Salford Predictive Modeler®

Model Compression via ISLE and RuleLearner®

This guide elaborates on powerful ways to combine the TreeNet® and GPS engines to achieve model compression and more.

Minitab

Background and Motivation

The principal idea behind model compression is that it may be possible to use modern regression techniques to post-process a complex model to create a radically simplified version that can perform almost as well as the original. It can go even further: if you build the original complex model in a specific way there is an excellent chance that the post-processed simplified model will perform even better than the best possible complex model!

Model compression is important for several reasons:

- ♦ Compressed models can perform dramatically faster in scoring. For real time scoring compression may be the only way to feasibly deploy a high performance TreeNet®.
- ♦ Compressed models may use many fewer predictors; compression may thus offer an alternative way to accomplish feature selection.
- ♦ Compression may provide models that are easier to explain using the component trees.

In this chapter we will cover model compression basics and try to explain why it can be expected to work. To begin, you need to understand that any model compression is a four stage process.

First, we use one of our essential data mining engines to build a predictive model. The engine could be CART®, MARS®, TreeNet®, or Random Forests®.

Second, we use the predictive model obtained to convert the original data (train and test partitions) into a new transformed version of the data.

Third, we use a regularized regression (GPS) to build a new model using the transformed data.

Finally, we extract a new compressed version of the original model.

✓ In our first release of model compression the first stage model must be a TreeNet, but in later releases the other engines can be used instead.

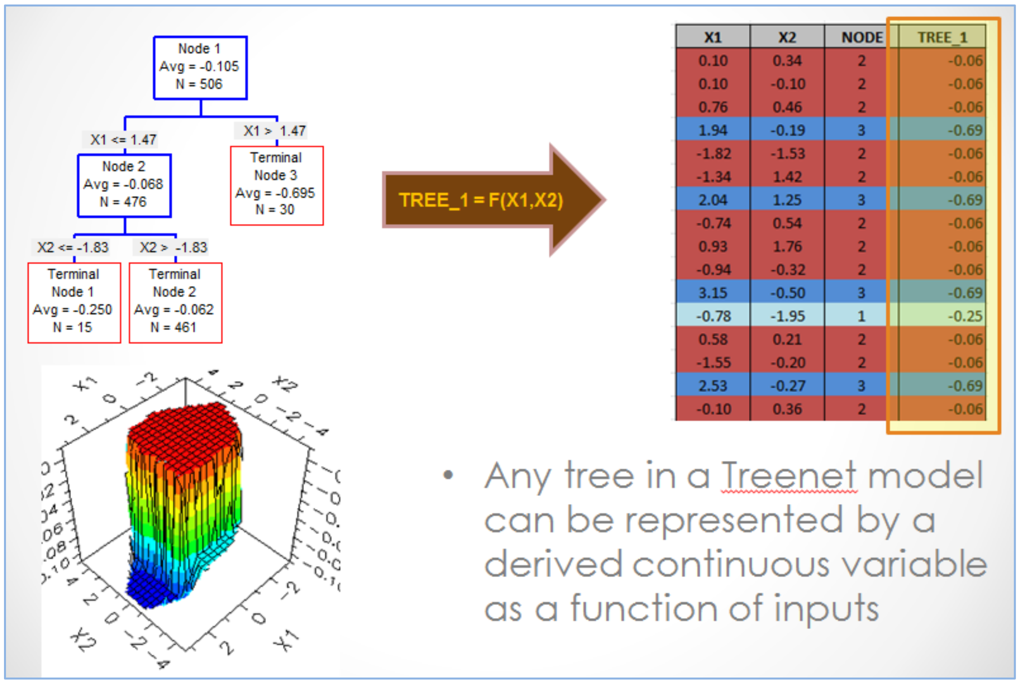
To understand how to work with model compression you need to have a basic grasp of working with the GPS regularized regression engine described earlier. If you have not already done so, please review those basics as we will assume this understanding in the discussion that follows.

The second step above requires further elaboration as there are several possible ways to re-express a TreeNet model into a linear combination of components later to be fed into the GPS engine for compression purposes. Two fundamentally different approaches arise here, one is called *ISLE* and the other is called *RuleLearner®*. We start by explaining the first one and then follow up with the second.

ISLE is an acronym for *Importance Sampled Learning Ensembles*, introduced by Jerome Friedman with co-author Bogdan Popescu in 2003.

✓ The technical paper is available on our website at http://www.salford-systems.com

While the ISLE approach has a far reaching theoretical setup, for all practical purposes it can be introduced as decomposing a TreeNet model into a set of variable transformations represented by each individual tree in that model. This stems from a simple observation that any tree can be viewed as a potentially multivariate transformation of the original predictors. This is illustrated below.



