
SPM Users Guide

Introducing TreeNet



This guide describes the TreeNet Product and illustrates some practical examples of its basic usage and approach

TreeNet Introduction

TreeNet is a revolutionary advance in data mining technology developed by Jerome Friedman, one of the world's outstanding data mining researchers. TreeNet offers exceptional accuracy, blazing speed, and a high degree of fault tolerance for dirty and incomplete data. It can handle both classification and regression problems and has been proven to be remarkably effective in traditional numeric data mining and text mining. (Please see our TreeNet Technical User Manual on text mining for further details.)

The next section introduces TreeNet and answers some basic questions about the methodology.

Basic Principles of TreeNet

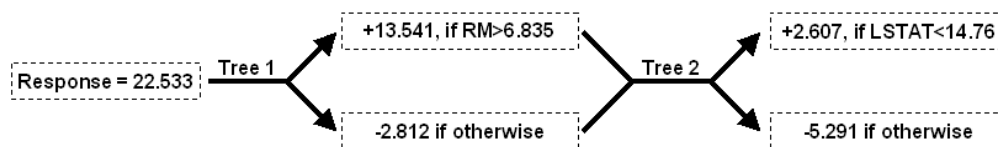
A TreeNet model normally consists of from several hundred to several thousand small trees, each typically containing about six terminal nodes. Each tree is devoted to contributing a small portion of the overall model and the final model prediction is constructed by adding up all the individual tree contributions. You can think of the TreeNet model as a black box that has been proven to deliver exceptionally accurate models. TreeNet offers detailed self-testing to document its reliability on your own data. In many ways the TreeNet model is not that mysterious, although it is undeniably complex. You do not need to concern yourself with that complexity because all results, performance measures, and explanatory graphs can be grasped by anyone familiar with basic data mining principles. However, for those wishing to better understand the details of TreeNet model construction, we provide an outline here.

The model is similar in spirit to a long series expansion (such as a Fourier or Taylor's series) - a sum of factors that becomes progressively more accurate as the expansion continues. The expansion can be written as:

$$F(X) = F_0 + \beta_1 T_1(X) + \beta_2 T_2(X) + \dots + \beta_M T_M(X)$$

In the equation above each T_i can be considered as a small tree. You should read this as a weighted sum of terms, each of which is obtained from the appropriate terminal node of a small tree.

By way of example we display the first few terms from a regression model based on a well-known data set: the Boston Housing data extracted from the 1970 US Census. This data set is usually used to build models predicting the median value of houses in a neighborhood based on quality of life variables, such as crime rate, school quality, and socioeconomic status of the neighborhood, and a few core housing descriptors, such as typical house age and size.



The model above begins with an estimate of mean home value (in 1970) of \$22,533. We can use this as a baseline from which adjustments will be made to reflect characteristics of the housing and the neighborhood. In the first term the model states that the mean value would be adjusted upwards by \$13,541 for larger homes, and adjusted upwards again by \$2,607 for neighborhoods with good socioeconomic status indicators. In practice, the adjustments are usually much smaller than shown in this



regression example and hundreds of adjustments may be needed. The final model is thus a collection of weighted and summed trees.

Although this example is a regression, exactly the same scheme is used for classification problems. For binary classification problems, a yes or no response is determined by whether the sign of the predicted outcome is positive or negative. For multi-class problems, a score is developed separately for each class via class-specific expansions and the scores are converted into a set of probabilities of class membership.

The Boston housing regression example above uses the smallest possible two-node tree in each stage. More complicated models tracking complex interactions are possible with three or more nodes at each stage. The TreeNet default uses a six-node tree, but the optimal tree size for any specific modeling task can only be determined via trial and error. We have worked on problems that are handled perfectly well with two-node trees, and one of our award-winning data mining models used nine-node trees. Fortunately, TreeNet models run very quickly, so it is easy to experiment with a few different-sized trees to determine which will work best for your problem.

Typical Questions and Answers about TreeNet

What are the advantages of TreeNet?

In our experience TreeNet is the closest tool we have ever encountered to a fully automated statistician. TreeNet can deal with substantial data quality issues, decide how a model should be constructed, select variables, detect interactions, address missing values, ignore suspicious data values, prevent the model from being dominated by just a few ultra-powerful variables, and resist any overfitting.

Typically, the end result is a model far more accurate than any that could be constructed using other data mining tools and at least as accurate as models painstakingly built by hand by experts working for weeks or months. Of course there is no substitute for a good understanding of the subject matter and problems being addressed and a reasonable familiarity with the data is always required for reliable results. However, given that, TreeNet can give you a substantial advantage in reaching high performance models quickly.

Below is a summary of TreeNet's key features:

- ◆ Automatic selection from thousands of candidate predictors.
 - No prior variable selection or data reduction is required.
- ◆ Ability to handle data without preprocessing.
 - Data do not need to be rescaled, transformed, or modified in any way.
 - Resistance to outliers in predictors or the target variable.
 - Automatic handling of missing values.
 - General robustness to dirty and partially inaccurate data.
- ◆ High Speed:
 - Trees are grown quickly; small trees are grown extraordinarily quickly.
 - TreeNet is able to focus on the data that are not easily predictable as the model evolves. Thus, as additional trees are grown, fewer and fewer data need to be processed.



- TreeNet is frequently able to focus much of its training on just 20% of the available data (all accomplished automatically without user intervention).
- ◆ Resistance to Overtraining.
- ◆ When working with large data bases even models with 2,000 trees show little evidence of overtraining.

TreeNet's robustness extends to the most serious of all data errors: when the target variable itself is sometimes incorrect, for example, when a "yes" is incorrectly recorded as a "no," or vice versa. In the machine learning world such data are said to be contaminated with erroneous target labels. For example, in medicine there is some risk that patients labeled as healthy are in fact ill and vice versa. This type of data error can be challenging for conventional data mining methods and can be catastrophic for conventional forms of "boosting." In contrast, TreeNet is generally immune to such errors as it dynamically rejects training data points too much at variance with the existing model.

In addition, TreeNet adds the advantage of a degree of accuracy usually not attainable by a single model or by ensembles such as bagging or conventional boosting. Independent real world tests in text mining, fraud detection, and credit worthiness have shown TreeNet to be dramatically more accurate on test data than other competing methods.

Of course no one method can be best for all problems in all contexts. Typically, if TreeNet is not well suited for a problem it will yield accuracies on par with that achievable with a single CART tree.

What are the advantages of TreeNet over a neural net?

Several of our most avid TreeNet users are former neural network advocates who have discovered how much faster and easier it is to get their modeling done with TreeNet while sacrificing nothing in the area of accuracy. In contrast to neural networks, TreeNet is not overly sensitive to data errors and needs no time-consuming data preparation, preprocessing or imputation of missing values. TreeNet is especially adept at dealing with errors in the target variable, a type of data error that could be catastrophic for a neural net. TreeNet is resistant to overtraining and can be 10 to 100 times faster than a neural net. Finally, TreeNet is not troubled by hundreds or thousands of predictors.

What is the technology underlying TreeNet and how does it differ from boosting?

TreeNet is a proprietary methodology developed in 1997 by Stanford's Jerome Friedman, a co-author of CART®, the author of MARS™ and PRIM, and the inventor of Projection Pursuit regression, the methodology known as "stochastic gradient boosting"; the trade secrets of the technology are embedded exclusively in TreeNet. Others may eventually try to develop technology based on the public descriptions offered by Professor Friedman, but hundreds of technical details remain exclusive to Salford Systems.

What is the TreeNet track record?

TreeNet technology has been tested in a broad range of industrial and research settings and has demonstrated considerable benefits. In tests in which TreeNet was pitted against expert modeling teams using a variety of standard data mining tools, TreeNet was able to deliver results within a few hours comparable to or better than results that required months of hands-on development by expert data mining teams.

TreeNet was designated "Most Accurate" in the KDD Cup 2004 data mining competition (sponsored by the Association for Computing Machinery's data mining SIG). TreeNet also won first place in all four



competitive categories in the 2002 Duke University/NCR Teradata Churn modeling competition and numerous other competitions since.

How does TreeNet fit into the Salford Systems data mining solution?

The Salford Systems data mining solution rests on two groups of technologies: CART, MARS and PRIM for accurate easy-to-understand models and TreeNet and RandomForests for ultra-high performing, but potentially very complex, models interpreted via supporting graphical displays. Even in circumstances where interpretability and transparency are mandatory and a model must be expressible in the form of rules, TreeNet can serve a useful function by benchmarking the maximum achievable accuracy against which interpretable models can be compared.

Building a Regression Model in TreeNet

Creating a Sample Example Model Data Set

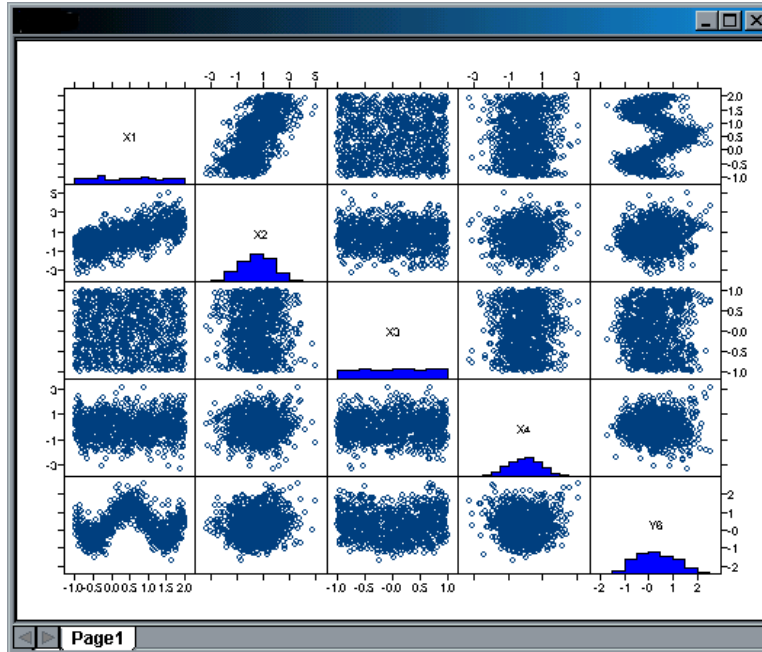
A sample data set SAMPLE.CSV is supplied as part of the TreeNet installation. It is located in the *Sample Data* folder.

The following table lists the major groups of variables in the data set.

Variable Names	Description
X1, ... , X10	Continuous predictors
Z1\$, Z2\$	Categorical predictors (character)
Y1	Continuous target
Y2	Binary target coded +1 and -1
Y3	Categorical target with 4 levels: 1, 2, 3, and 4
W	Weight variable
T	Learn/Test dummy indicator (0 – learn, 1 – test)

In our first run we will try to predict the continuous target Y1 using the continuous predictors X1–X10. We chose a regression problem for this first example because regression problems are somewhat simpler and produce fewer reports than classification problems. The scatter matrix for selected variables follows.






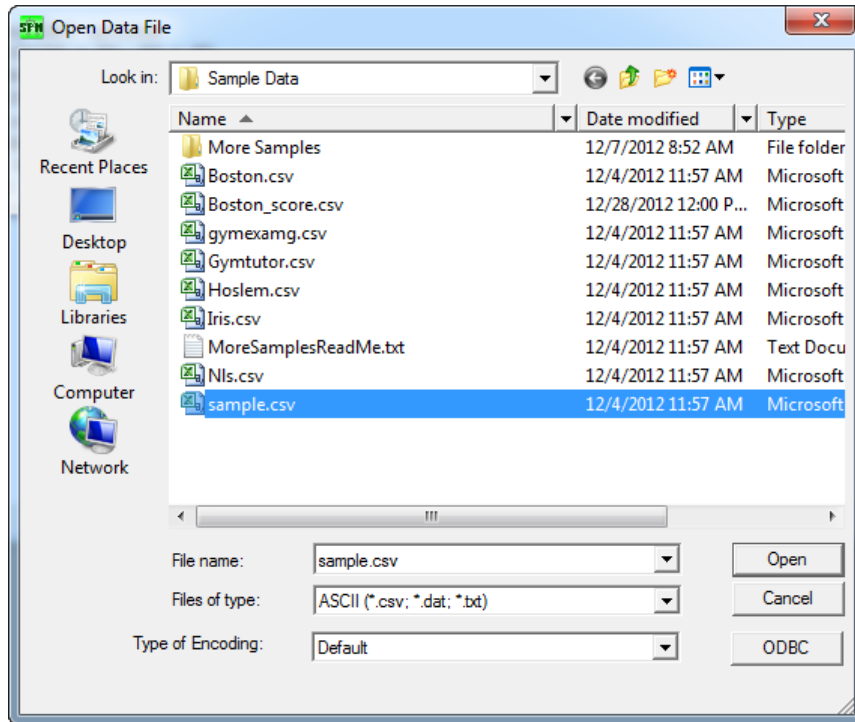
- ◆ The plots suggest only a strong nonlinear relationship between X1 and Y1. It is not clear whether there is any relationship between the remaining predictors and the target.
- ◆ An attempt to run a stepwise multiple linear regression of Y1 on X1–X10 selects five predictors, X1, X2, X6, X8, X9. Unfortunately, this model only has R-squared equal to .0179, making it practically useless. Later we will discover that this model fails to pick important predictors and at the same time mistakenly selects some irrelevant predictors.
- ✓ Exploiting possible periodic non-linearity in X1 as suggested by the corresponding plot may slightly improve the results, but the remaining relationships remain obscure.

Reading Data In

To open the input file `SAMPLE . CSV`:

- ◆ Select **Open>Data File...** from the **File** menu.
- ✓ You may simply click on the  button in the toolbar.
- ◆ Use the **Open Data File** dialog window to navigate into the *Sample Data* folder in the SPM installation directory.
- ◆ To speed up further access to the given location, you may want to set up the corresponding working directory first. (See the section titled “**Setting Up Working Directories**” earlier in this manual.)
- ◆ Choose **Delimited Text (*.csv,*.dat,*.txt)** in the **File of type:** selection box.
- ◆ Highlight `SAMPLE . CSV`.





- ◆ Click the **[Open]** button to load the data set into TreeNet.
- ☛ Make sure that the data set is not currently being accessed by another application. If it is, the data loading will fail. This is a common source of problems when dealing with Excel and ASCII files.
- ◆ In the resulting Activity window showing the basic info about the dataset, click the **[Model...]** button
- ✓ The activity window step can be skipped using the Options and Settings.

Setting up the Model

When the data set has been successfully opened, the **Model Setup** dialog window will appear. The Model Setup tabs are the primary control center for various aspects of the TreeNet model-building process. In this example, we will enter information into the **Model** tab and **Testing** tab only and then create a model using TreeNet default settings.

For this simple regression analysis, we complete the following simple steps:

- ◆ Choose **TreeNet** in the **Analysis Method** selection box
- ◆ In the **Model** tab, change the sort order to **File Order** in the **Sort:** selection box.
- ◆ Mark Y1 as the target by clicking the check box in the column labeled **Target**.
- ◆ Mark Z1\$, Z2\$, X1-X10 as the predictors by clicking the check boxes in the column labeled **Predictor**.
- ◆ Choose the **Regression** radio button in the section labeled **Analysis Type**.



Model Setup

Class Weights | Penalty | Lags | Battery | TN Advanced | Costs

Model | Categorical | Testing | Select Cases | TreeNet

Variable Selection

Variable Name	Target	Predictor	Categorical	Weight	Aux.
Y1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Y2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Y3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
W	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Z1\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Z2\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
X1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
X2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
X3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Sort: File Order

Select Predictors Select Cat. Select Aux.

