



Salford Predictive Modeler[®]

Introducing CART[®]

This guide provides a brief introduction to CART[®].

© 2019 Minitab, LLC. All Rights Reserved.

Minitab®, SPM®, SPM Salford Predictive Modeler®, Salford Predictive Modeler®, Random Forests®, CART®, TreeNet®, MARS®, RuleLearner®, and the Minitab logo are registered trademarks of Minitab, LLC. in the United States and other countries. Additional trademarks of Minitab, LLC. can be found at www.minitab.com. All other marks referenced remain the property of their respective owners.

Introduction

Welcome to CART®, a robust decision-tree tool for data mining, predictive modeling, and data preprocessing. CART automatically searches for important patterns and relationships, uncovering hidden structure even in highly complex data. CART trees can be used to generate accurate and reliable predictive models for a broad range of applications from bioinformatics to risk management and new applications are being reported daily. The most common applications include churn prediction, credit scoring, drug discovery, fraud detection, manufacturing quality control, and wildlife research. Several hundred detailed applications studies are available from our website at <http://www.salford-systems.com>.

CART uses an intuitive, Windows-based interface, making it accessible to both technical and non-technical users. Underlying the "easy" interface, however, is a mature theoretical foundation that distinguishes CART from other methodologies and other decision trees.

Salford Systems' CART is the only decision-tree system based on the original CART code developed by world-renowned Stanford University and University of California at Berkeley statisticians Breiman, Friedman, Olshen and Stone. The core CART code has always remained proprietary and less than 20% of its functionality was described in the original CART monograph. Only Salford Systems has access to this code, which now includes enhancements co-developed by Salford Systems and CART's originators.

There is only one true CART and Salford Systems in collaboration with CART's creators is the only source for this remarkable technology.

Based on decades of machine learning and statistical research, CART provides reliable performance and accurate results. Its market-proven methodology is characterized by:

- ◆ A complete system of reliable data analysis
 - When the CART monograph was first published it revolutionized the emerging field of decision trees. An entire methodology was introduced for the first time that included multiple tree-growing methods, tree pruning, methods to deal with unbalanced target classes, adapting to the cost of learning and the cost of mistakes, self-testing strategies, and cross validation. For the scientifically minded, rigorous mathematical proofs were provided to show that the underlying algorithms were mathematically sound and could be relied upon to yield trustworthy results.
 - The CART monograph, published in 1984, is now justly regarded as a landmark work and one of the most important mathematical events of the last 30 years. It is one of the most-frequently cited works in machine learning and data mining.
- ◆ An effective tree-growing methodology
 - CART introduced several new methods for growing trees, including the Gini and the innovative Twoing method, among others. These methods have proven effective in uncovering productive trees and generating insights into data. To cover a broad variety of problems, CART also includes special provisions for handling ordered categorical data and the growing of probability trees.
- ◆ A powerful binary-split search approach
 - CART trees deliberately restrict themselves to two-way splits of the data, intentionally avoiding the multi-way splits common in other methods. These binary decision trees divide the data into small segments at a slower rate than multi-way splits and thus detect more structure before too few data are left for analysis. Decision trees that use multi-way splits fragment the data rapidly, making it difficult to detect patterns that are visible only across broader ranges of data values.
 -

- ◆ An effective pruning strategy
 - CART's developers determined definitively that no stopping rule could be relied on to discover the optimal tree. They introduced the notion of over-growing trees and then pruning back; this idea, fundamental to CART, ensures that important structure is not overlooked by stopping too soon. Other decision-tree techniques use problematic stopping rules that can miss important patterns.
- ◆ Automatic self-test procedures
 - When searching for patterns in databases it is essential to avoid the trap of "over fitting", that is, of finding patterns that apply only to the training data. CART's embedded test disciplines ensure that the patterns found will hold up when applied to new data. Further, the testing and selection of the optimal tree are an integral part of the CART algorithm. In other decision-tree techniques, testing is conducted only optionally and after the fact and tree selection is based entirely on training data computations.
 - CART accommodates many different types of real-world modeling problems by providing a unique combination of automated solutions.
- ◆ Cross Validation and Repeated Cross Validation
 - Cross validation, one of CART's self-testing methods, allows modelers to work with relatively small data sets or to maximize sample sizes for training. We mention it here because implementing cross validation for trees is extraordinarily challenging and easy to get wrong technically. With CART you get cross validation as implemented by the people who invented the technology and introduced the concept into machine learning. In the SPM® we allow you to rerun many replications of cross validation using different random number seeds automatically so that you can review the stability of results across the replications and extract summaries from an averaging of the results.
- ◆ Surrogate splitters intelligently handle missing values
 - CART handles missing values in the database by substituting "surrogate splitters", back-up rules that closely mimic the action of primary splitting rules. The surrogate splitter contains information that typically is similar to what would be found in the primary splitter. You can think of the surrogate splitter as an imputation that is customized to the node in the tree in which it is needed and that makes use of other relevant information in the data.
 - Other trees treat all records with missing values as if the records all had the same unknown value; with that approach all such "missings" are assigned to the same bin. In CART, each record is processed using data specific to that record, allowing records with different data patterns to be handled differently and resulting in a better characterization of the data.
 - CART also automatically analyzes whether missing-ness is in itself predictive and will optionally incorporate such findings into the optimal model.
- ◆ Adjustable misclassification penalties help avoid the most costly errors
 - CART includes "cost-sensitive" learning so that models developed by CART can incorporate the seriousness of any mistake. In a binary classification problem we often label the outcomes 0 and 1 and, by default, assume that all classification errors are equally costly. But what if misclassifying a 1 as a 0 (a false negative) is far worse than misclassifying a 0 as a 1 (a false positive)?
 - CART users can specify a higher "cost" for the more serious mistakes, causing the software to steer the tree away from that type of error. That is, in response to the cost information CART will actually grow a different tree. The greater the cost of a specific kind of mistake the more CART will adjust the tree to avoid the high cost mistakes. Further, when CART cannot guarantee a correct classification, it will try to ensure that the errors it does make are less costly. If credit risks were classified as low, moderate, or high, for example, it would be more costly to misclassify a high-risk borrower as low-risk than as moderate-risk. Traditional data mining tools and many

decision trees cannot distinguish between these types of misclassification errors in their model construction processes.

- ◆ Alternative splitting criteria make progress when other criteria fail
 - CART includes seven single-variable splitting criteria, Gini, Symmetric Gini, Twoing, Ordered Twoing, Entropy and Class Probability for classification trees, and Least Squares and Least Absolute Deviation for regression trees. In addition, we offer one multi-variable or oblique splitting criterion, the Linear Combinations or LC method. CART includes some important extensions to the classic LC method.
 - The default Gini method frequently performs best, but by no means is Gini the only method to consider in your analysis. In some circumstances the Twoing method will generate more intuitive trees. To help you find the best method CART will optionally test all its methods automatically and summarize the results in tables and charts (AUTOMATE RULES).

CART Tutorial

This chapter provides a hands-on tutorial to introduce you to the CART graphical user interface—menus, commands, and dialogs. In this first tutorial, you will learn how to set up a simple CART analysis, how to navigate the dynamic tree displays, and how to save your work.

A word on our examples: CART can be applied to data from any subject. We have come across CART models in agriculture, banking, genetics, marketing, security, and zoology, among many others, and the citations to CART number in the thousands. Because analysts prefer to work with examples from their own fields we have included a few alternative case studies.

This chapter deals with a simple YES/NO outcome drawn from the field of credit risk. We recommend that you try to follow this first example as it primarily uses concepts with which most readers will be familiar.

Our first tutorial file, **GOODBAD.CSV**, contains data on 664 borrowers, 461 of whom repaid a loan satisfactorily and 203 who defaulted. Clearly, the defaulters have been oversampled; few lenders could afford to have a loss rate as high as 31%. While the data have their origin in the real world, the specific records included here have been fictionalized. Nevertheless, we have retained the broad statistical relationships between the variables to yield a realistic study.

The variables available on the file include:

- ◆ **TARGET** 0=good, 1=bad (defaulted)
- ◆ **AGE** Age of borrower in years
- ◆ **CREDIT_LIMIT** Loan amount
- ◆ **EDUCATION\$** Category of level of schooling attained
- ◆ **GENDER** Male or Female
- ◆ **HH_SIZE** Number of family members
- ◆ **INCOME** Per month
- ◆ **MARITAL\$** Marital status
- ◆ **N_INQUIRIES** Credit bureau measure
- ◆ **NUMCARDS** Number of credit cards
- ◆ **OCCUP_BLANK** No occupation listed
- ◆ **OWNRENT\$** Home ownership status
- ◆ **POSTBIN** Postal code
- ◆ **TIME_EMPLOYED** Years work experience

The goal of our analysis is to uncover the factors that are predictive of default. In such studies the predictors such as AGE and INCOME must pertain to the time at which the borrower was granted the loan and the TARGET records whether or not the loan was satisfactorily repaid subsequently. A successful default model could be used to create a credit score and help the lender differentiate between good and bad risks in future loan applicants.

Opening a File

To open the **GOODBAD.CSV** file:

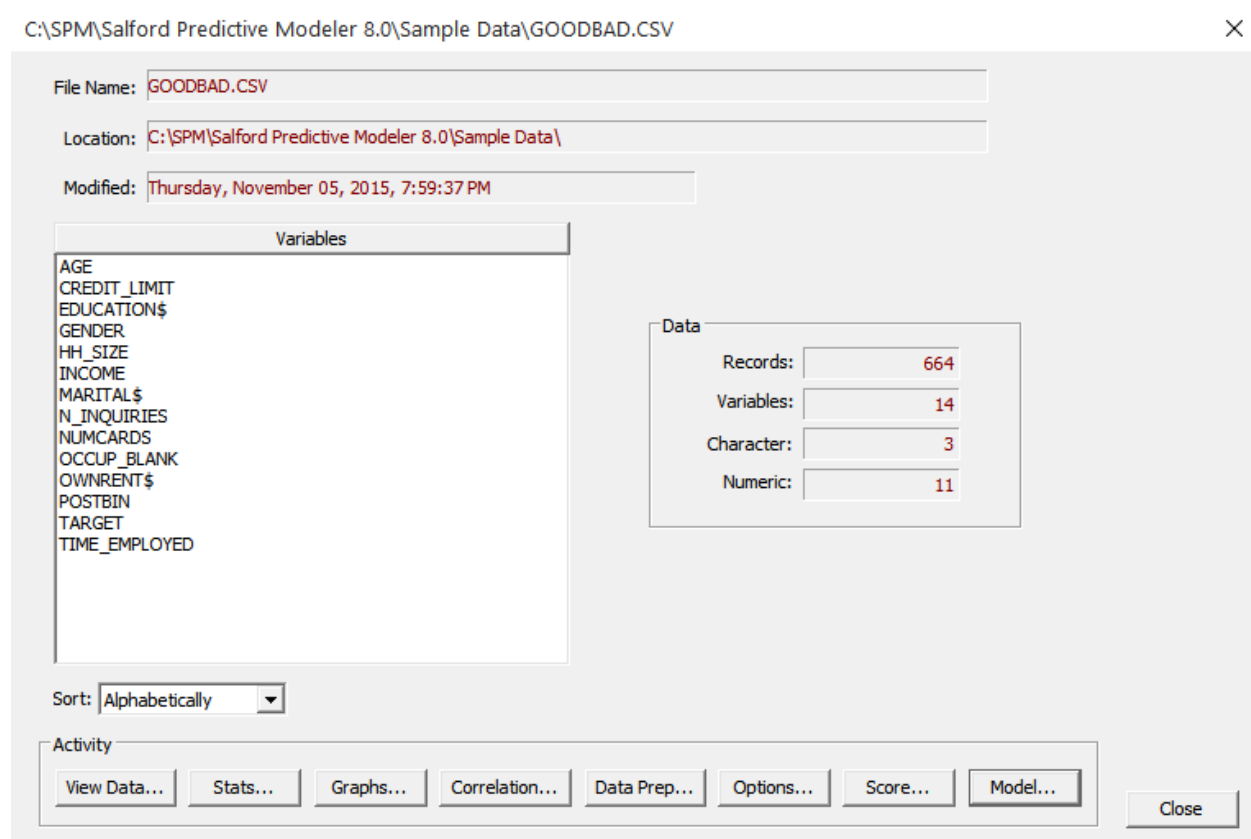
Select **Open>Data File...** from the **File** menu (or click on the  toolbar icon).

- ✓ You can reset default input and output directories; select **Options...** from the **Edit** menu and select the **Directories** tab.

In the **Open Data File** dialog, navigate to the SPM **Sample Data** directory and then select the **GOODBAD.CSV** file and click on the **[Open]** button or double-click the file name.

