they are identified as primary, secondary, and tertiary in correspondence with the levels of the target.



By default, model assessment in SAS Enterprise Miner assumes decision predictions when the target variable has a categorical measurement level (binary, nominal, or ordinal).

Ranking Predictions

Inputs			

➤

Prediction
720
520
580
470
630

A predictive model uses input measurements to optimally rank each case.

19
...

Ranking predictions order cases based on the input variables' relationships with the target variable. Using the training data, the prediction model attempts to rank **high value** cases higher than **low value** cases. It is assumed that a similar pattern exists in the scoring data so that **high value** cases have high scores. The actual, produced scores are inconsequential; only the relative order is important. The most common example of a ranking prediction is a credit score.



Ranking predictions can be transformed into decision predictions by taking the primary decision for cases above a certain threshold while making secondary and tertiary decisions for cases below the correspondingly lower thresholds. In credit scoring, cases with a credit score above 700 can be called good risks; those with a score between 600 and 700 can be intermediate risks; and those below 600 can be considered poor risks.

Estimate Predictions

Inputs			



prediction
0.65
0.33
0.54
0.28
0.75

A predictive model uses input measurements to optimally estimate the target value.

21

...


Estimate predictions approximate the expected value of the target, conditioned on the input values. For cases with numeric targets, this number can be thought of as the average value of the target for all cases having the observed input measurements. For cases with categorical targets, this number might equal the probability of a particular target outcome.

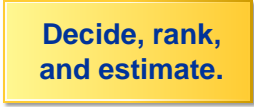
Prediction estimates are most commonly used when their values are integrated into a mathematical expression. An example is two-stage modeling, where the probability of an event is combined with an estimate of profit or loss to form an estimate of unconditional expected profit or loss. Prediction estimates are also useful when you are not certain of the ultimate application of the model.





Estimate predictions can be transformed into both decision and ranking predictions. When in doubt, use this option. Most SAS Enterprise Miner modeling tools can be configured to produce estimate predictions.

Modeling Essentials – Predict Review

 Predict new cases.



 Select useful inputs.

 Optimize complexity.

23 ...

To summarize the first model essential, when a model predicts a new case, the model generates a decision, a rank, or an estimate.

3.01 Quiz

Match the Predictive Modeling Application to the Decision Type.

Modeling Application	Decision Type
Loss Reserving	A. Decision
Risk Profiling	B. Ranking
Credit Scoring	C. Estimate
Fraud Detection	
Revenue Forecasting	
Voice Recognition	

25

The quiz shows some predictive modeling examples and the decision types.

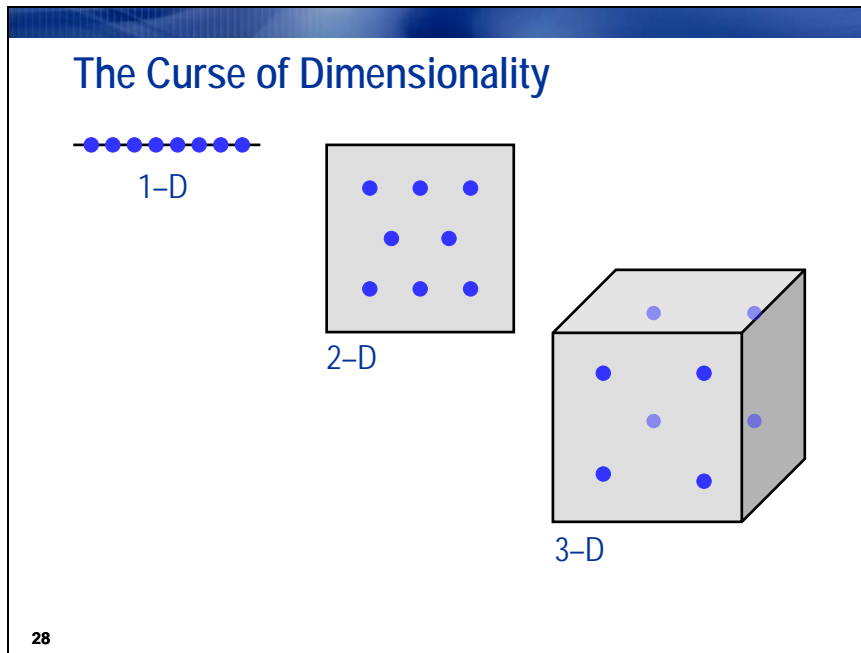
Modeling Essentials

- ▶ Predict new cases.
- ▶ **Select useful inputs.**
- ▶ Optimize complexity.

27

...

Consider the task of selecting useful inputs.



The *dimension* of a problem refers to the number of input variables (more accurately, *degrees of freedom*) that are available for creating a prediction. Data mining problems are often massive in dimension.

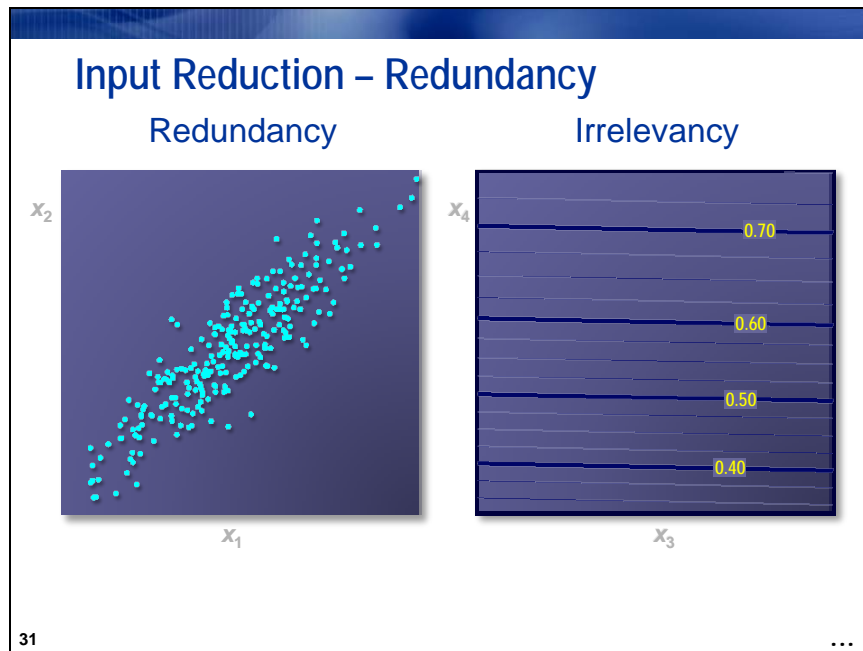
The *curse of dimensionality* refers to the exponential increase in data required to densely populate space as the dimension increases. For example, the eight points fill the one-dimensional space but become more separated as the dimension increases. In a 100-dimensional space, they would be similar to distant galaxies.

The curse of dimensionality limits your practical ability to fit a flexible model to noisy data (real data) when there are a large number of input variables. A densely populated input space is required to fit highly complex models. When you assess how much data is available for data mining, you must consider the dimension of the problem.

3.02 Quiz

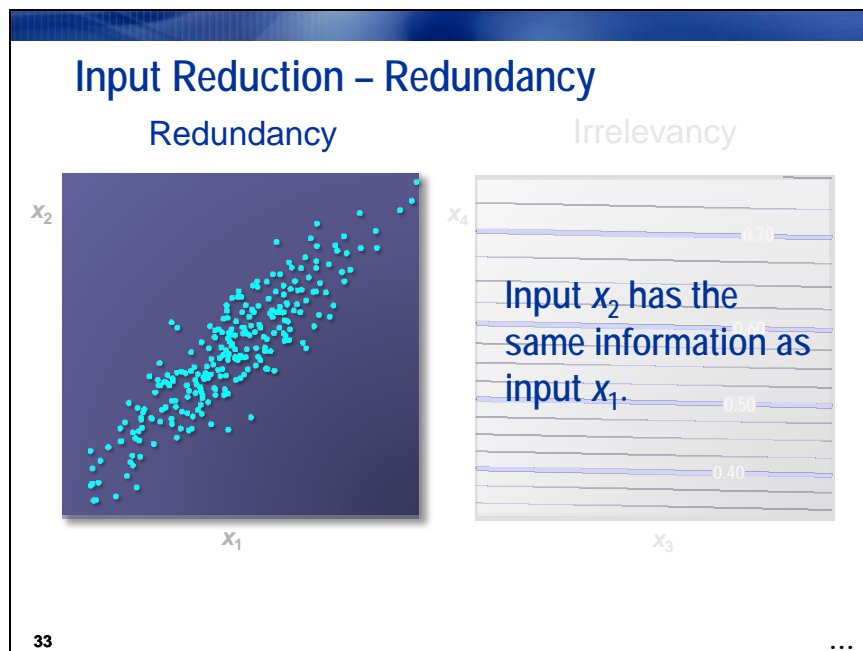
How many inputs are in your modeling data?

Mark your selection in the polling area.



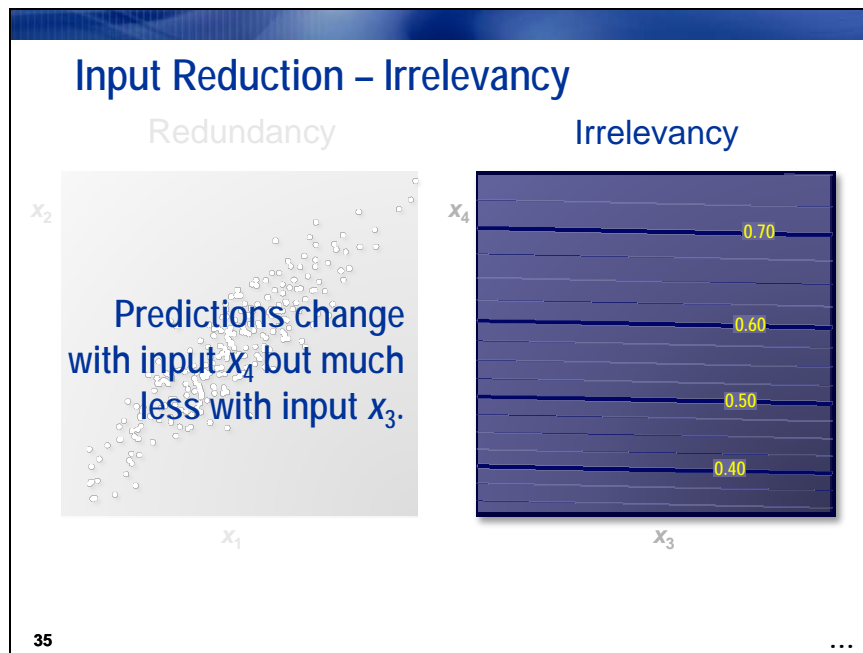
Input selection, that is, reducing the number of inputs, is the obvious way to thwart the curse of dimensionality. Unfortunately, reducing the dimension is also an easy way to disregard important information.

The two principal reasons for eliminating a variable are redundancy and irrelevancy.



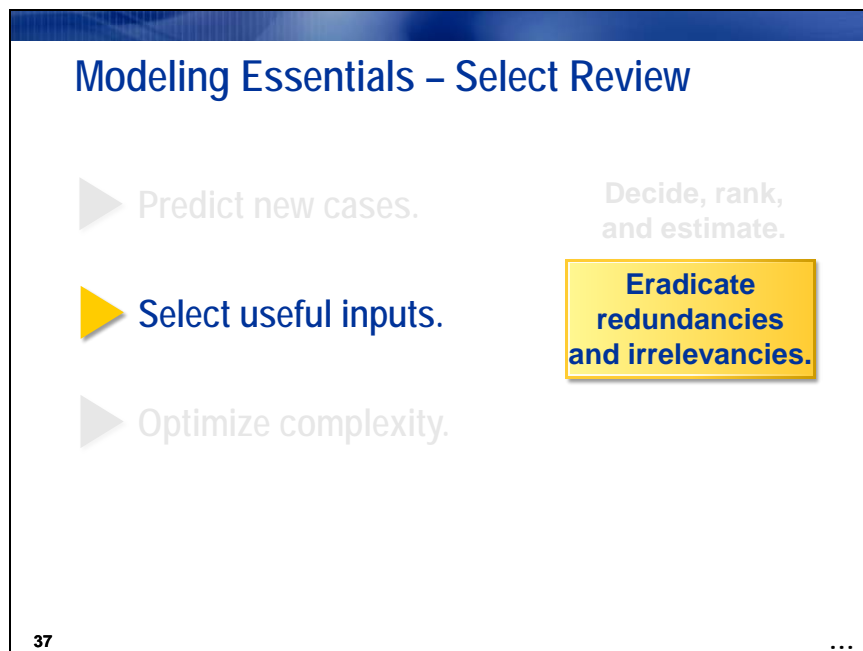
A *redundant* input does not give any new information that was not already explained by other inputs. In the example above, knowing the value of input x_1 gives you a good idea of the value of x_2 .

For decision tree models, the modeling algorithm makes input redundancy a relatively minor issue. For other modeling tools, input redundancy requires more elaborate methods to mitigate the problem.



An *irrelevant* input does not provide information about the target. In the example above, predictions change with input x_4 , but not with input x_3 .

For decision tree models, the modeling algorithm automatically ignores irrelevant inputs. Other modeling methods must be modified or rely on additional tools to properly deal with irrelevant inputs.



To thwart the curse of dimensionality, a model must select useful inputs. You can do this by eradicating redundant and irrelevant inputs (or more positively, selecting an independent set of inputs that are correlated with the target).

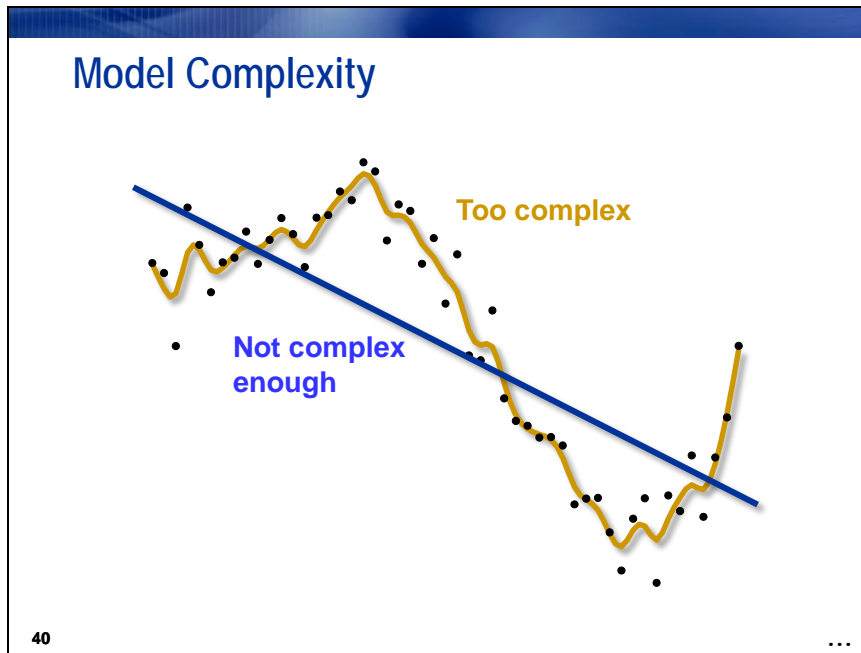
Modeling Essentials

- ▶ Predict new cases.
- ▶ Select useful inputs.
- ▶ **Optimize complexity.**

39

...

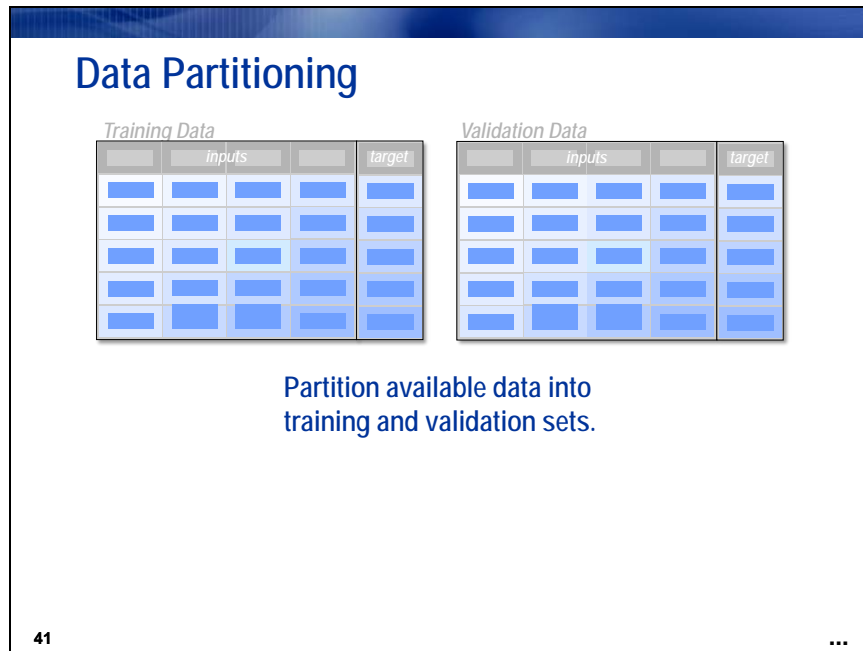
Finally, consider the task of optimizing model complexity.



Fitting a model to data requires searching through the space of possible models. Constructing a model with good generalization requires choosing the right complexity.

Selecting model complexity involves a trade-off between bias and variance. An insufficiently complex model might not be flexible enough, which can lead to *underfitting*, that is, systematically missing the signal (high bias).

A naïve modeler might assume that the most complex model should always outperform the others, but this is not the case. An overly complex model might be too flexible, which can lead to *overfitting*, that is, accommodating nuances of the random noise in the particular sample (high variance). A model with the right amount of flexibility gives the best generalization.



A quote from the popular introductory data analysis textbook *Data Analysis and Regression* by Mosteller and Tukey (1977) captures the difficulty of properly estimating model complexity.

"Testing the procedure on the data that gave it birth is almost certain to overestimate performance, for the optimizing process that chose it from among many possible procedures will have made the greatest use of any and all idiosyncrasies of those particular data...."

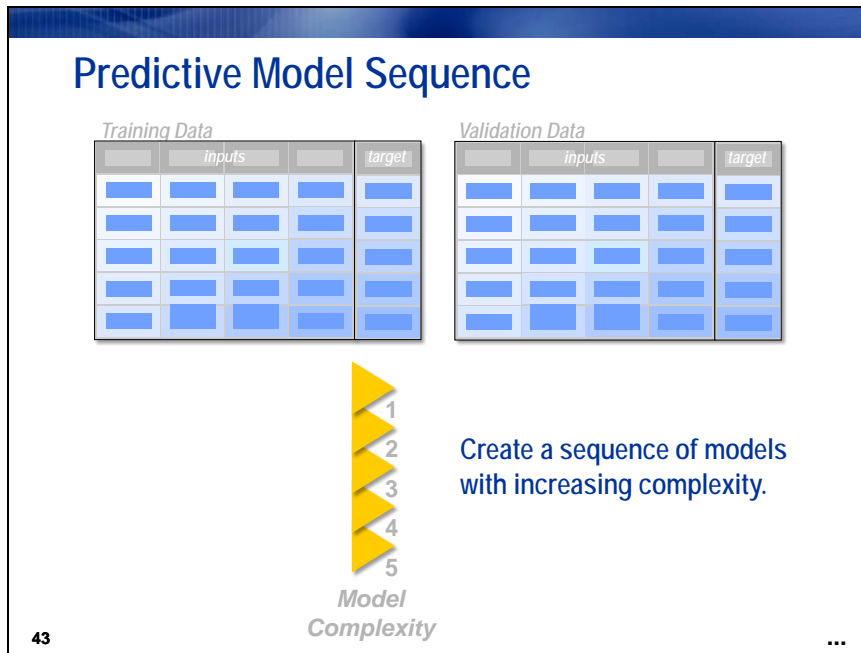
In predictive modeling, the standard strategy for honest assessment of model performance is *data splitting*. A portion is used for fitting the model, that is, the training data set. The remaining data is separated for empirical validation.

The *validation data set* is used for monitoring and tuning the model to improve its generalization. The tuning process usually involves selecting among models of different types and complexities. The tuning process optimizes the selected model on the validation data.

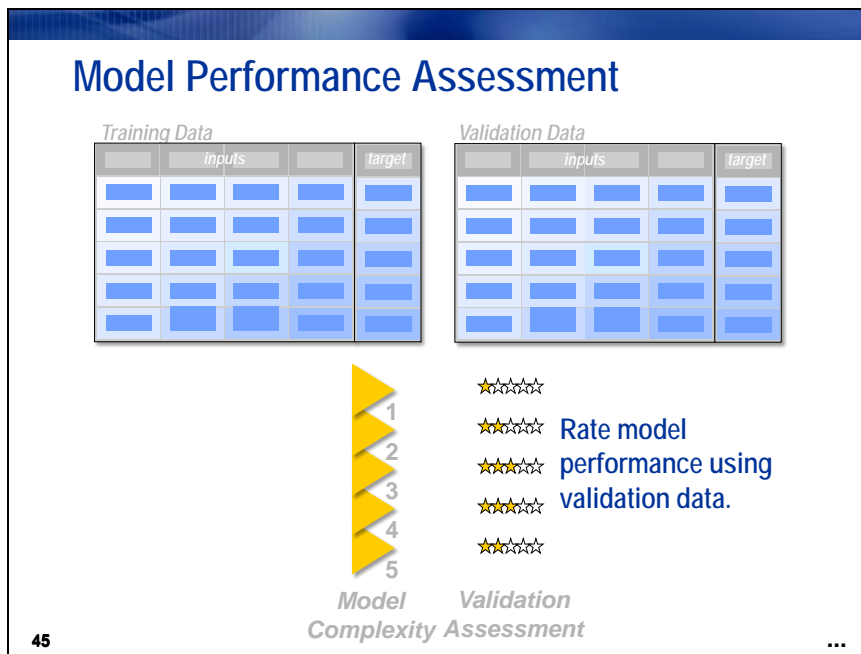


Because the validation data is used to select from a set of related models, reported performance will, on the average, be overstated. Consequently, a further holdout sample is needed for a final, unbiased assessment. The *test data set* has only one use, that is, to give a final honest estimate of generalization. Consequently, cases in the test set must be treated in the same way that new data would be treated. The cases cannot be involved in any way in the determination of the fitted prediction model. In practice, many analysts see no need for a final honest assessment of generalization. An optimal model is chosen using the validation data, and the model assessment measured on the validation data is reported as an upper bound on the performance expected when the model is deployed.

With small or moderate data sets, data splitting is inefficient; the reduced sample size can severely degrade the fit of the model. Computer-intensive methods, such as the *cross validation* and *bootstrap* methods, were developed so that all the data can be used for both fitting and honest assessment. However, data mining usually has the luxury of massive data sets.



In SAS Enterprise Miner, the strategy for choosing model complexity is to build a sequence of similar models using the training component only. The models are usually ordered with increasing complexity.



As Mosteller and Tukey caution in the earlier quotation, using performance on the training data set usually leads to selecting a model that is too complex. (The classic example is selecting linear regression models based on R square.) To avoid this problem, SAS Enterprise Miner selects the model from the sequence based on validation data performance.


Model Selection

Training Data

	inputs			target

Validation Data

	inputs			target



1
2
3
4
5

Model Complexity

Validation Assessment

☆☆☆☆☆

☆☆☆☆☆ Select the simplest

☆☆☆☆☆ model with the highest

☆☆☆☆☆ validation assessment.

☆☆☆☆☆

...

In keeping with Ockham's razor, the best model is the simplest model with the highest validation performance.

Modeling Essentials – Optimize Review

▶ Predict new cases.	Decide, rank, and estimate.
▶ Select useful inputs.	Eradicate redundancies and irrelevancies.
▶ Optimize complexity.	Tune models with validation data.

49 ...

To avoid overfitting (and underfitting) a predictive model, you should calibrate its performance with an independent validation sample. You can perform the calibration by partitioning the data into training and validation samples.

In Chapter 2, you assigned roles and measurement levels to the **PVA97NK** data set and created an analysis diagram and a simple process flow. In the following demonstration, you partition the raw data into training and validation components. This sets the stage for using the Decision Tree tool to generate a predictive model.



Creating Training and Validation Data

A critical step in prediction is choosing among competing models. Given a set of training data, you can easily generate models that very accurately predict a target value from a set of input values.

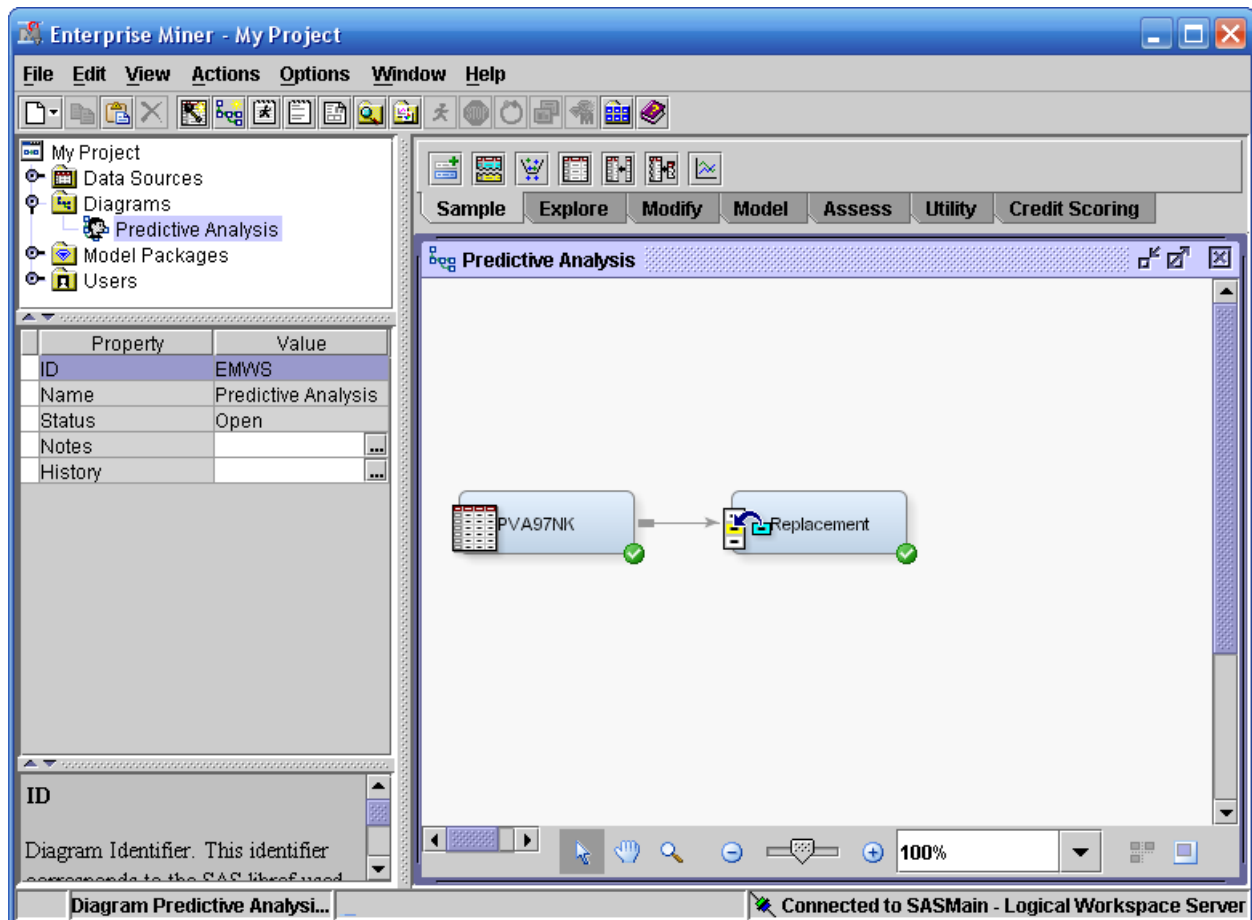
Unfortunately, these predictions might be accurate only for the training data itself. Attempts to generalize the predictions from the training data to an independent, but similarly distributed, sample can result in substantial accuracy reductions.

To avoid this pitfall, SAS Enterprise Miner is designed to use a *validation data set* as a means of independently gauging model performance. Typically, the validation data set is created by partitioning the raw analysis data. Cases selected for training are used to build the model, and cases selected for validation are used to tune and compare models.



SAS Enterprise Miner also enables the creation of a third partition, named the *test data set*. The test data set is meant for obtaining unbiased estimates of model performance from a single selected model.

This demonstration assumes that you completed the demonstrations in Chapter 2. Your SAS Enterprise Miner workspace should appear as shown below:

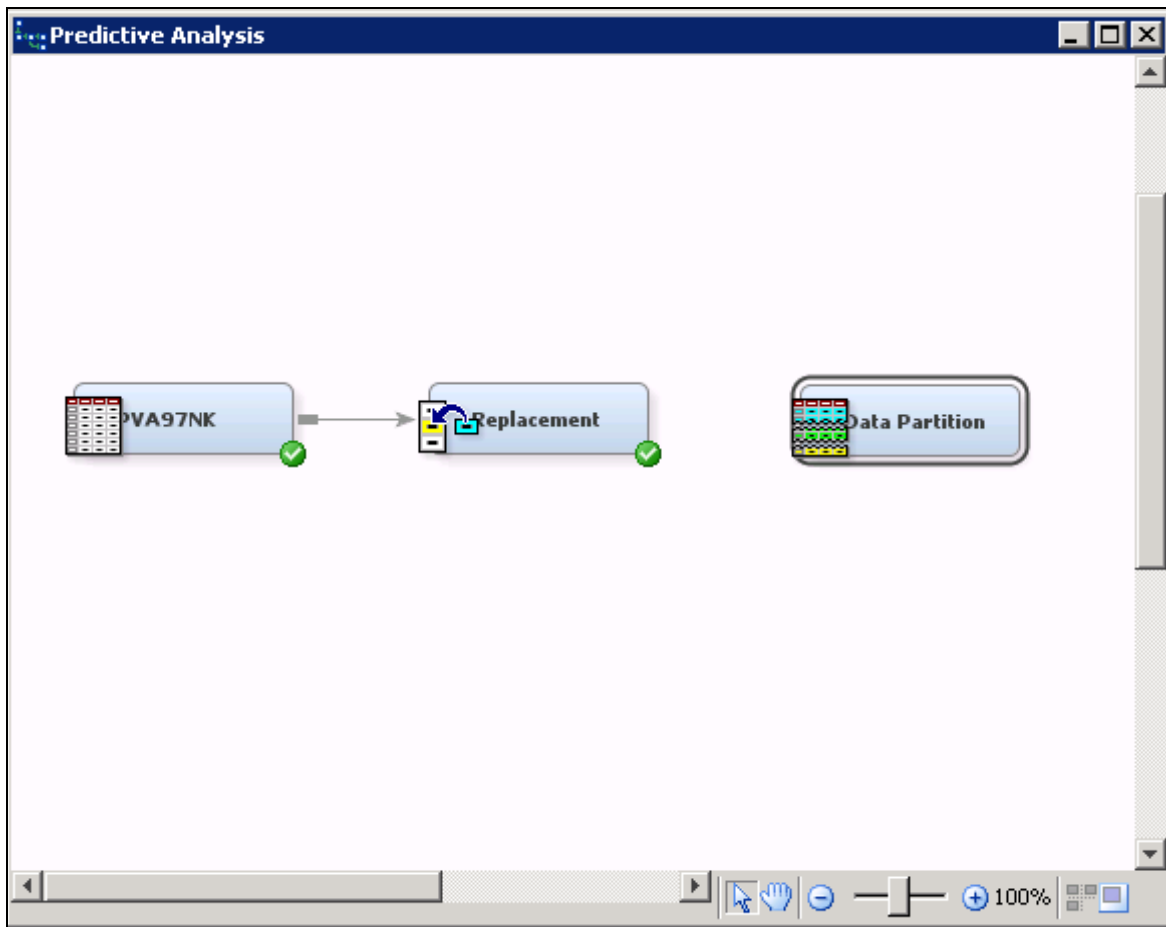


Follow these steps to partition the raw data into training and validation sets.

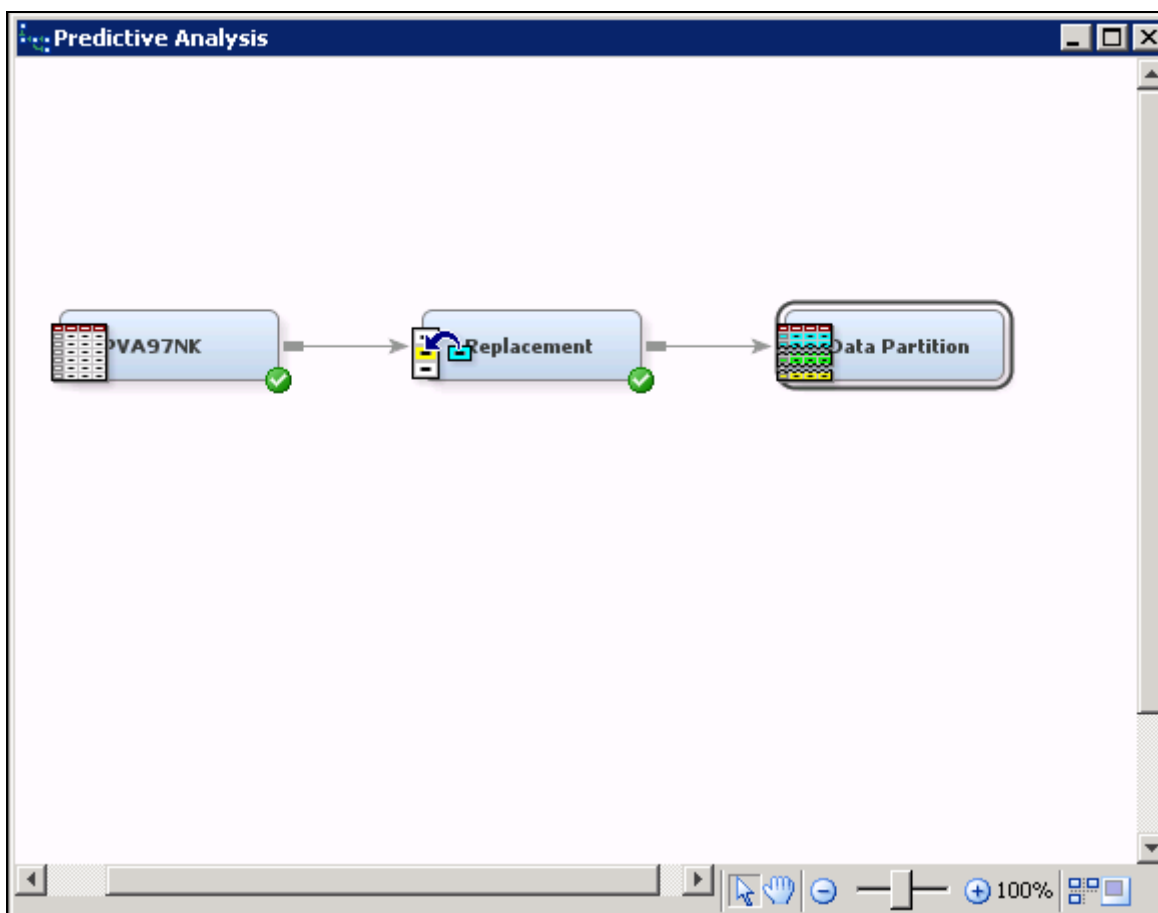
1. Select the **Sample** tool tab. The Data Partition tool is the second from the left.



2. Drag a **Data Partition** tool into the diagram workspace, next to the Replacement node.



3. Connect the **Replacement** node to the **Data Partition** node.



4. Select the **Data Partition** node and examine its Properties panel.

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
Report	
Interval Targets	Yes
Class Targets	Yes

Use the Properties panel to select the fraction of data devoted to the Training, Validation, and Test partitions. By default, less than half the available data is used for generating the predictive models.



There is a trade-off in various partitioning strategies. More data devoted to training results in more stable predictive models, but less stable model assessments (and vice versa). Also, the Test partition is used only for calculating fit statistics after the modeling and model selection is complete. Many analysts regard this as wasteful of data.

A typical partitioning compromise foregoes a Test partition and places an equal number of cases in Training and Validation.

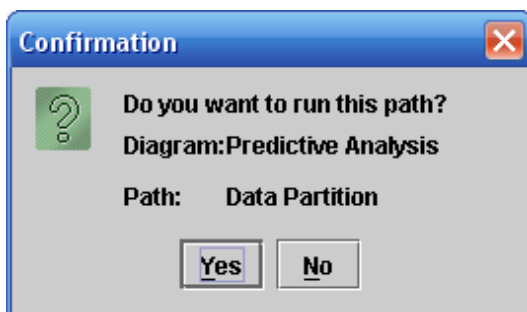
5. Type **50** as the Training value in the Data Partition node.
6. Type **50** as the Validation value in the Data Partition node.
7. Type **0** as the Test value in the Data Partition node.