Analysis Type

Classification

Regression

Unsupervised

Logistic Binary

Set Focus Class...

Target Variable

Y1

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

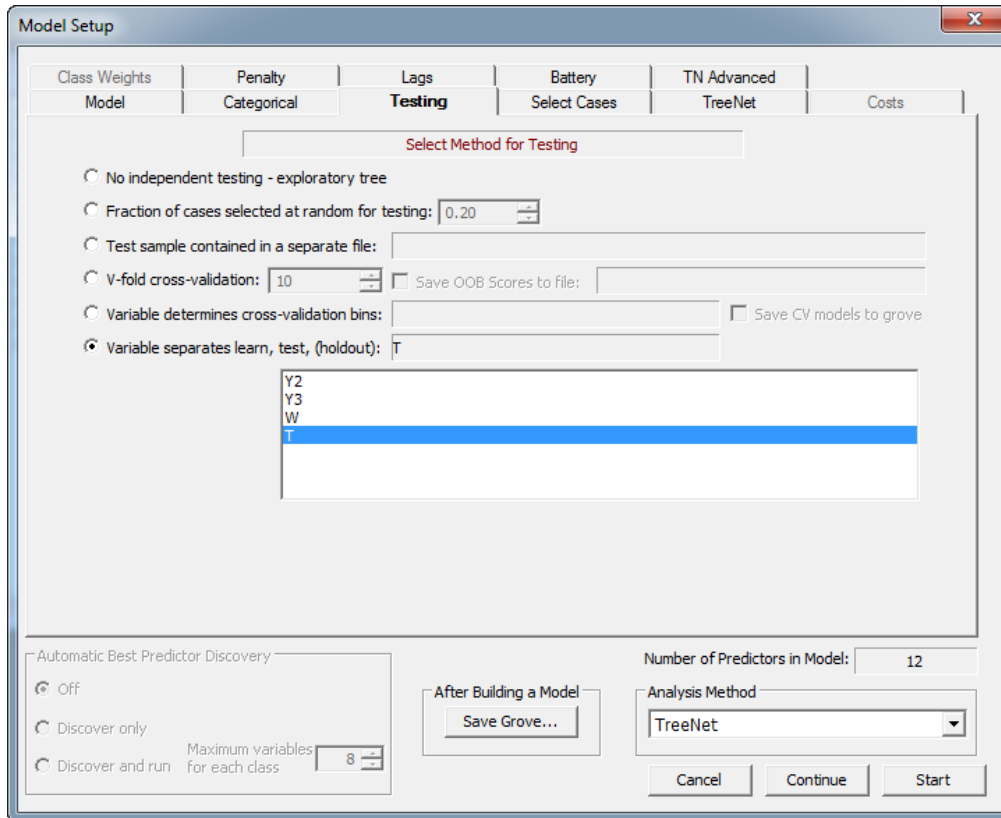
Analysis Method

TreeNet

Cancel Continue Start

- ◆ Select the **Testing** tab and choose the **Variable separates learn, test, (holdout)**: radio button. Choose variable T in the selection list box.





Model Setup

Class Weights | Penalty | Lags | Battery | TN Advanced | Costs

Model | Categorical | **Testing** | Select Cases | TreeNet

Select Method for Testing

No independent testing - exploratory tree
 Fraction of cases selected at random for testing: 0.20
 Test sample contained in a separate file:
 V-fold cross-validation: 10 Save OOB Scores to file:
 Variable determines cross-validation bins: Save CV models to grove
 Variable separates learn, test, (holdout): T

Y2
Y3
W
T

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 Method

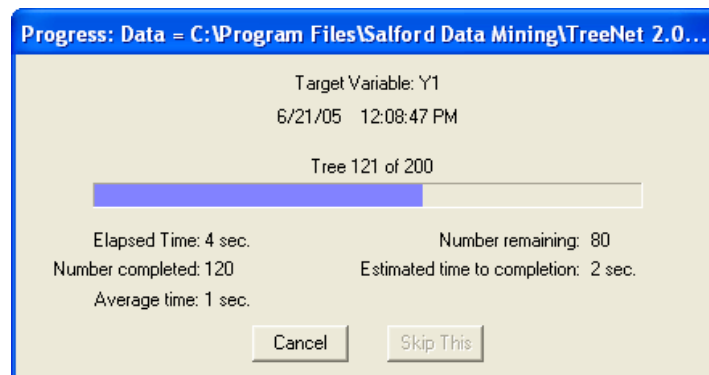
TreeNet

Cancel Continue Start

- ◆ The rest of the settings are left at their default values.

Running TreeNet

To begin the TreeNet analysis, click the **[Start]** button. The following progress indicator will appear on the screen while the model is being built, letting you know how much time the analysis should take and approximately how much time remains.



Progress: Data = C:\Program Files\Salford Data Mining\TreeNet 2.0...

Target Variable: Y1
6/21/05 12:08:47 PM

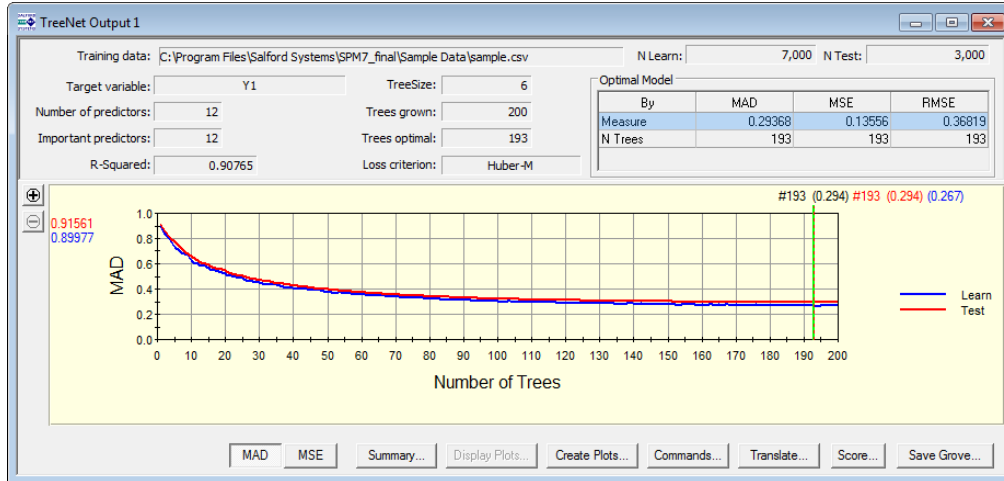
Tree 121 of 200

Elapsed Time: 4 sec. Number remaining: 80
Number completed: 120 Estimated time to completion: 2 sec.
Average time: 1 sec.

Cancel Skip This

Once the analysis is complete, text output will appear in the **Classic Output** window, and a new window, the **TreeNet Output**, opens.





The top portion reports various statistics about the current run. In our case all 12 predictors were used, individual trees had six terminal nodes, and 200 trees were grown, with the optimal model keeping 193 trees.

- ✓ Note that the R-squared of the optimal model on TEST data is 0.908, quite a bit better than an ordinary regression.

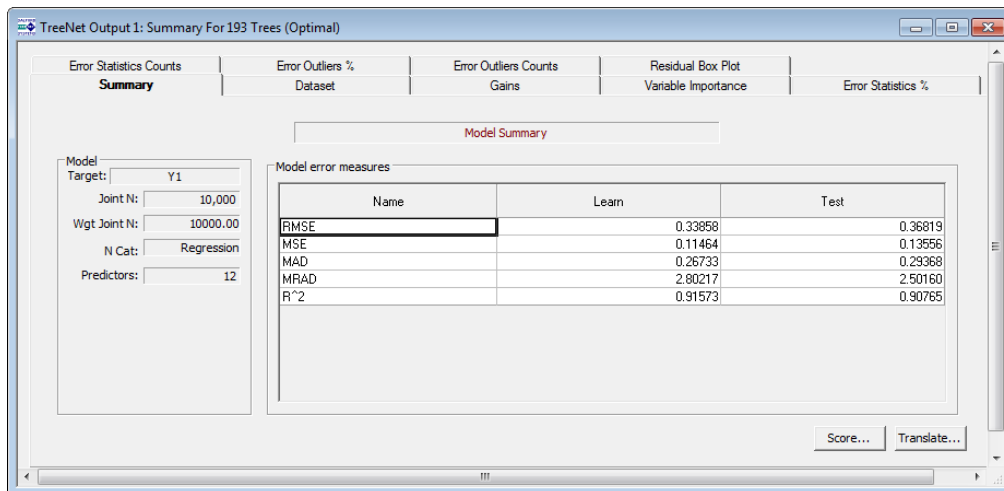
The bottom portion displays train and test mean absolute error curves as the TreeNet run progresses. The exact numeric minimum of the **Mean Absolute Deviation (MAD)** on the test data was attained at the 193-tree model highlighted by the green beam.

- ✓ To view the train and test mean squared error curves simply press the **[MSE]** button.

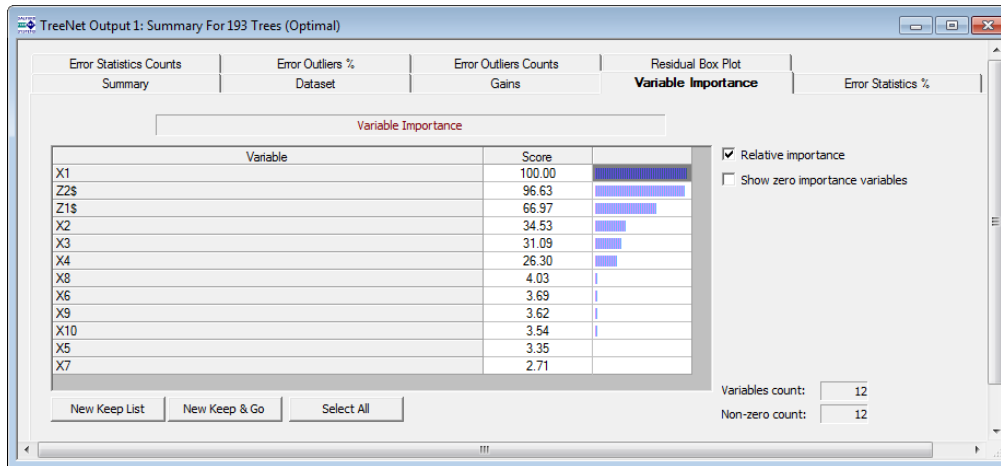
Viewing the TreeNet Results

The overall performance of the TreeNet regression model is summarized in model summary reports. To access the reports, click the **[Summary...]** button at the bottom of the TreeNet Output window.

As illustrated below, the **TreeNet Summary** contains summary model performance statistics as well as detailed information about gains, ROC, residuals, outliers, etc. This standard report is discussed in greater detail earlier in this manual.



Click on the **Variable Importance** tab to see the variable importance rankings. The scores reflect the contribution each variable makes in predicting the target variable. Clearly, variables Z1\$, Z2\$, X1, X2, X3, and X4 stand out as important.



Note that the TreeNet model suggests that the only six variables that matter are at the top of this report. With such a pattern we might try rerunning the model using only the clearly most important variables.

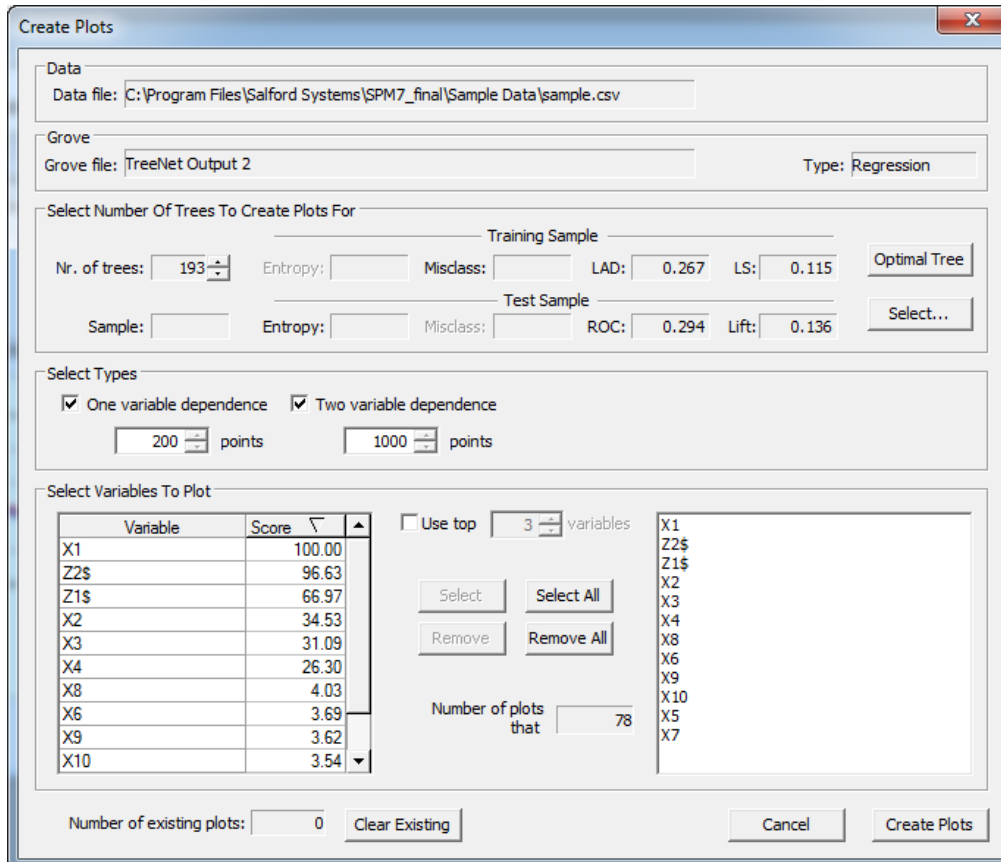
- ✓ The various summary reports, including Variable Importance rankings, R-squared, and gains charts apply to a SINGLE model. In the example above this is the model that consists of *exactly* 193 trees.
- ✓ Although the best model is the one normally recommended for further study there is nothing preventing you from looking at other models. In fact you can access results for any model in the TreeNet sequence of models, which means that you can review results for models with 1, 2, 3, 4, etc. trees.
- ✓ To economize on resources we display the overall performance for every size of model beginning with the one-tree model, but we only display detailed summary reports and graphs for select models. You can control how many such models are accompanied with details.

Creating and Viewing Dependency Plots

TreeNet has a powerful facility to generate partial dependency plots for any variable or pair of variables and any model size selected by the user. The principles behind the creation of plots are described later in this chapter, here we point to the basic steps needed to accomplish this task.

- ◆ Click the **[Create Plots...]** button in the **TreeNet Output** window.
- ✓ This operation is only available for the most recent TreeNet modeling run.





Create Plots

Data file: C:\Program Files\Salford Systems\SPM7_final\Sample Data\sample.csv

Grove file: TreeNet Output 2 Type: Regression

Select Number Of Trees To Create Plots For

Training Sample

Nr. of trees: 193 Entropy: Misclass: LAD: 0.267 LS: 0.115 Optimal Tree

Test Sample

Sample: Entropy: Misclass: ROC: 0.294 Lift: 0.136 Select...