- ✓ Make sure that **Delimited Text (*.csv, *.dat, *.txt)** is selected in the **Files of Type:** box to see files ending with the .CSV extension.

You may see a slightly different list of files in your directory. When you open GOODBAD, the **Activity** dialog opens automatically, as shown next.



C:\SPM\Salford Predictive Modeler 8.0\Sample Data\GOODBAD.CSV

File Name:

Location:

Modified:

Variables	
AGE	
CREDIT_LIMIT	
EDUCATION\$	
GENDER	
HH_SIZE	
INCOME	
MARITAL\$	
N_INQUIRIES	
NUMCARDS	
OCCUP_BLANK	
OWNRENT\$	
POSTBIN	
TARGET	
TIME_EMPLOYED	

Data	
Records:	664
Variables:	14
Character:	3
Numeric:	11

Sort:

Activity



You can see from here that the file contains 664 records and 14 variables, three of which are character, or text, columns. The variable names are also listed and you can change the order they are sorted in from **Alphabetical** to **File Order** using the **Sort:** drop-down control.

Start by clicking on the **[View Data...]** button to bring up a spreadsheet display of the file contents. Note that some of the cells are blank or contain only a “.”; these are missing values. The window offers a view-only display; you can scroll through the data but you cannot edit it from here.

	TARGET	AGE	CREDIT_LIMIT	EDUCATIONS	GENDER	HH_SIZE	INCOME	MARITALS	N_INQUIRIES
1	0	48	19387	HS	6	6	2399	Single	3
2	0	37	0	HS	6	1	9618	Married	1
3	0	33	0	College	6	6	0	Married	1
4	0	.	0	College	.	.	2700	Married	1
5	0	24	12000	College	6	4	2665	Single	4
6	0	.	11516	College	-4	2	4358	Married	0
7	0	30	19000	College	-4	.	3922	Married	0
8	0	22	14000	College	6	3	2900	Married	7
9	0	29	7000	College	-4	1	3500	Married	5
10	0	31	0	College	-4	3	2500	Married	10
11	0	31	0	College	-4	4	1976	Married	0
12	0	34	17929	College	-4	3	4500	Married	3
13	0	.	100000	College	-4	1	15000	Married	6
14	0	31	12000	College	-4	3	3500	Married	0
15	0	.	8000	College	-4	4	4000	Married	0
16	0	31	5120	College	-4	.	2700	Single	0
17	0	.	25670	HS	-4	2	5915	Married	1
18	0	.	25700	College	-4	.	4000	Single	6
19	0	33	15005	HS	6	1	3000	Married	1
20	0	27	100000	College	6	2	2500	Married	1
21	0	31	75000	College	-4	3	7794	Married	13

Close the **View Data** window, this will bring you back to the **Classic Output** window.

At this point we are ready to build our first model. Go ahead and activate the **Model Setup** window by doing one of the following actions:

- ◆ Click the **Model>Construct Model...** menu item.
- ◆ Alternatively, press the  button in the toolbar.
- ◆ Alternatively, press the  button in the toolbar and then press the **[Model...]** button in the **Activity** window.

Setting up a CART Run

The **Model Setup dialog** tabs are the primary controls for conducting CART analysis. Fortunately you only need to visit the first **Model** tab to get started so we now focus on this one tab.

- ✓ Tab headings are displayed in **RED** when the tab requires information from you before a model can be built.
- ✓ In our example, the tab is red because we have not yet selected a **TARGET** variable. Without this information CART does not know which of the 14 variables we are trying to analyze or predict. This is the only **required** step in setting up a model. Everything else is optional.

Selecting Target and Predictor Variables

In the **Model** tab:

- ◆ Make sure that the **Analysis Engine** is set to **CART Decision Tree**.
- ◆ Make sure that the **Target Type** is set to **Classification/Logistic Binary** (default setting).
- ◆ Select the binary variable named TARGET (coded 0/1) as the target of the modeling run by placing a checkmark in the **Target** column.
- ◆ Select which variables are to be used as predictors. CART is a capable automatic variable selector so you do not have to do any selection at all, but in many circumstances you will want to exclude certain variables from the model. In our first run select all the variables **except** POSTBIN. The easiest way to do this: click on the **Predictor** column heading to highlight the column, check the **Select Predictors** box underneath the column, and then uncheck POSTBIN.
- ✓ If you do not explicitly select the predictors CART is allowed to use, then CART will screen all variables for potential inclusion in its model.
- ✓ Even if all the variables available are reasonable candidates for model inclusion it can still be useful to focus on a subset for exploratory analyses.
- ✓ In this dataset TARGET is a categorical variable and should be checked as such (note, that if you have not selected Classification as the Target Type then you will be unable to check this Categorical box). The other categorical variables, such as MARITAL\$, have been automatically checked as categorical predictors because they are character (text) variables.

Model Setup

Limits	Costs	Priors	Penalty	Lags	Automate
Model	Categorical	Force Split	Constraints	Testing	Select Cases
Best Tree					
Method					

Variable Selection

Variable Name	Target	Predictor	Categorical	Weight	Aux.
MARITAL\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
N_INQUIRIES	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
NUMCARDS	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
OCCUP_BLANK	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
OWNRENT\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
POSTBIN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TARGET	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TIME_EMPLOYED	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Sort:

Filter: All/Selected Character Numeric

Select Predictors Select Cat. Select Aux.

Target Type: Classification/Logistic Binary Regression Unsupervised

Set Focus Class...

Target Variable: TARGET

Weight Variable:

Number of Predictors: 12

Automatic Best Predictor Discovery: Off Discover only Discover and run

Maximum variables for each class: 8

After Building a Model: Save Grove...

Number of Predictors in Model: 12

Analysis Engine: CART Decision Tree

Cancel Continue Start

We are now ready to grow a first tree.

To begin the CART analysis, click the **[Start]** button. While the model is being built a progress report will keep you informed about the actions being taken and some timing information (time elapsed, time remaining). Our example will run so fast you may not have a chance to notice everything on the progress indicator.

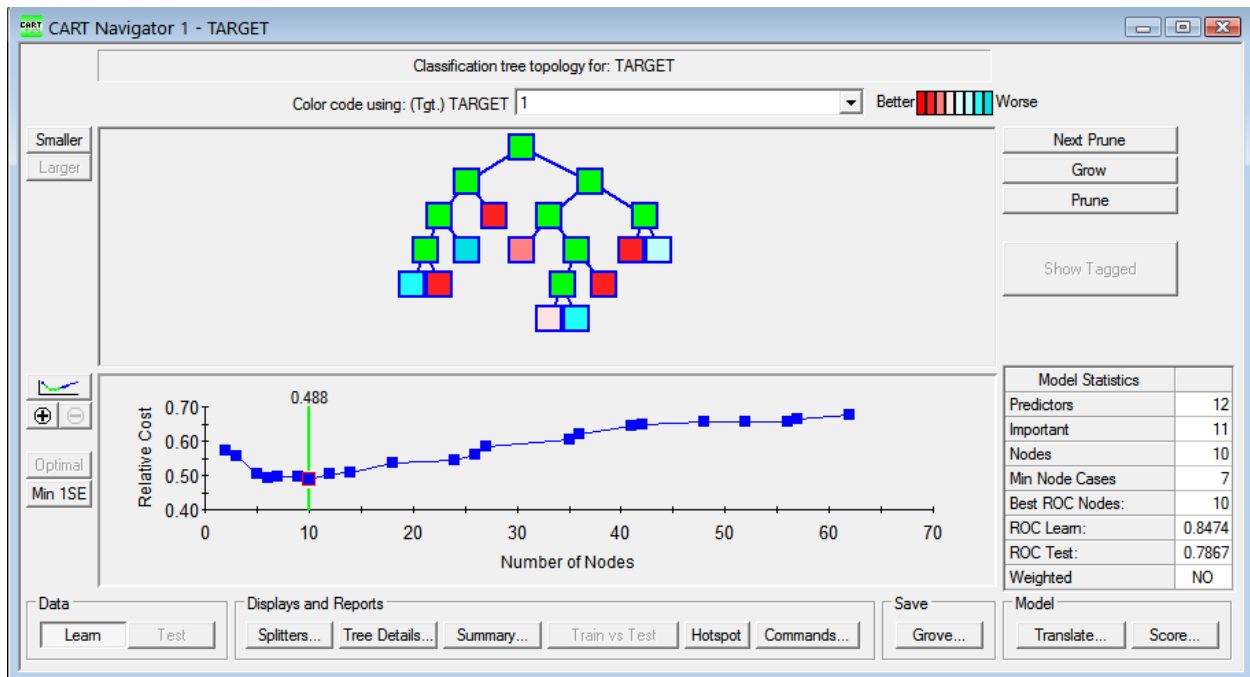
Once the analysis is complete, a new window, the **CART Navigator**, is opened. The navigator is the key to almost all CART output, reports and diagnostics, so it will function as a model summary and guide to everything you may want to know about the results. Experts may also redirect the classic text output and some other reports elsewhere. These items are discussed later in this manual.

Working with CART Navigator

The navigator packages everything you need to know about the CART tree. You can save the navigator as a grove file, email it to others, or just use it temporarily during a single CART session. The navigator will offer you many views of the model and its findings, will allow you to score new data, and can generate formatted text reports, tables, charts, and comparative performance summaries of competing models. The rest of this chapter is devoted to discovering the charms of the navigator.

The initial navigator display is just a simple overview of the shape of the tree or its topology in the top panel, and a predictive performance curve in the bottom panel. The tree topology, displayed in the top panel of the **Navigator** window, provides an immediate snapshot of the tree's size and depth. Here we have a tree with 10 terminal nodes (nodes at the bottom of the tree).

The color-coding helps us locate interesting terminal nodes. Bright red nodes isolate defaulters (Target class 1) and deep blue nodes are heavily populated with good borrowers. Other colors indicate more mixed results.



The tree displayed automatically is of the size determined by CART to be the most accurate classifier obtained. Other tree sizes are also available for display. In this example we can review trees with as few as two nodes or as many as 62 nodes.

The performance of the different-sized trees is displayed in the lower panel of the navigator. The curve is an error profile and traces the relationship between classification errors and tree size. We call it a **Relative Cost** curve (or if you prefer, a **Relative Error** curve) because it is always scaled to lie between 0 and 1. Zero (0) means no error or a perfect fit, and 1 represents the performance of random guessing. The best that we can do for the current tree is indicated by the green bar marking the low point on the error profile, where we hit a relative error of .488. If we settle for either too small or too large a tree we will not do as well as we could with the 10-node tree. Here we see the characteristic **U-shaped** curve with a partially flattened bottom.

- ✓ At this stage all you need to keep in mind is that we are looking for trees with low values of relative error.
- ✓ A tree with a relative error of 0 or near 0 is usually too good to be true. In almost all cases this results from including an inappropriate predictor in the model.
- ✓ It is possible to have a relative error greater than 1. This happens when the model is actually worse than random guessing.

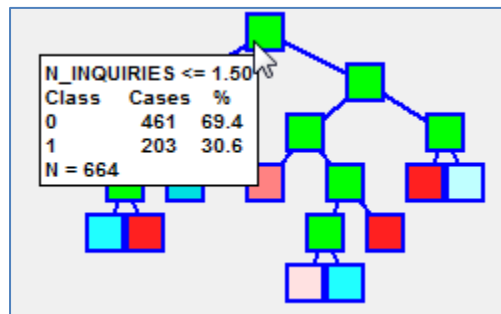
Returning to the navigator we see some core **model statistics** in the bottom right section. The report shows that we conducted the analysis with 12 predictors, of which 11 were found to have some value. The tree being displayed now has 10 terminal nodes and the smallest of these nodes contains seven records.

Just below the main model statistics are **ROC** measures. If you are not familiar with the ROC we include some introductory material on this important metric in the generic portion of this manual. For right now, all you need to know is that the ROC can range between 0 and 1 with higher values indicating better performance. Our model shows excellent performance with a test value of the ROC of .7867.

- ✓ Suppose we were to take a single good borrower and a single defaulter at random from a data set. Our ROC score tells us that in 78.67% of cases the defaulter would have a larger predicted model score than the good borrower.
- ✓ On the other hand, if you assigned scores at random, you would be right on average for 50% of all cases. Therefore, a good model needs to deliver substantially better than an ROC of .50. In real world credit risk scoring, an ROC of .70 would be considered respectable.
- ✓ The predictive performance of a model depends on many factors, including the nature and quality of the data and the inherent predictability of the data under study. You cannot expect every subject matter to support highly accurate models.