Data Set Allocations	
Training	50.0
Validation	50.0
Test	0.0



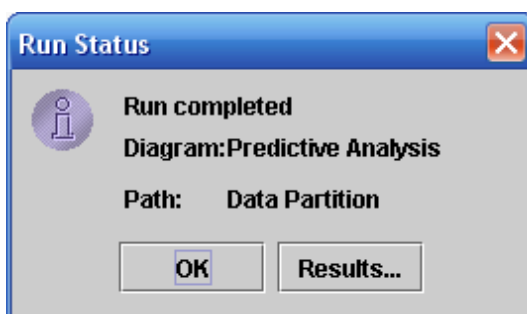
With smaller raw data sets, model stability can become an important issue. In this case, increasing the number of cases devoted to the Training partition is a reasonable course of action.

8. Right-click the **Data Partition** node and select **Run**.
9. Select **Yes** in the Confirmation window. SAS Enterprise Miner runs the Data Partition node.



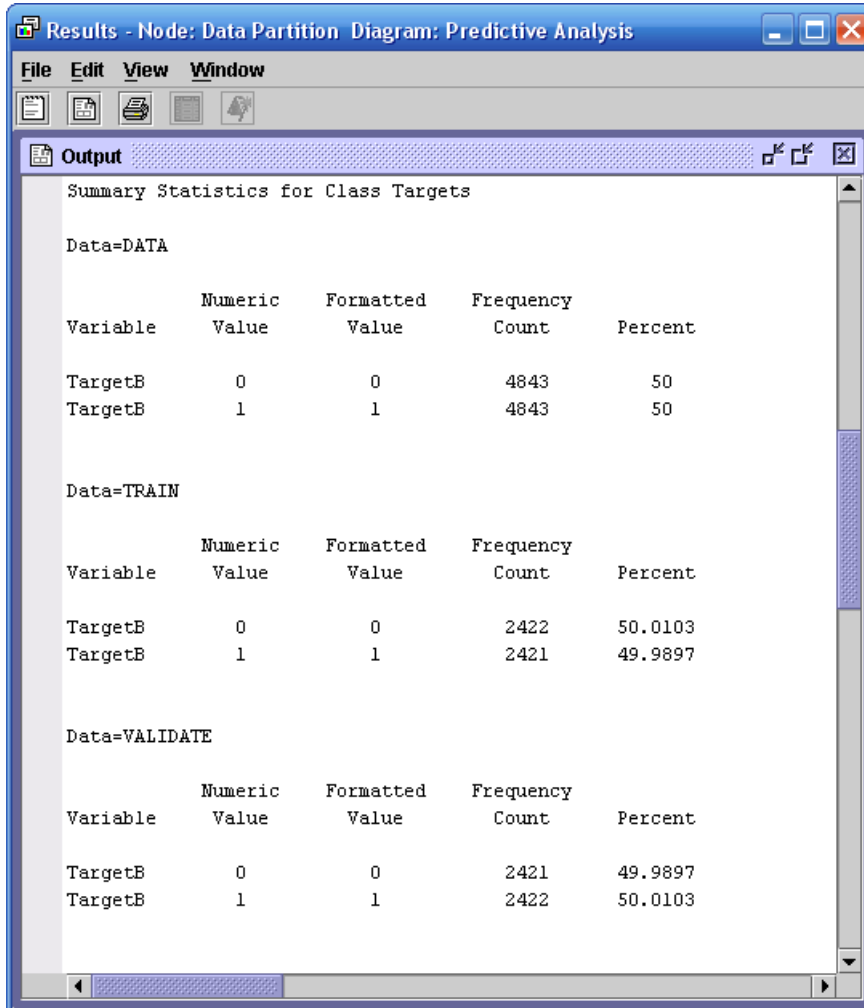
Only the Data Partition node should run, because it is the only "new" element in the process flow. In general, SAS Enterprise Miner only runs elements of diagrams that are new or that are affected by changes earlier in the process flow.

When the Data Partition process is complete, a Run Status window opens.



10. Select **Results...** in the Run Status window to view the results.

The Results - Node: Data Partition window provides a basic metadata summary of the raw data that feeds the node and, more importantly, a frequency table that shows the distribution of the target variable, **TargetB**, in the raw, training, and validation data sets.



Results - Node: Data Partition Diagram: Predictive Analysis

File Edit View Window

Output

Summary Statistics for Class Targets

Data=DATA

Variable	Numeric Value	Formatted Value	Frequency Count	Percent
TargetB	0	0	4843	50
TargetB	1	1	4843	50

Data=TRAIN

Variable	Numeric Value	Formatted Value	Frequency Count	Percent
TargetB	0	0	2422	50.0103
TargetB	1	1	2421	49.9897

Data=VALIDATE

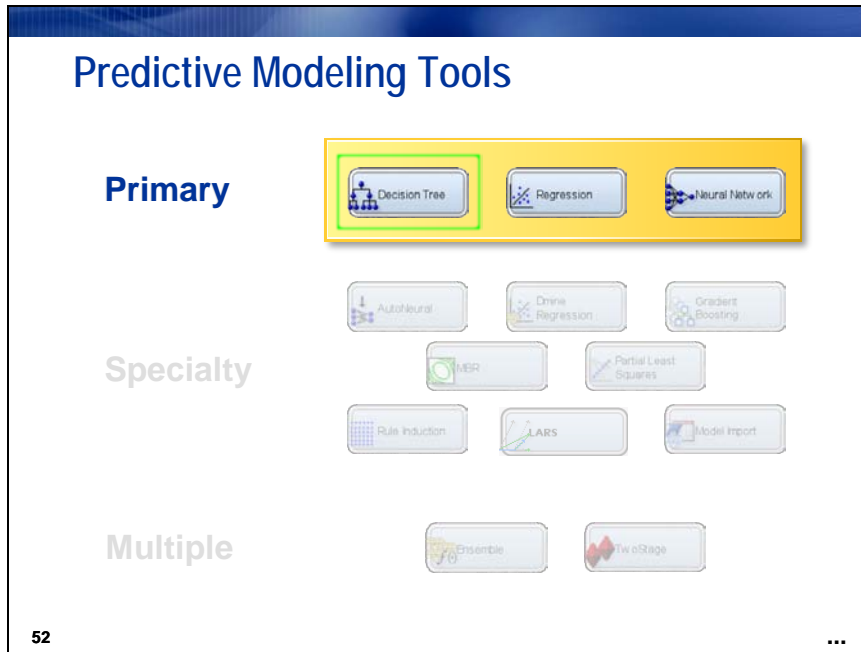
Variable	Numeric Value	Formatted Value	Frequency Count	Percent
TargetB	0	0	2421	49.9897
TargetB	1	1	2422	50.0103



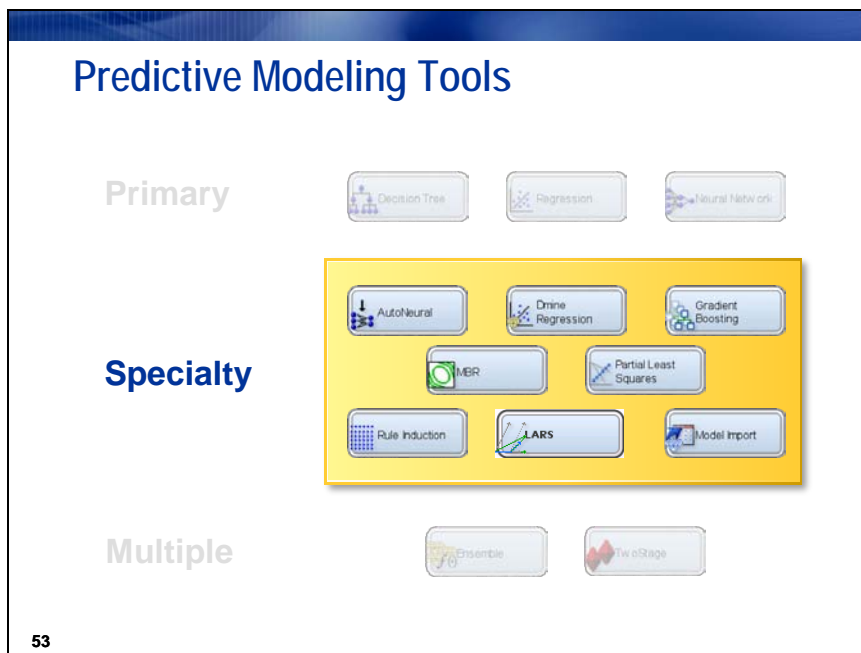
The Partition node attempts to maintain the proportion of zeros and ones in the training and validation partitions. The proportions are not exactly the same due to an odd number of zeros' and ones' cases in the raw data.

The analysis data was selected and partitioned. You are ready to build predictive models.

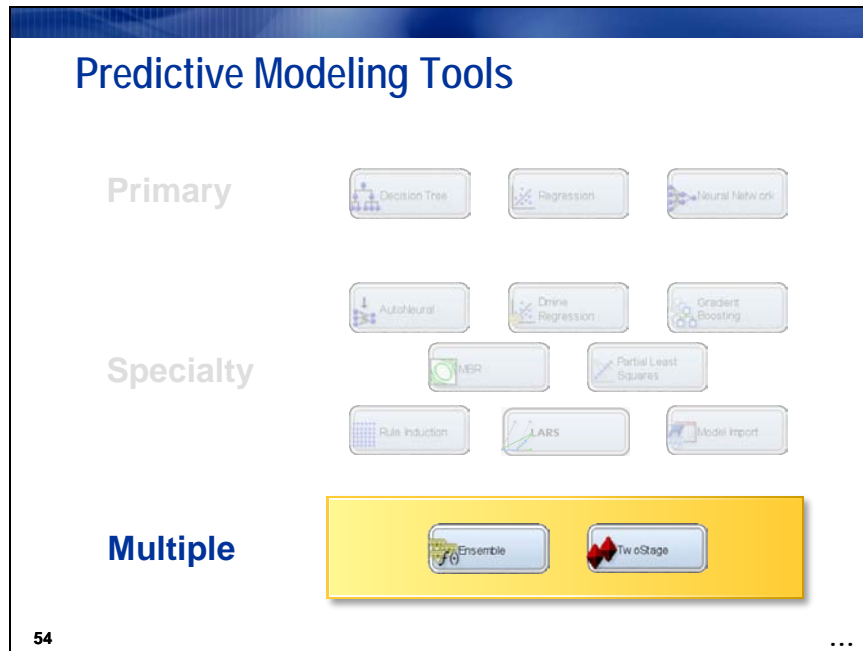
3.2 Cultivating Decision Trees



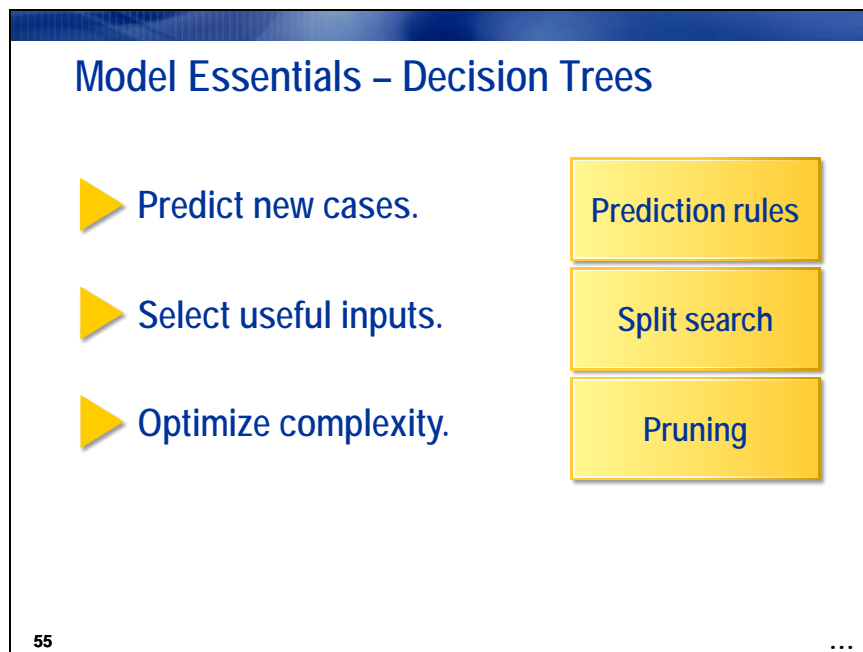
Many predictive modeling tools are found in SAS Enterprise Miner. The tools can be loosely grouped into three categories: primary, specialty, and multiple models. The **primary** tools interface with the three most commonly used predictive modeling methods: decision tree, regression, and neural network.



The specialty tools are either specializations of the primary tools or tools designed to solve specific problem types.



Multiple model tools are used to combine or produce more than one predictive model. The Ensemble tool is briefly described in a later chapter. The Two Stage tool is discussed in the Advanced Predictive Modeling Using SAS[®] Enterprise Miner[™] course.

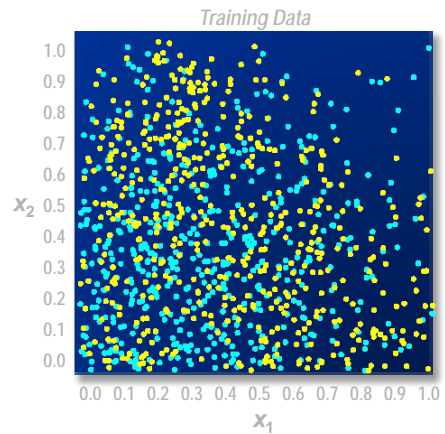


Decision trees provide an excellent introduction to predictive modeling. Decision trees, similar to all modeling methods described in this course, address each of the modeling essentials described in the introduction. Cases are scored using *prediction rules*. A *split-search* algorithm facilitates input selection. Model complexity is addressed by *pruning*.

The following simple prediction problem illustrates each of these model essentials.

Simple Prediction Illustration

Predict dot color
for each x_1 and x_2 .

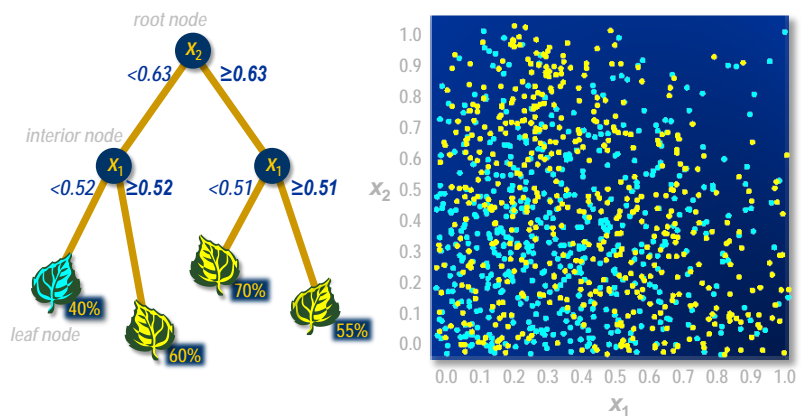


57

...

Consider a data set with two inputs and a binary target. The inputs, x_1 and x_2 , locate the case in the unit square. The target outcome is represented by a color; yellow is primary and blue is secondary. The analysis goal is to predict the outcome based on the location in the unit square.

Decision Tree Prediction Rules

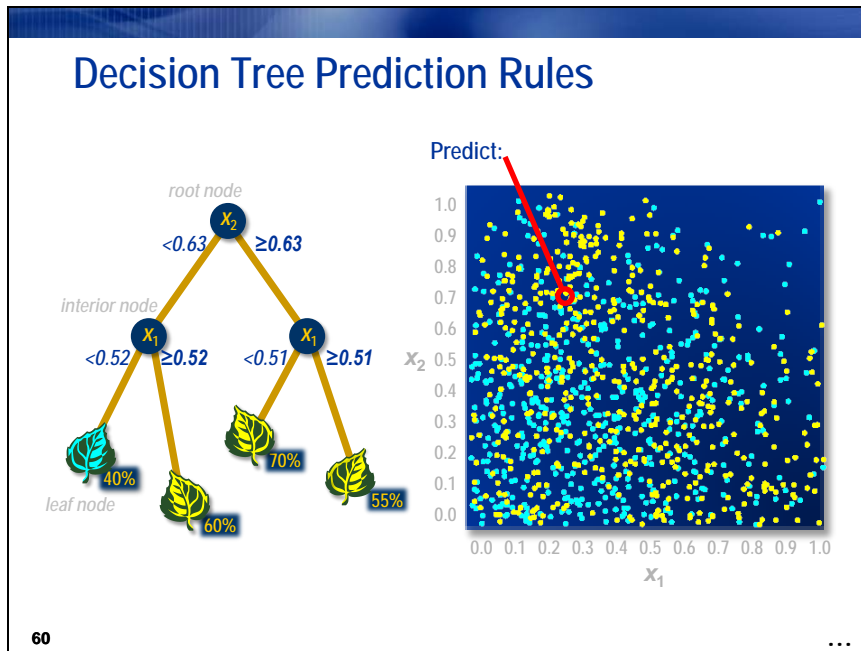


59

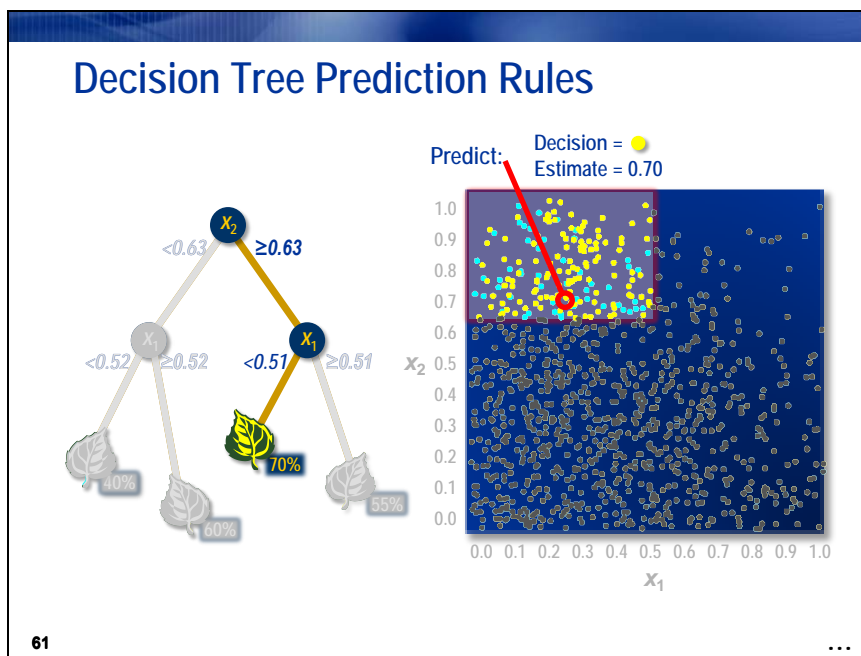
...

Decision trees use rules involving the values of the input variables to predict cases.

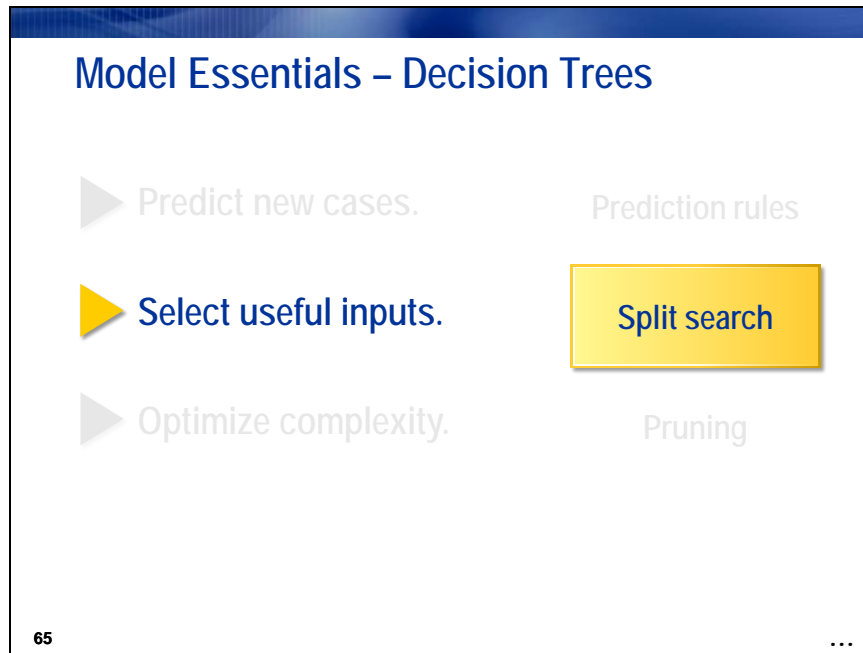
The rules are arranged hierarchically in a tree-like structure with nodes connected by lines. The nodes represent decision rules, and the lines order the rules. The first rule, at the base (top) of the tree, is called the *root node*. Subsequent rules are called *interior nodes*. Nodes with only one connection are called *leaf nodes*.



To score a new case, examine the input values and apply the rules defined by the decision tree.



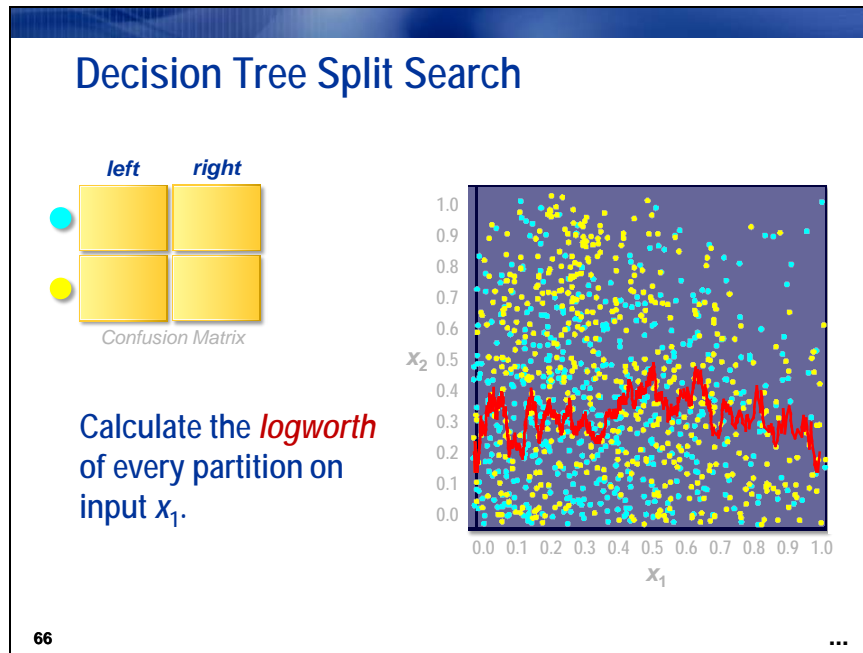
The input values of a new case eventually lead to a single leaf in the tree. A tree leaf provides a decision (for example, classify as yellow) and an estimate (for example, the primary-target proportion).



To select useful inputs, trees use a *split-search* algorithm. Decision trees confront the curse of dimensionality by ignoring irrelevant inputs.



Curiously, trees have no built-in method for ignoring redundant inputs. Because trees can be trained quickly and have simple structure, this is usually not an issue for model creation. However, it can be an issue for model deployment, in that trees might somewhat arbitrarily select from a set of correlated inputs. To avoid this problem, you must use an algorithm that is external to the tree to manage input redundancy.



Understanding the default algorithm for building trees enables you to better use the Tree tool and interpret your results. The description presented here assumes a binary target, but the algorithm for interval targets is similar. (The algorithm for categorical targets with more than two outcomes is more complicated and is not discussed.)

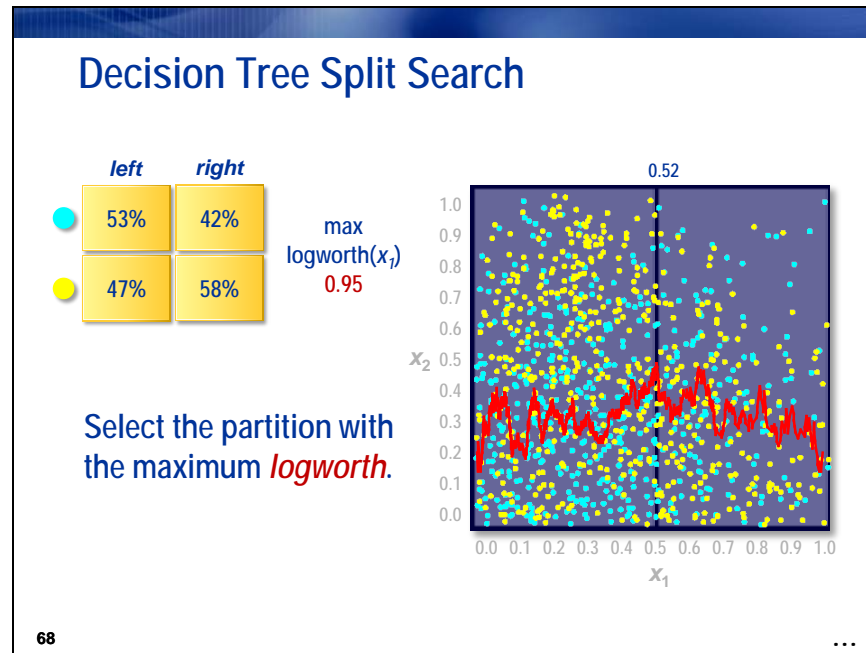
The first part of the algorithm is called the *split search*. The split search starts by selecting an input for partitioning the available training data. If the measurement scale of the selected input is *interval*, each unique value serves as a potential split point for the data. If the input is *categorical*, the average value of the target is taken within each categorical input level. The averages serve the same role as the unique interval input values in the discussion that follows.

For a selected input and fixed split point, two groups are generated. Cases with input values less than the split point are said to *branch left*. Cases with input values greater than the split point are said to *branch right*. The groups, combined with the target outcomes, form a 2x2 contingency table with columns specifying branch direction (left or right) and rows specifying target value (0 or 1). A Pearson chi-squared statistic is used to quantify the independence of counts in the table's columns. Large values for the chi-squared statistic suggest that the proportion of zeros and ones in the left branch is different than the proportion in the right branch. A large difference in outcome proportions indicates a good split.

Because the Pearson chi-squared statistic can be applied to the case of multiway splits and multi-outcome targets, the statistic is converted to a probability value or *p*-value. The *p*-value indicates the likelihood of obtaining the observed value of the statistic assuming identical target proportions in each branch direction. For large data sets, these *p*-values can be very close to zero. For this reason, the quality of a split is reported by $\text{logworth} = -\log(\text{chi-squared } p\text{-value})$.



At least one logworth must exceed a threshold for a split to occur with that input. By default, this threshold corresponds to a chi-squared *p*-value of 0.20 or a logworth of approximately 0.7.



The best split for an input is the split that yields the highest logworth.

Additional Split-Search Details (Self Study)

There are several peripheral factors that make the split search somewhat more complicated than what is described above.

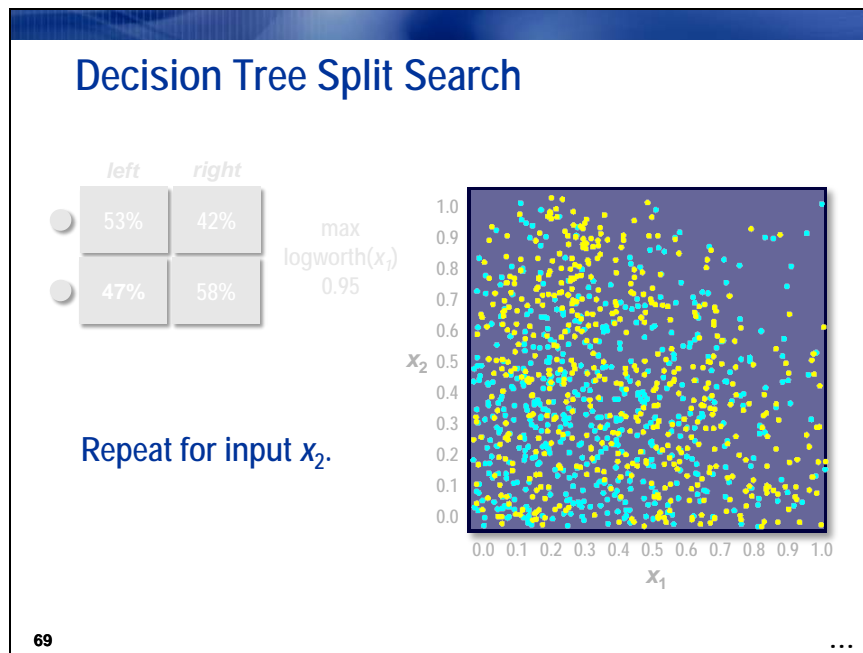
First, the tree algorithm settings disallow certain partitions of the data. Settings, such as the minimum number of observations required for a split search and the minimum number of observations in a leaf, force a minimum number of cases in a split partition. This minimum number of cases reduces the number of potential partitions for each input in the split search.

Second, when you calculate the independence of columns in a contingency table, it is possible to obtain significant (large) values of the chi-squared statistic even when there are no differences in the outcome proportions between split branches. As the number of possible split points increases, the likelihood of obtaining significant values also increases. In this way, an input with a multitude of unique input values has a greater chance of accidentally having a large logworth than an input with only a few distinct input values.

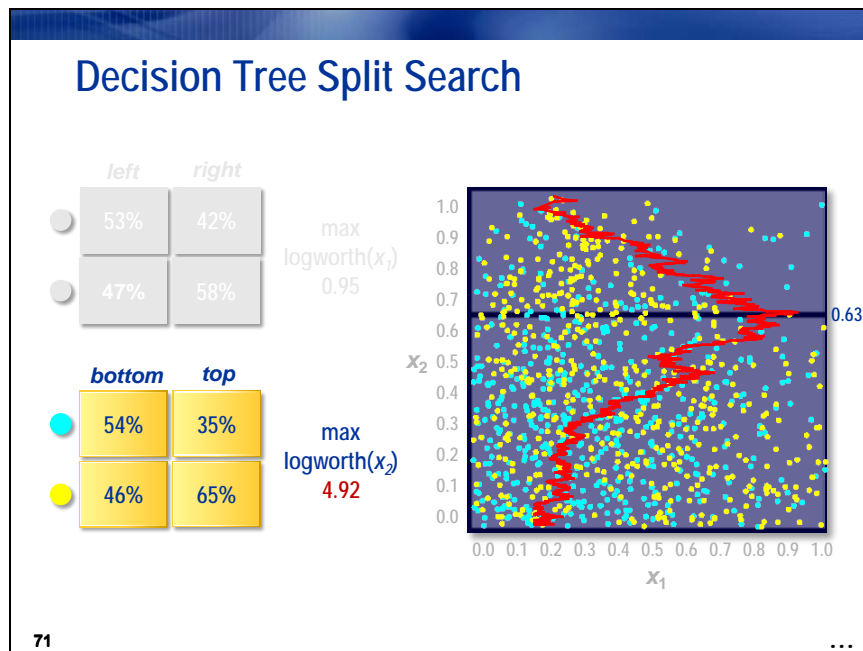
Statisticians face a similar problem when they combine the results from multiple statistical tests. As the number of tests increases, the chance of a false positive result likewise increases. To maintain overall confidence in the statistical findings, statisticians inflate the p -values of each test by a factor equal to the number of tests being conducted. If an inflated p -value shows a significant result, then the significance of the overall results is assured. This type of p -value adjustment is known as a *Bonferroni correction*.

Because each split point corresponds to a statistical test, Bonferroni corrections are automatically applied to the logworth calculations for an input. These corrections, called *Kass adjustments* (named after the inventor of the default tree algorithm used in SAS Enterprise Miner), penalize inputs with many split points by reducing the logworth of a split by an amount equal to the log of the number of distinct input values. This is equivalent to the Bonferroni correction because subtracting this constant from logworth is equivalent to multiplying the corresponding chi-squared p -value by the number of split points. The adjustment enables a fairer comparison of inputs with many and few levels later in the split-search algorithm.

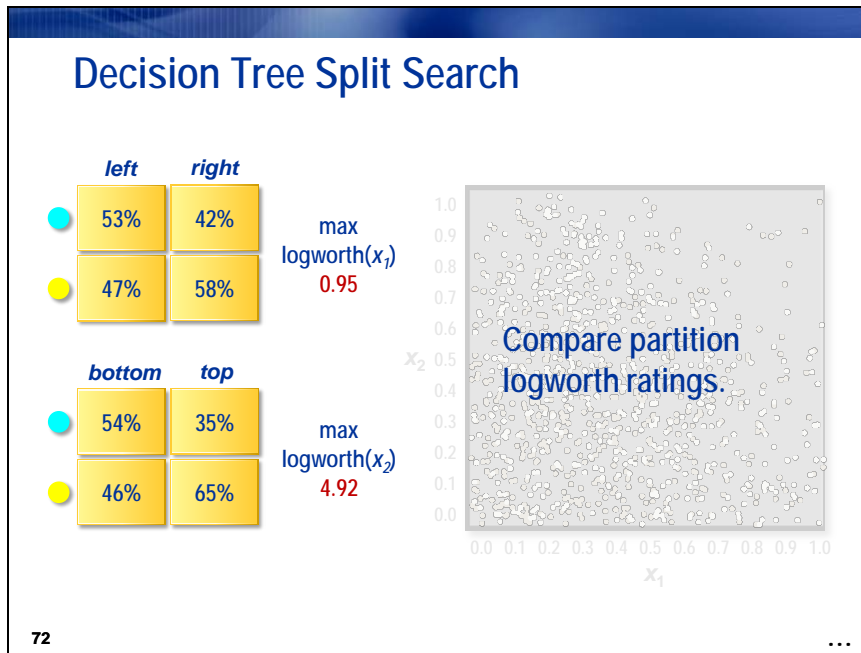
Third, for inputs with missing values, two sets of Kass-adjusted logworths are actually generated. For the first set, cases with missing input values are included in the left branch of the contingency table and logworths are calculated. For the second set of logworths, missing value cases are moved to the right branch. The best split is then selected from the set of possible splits with the missing values in the left and right branches, respectively.



The partitioning process is repeated for every input in the training data. Inputs whose adjusted logworth fails to exceed the threshold are excluded from consideration.



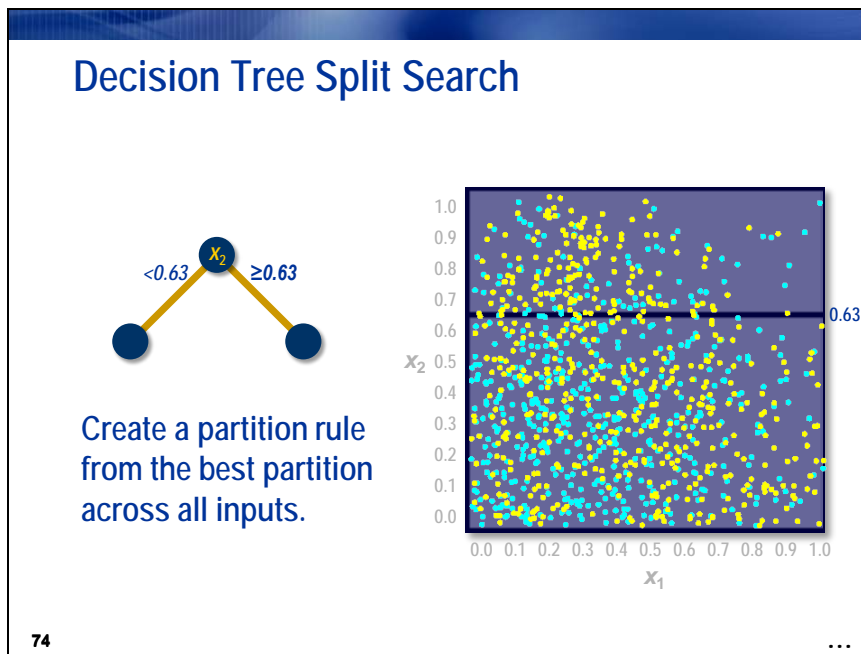
Again, the optimal split for the input is the one that maximizes the logworth function.



72

...

After you determine the best split for every input, the tree algorithm compares each best split's corresponding logworth. The split with the highest adjusted logworth is deemed best.

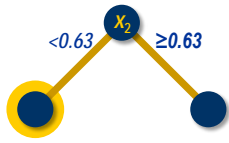


74

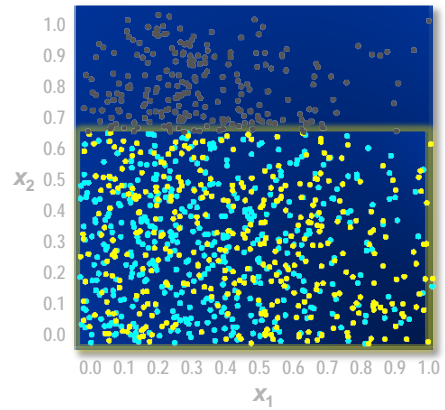
...