X1	X2	NODE	TREE_1
0.10	0.34	2	-0.06
0.10	-0.10	2	-0.06
0.76	0.46	2	-0.06
1.94	-0.19	3	-0.69
-1.82	-1.53	2	-0.06
-1.34	1.42	2	-0.06
2.04	1.25	3	-0.69
-0.74	0.54	2	-0.06
0.93	1.76	2	-0.06
-0.94	-0.32	2	-0.06
3.15	-0.50	3	-0.69
-0.78	-1.95	1	-0.25
0.58	0.21	2	-0.06
-1.55	-0.20	2	-0.06
2.53	-0.27	3	-0.69
-0.10	0.36	2	-0.06

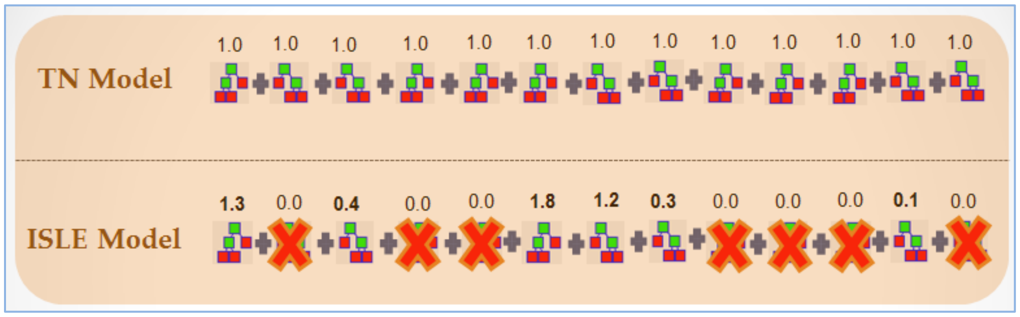
- Any tree in a Treenet model can be represented by a derived continuous variable as a function of inputs

One can therefore represent any TreeNet model as a linear combination of individual tree variables with coefficients 1.0.

✓ This assumes that the learn rate and initial guess adjustments were incorporated into the trees, which is trivial to do.

Thus, after these variables have been fed as inputs to the GPS engine, a new linear combination will have different coefficients. Given the GPS's inherent ability to do variable extraction by trying different variable selection strategies (elasticities), a fraction of predictors may have zero coefficients which in this context is equivalent to removing the corresponding tree from the model thus achieving model compression. As an added bonus, it is possible that the new model will have superior performance.

The ISLE compression is visualized on the next diagram.



Note how some trees have been eliminated (zero coefficients) while the others were adjusted to provide the best fit.

Another way to think about the model compression is to recall that a TreeNet model is an ensemble model consisting of many small CART-style decision trees. Each tree produces an output "score", and the final model is simply the grand total of all the individual scores. Usually TreeNet approaches its final result slowly, with each tree contributing a small part of the final model result. When we "compress" the

TreeNet model via GPS regression we select a subset of these trees and reweight them in order to arrive at a model which has fewer trees and possibly better performance as well. The abstract technical mechanism by which this is accomplished is best understood by studying GPS; here we provide an intuitive explanation for what is going on.

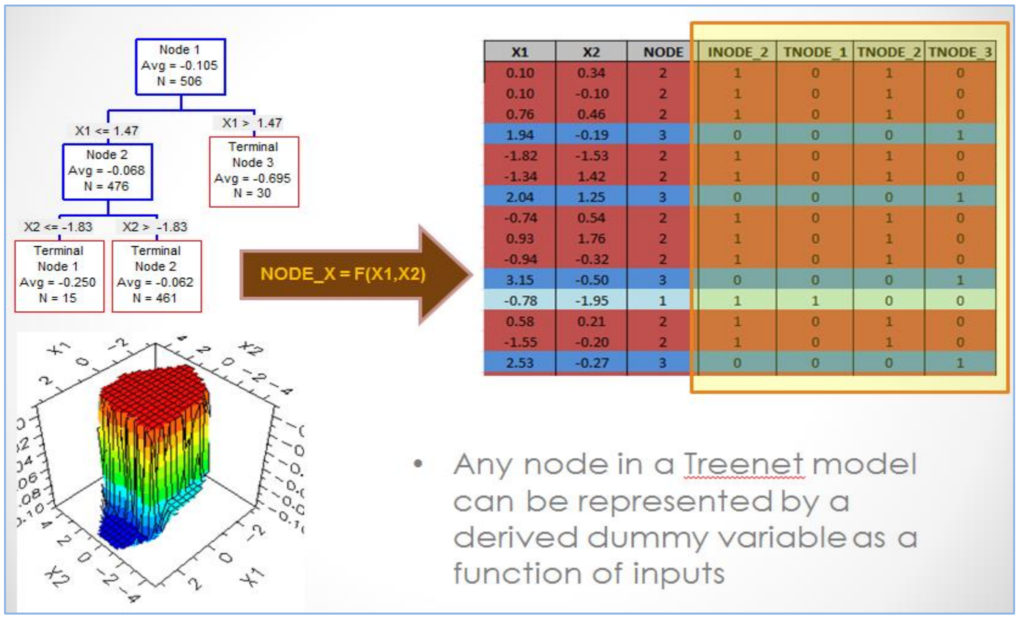
A TreeNet model is built myopically, looking only one step ahead as the model is built tree-by-tree. To obtain good results, each step is small, meaning that each tree can add only a very small amount to the overall TreeNet prediction. When the TreeNet model is complete we typically have very many of these trees and there is an excellent chance that some tree structures are repeated exactly, or almost exactly. The compression phase can then work as follows:

- ♦ Identical or almost identical trees can be aggregated into just one tree with an appropriate weight. If the same tree was grown 20 times we could replace these trees with just one tree and provide it with a weight of 20. In real world modeling the trees may differ in details which make exact reproduction in this way impossible, but we may be able to produce a very close approximation.
- ♦ When the TreeNet model is grown we do not know what the next tree will bring and in the process of arriving at final scores there may be some oscillation around a final solution or even backtracking in which some trees are essentially cancelled out of the final result. Once the model has been constructed we can accomplish any "cancelling out" by simply deleting trees and we can skip over any oscillations. The end result will be a smaller collection of trees.

The final result of the ISLE Model compression is a smaller model of the same type we started with. In this case, we always start with a TreeNet and we end up with a TreeNet. But the new TreeNet has fewer trees and each tree has its own weight which could be far different than 1.0!

The second approach (here introduced as the **RuleLearner**) of decomposing a TreeNet model into individual derived variables is introduced by pushing the decomposition process further down, all the way to the individual nodes. Observe that any tree spawns a set of dummy variables, one variable per each terminal and internal node.

- ✓ The only exception is the root node which includes the entire sample and would result to a trivial constant dummy variable.

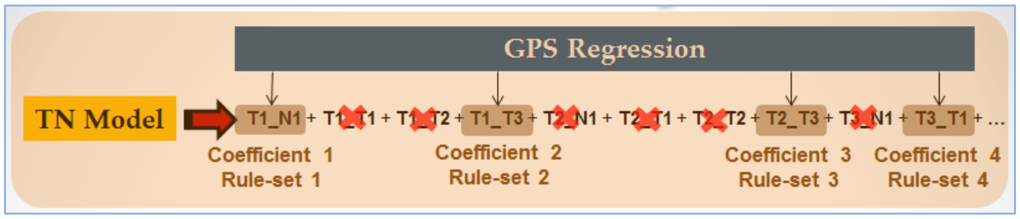


X1	X2	NODE	INODE_2	TNODE_1	TNODE_2	TNODE_3
0.10	0.34	2	1	0	1	0
0.10	-0.10	2	1	0	1	0
0.76	0.46	2	1	0	1	0
1.94	-0.19	3	0	0	0	1
-1.82	-1.53	2	1	0	1	0
-1.34	1.42	2	1	0	1	0
2.04	1.25	3	0	0	0	1
-0.74	0.54	2	1	0	1	0
0.93	1.76	2	1	0	1	0
-0.94	-0.32	2	1	0	1	0
3.15	-0.50	3	0	0	0	1
-0.78	-1.95	1	1	1	0	0
0.58	0.21	2	1	0	1	0
-1.55	-0.20	2	1	0	1	0
2.53	-0.27	3	0	0	0	1

- Any node in a Treenet model can be represented by a derived dummy variable as a function of inputs

Thus the complete set of node dummies represents all structural information contained in any TreeNet model. Note that the set is overrepresented, internal nodes largely overlap with their parents and completely include their children. This opens up even greater opportunities for possible model simplification by identifying the subset of influential nodes and finding coefficient adjustments associated with each node such that the original model fit is essentially recovered. Again, just as in the case of ISLE approach, the GPS regularized regression engine is perfectly suited for this task. Furthermore, it is possible that the resulting model may achieve higher performance compared to the original TreeNet model while maintaining an excellent compression ratio.

This process is illustrated on the following diagram.



Observe that each node dummy represents a collection of rules in terms of the original predictors leading into this node. Therefore, another way to look at this type of compression is in terms of extracting a collection of informative rule-sets (segments). Each rule-set operates on a subset of data or, to paraphrase, each rule-set represents a segment of the data. The regression coefficient assigned to this segment by the GPS is simply the segment-based adjustment to the overall model prediction. Note that segments can overlap with each other and may have either positive or negative adjustments of varying magnitude thus making the model prediction extremely versatile.

Both types of TreeNet model compression (ISLE and RuleLearner) are implemented in SPM®. In the following sections we describe the basic steps needed to setup model compression as well as explain various output windows produced along with the possible usage modes.

Model Compression in Action – ISLE EXAMPLE (coupled with RULELEARNER sub-example)

At this point it will suffice to say that setting up a model compression run from a user standpoint is very straightforward. We are essentially creating a “pipeline”, a sequence of two already introduced engines, first a TreeNet engine using the original predictors then followed up by a GPS engine that uses decomposed TreeNet model as its own inputs. The decomposition itself is done automatically by SPM based on the user selected method, which could be either ISLE or RuleLearner (as well as additional trivial modifications introduced below). In addition, it is assumed that each engine component of the pipeline (TreeNet and GPS) is configured using its own set of control parameters as usual.

