

PREDICTIVE MODELING: DECISION TREES

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 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).

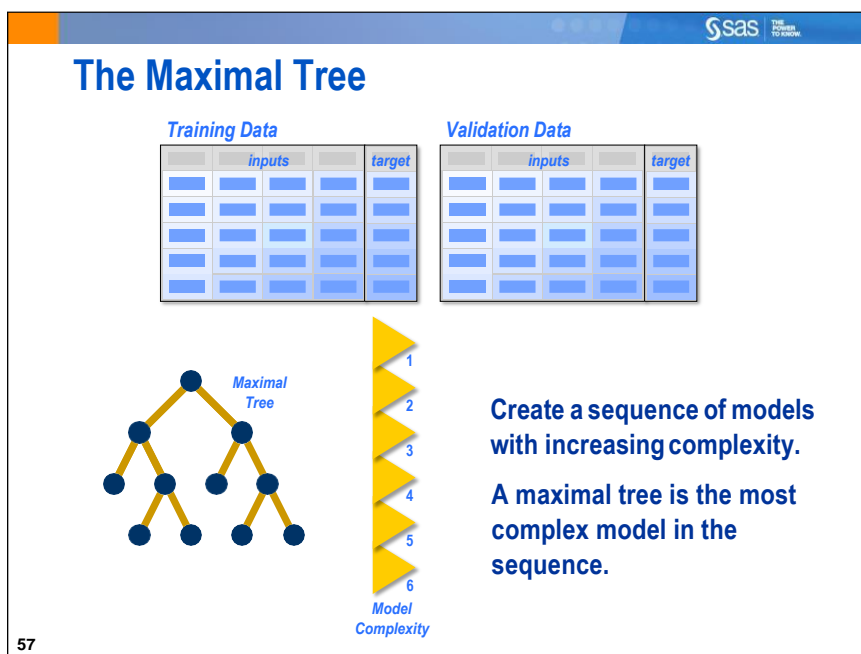
Decision Trees

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*

Modelling Complexity - Pruning

The maximal tree represents the most complicated model that 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*.

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 POWER TO KNOW.

The Maximal Tree

Training Data

inputs				target

Validation Data

inputs				target

1

2

3

4

5

Model Complexity

A maximal tree is the most complex model in the sequence.

58

The maximal tree 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 leaf fewer than the maximal tree.

THE POWER TO KNOW.

Subsequent Pruning

Training Data

inputs				target

Validation Data

inputs				target

Model Complexity

Continue pruning until all subtrees are considered.

66

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

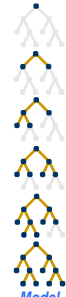
Selecting the Best Tree

Training Data

	inputs			target

Validation Data

	inputs			target



Model Complexity Assessment

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

★★★★★

Compare validation assessment between tree complexities.

67

Consider the validation assessment rating for each of the subtrees.

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




★★★★★

What are appropriate validation assessment ratings?

69

Validation assessment ratings.

 THE POWER OF DATA

Assessment Statistics


Validation Data

	inputs			target

Ratings depend on...

- ☆☆☆☆ target measurement (binary, continuous, and so on)
- ☆☆☆☆ prediction type (decisions, rankings, estimates)

70

 THE POWER OF DATA

Binary Target Predictions

	inputs			target
				1
				0
				0
				1
				1

→

prediction
primary
secondary
520
720
0.249

decisions

rankings

estimates


72

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.

Next, 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.


THE POWER TO KNOW

Ranking Optimization – Concordance

inputs			target	prediction
1	1	1	1	secondary
1	1	1	0	primary
1	1	1	0	520
1	1	1	1	720
1	1	1	1	6.249


target=0 → low score

target=1 → high score

Maximize concordance:
proper ordering of
primary and secondary
outcomes

77

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, the proportion of such pairs whose scores are correctly ordered.


THE POWER TO KNOW

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

78

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, the proportion of such pairs whose scores are incorrectly ordered.


THE POWER TO KNOW

Complexity Optimization – Summary

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

decisions
accuracy / misclassification

rankings
concordance / discordance

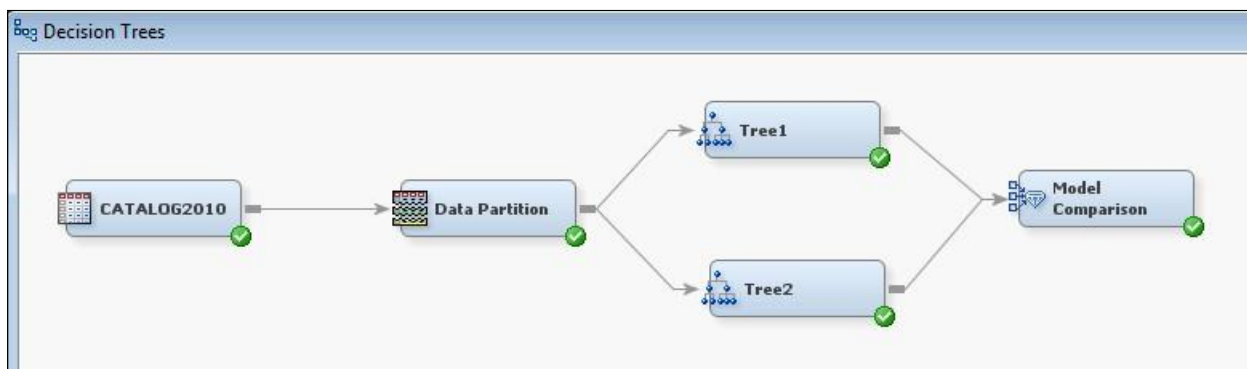
estimates
squared error

81

SAS Enterprise Miner also produces a summary statistic named the *ROC index*. This equals the percent of concordant cases plus one-half times the percent of tied cases. The area under the ROC curve corresponds to the ROC index (also called the c-statistic).

CATALOG CASE STUDY: CONSTRUCTING A DECISION TREE PREDICTIVE MODEL

Use the **CATALOG2010** data from Part 1 and fit two decision tree models in SAS Enterprise Miner. The diagram below shows the nodes that are needed to fit decision tree models. The steps include splitting the data into training and validation data sets using the Data Partition node, selecting useful inputs using Decision Tree nodes, and generating model assessment statistics and plots using the Model Comparison node.

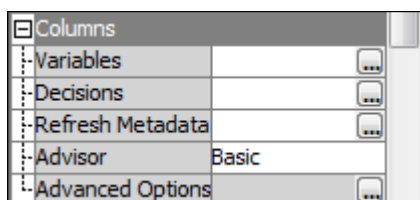


1. Use the project from Chapter 2 and add a new diagram.

Select **File** **New** **Diagram**. Name the diagram **Decision Trees**. Click **OK**.

2. Select the **CATALOG2010** data source.

- a. Drag the **CATALOG2010** data source onto the Decision Trees diagram.
- b. From the Properties panel on the left, click the ellipsis next to **Variables**.



The variable **ORDERSIZE** is a target for this data set, but for this part focus on the binary response variable **RESPOND**. To prevent the decision tree from using **ORDERSIZE**, you must set its role to **Rejected**.

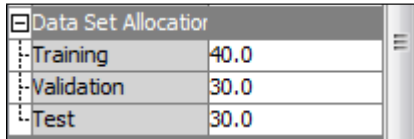
3. Set **ORDERSIZE** to **Rejected**.

- a. Change the role of the variable **ORDERSIZE** to **Rejected**.