The training data is partitioned using the best split rule.

Decision Tree Split Search



Repeat the process
in each subset.



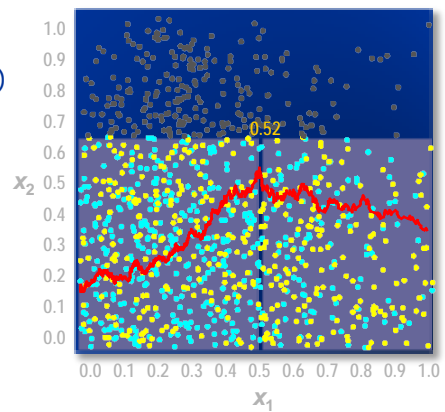
75

...

Decision Tree Split Search

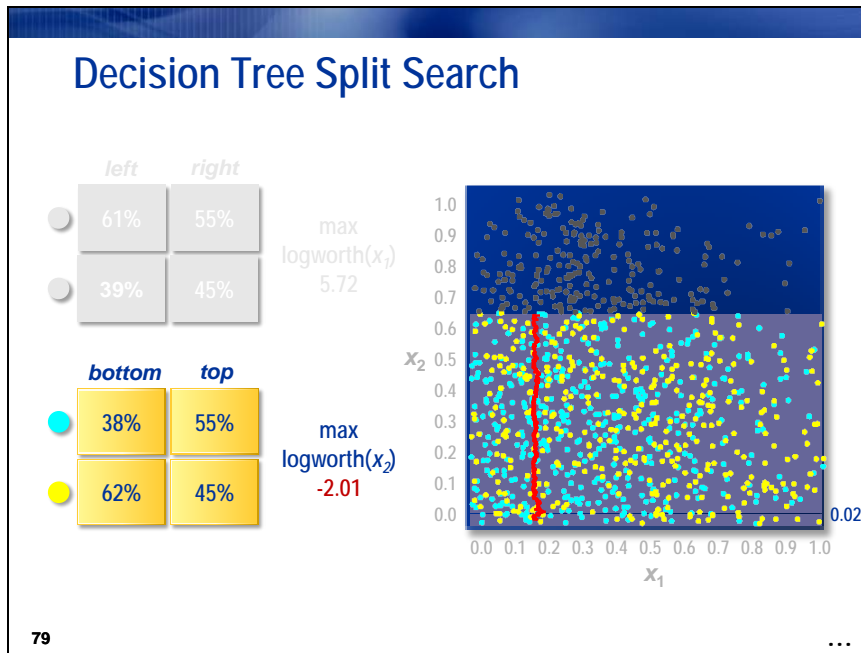
	left	right
●	61%	55%
●	39%	45%

max
logworth(x_1)
5.72



77

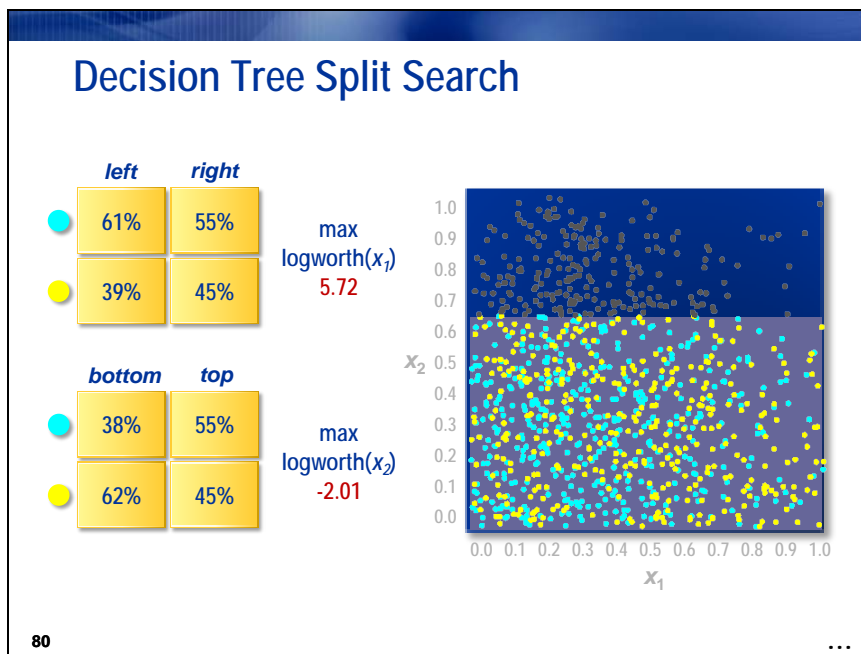
...



79

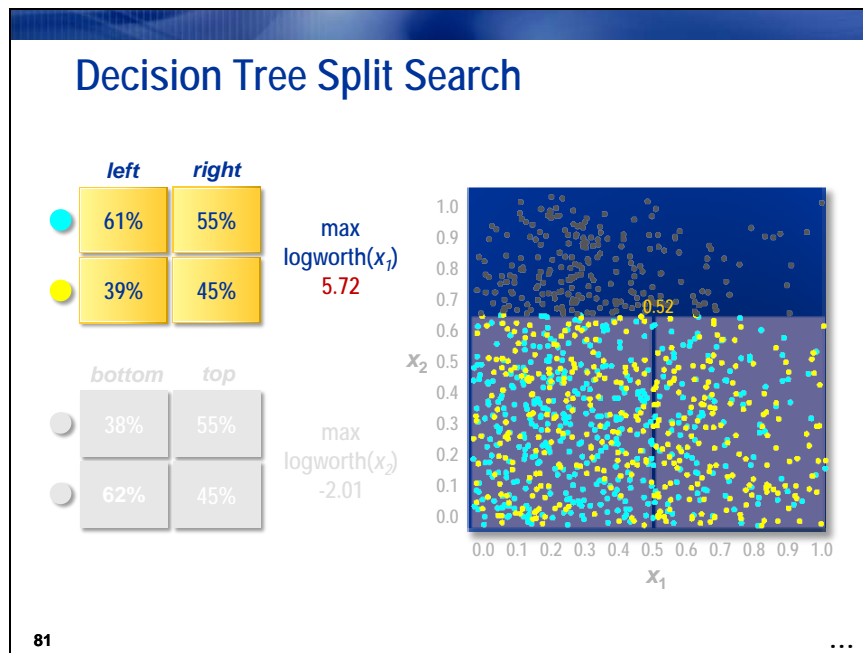
...

The logworth of the x_2 split is negative. This might seem surprising, but it results from several adjustments made to the logworth calculation. (The Kass adjustment was described previously. Another, called the *depth adjustment*, is outlined in the following Self-Study section.)



80

...



The split search continues within each leaf. Logworths are compared as before.

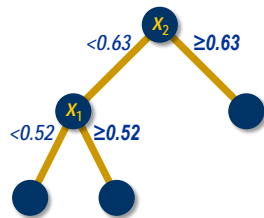
Additional Split-Search Details (Self-Study)

Because the significance of secondary and subsequent splits depends on the significance of the previous splits, the algorithm again faces a multiple comparison problem. To compensate for this problem, the algorithm increases the threshold by an amount related to the number of splits above the current split. For binary splits, the threshold is increased by $\log_{10}(2) \cdot d \approx 0.3 \cdot d$, where d is the depth of the split on the decision tree.

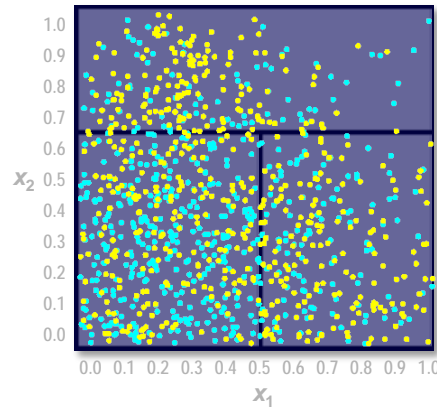


By increasing the threshold for each depth (or equivalently decreasing the logworths), the tree algorithm makes it increasingly easy for an input's splits to be excluded from consideration.

Decision Tree Split Search



Create a second partition rule.

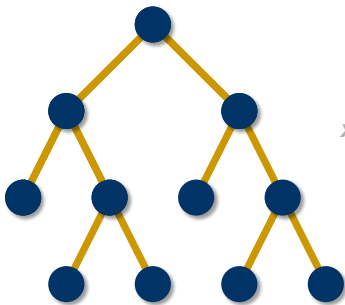


82

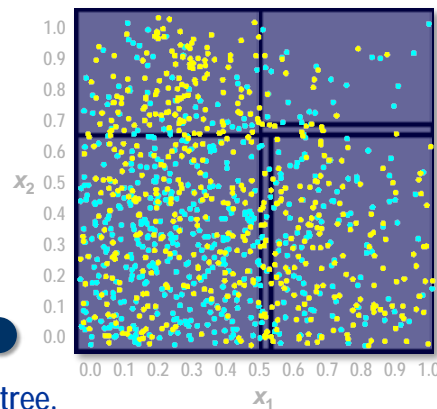
...

The data is partitioned according to the best split, which creates a second partition rule. The process repeats in each leaf until there are no more splits whose adjusted logworth exceeds the depth-adjusted thresholds. This process completes the split-search portion of the tree algorithm.

Decision Tree Split Search



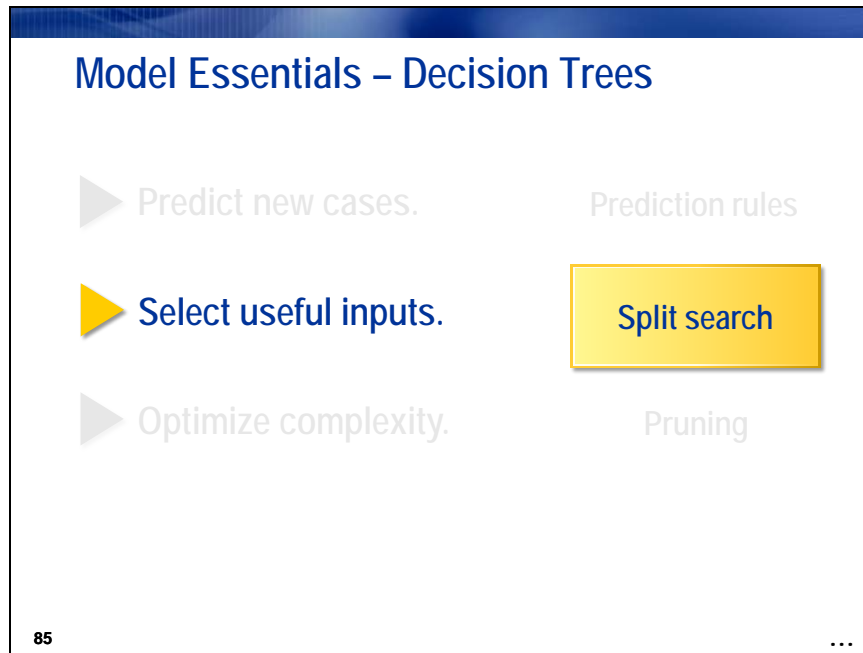
Repeat to form a maximal tree.



84

...

The resulting partition of the input space is known as the *maximal tree*. Development of the maximal tree is based exclusively on statistical measures of split worth on the training data. It is likely that the maximal tree will fail to generalize well on an independent set of validation data.



The split-search algorithm can easily cut through the curse of dimensionality and select useful modeling inputs. Because it is simply an algorithm, many variations are possible. (Some variations are discussed at the end of this chapter.)

The following demonstration illustrates the use of the split-search algorithm to construct a decision tree model.



Constructing a Decision Tree Predictive Model

This four-part demonstration illustrates the interactive construction of a decision tree model and builds on the process flows of previous demonstrations.

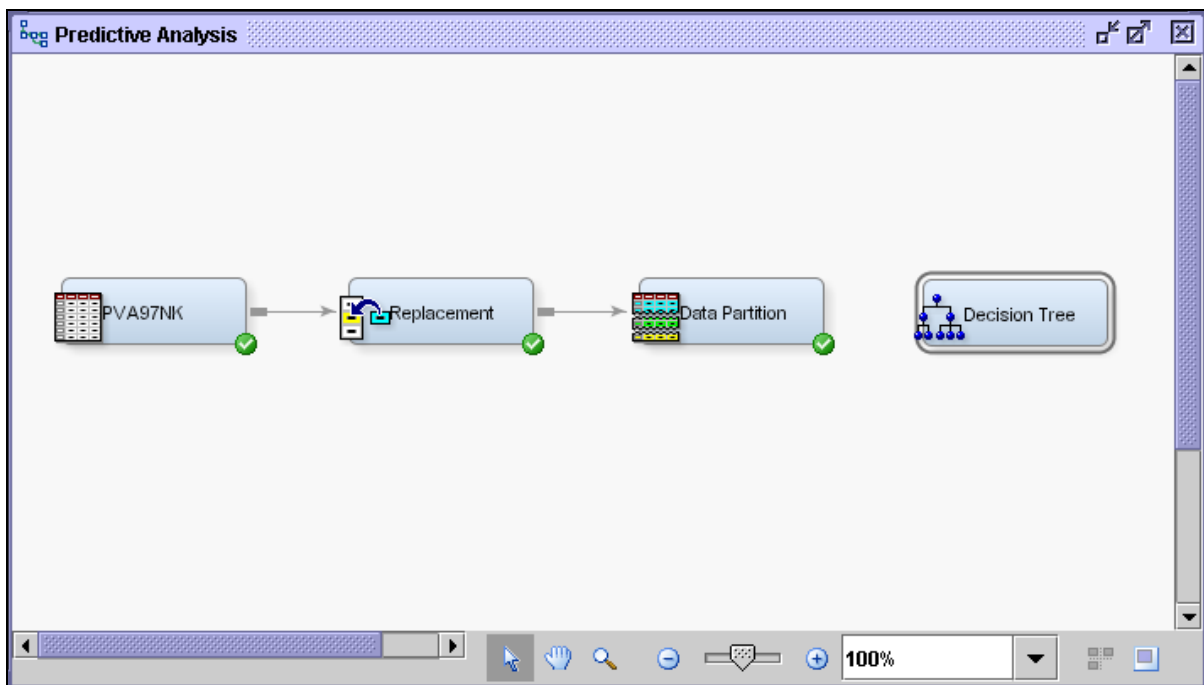
Preparing for Interactive Tree Construction

Follow these steps to prepare the Decision Tree tool for interactive tree construction.

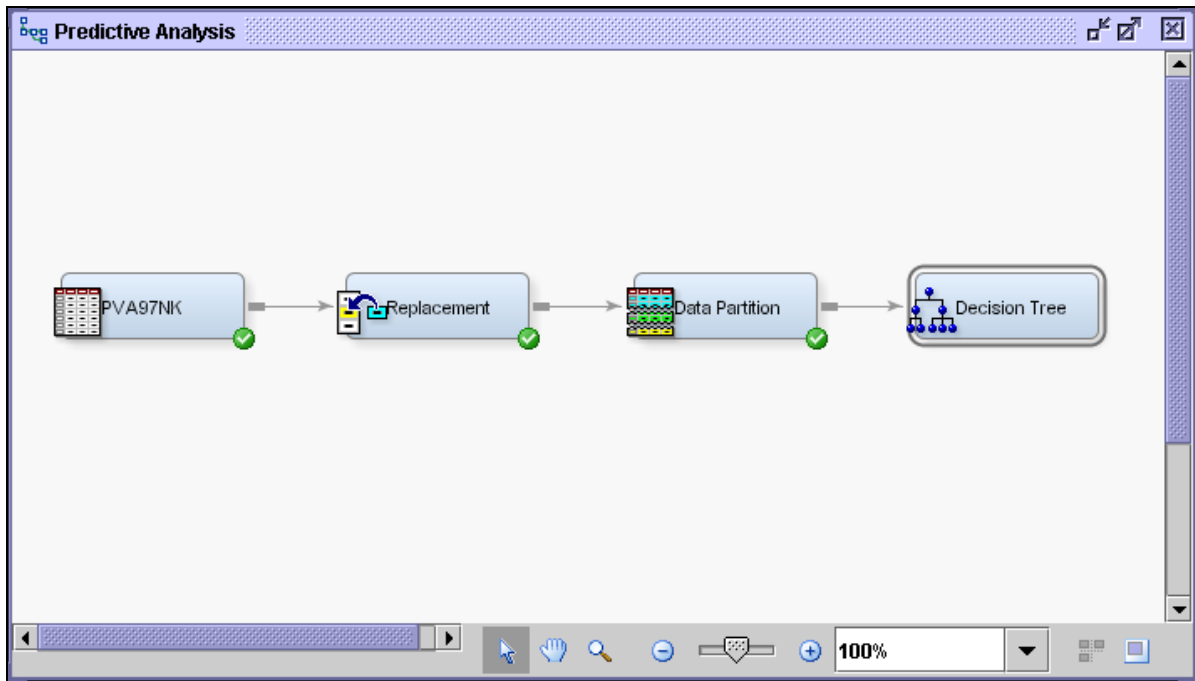
1. Select the **Model** tab. The Decision Tree tool is second from the left.



2. Drag a **Decision Tree** tool into the diagram workspace. Place the node next to the **Data Partition** node.



3. Connect the **Data Partition** node to the **Decision Tree** node.

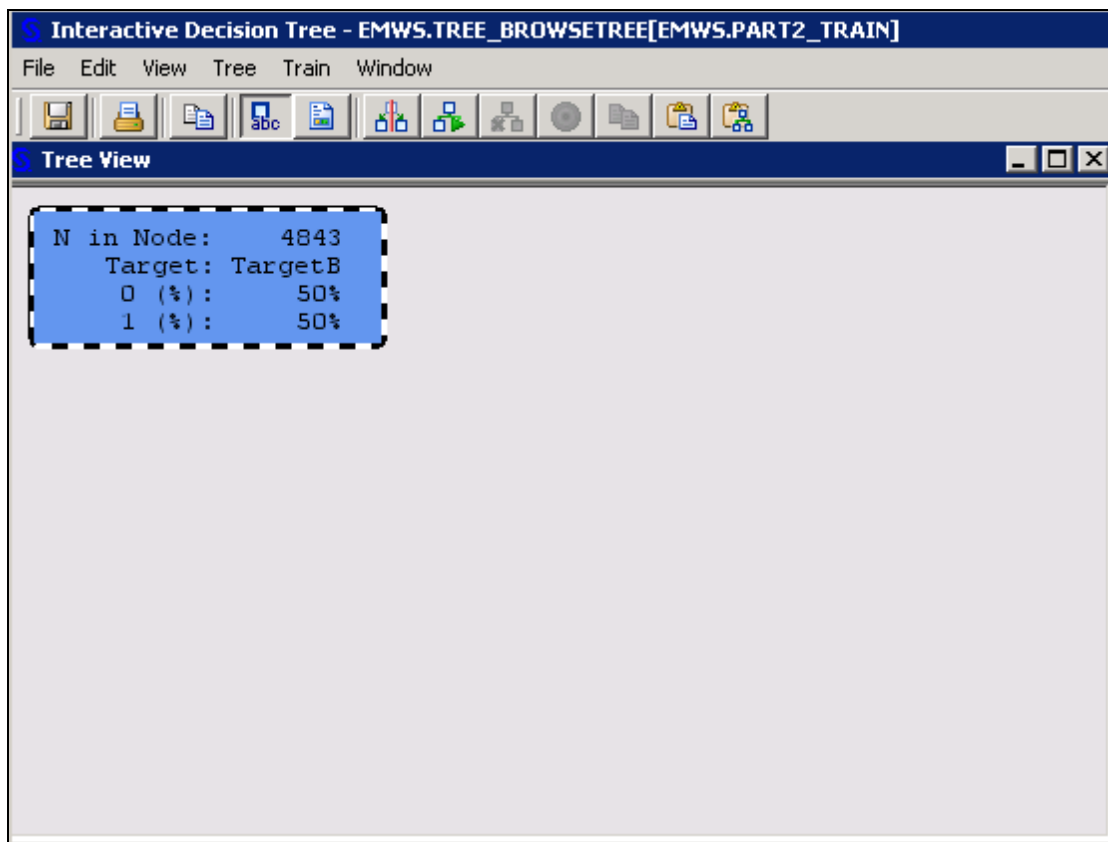


The Decision Tree tool can build predictive models autonomously or interactively. To build a model autonomously, simply run the Decision Tree node. Building models interactively, however, is more informative and is helpful when you first learn about the node, and even when you are more expert at predictive modeling.

4. Select **Interactive** ⇨  from the Decision Tree node's Properties panel.

Property	Value
General	
Node ID	Tree
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Interactive	...
Use Frozen Tree	No
Use Multiple Targets	No
Splitting Rule	
Interval Criterion	ProbF
Nominal Criterion	ProbChisq
Ordinal Criterion	Entropy
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	2
Maximum Depth	6
Minimum Categorical	5
Node	
Leaf Size	5

The SAS Enterprise Miner Interactive Decision Tree application opens.



Creating a Splitting Rule

Decision tree models involve recursive partitioning of the training data in an attempt to isolate concentrations of cases with identical target values. The blue box in the Tree window represents the unpartitioned training data. The statistics show the distribution of **TargetB**.

Use the following steps to create an initial splitting rule:

1. Right-click the purple box and select **Split Node...** from the menu. The Split Node 1 dialog box opens.

Split Node 1

Target Variable: TargetB

Variable	Variable Description	-Log(p)	Branches
GiftCnt36	Gift Count 36 Months	18.44	2
GiftAvgCard36	Gift Amount Average ...	17.24	2
GiftAvgLast	Gift Amount Last	15.02	2
GiftAvg36	Gift Amount Average ...	14.63	2
GiftAvgAll	Gift Amount Average ...	13.95	2
GiftCntCard36	Gift Count Card 36 M...	12.74	2
GiftCntCardAll	Gift Count Card All Mo...	11.91	2
StatusCatStarAll	Status Category Star ...	10.85	2
StatusCat96NK	Status Category 96NK	9.52	2
GiftCntAll	Gift Count All Months	9.28	2
PromCntCard36	Promotion Count Card...	8.73	2
GiftTimeFirst	Time Since First Gift	6.69	2
GiftTimeLast	Time Since Last Gift	5.70	2
PromCntAll	Promotion Count All M...	5.49	2
PromCntCardAll	Promotion Count Card...	4.59	2
DemMedHomeValue	Median Home Value R...	4.46	2
PromCntCard12	Promotion Count Card...	3.87	2
PromCnt36	Promotion Count 36 M...	3.35	2
PromCnt12	Promotion Count 12 M...	2.61	2
DemPctVeterans	Percent Veterans Reg...	1.99	2
DemAge	Age	1.51	2
REP_DemMedIncome	Replacement: DemMe...	1.47	2
DemHomeOwner	Home Owner	0.40	2

Buttons: Apply, Edit Rule..., OK, Cancel

The Split Node dialog box shows the relative value, $-\text{Log}(p)$ or *logworth*, of partitioning the training data using the indicated input. As the logworth increases, the partition better isolates cases with identical target values.

Gift Count 36 Months has the highest logworth, followed by **Gift Amount Average Card 36 Months** and **Gift Amount Last**. You can choose to split the data on the selected input or edit the rule for more information.

2. Select **Edit Rule...**. The GiftCnt36 - Interval Splitting Rule dialog box opens.

GiftCnt36 - Interval Splitting Rule

Target Variable: TargetB

Assign missing values to:

☒ A specific branch 2

☐ A separate missing values branch

☐ All branches

Branches

Branch		Split Point
1	<	2.5
2	>=	2.5

New split point:

Add Branch Remove Branch

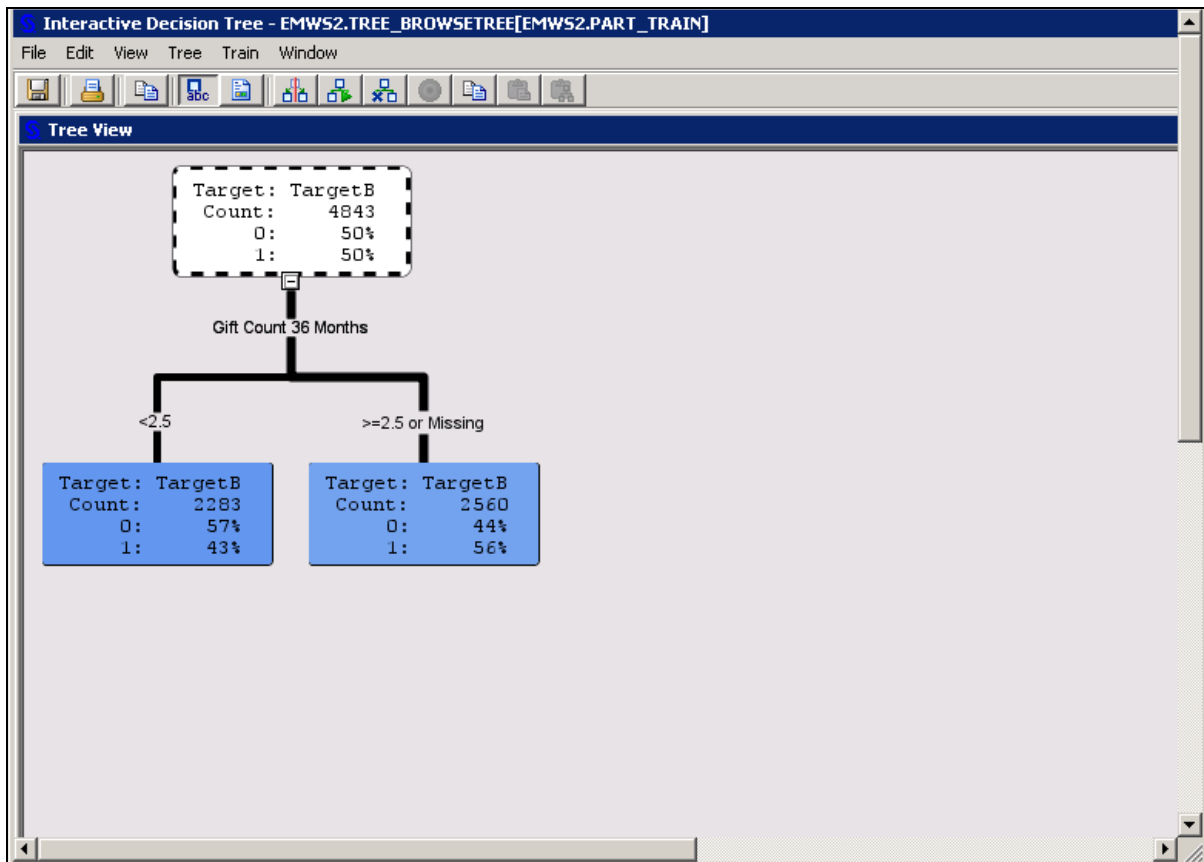
OK Cancel Apply Undo Reset

This dialog box shows how the training data is partitioned using the input **Gift Count 36 Months**. Two branches are created. The first branch contains cases with a 36-month gift count less than 2.5, and the second branch contains cases with a 36-month gift count greater than or equal to 2.5. In other words, with cases that have a 36-month gift count of zero, one or two branch left and three or more branch right. In addition, any cases with a missing or unknown 36-month gift count are placed in the first branch.



The interactive tree assigns any cases with missing values to the branch that maximizes the purity or logworth of the split by default. For **GiftCnt36**, this rule assigns cases with missing values to branch 2.

3. Select **Apply**. The GiftCnt36 - Interval Splitting Rule dialog box remains open, and the Tree View window is updated.



The training data is partitioned into two subsets. The first subset, corresponding to cases with a 36-month gift count less than 2.5, has a higher than average concentration of **TargetB=0** cases. The second subset, corresponding to cases with a 36-month gift count greater than 2.5, has a higher than average concentration of **TargetB=1** cases. The second branch has slightly more cases than the first, which is indicated by the N in node field.

The partition of the data also represents your first (non-trivial) predictive model. This model assigns to all cases in the left branch a predicted **TargetB** value equal to 0.43 and to all cases in the right branch a predicted **TargetB** value equal to 0.56.



In general, decision tree predictive models assign all cases in a leaf the same predicted target value. For binary targets, this equals the percentage of cases in the target variable's primary outcome (usually the **target=1** outcome).

Adding More Splits

Use the following steps to interactively add more splits to the decision tree:

1. Select the lower left partition. The Split Node dialog box is updated to show the possible partition variables and their respective logworths.

Split Node 3 [X]

Target Variable: TargetB

Variable	Variable Description	-Log(p)	Branches
DemMedHomeValue	Median Home Value Region	3.45360	2
DemPctVeterans	Percent Veterans Region	2.38360	2
StatusCatStarAll	Status Category Star All M...	1.81739	2
REP_DemMedIncome	Replacement: Median Inco...	1.56106	2
DemAge	Age	1.24470	2
GiftAvgAll	Gift Amount Average All Mo...	1.08539	2
GiftCnt36	Gift Count 36 Months	0.92244	2
PromCntCard12	Promotion Count Card 12 M...	0.88959	2
StatusCat96NK	Status Category 96NK	0.76545	2
GiftTimeFirst	Time Since First Gift	0.68583	2
DemHomeOwner	Home Owner	0.66592	2
GiftCntCardAll	Gift Count Card All Months	0.58917	2
GiftCntAll	Gift Count All Months	0.41340	2
PromCnt36	Promotion Count 36 Months	0.38333	2
PromCntCardAll	Promotion Count Card All M...	0.26385	2
GiftAvg36	Gift Amount Average 36 M...	0.19283	2
DemGender	Gender	0.15916	2
PromCntAll	Promotion Count All Months	0.13244	2
GiftCntCard36	Gift Count Card 36 Months	0.03293	2
GiftTimeLast	Time Since Last Gift	0.01715	2
GiftAvgCard36	Gift Amount Average Card ...	0.00000	2
GiftAvgLast	Gift Amount Last	0.00000	2
PromCnt12	Promotion Count 12 Months	0.00000	2
PromCntCard36	Promotion Count Card 36 M...	0.00000	2
DemCluster	Demographic Cluster	0.00000	2

Edit Rule...

OK Cancel Apply Refresh

The input with the highest logworth is **Median Home Value Region**.

2. Select **Edit Rule...**. The DemMedHomeValue - Interval Splitting Rule dialog box opens.

Target Variable: TargetB

Assign missing values to:

- ☒ A specific branch 2
- ☐ A separate missing values branch
- ☐ All branches

Branches

Branch		Split Point
1	<	67350
2	>=	67350

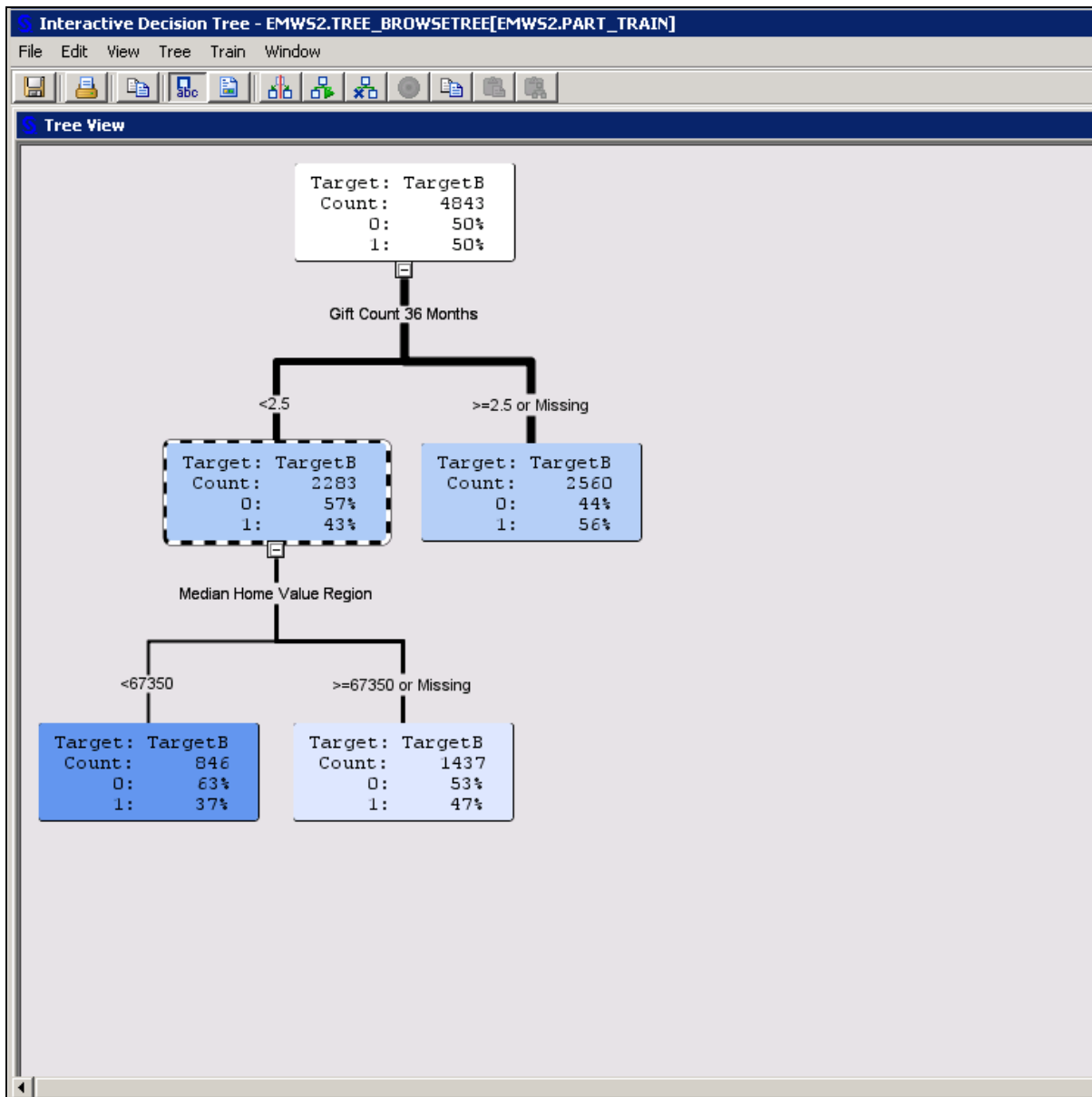
New split point:

Add Branch Remove Branch

OK Cancel Apply Undo Reset

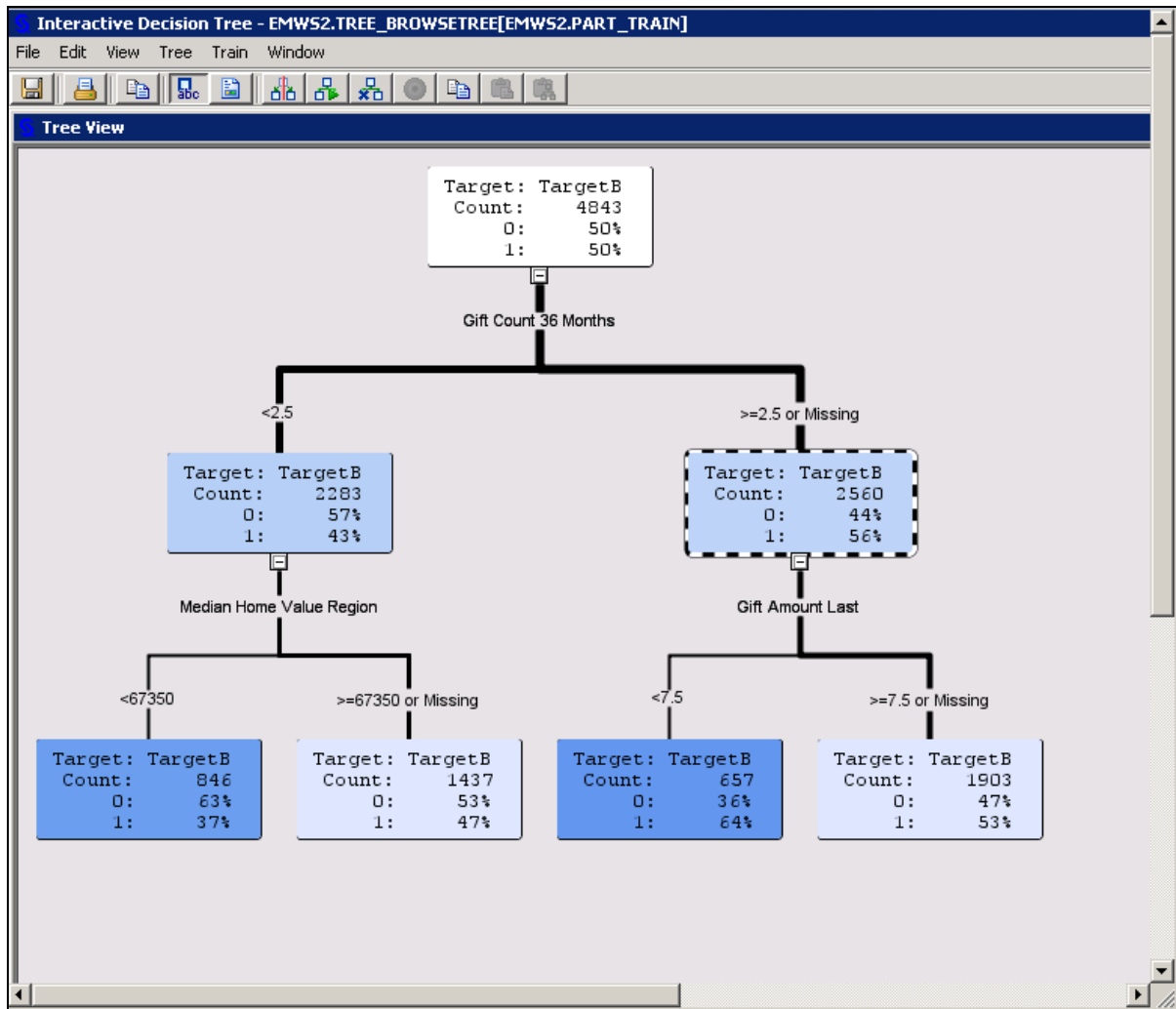
Branch 1 contains all cases with a median home value less than 67350. Branch 2 contains all cases that are equal to or greater than 67350.