The end result of the compression pipeline is the overall compression summary plus the usual output of the TreeNet engine (as if it were run by itself) as well as the standard GPS output of the follow up compression phase.

The output displays are deliberately designed for ease of comparison between the original uncompressed model and the resulting compressed model. The user can choose any desired model for viewing, scoring, and translation using the “point and click” approach thus balancing the degree of compression versus model size.

Setting up Model Compression

We start with the GOODBAD.CSV data set which is rather on the small side but runs very quickly and illustrates our main points well. The dataset comes from a credit default study and has 664 observations and 14 variables including a binary response variable named TARGET, with 1 indicating a bad credit standing account and 0 otherwise. The remaining predictors provide a nice mix of continuous and categorical measures which represent a typical modeling environment. The goal of the study is to build a predictive model for bad credit using the TreeNet approach and then possibly compress the model using either ISLE or RuleLearner approaches.

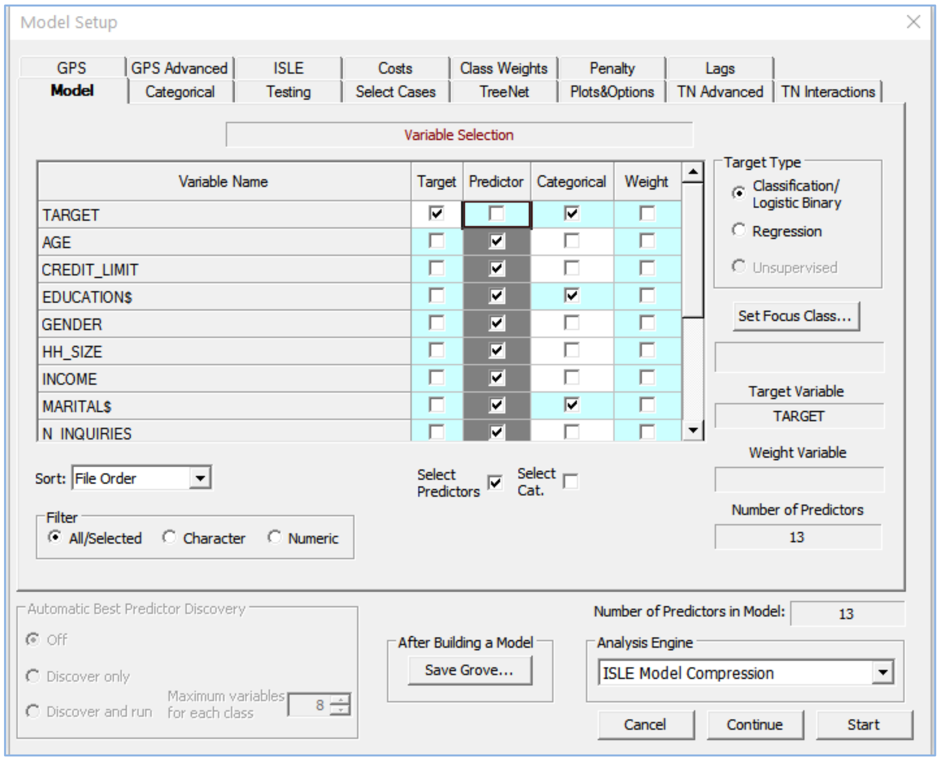
Use the **File/Open/Data File…** menu item to open the GOODBAD.CSV dataset, then click the **[Model…]** button in the resulting activity window to open the **Model Setup** window.

In the Model tab:

- ♦ Set **Analysis Engine** to **ISLE Model Compression** – this activates the pipeline part of the SPM – the **ISLE** tab. This exercise is useful for elaborating on the **RuleLearner** component as well.

✓ Note that this mode activates the **TreeNet** tab and the **GPS** and **GPS Advanced** tabs, thus giving you full control over other parts of the pipeline.

- ♦ Set **Sort:** to **File Order** – this sorts all variables according to the position in the dataset.
- ♦ Check TARGET variable as the **Target**.
- ♦ Check all remaining variables as predictors.
- ♦ Select **Target Type** to **Classification/Logistic Binary** to take advantage of the natural capability of both TreeNet and GPS engine to handle binary targets effectively.