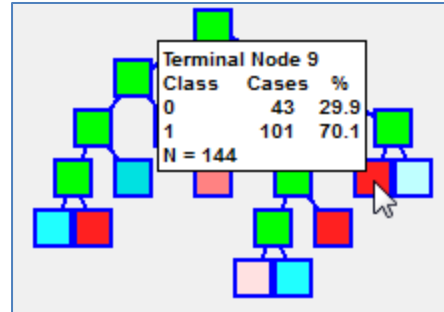
The **color-coding** of the terminal nodes is controlled from the pull down control at the top of the navigator. For 0/1 target variables the default coloring uses red to indicate a high concentration of 1s. You can change that if you prefer to have red represent another class instead, and you can also turn off special color coding, leaving all the terminal nodes red.

CART offers many ways to view the tree details and interior. We will start by hovering the mouse over a node. Beginning with the root node at the top of the tree, we note that we started with 461 GOODs (0s) and 203 BADs (1s), for a bad rate of 30.6.



- ✓ You can change the detail revealed when you hover your mouse over navigator nodes. Right-mouse-click in the “**gray**” area of the navigator window to bring up the patterns available, then left-mouse-click on your preferred display. You can also use **View>Node Display** menu to control mouse hover displays.

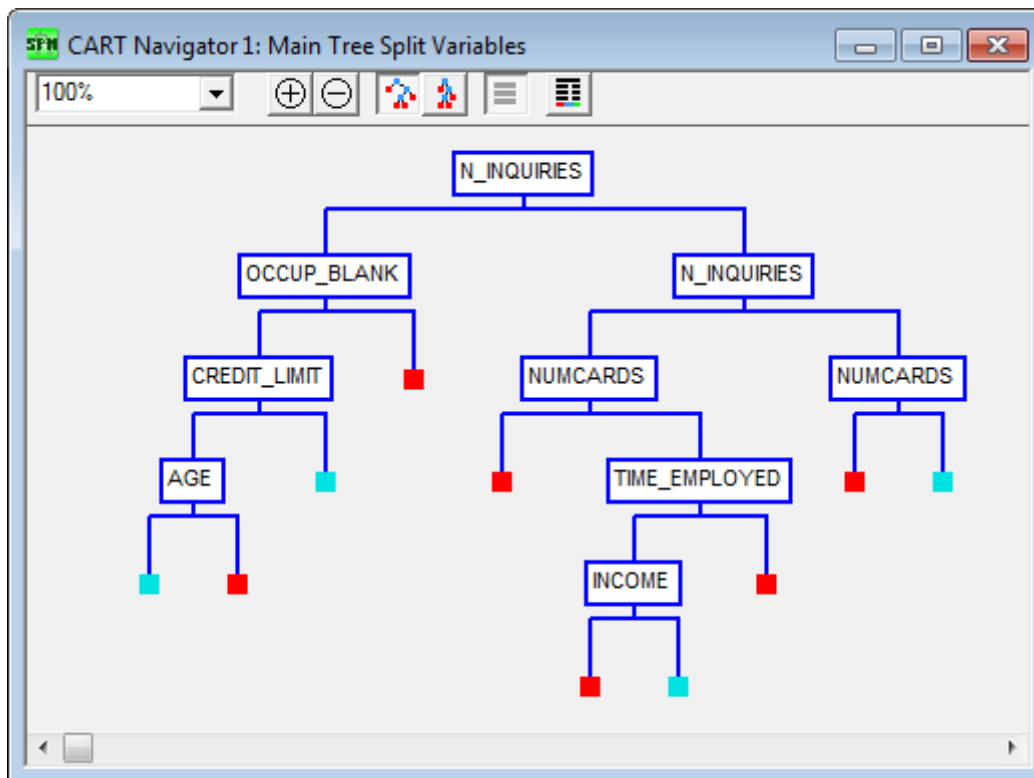
Now hover over the bright red node near the bottom right of the tree. This is terminal node 9, which has a bad rate of 70.1%, substantially higher than the baseline rate of 30.6% in the root. Visiting the other bright red nodes reveals similarly concentrated groups of defaulters.



Having established that our tree appears to be a promising model we now want to drill deeper into the results.

Viewing the Main Splitters

There is a convenient way to get a bird's eye view of the model is to reveal only the variables used in each node. At the bottom left of the navigator press the **[Splitters...]** button to see:



The color coding here is a simplified one: red means “above average” risk and blue means “below average risk.” Because the CART tree splitters always send low values of a splitter to the left and high values to the right, reading this display is easy. Going down the right side of the tree we see that if a person has a large number of inquiries but few credit cards they are quite high risk. Presumably this means that the person has probably attempted to obtain additional cards in the recent past but has failed. Looking down the left-hand side of the tree we see that persons who have a low number of inquiries but did not report an occupation are also high risk.

- ✓ Remember that these data are fictionalized and so should not be thought of as a completely faithful representation of real world credit risk. Some surprises are inevitable in this example.

We find the splitters view of the tree helpful in giving us a quick overview of the main drivers in the tree. We see the variables used at the top of the tree and the direction of their effect. At the bottom left we see that being older is a default risk factor and at the bottom middle we see that a lower income is also a risk factor. These are just quick impressions that help us acquire a feel for the message of the tree.

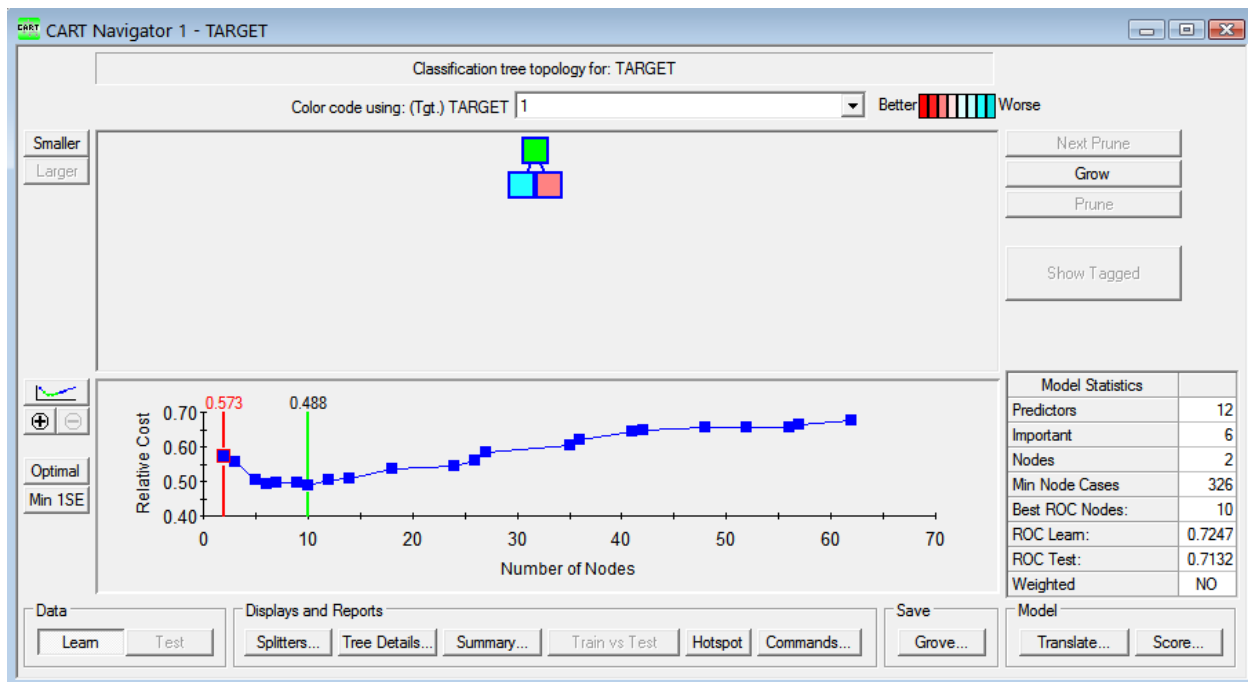
- ✓ The splitters view is an excellent way to quickly detect significant data errors. If you see a pattern of outcomes that is very different from what is expected or even possible you have identified a potential data flaw that needs to be investigated.

Exploring Trees of Different Sizes

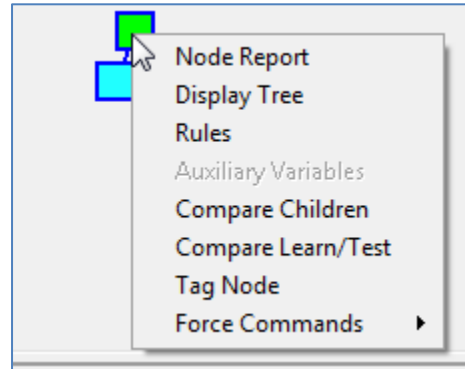
When a CART run completes, it displays the CART “optimal” tree: typically the tree with the smallest misclassification rate (or equivalently the highest classification accuracy). There are reasons to want to look at trees of different sizes, however:

- ◆ The relative error profile is often flat near its minimum. This means that smaller trees exist that are almost as accurate as the best tree found.
- ◆ Classification accuracy is not the only sensible criterion to use to select a model. Many data mining specialists prefer to use the area under the ROC curve as their model selection criterion.
- ◆ For decision making purposes you may be interested only in the top-performing nodes of the tree. If so, the accuracy and reliability of these nodes are all that matter and the overall performance of the tree is not relevant.
- ◆ Judgment can play an important role in the final tree selection.

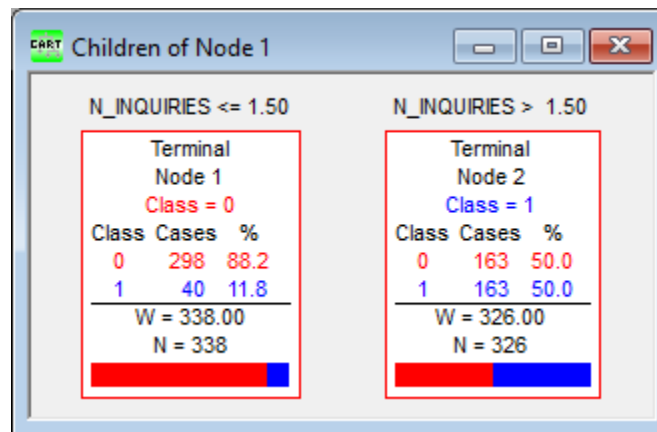
The navigator makes it very easy to view, display, and obtain reports for every size of tree found by CART in its tree-building process. Select the **Navigator** window and then use your **left** and **right arrow** keys to display different-sized trees in the **navigator topology** display. Begin by moving all the way to the left to reach the two-node tree:



Technically we could go one step further to arrive at the one-node tree (the null tree), but we make the two-node tree the smallest we will display. This tree makes use of only one predictor and is actually quite predictive, with a relative error rate of .573 and a test sample ROC value of .7132. This is unusually good for a single predictor and is far from typical. To take a closer look, move your mouse over the root and **right-click** to reveal this menu:



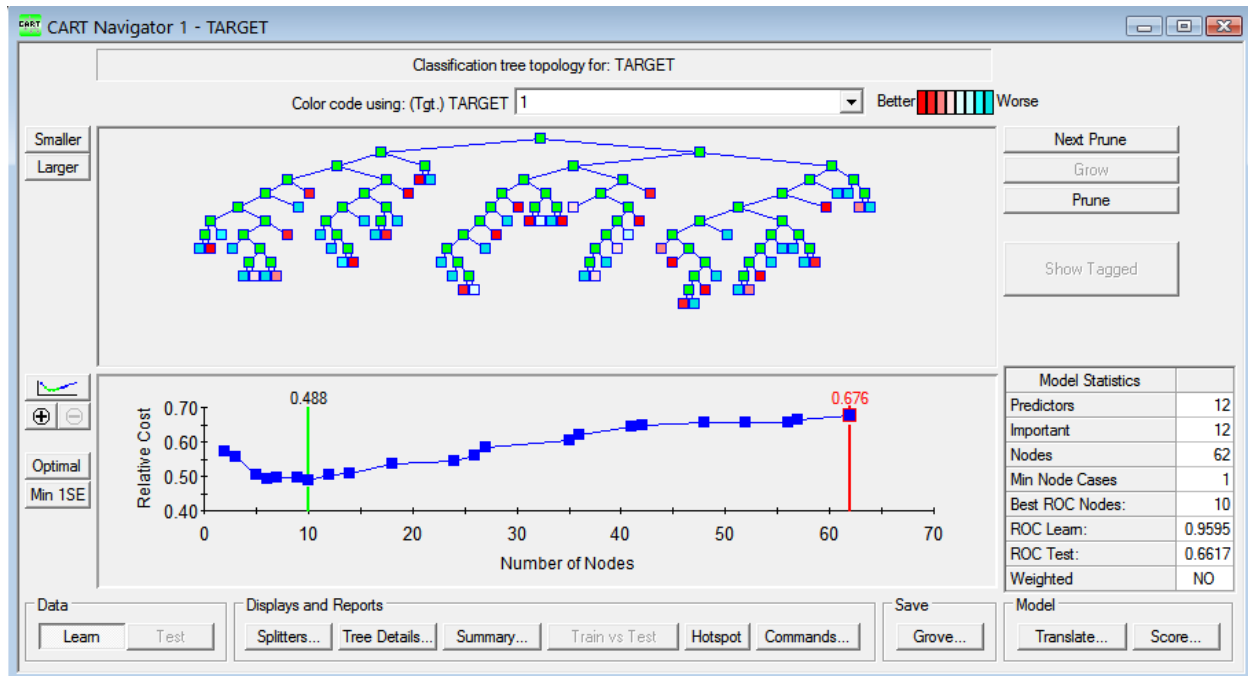
Select **Compare Children** to get the following display:



We see that having had more than one recent inquiry about a borrower at the credit bureau is a powerful indicator of default risk. Recall that the default rate in these data is 30.6% overall, whereas it is only 11.8% among those with one or no recent inquiries and 50% for those with two or more recent inquiries. (You can customize the colors and details shown in this window using the **View > Node Detail...** menu discussed later.)