3. Select **Apply**. The Tree View window is updated to show the additional split.



Both left and right leaves contain a lower-than-average proportion of cases with **TargetB=1**.

4. Repeat the process for the branch that corresponds to cases with **Gift Count 36 Month** in excess of 2.5.



The right-branch cases are split using the **Gift Amount Last** input. This time, both branches have a higher-than-average concentration of **TargetB=1** cases.



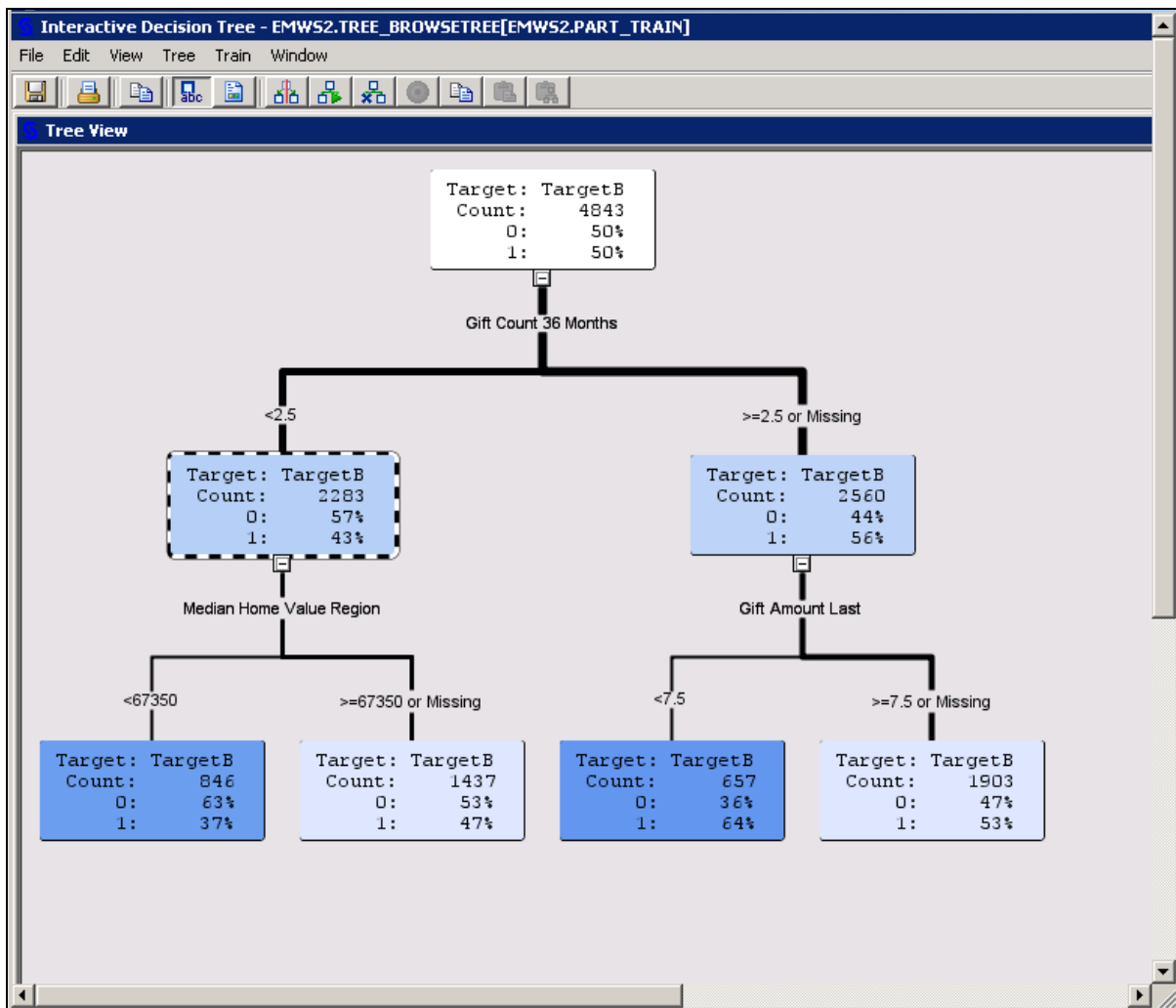
It is important to remember that the logworth of each split, reported above, and the predicted **TargetB** values are generated using the **training data only**. The main focus is on selecting useful input variables for the first predictive model. A diminishing marginal usefulness of input variables is expected, given the split-search discussion at the beginning of this chapter. This prompts the question: Where do you stop splitting? The validation data provides the necessary information to answer this question.

Changing a Splitting Rule

The bottom left split on **Median Home Value** at 67350 was found to be optimal in a statistical sense, but it might seem strange to someone simply looking at the tree. (Why not split at the more socially acceptable value of 70000?) You can use the Splitting Rule window to edit where a split occurs.

Use the following steps to define your own splitting rule for an input.

1. Select the node above the label **Median Home Value Region**.



2. Right-click this node and select **Split Node...** from the menu. The Split Node window opens.

Split Node 3 [X]

Target Variable: TargetB

Variable	Variable Description	-Log(p)	Branches
DemMedHomeValue	Median Home Value Region	3.36085	2
DemPctVeterans	Percent Veterans Region	2.38360	2
StatusCatStarAll	Status Category Star All M...	1.81739	2
REP_DemMedIncome	Replacement: Median Inco...	1.56106	2
DemAge	Age	1.24470	2
GiftAvgAll	Gift Amount Average All Mo...	1.08539	2
GiftCnt36	Gift Count 36 Months	0.92244	2
PromCntCard12	Promotion Count Card 12 M...	0.88959	2
StatusCat96NK	Status Category 96NK	0.76545	2
GiftTimeFirst	Time Since First Gift	0.68583	2
DemHomeOwner	Home Owner	0.66592	2
GiftCntCardAll	Gift Count Card All Months	0.58917	2
GiftCntAll	Gift Count All Months	0.41340	2
PromCnt36	Promotion Count 36 Months	0.38333	2
PromCntCardAll	Promotion Count Card All M...	0.26385	2
GiftAvg36	Gift Amount Average 36 M...	0.19283	2
DemGender	Gender	0.15916	2
PromCntAll	Promotion Count All Months	0.13244	2
GiftCntCard36	Gift Count Card 36 Months	0.03293	2
GiftTimeLast	Time Since Last Gift	0.01715	2
GiftAvgCard36	Gift Amount Average Card ...	0.00000	2
GiftAvgLast	Gift Amount Last	0.00000	2
PromCnt12	Promotion Count 12 Months	0.00000	2
PromCntCard36	Promotion Count Card 36 M...	0.00000	2
DemCluster	Demographic Cluster	0.00000	2

Edit Rule...

OK Cancel Apply Refresh

3. Select **Edit Rule...**. The DemMedHomeValue - Interval Splitting Rule dialog box opens.

DemMedHomeValue - Interval Splitting Rule

Target Variable: TargetB

Assign missing values to:

- ☒ A specific branch 2
- ☐ A separate missing values branch
- ☐ All branches

Branches

Branch		Split Point
1	<	67350
2	>=	67350

New split point:

4. Type **70000** in the New split point field.

DemMedHomeValue - Interval Splitting Rule

Target Variable: TargetB

Assign missing values to:

☒ A specific branch 2

☐ A separate missing values branch

☐ All branches

Branches

Branch		Split Point
1	<	67350
2	>=	67350

New split point:

5. Select **Add Branch**. The Interval Splitting Rule dialog box shows three branches.

DemMedHomeValue - Interval Splitting Rule

Target Variable: TargetB

Assign missing values to:

☒ A specific branch 2

☐ A separate missing values branch

☐ All branches

Branches

Branch		Split Point
1	<	67350
2	<	70000
3	>=	70000

New split point:

Add Branch Remove Branch

OK Cancel Apply Undo Reset

6. Select **Branch 1** by highlighting the row.

DemMedHomeValue - Interval Splitting Rule

Target Variable: TargetB

Assign missing values to:

☒ A specific branch 2

☐ A separate missing values branch

☐ All branches

Branches

Branch		Split Point
1	<	67350
2	<	70000
3	>=	70000

New split point:

7. Select **Remove Branch**.

DemMedHomeValue - Interval Splitting Rule

Target Variable: TargetB

Assign missing values to:

☒ A specific branch 1

☐ A separate missing values branch

☐ All branches

Branches

Branch		Split Point
1	<	70000
2	>=	70000

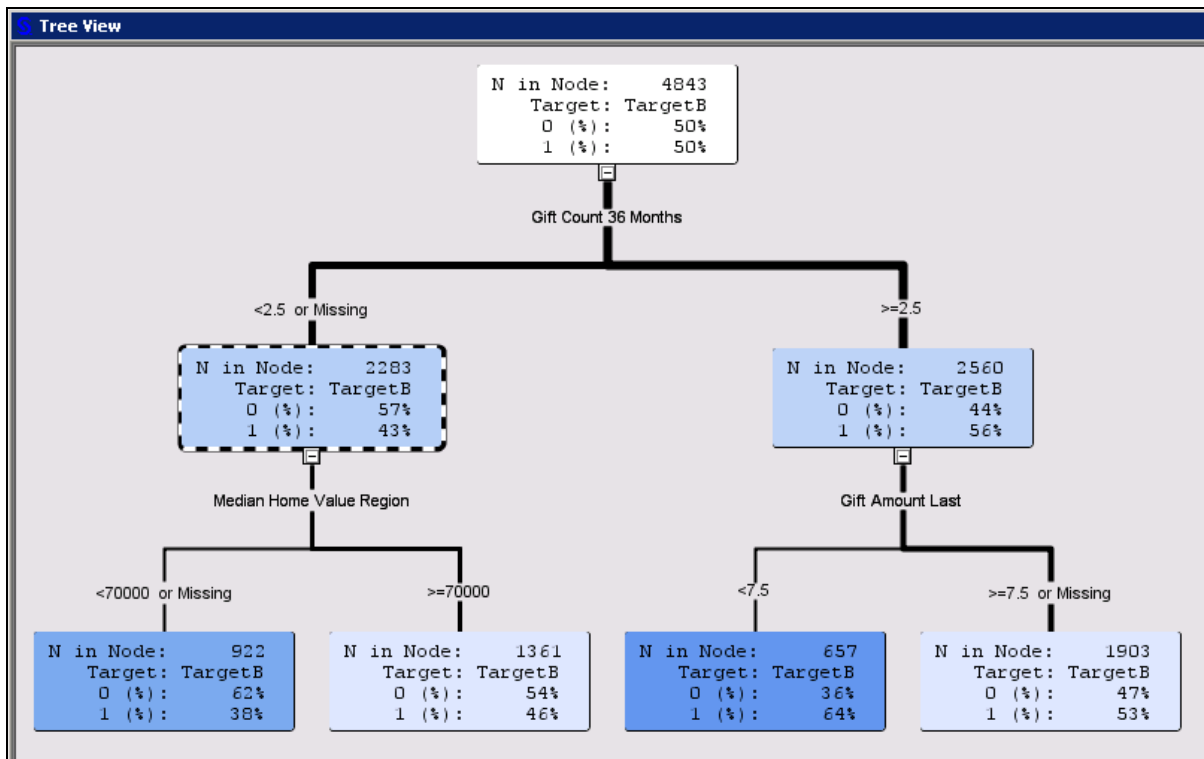
New split point:

The split point for the partition is moved to 70000.



In general, to change a Decision Tree split point, add the new split point first and then remove the unwanted split point.

8. Select **OK** to close the Interval Splitting Rule dialog box and to update the Tree View window.



Overriding the statistically optimal split point slightly changed the concentration of cases in both nodes.

Creating the Maximal Tree

As demonstrated thus far, the process of building the decision tree model is one of carefully considering and selecting each split sequentially. There are other, more automated ways to grow a tree.

Follow these steps to automatically grow a tree with the SAS Enterprise Miner Interactive Tree tool:

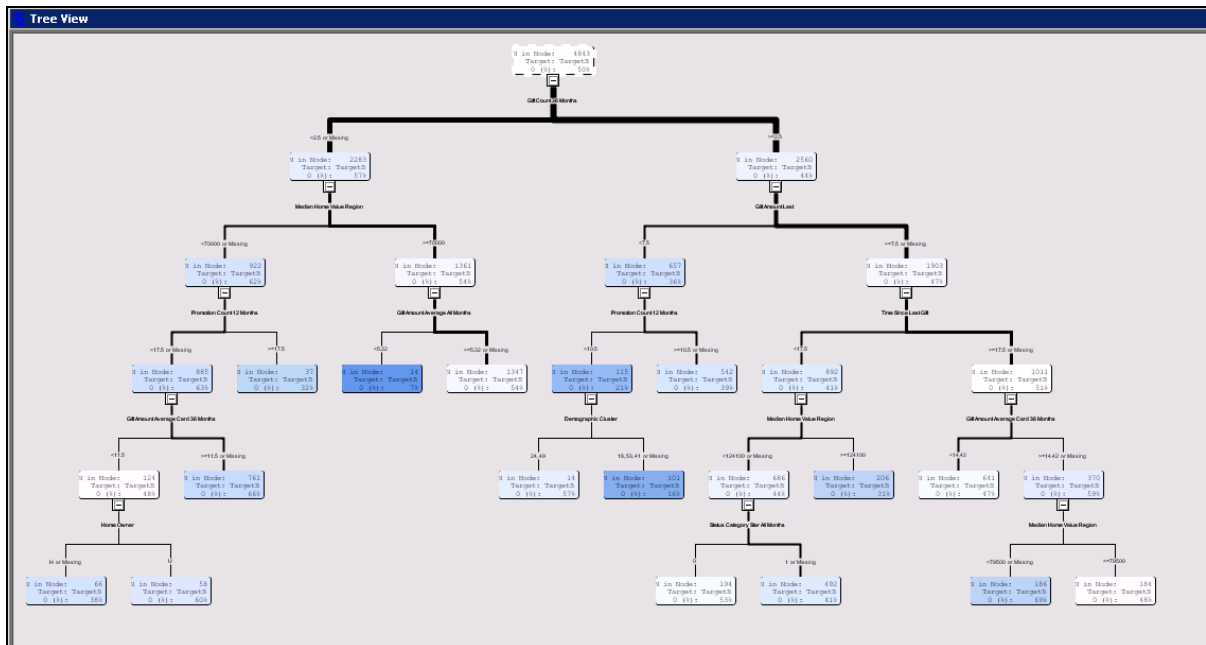
1. Select the root node of the tree.
2. Right-click and select **Train Node** from the menu. The tree is grown until stopping rules prohibit further growth.



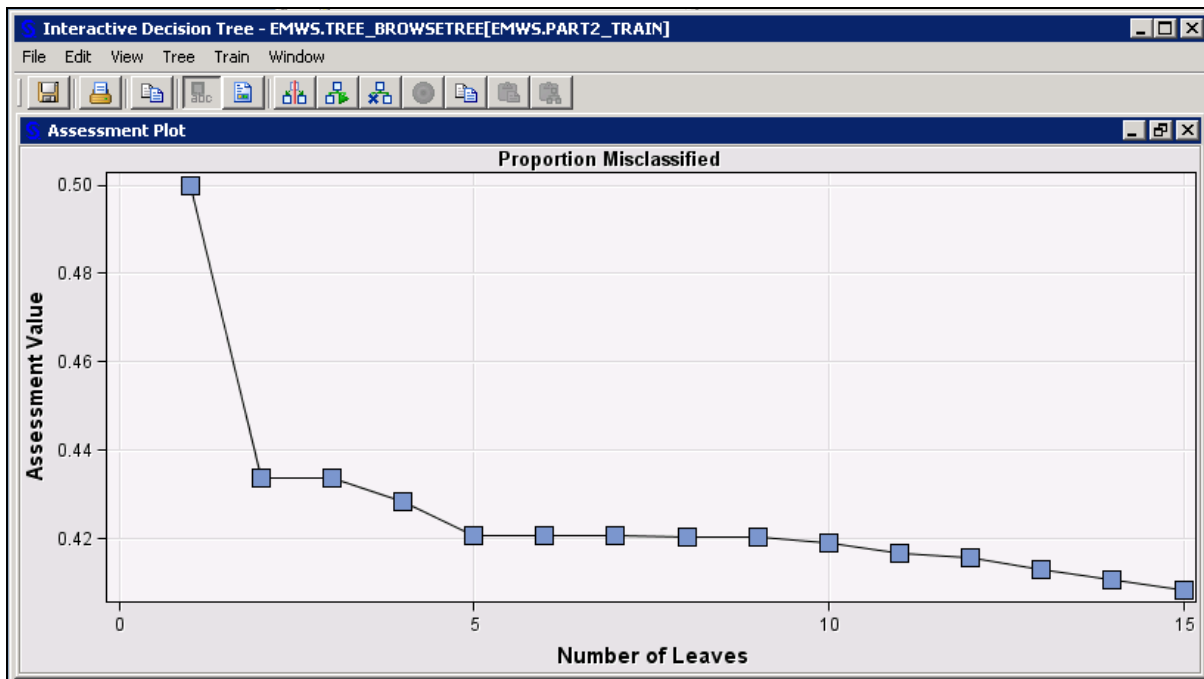
Stopping rules were discussed previously and in the Self-Study section at the end of this chapter.

It is difficult to see the entire tree without scrolling. However, you can zoom into the Tree View window to examine the basic shape of the tree.

3. Select **Options** ⇒ **Zoom** ⇒ **50%** from the main menu. The tree view is scaled to provide the general shape of the maximal tree.



Plots and tables contained in the Interactive Tree tool provide a preliminary assessment of the maximal tree.

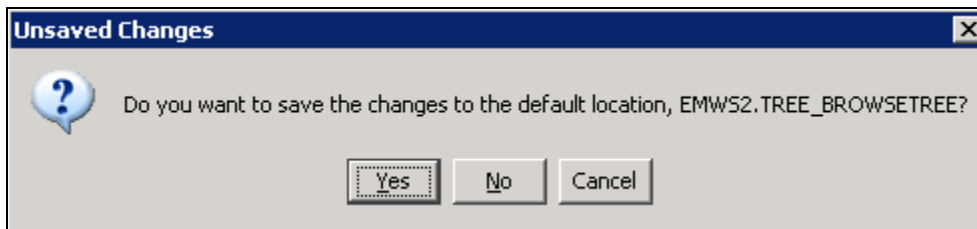
4. Select **View** ⇒ **Assessment Plot**.

While the majority of the improvement in fit occurs over the first few splits, it appears that the maximal, fifteen-leaf tree generates a lower misclassification rate than any of its simpler predecessors. The plot seems to indicate that the maximal tree is preferred for assigning predictions to cases. However, this plot is misleading.

Recall that the basis of the plot is the validation data. Using the same sample of data both to evaluate input variable usefulness and to assess model performance commonly leads to overfit models. This is the mistake that the Mosteller and Tukey quotation cautioned you about.

You created a predictive model that assigns one of 15 predicted target values to each case. Your next task is to investigate how well this model generalizes to another similar, but independent, sample: the validation data.

Select **File** ⇒ **Exit** to close the Interactive Tree application. A dialog box opens. You are asked whether you want to save the maximal tree to the default location.



Select **Yes**. The Interactive Tree results are now available from the Decision Tree node in the SAS Enterprise Miner flow.

3.3 Optimizing the Complexity of Decision Trees

Model Essentials – Decision Trees

- Predict new cases. Prediction rules
- Select useful inputs. Split search
- Optimize complexity. Pruning**


88 ...

The maximal tree represents the most complicated model you are willing to construct from a set of training data. To avoid potential overfitting, many predictive modeling procedures offer some mechanism for adjusting model complexity. For decision trees, this process is known as *pruning*.

Predictive Model Sequence

Training Data			
	Inputs		target

Validation Data			
	Inputs		target



1

2

3

4

5

6

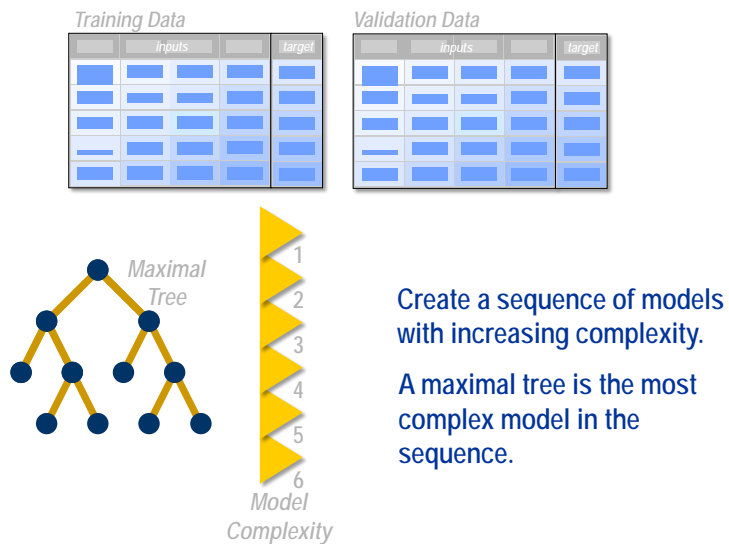
Create a sequence of models with increasing complexity.

Model Complexity

89 ...

The general principle underlying complexity optimization is creating a sequence of related models of increasing complexity from training data and using validation data to select the optimal model.

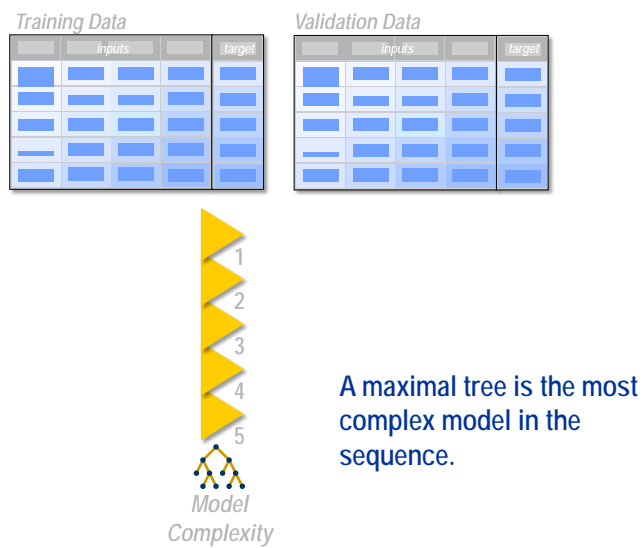
The Maximal Tree



90

...

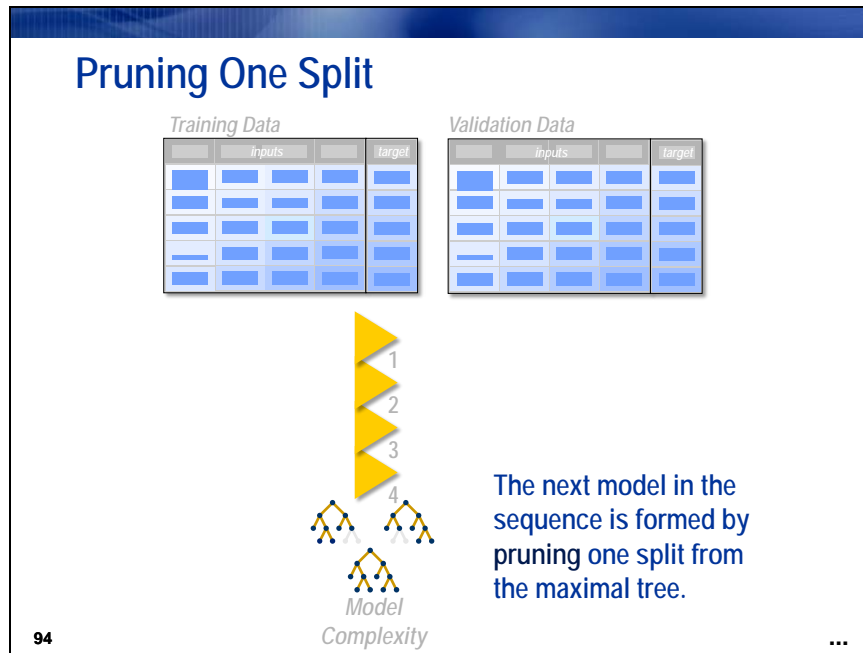
The Maximal Tree



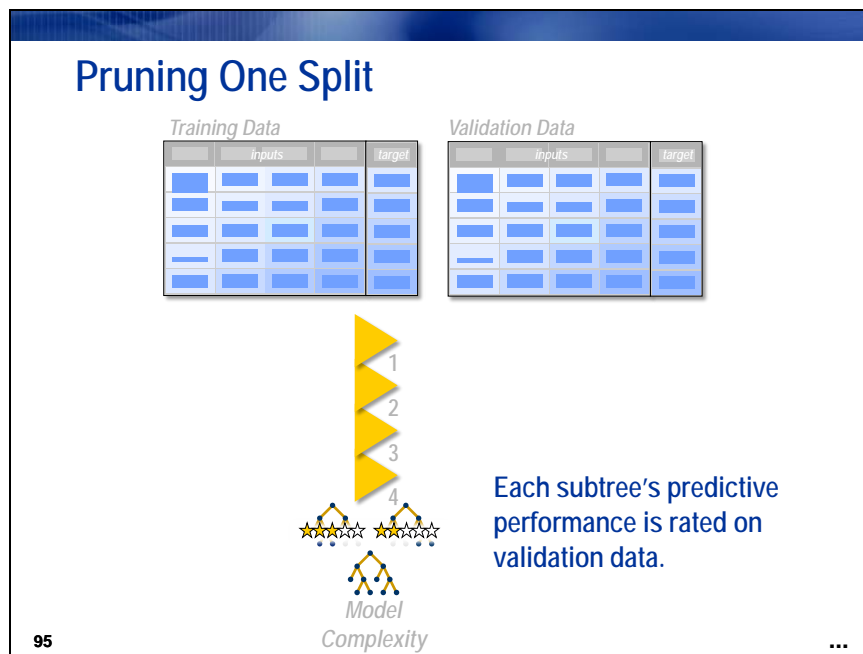
92

...

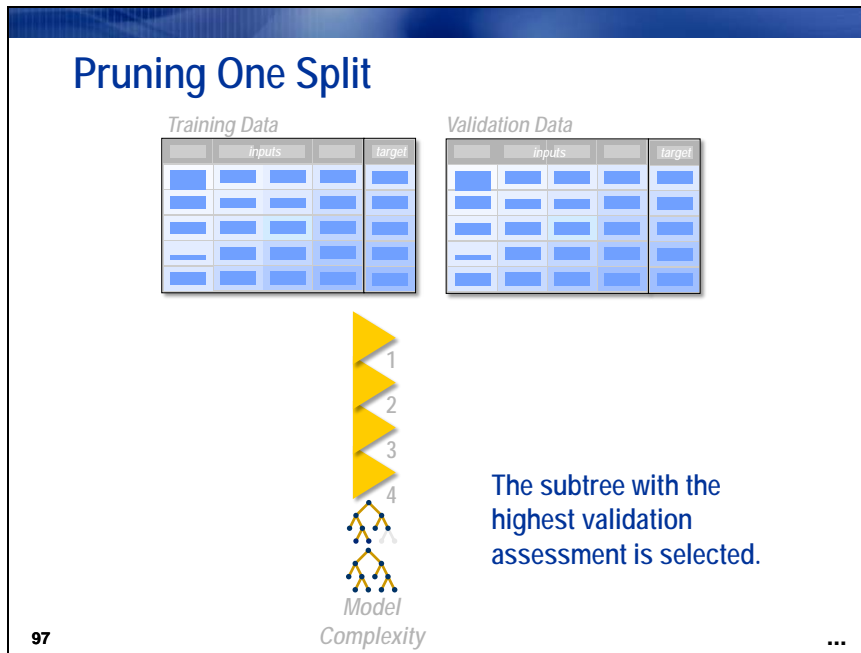
The maximal tree that you generated earlier represents the most complex model in the sequence.



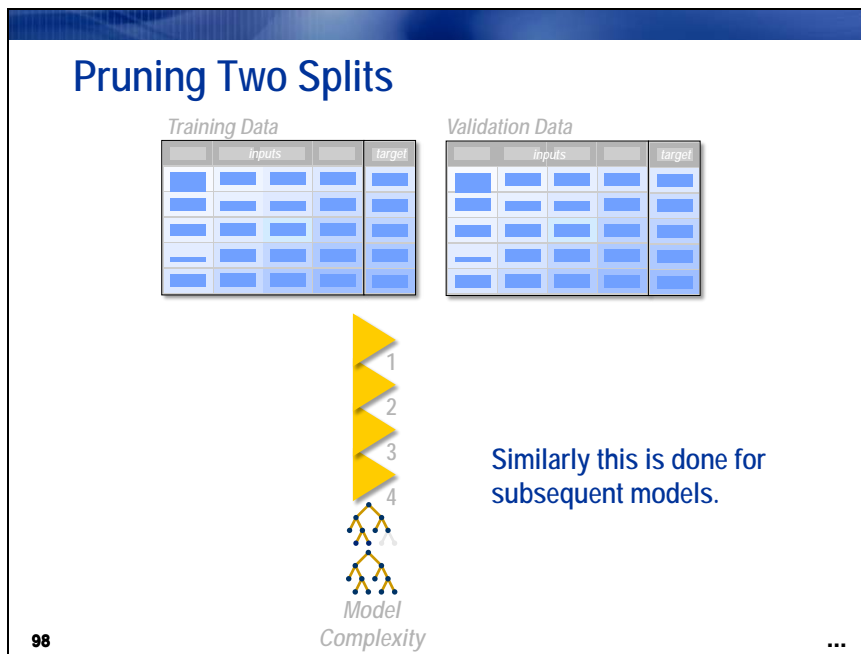
The predictive model sequence consists of *subtrees* obtained from the maximal tree by removing splits. The first batch of subtrees is obtained by *pruning* (that is, removing) one split from the maximal tree. This process results in several models with one less leaf than the maximal tree.



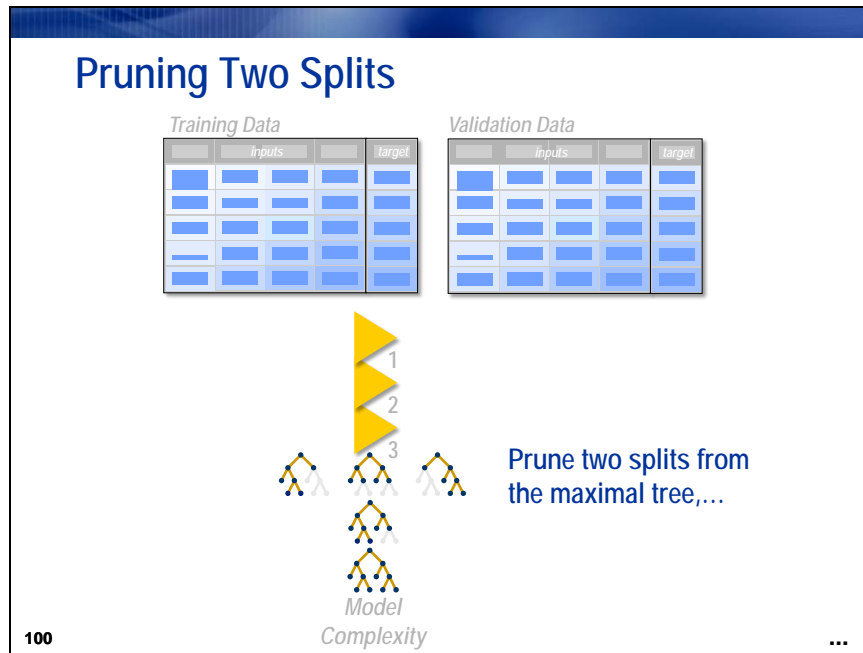
The subtrees are compared using a model rating statistic calculated on validation data. (Choices for model fit statistics are discussed later.)



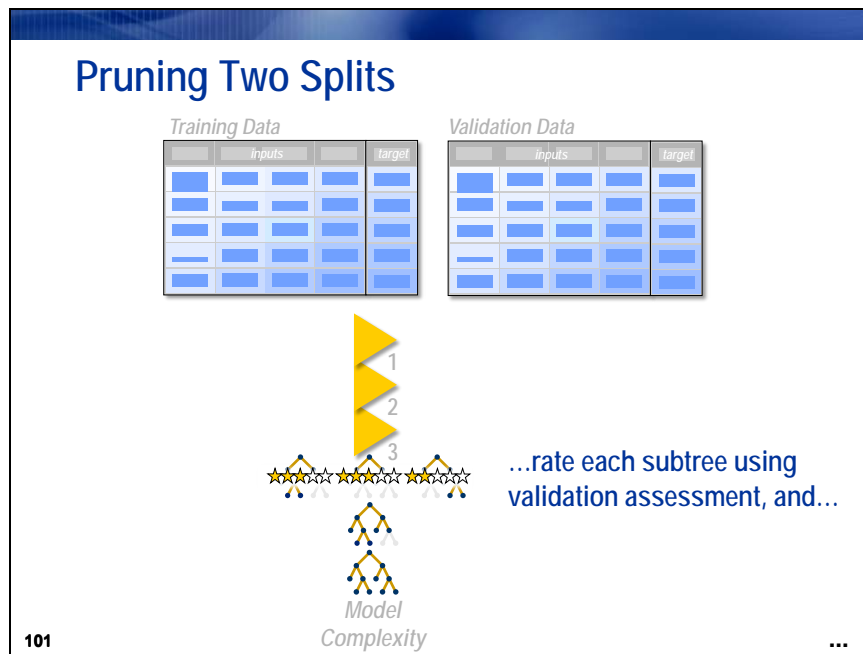
The subtree with the best validation assessment statistics is selected to represent a particular level of model complexity.



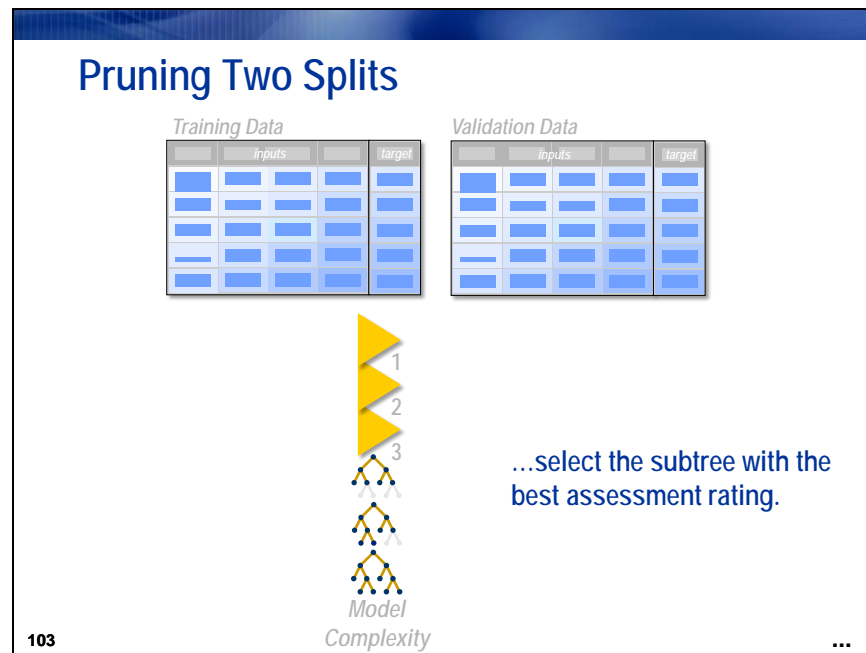
A similar process is followed to obtain the next batch of subtrees.