MONLAST	Input	Interval	No		No	.	.
ORDERSIZE	Rejected	Interval	No		No	.	.
PCPAYM	Rejected	Binary	No		No	.	.
RESPOND	Residual	Binary	No		No	.	.
STATE	Segment	Nominal	No		No	.	.
TENURE	Sequence	Interval	No		No	.	.
TOTORDQ01	Target	Interval	No		No	.	.
TOTORDQ02	Text	Interval	No		No	.	.
TOTORDQ03	Text Locat	Interval	No		No	.	.
TOTORDQ04	Time ID	Interval	No		No	.	.
TOTORDQ05	Input	Interval	No		No	.	.

- b. Click **OK** to close the Variables window.
4. Partition the data set into training and validation data sets.
 - a. Click the **Sample** tab and drag a **Data Partition** node into the diagram workspace.
 - b. Connect the **CATALOG2010** data node to the **Data Partition** node. Select the **Data Partition** node and examine its Properties panel.

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.



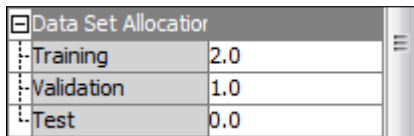
Data Set Allocation	
Training	40.0
Validation	30.0
Test	30.0

There is a trade-off in various partitioning strategies. More data is 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 two-thirds of the cases into training and one-third into validation.

5. Divide the data set into training and validation data sets.

Type **2** as the Training value, **1** as the Validation value, and **0** as the Test value in the Data Partition node. Notice that the values under Data Set Allocations do not have to add up to 100%. The proportions of the values that are entered are determined if they do not add up to 100%.



Data Set Allocation	
Training	2.0
Validation	1.0
Test	0.0

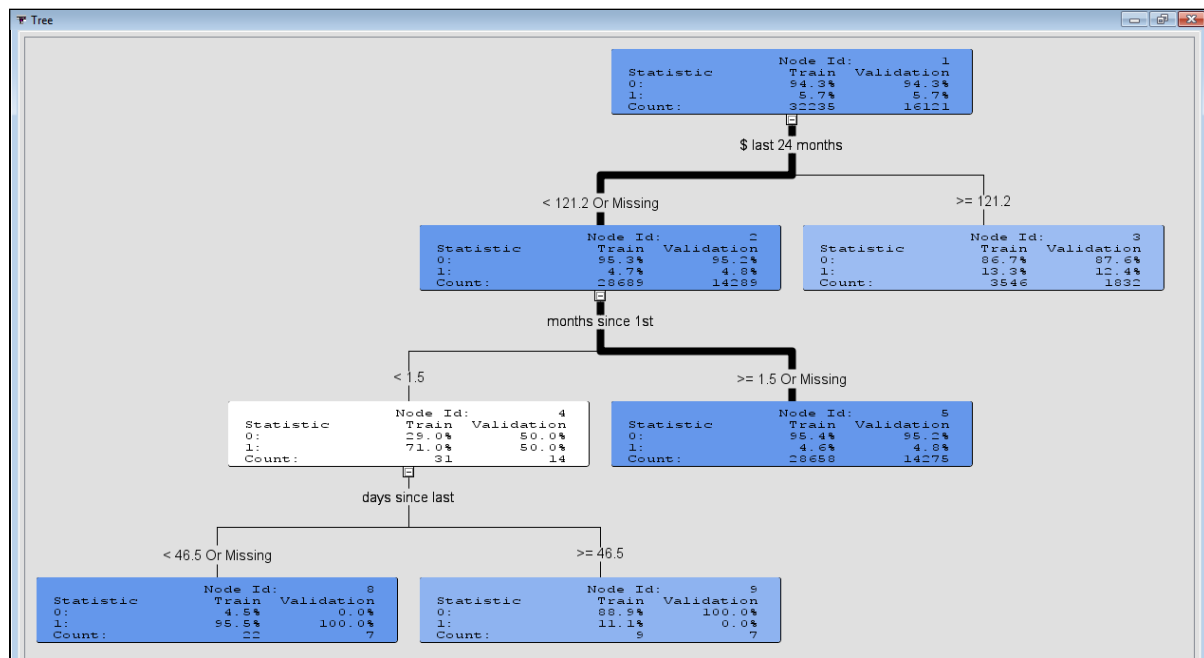
6. Run the Data Partition node and view the results.
 - a. Right-click the **Data Partition** node and click **Run**.
 - b. Click **Yes** in the Confirmation window.
 - c. SAS Enterprise Miner then runs the Data Partition node. When the data partition is complete, a Run Status window appears. Select **Results** in the Run Status window to view the results. Scroll down to the Summary Statistics for Class Targets information.

49						
50	Summary Statistics for Class Targets					
51						
52	Data=DATA					
53						
54		Numeric	Formatted	Frequency		
55	Variable	Value	Value	Count	Percent	Label
56						
57	RESPOND	0	0	45617	94.3358	response target
58	RESPOND	1	1	2739	5.6642	response target
59						
60						
61	Data=TRAIN					
62						
63		Numeric	Formatted	Frequency		
64	Variable	Value	Value	Count	Percent	Label
65						
66	RESPOND	0	0	30410	94.3385	response target
67	RESPOND	1	1	1825	5.6615	response target
68						
69						
70	Data=VALIDATE					
71						
72		Numeric	Formatted	Frequency		
73	Variable	Value	Value	Count	Percent	Label
74						
75	RESPOND	0	0	15207	94.3304	response target
76	RESPOND	1	1	914	5.6696	response target
77						

The Results - Data Partition window provides a frequency table that shows the distribution of the target variable, **RESPOND**, in the raw, training, and validation data sets.

7. Add a decision tree node to the diagram.
 - a. Close the Results window.
 - b. Click the **Model** tab and drag a **Decision Tree** node into the diagram workspace.
 - c. Connect the **Data Partition** node to the **Decision Tree** node.
8. Right-click and run the Decision Tree node. View the results.
9. View the generated decision tree.
 - a. Maximize the Tree window.

b. Right-click in the Tree window and select **View Fit to Page**.



In the initial split at the top of the diagram, the training data is partitioned into two subsets. The first subset, corresponding to cases with a dollar value of purchases in the past 24 months less than 121.20, has a higher than average concentration of **RESPOND=0** cases. The second subset, corresponding to cases with a dollar value of purchases in the past 24 months greater than or equal to 121.20, has a higher than average concentration of **RESPOND=1** cases. The first branch has more cases than the second, which is indicated by **Count**.

The partition of the data also represents a predictive model. This model assigns to all cases in the left branch a predicted **RESPOND** value equal to 0.047 and to all cases in the right branch a predicted **RESPOND** value equal to 0.133.

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).

The second split is based on the variable **TENURE** (months since first order) where the left branch contains the cases with months since first order less than 1.5, and the right branch contains the cases with months since first order greater than or equal to 1.5 or missing.

The third partition uses **DaysSinceLast** (number of days since the most recent purchase) to form a split where the cases with the number of days less than 46.5 or missing are assigned to the left branch and cases with the number of days 46.5 or greater are assigned to the right branch.

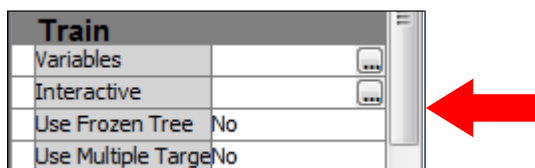
It is important to remember that the logworth of each split and the predicted **RESPOND** values are generated using the *training data only*. The main focus is on selecting useful input variables for the first predictive model. A diminishing marginal usefulness of input variables is expected, given 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.