Select Types

One variable dependence Two variable dependence

200 points 1000 points

Select Variables To Plot

Use top 3 variables

Variable	Score
X1	100.00
Z2\$	96.63
Z1\$	66.97
X2	34.53
X3	31.09
X4	26.30
X8	4.03
X6	3.69
X9	3.62
X10	3.54

Select Select All Remove Remove All

Number of plots that 78

X1 Z2\$ Z1\$ X2 X3 X4 X8 X6 X9 X10 X5 X7

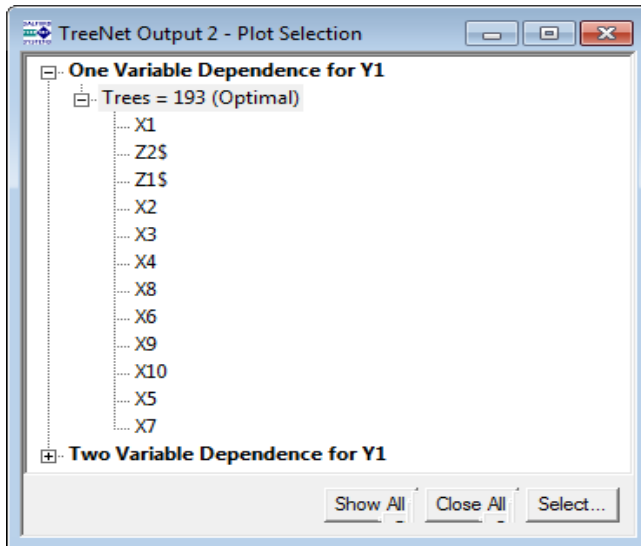
Number of existing plots: 0 Clear Existing Cancel Create Plots

- ◆ In the **Create Plots** window, press the **[Select All]** button to include all variables on the plots creation list.
- ◆ Put a check mark in the **One variable dependence** and **Two variable dependence** check boxes.
- ◆ Press the **[Create Plots]** button to initiate plots creation process.
- ✓ Note that by default plots will be created for the optimal model that has 193 trees, this can be easily changed by the user.

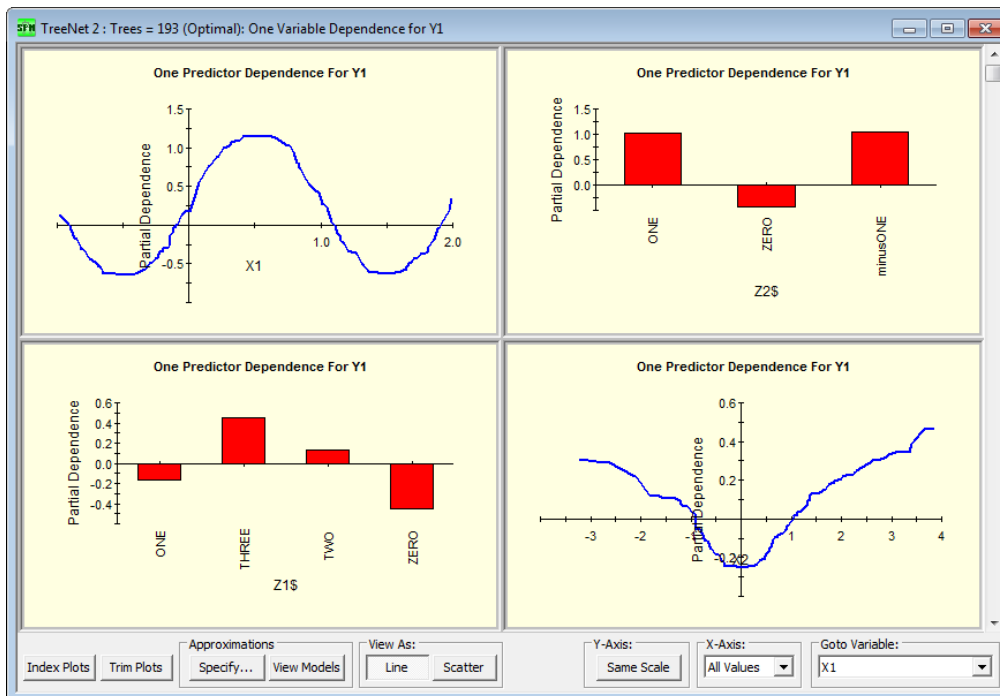
A progress bar appears at the bottom of the SPM main window indicating that the requested plots are being created. Upon successful completion of the process, the **Plots Selection** window will appear, it allows you to view different plots individually or all at once.

Clicking on the “+” signs expands any of the selection items.





Double-click on any of the selections to view individual plots. Alternatively, highlight a selection and click the **[Select...]** button. The **One-Variable Dependence** plots for the top important predictors are shown below.



You can request **[Create Plots...]** multiple times to generate collections of plots by variable and model size. They are automatically stored in the Plots Selection window which can be opened at any time by pressing the **[Display Plots...]** button at the bottom of the TreeNet Output window.

Press the **[Scatter]** button to view individual data points.

- ✓ By default, TreeNet automatically generates plots for the top three most important variables. You may change this number in the TreeNet tab of the Options window available through the Edit -> Options menu.



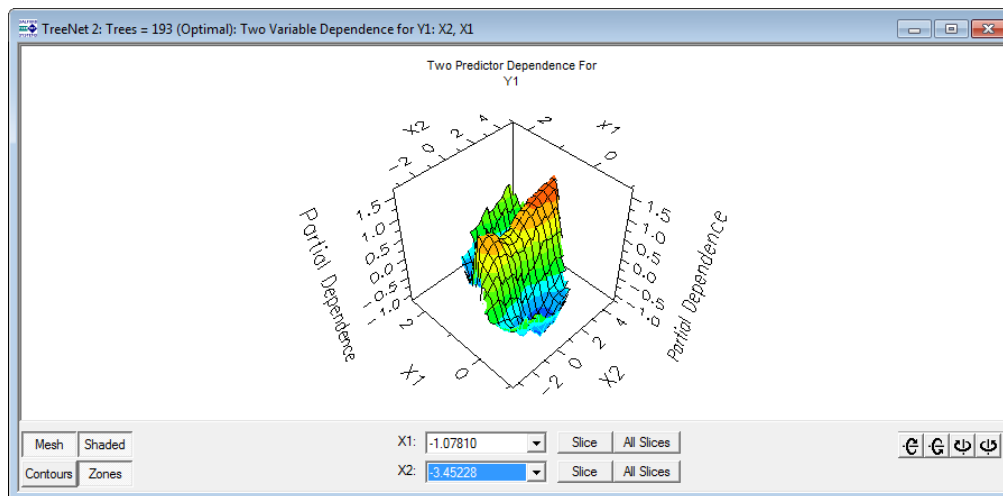
One-variable dependence plots are constructed by randomly selecting a subset of 200 data points and then averaging out all variables except for the variable in focus.


- ✓ If the underlying relationship is additive in nature (no interactions), the one-variable plots simply represent the main effects in our regression model.

Notice how the periodic structure in X1 has been uncovered.

- ✓ It is interesting to note that even though the underlying TreeNet model is by the nature of its construction piece-wise constant, having hundreds of overlapping trees creates nearly continuous dependency patterns.

When we have reason to suspect the presence of two-way interactions, it is beneficial to check the two-variable dependence plots. Two-variable plots for X1 and X2 are shown below.



The 3-D plots may be rotated using the  buttons to obtain a better view. In addition, you may use the **[Slice]** or **[All Slices]** buttons to view either single or multiple 2-D surface slices.

Additional 2-D and 3-D plots can be generated immediately after the current model has been built. The new plots can be requested for any model (not necessarily the optimal one) and can include any available predictor. Press the **[Create Plots...]** button to gain access to this powerful feature.

- ✓ The various summary reports, including Variable Importance rankings, R-squared, and gains charts and all the 2D and 3D graphs displayed above apply to a SINGLE model. In the example above this is the model that consists of *exactly* 193 trees.
- ✓ Occasionally you may have some reason for wanting to work with a model quite a bit smaller than the optimal model. We know of one large scale commercial user of TreeNet that opted to deploy a 30-tree model when an 800-tree model was optimal.
- ✓ If you decide to work with a cut-back model, you can view graphs and displays for the smaller model, but you may have to rerun the model and either stop its evolution early or explicitly request that details be saved for many sub-models.



Viewing Results for Non-optimal Trees

The optimal model(s) are always accompanied by a comprehensive set of diagnostics, reports and graphs. In addition, TreeNet includes complete reports for selected other models and partial reports for all models. In the case of regression, complete reports are always available for two potentially different models: the model with the number of trees yielding the optimal Least Squares, and the model containing the number of trees yielding the optimal LAD. A partial model summary is computed for all models but contains only core performance measures and the variable importance ranking. You may select any model by pressing the [Ctrl] + arrow keys.

- ✓ It is possible to increase the number of models with full summaries in the TreeNet tab of the Model Setup dialog.
- ✓ Requesting too many complete summaries might result in a significant program slowdown and may cause memory problems. Remember that each model might require 50 to several hundred graphs and if you save them all for 2000 models you can easily bog down your machine.



Building a Logistic Binary Regression Model in TreeNet

TreeNet “Binary Logistic” is the recommended way to model any binary outcome and is probably the most popular use of TreeNet. Although most analysts think of this type of model as CLASSIFICATION, in TreeNet we consider it as a special type of regression. This is because we want to produce a PROBABILITY of response for and not just a simple prediction of “YES” or “NO.” TreeNet’s ability to produce extremely accurate rankings of data from “most likely” to “least likely” underpins its strong performance in many data mining settings.

The first point to make is that this section is concerned exclusively with the binary or two-valued target variable. This variable is often coded as “0” and “1” or “YES” and “NO” or “RESPONSE” and “NONRESPONSE,” etc. Such models are by far the most common in data mining applications and are used to predict:

- ◆ Default in credit risk scoring
- ◆ Response in targeted marketing campaigns
- ◆ Fraud in insurance, credit card transactions, and loan applications
- ◆ Disease (or abnormal versus normal) in biomedical settings
- ✓ If you want to analyze such data then the logistic regression style of TreeNet is likely to give you the best results.

There is no such thing as a “best model” absent a context and an objective. When you run TreeNet, the software builds its model in stages, adding a tree at each stage in an effort to refine performance. Typically, the first tree yields a modest performance, the second tree improves on it, the third tree improves further, and so on. At some point in the process adding further trees does no good and may even do some harm! The entire set of trees grown is called the model sequence, and after the specified number of trees has been generated, we want to determine how many of the trees to keep. To assist in this process we track the following four performance criteria at each stage of the modeling sequence and identify the best number of trees to keep for each of the performance criteria:

- ◆ **Classification Accuracy:** this is based on a straightforward tally of how often the model tags a record correctly or incorrectly. If you intend to use the model to simply tag data as either “YES” or “NO” then this criterion will help you select the best model.
- ◆ **Area under the ROC curve:** this is the most commonly-used model criterion in machine learning and is a measure of overall model performance tied closely to the ability of the model to correctly RANK records from most likely to least likely to be a “1” or a “0.”
- ◆ **Average Log-Likelihood:** similar to ROC but emphasizes the probability interpretation of the model predictions. This is offered for technical users who may prefer this measure for theoretical reasons.
- ◆ **Lift in the top X-percentile:** focuses on the ability of the model to concentrate the responders or the “1”s in the highest ranking records. This criterion is useful if the model is intended to be used exclusively to identify, say, the top 1%, 5% or 10% of a data set. In this case, we can safely ignore model performance in the lower percentiles.
- ✓ It is not possible to tell in advance how many trees will be optimal.
- ✓ The best number of trees can vary quite a bit depending on which performance criterion you select.
- ✓ Typically, the best model for classification accuracy has many fewer trees than the best model for best probability estimates (Ave LL) or overall performance (ROC). Best lift is also usually accomplished with relatively few trees.



- ✓ The best model is normally determined by performance on test data or via cross validation. When no independent testing is specified (exploratory runs), the optimal model is always set to the largest number of trees.
- ✓ It is possible to manually select a non-optimal model of any size for graphical display, obtaining summary reports, scoring and translation.

In what follows we describe how to set up a logistic regression run in TreeNet, we also provide detailed comments on every setup option available to the end user.

Setting up a Logistic Regression Run

Logistic regression modeling in TreeNet is controlled from the **Model Setup** dialog. This window automatically appears after a data file has been successfully accessed. You can also request the **Model**

Setup dialog using the **Model–Construct Model...** menu or by clicking the  toolbar icon.

☛ A data set must be currently open in TreeNet for this operation to work.

Model Setup – Model Tab

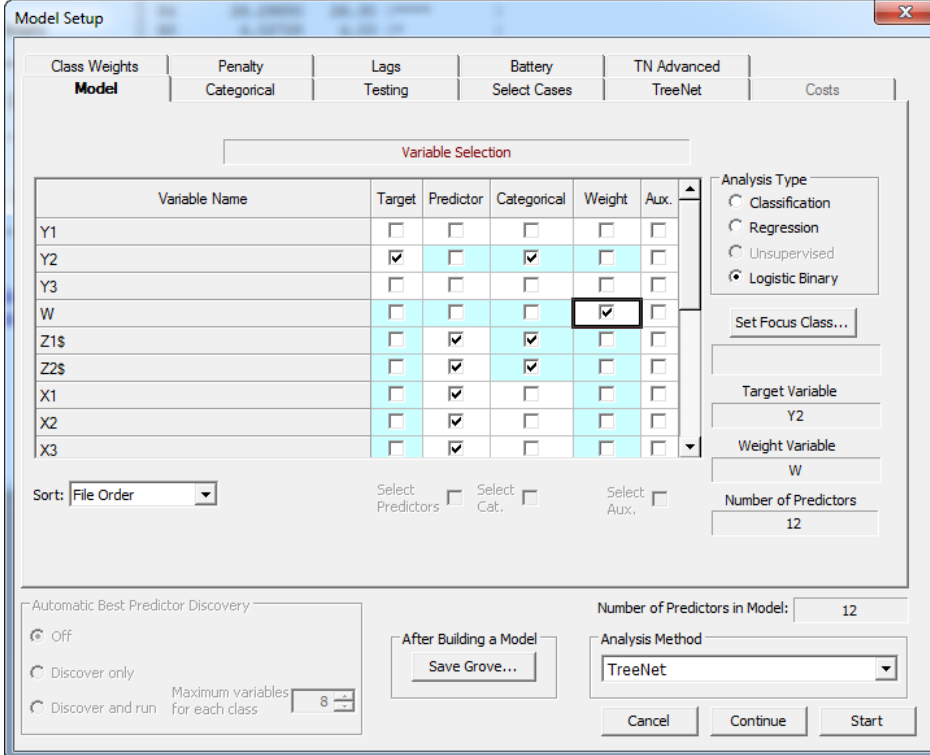
In the **Model** tab we specify our core model variables. Variables specified in this dialog include a target (dependent), predictors (independent), categorical designation, and optionally a single case-weight.

The variables and tree type for our model are selected as follows:

- ◆ Choose the **Logistic Binary** radio button in the section titled **Analysis Type**.
- ◆ Select the target variable Y2 by placing a check mark in the corresponding variable check box under the **Target** column.
- ✓ You may change the **Sort:** order to **alphabetically** when it is convenient.
- ◆ Select potential predictors by placing check marks in the corresponding variable's check boxes under the **Predictor** column.
- ✓ You may highlight multiple variables first by using a left mouse click and/or the <Shift> or <Ctrl> keys while selecting in the Variable Name column. Once selected you may either check or uncheck the corresponding check marks all at once for the selected variables by using the check box at the bottom of the appropriate column.
- ◆ Optionally, check a weight variable using a check box under the **Weight** column.
- ✓ Only one weight variable can be active at a time.
- ◆ Indicate which predictors are categorical by placing check marks under the **Categorical** column for the corresponding variables.