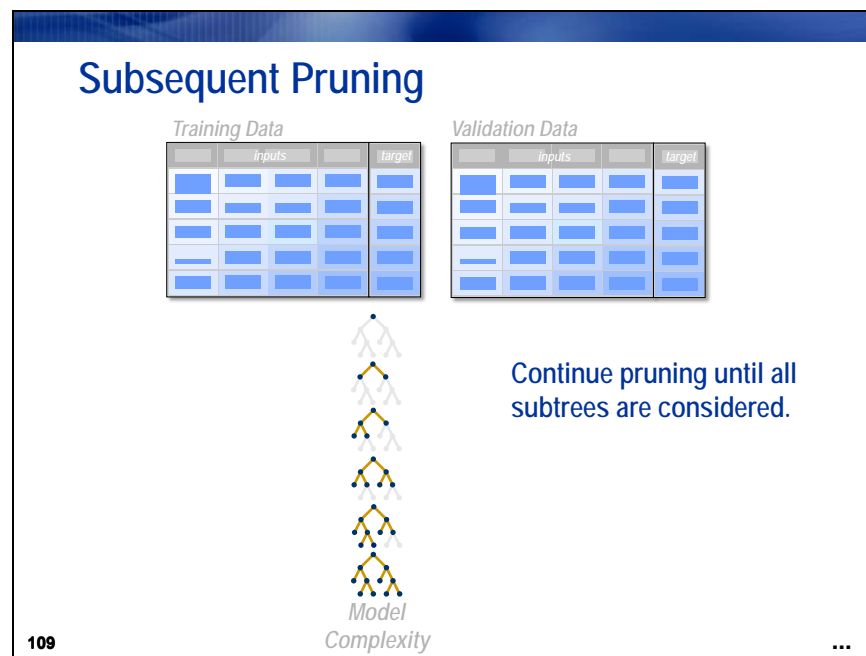
This time the subtrees are formed by removing two splits from the maximal tree.



Once again, the predictive rating of each subtree is compared using the validation data.



The subtree with the highest predictive rating is chosen to represent the given level of model complexity.



The process is repeated until all levels of complexity are represented by subtrees.

Selecting the Best Tree

Training Data

	inputs		target

Validation Data

	inputs		target

Model
Complexity

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

Validation
Assessment

Compare validation
assessment between
tree complexities.

110 ...

Consider the validation assessment rating for each of the subtrees.

Selecting the Best Tree

Training Data

	inputs		target

Validation Data

	inputs		target

Model
Complexity

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

Validation
Assessment

Which subtree should
be selected as the best
model?

111 ...

Which of the sequence of subtrees is the best model?


Validation Assessment

Training Data

	inputs			target

Validation Data

	inputs			target



Model Complexity

Validation Assessment

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

Choose the simplest model with the highest validation assessment.

112 ...

In SAS Enterprise Miner, the simplest model with the highest validation assessment is considered best.


Validation Assessment

Training Data

	inputs			target

Validation Data

	inputs			target



Model Complexity

Validation Assessment

★★★★★

What are appropriate validation assessment ratings?

113 ...

Of course, you must finally define what is meant by *validation assessment ratings*.

Assessment Statistics

Validation Data

	inputs			target

Ratings depend on...

☆☆☆☆ target measurement

☆☆☆☆ prediction type.

115

...

Two factors determine the appropriate validation assessment rating, or more properly, *assessment statistic*:

- the target measurement scale
- the prediction type

An appropriate statistic for a binary target might not make sense for an interval target. Similarly, models tuned for decision predictions might make poor estimate predictions.

Binary Targets

	inputs			target
				1
				0
				0
				1
				1

primary outcome

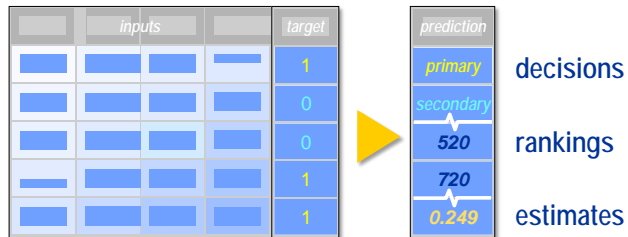
secondary outcome

117

...

For this discussion, a binary target is assumed with a primary outcome (**target=1**) and a secondary outcome (**target=0**). The appropriate assessment measure for interval targets is noted below.

Binary Target Predictions



119

...

There are different summary statistics corresponding to each of the three prediction types:

- decisions
- rankings
- estimates

The statistic that you should use to judge a model depends on the type of prediction that you want.

Decision Optimization

inputs				target	prediction
				1	primary
				0	secondary
				0	820
				1	720
				1	0.249

decisions

122

...

Consider decision predictions first. With a binary target, you will typically consider two decision types:

- the primary decision, corresponding to the primary outcome
- the secondary decision, corresponding to the secondary outcome

Decision Optimization – Accuracy

inputs				target	prediction
				1	primary
				0	secondary
				0	820
				1	720
				1	0.249

true positive

true negative

Maximize *accuracy*:
agreement between
outcome and prediction

123

...

Matching the primary decision with the primary outcome yields a correct decision called a *true positive*. Likewise, matching the secondary decision to the secondary outcome yields a correct decision called a *true negative*. Decision predictions can be rated by their *accuracy*, that is, the proportion of agreement between prediction and outcome.

Decision Optimization – Misclassification

	inputs		target	prediction
			1	secondary
			0	primary
			0	520
			1	720
			1	0.249

false negative
 false positive
 Minimize *misclassification*:
 disagreement between
 outcome and prediction

125

...

Mismatching the secondary decision with the primary outcome yields an incorrect decision called a *false negative*. Likewise, mismatching the primary decision to the secondary outcome yields an incorrect decision called a *false positive*. Decision predictions can be rated by their *misclassification*, the proportion of disagreement between prediction and outcome.

Ranking Optimization

	inputs		target	prediction
			1	secondary
			0	primary
			0	520
			1	720
			1	0.249

decisions
 rankings
 estimates

127

...

Consider ranking predictions for binary targets. With ranking predictions, a score is assigned to each case. The basic idea is to rank the cases based on their likelihood of being a primary or secondary outcome. Likely primary outcomes receive high scores and likely secondary outcomes receive low scores.

Ranking Optimization – Concordance

	inputs		target	prediction
			1	secondary
			0	primary
			0	520
			1	720
			1	0.249

target=0 → low score
target=1 → high score

Maximize *concordance*:
proper ordering of
primary and secondary
outcomes

129

...

When a pair of primary and secondary cases is correctly ordered, the pair is said to be in *concordance*. Ranking predictions can be rated by their degree of concordance, that is, the proportion of such pairs whose scores are correctly ordered.

Ranking Optimization – Discordance

	inputs		target	prediction
			1	secondary
			0	primary
			0	720
			1	520
			1	0.249

target=0 → high score
target=1 → low score

Minimize *discordance*:
improper ordering of
primary and secondary
outcomes

132

...

When a pair of primary and secondary cases is incorrectly ordered, the pair is said to be in *discordance*. Ranking predictions can be rated by their degree of discordance, that is, the proportion of such pairs whose scores are incorrectly ordered.

Estimate Optimization

	inputs			target	prediction	
				1	secondary	decisions
				0	primary	
				0	720	rankings
				1	520	
				1	0.249	estimates

133

...

Finally, consider estimate predictions. For a binary target, *estimate predictions* are the probability of the primary outcome for each case. Primary outcome cases should have a high predicted probability; secondary outcome cases should have a low predicted probability.

Estimate Optimization – Squared Error

	inputs			target	prediction
				1	secondary
				0	primary
				0	720
				1	520
				1	0.249

$(\text{target} - \text{estimate})^2$

Minimize *squared error*:
squared difference between
target and prediction

135

...

The squared difference between target and estimate is called the *squared error*.

Averaged over all cases, squared error is a **fundamental** statistical measure of model performance.

$$\text{Average square error} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

where, for the i^{th} case, y_i is the actual target value and \hat{y}_i is the predicted target value.

When calculated in an unbiased fashion, the average square error is related to the amount of bias in a predictive model. A model with a lower average square error is less biased than a model with a higher average square error.



Because SAS Enterprise Miner enables the target variable to have more than two outcomes, the actual formula used to calculate average square error is slightly different (but equivalent, for a binary target). Suppose the target variable has L outcomes given by (C_1, C_2, \dots, C_L) , then

$$\text{Average square error} = \frac{1}{N \cdot L} \sum_{i=1}^N \sum_{j=1}^L (I(y_i = C_j) - \hat{p}_{ij})^2$$

where, for the i^{th} case, y_i is the actual target value, \hat{p}_{ij} is the predicted target value for the j^{th} class, and the following:

$$I(y_i = C_j) = \begin{cases} 1 & y_i = C_j \\ 0 & y_i \neq C_j \end{cases}$$

Complexity Optimization – Summary

	inputs			target	prediction
				1	secondary
				0	primary
				0	720
				1	520
				1	0.249

decisions

accuracy / misclassification

rankings

concordance / discordance

estimates

squared error

In summary, decisions require high accuracy or low misclassification, rankings require high concordance or low discordance, and estimates require low (average) squared error.



Assessing a Decision Tree

Use the following steps to perform an assessment of the maximal tree on the validation sample.

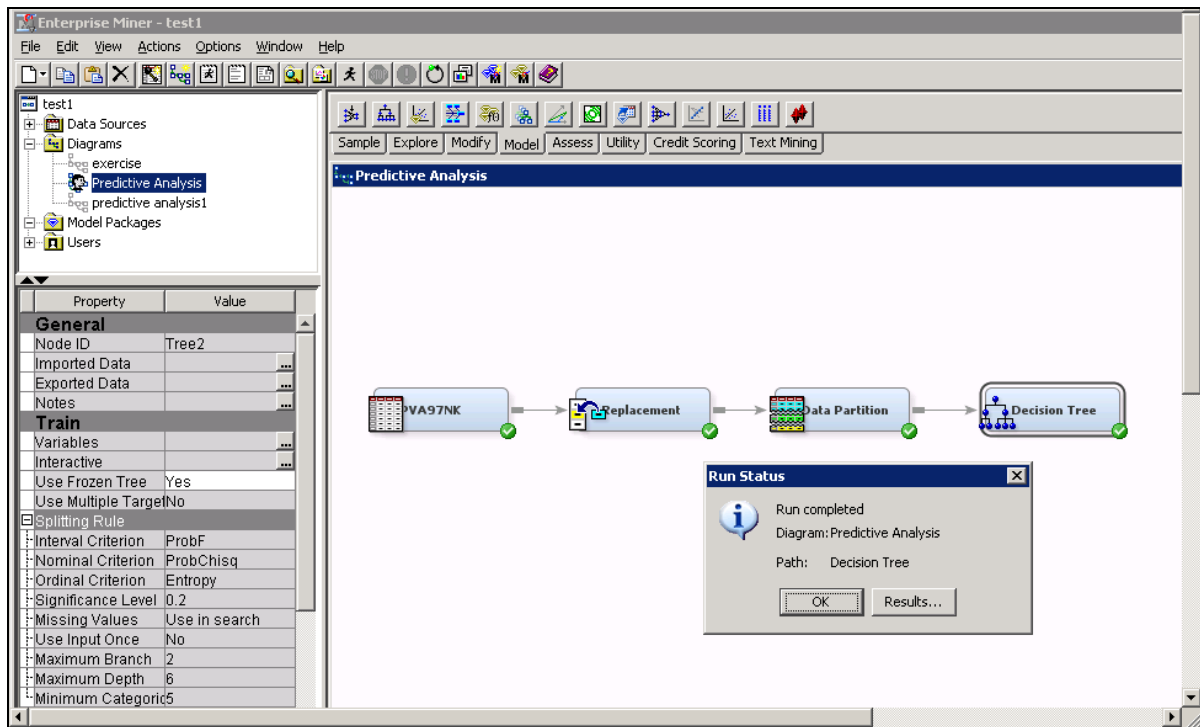
1. Select the **Decision Tree** node. In the Decision Tree node's properties, change **Use Frozen Tree** from No to **Yes**.

The screenshot shows the Enterprise Miner software interface. On the left, the 'My Project' tree lists 'Data Sources', 'Diagrams', 'Model Packages', and 'Users'. The 'Predictive Analysis' diagram is selected. The main workspace displays a workflow: 'PVA97NK' (Data Source) → 'Replacement' (Model) → 'Data Partition' (Model) → 'Decision Tree' (Model). The 'Decision Tree' node is selected, and its properties are shown in the bottom-left pane. A red rectangle highlights the 'Use Frozen Tree' property, which is set to 'Yes'.

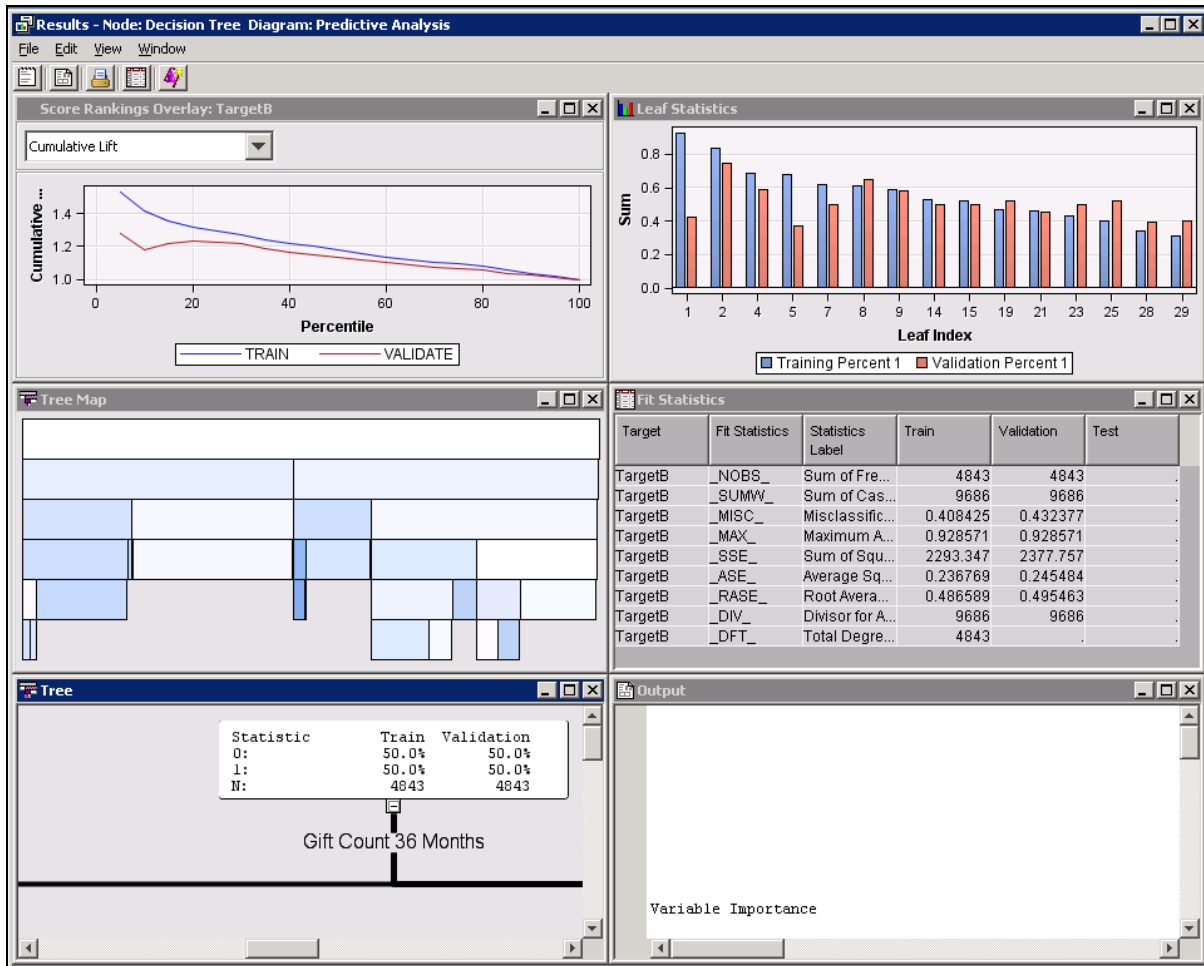
Property	Value
General	
Node ID	Tree5
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Interactive	
Use Frozen Tree	Yes
Use Multiple Targets	No
Splitting Rule	
Interval Criterion	ProbF
Nominal Criterion	ProbChisq
Ordinal Criterion	Entropy
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	2
Maximum Depth	6
Minimum Categorical S	15
Use Frozen Tree	

The Frozen Tree property prevents the maximal tree from being changed by other property settings when the flow is run.

2. Right-click the **Decision Tree** node and run it. Select **Results**.

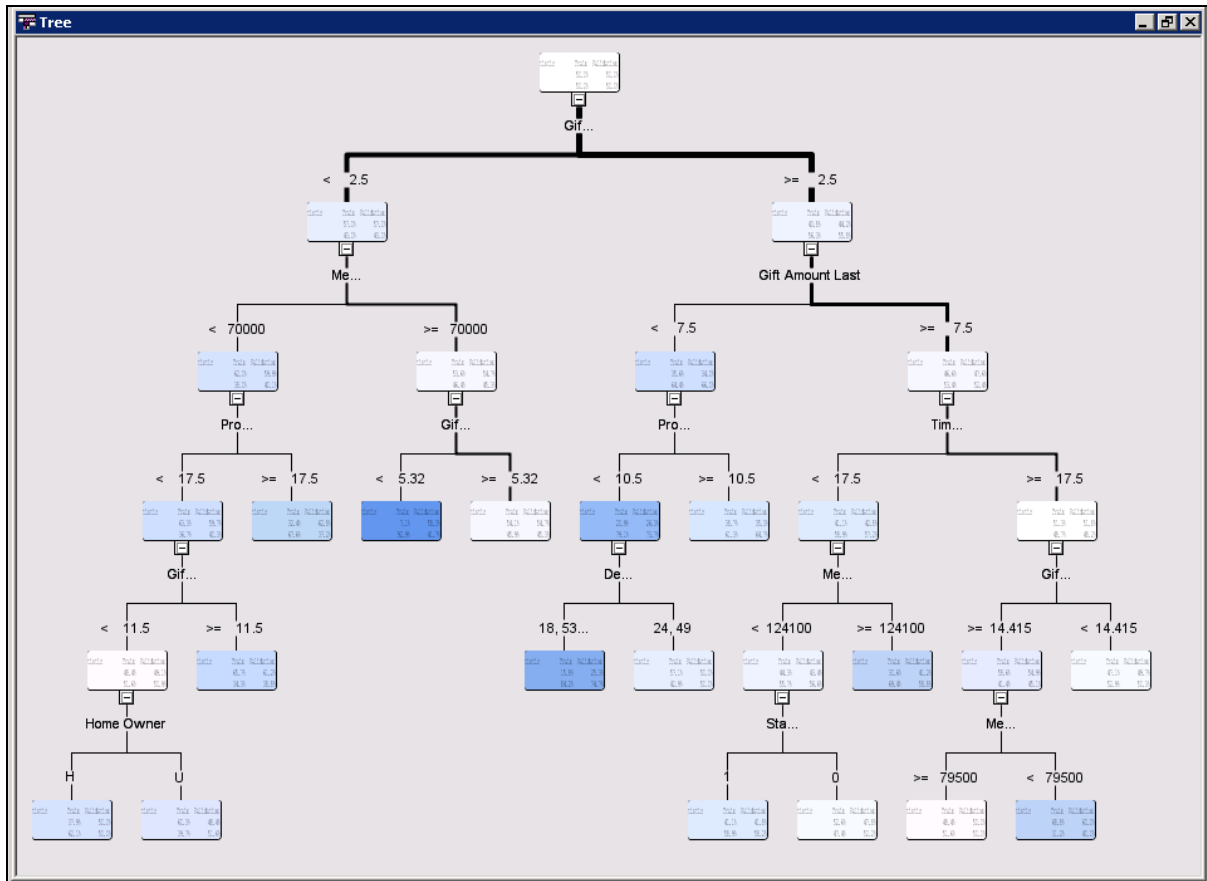


The Results window opens.



The Results window contains a variety of diagnostic plots and tables, including a cumulative lift chart, a tree map, and a table of fit statistics. The diagnostic tools shown in the results vary with the measurement level of the target variable. Some of the diagnostic tools contained in the results shown above are described in the following Self-Study section.

The saved tree diagram is in the bottom left corner of the Results window. You can verify this by maximizing the tree plot.



4. The main focus of this part of the demonstration is on assessing the performance of a 15-leaf tree, created with the Interactive Decision Tree tool, on the validation sample. Notice that several of the diagnostics shown in the results of the Decision Tree node use the validation data as a basis.

Select **View** ⇒ **Model** ⇒ **Iteration Plot**.



The plot shows the Average Square Error corresponding to each subtree as the data is sequentially split.

This plot is similar to the one generated with the Interactive Decision Tree tool, and it confirms suspicions about the optimality of the 15-leaf tree. The performance on the training sample becomes monotonically better as the tree becomes more complex. However, the performance on the validation sample only improves up to a tree of, approximately, four or five leaves, and then diminishes as model complexity increases.



The validation performance shows evidence of model overfitting. Over the range of one to approximately four leaves, the precision of the model improves with the increase in complexity. A marginal increase in complexity over this range results in better accommodation of the systematic variation or signal in data. Precision diminishes as complexity increases past this range; the additional complexity accommodates idiosyncrasies in the training sample, and the model extrapolates less well.

Both axes of the Iteration Plot, shown above, require some explanation.

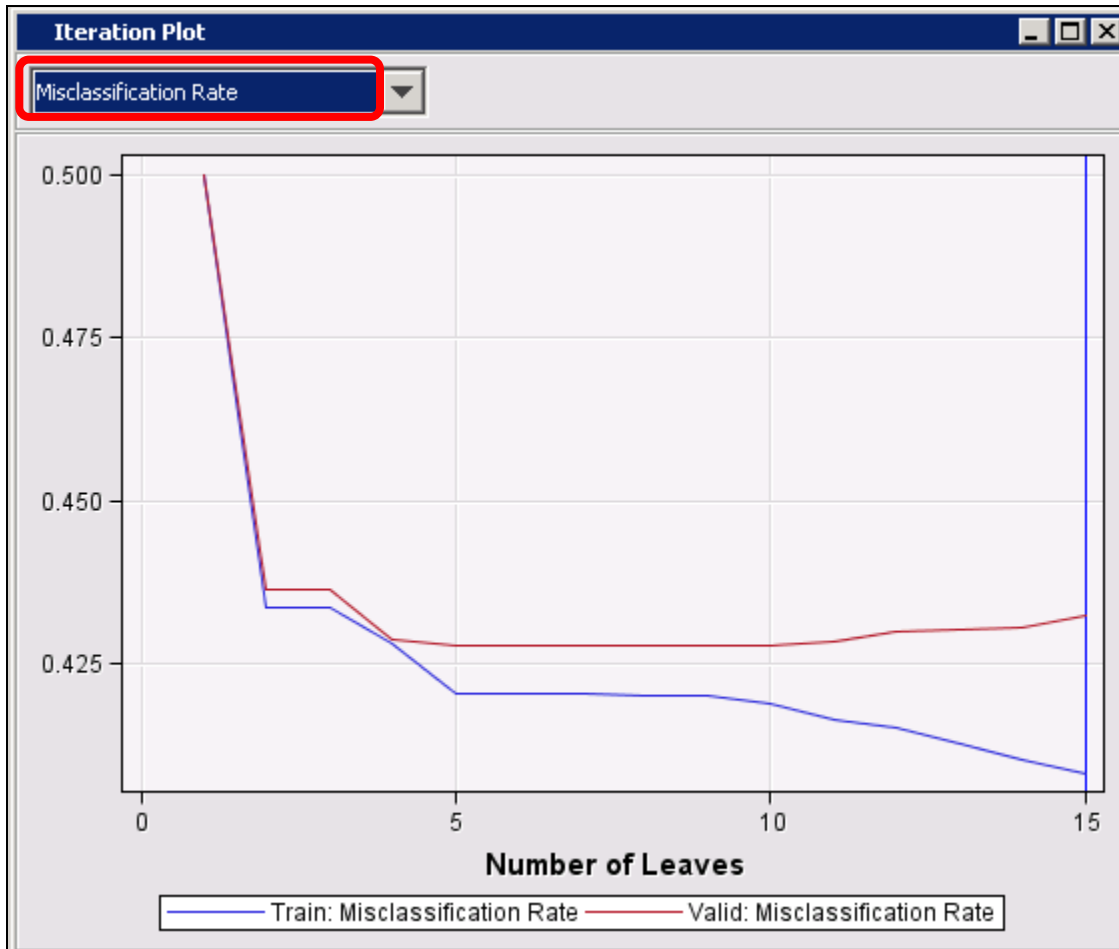
Average Square Error:

The formula for average square error was given previously. For any decision tree in the sequence, necessary inputs for calculating this statistic are the actual target value and the prediction. Recall that predictions generated by trees are the proportion of cases with the primary outcome (**TargetB=1**) in each terminal leaf. This value is calculated for both the training and validation data sets.

Number of leaves:

Starting with the four-leaf tree that you constructed earlier, it is possible to create a sequence of simpler trees by removing partitions. For example, a three-leaf tree can be constructed from your four-leaf tree by removing either the left partition (involving **Median Home Value**) or the right partition (involving **Gift Amount Last**). Removing either split might affect the generated average square error. The subtree with the lowest average square error (measured, by default, on the validation data) is the split that remains. To create simpler trees, the process is repeated by, once more, starting with the four-leaf tree that you constructed and removing more leaves. For example, the two-leaf tree is created from the four-leaf tree by removing two splits, and so on.

- To further explore validation performance, select the arrow in the upper left corner of the Iteration Plot, and switch the assessment statistic to **Misclassification Rate**.



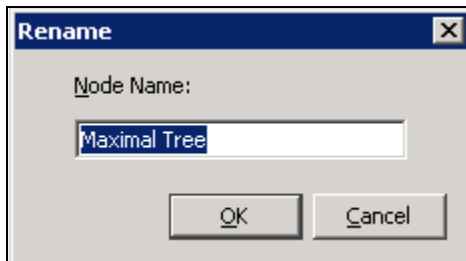
The validation performance under Misclassification Rate is similar to the performance under Average Square Error. The optimal tree appears to have, roughly, four or five leaves.

Proportion misclassified:

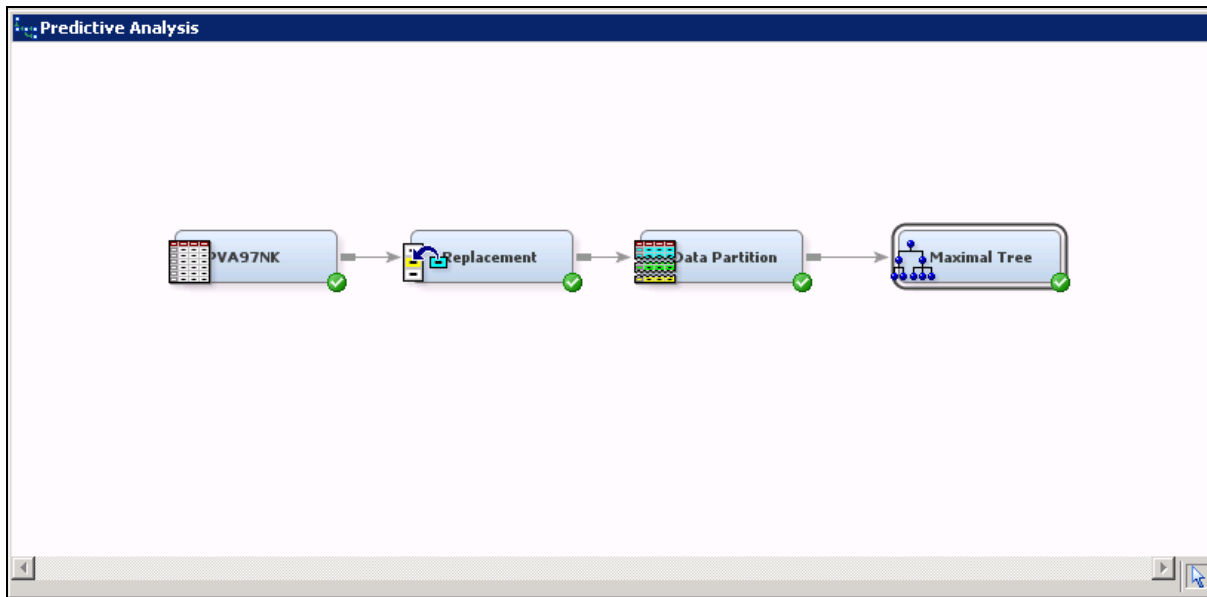
The name *decision tree model* comes from the decision that is made at the leaf or terminal node. In SAS Enterprise Miner, this decision, by default, is a classification based on the *predicted target value*. A predicted target value in excess of 0.5 results in a primary outcome classification, and a predicted target value less than 0.5 results in a secondary outcome classification. Within a leaf, the predicted target value decides the classification for all cases regardless of each case's actual outcome. Clearly, this classification will be inaccurate for all cases that are not in the assigned class. The fraction of cases for which the wrong decision (classification) was made is the proportion misclassified. This value is calculated for both the training and validation data sets.

- The current Decision Tree node will be renamed and saved for reference. Close the Decision Tree Results window.

7. Right-click on the **Decision Tree** node and select **Rename**. Name the node **Maximal Tree**.



The current process flow should look similar to the following:



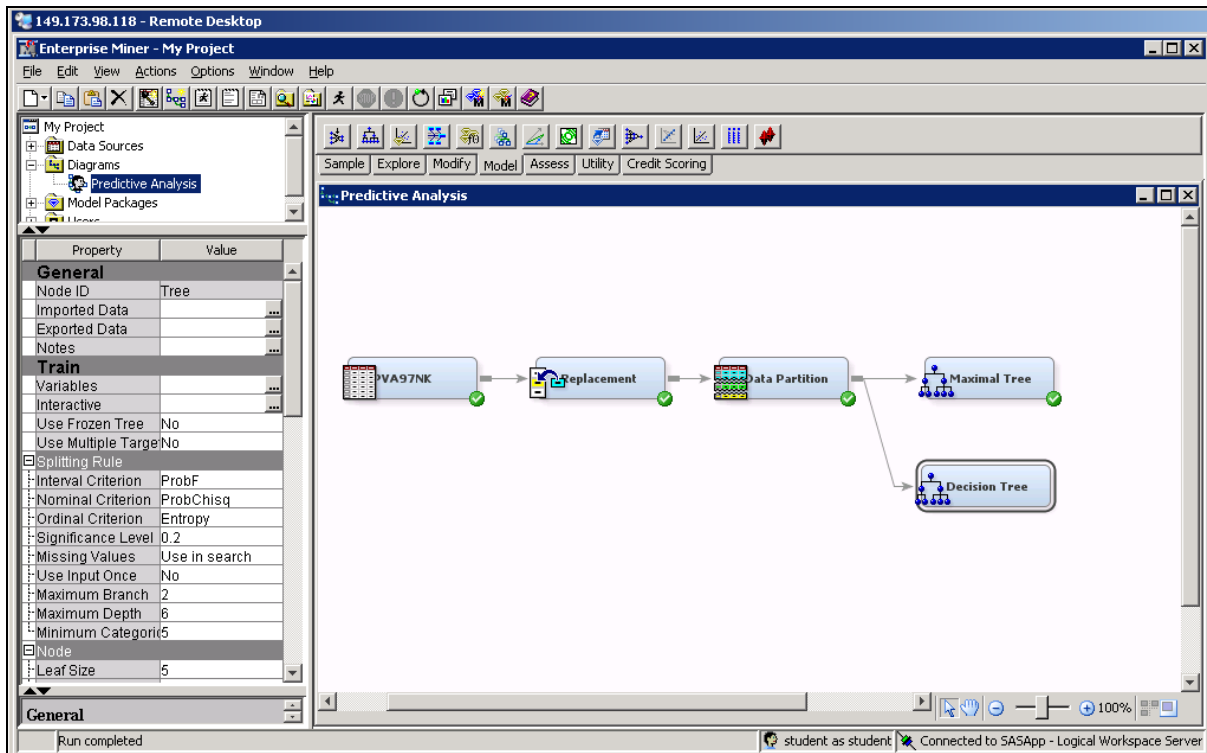
The next step is to generate the optimal tree using the default Decision Tree properties in SAS Enterprise Miner.

Pruning a Decision Tree

The optimal tree in the sequence can be identified using tools contained in the Decision Tree node. The relevant properties are listed under the Subtree heading.

1. Drag another **Decision Tree** node from the Model tab into the diagram.


Connect it to the **Data Partition** node so that your diagram looks similar to what is shown below.




2. Select the new **Decision Tree** node and scroll down in the Decision Tree properties to the **Subtree** section. The main tree pruning properties are listed under the Subtree heading.

Property	Value
Maximum Depth	6
Minimum Categorical Split	5
Node	
Leaf Size	5
Number of Rules	5
Number of Surrogate Rules	0
Split Size	
Split Search	
Exhaustive	5000
Node Sample	20000
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Decision
Assessment Fraction	0.25
Cross Validation	
Perform Cross Validation	No
Number of Subsets	10
Number of Repeats	1
Seed	12345
Observation Based Importance	
Observation Based Importance	No
Number Single Variable Importance	5
P-Value Adjustment	
Bonferroni Adjustment	Yes
Time of Kass Adjustment	Before
Inputs	No
Number of Inputs	1
Split Adjustment	Yes

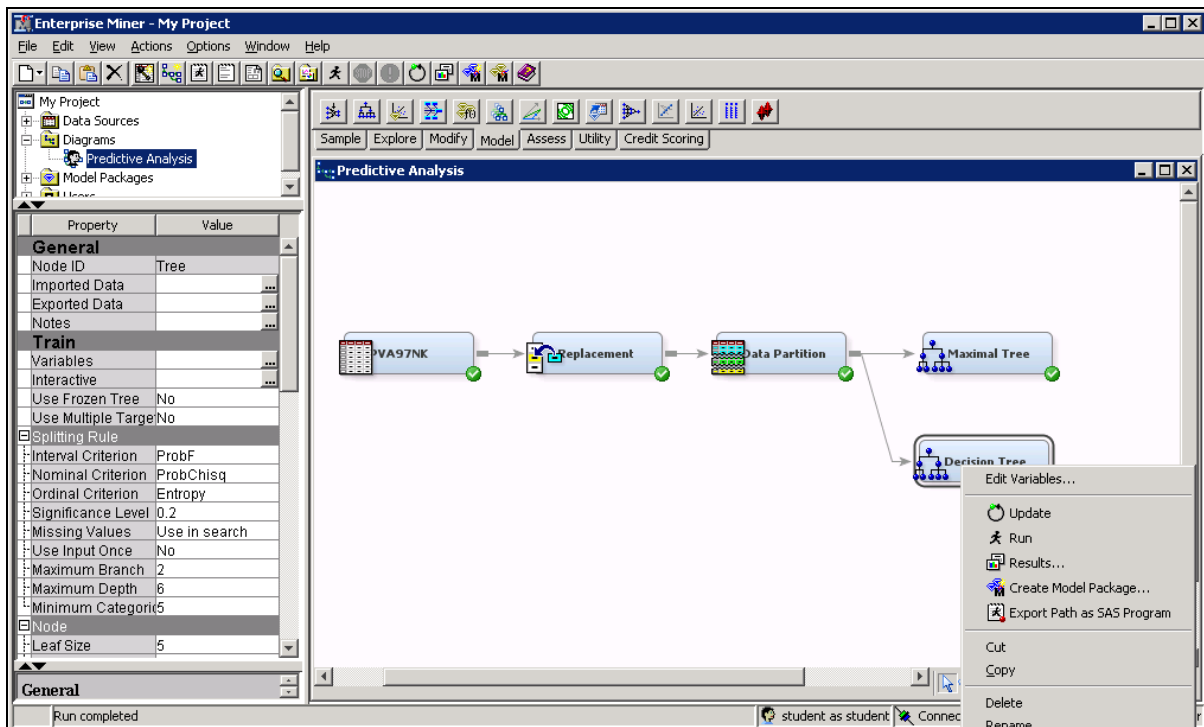
The default method used to prune the maximal tree is Assessment. This means that algorithms in SAS Enterprise Miner choose the best tree in the sequence based on some optimality measure.

 Alternative method options are Largest and N. The Largest option provides an autonomous way to generate the maximal tree. The N option generates a tree with N leaves. The maximal tree is upper-bound on N .

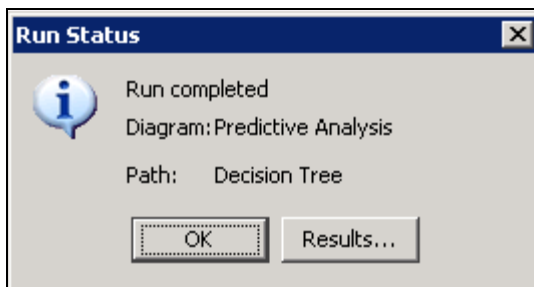
The Assessment Measure property specifies the optimality measure used to select the best tree in the sequence. The default measure is Decision. The default measure varies with the measurement level of the target variable, and other settings in SAS Enterprise Miner.