Interactive Decision Tree (Self-Study)

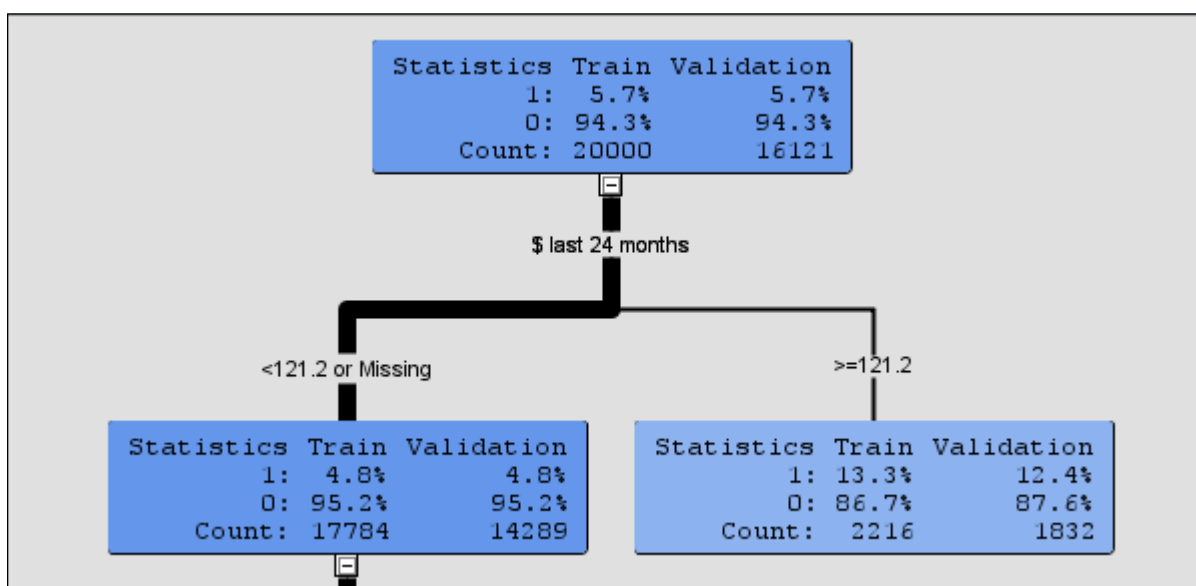
The Decision Tree tool can build predictive models autonomously or interactively. To build a model autonomously, simply run the Decision Tree node. Building models interactively, however, can be informative when you first learn about the node, and even when you are more expert at predictive modeling.

- Open the Decision Tree Tool in interactive mode (this demonstration continues from the Decision Tree constructed during the previous demonstration).

Click the ellipsis next to **Interactive** from the Decision Tree node's Properties panel



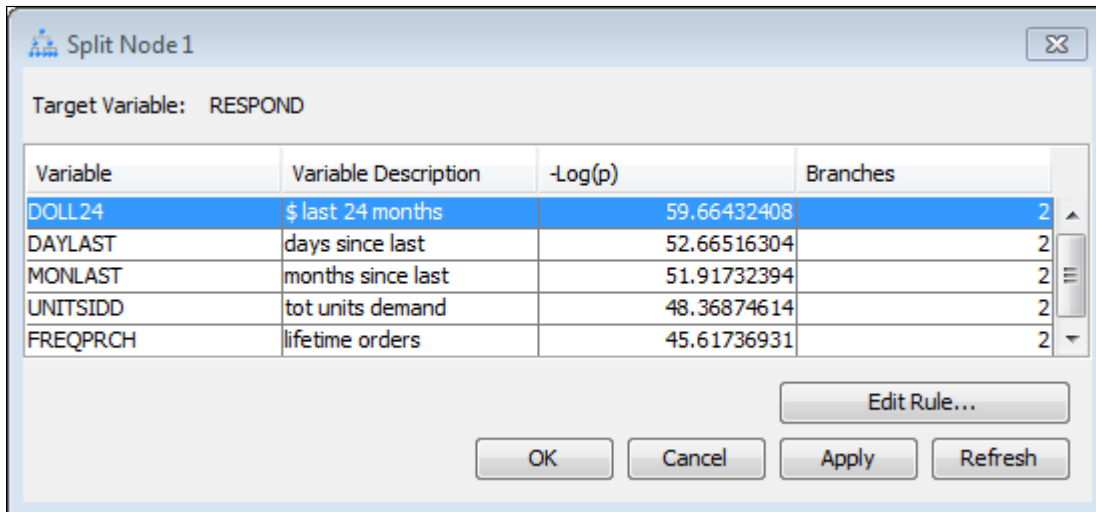
The SAS Enterprise Miner Interactive Decision Tree application opens.



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 **RESPOND**. Notice that the Train data set has been limited to 20,000 observations. This is based on a macro variable that limits data size for interactive tree fitting. You can override this limit with a macro variable in the start-up code. See the Help in SAS Enterprise Miner for more information. Limiting the data size for interactive tree building helps optimize the efficiency of the program.

11. Partition the training data set.

Right-click the top blue box and select **Split Node** from the menu. The Split Node 1 dialog box appears.



The Split Node 1 dialog box is shown with the Target Variable set to RESPOND. It contains a table with the following data:

Variable	Variable Description	-Log(p)	Branches
DOLL24	\$ last 24 months	59.66432408	2
DAYLAST	days since last	52.66516304	2
MONLAST	months since last	51.91732394	2
UNITSIDD	tot units demand	48.36874614	2
FREQPRCH	lifetime orders	45.61736931	2

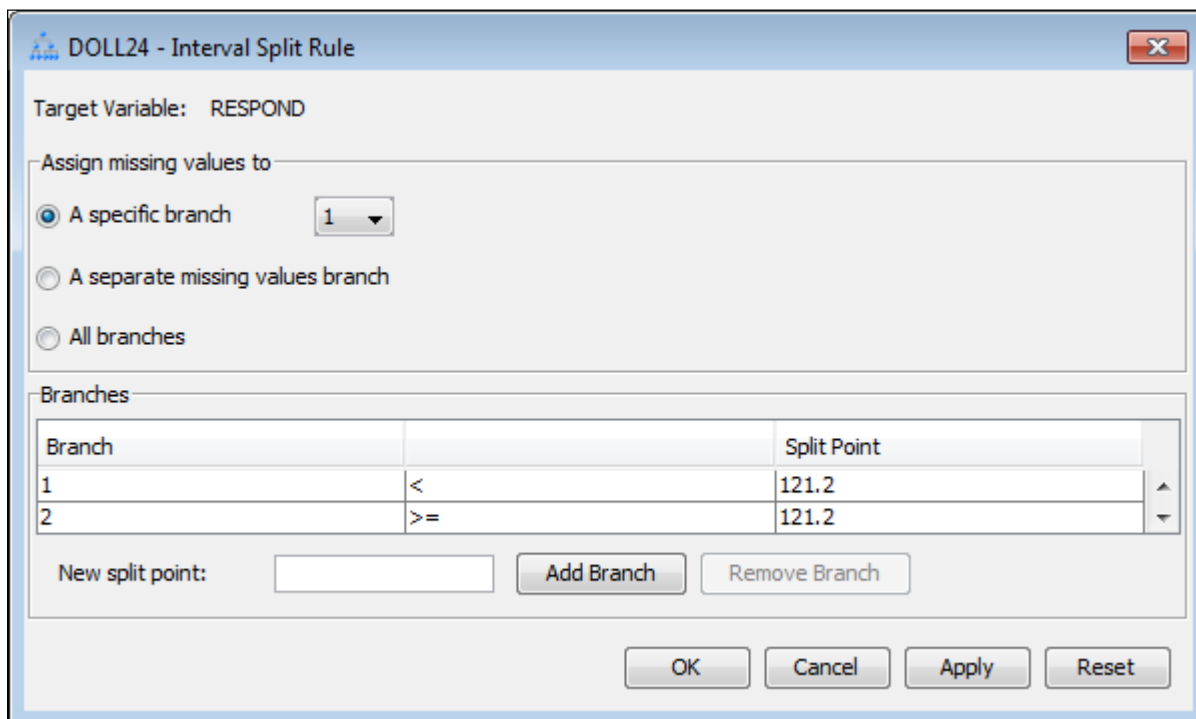
Buttons at the bottom include OK, Cancel, Apply, Refresh, and an Edit Rule... button.