The **Model** tab will now appear as follows:





Variable Name	Target	Predictor	Categorical	Weight	Aux.
Y1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Y2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Y3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
W	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Z1\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Z2\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
X1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
X2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
X3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- ✓ Correctly declaring variables as categorical is less important for TreeNet than for CART or other modeling methods. TreeNet is capable of uncovering the true relationship between the target and a categorical predictor whether the declaration is made or not. However, some graphs will be much easier to read if the declarations are made. For high level categorical predictors, however, it may be advantageous to treat such predictors as numeric.
- ⚠ All character variables (variables ending in \$) will be automatically checked as categorical and cannot be unchecked.
- ⚠ Using high-level categorical variables may significantly increase the running time and impose severe memory requirements. Make sure that a continuous variable has not been accidentally checked as categorical.
- ✓ Consider letting TreeNet treat a high-level categorical variable as a continuous predictor if it has more than about 30 levels. This is unlikely to hurt the model and may actually improve results quite a bit. This is possible only if the HLC is coded as numeric rather than text.
- ✓ The **Aux.** column is not used by the TreeNet engine.
- ✓ Press the [Set Focus Class...] to override the default value used by TreeNet as the focus target class. This will not change the essence of the modeling itself but may simplify interpretation of the graphs and various reports.

Model Setup – Testing Tab

The Testing tab specifies how the resulting TreeNet model will be evaluated. The six following testing modes are available in TreeNet.

- ◆ **No independent testing – exploratory tree.** In this mode no external evaluation will be performed. The entire data set will be used for training; therefore, the final results will include only training data and the best model is almost invariably the largest.



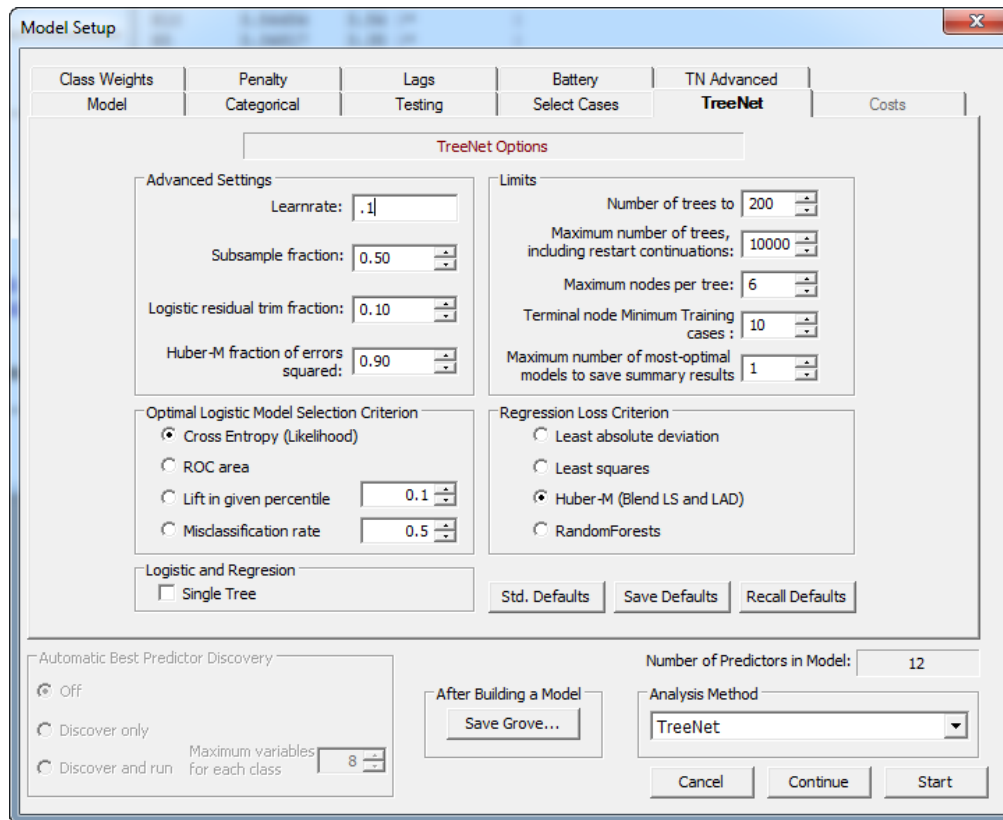
- ◆ **Fraction of cases selected at random for testing.** The specified fraction of data will be set aside for testing. Final results will be reported for both train and test data.
- ◆ **Test sample contained in a separate file.** The data for testing will be contained in the specified file. Use this mode when you want to use one data file for training and a separate data file for testing. When this mode is activated, a file selection dialog window will appear.
- ◆ **V-fold Cross Validation.** Usually used with small datasets when one cannot afford to reserve some data for testing. The entire dataset is used for learning purposes, and then is partitioned into ten bins. At each fold in 10-fold CV, nine bins are used as a training set and the remaining bin is used as a test set. After all 10 folds are completed, the test results from each fold are averaged to get a fair test estimate of the all-data model performance.
- ◆ **Variable determines cross-validation samples.** Same as the above option but the user has full control over bin creation by introducing a special variable with integer values 1 through 10 to mark the bins. You can opt for a different number of folds.
- ✓ Use “Cross Validation” when the supply of data is limited.
- * Use extreme care when creating your own CV bins. Each bin should be of the same size and balanced on the target variable.
- * 10-fold cross-validation runs are on average 10 times slower than using a test sample. For small data sets you may not even notice the difference but for large data sets using cross validation will substantially increase your run times.
- ◆ **Variable separates learn, test, (holdout) samples.** This is the most flexible mode. All you need is a variable coded as 0, 1, or 2. When its value is 0, the corresponding observation will go into the learn data; when the value is 1, the observation will go into the test data; otherwise observation will not be included into any partition and assumed to be reserved as a holdout sample.
- ✓ Use the indicator variable mode when you want complete control over which part of the data is used for testing. You may also create multiple indicator variables in the same data set to experiment with different random seeds or learn/test ratios.

In our example we are using **A variable separates learn and test samples**. The sample data set already has a test indicator variable, called T.

Model Setup – TreeNet Tab

The **TreeNet** tab contains important parameter settings used in the model-building process. This setup dialog allows the user to control multiple aspects of the TreeNet modeling process. Each of the parameters displayed in the following illustration is discussed in detail below.





Learn rate

 `TREENET LEARNRATE=<x | AUTO>`

Controls the rate at which the model is updated after each training stage. This is also referred to as “shrinkage” in Friedman’s original articles.

The default is AUTO, and the allowable range is 0.0001 to 1.0 inclusive. AUTO is calculated as follows:

$$AUTO \text{ value} = \max(0.01, 0.1 * \min(1, nl/10000))$$

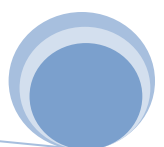
where *nl* = number of LEARN records.

- This default uses very slow learn rates for small data sets and uses 0.10 for all data sets with more than 10,000 records. We strongly recommend that you experiment with different learn rates, especially rates slower than the default for larger data sets.
- High learn rates and especially values close to 1.0 typically result in overfit models with poor performance.
- Values much smaller than .01 significantly slow down the learning process and might be reserved for overnight runs.

Subsample fraction

 `TREENET SUBSAMPLE=<x>`

This setting controls the proportion of data used for learning at each training cycle. A new random draw from the learn sample is conducted at each cycle (sampling without replacement). This not only speeds




up the modeling time, but also guards against overfitting and explains occasional minor local increases in the learn error curve as a TreeNet run progresses.

The default is 0.5 (half of the learn sample is used at each modeling iteration), and the allowable range is 0.01 to 1.0 inclusive.

- ✓ It may be necessary to use values greater than the .5 default with small training files. Unlike the learn rate, using a sampling rate of 1.0 is not necessarily catastrophic, but values less than 1.0 are still strongly recommended.
- ✓ You may need to experiment to determine the best rate.
- ✓ Sampling rates that are too small can hurt accuracy substantially while yielding no benefits other than speed.

Logistic residual trim fraction

 TREENET INFLUENCE=<x>

Influence trimming is a form of data selection designed to focus the TreeNet learning process on the most important data records. It is the mechanism by which suspicious data are ignored. There are two types of data records that are dropped due to the influence trimming: those that are far from the decision boundary and classified correctly and those that are far from the decision boundary and misclassified. The former have very small logistic residuals, contribute next to nothing to the model refinements during subsequent iterations and therefore can be safely ignored. The latter have very large logistic residuals, which could be an indication of being outliers or otherwise problematic records with the potential to severely distort model evolution and thus needed to be ignored too.

- ✓ If you have no concerns about the quality of your data you might consider turning this feature off by setting the factor to 0.
- ✓ It is always worth experimenting with a few values of this factor to see what works best. We recommend settings of 0, 0.1, and even 0.2.


The Influence trimming factor can vary between 0.0 and 0.2, with larger values generally having larger effects.

M-Regression breakdown parameter

 TREENET BREAKDOWN=<x>

This setting is ignored in binary logistic runs.

Optimal Logistic Model Selection Criterion

 TREENET OPTIMAL=[CXE | ROC | LIFT | CLASS]

By selecting “Logistic Regression” you select a specific way to generate the individual TreeNet trees but you still have four ways to evaluate model performance. See the discussion on Best Models above for details.


Note that the “Lift in the given percentile” and “Misclassification rate” measures also require additional user’s input, namely the percentile and the threshold (both expressed as fractions between 0 and 1).



The sequence of trees generated by TreeNet during the model-building phase is determined by control parameters such as the number of nodes and the learn rate. The model selection criterion determines the size of the model extracted from this sequence.

- ✓ Changing the model selection criterion does not change the actual model sequence which is constructed internally by working with the logistic loss on the learn sample. The optimal model size is then found for each of the four available performance criteria using the test sample or cross-validation. Optimal lift models will generally be much smaller than the optimal CXE models in the same sequence.

Number of trees to use

 TREENET TREES=<n>

A TreeNet model is essentially a collection of trees of a fixed size. New trees are created as the modeling process progresses. This option specifies how many trees should be grown before the modeling process ends.


- ✓ You should expect to grow hundreds or even thousands of trees to get a maximum performance model.
- ✓ The default setting for 200 trees is designed to facilitate interactive model selection and provide very rapid feedback. Once you have a good feel for your data and how you want to tackle your modeling effort, be prepared to allow for 500, 1000, 2000 or more trees.
- ✓ Be sure to use a slow learn rate when growing many trees.
- ✓ If the “optimal” model contains close to the maximum number of trees allowed (e.g., more than $\frac{3}{4}$ of the maximum) consider growing more trees.

Maximum number of trees including restart continuations

 TREENET MAXTREES=<n>

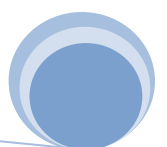
This is an obsolete control that will be eliminated in future versions. You will rarely need to change this setting, unless you want to experiment with more than 10000 trees for model building.

Maximum nodes per tree

 TREENET NODES=<n>

This setting controls the fixed size of each individual tree used in the TreeNet model. At each step, a fixed-size tree will be grown to correct the current prediction error.

- ✓ You can grow TreeNet models made up of two-node trees. These tiny trees (also known as “stumps”) can be surprisingly powerful and are well worth experimenting with.
- ✓ Two-node trees contain only a single variable and a single split. Because only one variable is ever involved in a model update, the models cannot detect interactions and thus can be described as main-effects additive models.
- ☛ In the presence of missing values in the learn sample, TreeNet may choose to build more nodes in a tree to accommodate missing value pre-splits.
- ☛ TreeNet may also choose to build smaller trees when the larger trees can’t be physically constructed due to sample size or other limitations.



- More than two nodes are required to detect interactions and the default six-node tree appears to do an excellent job.
- Nothing prevents you from requesting trees with quite a few nodes (e.g., more than 12) but doing so is likely to undermine TreeNet's slow-learning strategy. Large trees can also place unnecessary burdens on your computer's memory resources.

Terminal node Minimum Training cases

 TREENET MINCHILD=<n>

This setting controls how small individual terminal nodes are allowed to be. In our example, the value setting of 10 indicates that no terminal nodes with fewer than 10 observations are allowed in the model.

Setting this control to larger values may result to smoother models; however, past a certain point the model may start experiencing the loss of accuracy. The best values for this control are usually determined experimentally.

- ✓ When working with small training samples it may be vital to lower this setting to five or even three.

Maximum number of most-optimal models to save summary results


 TREENET FR=<n>

If you generate a model with 1000 trees you actually have access to 1000 different models beginning with a model containing one tree, then a model containing two trees, etc. Each of these models potentially comes with a complete set of summary reports and graphs. To conserve resources we store details for only the best models but give you the option of producing extensive reports on as many different models as you would like to review.

This setting will control how many models receive complete summary results reported and saved. When this setting is set to one, the full summary results will only be saved for the single best model for each of the four optimality criteria (two for regression). Settings above one will include additional next best optimal models uniformly distributed among the available optimality criteria.

- Setting this parameter to a large value will increase the running time for your models and possibly cause you to run out of memory.
- We recommend keeping this setting well below 10 for routine modeling.

Regression Loss Criterion

 TREENET LOSS=[LAD | LS | HUBER | RF]

This setting is ignored in the binary logistic regression runs.



Defaults buttons

The group of buttons in the lower right corner of the TreeNet tab allows the user to specify new defaults for any subsequent TreeNet run. The defaults are saved in the TreeNet initialization file and persist from session to session.

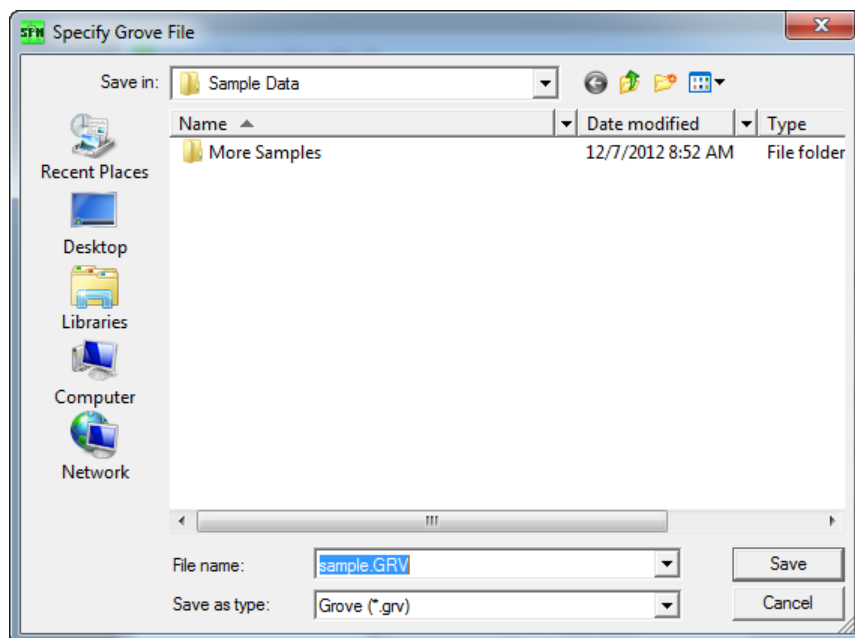
- ◆ **[Std. Defaults]** – press this button to restore the original “factory” settings.
- ◆ **[Save Defaults]** – press this button to save the current settings as the new defaults.
- ◆ **[Recall Defaults]** – press this button to recall previously-saved defaults.

For example, you might decide that your typical TreeNet run should use a 0.01 learn rate, 1000 trees, two nodes (to force additive models), and the least squares loss criterion. Just set these parameters and press **[Save Defaults]**. In future sessions, TreeNet will automatically assume the new settings unless you manually override them.