 Metadata plays an important role in how SAS Enterprise Miner functions. Recall that a binary target variable was selected for the project. Based on this, SAS Enterprise Miner assumes that you want a tree that is optimized for making the best **decisions** (as opposed to the best rankings or best probability estimates). That is, under the current project settings, SAS Enterprise Miner chooses the tree with the lowest misclassification rate on the validation sample by default.

3. Right-click on the new **Decision Tree** node and select **Run**.



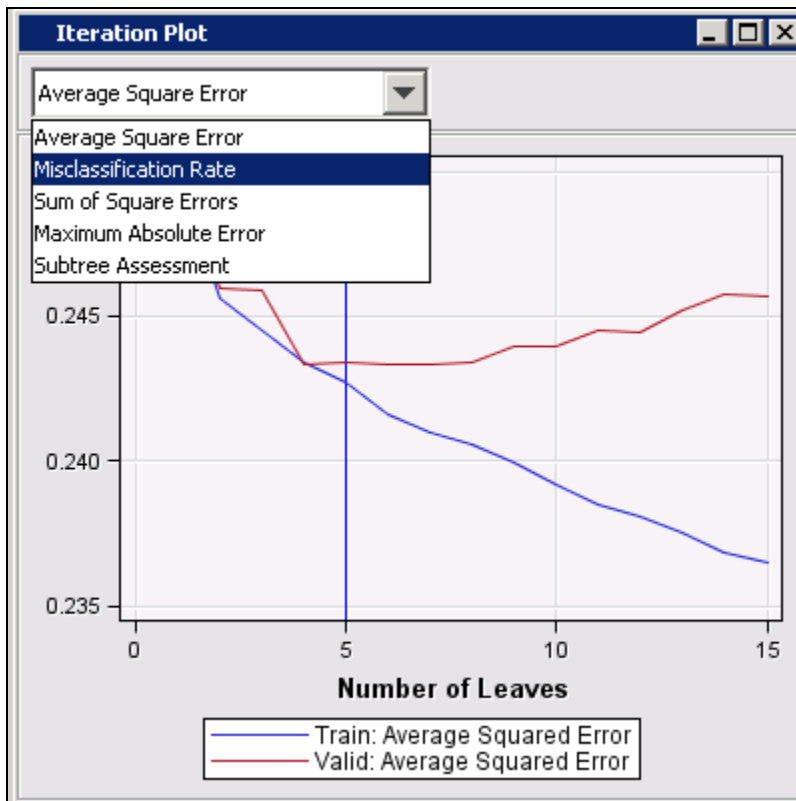
4. After the Decision Tree node runs, select **Results...**



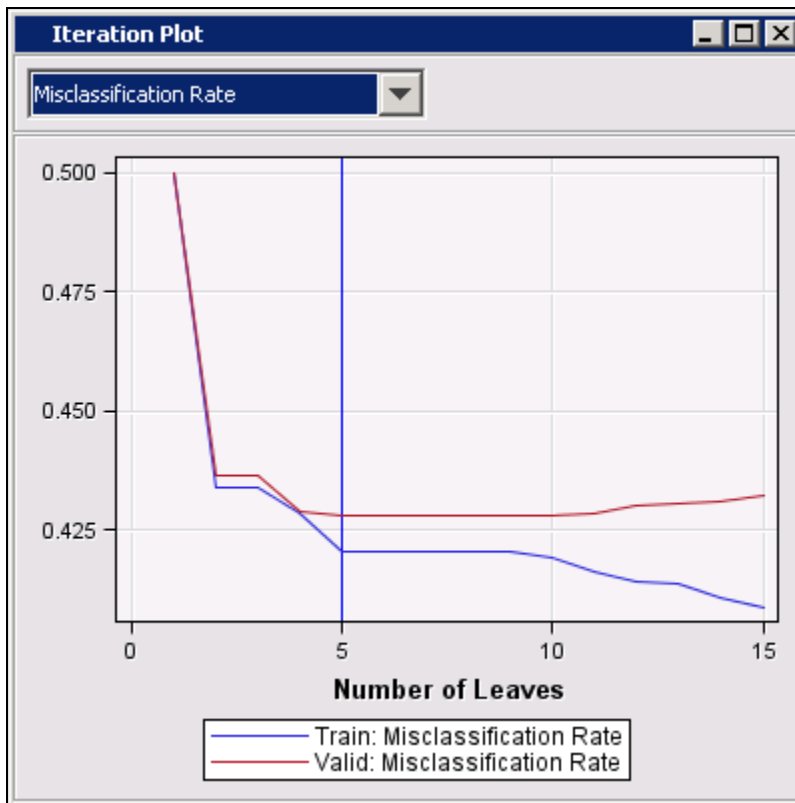
5. In the Result window, select **View** ⇒ **Model** ⇒ **Iteration Plot**.

The Iteration Plot shows model performance under Average Square Error by default. However, this is not the criterion used to select the optimal tree.

6. Change the basis of the plot to **Misclassification Rate**.

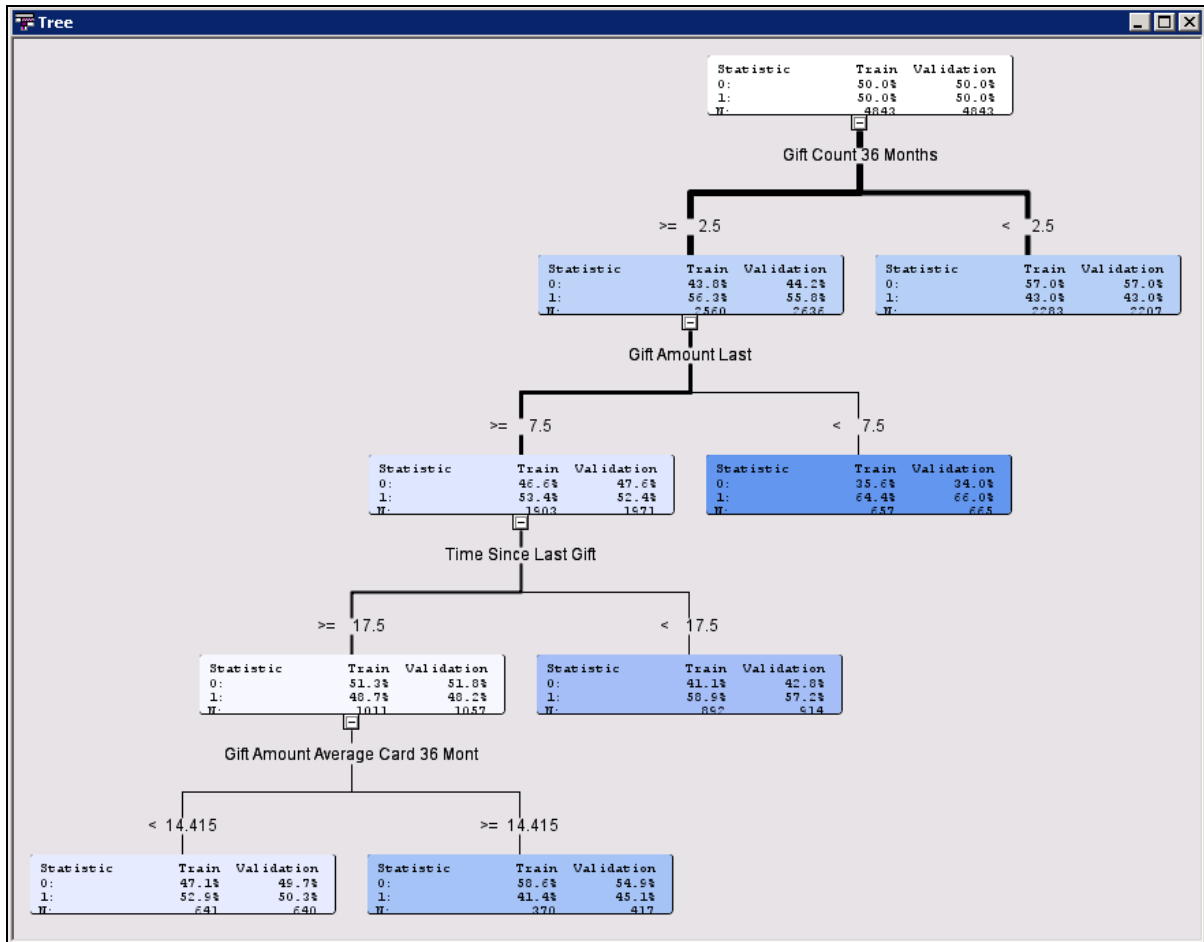


The five-leaf tree has the lowest associated misclassification rate on the validation sample.



Notice that the first part of the process in generating the optimal tree is very similar to the process followed in the Interactive Decision Tree tool. The maximal, 15-leaf tree is generated. The maximal tree is then sequentially pruned so that the sequence consists of the best 15-leaf tree, the best 14-leaf tree, and so on. *Best* is defined at any point, for example, eight, in the sequence as the eight-leaf tree with the lowest Validation Misclassification Rate among all candidate eight-leaf trees. The tree in the sequence with the lowest overall validation misclassification rate is selected.

The optimal tree generated by the Decision Tree node is found in the bottom left corner of the Results window.



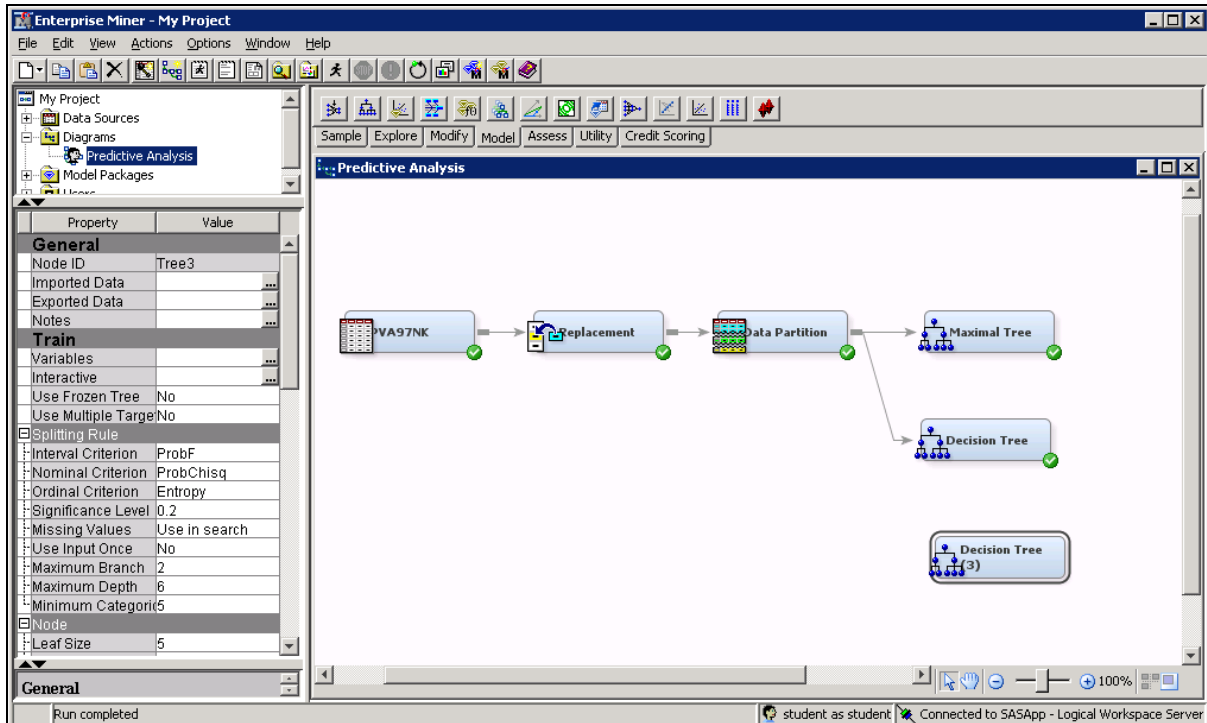
7. Close the Results window of the Decision Tree node.

Alternative Assessment Measures

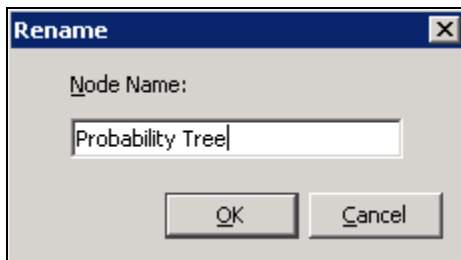
The Decision Tree generated above is optimized for decisions, that is, to make the best classification of cases into donors and non-donors. What if instead of classification, the primary objective of the project is to assess the **probability of donation**? The discussion above describes how the different assessment measures correspond to the optimization of different modeling objectives.

Generate a tree that is optimized for generating probability estimates. The main purpose is to explore how different modeling objectives can change the optimal model specification.

1. Navigate to the **Model** tab. Drag a new **Decision Tree** into the process flow.

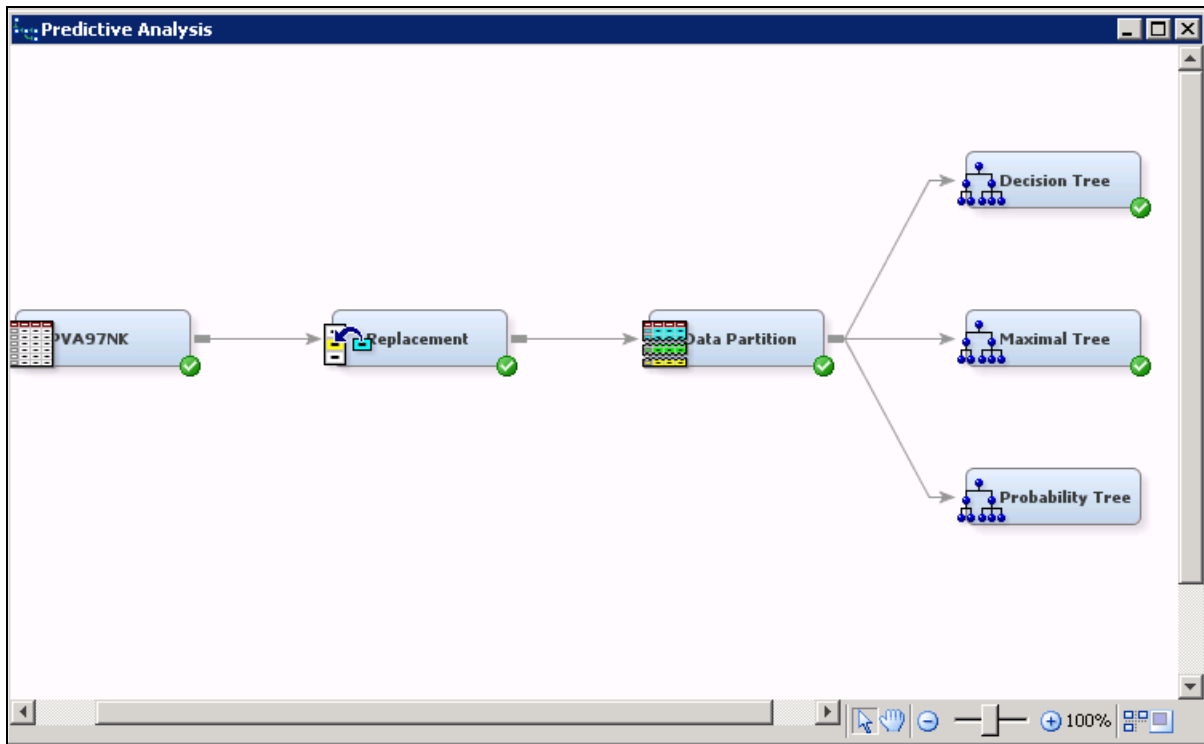


2. Right-click the new **Decision Tree** node and select **Rename**.
3. Name the new Decision Tree node **Probability Tree**.



4. Connect the **Probability Tree** node to the **Data Partition** node.

Your diagram should look similar to the following:



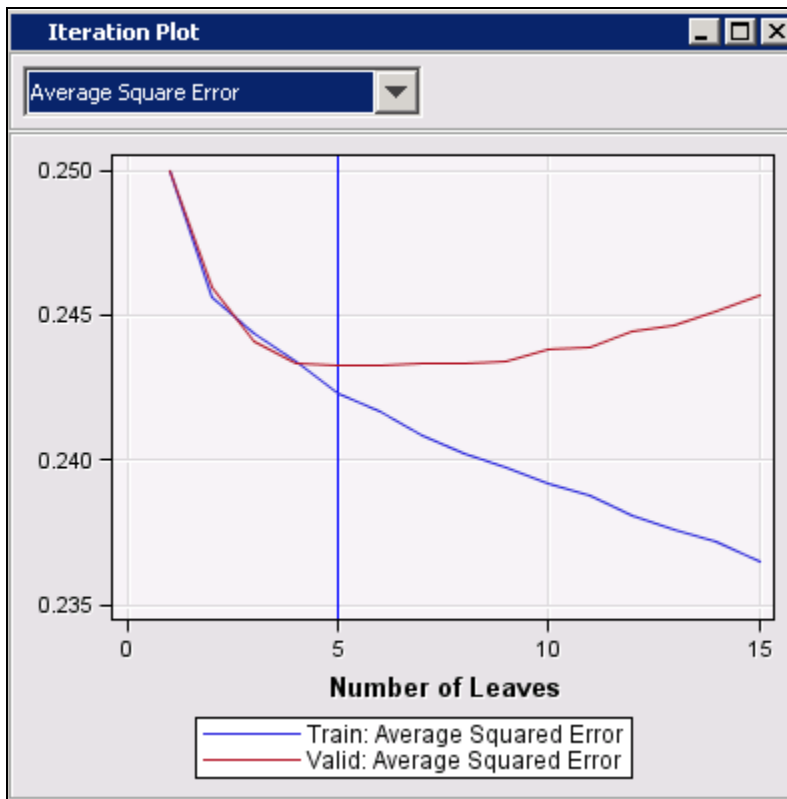
5. Recall that an appropriate assessment measure for generating optimal probability estimates is Average Square Error.

6. Scroll down in the properties of the Probability Tree node to Subtree.
7. Change the Assessment Measure property to **Average Square Error**.

Property	Value
Maximum Branch	2
Maximum Depth	6
Minimum Categorical Split	5
Node	
Leaf Size	5
Number of Rules	5
Number of Surrogate Rules	0
Split Size	
Split Search	
Exhaustive	5000
Node Sample	20000
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25
Cross Validation	
Perform Cross Validation	No
Number of Subsets	10
Number of Repeats	1
Seed	12345
Observation Based Impurity	
Observation Based Impurity	No
Number Single Var Impurity	5
P-Value Adjustment	
Bonferroni Adjustment	Yes
Time of Kass Adjustment	Before
Inputs	No
Number of Inputs	1
Split Adjustment	Yes
Output Variables	
Leaf Variable	Yes
Performance	Disk

8. Run the Probability Tree node.
9. After it runs, select **Results...**.

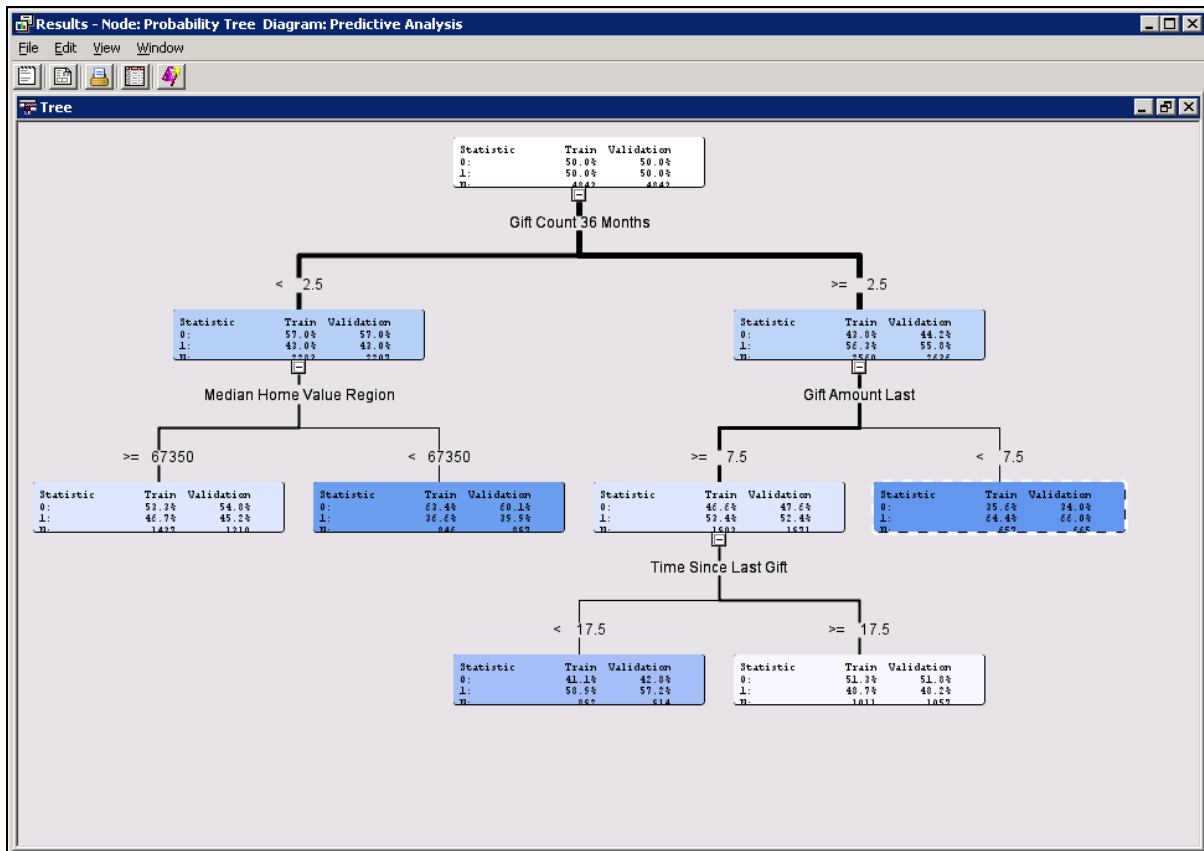
10. In the Results window, select **View** ⇒ **Model** ⇒ **Iteration Plot**.



A five-leaf tree is also optimal under the Validation Average Square Error criterion. However, viewing the Tree plot reveals that, although the optimal Decision (Misclassification) Tree and the optimal Probability Tree have the same number of leaves, they are not identical trees.

11. Maximize the Tree plot at the bottom left corner of the Results window.

The tree shown below is a different five-leaf tree than the tree optimized under Validation Misclassification.



It is important to remember how the assessment measures work in model selection. In the first step, cultivating (or growing or splitting) the tree, the important measure is *logworth*. Splits of the data are made based on logworth to isolate subregions of the input space with high proportions of donors and non-donors. Splitting continues until a boundary associated with a stopping rule is reached. This process generates the maximal tree.

The maximal tree is then sequentially pruned. Each subtree in the pruning sequence is chosen based on the selected assessment measure. For example, the eight-leaf tree in the Decision (Misclassification) Tree pruning sequence is the eight-leaf tree with the lowest misclassification rate on the validation sample among all candidate eight-leaf trees. The eight-leaf tree in the Probability Tree pruning sequence is the eight-leaf tree with the lowest average square error on the validation sample among all candidate eight-leaf trees. This is how the change in assessment measure can generate a change in the optimal specification.



In order to minimize clutter in the diagram, the Maximal Tree node is deleted and does not appear in the notes for the remainder of the course.

3.4 Understanding Additional Diagnostic Tools (Self-Study)

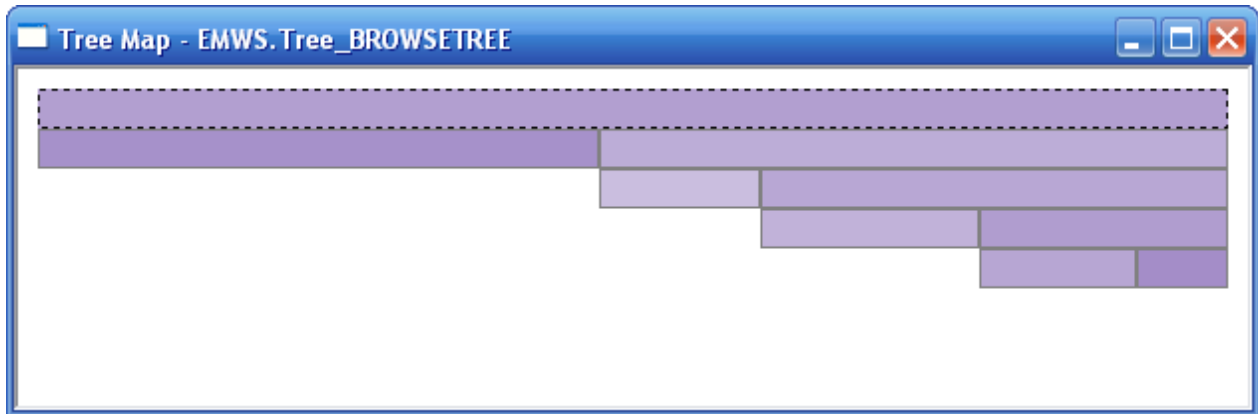
Several additional plots and tables contained in the Decision Tree node Results window are designed to assist in interpreting and evaluating a decision tree. The most useful are demonstrated on the next several pages.



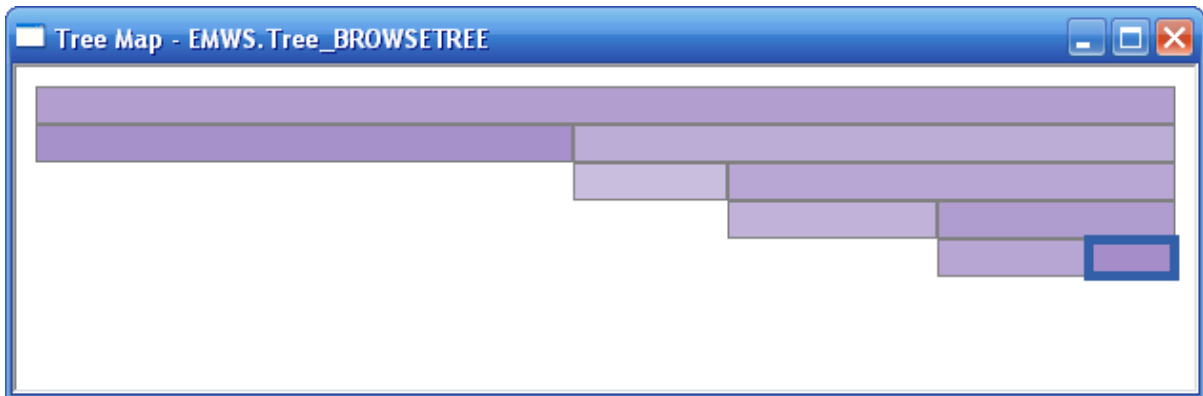
Understanding Additional Plots and Tables (Optional)

Tree Map

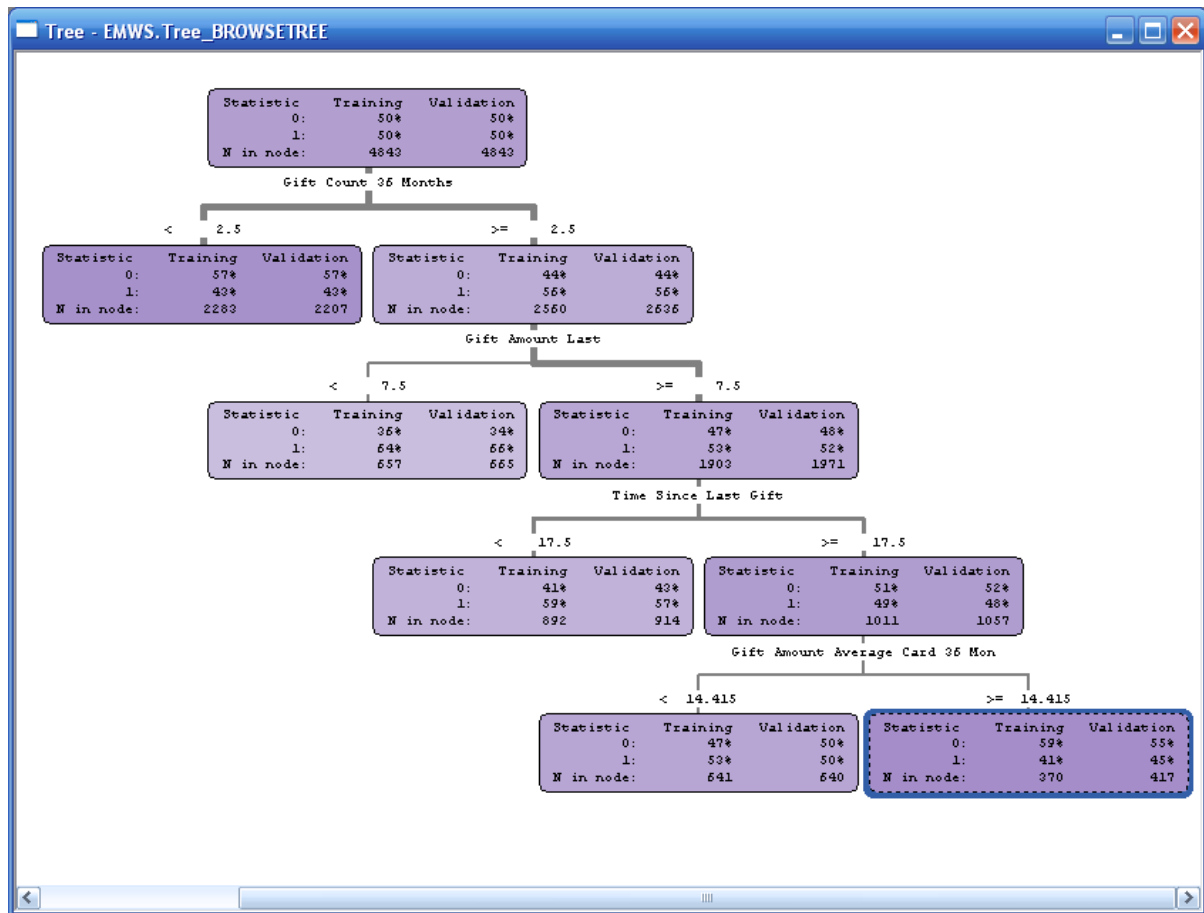
The Tree Map window is intended to be used in conjunction with the Tree window to gauge the relative size of each leaf.



1. Select the small rectangle in the lower right corner of the Tree map.

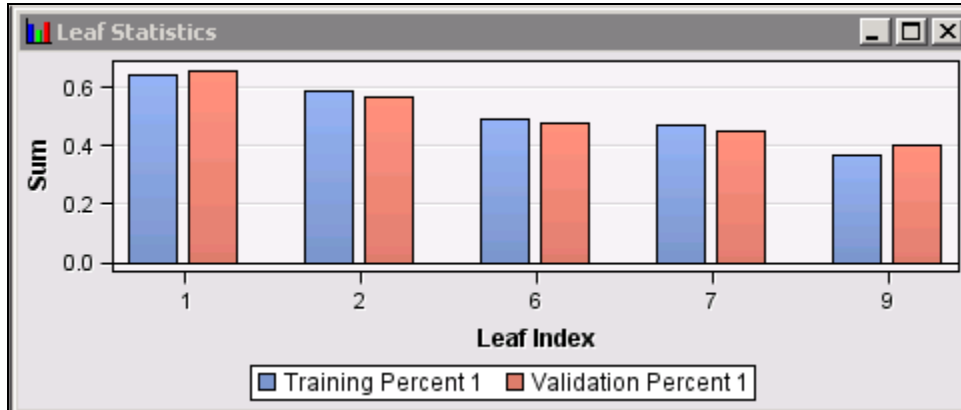


The lower right leaf in the tree is also selected.



The size of the small rectangle in the tree map indicates the fraction of the training data present in the corresponding leaf. It appears that only a small fraction of the training data finds its way to this leaf.

Leaf Statistic Bar Chart



This window compares the blue predicted outcome percentages in the left bars (from the training data) to the red observed outcome percentages in the right bars (from the validation data).

This plot reveals how well training data response rates are reflected in the validation data. Ideally, the bars should be of the same height. Differences in bar heights are usually the result of small case counts in the corresponding leaf.

Variable Importance

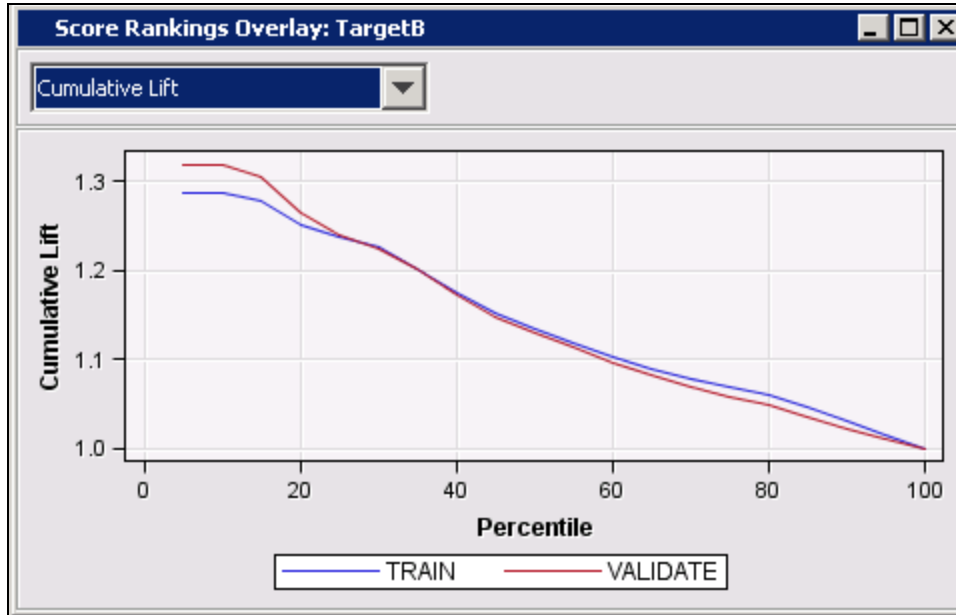
The following steps open the Variable Importance window.

Select **View** ⇒ **Model** ⇒ **Variable Importance**. The Variable Importance window opens.

Variable Importance					
Variable Name	Label	Number of Splitting Rules	Importance	Validation Importance	Ratio of Validation to Training Importance
GiftCnt36	Gift Count 3...	1	1	1	1
GiftAvgLast	Gift Amount...	1	0.524135	0.671883	1.281891
DemMedH...	Median Ho...	1	0.50277	0.10861	0.216024
GiftTimeLast	Time Since ...	1	0.480916	0.445303	0.925948
DemPctVet...	Percent Vet...	0	0	0	.
GiftAvg36	Gift Amount...	0	0	0	.
GiftAvgCard...	Gift Amount...	0	0	0	.
DemAge	Age	0	0	0	.
GiftAvgAll	Gift Amount...	0	0	0	.
GiftCntAll	Gift Count A...	0	0	0	.
GiftCntCard...	Gift Count ...	0	0	0	.
GiftCntCard...	Gift Count ...	0	0	0	.
GiftTimeFirst	Time Since ...	0	0	0	.
PromCntCa...	Promotion ...	0	0	0	.
PromCnt12	Promotion ...	0	0	0	.
PromCnt36	Promotion ...	0	0	0	.
PromCntAll	Promotion ...	0	0	0	.
PromCntCa...	Promotion ...	0	0	0	.
PromCntCa...	Promotion ...	0	0	0	.
StatusCatSt...	Status Cate...	0	0	0	.
REP_Dem...	Replaceme...	0	0	0	.
DemCluster	Demograp...	0	0	0	.
DemGender	Gender	0	0	0	.
DemHome...	Home Owner	0	0	0	.
StatusCat9...	Status Cate...	0	0	0	.

The Variable Importance window provides insight into the importance of inputs in the decision tree. The magnitude of the importance statistic relates to the amount of variability in the target **explained** by the corresponding input relative to the input at the top of the table. For example, **Gift Amount Last** explains 52.4% of the variability explained by **Gift Count 36 Months**.

The Score Rankings Overlay Plot



The Score Rankings Overlay plot presents what is commonly called a *cumulative lift chart*. Cases in the training and validation data are ranked, based on decreasing predicted target values. A fraction of the ranked data is selected (given by the decile value). The proportion of cases with the primary target value in this fraction is compared to the proportion of cases with the primary target value overall (given by the cumulative lift value). A useful model shows high lift in low deciles in both the training and validation data.