Variable Selection

Variable Name	Target	Predictor	Categorical	Weight
TARGET	✓		✓	
AGE		✓		
CREDIT_LIMIT		✓		
EDUCATION$		✓	✓	
GENDER		✓		
HH_SIZE		✓		
INCOME		✓		
MARITAL$		✓	✓	
N_INQUIRIES		✓		

Sort: File Order

Select Predictors ✓ Select Cat. ☐

Filter: [✓ All/Selected] Character Numeric

Target Type: [✓ Classification/ Logistic Binary] Regression Unsupervised

Target Variable: TARGET

Weight Variable:

Number of Predictors: 13

Automatic Best Predictor Discovery: [✓ Off] Discover only Discover and run; Maximum variables for each class: 8

After Building a Model

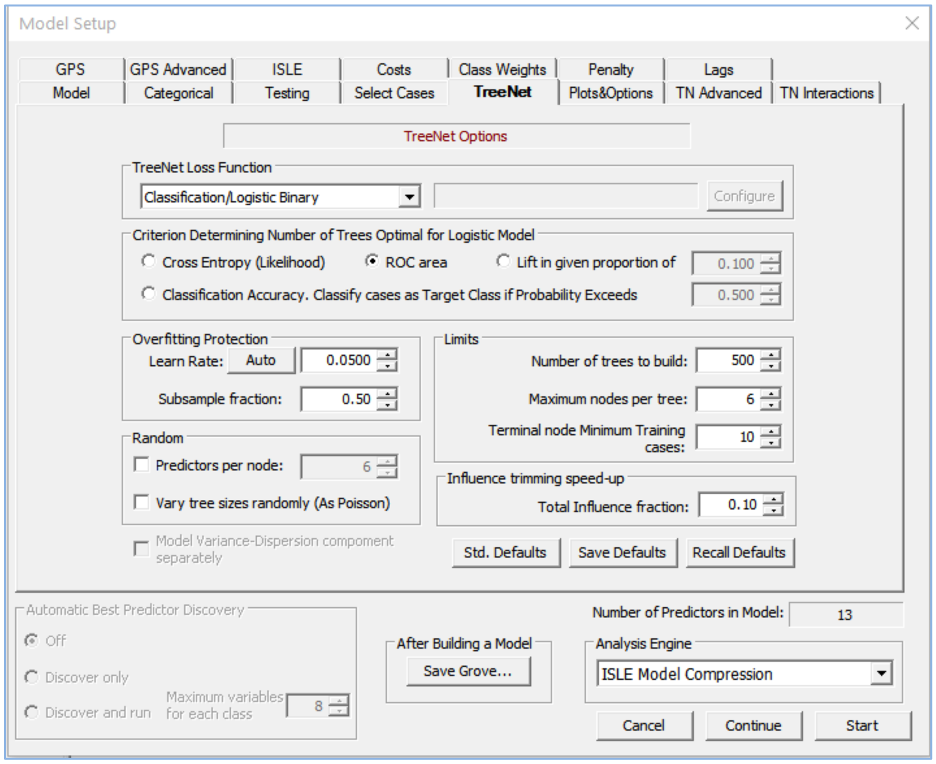
Number of Predictors in Model: 13

Analysis Engine: ISLE Model Compression

Switch to the Testing tab, and make sure that **Fraction of cases selected at random** for testing is set to 0.2.

We now need to set up the TreeNet component of the pipeline, as if you were setting up a stand-alone TreeNet model. Using the TreeNet tab, we set the following controls.

- ♦ Set the **Learn Rate** to 0.05.
- ♦ Set the **Number of trees** to 500.
- ♦ Set the **Criterion Determining Number of Trees Optimal for Logistic Model** to **ROC area**.
- ♦ Leave the remaining controls at their defaults (consult with the screenshot below).
- ✓ See the guide on TreeNet for detailed description of all of the available controls.



TreeNet Options

TreeNet Loss Function: Classification/Logistic Binary

Criterion Determining Number of Trees Optimal for Logistic Model
- Cross Entropy (Likelihood)
- [✓ ROC area]
- Lift in given proportion of 0.100
- Classification Accuracy. Classify cases as Target Class if Probability Exceeds 0.500

Overfitting Protection
- Learn Rate: Auto 0.0500
- Subsample fraction: 0.50

Limits
- Number of trees to build: 500
- Maximum nodes per tree: 6
- Terminal node Minimum Training cases: 10

Random
- Predictors per node: 6
- Vary tree sizes randomly (As Poisson)