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.

DOLL24 (dollar value of purchases during the past 24 months) has the highest logworth, followed by **DAYLAST** (days since last purchase) and **MONLAST** (months since last purchase). You can choose to split the data on the selected input or edit the rule for more information.

12. Edit the splitting rule.

Click **Edit Rule**. The DOLL24 - Interval Split Rule dialog box appears.



The dialog box titled "DOLL24 - Interval Split Rule" shows the configuration for splitting data based on the target variable "RESPOND".

Target Variable: RESPOND

Assign missing values to:

- ☒ A specific branch (dropdown menu shows 1)
- ☐ A separate missing values branch
- ☐ All branches

Branches:

Branch		Split Point
1	<	121.2
2	>=	121.2

New split point: **Add Branch** **Remove Branch**

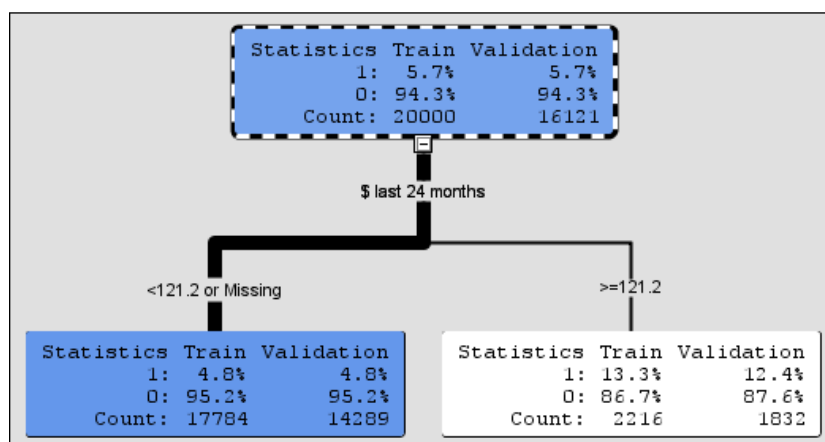
Buttons: OK, Cancel, Apply, Reset

This dialog box shows how the training data is partitioned using the input **DOLL24**. Two branches are created. The first branch contains cases with a dollar value of purchases in the past 24 months less than 121.2, and the second branch contains cases with a dollar value of purchases in the past 24 months greater than or equal to 121.2. In addition, any cases with missing or unknown dollar value of purchases in the past 24 months 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.

13. Apply the splitting rule to the root node.

Click **Apply** or **OK**. The DOLL24 - Interval Splitting Node dialog box remains open, and the Tree View window is updated.



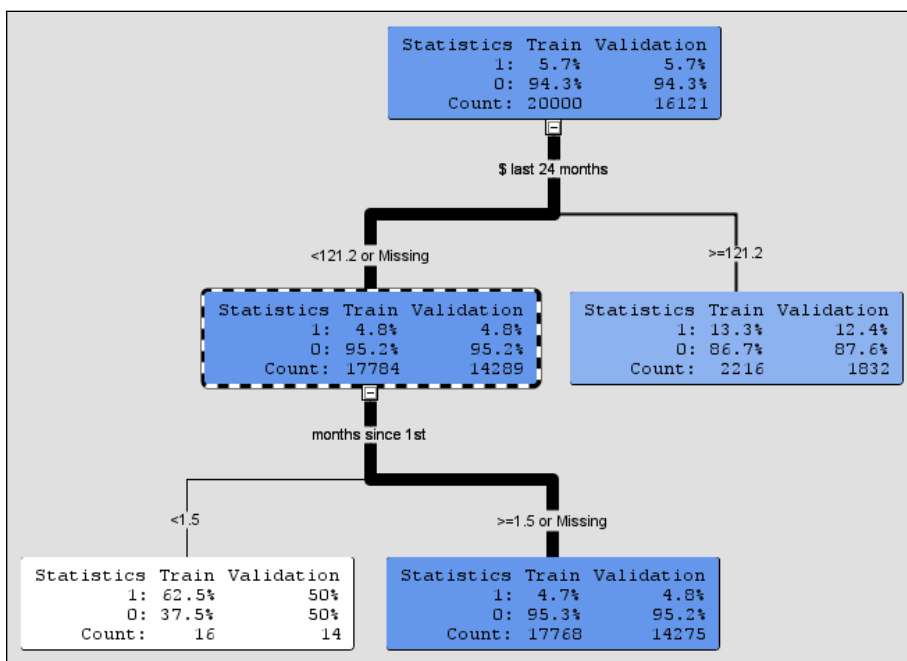
The training data is partitioned into two subsets. The first subset, corresponding to cases with a dollar value of purchases in the past 24 months less than 121.2 (or where the value is missing), has a higher than average concentration of **RESPOND**=0 cases. The second subset, corresponding to cases with a dollar value of purchases in the past 24 months greater than or equal to 121.2, has a higher than average concentration of **RESPOND**=1 cases. The first branch has more cases than the second, which is indicated by **Count**.

The partition of the data also represents a predictive model. This model assigns to all cases in the left branch a predicted **RESPOND** value equal to 0.048 and to all cases in the right branch a predicted **RESPOND** value equal to 0.124.

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).