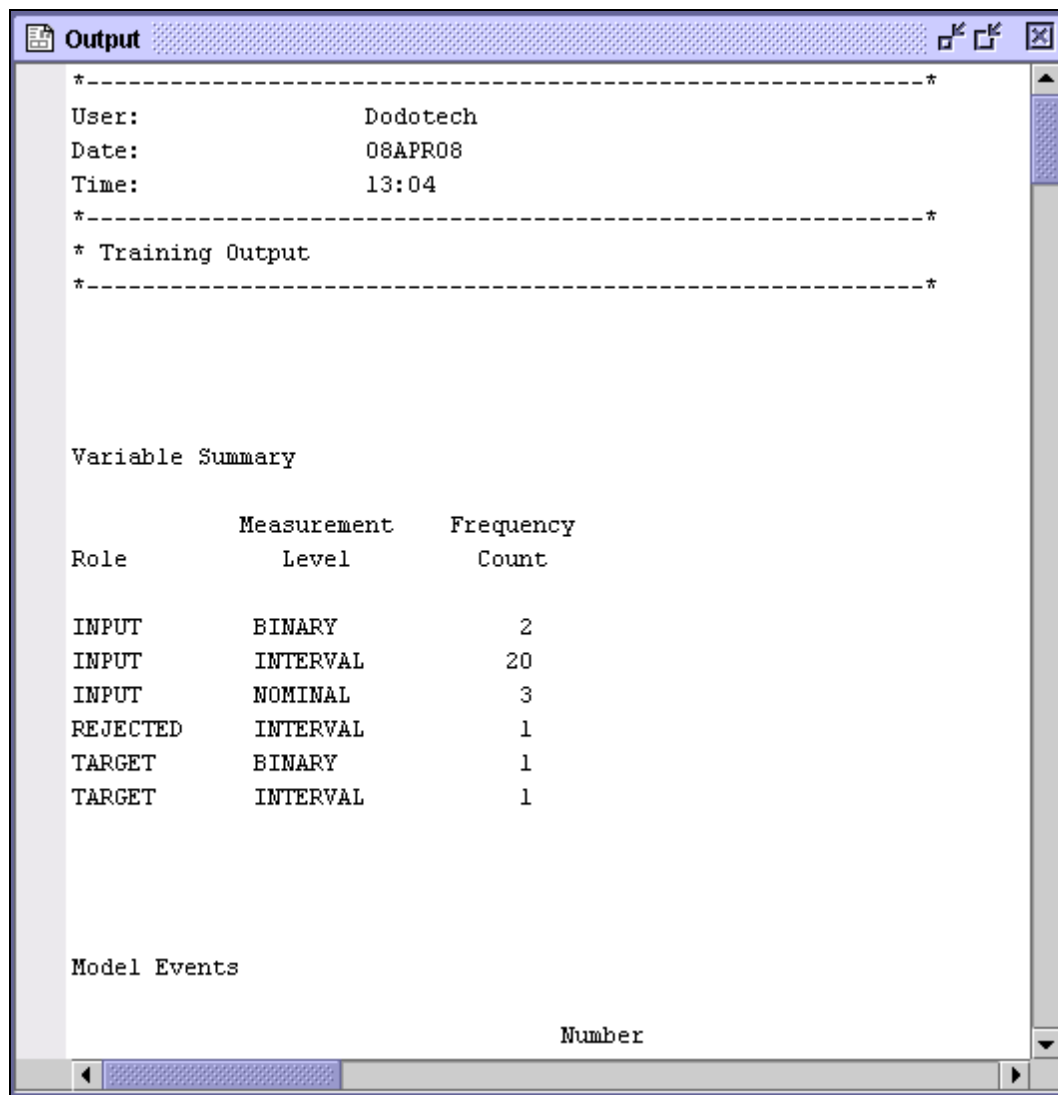
For example, the Score Ranking plot shows that in the top 20% of cases (ranked by predicted probability), the training and validation lifts are approximately 1.26. This means that the proportion of primary outcome cases in this top 20% is about 26% more likely to have the primary outcome than a randomly selected 20% of cases.

Fit Statistics

Fit Statistics						
Target	Fit Statistics	Statistics Label	Train	Validation	Test	
TargetB	_NOBS_	Sum of Fre...	4843	4843	.	
TargetB	_SUMW_	Sum of Cas...	9686	9686	.	
TargetB	_MISC_	Misclassific...	0.420607	0.42804	.	
TargetB	_MAX_	Maximum A...	0.643836	0.643836	.	
TargetB	_SSE_	Sum of Squ...	2351.152	2357.331	.	
TargetB	_ASE_	Average Sq...	0.242737	0.243375	.	
TargetB	_RASE_	Root Avera...	0.492684	0.493331	.	
TargetB	_DIV_	Divisor for A...	9686	9686	.	
TargetB	_DFT_	Total Degre...	4843	.	.	

The Fit Statistics window is used to compare various models built within SAS Enterprise Miner. The misclassification and average square error statistics are of most interest for this analysis.

The Output Window







The Output window provides information generated by the SAS procedures that are used to generate the analysis results. For the decision tree, this information includes variable importance, tree leaf report, model fit statistics, classification information, and score rankings.

3.5 Autonomous Tree Growth Options (Self-Study)

Autonomous Decision Tree Defaults

Default Settings

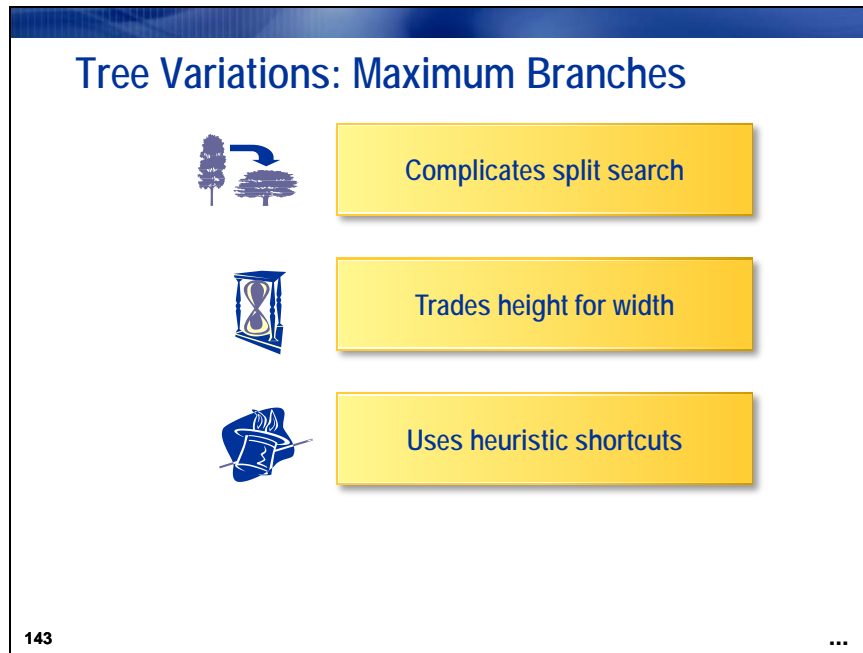
	Maximum Branches	2
	Splitting Rule Criterion	Logworth
	Subtree Method	Average Profit
	Tree Size Options	Bonferroni Adjust Split Adjust Maximum Depth Leaf Size

142
...

The behavior of the tree algorithm in SAS Enterprise Miner is governed by many parameters that can be divided into four groups:

- the number of splits to create at each partitioning opportunity
- the metric used to compare different splits
- the method used to prune the tree model
- the rules used to stop the autonomous tree growing process

The defaults for these parameters generally yield good results for an initial prediction.



SAS Enterprise Miner accommodates a multitude of variations in the default tree algorithm. The first involves the use of multiway splits instead of binary splits.

Theoretically, there is no clear advantage in doing this. Any multiway split can be obtained using a sequence of binary splits. The primary change is cosmetic. Trees with multiway splits tend to be wider than trees with only binary splits.

The inclusion of multiway splits complicates the split-search algorithm. A simple linear search becomes a search whose complexity increases geometrically in the number of splits allowed from a leaf. To combat this complexity explosion, the Tree tool in SAS Enterprise Miner resorts to heuristic search strategies.

Tree Variations: Maximum Branches

Train	
Variables	
Interactive	
Use Frozen Tree	No
Use Multiple Targets	No
Splitting Rule	
Interval Criterion	ProbF
Nominal Criterion	ProbChisq
Ordinal Criterion	Entropy
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	2
Maximum Depth	6
Minimum Categorical Size	5
Node	
Leaf Size	5
Number of Rules	5
Number of Surrogate Rules	0
Split Size	
Split Search	
Exhaustive	5000
Node Sample	20000
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25

Maximum branches in split

Exhaustive search size limit




144

...

Two fields in the Properties panel affect the number of splits in a tree. The Maximum Branch property sets an upper limit on the number of branches emanating from a node. When this number is greater than the default of 2, the number of possible splits rapidly increases. To save computation time, a limit is set in the Exhaustive property as to how many possible splits are explicitly examined. When this number is exceeded, a heuristic algorithm is used in place of the exhaustive search described above.

The heuristic algorithm alternately merges branches and reassigns consolidated groups of observations to different branches. The process stops when a binary split is reached. Among all candidate splits considered, the one with the best worth is chosen. The heuristic algorithm initially assigns each consolidated group of observations to a different branch, even if the number of such branches is more than the limit allowed in the final split. At each merge step, the two branches that degrade the worth of the partition the least are merged. After the two branches are merged, the algorithm considers reassigning consolidated groups of observations to different branches. Each consolidated group is considered in turn, and the process stops when no group is reassigned.

Tree Variations: Splitting Rule Criterion

	Yields similar splits
	Grows enormous trees
	Favors many-level inputs

145 ...

In addition to changing the number of splits, you can also change how the splits are evaluated in the split-search phase of the tree algorithm. For categorical targets, SAS Enterprise Miner offers three separate split-worth criteria. Changing from the chi-squared default criterion typically yields similar splits if the number of distinct levels in each input is similar. If not, the other split methods tend to favor inputs with more levels due to the multiple comparison problem discussed above. You can also cause the chi-squared method to favor inputs with more levels by turning off the Bonferroni adjustments.

Because Gini reduction and Entropy reduction criteria lack the significance threshold feature of the chi-squared criterion, they tend to grow enormous trees. Pruning and selecting a tree complexity based on validation profit limit this problem to some extent.

Tree Variations: Splitting Rule Criterion

Property	Value
Splitting Rule	
Interval Criterion	ProbF
Nominal Criterion	ProbChisq
Ordinal Criterion	Entropy
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	2
Maximum Depth	6
Minimum Categorical Size	5
Node	
Leaf Size	5
Number of Rules	5
Number of Surrogate Rules	0
Split Size	
Split Search	
Exhaustive	5000
Node Sample	20000
Splitting	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25
Cross Validation	
Perform Cross Validation	No
Number of Subsets	10
Number of Repeats	1
Seed	12345
Observation Based Importa	
Observation Based Importa	No
Number Single Var Importa	5
P-value Adjustment	
Bonferroni Adjustment	Yes
Time of Kass Adjustment	Before
Inputs	No
Number of Inputs	1
Split Adjustment	Yes

Split Criterion

Chi-square logworth
Entropy
Gini

Categorical Criteria

Variance
Prob-F logworth

Interval Criteria

Logworth adjustments

146

A total of five choices in SAS Enterprise Miner can evaluate split worth. Three (chi-squared logworth, entropy, and Gini) are used for categorical targets, and the remaining two (variance and ProbF logworth) are reserved for interval targets.

Both chi-squared and ProbF logworths are adjusted (by default) for multiple comparisons. It is possible to deactivate this adjustment.

The split worth for the entropy, Gini, and variance options are calculated as shown below. Let a set of cases S be partitioned into p subsets S_1, \dots, S_p so that

$$S = \bigcup_{i=1}^p S_i$$

Let the number of cases in S equal N and the number of cases in each subset S_i equal n_i . Then the worth of a particular partition of S is given by the following:

$$worth = I(S) - \sum_{i=1}^p w_i I(S_i)$$

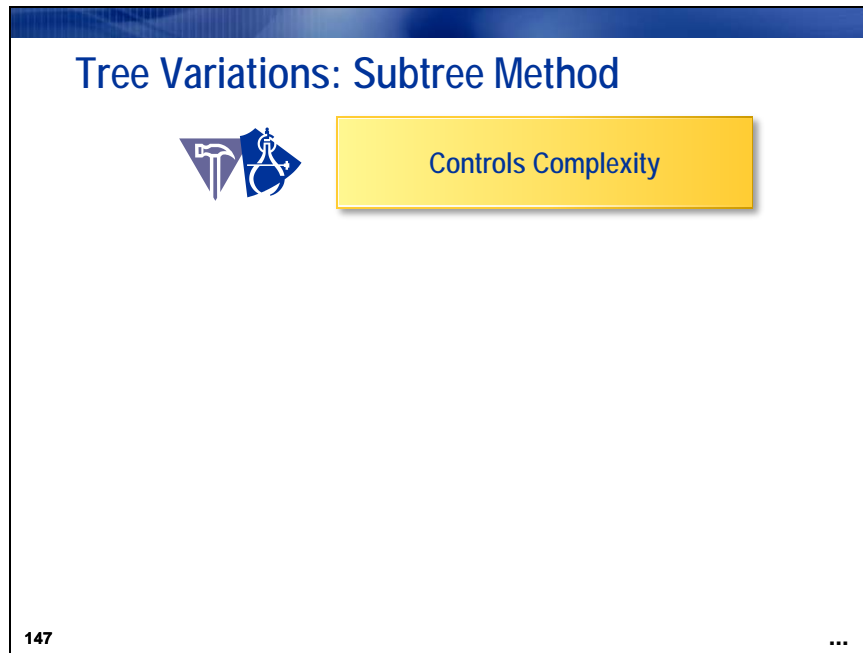
where $w_i = n_i/N$ (the proportion of cases in subset S_i), and for the specified split worth measure, $I(\cdot)$ has the following value:

$$I(\cdot) = \sum_{\text{classes}} p_{\text{class}} \cdot \log_2 p_{\text{class}} \quad \text{Entropy}$$

$$I(\cdot) = 1 - \sum_{\text{classes}} p_{\text{class}}^2 \quad \text{Gini}$$

$$I(\cdot) = \frac{1}{N_{\text{node cases}}} \sum (Y_{\text{case}} - \bar{Y})^2 \quad \text{Variance}$$

Each worth statistic measures the change in $I(S)$ from node to branches. In the Variance calculation, \bar{Y} is the average of the target value in the node with Y_{case} as a member.



SAS Enterprise Miner features two settings for regulating the complexity of subtree: modify the pruning process or the Subtree method.

By changing the model assessment measure to average square error, you can construct what is known as a *class probability tree*. It can be shown that this action minimizes the imprecision of the tree. Analysts sometimes use this model assessment measure to select inputs for a flexible predictive model such as neural networks.

You can deactivate pruning entirely by changing the subtree to **Largest**. You can also specify that the tree be built with a fixed number of leaves.

Tree Variations: Subtree Method

Property	Value
Splitting Rule	
Interval Criterion	ProbF
Nominal Criterion	ProbChisq
Ordinal Criterion	Entropy
Significance Level	0.2
Missing Values	Use In search
Use Input Once	No
Maximum Branch	2
Maximum Depth	6
Minimum Categorical Size	5
Nodes	
Leaf Size	5
Number of Rules	5
Number of Surrogate Rules	0
Split Size	
Split Search	
Exhaustive	5000
Node Sample	20000
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25
Cross Validation	
Perform Cross Validation	No
Number of Subsets	10
Number of Repeats	1
Seed	12345
Observation Based Importa	
Observation Based Importa	No
Number Single Var Importa	5
Bayesian Adjustment	
Bayesian Adjustment	Yes

Assessment
Largest
N

Pruning options
Pruning metrics

Decision
Average Square Error
Misclassification
Lift

148 ...

The pruning options are controlled by two properties: Method and Assessment Measure.

Tree Variations: Tree Size Options

 Avoids orphan nodes

 Controls sensitivity

 Grows large trees

149 ...

The family of adjustments that you modify most often when building trees are the rules that limit the growth of the tree. Changing the minimum number of observations required for a split search and the minimum number of observations in a leaf prevents the creation of leaves with only one or a handful of cases. Changing the significance level and the maximum depth allows for larger trees that can be more sensitive to complex input and target associations. The growth of the tree is still limited by the depth adjustment made to the threshold.

Tree Variations: Tree Size Options

Property	Value
<input type="checkbox"/> Splitting Rule	
<input type="checkbox"/> Interval Criterion	ProbF
<input type="checkbox"/> Nominal Criterion	ProbChisq
<input type="checkbox"/> Ordinal Criterion	Entropy
<input type="checkbox"/> Significance Level	0.2
<input type="checkbox"/> Missing Values	Use in search
<input type="checkbox"/> Use Input Once	No
<input type="checkbox"/> Maximum Branch	2
<input type="checkbox"/> Maximum Depth	6
<input type="checkbox"/> Minimum Categorical Size	5
<input type="checkbox"/> Node	
<input type="checkbox"/> Leaf Size	5
<input type="checkbox"/> Number of Rules	5
<input type="checkbox"/> Number of Surrogate Rules	0
<input type="checkbox"/> Split Size	
<input type="checkbox"/> Split Search	
<input type="checkbox"/> Exhaustive	5000
<input type="checkbox"/> Node Sample	20000
<input type="checkbox"/> Splitting	
<input type="checkbox"/> Method	Assessment
<input type="checkbox"/> Number of Leaves	1
<input type="checkbox"/> Assessment Measure	Average Square Error
<input type="checkbox"/> Assessment Fraction	0.25
<input type="checkbox"/> Cross Validation	
<input type="checkbox"/> Perform Cross Validation	No
<input type="checkbox"/> Number of Subsets	10
<input type="checkbox"/> Number of Repeats	1
<input type="checkbox"/> Seed	12345
<input type="checkbox"/> Observation Based Importa	
<input type="checkbox"/> Observation Based Importa	No
<input type="checkbox"/> Number Single Var Importa	5
<input type="checkbox"/> P-Value Adjustment	
<input type="checkbox"/> Bonferroni Adjustment	Yes
<input type="checkbox"/> Time of Kass Adjustment	Before
<input type="checkbox"/> Inputs	No
<input type="checkbox"/> Number of Inputs	1
<input type="checkbox"/> Split Adjustment	Yes

150

Annotations:

- Logworth threshold (points to Significance Level)
- Maximum tree depth (points to Maximum Depth)
- Minimum leaf size (points to Leaf Size)
- Threshold depth adjustment (points to Split Adjustment)

Changing the logworth threshold changes the minimum logworth required for a split to be considered by the tree algorithm (when using the chi-squared or ProbF measures). Increasing the minimum leaf size avoids orphan nodes. For large data sets, you might want to increase the maximum leaf setting to obtain additional modeling resolution. If you want big trees and insist on using the chi-squared split-worth criterion, deactivate the Split Adjustment option.



Exercises

1. Initial Data Exploration

A supermarket is offering a new line of organic products. The supermarket's management wants to determine which customers are likely to purchase these products.

The supermarket has a customer loyalty program. As an initial buyer incentive plan, the supermarket provided coupons for the organic products to all of the loyalty program participants and collected data that includes whether these customers purchased any of the organic products.

The **ORGANICS** data set contains 13 variables and over 22,000 observations. The variables in the data set are shown below with the appropriate roles and levels:

Name	Model Role	Measurement Level	Description
ID	ID	Nominal	Customer loyalty identification number
DemAffl	Input	Interval	Affluence grade on a scale from 1 to 30
DemAge	Input	Interval	Age, in years
DemCluster	Rejected	Nominal	Type of residential neighborhood
DemClusterGroup	Input	Nominal	Neighborhood group
DemGender	Input	Nominal	M = male, F = female, U = unknown
DemRegion	Input	Nominal	Geographic region
DemTVReg	Input	Nominal	Television region
PromClass	Input	Nominal	Loyalty status: tin, silver, gold, or platinum
PromSpend	Input	Interval	Total amount spent
PromTime	Input	Interval	Time as loyalty card member
TargetBuy	Target	Binary	Organics purchased? 1 = Yes, 0 = No
TargetAmt	Rejected	Interval	Number of organic products purchased



Although two target variables are listed, these exercises concentrate on the binary variable **TargetBuy**.

- a. Create a new diagram named **Organics**.

- b. Define the data set **AAEM61.ORGANICS** as a data source for the project.
- 1) Set the roles for the analysis variables as shown above.
 - 2) Examine the distribution of the target variable. What is the proportion of individuals who purchased organic products? _____
 - 3) The variable **DemClusterGroup** contains collapsed levels of the variable **DemCluster**. Presume that, based on previous experience, you believe that **DemClusterGroup** is sufficient for this type of modeling effort. Set the model role for **DemCluster** to Rejected.
 - 4) As noted above, only **TargetBuy** will be used for this analysis and should have a role of Target. Can **TargetAmt** be used as an input for a model used to predict **TargetBuy**? Why or why not? _____

 - 5) Finish the **Organics** data source definition.
- c. Add the **AAEM61.ORGANICS** data source to the Organics diagram workspace.
- d. Add a **Data Partition** node to the diagram and connect it to the **Data Source** node. Assign 50% of the data for training and 50% for validation.
- e. Add a **Decision Tree** node to the workspace and connect it to the **Data Partition** node.
- f. Create a decision tree model autonomously. Use average square error as the model assessment statistic.
- 1) How many leaves are in the optimal tree? _____
 - 2) Which variable was used for the first split? _____
What were the competing splits for this first split? _____
- g. Add a second **Decision Tree** node to the diagram and connect it to the **Data Partition** node.
- 1) In the Properties panel of the new Decision Tree node, change the maximum number of branches from a node to **3** to allow for three-way splits.
 - 2) Create a decision tree model using average square error as the model assessment statistic.
 - 3) How many leaves are in the optimal tree? _____
- h. Based on average square error, which of the decision tree models appears to be better? _____

3.6 Chapter Summary

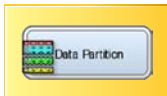
Predictive modeling fulfills some of the promise of data mining. Past observation of target events coupled with past input measurements enables some degree of assistance for the prediction of future target event values, conditioned on current input measurements. All of this depends on the stationarity of the process being modeled.

A predictive model must perform three essential tasks:

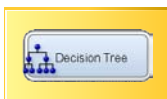
1. It must be able to systematically predict new cases, based on input measurements. These predictions might be decisions, rankings, or estimates.
2. A predictive model must be able to thwart the curse of dimensionality and successfully select a limited number of useful inputs. This is accomplished by systematically ignoring irrelevant and redundant inputs.
3. A predictive model must adjust its complexity to avoid overfitting or underfitting. In SAS Enterprise Miner, this adjustment is accomplished by splitting the raw data into a training data set for building the model and a validation data set for gauging model performance. The performance measure used to gauge performance depends on the prediction type. Decisions can be evaluated by accuracy or misclassification, rankings by concordance or discordance, and estimates by average square error.

Decision trees are a simple but potentially useful example of a predictive model. They score new cases by creating prediction rules. They select useful inputs via a split-search algorithm. They optimize complexity by building and pruning a maximal tree. A decision tree model can be built interactively, automatically, or autonomously. For autonomous models, the Decision Tree Properties panel provides options to control the size and shape of the final tree.

Decision Tree Tools Review



Partition raw data into training and validation sets.



Interactively grow trees using the Tree Desktop application. You can control rules, selected inputs, and tree complexity.

Autonomously grow decision trees based on property settings. Settings include branch count, split rule criterion, subtree method, and tree size options.

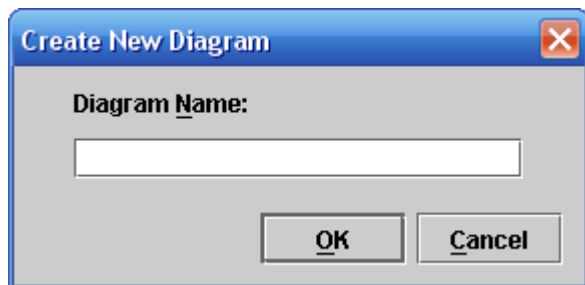
3.7 Solutions

Solutions to Exercises

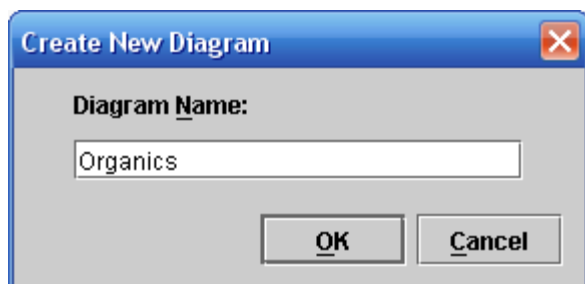
1. Initial Data Exploration

- a. Create a new diagram named **Organics**.

1) Select **File** ⇒ **New** ⇒ **Diagram...**. The Create New Diagram window opens.

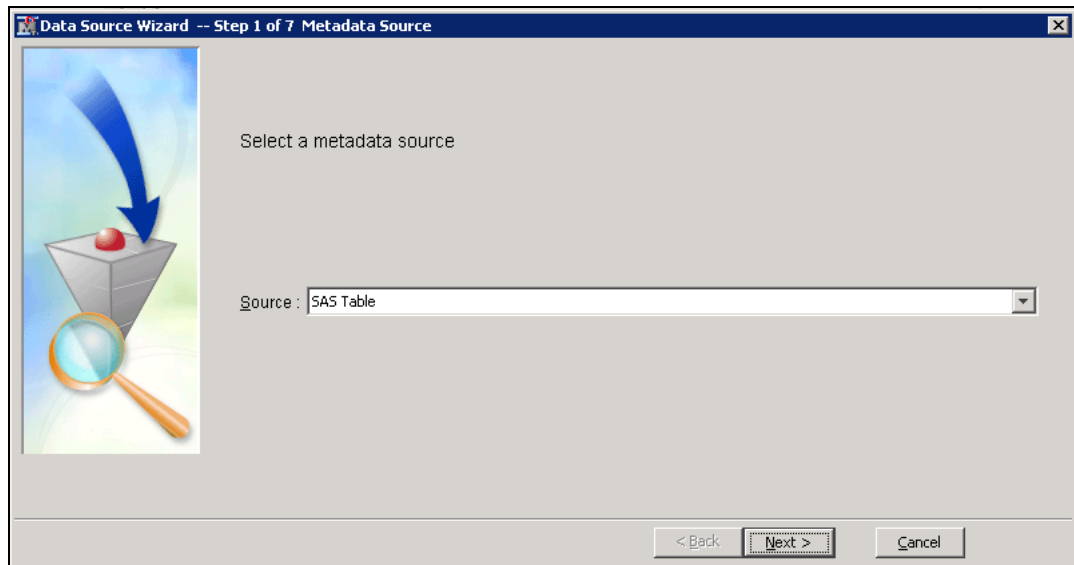


2) Type **Organics** in the Diagram Name field.

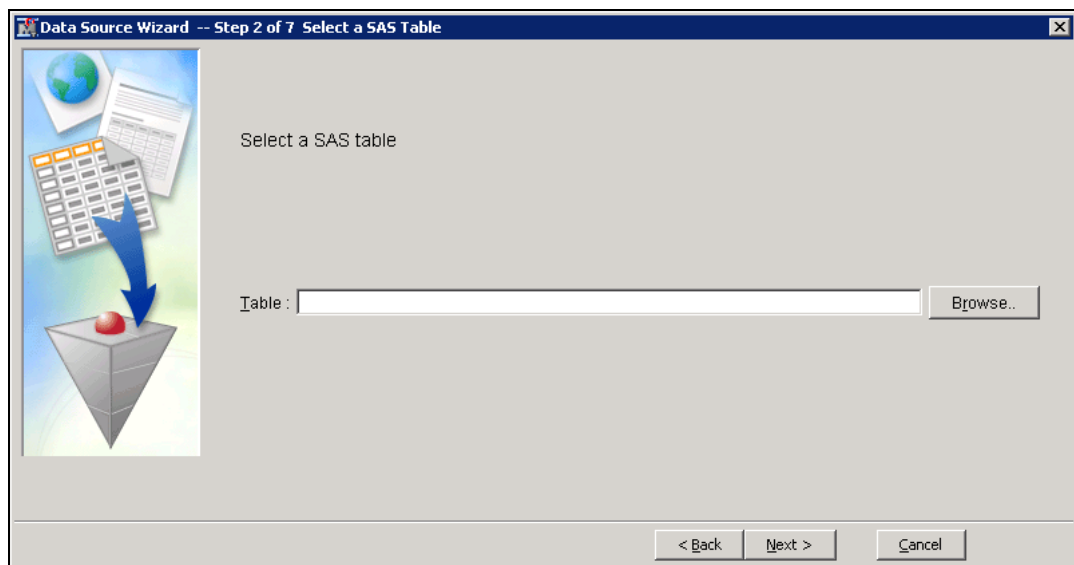


3) Select **OK**.

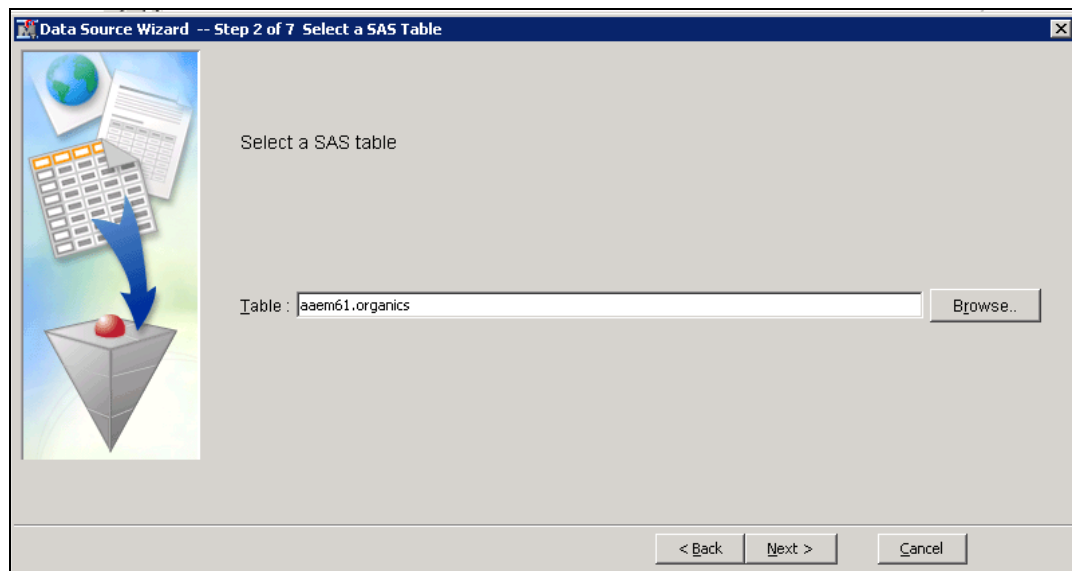
- b. Define the data set **AAEM61.ORGANICS** as a data source for the project.
- 1) Set the model role for the target variable.
 - 2) Examine the distribution of the target variable. What is the proportion of individuals who purchased organic products?
 - a) Select **File** ⇒ **New** ⇒ **Data Source...**. The Data Source Wizard window opens.



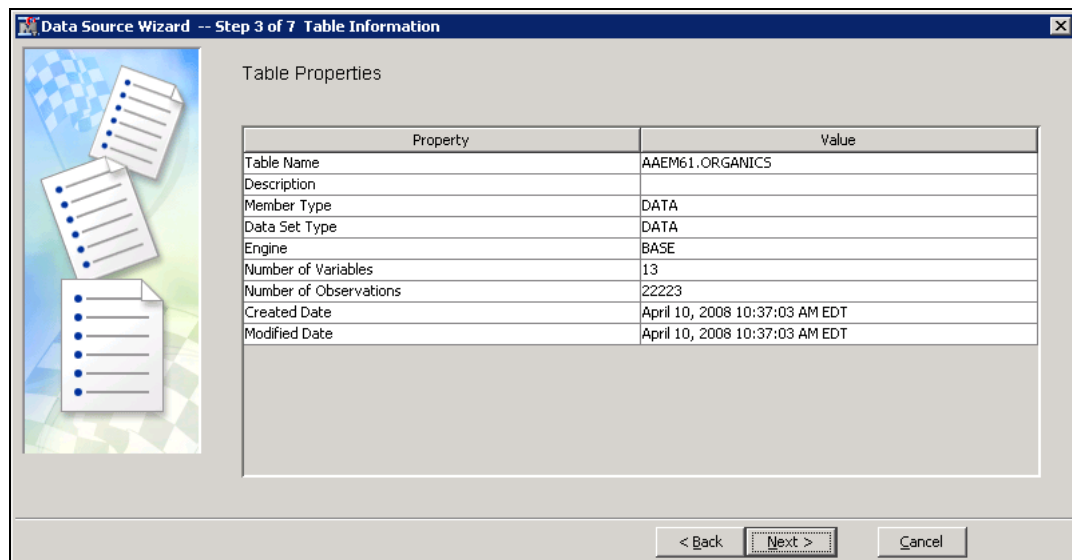
- b) Select **Next >**. The wizard proceeds to Step 2.



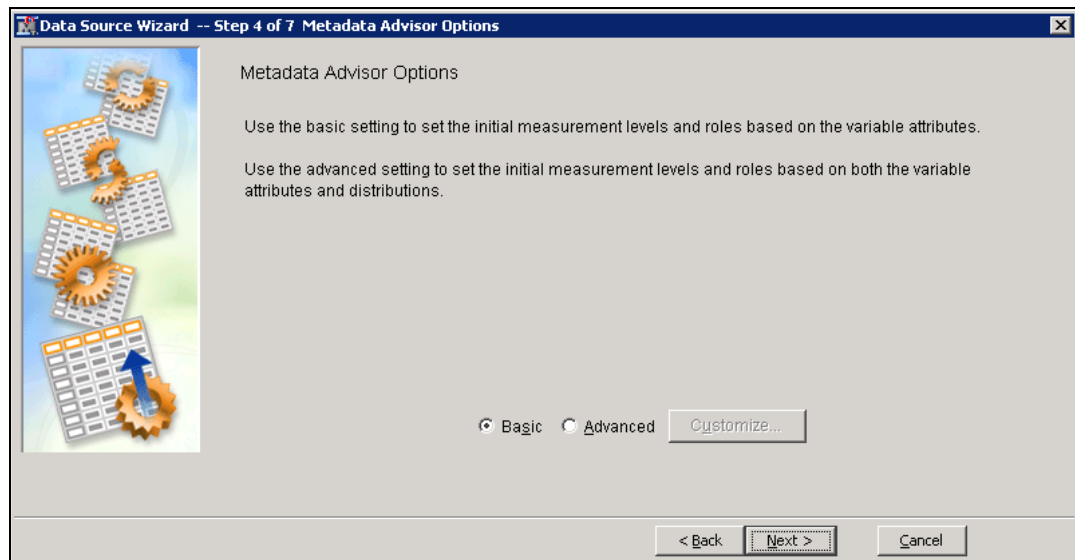
- c) Type **AAEM61.ORGANICS** in the Table field.



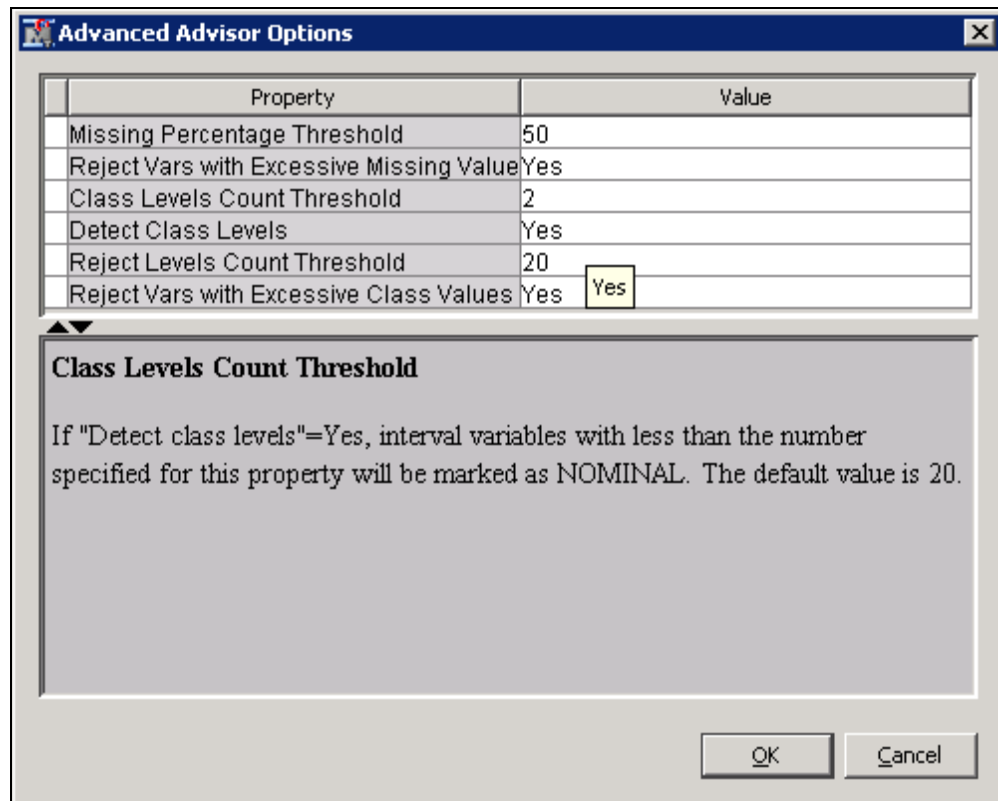
- d) Select **Next >**. The wizard proceeds to Step 3.