Influence trimming speed-up
- Total Influence fraction: 0.10

Model Variance-Dispersion component separately

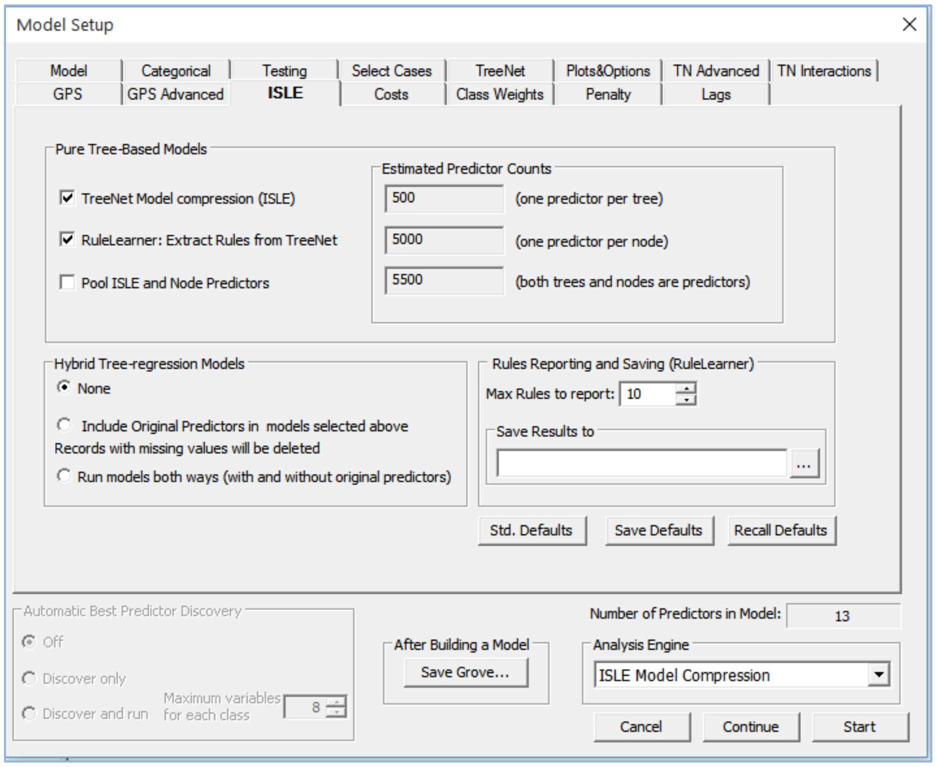
Automatic Best Predictor Discovery
- [✓ Off]
- Discover only
- Discover and run
- Maximum variables for each class: 8

After Building a Model

Number of Predictors in Model: 13

Analysis Engine: ISLE Model Compression

Next you may want to set up the GPS component of the pipeline using the **GPS** and **GPS Advanced** tabs. The default settings in those tabs are good enough for our purposes, you may however, review the GPS guide and later experiment with different number of steps, points, as well as elasticities and learn rates. Finally, switch to the ISLE tab to setup the actual pipeline “linkage” parameters, which could be as simple as specifying whether you want ISLE or RuleLearner compression or both (Pooled ISLE and Node Predictors).


Model Setup

Model | Categorical | Testing | Select Cases | TreeNet | Plots&Options | TN Advanced | TN Interactions
GPS | GPS Advanced | **ISLE** | Costs | Class Weights | Penalty | Lags

Pure Tree-Based Models
- [✓] TreeNet Model compression (ISLE)
- [✓] RuleLearner: Extract Rules from TreeNet
- [ ] Pool ISLE and Node Predictors

Estimated Predictor Counts
- 500 (one predictor per tree)
- 5000 (one predictor per node)
- 5500 (both trees and nodes are predictors)

Hybrid Tree-regression Models
- [✓ None]
- Include Original Predictors in models selected above. Records with missing values will be deleted
- Run models both ways (with and without original predictors)

Rules Reporting and Saving (RuleLearner)
- Max Rules to report: 10
- Save Results to: (empty)

Std. Defaults | Save Defaults | Recall Defaults

Automatic Best Predictor Discovery: [✓ Off] / Discover only / Discover and run; Maximum variables for each class: 8

Number of Predictors in Model: 13

After Building a Model: Save Grove...

Analysis Engine: ISLE Model Compression

Cancel | Continue | Start

Note that we have requested both ISLE and RuleLearner which will trigger two independent compression runs after the initial TreeNet model has been built: one run to try the ISLE compression and the other to try the RuleLearner compression. This gives you the convenience of not building the TreeNet model (the first stage of the pipeline) twice and thus saving the overall run time.

The control options on the **ISLE** tab under the Pure Tree-Based Models section are:

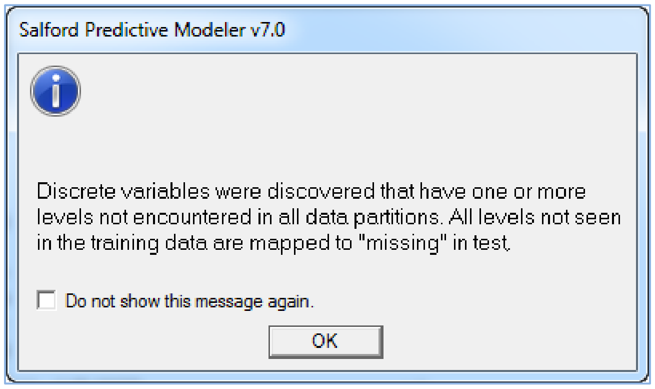
- **TreeNet Model Compression (ISLE)** – uses tree-based variables as intermediate inputs to the GPS compression stage.
- **RuleLearner: Extract Rules from TreeNet** – uses node-based variables as intermediate inputs to the GPS compression stage.
- **Pool ISLE and Node Predictors** – technically a hybrid approach, uses the combined set of both tree-based and node-based variables as intermediate inputs to the GPS compression stage. Note that the end result of this could be a compressed model that has both types of components.
- **Estimated Predictor Counts** – reports the total number of intermediate variables that will be created during the pipeline process. Note that the RuleLearner approach dramatically increase the memory and computational burden of the process, use it judiciously!
- **Hybrid Tree-regression Models** – allows to include the original predictors into the intermediate set of inputs to the GPS compression stage. This sometimes helps to compress the model even further in the presence of strong linear effects.