- ✓ CART trees are grown by a procedure called “binary recursive partitioning.” The **binary** indicates that when a node is split it is divided into two and only two child nodes. This is a distinctive characteristic of the CART tree and is one source of the power of the CART technology.
- ✓ CART easily creates the equivalent of multi-way splits by using a variable more than once. We show an example below.

Close the “**Children of Node 1**” window and use the right-arrow key to move all the way to the other extreme: the largest tree grown in this run. From the relative error profile and the model statistics you can see that this tree has 62 nodes, its relative error is .676, and the test ROC is .6617.

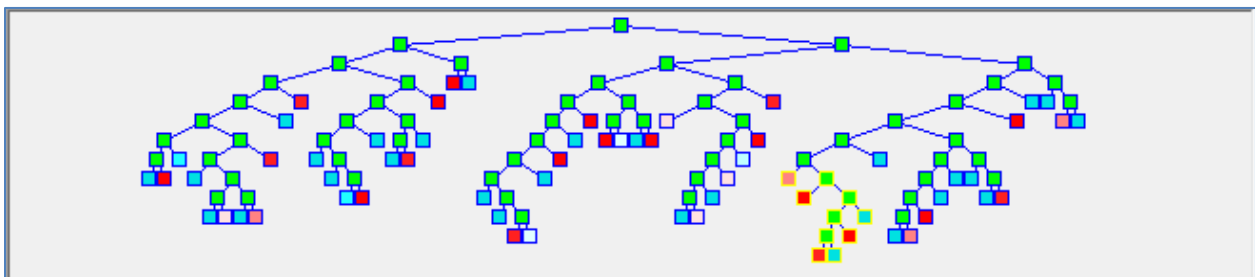


This largest tree is quite a bit worse than the simple two-node tree, indicating that the large tree is seriously “overfit.” While the largest tree is almost always overfit it is not necessarily worse than the smallest tree. In some cases the largest tree is also quite accurate, though in this example it is not.

The largest tree is actually the starting point for CART analysis. CART first splits the root node, then splits the resulting children, then splits the grandchildren, and so on. The CART tree does not stop until it literally runs out of data. This is in contrast to other decision trees that use a “stopping rule.”

- ✓ The CART approach to decision tree construction is based on the foundation that it is impossible to know for sure when to stop growing a decision tree. (You can prove this mathematically.) Therefore, CART does not stop, but rather grows and grows and grows.
- ✓ CART uses extraordinarily fast proprietary algorithms so it does not take much time to grow the initial largest tree.

Once we have the largest tree constructed we begin **pruning**. (This is done for you automatically.) The pruning process trims the tree by removing the splits and branches that are least useful. A pruning step often removes just one split but sometimes several splits are removed together. (The mathematical details are provided in the original CART monograph.) To see which nodes are removed in the next pruning step, press the **[Next Prune]** button at the upper right side of the **Navigator** window. The nodes to be pruned next will be highlighted in yellow.




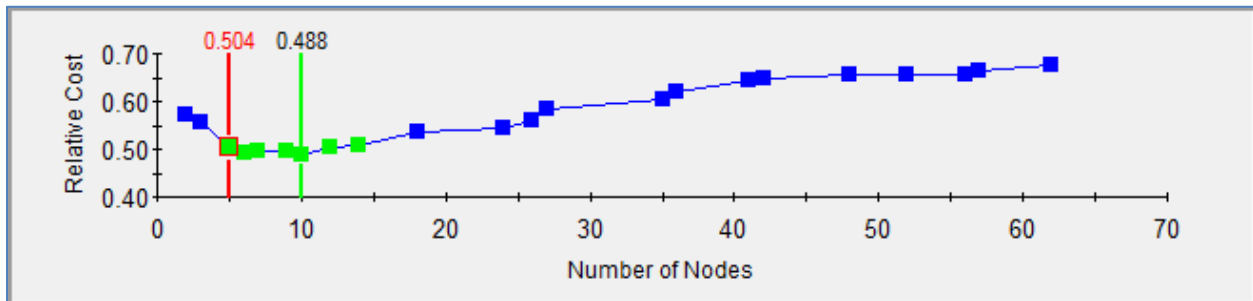
Use the **left arrow** key to return to the CART optimal tree marked with the green bar.

- ✓ The **Home** key is a short cut to return to the CART optimal tree in the navigator.

Here we can clearly see which node would be pruned next if we wanted to select a smaller tree. The reason CART would prune this particular node next is that by doing so CART would retain as much accuracy as possible. Now press the **[Next Prune]** button again to turn off the node highlighting.

Look again at the relative error profile and note the flat region near the 10-node mark.

It is natural to suspect that one of these smaller trees is practically just as good as the optimal tree. If you click on the  on the left side of the navigator you will see a portion of the relative error profile turn green.



This tells us exactly which sizes of trees exhibit an accuracy performance that is statistically indistinguishable from the optimal tree. The CART authors suggested that we use a “1 standard error” or 1SE rule to identify these trees and in the display we have moved to the smallest of these trees.

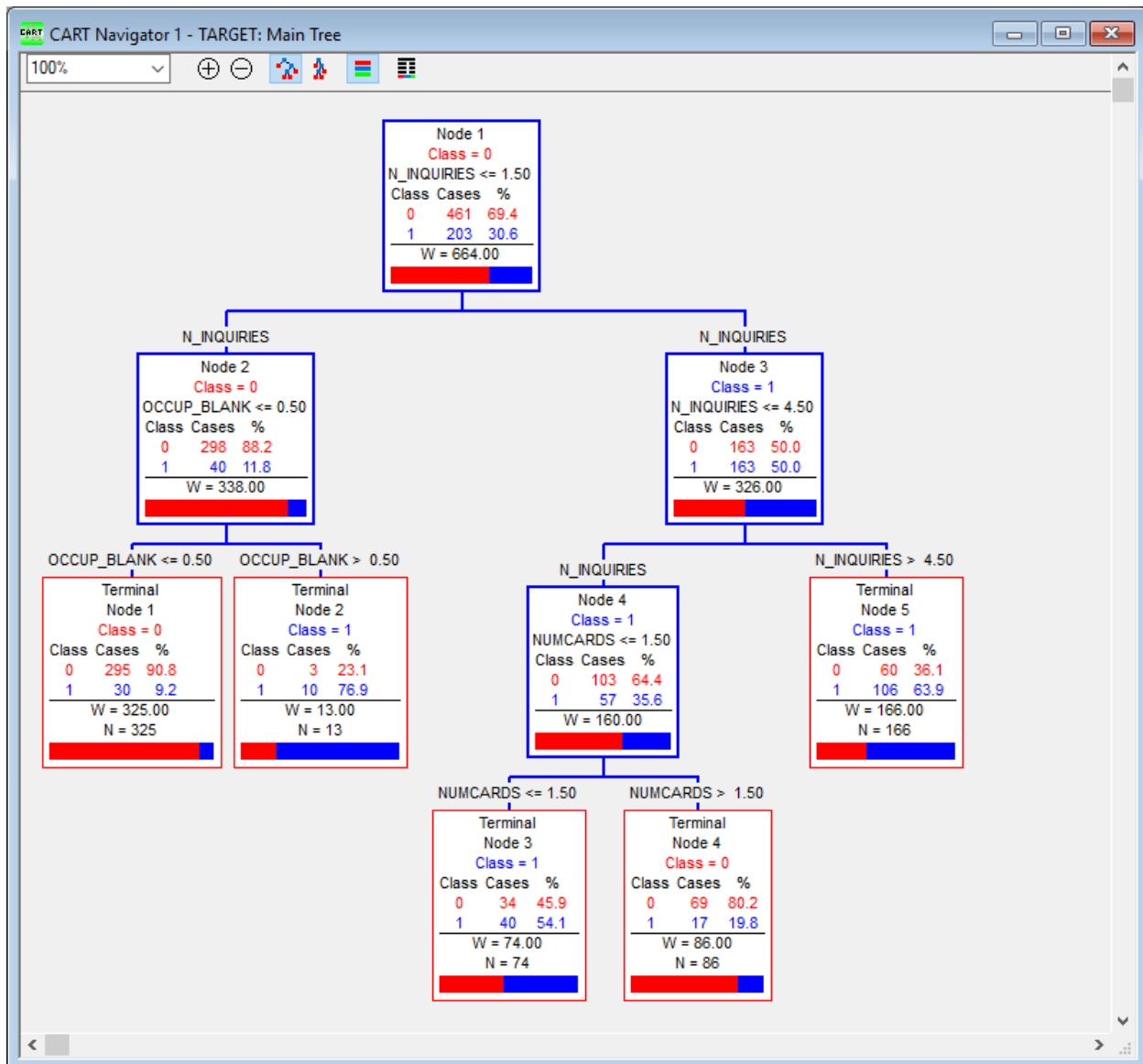
- ✓ The 1SE tree is the smallest tree displaying an error rate that is no worse than one standard error above the optimal tree.
- ✓ Because determining which tree is actually best is subject to statistical error we cannot be absolutely certain which tree is best. Every tree marked in green is a defensible candidate for “best tree.”
- ✓ In our example the 1SE tree has five terminal nodes, with a relative error of .504 and a test ROC of .7552. The optimal tree has a relative error of .488 and a test ROC of .7867. The optimal tree is “better” but it is also twice the size and our measurements are always subject to some statistical uncertainty. For the next displays we will work with the 1SE tree.

A tree of a specific size can be selected in several ways:

- ◆ Use the **mouse to click** on a blue box in the error profile.
- ◆ Use the **left** and **right arrow** keys to reach a specific tree.
- ◆ Press the **[Grow]** or **[Prune]** buttons on the right side of the navigator.
- ◆ From the **T**ree menu select a tree or list of trees

Viewing the Main Tree

The **[Tree Details...]** button on the navigator brings up an industry standard view of a decision tree. This view includes node-specific sample breakdowns so that we can see performance throughout the tree. Starting with the five-node 1SE tree selected, click on the **[Tree Details...]** button at the bottom of the **N**avigator window (or right-click on the root node and select the **Display Tree** option) to get:



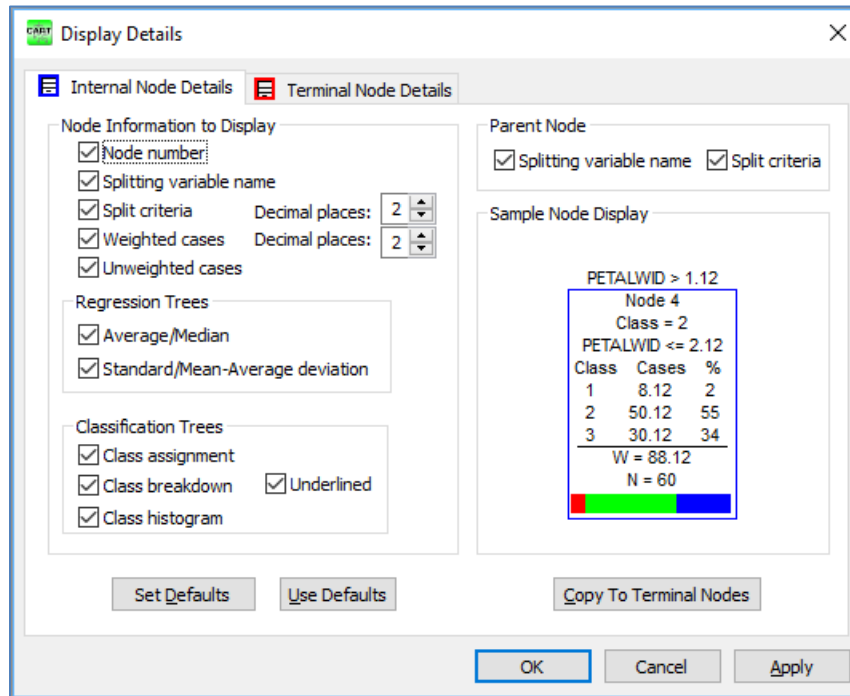
The example shows how CART creates multi-way splits from binary split building blocks. The root node first splits on $N_INQUIRIES > 1.5$ and then again on $N_INQUIRIES > 4.5$. This creates three regions for $N_INQUIRIES$: {0 or 1}, {2, 3, or 4}, and {5 or more}.