14. Split the left leaf node.

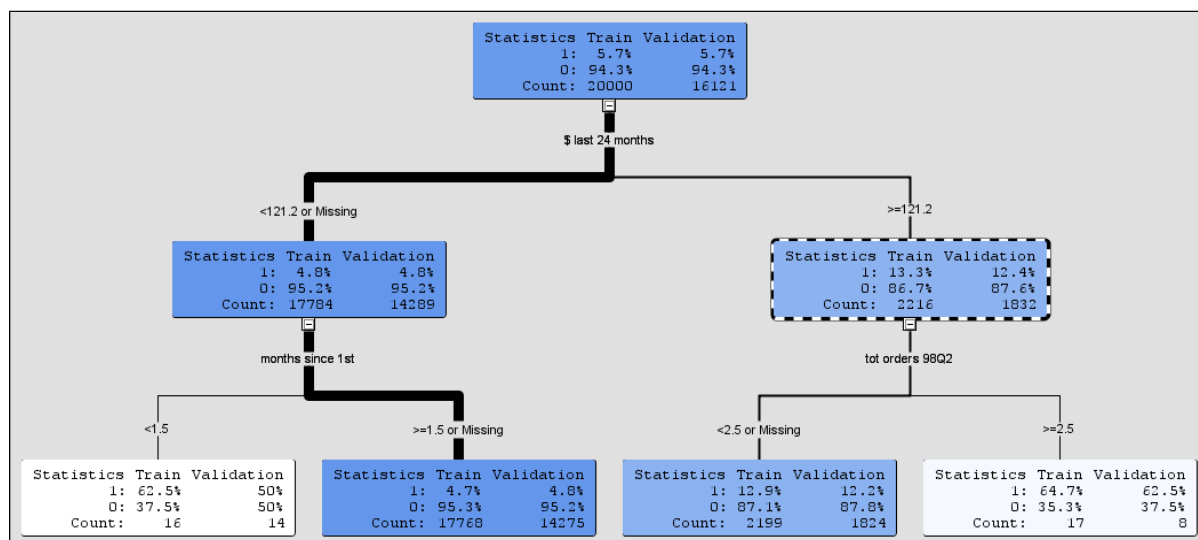
- Right-click the left branch where **DOLL24** <121.2.
- Select **Split Node** from the menu.
- Click **OK** in the Split Node dialog box.



The Tree View window is updated to show the additional split. The new split is based on the variable **TENURE** (months since first order), where the left branch contains the cases with months since first order less than 1.5, and the right branch contains the cases with months since first order greater than or equal to 1.5.

15. Split the right leaf node.

- Right-click the branch on the right, where **DOLL24** ≥ 121.2 .
- Select **Split Node** from the menu.
- Click **OK** in the Split Node dialog box.



The variable **TOTORDQ22** (total orders in second quarter 1998) is now used to form a split where the cases with the number of orders less than 2.5 or missing are assigned to the left branch and cases with the number of orders 2.5 or greater are assigned to the right branch.

It is important to remember that the logworth of each split, reported above, and the predicted **RESPOND** 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.

You can also change the splitting rule to accommodate your business needs. For example, the current splitting rule for number of orders is 2.5. For business reasons, you might want a split at 2 orders.

16. Modify the splitting rule by changing the splitting point.

- Select the node above the label **tot orders 98Q2**.
- Right-click the node and select **Split Node** from the menu.
- Click **Edit Rule**. The Interval Splitting Rule dialog box appears.
- Type **2** in the **New split point** field.

e. Click **Add Branch**. The Interval Split Rule dialog box shows three branches.

The dialog box is titled "TOTORDQ22 - Interval Split Rule". It has a "Target Variable:" field set to "RESPOND". Under "Assign missing values to", the radio button "A specific branch" is selected, with a dropdown menu showing "1". Below this, there are two unselected radio buttons: "A separate missing values branch" and "All branches". The "Branches" section contains a table with three rows:

Branch		Split Point
1	<	2
2	<	2.5
3	>=	2.5

Below the table, there is a "New split point:" text box, an "Add Branch" button, and a "Remove Branch" button. At the bottom, there are four buttons: "OK", "Cancel", "Apply", and "Reset".

17. Remove branch 2.

Select branch 2 by highlighting the row. Click **Remove Branch**.

The dialog box is the same as the previous one, but now it only has two branches in the table:

Branch		Split Point
1	<	2
2	>=	2

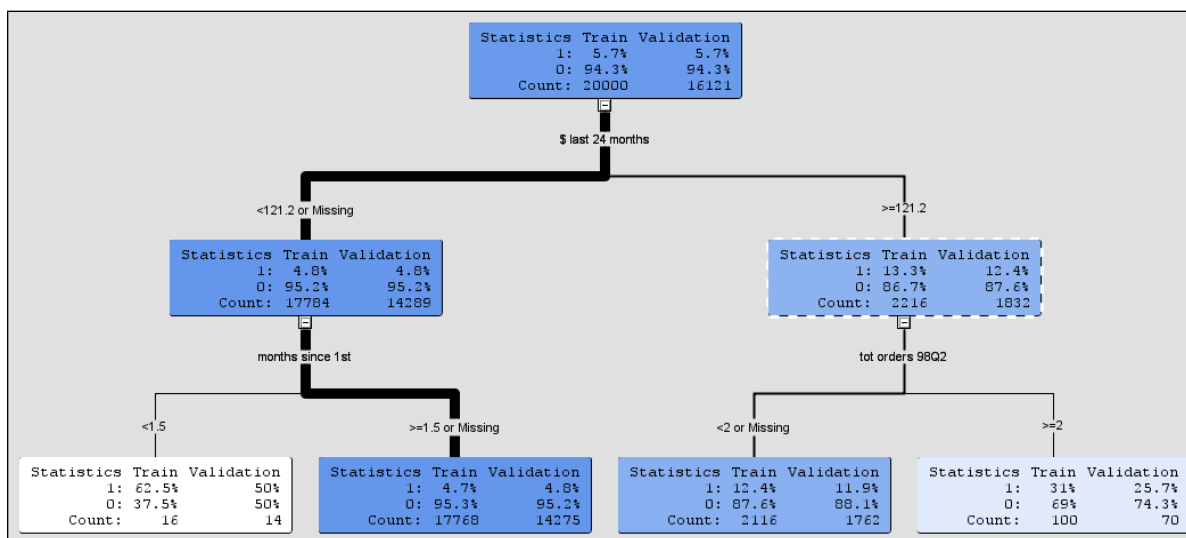
The "Remove Branch" button is now disabled. The "OK" button is highlighted with a dashed border.

The split point for the partition is moved to two orders.

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

18. Update the splitting rule.

Click **OK** both to close the Interval Split Rule dialog box and to update the Tree View window.



Overriding the statistically optimal split slightly changed the concentration of **RESPOND=1**.

As demonstrated here, because the process of building the decision tree model can be one of carefully considering and selecting each split sequentially, you can grow the tree automatically.

19. Grow the tree automatically.

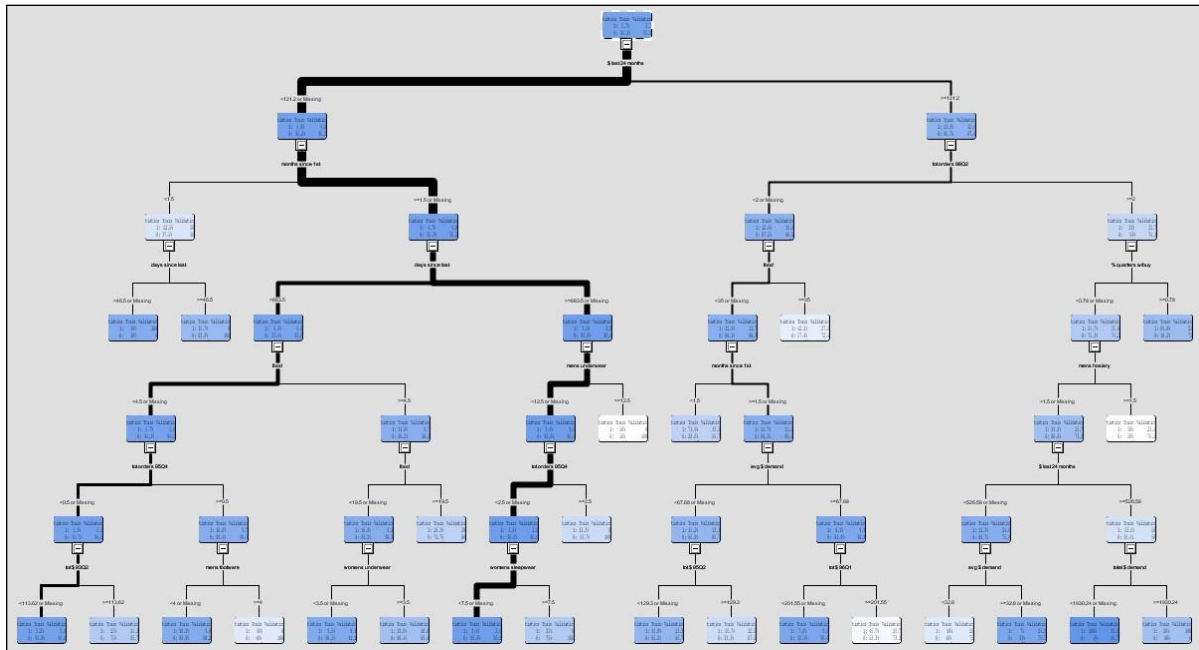
Select the root node of the tree. Right-click and select **Train Node** from the menu. The tree is grown until stopping rules prohibit further growth.