Save Grove

Any time during model setup, the **[Save Grove...]** button can be used to save the currently active TreeNet model to a grove file. The grove file, a binary file that contains all the information about the TreeNet model, must be saved if you want to apply the model to a new data set or translate the model into one of the supported languages at a later time.

After the **[Save Grove...]** button is pressed, the **Specify Grove File** dialog window appears:




Type in the name of the grove file to be created and click the **[Save]** button.

- ✓ The grove file will be created on your hard drive only after the model-building process is completely finished, this means after plots have been created using the interactive GUI controls.
- ✓ You can also save the grove file from the Results window (see below) after the run is finished.



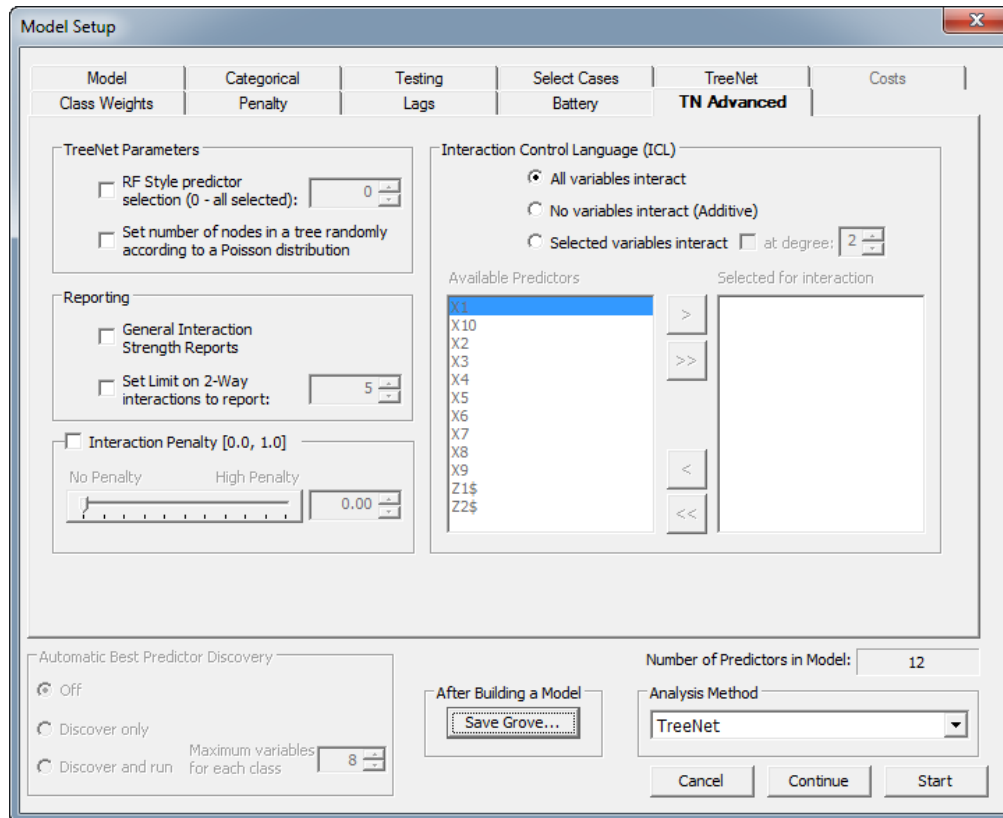
Single Tree

 `TREENET ONETREE=YES TREES=1`

TreeNet has a special mode of running to build a single tree. The resulting tree is displayed in a Navigator window as if it were built by the CART engine. See later section for more details.

Model Setup – TN Advanced Tab


This tab provides access to some of the more advanced features of the TreeNet engine.



The screenshot shows the 'Model Setup' dialog box with the 'TN Advanced' tab selected. The 'Interaction Control Language (ICL)' section is expanded, showing three radio button options: 'All variables interact' (selected), 'No variables interact (Additive)', and 'Selected variables interact' (with a degree of 2). Below this are two lists: 'Available Predictors' (X1, X10, X2, X3, X4, X5, X6, X7, X8, X9, Z1\$, Z2\$) and 'Selected for interaction' (empty). The 'Number of Predictors in Model' is set to 12. The 'Analysis Method' is set to 'TreeNet'. Other sections include 'TreeNet Parameters' (RF Style predictor selection, Set number of nodes), 'Reporting' (General Interaction Strength Reports, Set Limit on 2-Way interactions), and 'Automatic Best Predictor Discovery' (Off, Discover only, Discover and run).

Most of the controls deal with the interaction handling and reporting components of TreeNet and will be described later in a separate section. Here we focus on the remaining controls that might be of interest.

RF Style predictor selection

 `TREENET PREDS=<n>`

In the original TreeNet approach, all predictors would be tried as potential splitters in each node as a tree is being built. This is identical in spirit to the conventional CART process. However, in developing the Random Forests algorithm Leo Breiman has demonstrated that additional benefits may be acquired if one uses a randomly selected subset of predictors as potential splitters during the tree construction process. A new subset of predictors is randomly selected at each node and then used in searching for the optimal split value. This further reduces the mutual correlation structure of the ensemble and improves its predictive power.



The RF Style predictor control incorporates this approach into the TreeNet algorithm. One can now set the number of predictors to be sampled at each node during TreeNet model building and explore its impact on the resulting performance.

- ✓ Breiman suggested the square root of the total number of predictors as a good starting point for this control.
- ✓ By default, this control is set to 0 which allows all predictors to be sampled thus restoring the original TreeNet approach.

Random Poisson Number of Nodes

 TREENET RNODES=[YES|NO]

Jerome Friedman recently suggested that fixing the number of nodes for each tree in the TreeNet process may be too rigid in some contexts. For example, the RuleLearner methodology aimed at extracting useful rule-sets from a TreeNet model (described in a later chapter) may benefit from trees having varying size as the TreeNet model develops.

When the box is checked, TreeNet will make a Poisson draw with the mean value taken from the “Maximum nodes per tree” control from the TreeNet tab. The resulting value will be used as the actual requested size of the next tree to be built. Thus as the TreeNet process develops, each tree will have varying size in terms of the number of terminal nodes.

Model Setup – Class Weights Tab

 CW [BALANCE|UNIT|SPECIFY]

The original intent behind the concept of class weights is to allow powerful modeling controls similar to PRIORS controls in the CART engine. Internally, TreeNet trees are optimized for speed and efficiency and do not directly support the prior probabilities concept. This is where class weights become handy.

Class weights allow you to specify weights for each member of a class. For example, an observation from class 1 could count twice as much as an observation from class 0. Note that this is as if one doubled (ignoring some fine differences) the prior probability of class 1. Class weights are distinct from individual case or record weights and both types of weights can be used at the same time.

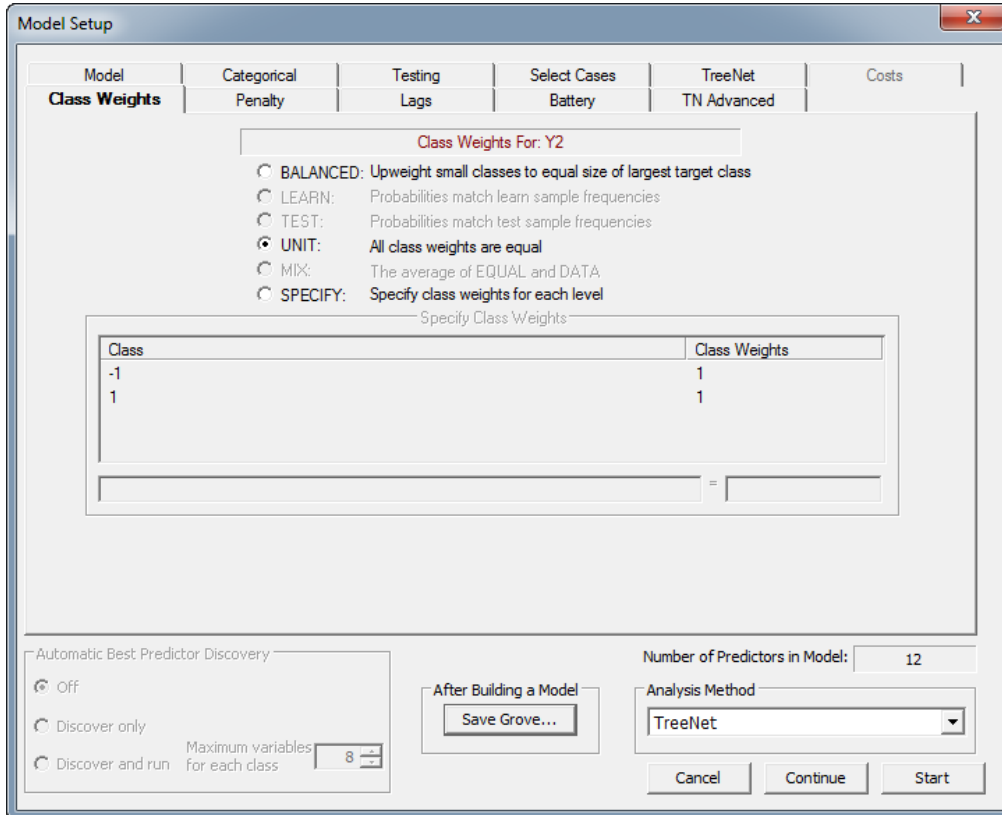
The following three class weights options for the logistic binary model are available:

- ◆ **BALANCED** - As the name implies, this setting is intended to rebalance unequal class sizes. Automatic reweighting ensures that the sum of all weights in each class are equal, eliminating any need for manual balancing of classes via record selection or definition of weights.
 - ✓ This is equivalent to the PRIORS EQUAL setting of CART.
- ◆ **UNIT** - This setting takes the relative sizes of the two classes as given and does not attempt to compensate by adjusting weights for the smaller class. This is the default setting and is recommended for the binary logistic regression.
 - ✓ This is equivalent to PRIORS LEARN setting of CART.
- ◆ **SPECIFY** - Lets the user specify class weights. This is the most flexible option. You can set any positive values for each class.



- While class weights can influence how a TreeNet model evolves and how it will perform in prediction, these weights will not influence any reports involving record or class counts. The class weights are “virtual” and are used to influence the model-building process.

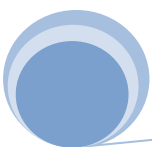
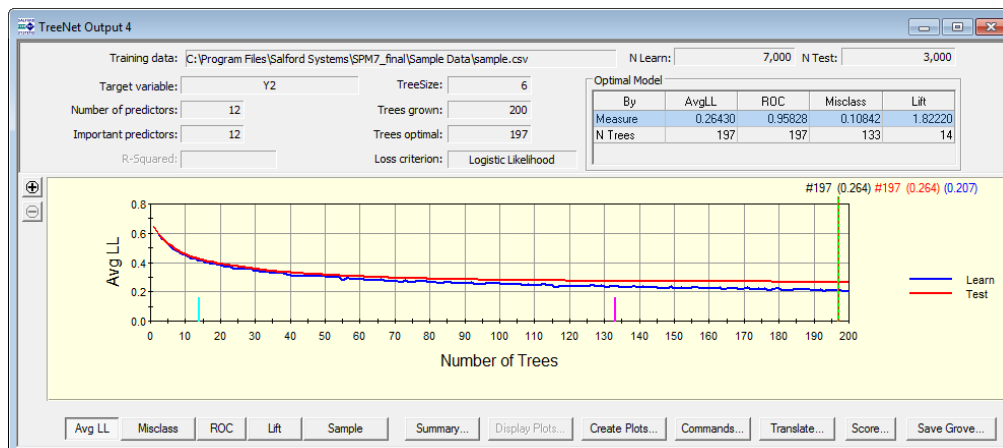
In most TreeNet runs, we recommend using UNIT class weights. This simplifies reading the output and usually produces superior results.



Press the **[Start]** button to build the binary logistic model.

Viewing Binary Logistic TreeNet Results

A TreeNet modeling run starts after the **[Start]** button is pressed. A progress report window appears. After the trees have been grown, the **TreeNet Output** window automatically appears:



Information available in the TreeNet results window includes the following:

- ◆ General information about run settings is displayed in the upper section of the window: training data file, target variable, number of predictors, requested tree size, total number of trees to be grown, type of loss function.
- ◆ TreeNet also reports the optimal model sizes at which the test set error criteria are optimized. For the current run, the optimal likelihood and ROC models are attained when 197 trees are grown, while the 133-tree model has the best classification performance and the 14-tree model has the best lift in the top 10%.
- ◆ The lower part of the window displays the run history: the tree number on the horizontal axis and train/test error profiles on the vertical axis. You can switch among the four error profiles using the **[Ave LL]**, **[Misclass]**, **[ROC]**, and **[Lift]** buttons. Note that the test error curve lies slightly above the learn error curve and that they are in close agreement showing little to no overfitting.
- ◆ A vertical green beam marks the optimal model with respect to the user-specified optimality criterion (currently pressed button). The actual number of trees and the corresponding value of the criterion are reported just above the graph. In our example, the optimal model has a 0.264 average negative log-likelihood.
- ◆ An additional profile can be requested by pressing the **[Sample]** button. The resulting curve shows the actual percentage of learn data used at each iteration which starts at the user specified sampling rate but may decrease due to influence trimming (see earlier) as the model progresses.
- ✓ The profile always starts at user specified sampling rate (0.5 by default) but may decrease due to influence trimming (see earlier) as the model progresses.

Summary Reports – General

The **TreeNet Output: Summary** window is activated when the **[Summary...]** button is pressed in the **TreeNet Output** window. TreeNet provides variable importance information for models of all sizes and gains, misclassification, prediction success information and other standard tables for a selected subset of model sizes, which always includes the optimal models corresponding to each of the four evaluation criteria.

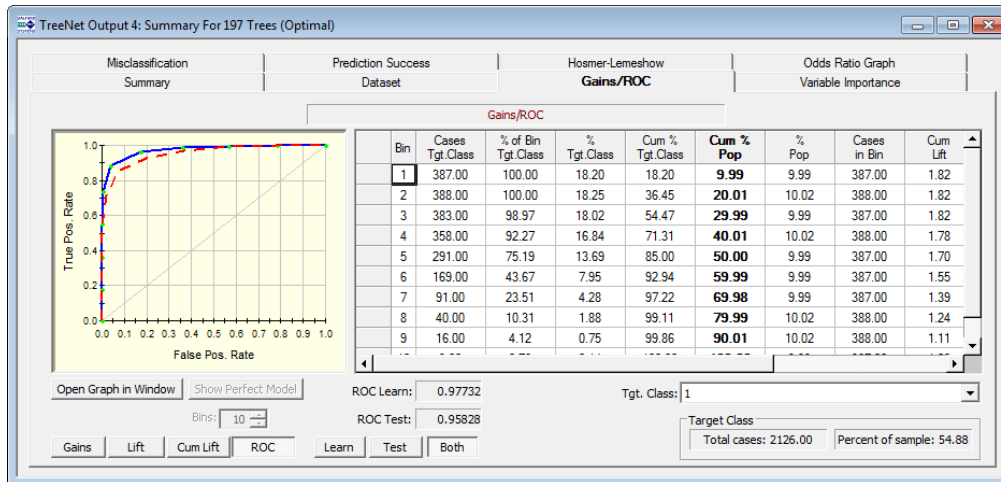
- ✓ You can specify the number of models for which the full results are available in the TreeNet tab of the Model Setup window.