With a **mouse click** you can:

- ◆ Zoom in or Zoom out by pressing the or keys.
- ◆ Fine-tune the scale by changing the selection box.
- ◆ Experiment with two alternative node-spacing modes (and buttons).
- ◆ Turn on and off the color coding of target classes (button).
- ◆ Open the **Display Details** window (button).

Try clicking on these controls now to see what they do.

The detail appearing in each of the nodes can be customized separately for internal and terminal nodes. From the **View** menu, select **Node Detail...**; the following dialog appears:



The current display settings are reviewed in a sample node in the right panel. Click on the check boxes to turn each option on and off and then click the **[Apply]** button to update the **Main Tree** display. To save your preferred display options as the default settings, click the **[Set Defaults]** button.

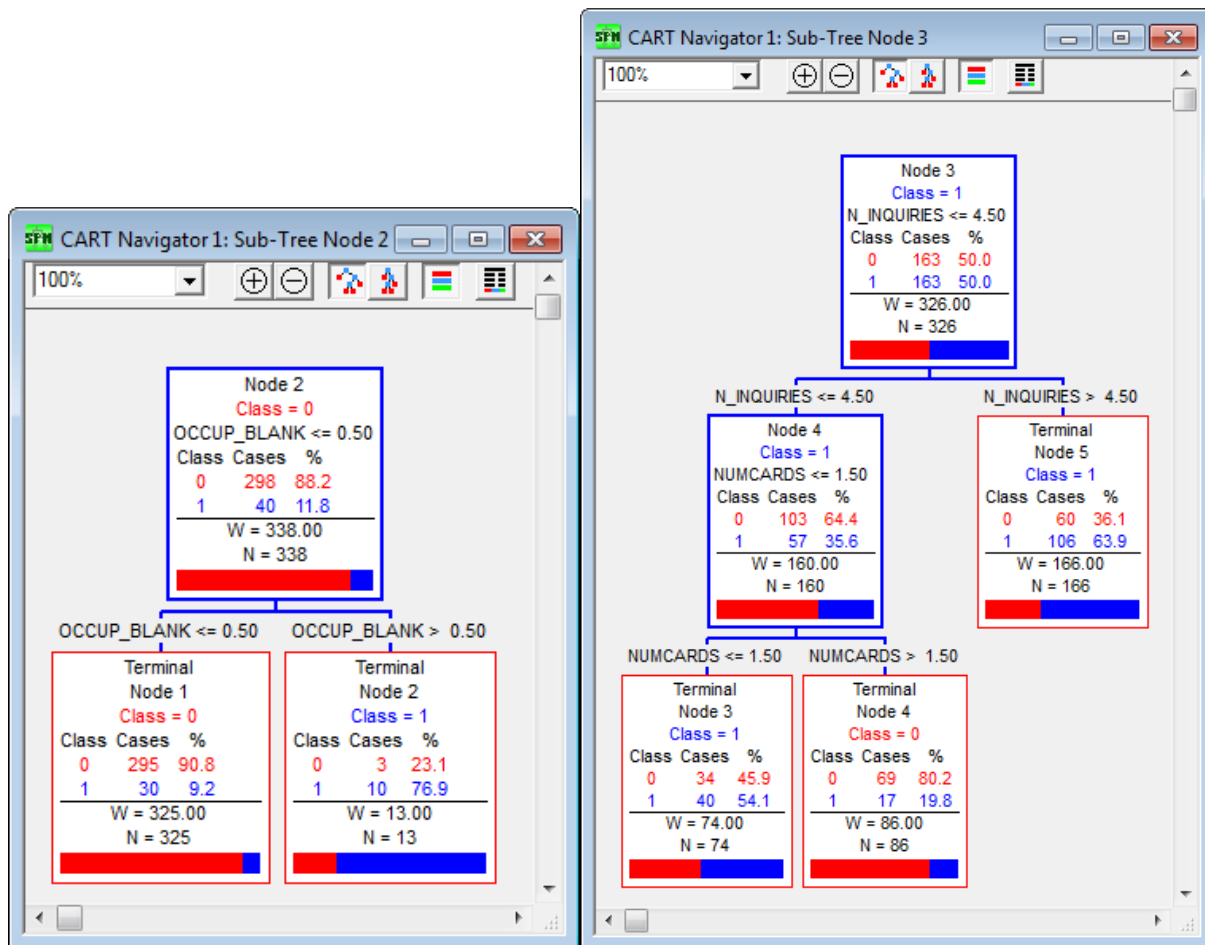
The internal and terminal node detail can be specified separately as each is given its own tab. Press the **[Copy to Terminal Nodes]** or **[Copy to Internal Nodes]** buttons if you wish the current setup to be copied into the other tab.

- ✓ The **[Set Defaults]** button only sets the defaults for the currently active tab. If you want to set defaults for both terminal and internal nodes, press this button twice, once for each tab.

Viewing Sub-trees

Sometimes the tree you want to examine closely is too large to display comfortably on a single screen, and looking at a sub-tree is more convenient. Sometimes you will want to look at two separated parts of the tree side by side. To view sub-trees, first go back to the **Navigator** (you can close the tree details window or select the navigator from the **Window** menu).

Next, right-click on an internal node, and select **Display Tree**. Below we have done this twice: once for the right child of the root and again for the left child, bringing up two sub-tree displays. Below we display the two windows side by side.



Assigning Labels and Color Codes

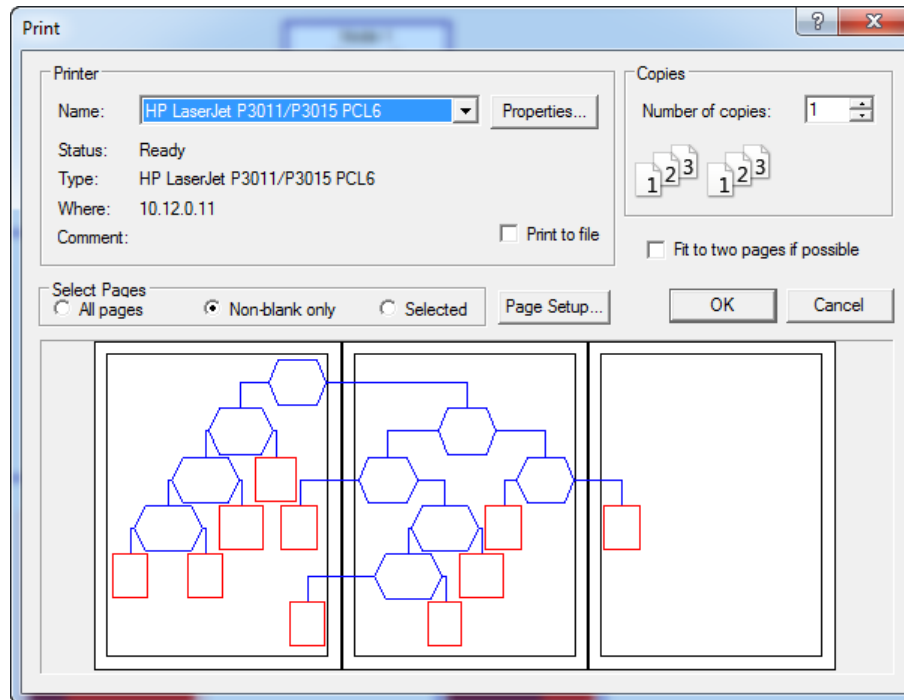
Trees detailing sample breakdowns can be displayed with or without colors; the node histograms are always color-coded. Instructions for customizing the colors appear below. If your target variable is coded as text then the text value labels will be displayed where required, but if your target variable is coded as a number you can replace the numbers with labels with **Class Names**.

Class names (up to 32-characters) and colors can be assigned to each level of the target variable from **View** menu:

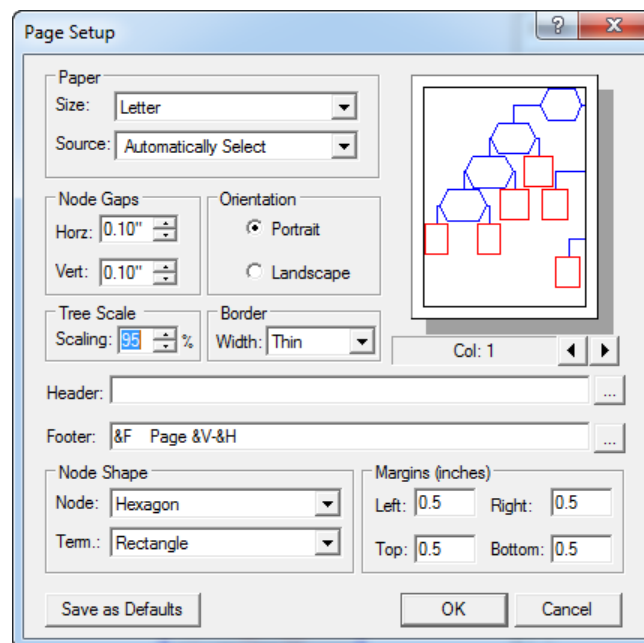
- ◆ Select **Assign Class Names...**
- ◆ Click on the **Assigned Name** text box and enter a label for that class.
- ◆ Click on the **[Color...]** button to select a color from the palette, then click the **[OK]** button.
- ◆ Click the **[Apply]** button to enter the name/color; repeat the above steps for the other levels.

Printing the Main Tree

To print the Main Tree, bring the **CART Navigator: Main Tree** window to the foreground and then select **Print** from the **File** menu (or use **<Ctrl+P>**). In the **Print** dialog box, illustrated below, you can select the pages that will be printed and the number of copies, as well as specify printer properties. You can also preview the page layout; CART will automatically shift the positions of the nodes so they are not split by page breaks.



You can see from the preview that a small section of the GOODBAD main tree spills over to a second and third page. To resize and reorient the tree, click on the **[Page Setup...]** button and reduce the Tree Scale control until the tree fits on two pages:



The **[Page Setup...]** is most useful with larger trees because a little tweaking can reduce the total page count dramatically. You can often obtain convenient thumbnail displays of the most complex tree by selecting **Fit to two pages if possible** on the **Print** menu.

Tree Summary Reports

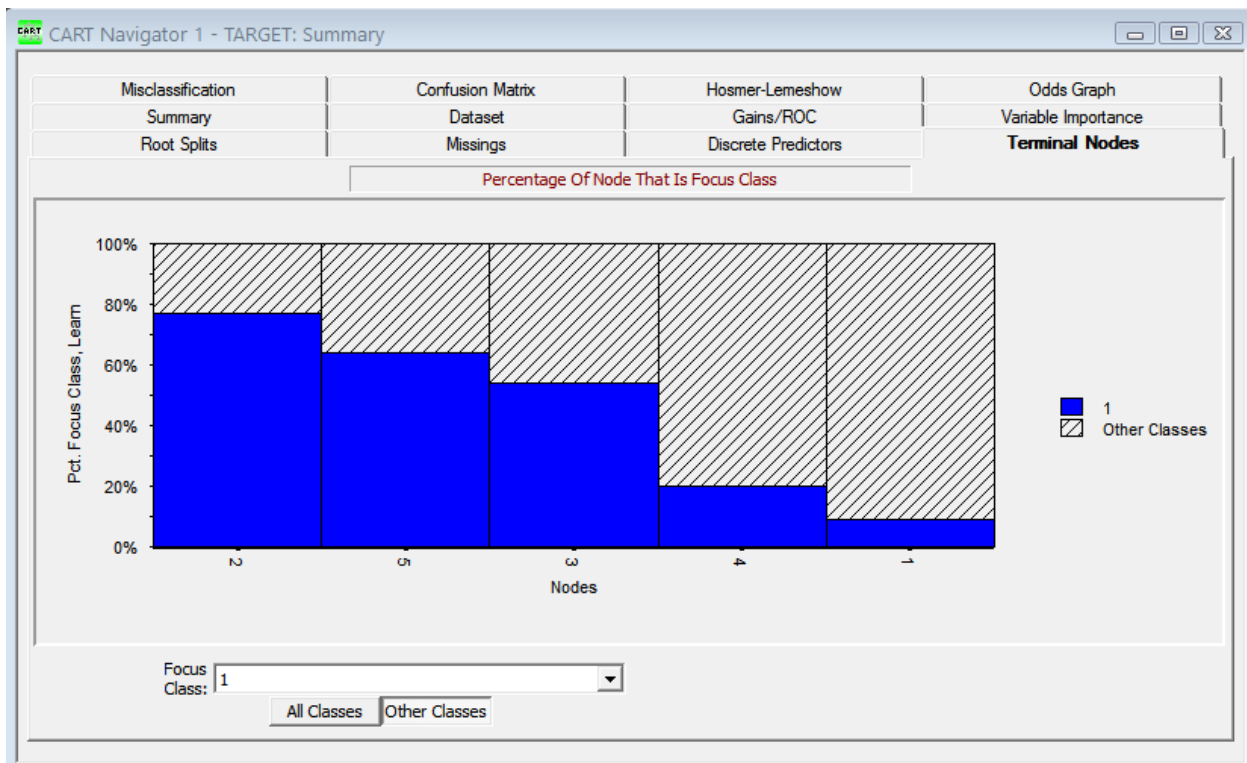
The overall performance of the current tree is summarized in various summary reports. To access the reports, press the **[Summary...]** button at the bottom of the **Navigator** window (or use the **Tree>Tree Summary Reports...** menu).