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

20. View the entire fitted tree.

Right-click in the tree and select **View Fit to page**.

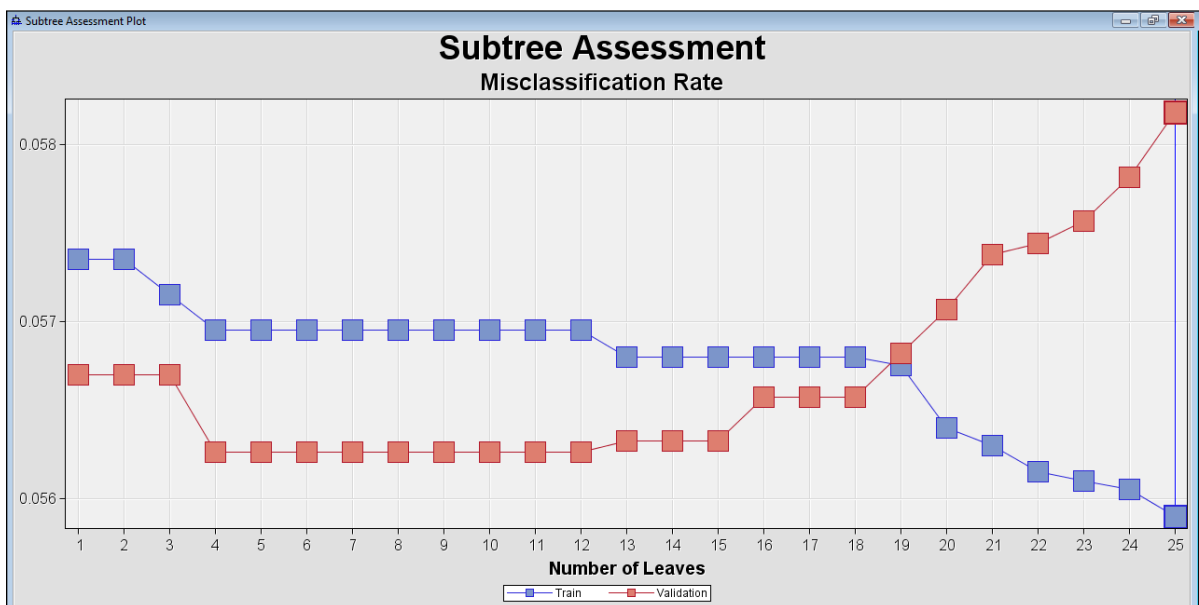
The tree view is scaled to provide the general shape of the maximal tree.



Plots and tables that are contained in the interactive tree tool provide a preliminary assessment of the maximal tree.

21. View the assessment plot.


Select **View**  **Subtree Assessment Plot**.



The plot shows that the maximal 25-leaf tree generates a lower misclassification rate than any of its simpler predecessors on the Training data. The best tree based on the validation data has 4 leaves.

Recall that the basis of the plot is the training data. Using the same sample of data to evaluate input variable usefulness *and* to assess model performance commonly leads to overfit models. To circumvent this problem, the assessment is done on the validation data.

22. Exit the generated tree.

Select **File**  **Exit**. A dialog box asks whether you want to save the tree that you created. Click **No** to prevent saving the maximal tree.

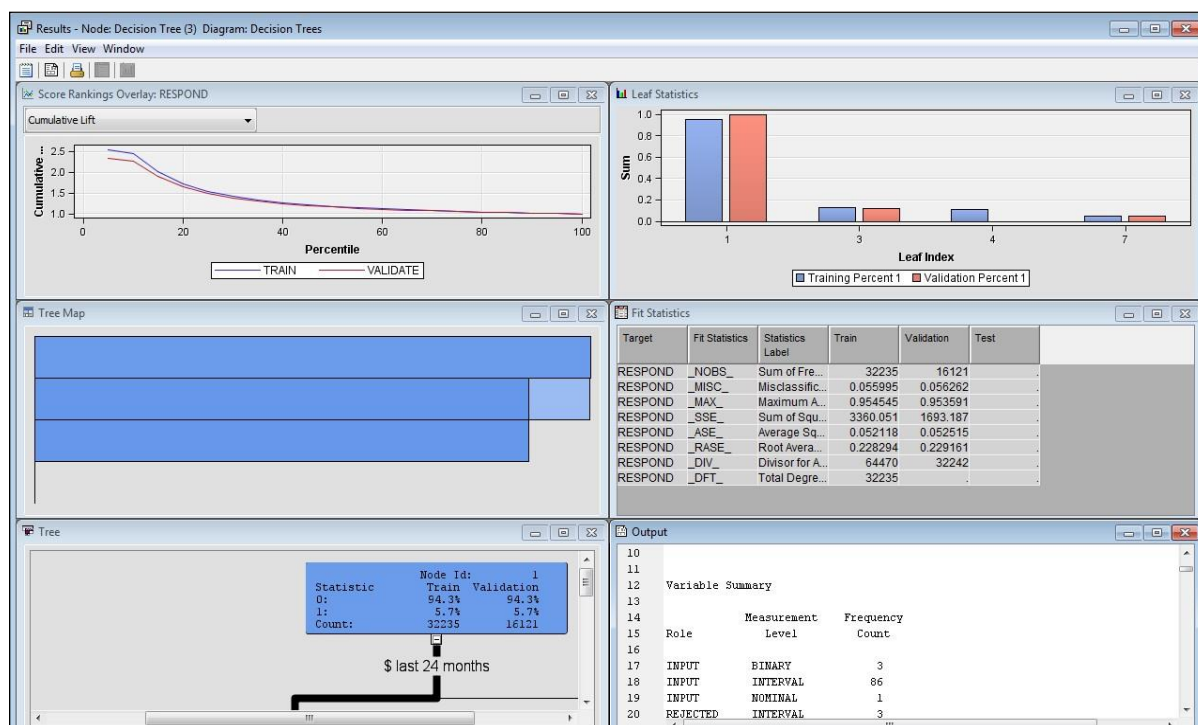
23. Re-create the tree using automated tools.

Right-click the **Decision Tree** node and click **Run**. Click **Yes**.

If you accidentally saved the Interactive Tree results, delete the current Decision Tree node, drag another Decision Tree node to the diagram, connect the Data Partition node to it, and rerun the node.