You can select alternative models by using the arrow keys or making a left mouse click. Models without the full summary (variable importance only plus basic data info) will appear as dashed lines. You can jump directly among the models with the full summary available by using the combination of **<Ctrl>** key and the corresponding arrow keys or mouse clicks.

Summary Reports – Gains/ROC

The **Gains/ROC** tab is located in the **TreeNet Output: Summary** window. The gains chart and the corresponding table are often used to judge the overall performance of a model on a given data set. It appears as follows:





The following table defines the columns of this report:

Gains Tables

TARGET denotes the actual response (0 or 1, YES or NO, etc.) and it must be known for each case.

RESPONSE denotes the scores predicted by the given model.

K denotes the predetermined number of bins.

- Sort the data by descending **RESPONSE**.
- Divide the data into **K** bins such that each bin has an approximately equal number of weighted cases.

- **% of Bin Class 1** – the percentage of 1s in the given bin.

- **% Class 1** – the weighted number of class 1 cases in the bin divided by the total weighted count of class 1 cases.

- **Cum % Class 1** – the simple cumulate of the **% Class 1** column. This is also known as **sensitivity**, assuming that the classification threshold is set at the corresponding bin boundary.

- **Cases in Bin** – the weighted number of cases in the given bin.

- **% Pop** – the weighted number of cases in the given bin divided by the total weighted number of cases.

- **Cum % Pop** – the simple cumulate of the **% Pop** column.

- **Cum Lift** – **Cum % Class 1** divided by **Cum % Pop**.

- **Lift Pop** – **% of Bin Class 1** divided by **% Pop**.

The graph to the left displays:

- ◆ When the [**Gains**] button is pressed – **Cum % Class 1** versus **Cum % Pop**.
- ◆ When the [**Lift**] button is pressed – **Lift Pop** versus **Cum % Pop**.
- ◆ When the [**Cum. Lift**] button is pressed – **Cum Lift** versus **Cum % Pop**.
- ◆ When the [**ROC**] button is pressed – **Cum % Class 1** versus (1 – **specificity**) (The specificity is defined similarly to the sensitivity but with respect to the class 0.)

The number of bins can be changed using the control.

- ✓ You can change the number of bins only when the [**Learn**] or [**Test**] buttons are pressed. This allows extra flexibility in displaying the results over different data partitions.



The class in focus (the class that is referred to as 1s) can be selected using the **Tgt. Class** selection box.

- ✓ The ROC curve, while similar to the gains curve, has a number of theoretical advantages, especially when considering the model performance on the dominant class (in which case the gains curve may degenerate into the 45-degree line).

When the test data are available, both learn and test gains and ROC charts can be displayed using the **[Learn]**, **[Test]**, or **[Both]** buttons.

One may use the following simple interpretation of gains. Assume that a scoring model is built to predict potential responders to a certain mail order. If every member of the population is given the offer, all responders would be targeted. Now assume that we would actually like to send mail offers to only 10% of the population. When no model is available and offers are mailed purely randomly, we should expect to collect only about $1/10^{\text{th}}$ of the responders. In the presence of a model, instead of random mailing, one could send offers to the 10% of the population in the top bin (the highest scores). In this case the expected percent of covered responders would be **Lift Pop** times $1/10^{\text{th}}$. The interpretation is similar for the other bins and cumulative lift.

- ✓ This is a special case (0/1 response) of the regression gains described earlier on.
- ✓ Gains/ROC based on the test data are usually more realistic than gains/ROC based on the training data.
- ✓ Test gains/ROC are not available when independent testing is disabled (Test tab in the Model Setup).

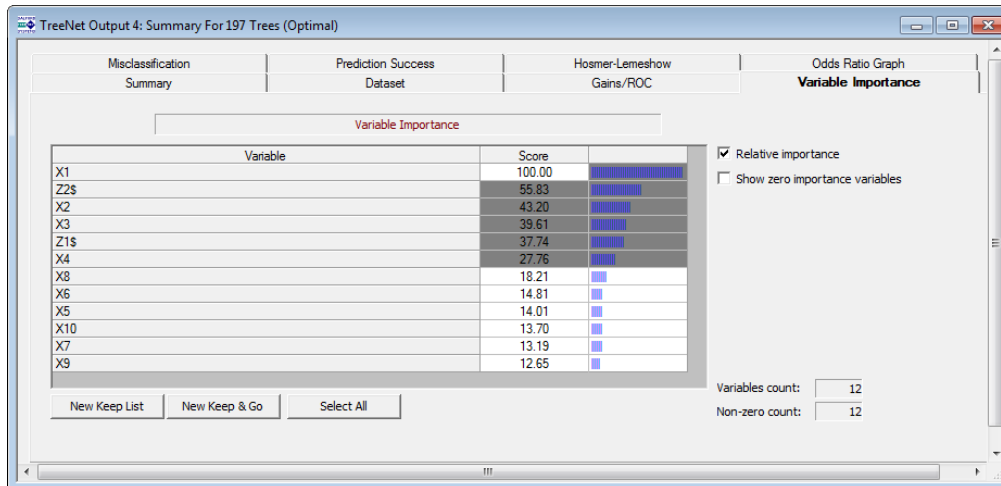
Summary Reports – Variable Importance

The raw variable importance score is computed as the cumulative sum of improvements of all splits associated with the given variable across all trees up to a specific model size. Note that smaller models will typically involve fewer variables, thus limiting the number of variables with non-zero importance scores, whereas larger models will often utilize a large number of predictors.

The relative importance **Score**, expressed on a scale of 0 to 100, simply rescales the importances so that the most important variable always gets a score of 100. All other variables are rescaled to reflect their importance relative to the most important. In our example, the continuous predictors X1, X2, X3, X4 and the categorical predictors Z1\$, Z2\$ were clearly identified as important. The remaining predictors are reported to have rather small contributions and most probably reflect random noise patterns.

- ✓ This conclusion is further supported by the analysis of contribution plots described in a later section.





- ✓ The rows in the table can be re-sorted differently by clicking on the corresponding column header. Thus, you can sort the variables by name and change the direction of the sort (ascending or descending).

Note that one can highlight a contiguous list of variables in the table and then press the **[New Keep List]** button to open a Notepad window containing the list of selected variables in the command line text form. Alternatively, press the **[New Keep & Go]** button to rebuild TreeNet model using the selected list of variables as predictors. This facilitates manual variable selection approaches based on variable importance.

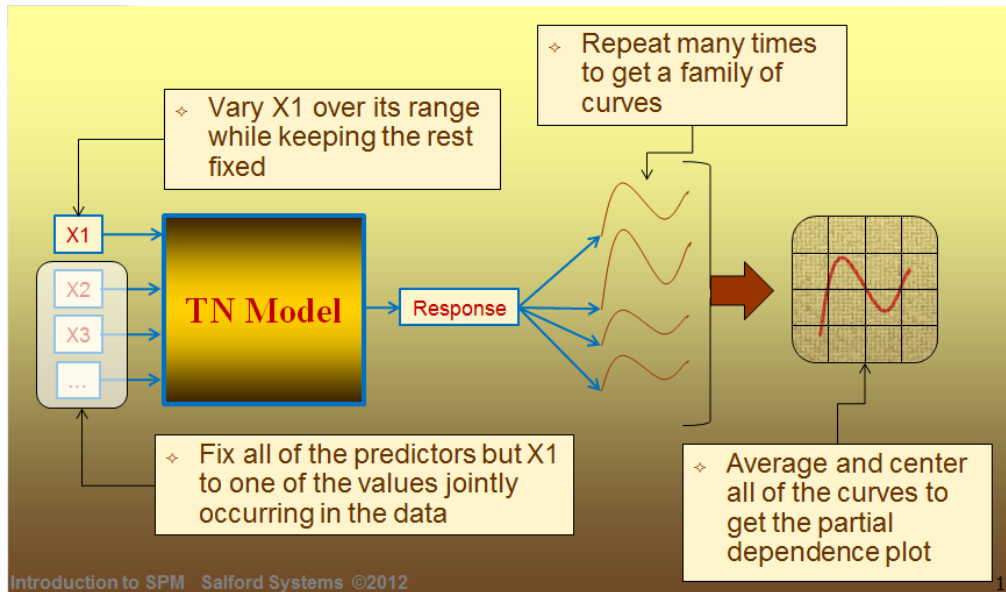
TreeNet Dependence Plots

Partial dependence plots represent a very powerful interpretational side of TreeNet. They were designed to help understanding how individual variables or pairs of variables contribute to the predicted response once the effects of all remaining predictors have been accounted for.

Depending on the complexity of a TreeNet model, the plots can be either exact or at the very least provide useful visual approximations to the nature of dependencies. The former case arises for truly additive models or models with completely isolated interacting pairs of variables, for example, utilizing the ICL language introduced later in this manual.

The actual procedure used by the TreeNet engine to construct partial dependence plots is shown on the following diagram.





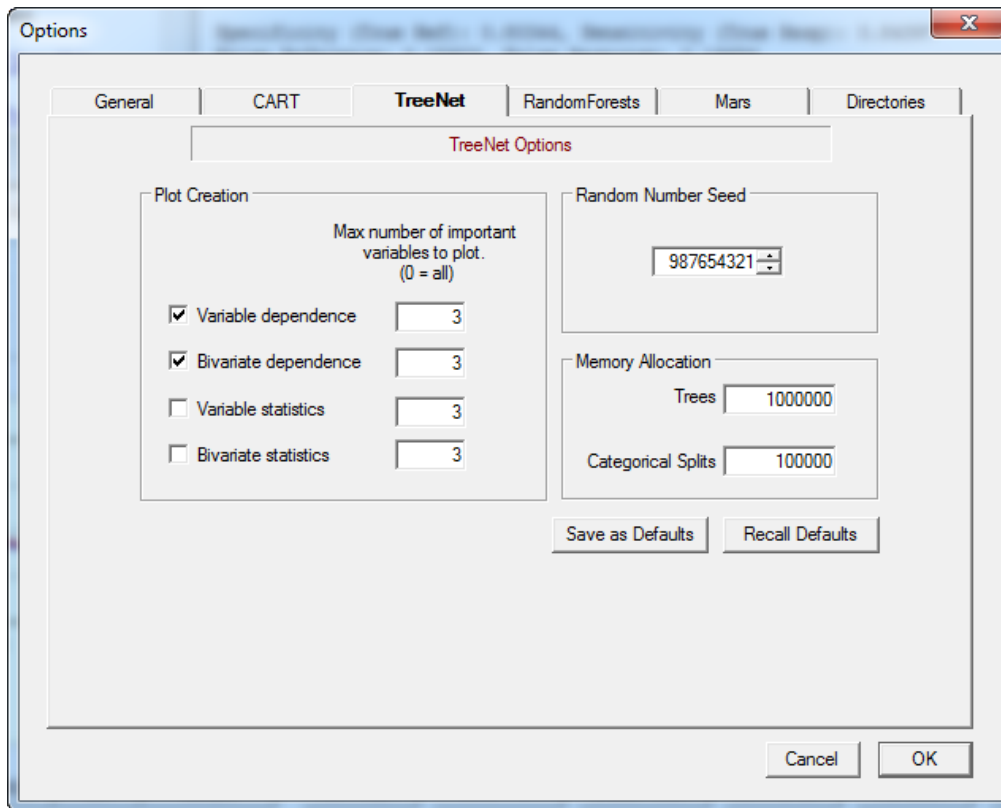
In order to obtain the univariate partial dependence plot with respect to X_1 , all of the remaining predictors are fixed at the joint set of values sampled from the dataset while X_1 varies over its range to produce one instance of the dependence curve in terms of model response. This procedure is repeated multiple times for all available learn records, such that each record gives another joint set of fixed values and the resulting dependence curve with respect to X_1 . In the end, the entire family of dependence curves is averaged and centered in order to produce the final dependence plot on X_1 displayed by the GUI.

A similar process can be conducted for any pair of variables, each iteration producing a partial dependence surface and the averaged final result displayed as the bivariate dependence plot.

Creating Dependence Plots

Dependence plots can be created automatically and manually by user request. To create plots automatically, go to the **Edit>Options** menu and switch to the **TreeNet** tab.





Put a checkmark in the **Variable dependence** box to request automatic creation of univariate dependence plots for the specified number of the top most important variables. Put a checkmark in the **Bivariate dependence** box to request automatic creation of bivariate dependence plots for the specified number of the top most important variables (all possible pairs will be considered).

☛ Constructing bivariate dependence plots may be time consuming.

Press the [**Save as Defaults**] button.

From now on, at the end of each TreeNet modeling run the plots will be automatically created for the specified number of variables taken from the variable importance list based on the optimal model. All plots will also be automatically added to the grove and can be saved for future reference.

- ✓ You may consider disabling automatic plot generation when running extensive batteries of runs to have significant time savings.
- ✓ The **Variable statistics** and **Bivariate statistics** check boxes activate legacy calculation of additional statistics based on plots. This approach has been largely superseded by the interaction reporting facility described in the next section.

Immediately after a TreeNet modeling run, a user may also request creating additional plots for models of any size and variables of choice.

To create additional plots by hand:

- ◆ In the **TreeNet Output** window, click on the model of interest (so that the red dotted vertical line is positioned at the right number of trees) and then press the [**Create Plots...**] button.



Create Plots

Data
Data file: C:\Program Files\Salford Systems\SPM7_final\Sample Data\sample.csv

Grove
Grove file: TreeNet Output 8 Type: Classification

Select Number Of Trees To Create Plots For

Training Sample
Nr. of trees: 150 Entropy: 0.225 Misclass: 0.0831 ROC: 0.974 Lift: 1.84 Optimal Tree

Test Sample
Sample: 0.293 Entropy: 0.27 Misclass: 0.112 ROC: 0.957 Lift: 1.82 Select...

Select Types
 One variable dependence Two variable dependence
200 points 1000 points

Select Variables To Plot

Variable	Score
X1	100.00
Z2\$	56.29
X2	42.09
Z1\$	37.91
X3	37.84
X4	27.04
X8	15.18
X6	13.45
X5	12.14
X7	11.08

Use top 3 variables

Select Select All Remove Remove All

X1
Z2\$
X2
Z1\$
X3
X4

Number of plots that 21

Number of existing plots: 0 Clear Existing Cancel Create Plots

- ✓ The model size is set to the number of trees (150 in this example) that was selected in the TreeNet Output window. One can change the model size directly in this window by manipulating the spin controls or pressing the **[Optimal Tree]** or **[Select...]** buttons.
- ◆ Put a checkmark in the **One variable dependence** box to request univariate dependence plots and in the **Two variable dependence** to request bivariate dependence plots.
- ◆ Change the number of points sampled for plot construction from the suggested default values if necessary.
- ✓ For the univariate dependence plots, consider increasing the number of points to the size of the learn sample (when feasible). This will ensure that all points, even potential outliers, will appear on the plot.
- ◆ In the **Select Variables To Plot** section, highlight the variables of interest and press the **[Select]** or **[Select All]** buttons to include the variables for plot construction. You may also remove variables from the plots list by pressing the **[Remove]** and **[Remove All]** buttons. You can also specify the number of top variables to use.
- ✓ Note that the variables are listed according to their importance in the model of specified size. The **Score** column lists the variable importance scores.
- ◆ Press the **[Create Plots]** button to start the process.
- ◆ The resulting Plot Selection window will now include additional plots.
- ◆ This process can be repeated as many times as necessary provided that the system resources remain available.