Tree Summary Reports present information on the currently-selected tree, i.e., the tree displayed in the top panel of the Navigator. To view summary reports for another size of tree, you must first select that tree in the navigator.

Most summary report tabs available in CART are generic to all data mining engines in the SPM and are described in a separate chapter. Here we only comment on the CART-specific tabs.

Terminal Nodes

The **Terminal Node** tab provides a graphical representation of the ability of the tree to capture the BADs in the terminal nodes.



Observe that we selected BAD as the target class. This sorts the nodes so that those with the highest concentrations of BAD are listed first.

Note the following controls:

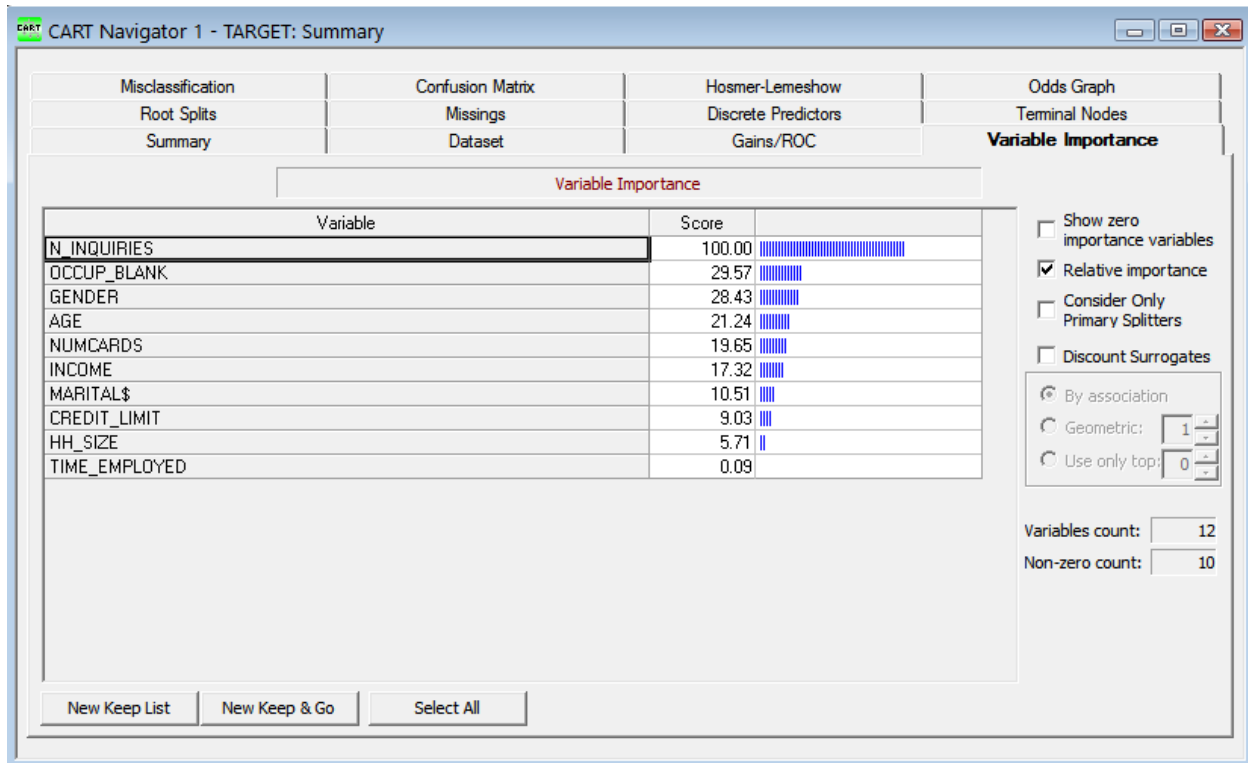
- ◆ Use the **Focus Class** selection box to put the class of interest in focus.
- ◆ Press the **[All Classes]** button to represent each class with its own color.
- ◆ Press the **[Other Classes]** button to color only the selected class while the other classes are just colored gray.

Node 2 has the highest concentration of BADs, closely followed by nodes 5, 3 and 4. Hover the mouse over a bar to see the precise fraction of the node that is BAD. This is a graphical display of the information that is also in the gains chart.

If you have separate test data you can request a learn/test comparison of the terminal nodes in this window.

Variable Importance

The **Variable Importance** tab rank variables according to their contribution into the currently selected tree.



It is natural to expect that the root node splitter will be the most important variable in a CART tree and indeed in our example this is the case. However, you cannot count on it coming out this way in every tree. Sometimes a variable that splits the tree below the root is most important because it ends up splitting many nodes in the tree and splitting powerfully. Variable importance is determined by looking at every node in which a variable appears and taking into account how good a splitter it is. You should think of the variable importance ranking as a summary of a variable's contribution to the overall tree when all nodes are examined. The formulas for variable importance calculations are detailed in the CART monograph.

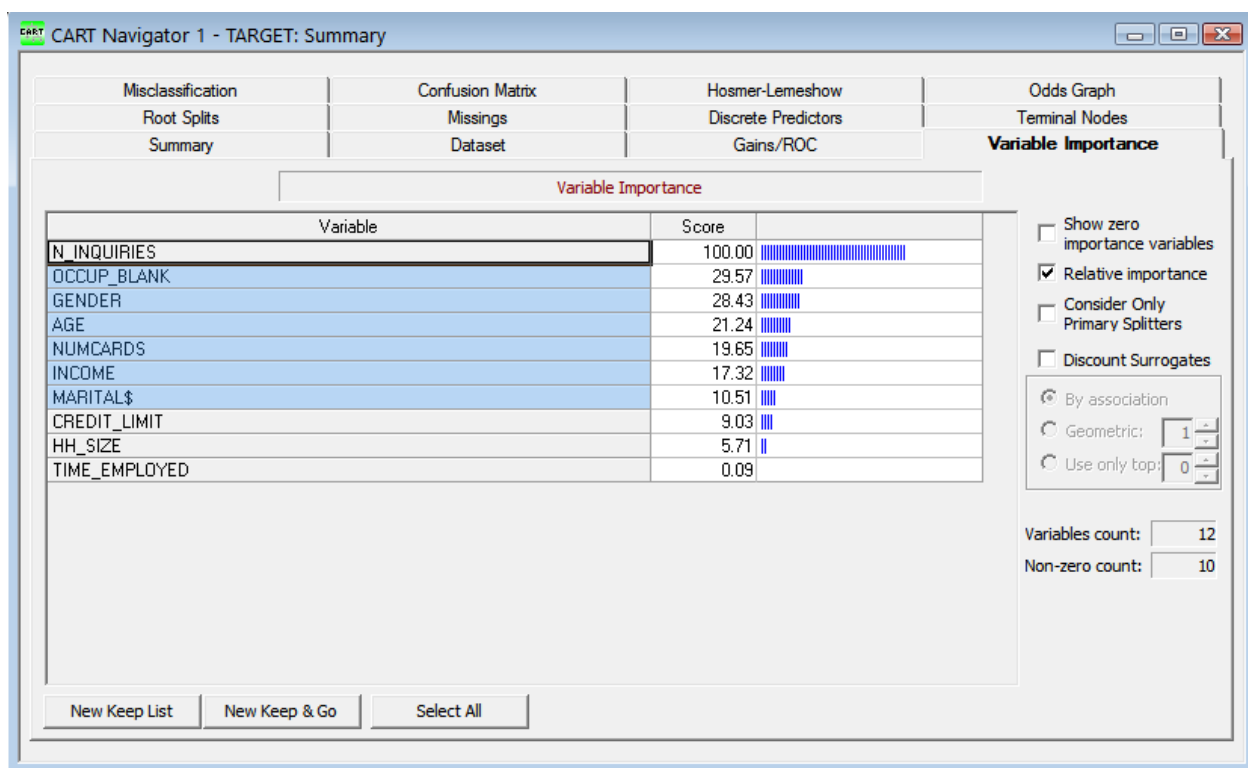
Variables earn credit towards their importance in a CART tree in two ways: as primary splitters that actually split a node, and as surrogate splitters (back-up splitters to be used when the primary splitter is missing). To see how the importance scores change if considered only as primary splitters, click the **Consider Only Primary Splitters** check box; CART automatically recalculates the scores.

- ✓ Comparing the standard CART variable importance rankings with the **Consider Only Primary Splitters** can be very informative. Variables that appear to be important but rarely split nodes are probably highly correlated with the primary splitters and contain very similar information.

- ✓ Click on the **Discount Surrogates** check box if you want to discount the contribution of surrogates into the importance scores without completely eliminating them. We offer three discounting strategies: **By association**, **Geometric**, and **Use only top <n>**.
- ✓ To see the raw improvement-based importance scores calculated by CART uncheck the **Relative importance** check box.

By default, variables with zero importance (which means they are not entering into the tree as a main splitter nor as a surrogate) are not shown. Click on the **Show zero importance variables** selection box in you insist on seeing them.

Click inside any column of the variable importance chart to start highlighting rows. You can use this to select variables on which to focus on in a new analysis. Below we have selected the seven variables that have importance scores above 10.0 (arbitrary selection).



Once you have highlighted variables in this way on the variable importance chart you can automatically build a new model using only those predictors. Just press the **[New Keep & Go]** button to initiate a new modeling process.

Clicking on the **[New Keep List]** button creates a list of those variables and places them on a KEEP list in a new notepad. You can edit this KEEP command and place it in scripts or just save it for later use.

Detailed Node Reports

To see what else we can learn about our CART trees, return to the **CART Navigator** by closing the **Summary Results** window or by selecting **Navigator** from the **Window** menu. Move the mouse pointer to the root (top) node in the tree topology panel and click to activate a Node Report dialog (or right-click on the root node and select **Node Report**).

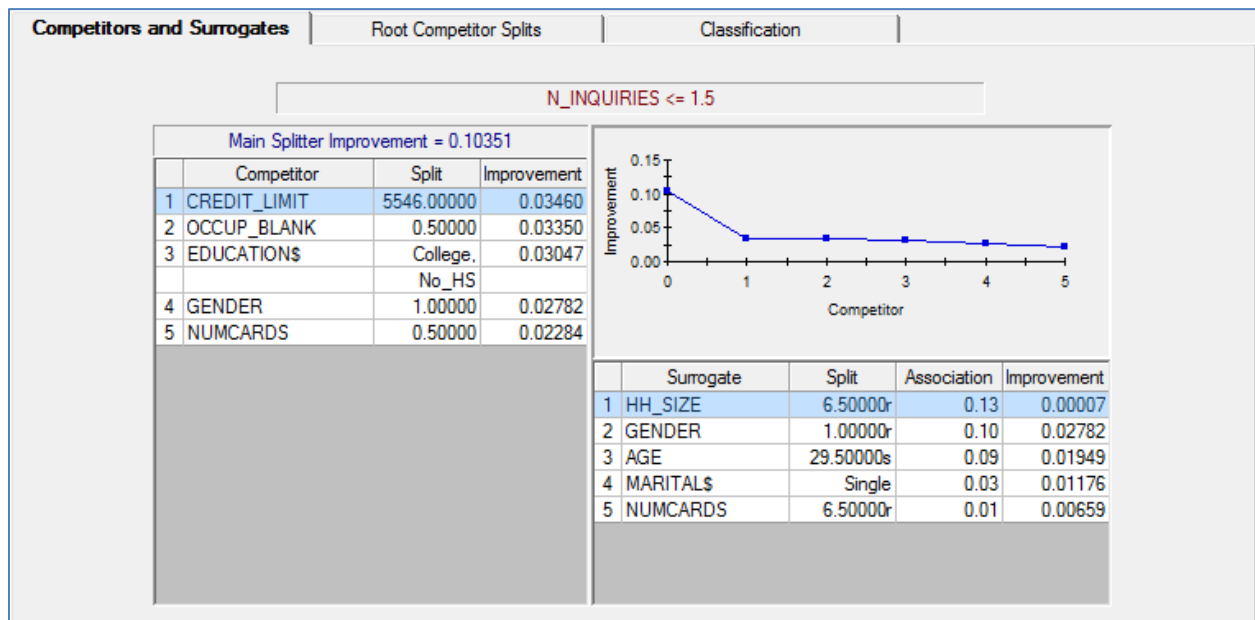
You can click on any node to see the corresponding node report.