24. Inspect the results.

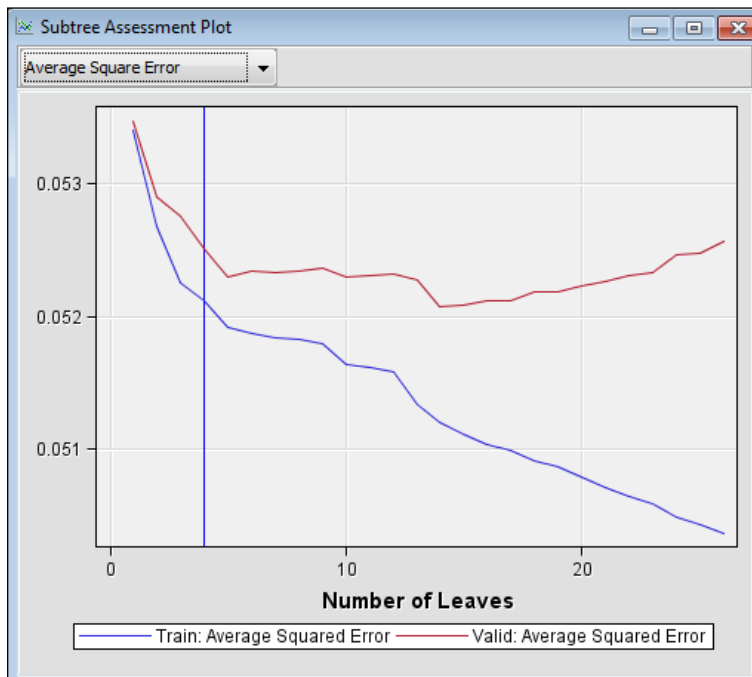
When the node has completed, select **Results**.



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. Notice that several of the diagnostics shown in the results of the Decision Tree node use the validation data as a basis.

25. View the average squared error over all the iterations on the training and validation data sets.

Select **View**  **Model**  **Subtree Assessment Plot.**

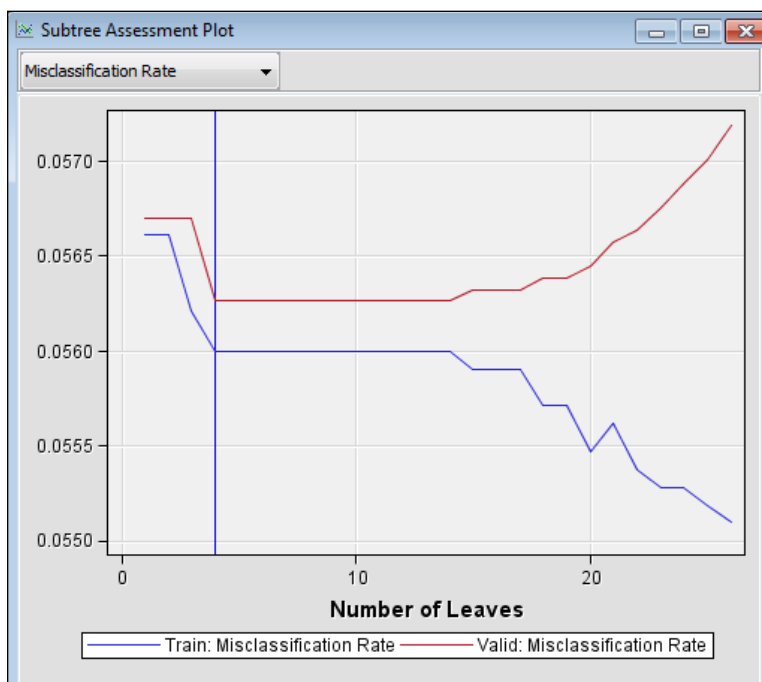


The plot shows the average square error corresponding to each tree in the sequence as the data is sequentially split. The plot shows that the performance on the training sample becomes monotonically better as the tree becomes more complex. However, the performance on the validation sample only improves up to a tree of, approximately, five leaves, and then diminishes as model complexity increases.

The validation performance shows evidence of model overfitting. Over the range of one to approximately nine leaves, the precision of the model generally 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.

26. Switch the assessment statistic to misclassification rate.

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



The validation performance under Misclassification Rate shows that the optimal tree appears to have four leaves.

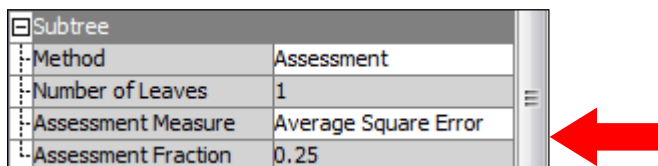
27. Rename the decision tree node.

Close the Decision Tree Results window. Right-click the **Decision Tree** node and select **Rename**. Name the node **Tree1**.

The next step generates the optimal tree using a different assessment measure. The relevant properties are listed under the Subtree heading.

28. Add another decision tree node to the diagram.

- Drag another Decision Tree node into the diagram and connect the Data Partition node to it.
- Rename the node **Tree2**. 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. Change the Assessment Measure property to **Average Square Error**.



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. In this case, it uses the decision criterion by default, which uses

misclassification rate when a binary target is used and no profit matrix information is supplied.

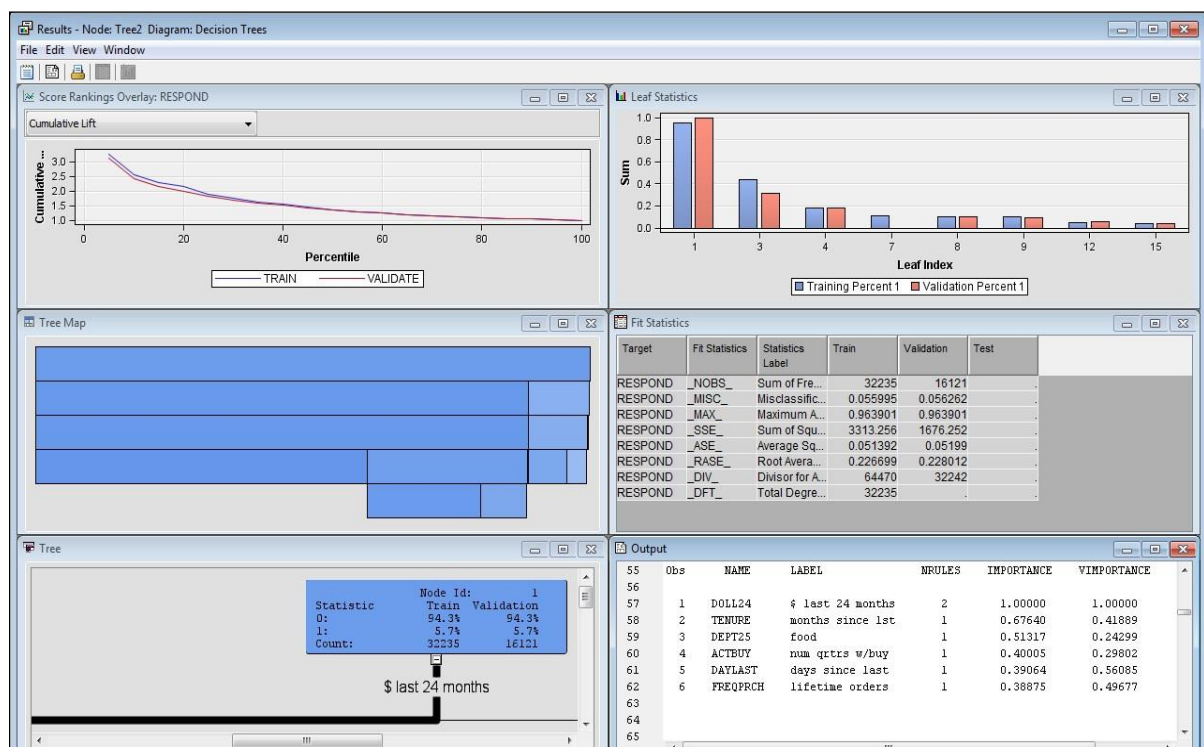
Alternative method options are Largest and N. The Largest option provides an autonomous way to generate the maximal tree. The N option generates a tree with N leaves. The maximal tree is upper-bound on N .