Keep in mind that each pipeline mode, especially when it involves the node-based variables (RuleLearner) may take significant amount of run time. It is best to start experimenting with a single ISLE or RuleLearner model first and then proceed with more elaborate setups.

- **Max Rules to Report** – sets the default number of the top important rules that will be reported and saved.
- **[Save Results to]** – click this button if you want to save the top important rules into a file.

Press the **[Start]** button to initiate the modeling pipeline.

Depending on your prior system settings, the following window may occur:


Salford Predictive Modeler v7.0

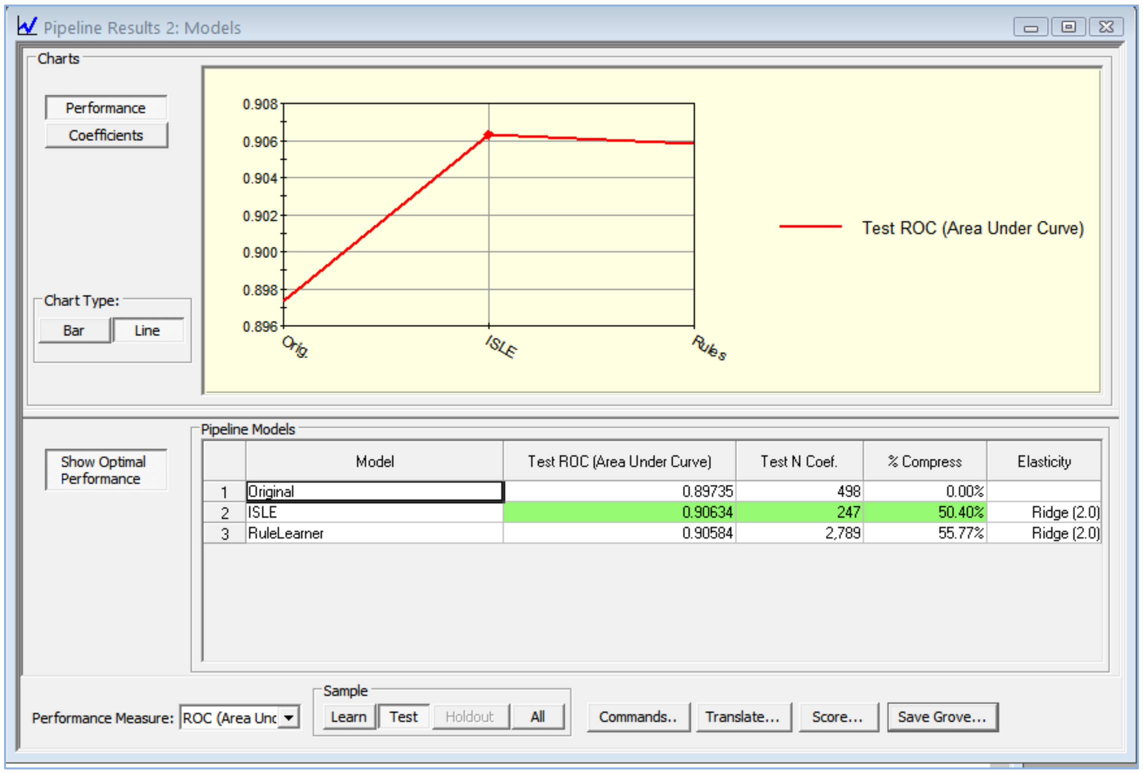
Discrete variables were discovered that have one or more levels not encountered in all data partitions. All levels not seen in the training data are mapped to "missing" in test.

This indicates that some of the categorical predictors available in the dataset had under-represented categories that could not be assigned to both learn and test partitions. You may ignore this warning for now and press the **[OK]** button to proceed with the modeling run.

As the run develops, you will first see the TreeNet engine progress report followed by the GPS engine progress report for the ISLE run and then again for the RuleLearner run. This indicates that the two separate pipelines are being properly executed.

Viewing Model Compression Results

A successfully finished compression run with more than one pipeline will produce the **Pipeline Results: Models** summary window shown below.



	Model	Test ROC (Area Under Curve)	Test N Coef.	% Compress	Elasticity
1	Original	0.89735	498	0.00%	
2	ISLE	0.90634	247	50.40%	Ridge (2.0)
3	RuleLearner	0.90584	2,789	55.77%	Ridge (2.0)

In our case, three different models are reported:

- **Original** – the original best TreeNet model before compression.
- **ISLE** – best model using ISLE (tree-based variables) compression.
- **RuleLearner** – best model using RuleLearner (node-based variables) approach.

The graph area displays the currently selected performance measures for each model.