Most tabs described below are applicable to any node. Some of the tabs may only be applicable to the root node, internal nodes only, or nodes based on categorical splits.

Competitors and Surrogates tab

This tab is available for non-terminal nodes only.

As illustrated below, the first of the three tabs in the non-terminal node report provides node-specific information for both the competitor and the surrogate splits for the selected node (in this case, the root node).



The splitting rule, *Is N_INQUIRIES <= 1.5*, is displayed in the top line, and the **Main Splitter Improvement** is displayed in the following line on the left. Splitter improvement is the metric CART uses to evaluate the quality of all splits; it is computed differently for different splitting rules. A table of the top five competitor splits in decreasing order of importance is displayed in the left panel. Each competitor is identified by a variable name, the value at which the split would be made, and the improvement yielded by the split.

- ✓ You may need to alter the width of the columns in this display to make everything we discuss here visible. Just position your mouse in the column header and over the border you wish to move. When the cursor changes to a cross-hairs right-click and drag the border to widen or narrow the column.

The best competitor, CREDIT_LIMIT, would split at the value 5546 and would yield an improvement of 0.0346, quite a bit below the main splitter improvement of 0.1035. Improvement scores should be looked at in relative rather than absolute terms. The improvement of the main splitter is almost three times that of the best competitor, an unusually large (but not suspiciously large) ratio. The quality of the competitor splits relative to the primary split can also be evaluated by inspecting the line graph displayed in the upper-right panel. The improvement yielded by each competitor split appears on the y-axis and the number or rank of the competitor split on the x-axis, with the primary splitter improvement displayed at x=0. The graph makes plain that the primary splitter is quite a bit better than the closest competitor but that the 2nd, 3rd, and 4th competitors all have similar improvements.

Surrogates are an important innovation in CART technology and play a key role in CART prediction and tree interpretation. A surrogate splitter is a splitter that is “similar to” the main splitter in how it assigns cases in a node to the left and right children. The top surrogate is the splitter that comes closest to matching the main splitter’s left-right assignments, but “closest” does not necessarily mean close. In the example, the top surrogate has an association score of 0.13 (on a scale of 0.00 to 1.00), which is a rather weak association. (You can think of the association as akin to correlation, but scores above 0.20 represent a good degree of matching.)

When a splitter does not have any close matching surrogates it means that the information content of that variable is unique and cannot be easily substituted for by any other variable. In this example, it should not be surprising to learn that the credit bureau variable N_INQUIRIES contains unique information not reflected in the other variables.

The top five surrogates are ranked by association score and are listed in the bottom-right panel, along with the splitting criterion and the improvement yielded by the surrogate split. In this example, the best surrogate, HH_SIZE, has an association value of 0.13, and a low improvement of 0.0007. The next surrogate, GENDER, is ranked 2nd because of its association score but offers a much better improvement.

- ✓ Surrogates play the role of splitter when the primary splitter is missing. They play the role of “backup splitter” and are consulted in order. If both the primary and first surrogate splitter are missing, CART would make use of the 2nd ranked surrogate.

Classification tab

This tab is available for any node.

The **Classification** tab displays node frequency distributions in a bar graph (or, optionally, a pie chart or horizontal bar chart) for the parent-, left -and right-child nodes. If you use a test sample, frequency distributions for learn and test samples can be viewed separately using the **[Learn]** or **[Test]** buttons.

Below we show the report for the root node. The left child is now clearly dominated by GOODs and the right child contains an equal number of GOODs and BADs.



The window offers a choice between bar charts, pie charts and a horizontal bar chart embedding the sample split. You can switch between counts and percentages by pressing the **[Cases]** or **[Pct]** buttons. The horizontal bar chart offers an alternative view of the class partitions. Each colored bar represents one target class. The vertical line shows how the class was partitioned between two children, with the percentage of the class going to the left child shown on the left side and the percentage of the class going to the right child shown on the right side. In this example, less than 20% of Class 1 went to the left side and more than 80% went to the right side.

Root Competitor Splits tab

This tab is only available for the root node.

In the root node a splitter has access to all the data. Thus, we have a special interest in the performance of variables as splitters in the root. This report lists every variable available for splitting and includes this additional information:

- ◆ **Competitor** – list of variables according to their improvement strength as a competitor.
- ◆ **Split** – the best split value found in this variable (for categorical variables, a list of categories is provided).
- ◆ **Improvement** – the corresponding split improvement.
- ◆ **N Left** – the number of observations with legitimate values of the splitter going to the left.
- ◆ **N Right** – the number of observations with legitimate values of the splitter going to the right.
- ◆ **N Missing** – the number of observations with missing values of the splitter.

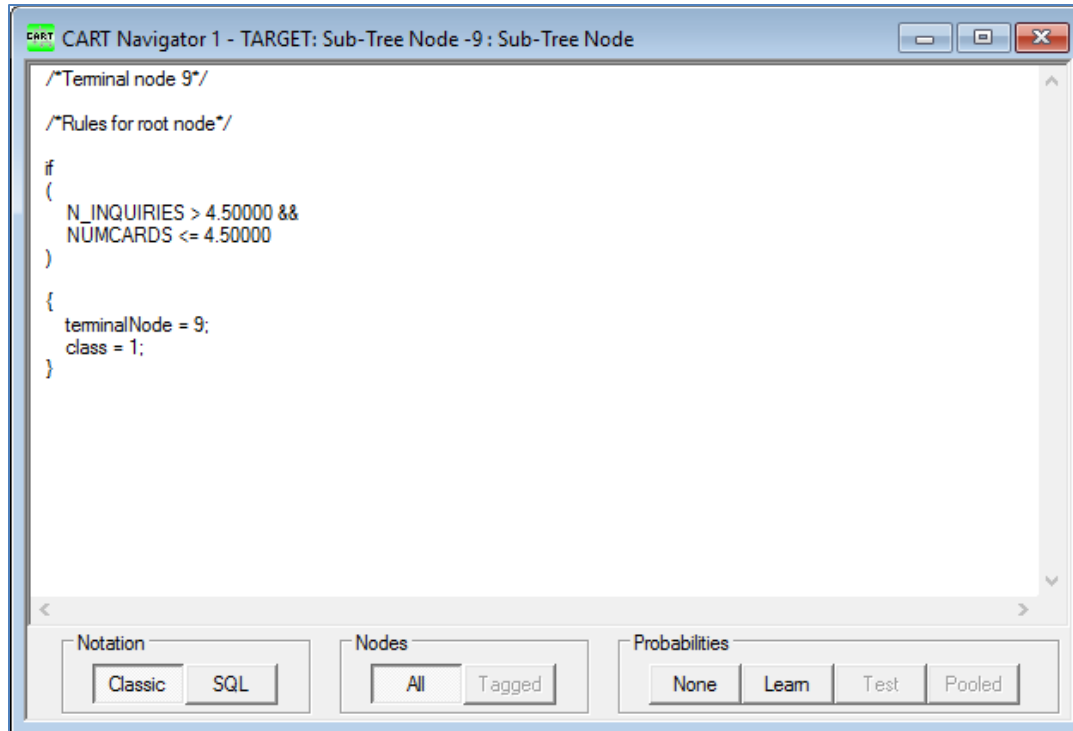
Competitors and Surrogates		Root Competitor Splits		Classification		
N_INQUIRIES <= 1.5						
	Competitor	Split	Improvement	N Left	N Right	N Missing
Main	N_INQUIRIES	1.50000	0.10351	338	326	0
1	CREDIT_LIMIT	5546.00000	0.03460	278	386	0
2	OCCUP_BLANK	0.50000	0.03350	628	36	0
3	EDUCATION\$	College,	0.03047	520	133	11
		No_HS				
4	GENDER	1.00000	0.02782	394	195	75
5	NUMCARDS	0.50000	0.02284	228	436	0
6	INCOME	606.50000	0.02076	49	615	0
7	AGE	29.50000	0.01949	243	337	84
8	MARITAL\$	Single	0.01176	335	328	1
9	TIME_EMPLOYED	5.25000	0.00767	612	52	0
10	OWNRENT\$	Parents,	0.00622	235	286	143
		Rent				
11	HH_SIZE	1.50000	0.00512	93	451	120

In some circumstances you may be uncomfortable with a main splitter because it is too frequently missing or because it generates a highly uneven split. For example, OCCUP_BLANK puts 628 cases on the left and only 36 cases on the right. OWNRENT\$ has 143 cases missing. Other sections of the manual discuss what you can do if your main splitter exhibits such possibly undesirable characteristics.

The Rules tab

This tab is available for any node other than the root, for which the rules would be empty.

The **Rules** tab will display text rules describing how to reach the node selected, and thus is available for every node except the root. Select **Terminal node 9** from the **10-node tree**, click on the node and then select the **Rules** tab to see:



Node 9

segment is described by the joint set of rules on N_INQUIRIES and NUMCARDS and is estimated to be 70% BAD.

- ✓ Press one of the **Probabilities** buttons to see them displayed.

The rules are formatted as C-compatible code to facilitate applying new data to CART models in other applications. The rule set can be exported as a text file, cut and pasted into another application, and/or sent to the printer.

Splitter tab

This tab is only available for a non-terminal node which was split on a categorical predictor.

When a node is split on a categorical variable, an additional **Splitter** tab is available in the **Node Information** window for all internal nodes. In our example, we will not see a categorical splitter in the tree unless we expand the tree out to 26 nodes. If you do that and go to the parent of Terminal Node 3 (at the bottom left) you will see that it splits on the categorical EDUCATION\$ variable. Click that node and select the **Splitter** tab:

Competitors and Surrogates	Classification	Rules	Splitter
Is EDUCATION\$ =HS,No_HS			
Levels That Go Left		Levels That Go Right	
HS		College	
No_HS			


With only three education levels we can readily see whether a level goes to the left or the right. This report is most useful for following high-level categorical splits or for tracing which levels end up where when the same categorical variable is used as the main splitter multiple times.

Saving the Navigator/Grove File

To save the **Navigator** so that you can subsequently reopen the file for further exploration in a later CART session, first make sure that the Navigator is your active window (click anywhere on the Navigator) and press the **[Grove...]** button. This will open the generic **Save As** dialog which allows you to save the grove file.

- ✓ Previous versions of CART saved two types of tree files: Navigator files (with extensions like .nav or .nv3) and grove files. Starting with CART 6.0, the Navigator file is embedded inside the grove file and no longer makes use of a separate Navigator file format. The SPM will recognize and read old navigator files and you can load these from the **File-Open-Open Navigator** menu selection.
- ✓ If the trees you are building are large (e.g., several thousand terminal nodes), Windows' system resources can become depleted. To avoid memory problems, consider periodically closing the open **Navigator** windows you will not need.