The Assessment Measure property specifies the optimality measure that is 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 default project settings, SAS Enterprise Miner chooses the tree with the lowest misclassification rate on the validation sample. **In this example, SAS Enterprise Miner uses the tree with the lowest average square error on the validation sample.**

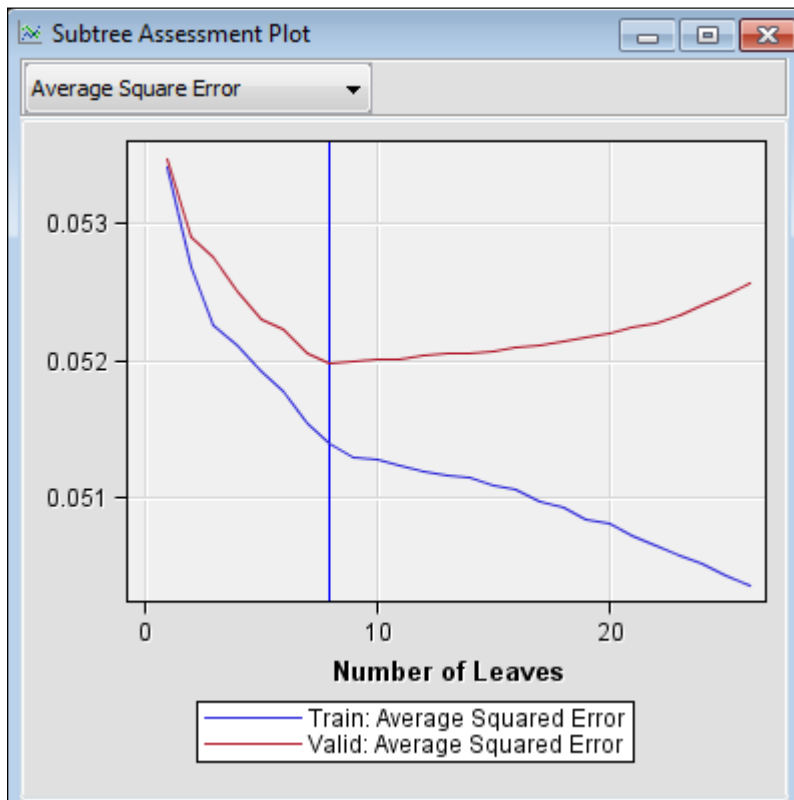
29. Run the second tree node.

Right-click the new **Tree2** node and click **Run**. Click **Yes** in the Confirmation window. After the Decision tree node runs, select **Results**.



30. View the subtree assessment plot.

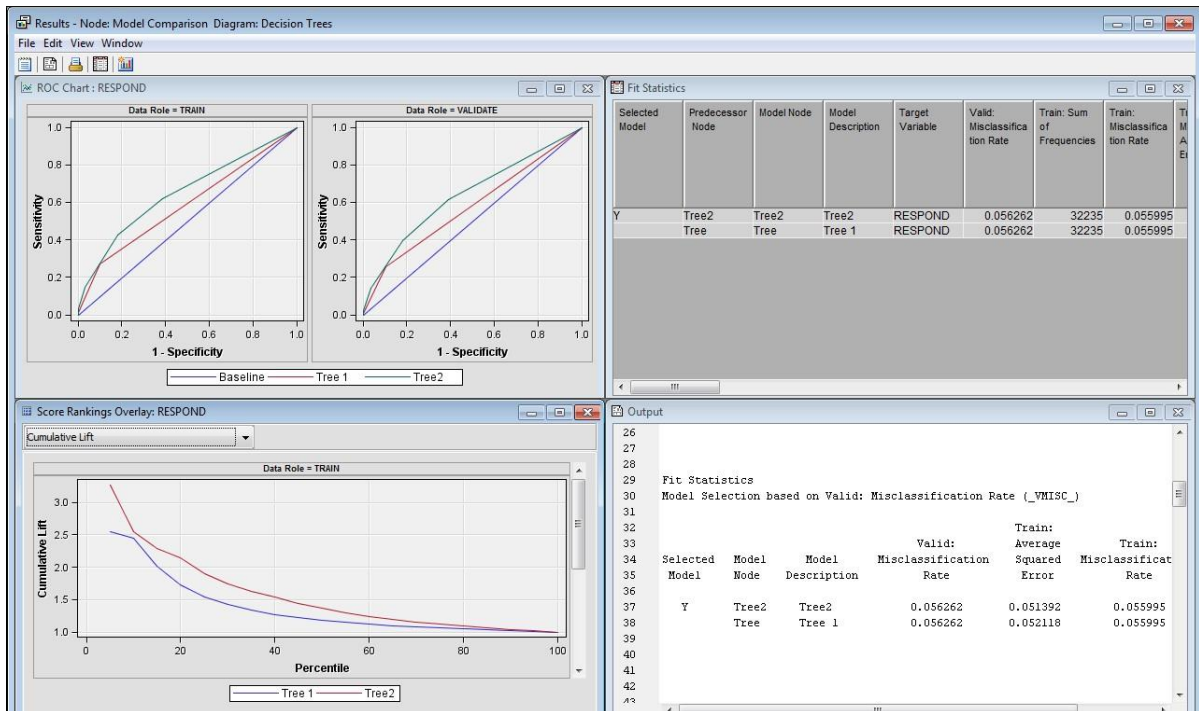
In the Results window, select **View**  **Model**  **Subtree Assessment Plot**.



The optimal tree based on average square error has eight leaves.

31. Compare the two decision tree models.

Close the Results window. Click the **Assess** tab and drag the **Model Comparison** node into the diagram workspace. Connect both **Decision Tree** nodes to the **Model Comparison** node. Run the Model Comparison node and view the results



The second tree had a higher lift and a higher ROC statistic along with a lower average squared error on the validation data set.

TASK: ORGANICS DATASET

1. Initial Data Exploration

A supermarket offers 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 more than 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 **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** is used for this analysis, and it should have a role of **Target**. Can **TargetAmt** be used as an input for a model that is used to predict **TargetBuy**? Why or why not?
 - 5) Finish the **ORGANICS** data source definition.
- c. Add the **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 65% of the data for training and 35% 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 **Misclassification** as the model assessment statistic.
 - 1) How many leaves are in the optimal tree?
 - 2) Which variables were used for the first split? What were the competing splits for this first split?
 - 3) Which variables were used for the second split for all branches from first split?
 - 4) Discuss the results and provide your insights
 - 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 **Misclassification** as the model assessment statistic.
 - 3) How many leaves are in the optimal tree?
 - h. Based on **Misclassification rate**, which of the decision tree models appears to be better?