- e) Select **Next >**. The wizard proceeds to Step 4.



- f) Select the **Advanced** radio button and select **Customize...**. The Advanced Advisor Options window opens.
- g) Type **2** as the Class Levels Count Threshold value.



- h) Select **OK**. The Advanced Advisor Options window closes and you are returned to Step 4 of the Data Source Wizard.

i) Select **Next >**. The wizard proceeds to Step 5.



By customizing the Advanced Metadata Advisor, most of the roles and levels are correctly set.

j) Select **Role** ⇒ **Rejected** for **TargetAmt**.

Data Source Wizard -- Step 5 of 7: Column Metadata

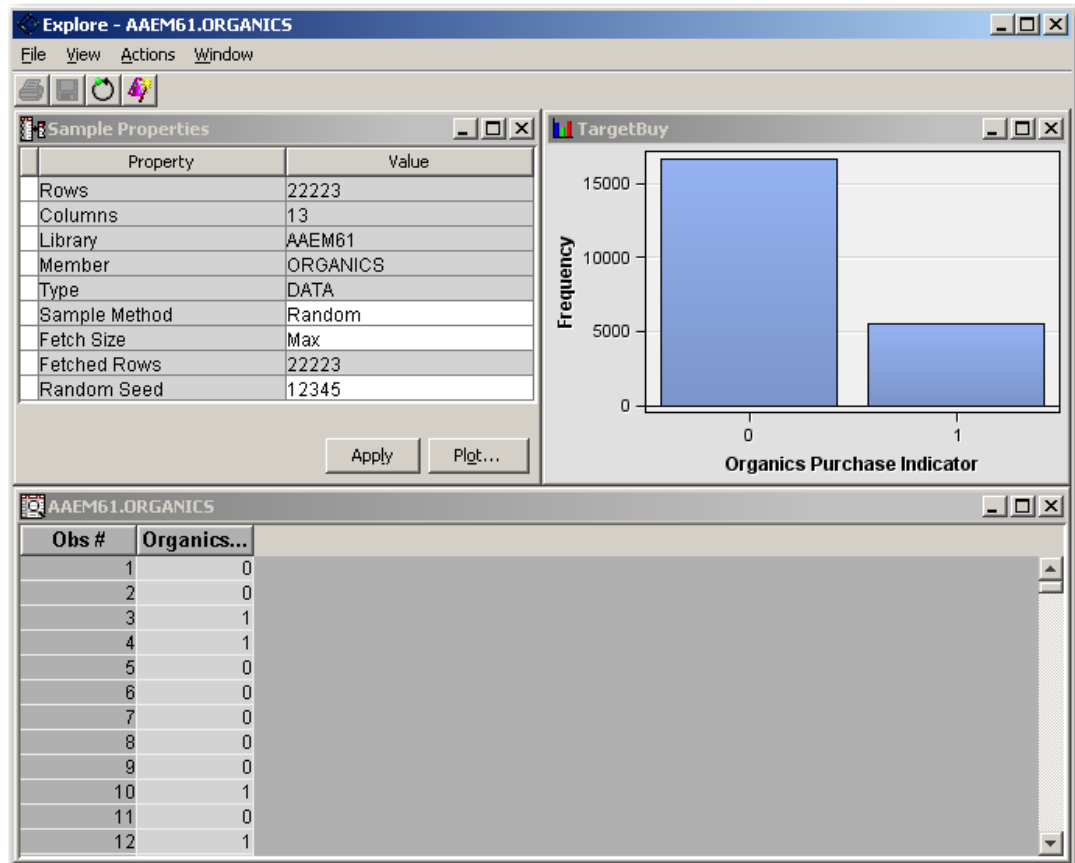
(none) ☐ not Equal to ☐ Apply ☐ Reset

Columns: ☐ Label ☐ Mining ☐ Basic ☐ Statistics

Name	Role	Level	Report	Order	Drop	Lower Limit	Upper Limit
DemAffl	Input	Interval	No		No	.	
DemAge	Input	Interval	No		No	.	
DemCluster	Rejected	Nominal	No		No	.	
DemClusterG	Input	Nominal	No		No	.	
DemGender	Input	Nominal	No		No	.	
DemReg	Input	Nominal	No		No	.	
DemTVReg	Input	Nominal	No		No	.	
ID	ID	Nominal	No		No	.	
PromClass	Input	Nominal	No		No	.	
PromSpend	Input	Interval	No		No	.	
PromTime	Input	Interval	No		No	.	
TargetAmt	Rejected	Interval	No		No	.	
TargetBuy	Target	Binary	No		No	.	

Show code Explore Refresh Summary < Back Next > Cancel

k) Select **TargetBuy** and select **Explore**. The Explore window opens.



The proportion of individuals who purchased organic products appears to be 25%.

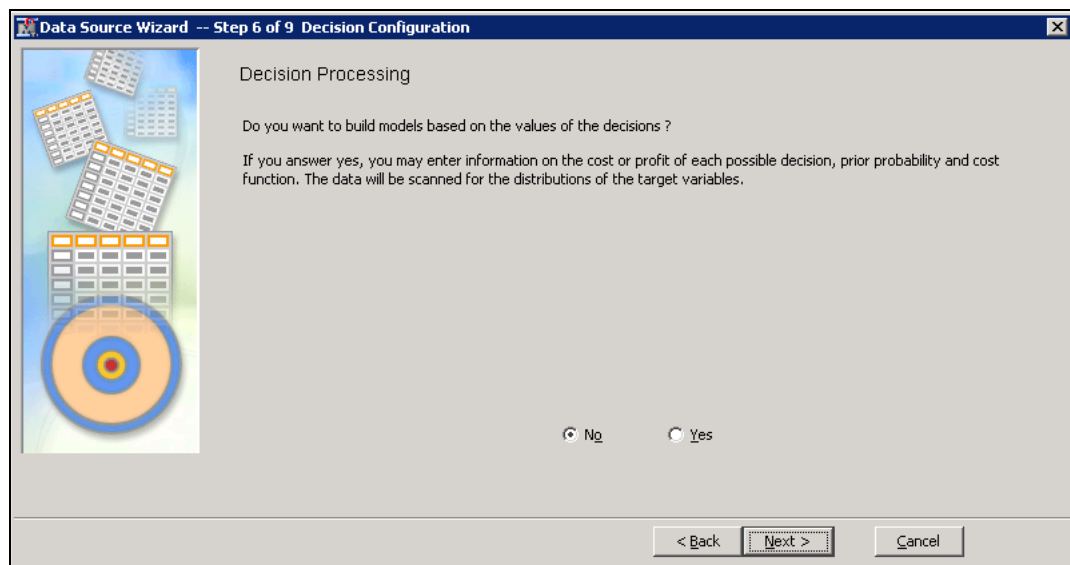
- 1) Close the Explore window.
- 3) The variable **DemClusterGroup** contains collapsed levels of the variable **DemCluster**. Presume that, based on previous experience, you believe that **DemClusterGroup** is sufficient for this type of modeling effort. Set the model role for **DemCluster** to Rejected.

This is already done using the Advanced Metadata Advisor. Otherwise, select **Role** ⇨ **Rejected** for **DemCluster**.

- 4) As noted above, only **TargetBuy** will be used for this analysis and should have a role of Target. Can **TargetAmt** be used as an input for a model used to predict **TargetBuy**? Why or why not?

No, using TargetAmt as an input is not possible. It is measured at the same time as TargetBuy and therefore has no *causal* relationship to TargetBuy.

- 5) Finish the **Organics** data source definition.
 - a) Select **Next >**. The wizard proceeds to Step 6.



- b) Select **Next >**. The wizard proceeds to Step 7.

Data Source Wizard -- Step 7 of 8: Data Source Attributes

You may change the name and the role, and can specify a population segment identifier for the data source to be created.

Name :

Role :

Segment :

Notes :

< Back **Next >** Cancel

- c) Select **Next >**. The wizard proceeds to Step 8.

Data Source Wizard -- Step 8 of 8: Summary

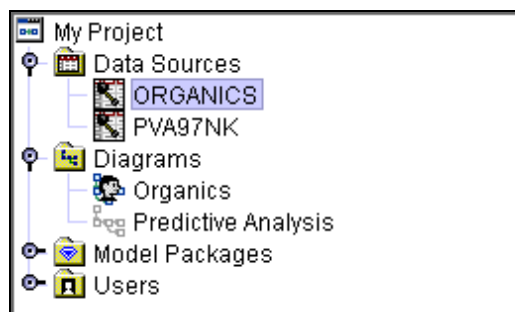
Metadata Completed.

Library: AAEM61
Table: ORGANICS
Role: Raw

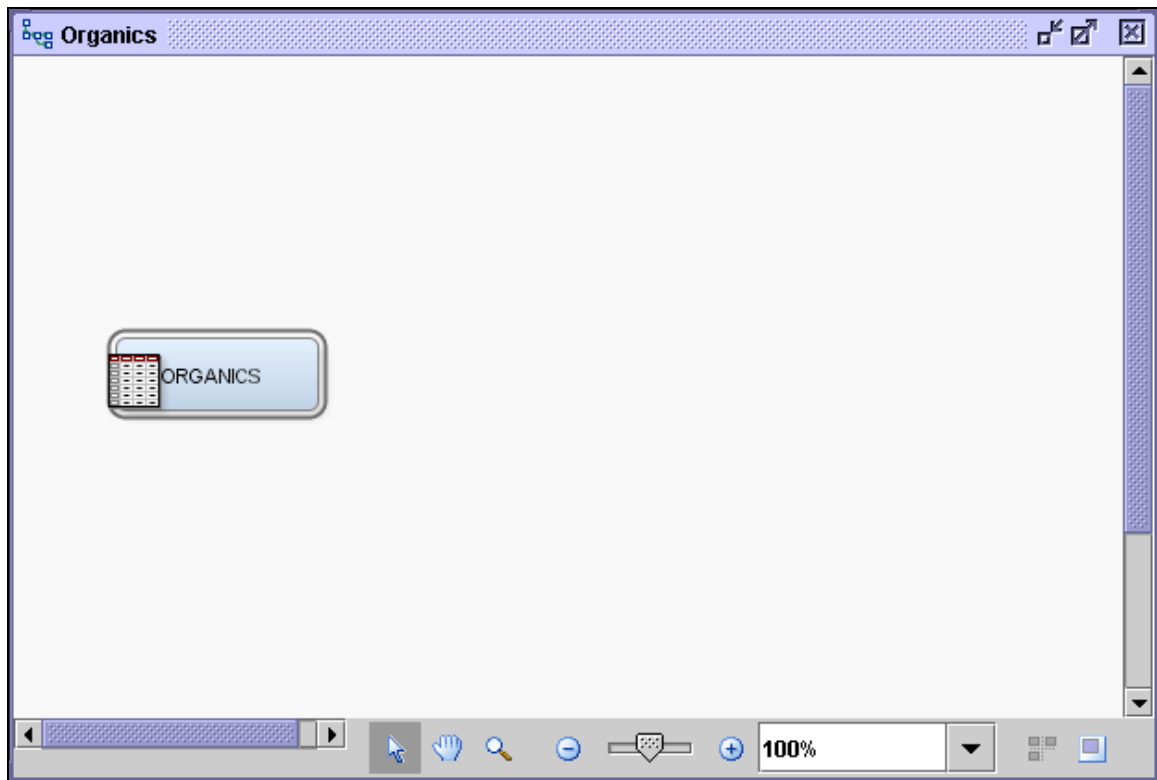
Role	Level	Count
ID	Nominal	1
Input	Interval	4
Input	Nominal	5
Rejected	Interval	1
Rejected	Nominal	1
Target	Binary	1

< Back **Finish** Cancel

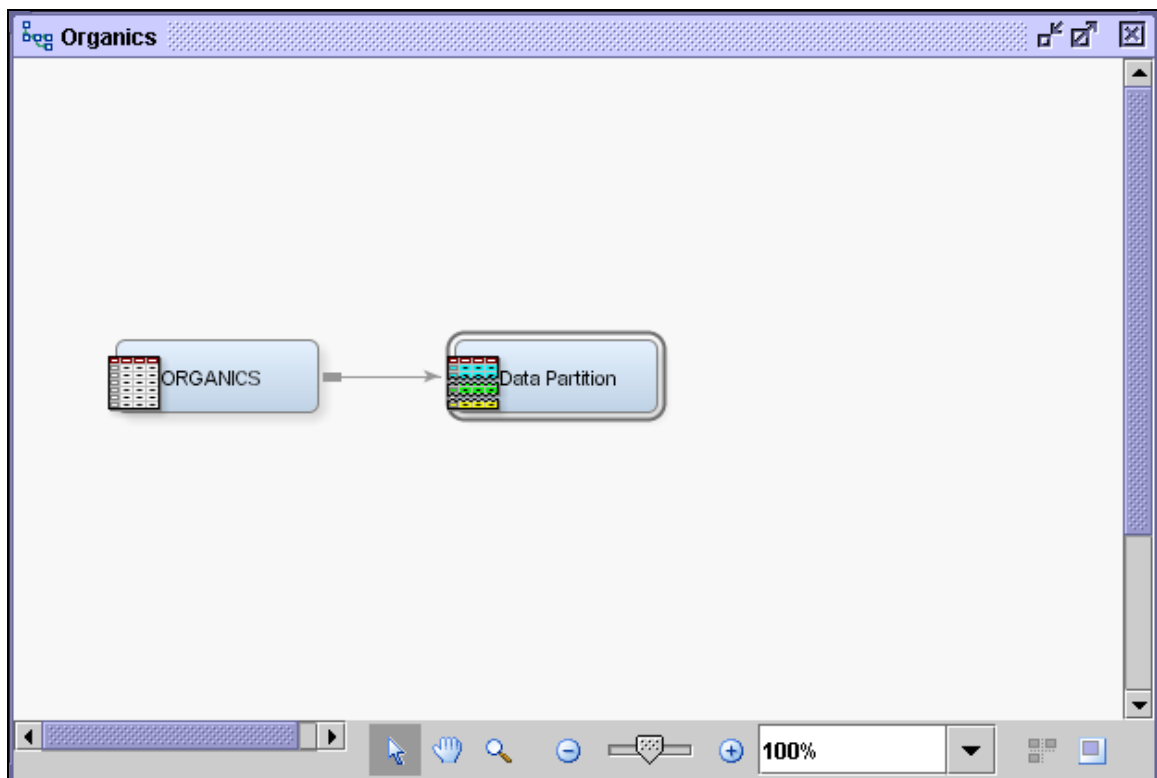
- d) Select **Finish**. The wizard closes and the **Organics** data source is ready for use in the Project Panel.



- c. Add the **AAEM61.ORGANICS** data source to the Organics diagram workspace.



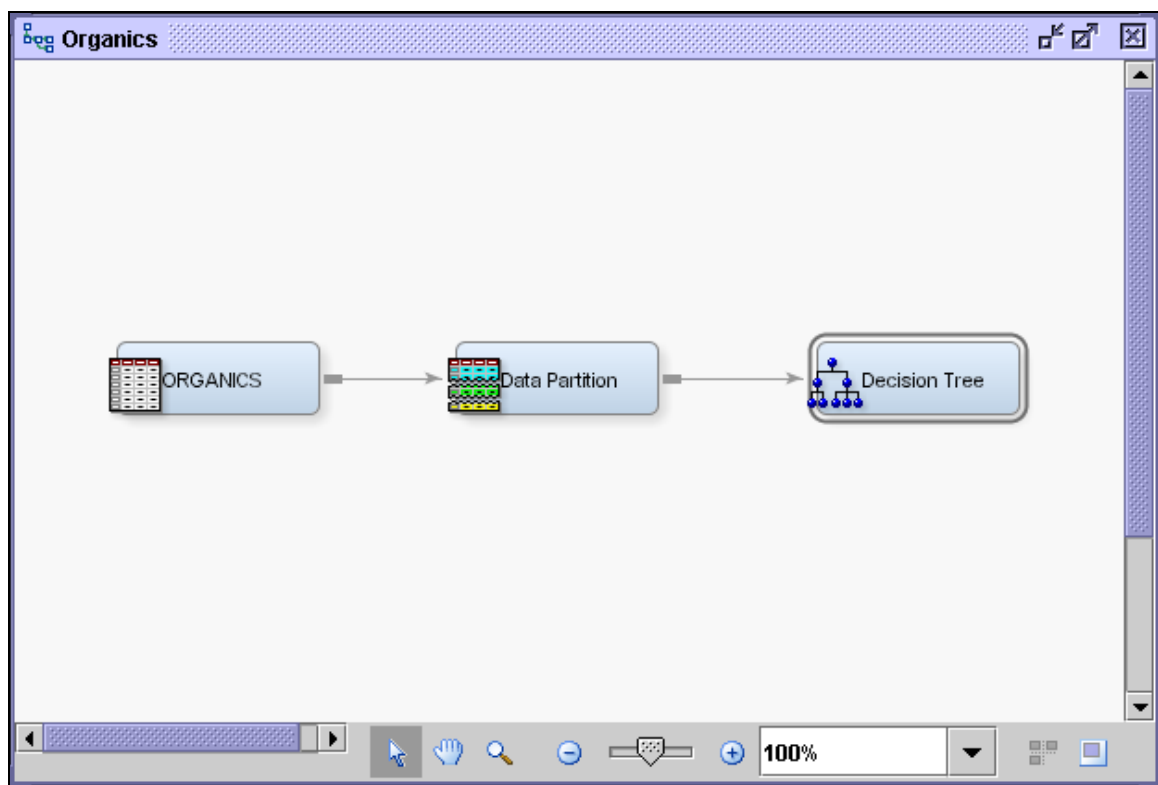
- d. Add a **Data Partition** node to the diagram and connect it to the **Data Source** node. Assign 50% of the data for training and 50% for validation.



- 1) Type **50** as the Training and Validation values under Data Set Allocations.
- 2) Type **0** as the Test value.

Train	
Variables	
Output Type	Data
Partitioning Method	Default
Random Seed	12345
Data Set Allocations	
Training	50.0
Validation	50.0
Test	0.0

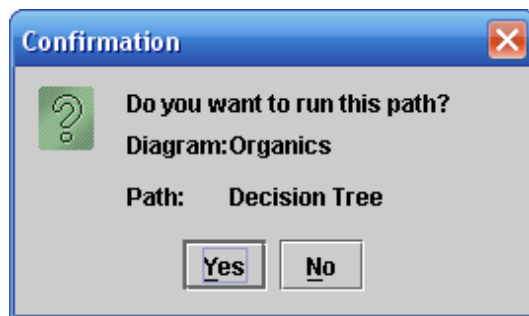
- e. Add a **Decision Tree** node to the workspace and connect it to the **Data Partition** node.



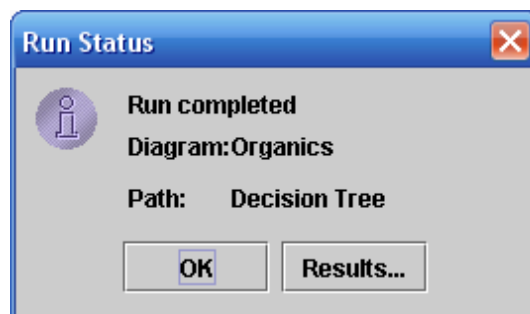
- f. Create a decision tree model autonomously. Use average square error as the model assessment statistic.
- Select **Average Square Error** as the Assessment Measure property.

Train	
Variables	
Interactive	
Splitting Rule	
Criterion	Default
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	2
Maximum Depth	6
Minimum Categorical S	5
Node	
Leaf Size	5
Number of Rules	5
Number of Surrogate R	0
Split Size	
Split Search	
Exhaustive	5000
Node Sample	20000
Subtree	
Method	Assessment
Number of Leaves	1
Assessment Measure	Average Square Error
Assessment Fraction	0.25

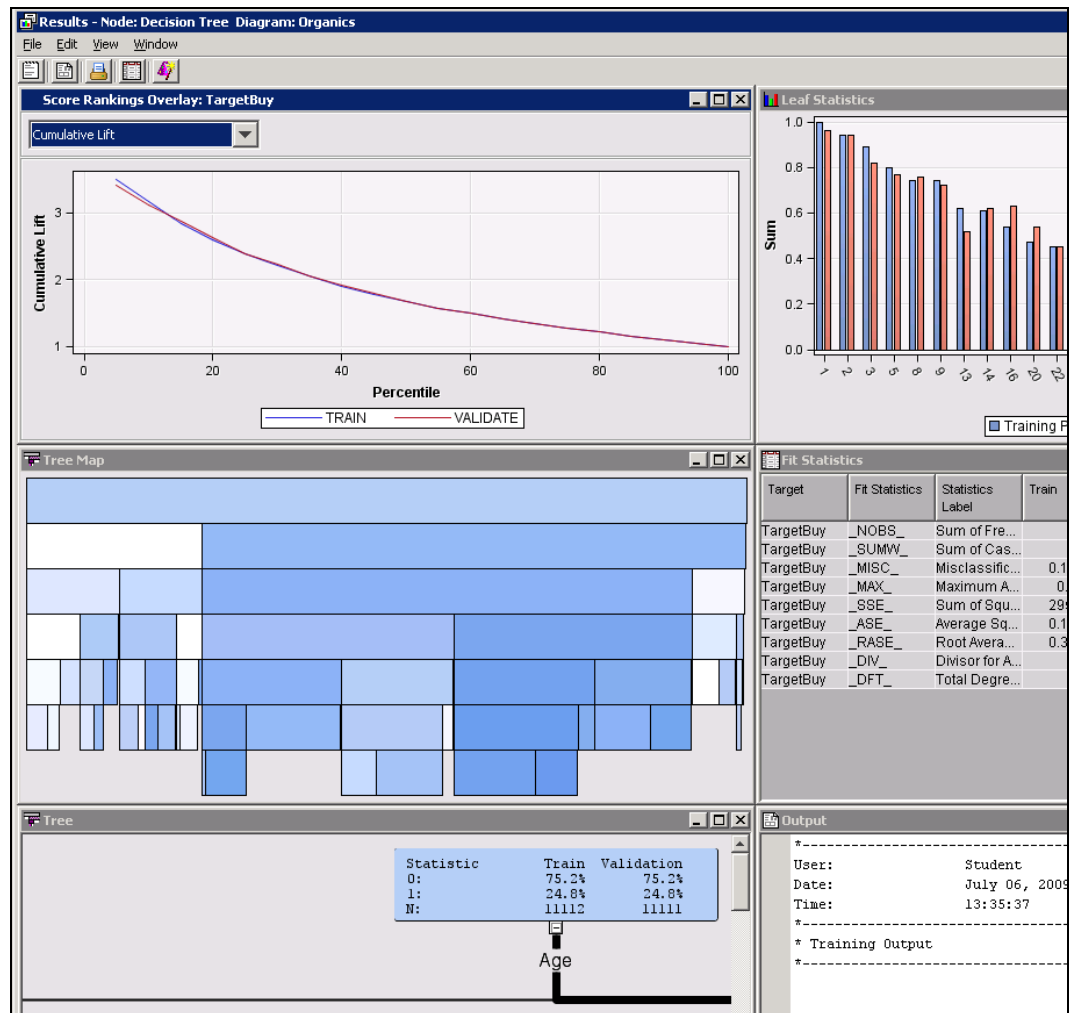
- Right-click on the **Decision Tree** node and select **Run** from the Option menu.
- Select **Yes** in the Confirmation window.



- 1) How many leaves are in the optimal tree?

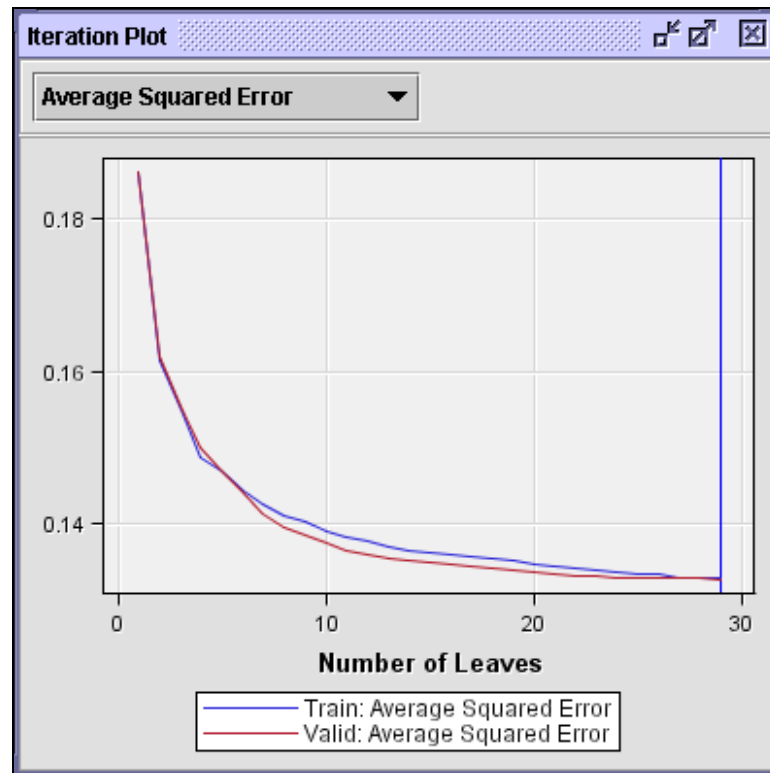


- a) When the Decision Tree node run finishes, select **Results...** from the Run Status window. The Results window opens.



The easiest way to determine the number of leaves in your tree is via the iteration plot.

- b) Select **View** ⇒ **Model** ⇒ **Iteration Plot** from the Result window menu. The Iteration Plot window opens.



Using average square error as the assessment measure results in a tree with 29 leaves.

- 2) Which variable was used for the first split? What were the competing splits for this first split?



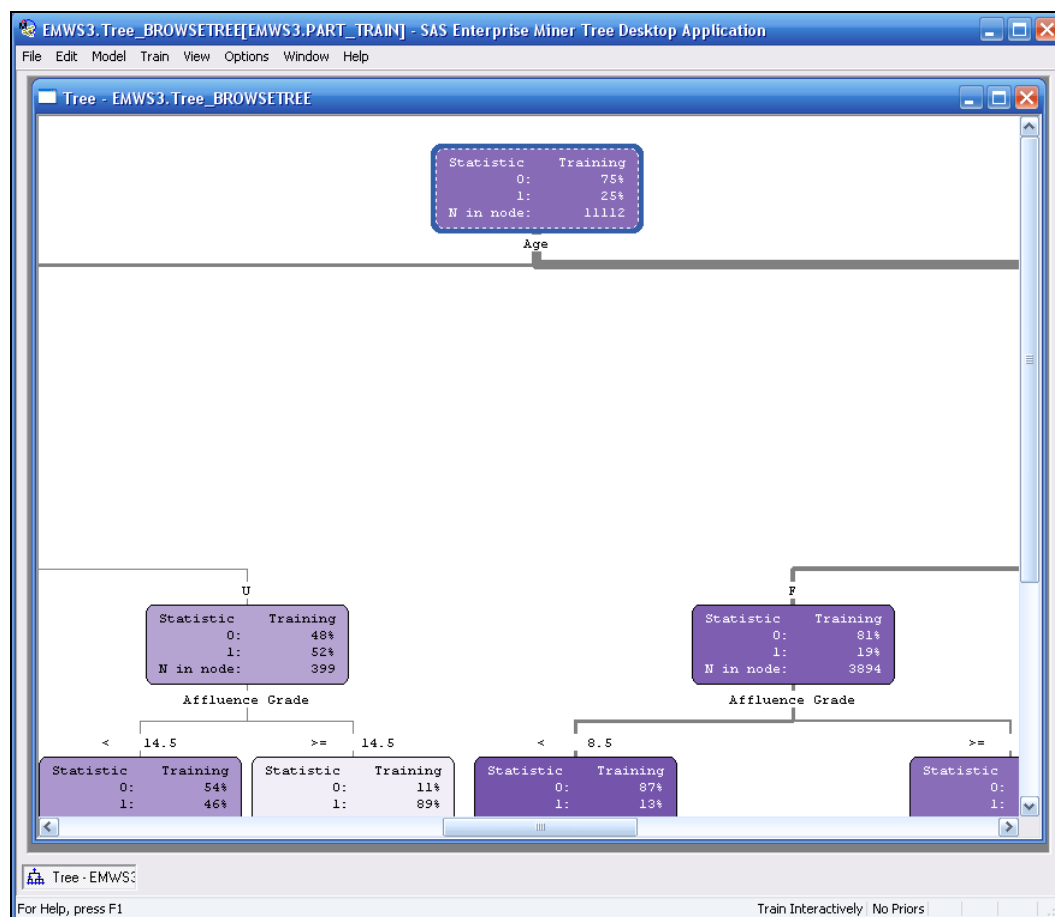
These questions are best answered using interactive training.

- a) Close the Results window for the Decision Tree model.

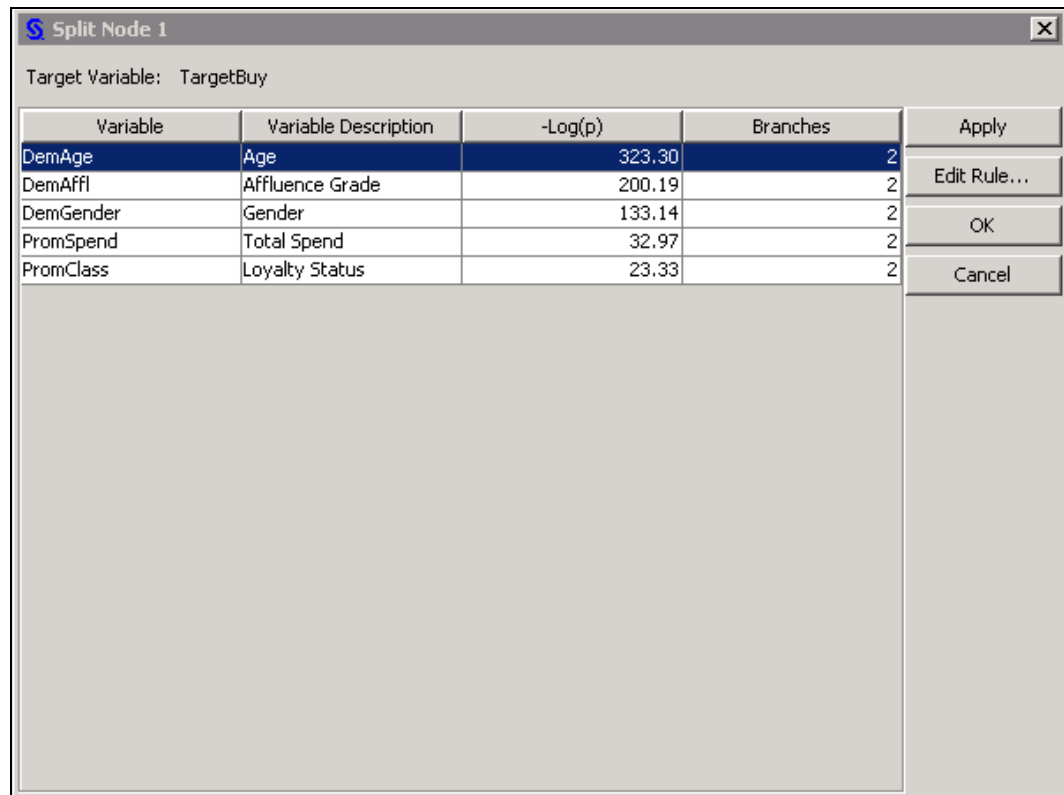
Train	
Variables	...
Interactive	...

- b) Select the Interactive ellipsis (...) from the Decision Tree node's Properties panel.

The SAS Enterprise Miner Tree window opens.



- c) Right-click the root node and select **Split Node...** from the Option menu. The Split Node 1 window opens with information that answers the two questions.



Split Node 1

Target Variable: TargetBuy

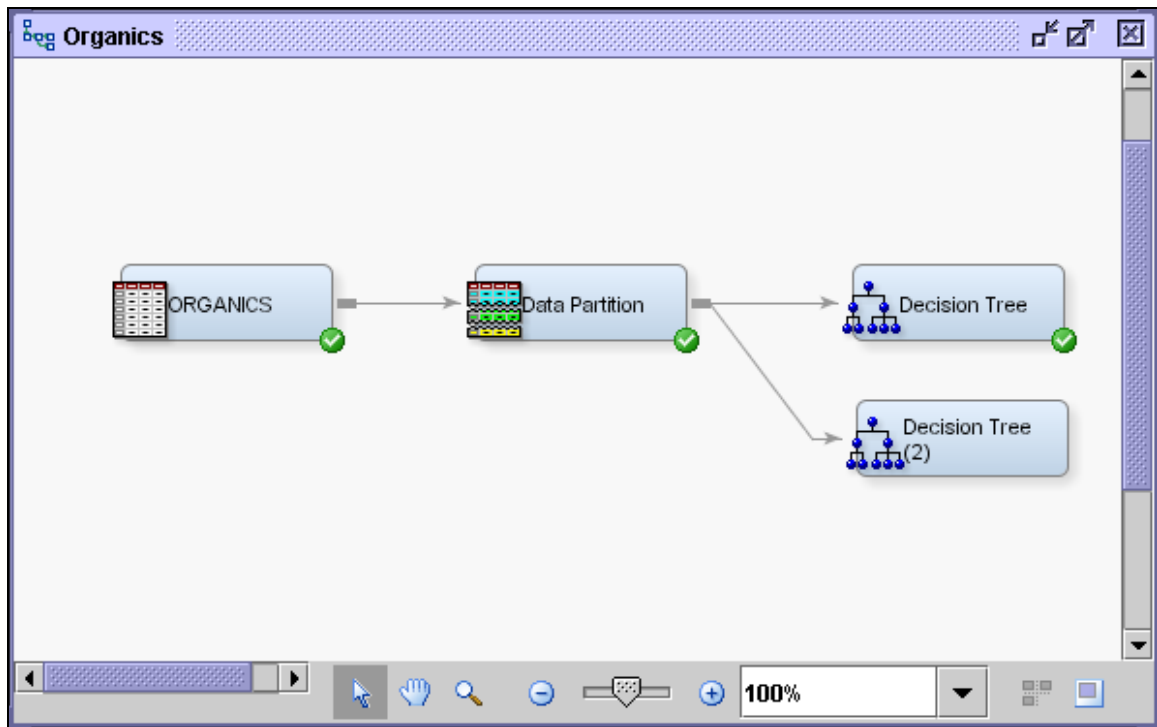
Variable	Variable Description	-Log(p)	Branches
DemAge	Age	323.30	2
DemAffl	Affluence Grade	200.19	2
DemGender	Gender	133.14	2
PromSpend	Total Spend	32.97	2
PromClass	Loyalty Status	23.33	2

Buttons: Apply, Edit Rule..., OK, Cancel

Age is used for the first split.

Competing splits are **Affluence Grade**, **Gender**, **Total Spend**, and **Loyalty Status**.

- g. Add a second **Decision Tree** node to the diagram and connect it to the **Data Partition** node.



- 1) In the Properties panel of the new Decision Tree node, change the maximum number of branches from a node to **3** to allow for three-way splits.

Train	
Variables	
Interactive	
Splitting Rule	
Criterion	Default
Significance Level	0.2
Missing Values	Use in search
Use Input Once	No
Maximum Branch	3
Maximum Depth	6
Minimum Categorical S	5

- 2) Create a decision tree model again using average square error as the model assessment statistic.
- 3) How many leaves are in the optimal tree.

Based on the discussion above, the optimal tree with three-way splits has 33 leaves.

h. Based on average square error, which of the decision tree models appears to be better?

- 1) Select the first **Decision Tree** node.
- 2) Right-click and select **Results...** from the Option menu. The Results window opens.
- 3) Examine the Average Squared Error row of the Fit Statistics window.

Target	Fit Statistics	Statistics Label	Train	Validation
TargetBuy	_ASE_	Average Squared Error	0.132861	0.132773

- 4) Close the Results window.
- 5) Repeat the process for the Decision Tree (2) model.

Target	Fit Statistics ▲	Statistics Label	Train	Validation	Test
TargetBuy	_ASE_	Average Sq...	0.133009	0.132662	.

It appears that the second tree with three-way splits has the lower validation average square error and is the better model.

Solutions to Student Activities (Polls/Quizzes)

3.01 Quiz – Correct Answer

Match the Predictive Modeling Application to the Decision Type.

Modeling Application	Decision Type
<input type="checkbox"/> C Loss Reserving	A. Decision
<input type="checkbox"/> B Risk Profiling	B. Ranking
<input type="checkbox"/> B Credit Scoring	C. Estimate
<input type="checkbox"/> A Fraud Detection	
<input type="checkbox"/> C Revenue Forecasting	
<input type="checkbox"/> A Voice Recognition	