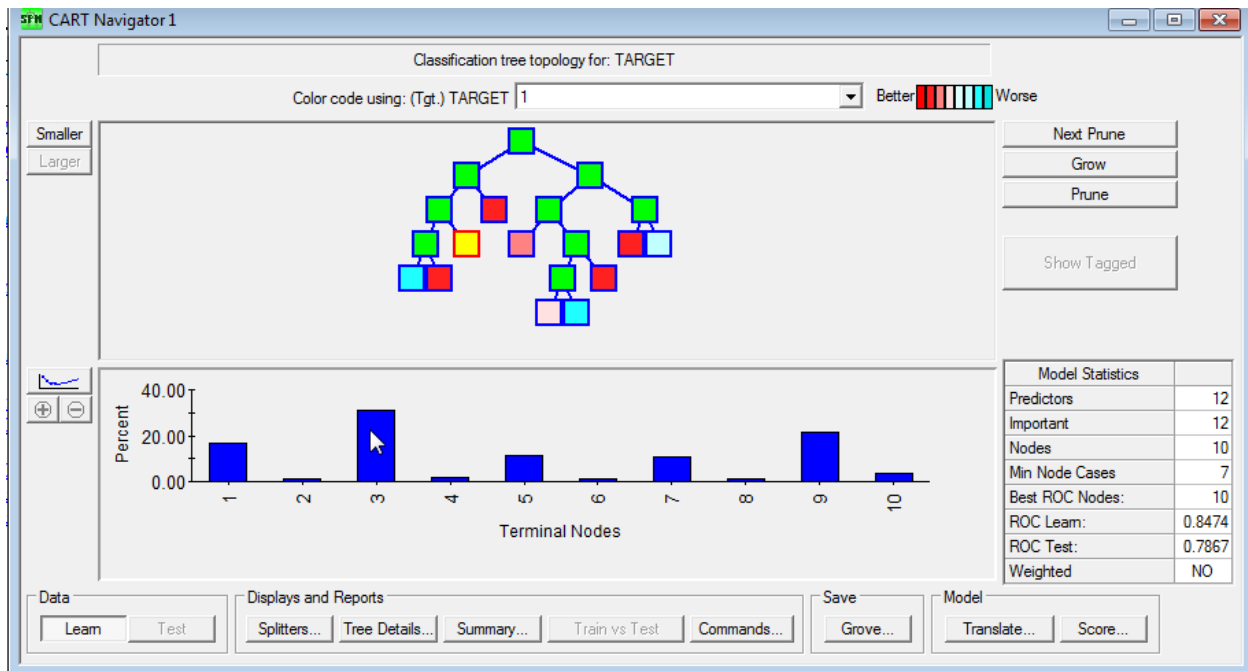
More Navigator Controls

Press the  button on the left margin of the **Navigator** window to cycle through three display modes in the bottom half of the **Navigator** window:


- ◆ **Standard Relative Cost curve.**
- ◆ **Color-coded Relative Cost curve**, to display standard error rule.
- ◆ **Percent population by node histogram.**

The first two displays show the relative cost curve depending on the number of terminal nodes, while the last display reports how the original data set is distributed into the terminal nodes in the currently-selected tree.

- ✓ If you click on an individual bar in the “percent population by node” display, the corresponding node in the tree topology is briefly highlighted.



Classification tree topology for: TARGET

Color code using: (Tgt.) TARGET 1 Better  Worse

Smaller
Larger

Next Prune
Grow
Prune
Show Tagged

Model Statistics	
Predictors	12
Important	12
Nodes	10
Min Node Cases	7
Best ROC Nodes:	10
ROC Learn:	0.8474
ROC Test:	0.7867
Weighted	NO

Data: Learn Test
Displays and Reports: Splitters... Tree Details... Summary... Train vs Test Commands...
Save: Grove...
Model: Translate... Score...

Press the **[Smaller]** or **[Larger]** buttons to change the scale of the tree topology in the top half of the **Navigator** window to become larger or smaller. This is useful when analyzing large trees.

When applicable, you may switch between learn or test counts displayed for each node by pressing the **[Learn]** or the **[Test]** buttons. Since cross validation was used in this example, only learn counts are available on the node-by-node basis.

CART Text Output

The **Classic Output** window contains the detailed technical log that will always be produced by the non-GUI CART running on UNIX, Linux, and mainframe platforms. Most modelers can safely ignore this window because the same information is reported in the GUI displays we have been demonstrating in this tutorial. The classic text output will contain some exclusive reports and advanced information of interest to experienced modelers.

To turn to the text output, select **Classic Output** (shortcut: Ctrl-Alt-C) from the **Window** menu, or click on the window if you can see it. The classic output contains an outline panel on the left with hyperlinks for jumping to the specific locations. Below we selected the second topic in the outline: **Missing Value Prevalence**:

The screenshot shows the 'Classic Output (Ctrl+Alt+C)' window. On the left is a 'Report Contents' sidebar with a tree structure. The main area displays the 'Missing Value Prevalence' report, which includes a table of variable prevalence for the 'Learn' phase and a table of 'Root Node Competitors In Order of Improvement'.

Report Contents

- Session Start
- Tree 1
 - Start
 - Cardinality Summary
 - Target Frequency Table
 - Descriptive Statistics
 - Missing Value Prevalence
 - Root Node Competitors
 - Tree Sequence
 - Node Information
 - Terminal Nodes
 - Misclassification
 - Class Table (CV)
 - Class Prob Table (CV)
 - Class Table (Learn)
 - Class Prob Table (Learn)
 - Variable Importance
 - Variable Importance by N Classes, Importance and
 - Dataset Position
 - CART Settings
 - Competitor List
 - CV Partition Details - For Target Class 1
 - CV ROC By Pruning - For Target Class 1
 - Tree Optimality Criteria

Missing Value Prevalence

```

=====
                        Learn
=====
OWNRENT$    0.2154
HH_SIZE     0.1807
AGE         0.1265
GENDER      0.1130
EDUCATION$  0.0166
MARITAL$    0.0015

PRIORS SET EQUAL

Preprocessor CPU TIME: 00:00:00.03

=====
Root Node Competitors In Order of Improvement
=====

```

Competitor	Split	Improvement	Imp. Ratio	N Left	N Right
N_INQUIRIES	1.5	0.103506		338	326
CREDIT_LIMIT	5546	0.034604	0.334320	278	386
OCCUP_BLANK	0.5	0.033502	0.323671	628	36
EDUCATION\$	"College",	0.030469			

You can save a copy of the text output as a record of your analysis using the **File>Save>Save Output...** menu. You can also copy and paste sections of the output into another application or to the clipboard. The font used in the **Report** window can be changed using the **Edit>Fonts...** menu. Use a mono-spaced font such as `Courier` to maintain the alignment of tabular output.

- ✓ You can always regenerate most of the classic output from a saved Grove file by using the **TRANSLATE** facility built into every grove.
- ✓ Advanced users may want to use text parsing scripts (Perl, Python, etc.) to process the classic output to create custom reports.

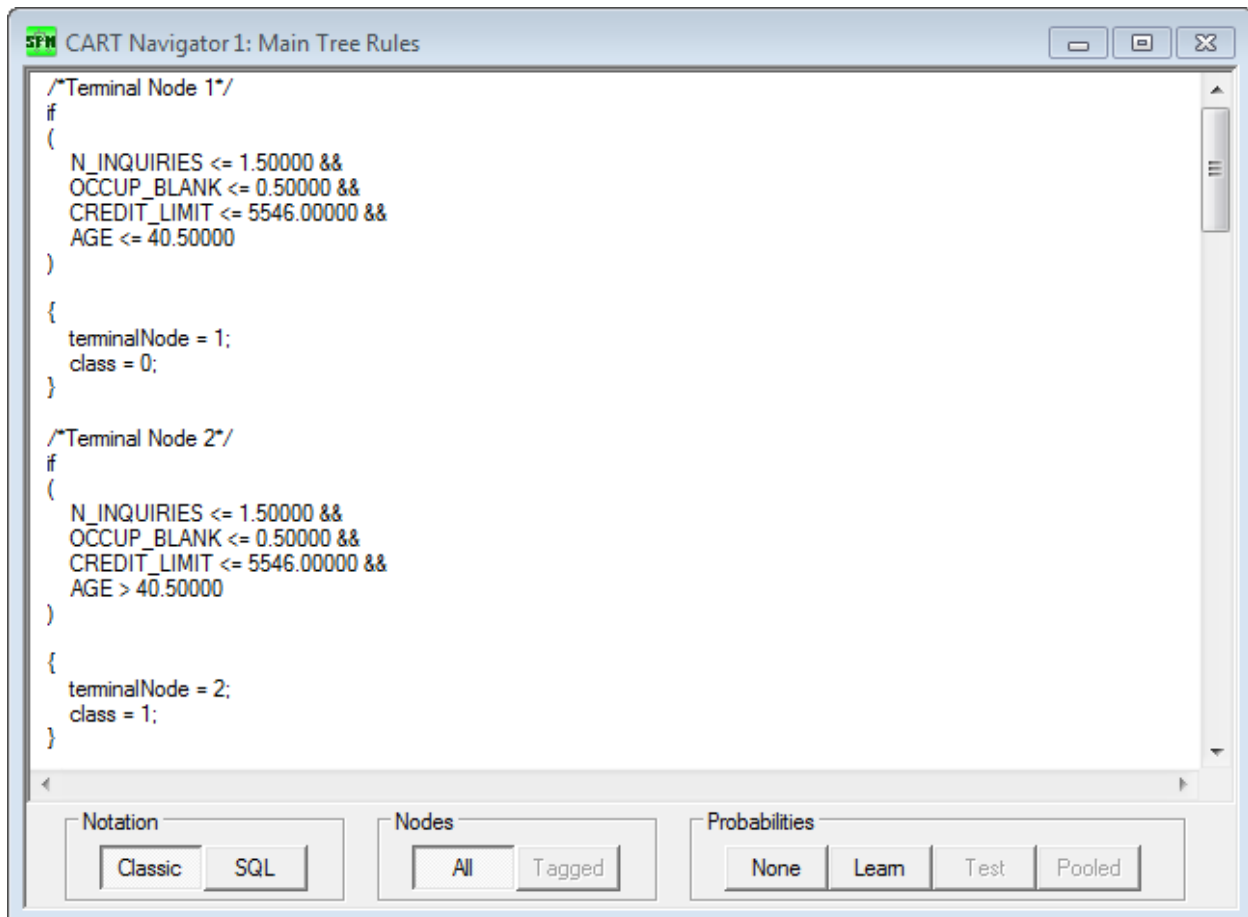
Displaying and Exporting Tree Rules

Decision trees can be viewed as flow charts or as sets of rules that define segments of interest in a database. The rules for a CART tree can be rendered in two quite different ways:

- ◆ As simple rules that are easy to read and understand (approximate model).
- ◆ As complex computer code (more accurate model).

This section focuses on the first form of the rules. The second form is discussed in the sections on scoring and translation.

Every node displayed in a navigator can be described by the rules that lead to it from the root. To view the rules just right click on the node and select **Rules**. If you select the root node for rule extraction you actually get the rules for every terminal node in the tree. Below we show this for our example.



The screenshot shows a window titled "SFM CART Navigator 1: Main Tree Rules". The main area contains the following code:

```

/*Terminal Node 1*/
if
(
  N_INQUIRIES <= 1.50000 &&
  OCCUP_BLANK <= 0.50000 &&
  CREDIT_LIMIT <= 5546.00000 &&
  AGE <= 40.50000
)
{
  terminalNode = 1;
  class = 0;
}

/*Terminal Node 2*/
if
(
  N_INQUIRIES <= 1.50000 &&
  OCCUP_BLANK <= 0.50000 &&
  CREDIT_LIMIT <= 5546.00000 &&
  AGE > 40.50000
)
{
  terminalNode = 2;
  class = 1;
}

```

At the bottom of the window, there are three groups of buttons:

- Notation:** Classic, SQL
- Nodes:** All, Tagged
- Probabilities:** None, Learn, Test, Pooled

You have a few further options on this window:

- ◆ The rules can be displayed as standard C or SQL programming code.
- ◆ Probabilities can be based on Learn data, Test data (if available), or on the combination of learn and test data.
- ◆ Rules can be displayed for specific nodes only (those you have tagged on the navigator via the right mouse click menu).

This rules display is intended only as a rough guide. The rules produced are only an approximate version of the CART model because they do not contain information about surrogate splits. You should use the

Translate feature (available by pressing the **[Translate...]** button in the **Navigator** window) to get the complete representation of the CART model, including surrogates.

Scoring and Translation of CART models

There are many reasons to score data with a CART model. You might want to run a quick test of the model's predictive power on new data, or you might actually embed your model into a business process. CART gives you several options for doing this:

- ◆ CART can score data from any source using any previously-built CART model. All you need to do is to attach to your data source, let CART know which grove file to use, and decide where you want the results stored (Scoring facility of the SPM).
- ◆ You can TRANSLATE your model into one of several programming languages including SAS, Classic, C, PMML, Java and Topology. The code produced needs no further modification and is ready to be run in accordance with the instructions provided in the main reference manual (Translation facility of the SPM).
- ◆ CART scoring engines are available for deployment on high performance servers that can rapidly process millions of records in batch processes (Requires a separate custom agreement).

The scoring and translation facilities are described in the generic section of this manual common to all engines. Press the **[Score...]** button in the **Navigator** window to activate the scoring dialog window. Press the **[Translate...]** button in the **Navigator** window to activate the translation dialog. Press the **[Select...]** button to pick tree of any desirable size. Press the **[Optimal]** button to pick the optimal tree suggested by CART.