The following commands are used to create plots automatically:

```
TREENET PLOTS=[YES|NO],[YES|NO],[YES|NO],[YES|NO]
```

```
TREENET MV=<n>,<n>,<n>,<n>
```

```
TREENET MP=<n>,<n>,<n>,<n>
```

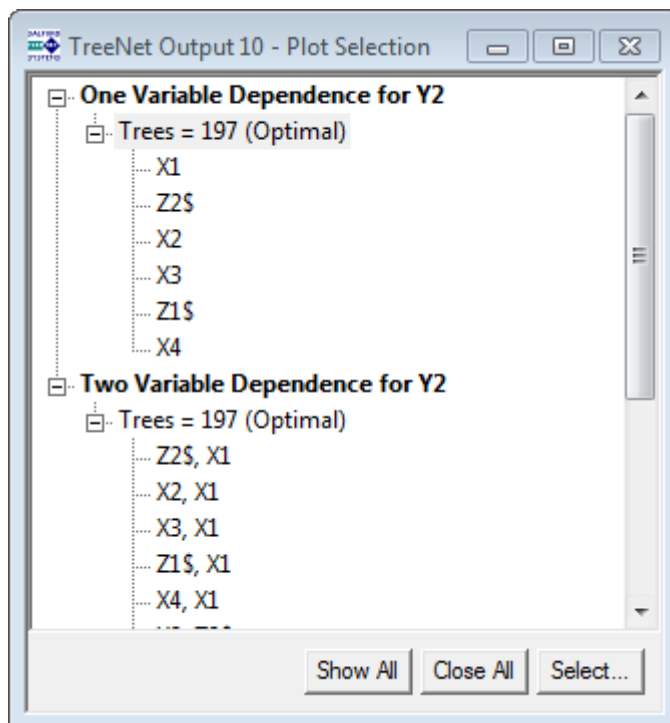
```
TREENET PP=<n>,<n>,<n>,<n>
```

```
TREENET PF="<file name>"
```

See the Command Reference chapter for details.

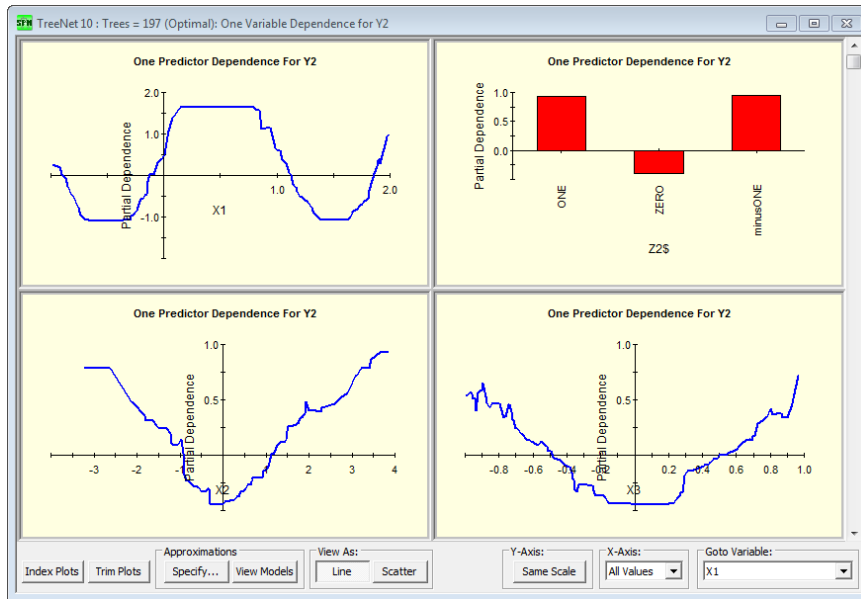
Viewing Dependence Plots

TreeNet Output – Plot Selection window contains all plots that are currently available for viewing. The plots are organized by type (univariate and bivariate), model size in trees, and individual variables. One can access this window any time by pressing the **[Display Plots...]** button (disabled when no plots are available).



Double-click on the **Trees=** line to open all plots for the selected model size.





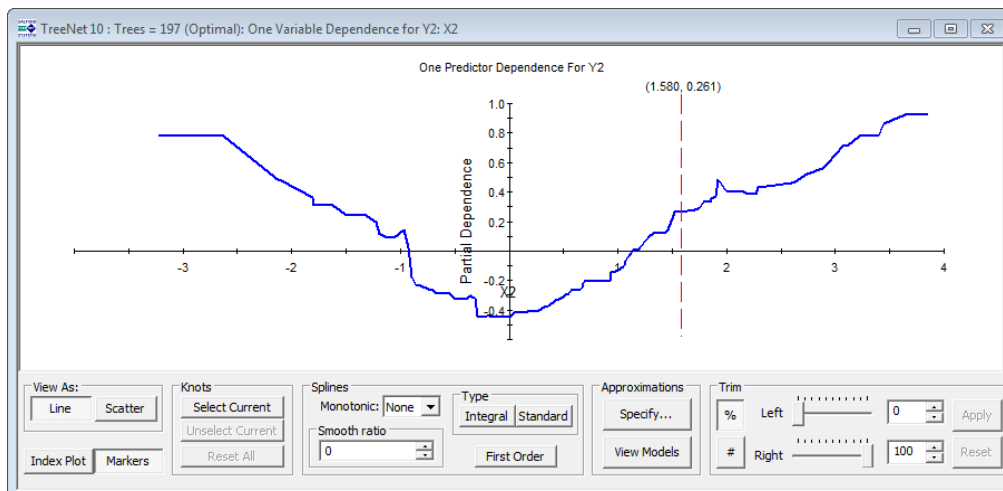
Press the **[Scatter]** button to view individual points that were sampled in plots construction. This is useful in identifying possible outliers or “empty” areas of data.

Press the **[Index Plots]** button to “de-scale” the horizontal axis such that all sampled points are spaced uniformly. This helps viewing “congested” areas of plots, especially when outliers are present.

Press the **[Trim Plots]** button to truncate plot tails usually due to outliers.

Press the **[Same Scale]** button to impose identical scale on all vertical axes and see the comparative effect of different variable contributions.

Click on the individual plot to open the most detailed view.



Note the vertical red dotted slider that can be moved within the graph area with the (X, Y) pair of values displayed above it. One can use the slider to view plot values as well as set a number of knot points for manual spline transformations.



Press the **[Line]** or the **[Scatter]** buttons to switch between the line and dotted display of sampled points (see earlier comments).

Press the **[Index]** button to remove horizontal scaling of the plot (see earlier comments).

Use the **Trim** controls to remove tails of the graphs (see earlier comments).

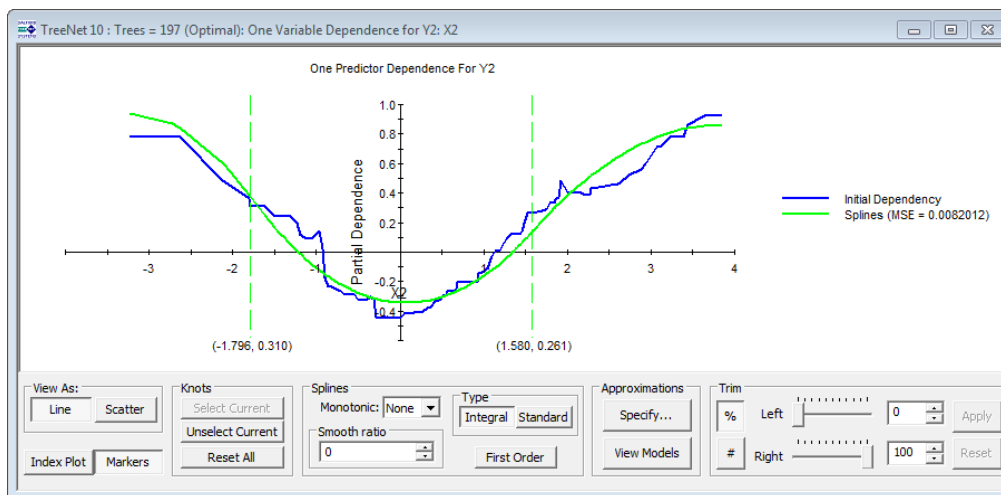
Plot Approximations

The plot approximations machinery was designed to facilitate model extraction based on TreeNet model insights. The idea is that each dependency plot represents a variable transformation; therefore, once a variable is transformed it can be used in the conventional regression modeling as a main effect. Note that the transformation becomes exact in the absence of interactions in which case the TreeNet engine can be used as an ultimate transformation discovery machine,

Plot approximations replace the internally constructed dependence plots by simple analytical expressions calculated to have the best fit to the dependence plot of interest.

By far, the spline approximations are the most flexible to accomplish this task. Each spline is defined by a set of knots (points on the horizontal X-axis) and its order (first-order spline for piece-wise linear segments and second-order splines for the smooth quadratic segments). The internal calculations ensure the continuity and/or the smoothness of the spline approximation.

To add a knot, simply position the vertical slider at the point of interest and press the **[Select Current]** button. Note that each added knot will be represented by a vertical green dotted line. One can use the mouse pointer to move the existing knots left or right to achieve fine adjustment of the spline fit. In the example below, we show a two-knot spline solution to the X2 dependence plot.



By default, second-order splines are calculated. Press the **[First Order]** button to replace them with the first-order splines.

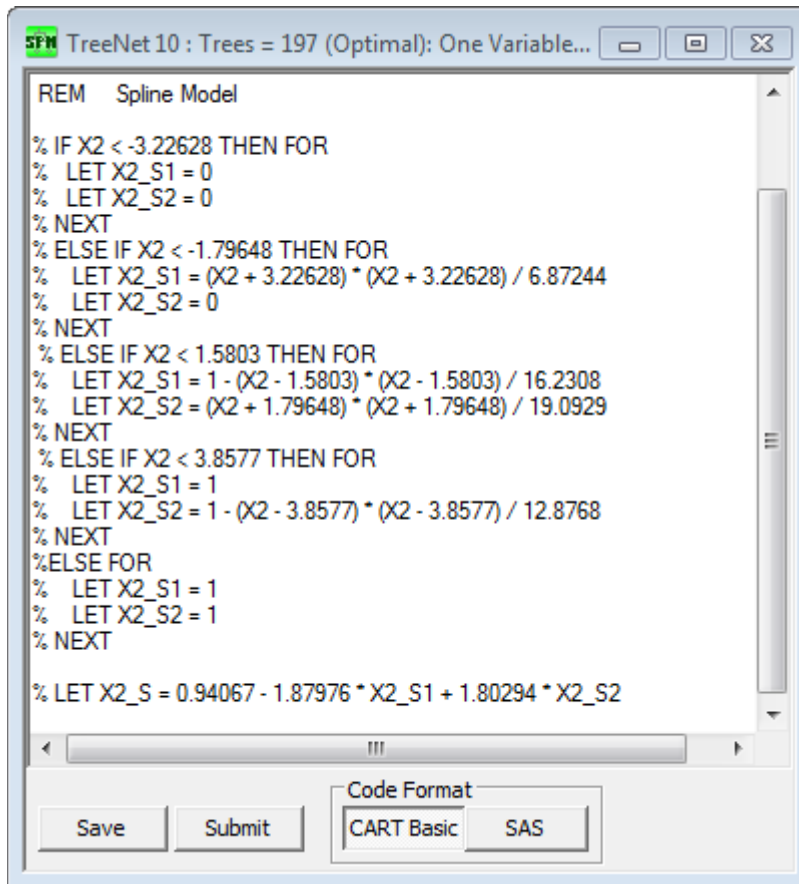
One can also enforce monotonicity constraint on the spline solution using the **Monotonic:** control.

One can also enforce additional smoothness on the spline solution using the **Smooth ratio** control.

✓ The **[Standard]** versus **[Integral]** spline type control is no longer supported. All splines are now integral spline because of their superior characteristics.



To view the spline solution in code terms, press the **[View models]** button.



```

SPM TreeNet 10 : Trees = 197 (Optimal): One Variable...
REM Spline Model
% IF X2 < -3.22628 THEN FOR
% LET X2_S1 = 0
% LET X2_S2 = 0
% NEXT
% ELSE IF X2 < -1.79648 THEN FOR
% LET X2_S1 = (X2 + 3.22628) * (X2 + 3.22628) / 6.87244
% LET X2_S2 = 0
% NEXT
% ELSE IF X2 < 1.5803 THEN FOR
% LET X2_S1 = 1 - (X2 - 1.5803) * (X2 - 1.5803) / 16.2308
% LET X2_S2 = (X2 + 1.79648) * (X2 + 1.79648) / 19.0929
% NEXT
% ELSE IF X2 < 3.8577 THEN FOR
% LET X2_S1 = 1
% LET X2_S2 = 1 - (X2 - 3.8577) * (X2 - 3.8577) / 12.8768
% NEXT
% ELSE FOR
% LET X2_S1 = 1
% LET X2_S2 = 1
% NEXT
% LET X2_S = 0.94067 - 1.87976 * X2_S1 + 1.80294 * X2_S2
  
```

Code Format

Note that the option to switch between the **[CART Basic]** and **[SAS]** languages.

Alternatively, one may attempt fixed type dependence plot approximations. Click the **[Specify...]** button to try the following common approximations:

- ◆ **Logit** – popular S-shaped transformation used in logistic regression and neural nets.
- ◆ **Inverse** – inverse transformation popular in econometrics for modeling demand curves and diminishing returns.
- ◆ **Exponent** – exponential transformation to model growth phenomena, etc.
- ◆ **Logarithm** – another popular transformation to model supply curves, etc.
- ◆ **Polynomial** – simple analytical transformation that uses higher order polynomial terms.



TN Dependency Approximations Setup

Splines

Number of Knots: 2

Type: Integral Standard

Order: First Order

Monotonic: None

Smooth rate: 0

Logit

$Y = \text{[]} + \text{[]} / (1 + \exp(\text{[]} * X + \text{[]}))$

Inverse

$Y = \text{[]} + \text{[]} / (\text{[]} + X)$

Exponent

$Y = \text{[]} + \text{[]} * \exp(\text{[]} * X)$

Logarithm

$Y = \text{[]} + \text{[]} * \log(\text{[]} + X)$

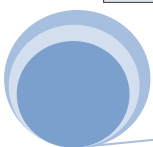
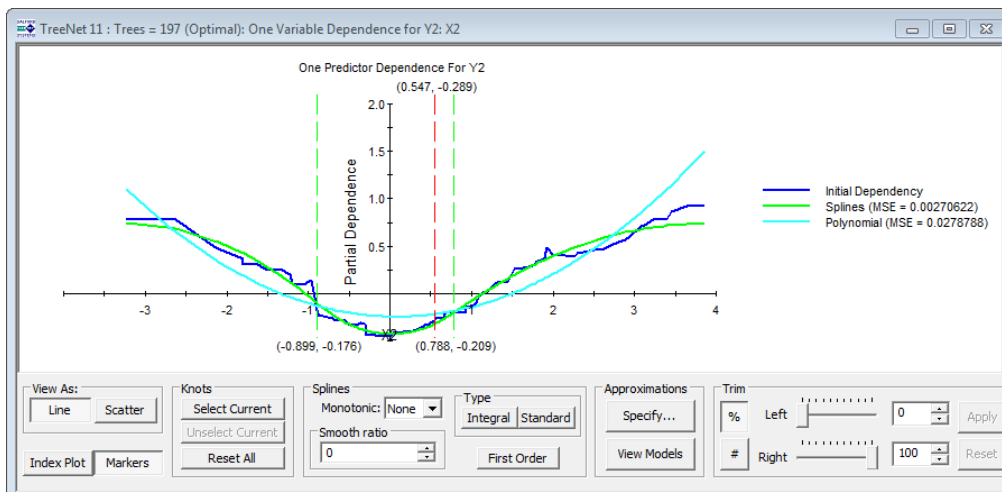
Polynomial

Degree: 2

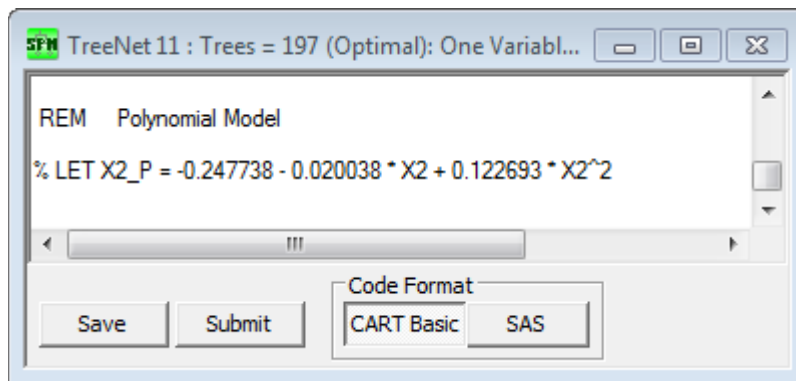
Trimmed Dependences Approximation

OK Apply Reset All Cancel

Press the **[Apply]** button to immediately see the result of the transformation directly superimposed over the graph. In the example below we requested the second-degree polynomial approximation represented by the cyan curve.



To see the analytic expression press the **[View Models]** button.



- ✓ One can generate approximations for all univariate graphs simultaneously by pressing the [Specify...] button on the One Variable Dependency window organizing all plots.
- ✓ There is no approximations facility for the bivariate plots due to their natural complexity.
- ✓ There is no command line support for the approximations.

Interactions and TreeNet

Interaction reporting and enforcing is one of the most recent additions to the TreeNet engine. Most controls and reports are available through command line with some features supported at the GUI level. This part of TreeNet is likely to undergo major changes and enhancements in the future releases, even though the core underlying machinery is fully developed.

The main machinery behind interaction reporting is based on the comparison between a genuine bivariate plot (where variables are allowed to interact) and an additive combination of the two corresponding univariate plots. By gaging the difference between the two response surfaces one can measure the strength of interaction effect for the given pair of variables. The whole process is automated to quickly identify variables and pairs of variables most suspect to have strong interactions.

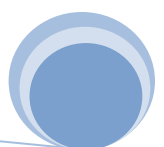
The core idea behind interaction enforcing is that variables can only interact within individual trees. Therefore, providing fine controls over how individual variables can enter into a tree is the key to allowing or disallowing different interactions. For example, requesting 2-node trees (one split per tree) will effectively disable all interactions because any TreeNet model is always additive across trees. This approach can be further expanded to multiple node trees by controlling which variables are allowed to jointly appear within individual branches.

Interactions Reporting

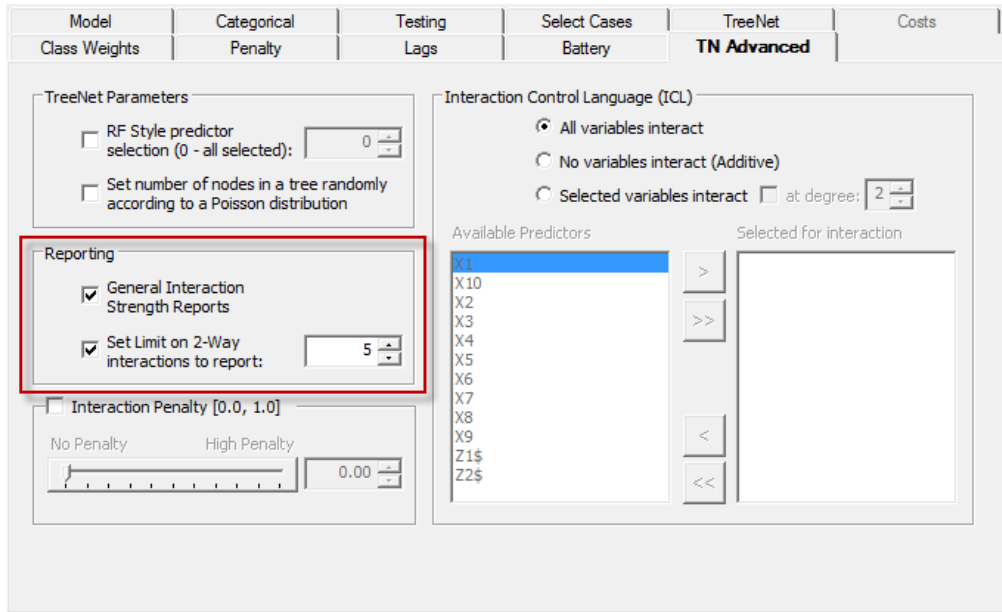
```
TRENET INTER=YES
```

To activate interactions reporting:

- ◆ Type in the above command in the command line or by check the **General Interaction Strength Reports** box in the **TN Advanced** tab of the **Model Setup** window.
- ◆ Set the number of 2-way interaction tables to report (default 5).



- Generating interaction reports consumes approximately same amount of time as generating all dependence plots which may be comparable with the model building time and even longer.



From now on every TreeNet run will include a new report named **TreeNet Interactions** in the **Classic Output** window.

We rebuilt the logistic regression model from the previous run and obtained the following report:

```

=====
TreeNet Interactions
=====

New Predictor in Tree Penalty: 0.00
No inhibition of Interactions
# Variables tested for 2-way interactions: Top 5

Whole Variable Interactions
Measure Predictor
-----
25.21229 Z2$
22.78410 Z1$
15.36705 X1
12.40157 X3
11.57384 X2
 5.03785 X4
 4.14299 X8
 3.34241 X6
 3.21837 X5
 2.75992 X10
 2.59096 X7
 2.55953 X9
    
```



This table provides the list of all available predictors sorted according to the suspected overall interaction strength. The interaction strength is on the % scale, which in this example indicates that about 25% or the total variation in the predicted response can be attributed to interaction of the categorical variable Z2\$ with someone else. The same goes for all remaining variables. The report is only approximate and may require further investigation, including trying various interaction enforcing strategies described in the next section.

The overall interaction report is then followed by detailed 2-way interaction reports for the top 5 (default) interacting variables. Of those, we only include the top two.

```

=====
2-way Interaction Stats
=====

Predictor: Z2$
  Measure1   Measure2  Predictor
-----
  23.58970   41.62182  Z1$
  10.76775   12.86287  X1
   3.92681    8.34831  X3
   3.14635    6.53373  X2
   1.25122    2.74682  X4
   1.22972    2.95252  X8
   0.51351    1.24242  X9
   0.22626    0.54815  X10
   0.09632    0.23324  X6
   0.00000    0.00001  X7
   0.00000    0.00001  X5

Predictor: Z1$
  Measure1   Measure2  Predictor
-----
  23.58970   41.62182  Z2$
   1.33213    1.81731  X1
   1.02704    4.05412  X3
   0.39477    1.79535  X4
   0.36944    3.18801  X8
   0.34368    3.16610  X7
   0.30411    1.13615  X2
   0.13594    1.19171  X5
   0.13206    1.21656  X10
   0.11972    1.11434  X9
   0.06144    0.55828  X6
  
```

Again, all measures are on the % scale. The Measure1 shows the contribution of the suspected interacting pair normalized to the overall response surface (total variation of the predicted response) whereas the Measure2 shows the contribution of the suspected interacting pair normalized to the pair of variables itself (in terms of combined main effects).

- ✓ Therefore, Measure1 is always smaller than Measure2.
- ✓ Very small Measure1 combined with large Measure2 indicates string interaction per se even though the overall contribution of this pair compared to the remaining variables is rather insignificant.



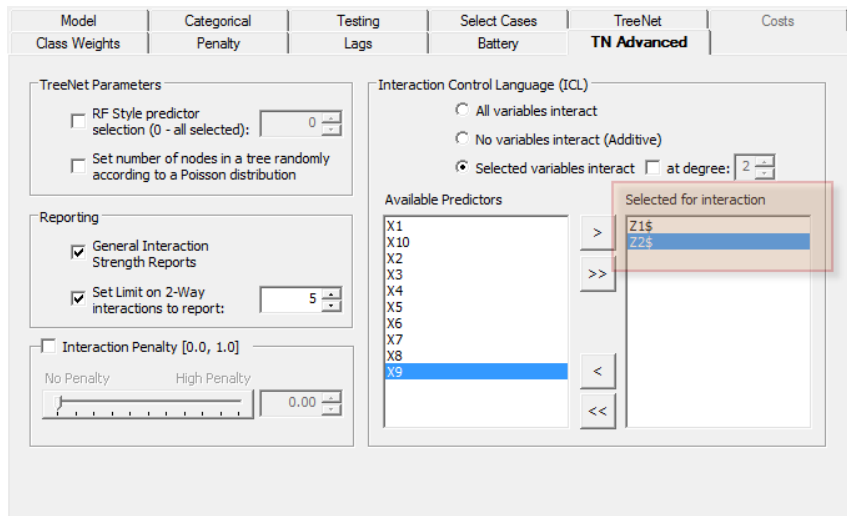
It is worth reminding that all interaction reports are based on the patterns in which variables appear together on the same branch of a tree in the TreeNet model. The reports can be considered as no more than a set of hypotheses because apparent interactions may be due to chance factors or, alternatively, may not be needed to obtain the best possible model by changing the overall model composition. One often uses this report as a starting point to guide the follow up use of the ICL commands (see the next section). One can test whether an interaction is "real" by preventing it from occurring in a new TreeNet model. If imposing such a restriction does not harm model performance then the interaction is declared to be "spurious." In general, it is best to constrain the models by first restricting the less prominent interactions and gradually imposing more and more restrictions.

Interactions Enforcing

The GUI offers several straightforward ways to control interactions during model building process. In the **TN Advanced** tab of the **Model Setup** window one can choose:

- ◆ **All variables interact** – the model is not restricted in any way (default)
- ◆ **No variables interact (Additive)** – the strongest restriction that forces additivity in all predictors.
- ◆ **Selected variables interact** – the user selected group of variables is allowed to interact among themselves up to the specified degree.

In the following example we only allowed interactions between Z1\$ and Z2\$ while forcing the rest of the variables to be purely additive.



The screenshot shows the 'TN Advanced' tab of the 'Model Setup' window. The 'Interaction Control Language (ICL)' section is active, with the following settings:

- All variables interact
- No variables interact (Additive)
- Selected variables interact at degree: 2

The 'Available Predictors' list contains X1, X10, X2, X3, X4, X5, X6, X7, X8, and X9. The 'Selected for interaction' list contains Z1\$ and Z2\$.



```

=====
TreeNet Interactions
=====

New Predictor in Tree Penalty: 0.00
No inhibition of Interactions
# Variables tested for 2-way interactions: Top 5

Whole Variable Interactions
  Measure Predictor
-----
 33.19123 Z2$
 33.19123 Z1$
  0.00004 X2
  0.00004 X3
  0.00004 X10
  0.00004 X1
  0.00004 X4
  0.00004 X5
  0.00003 X7
  0.00003 X6
  0.00003 X9
  0.00003 X8


=====
2-way Interaction Stats
=====

Predictor: Z2$
  Measure1      Measure2 Predictor
-----
 33.19122      52.84644 Z1$
  0.00000      0.00000 X1
  0.00000      0.00000 X2
  0.00000      0.00001 X10
  0.00000      0.00000 X3
  0.00000      0.00000 X5
  0.00000      0.00000 X9
  0.00000      0.00000 X4

```

One can clearly see from the reports that all variables except for Z1\$ and Z2\$ now enter in a pure additive way.

The command line offers even greater controls over the specific structure of interactions to be allowed or disallowed in a TreeNet model. Here we provide brief highlights on the subject, the complete description of the command listed below can be found in the Command Reference chapter of this manual.

 ICL ADDITIVE = <variable list>

This command has the highest priority and prevents any and all interactions in the TreeNet among the variables listed on the command. In the presence of any other ICL command (see below), the remaining variables are allowed to interact as usual.



```
ICL ALLOW <variable list> /degree
```

This command allows the listed variables to interact only among themselves, provided that they do not occur on the ADDITIVE list. Multiple ALLOW commands with possibly overlapping lists of variables are admissible. The variables are allowed to interact only if they occur on one or more ALLOW lists and do not occur on the ADDITIVE list,

- ✓ Thus, the best practice is either defining an ADDITIVE list or defining a collection of ALLOW lists but never both. The former case is equivalent to a pair of complementing ADDITIVE and ALLOW statements, while the latter is equivalent to having an implied complementing ADDITIVE command.

```
ICL DISALLOW <variable list> /degree
```

This command only works “inside” of an existing ALLOW command, it further refines additional combinations of variables that are specifically not allowed to interact among themselves even though they are allowed to interact with any other member of the ALLOW group.

Scoring and Translating TreeNet Models

You can score a TreeNet model directly from an open grove (**TreeNet Output** window) by pressing the **[Score...]** button or using the **Model>Score Data...** menu item. See the scoring chapter for specific details on the available controls.

Note that you can specify the specific model size (in trees) for scoring by pressing the **[Select...]** button on the **Score Data** dialog window. The default model is the optimal number of trees for the specified performance criterion.

Similarly, you can translate a TreeNet model directly from an open grove (**TreeNet Output** window) by pressing the **[Translate...]** button or using the **Model>Translate Model...** menu item. See the translating chapter for specific details on the available controls.

- ✓ TreeNet has additional translating mode LANGUAGE=PLOTS (see the next section) which allows to convert all created plots into XML-style output.

Rare TreeNet Commands

Most of the engine specific control parameters occur on the TREENET command. Type HELP TREENET in the command line to see a detailed description of every available option. You can also consult extensive command reference chapter of this manual.

Here we point out some interesting features available only through the command line and otherwise not directly accessible through the Model Setup window.

```
TRENET FULLREPORT = [YES|NO]
```

Only applies to classification models, turns on additional extensive reporting like prediction success tables, detailed variable importance tables, etc.

```
TRENET GB = <n>
```

Defines the number of bins in gains charts in the text output.

```
TRENET INIT = <variable>
```



Specifies a variable that contains “start values” for the model. This is very useful in hybrid modeling approaches where TreeNet model is developed as a follow up model to an already built model. For example, a linear regression model (captures main linear trends) may be followed by TreeNet to explore departures from linearity and also interactions.

```
☞ TREENET PF = <file name>
```

```
☞ TRANSLATE LANGUAGE = PLOTS
```

Two alternative commands to request saving of all TreeNet plots into an XML-style text file.

```
☞ TREENET LOWMEMORY = [YES|NO]
```

Forces TreeNet to reduce memory footprint by 33%, the run time may double.

Typical Command Files

We conclude our discussion of the TreeNet engine by providing sample command files of how to build various TreeNet models as well as score and translate them. The sample files are located in the **Sample Data** folder.

Note the comment lines starting with REM which show alternative forms of the key commands.

Treenet_reg_build.cmd – builds a continuous response TreeNet model and saves it into a grove file.

Treenet_reg_score.cmd – scores a previously saved continuous response model and saves the predicted values.

Treenet_reg_translate.cmd – translates a previously saved continuous response model into SAS-compatible code.

Treenet_logit_built.cmd – builds a binary response TreeNet model and saves it into a grove file.

Treenet_logit_score.cmd – scores a previously saved binary response model and saves the predicted probabilities.

Treenet_logit_translate.cmd – translates a previously saved binary response model into SAS-compatible code.