Note that the original TreeNet model had 498 trees (coefficients) while the ISLE compressed model has only 247 trees (50.40% compression rate) while improving the test sample ROC performance.

Likewise, the original TreeNet model had 4980 nodes (6-node default regression trees used by the TreeNet engine actually have 10 nodes each, counting both terminal and internal nodes and excluding the root node). The RuleLearner compressed model has 2789 selected nodes (55.77% compression rate) while improving the test sample ROC performance as well.

Thus, we have successfully accomplished the original goal of reducing the complexity of the TreeNet model by a significant amount while improving the test sample performance.

The following key controls are located on the results window:

- **Charts** – one can switch between the **[Performance]** and **[Coefficients]** display mode of the graph.
- **Chart Type** – one can switch between the **[Bar]** and **[Line]** style of the graph.
- **[Show Optimal Performance]** – press this button to highlight the best performing model using green background in the table.
- **Performance Measure:** - choose among a variety of different performance measures. Once a new measure is selected, all displays will be updated.
- **Sample –** these buttons select among the learn and test data partitions.
- **[Score…]**, **[Translate…]**, **[Commands…]**, **[Save Grove…]** – these are standard action buttons available in any engine.

Scoring Compressed Model

Press the **[Score…]** button to activate the **Score Data** control window.

Use the **Select model to score:** selection box to pick the model of interest (the Original, ISLE, or RuleLearner).

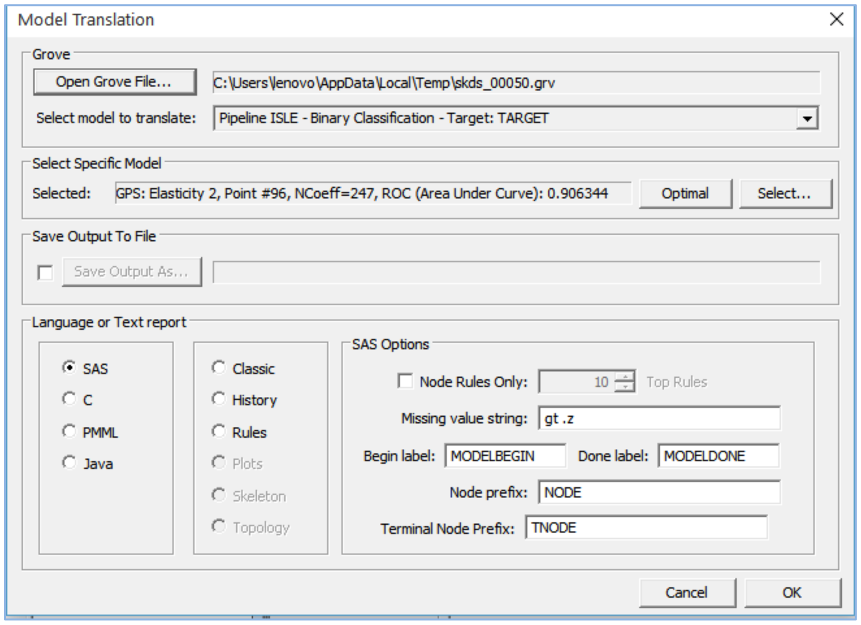
Press the **[Select…]** button in the **Select Specific Model** section if you want to pick a different model on the GPS path for the current compression approach.

The rest of the scoring controls have been described earlier in the manual.

Press the **[Score]** button to initiate the scoring process.

Translating Compressed Model

Press the **[Translate…]** button to activate the **Model Translation** control window.



C:\Users\lenovo\AppData\Local\Temp\skds_00050.grv

Select model to translate: Pipeline ISLE - Binary Classification - Target: TARGET

Selected: GPS: Elasticity 2, Point #96, NCoeff=247, ROC (Area Under Curve): 0.906344

(● SAS) ○ C ○ PMML ○ Java

○ Classic ○ History ○ Rules ○ Plots ○ Skeleton ○ Topology

☐ Node Rules Only: 10 Top Rules

Missing value string: gt .z

Begin label: MODELBEGIN Done label: MODELDONE

Node prefix: NODE

Terminal Node Prefix: TNODE

Use the **Select Specific Model** selection box to pick the model of interest (the Original, ISLE, or RuleLearner).

Press the **[Select…]** button in the **Select Specific Model** section if you want to pick a different model on the GPS path for the current compression approach.

The rest of the translation controls have been described earlier in the manual.

Press the **[OK]** button to initiate the model translation process.

We used the above procedure to translate the 2-tree ISLE model into SAS. An example output code would have the following layout:

```
/* Tree 2 of 276 */

tnscore = -0.844412208; /* GPS ISLE Tree Coefficients */

<Block of code representing the tree logic to generate “response” update for the
current tree>

tnscore = tnscore + 0.126800358 * response;

/* Tree 5 of 276 */

<Block of code representing the tree logic to generate “response” update for the
current tree>
tnscore = tnscore + 1.25642133 * response;
return;
```

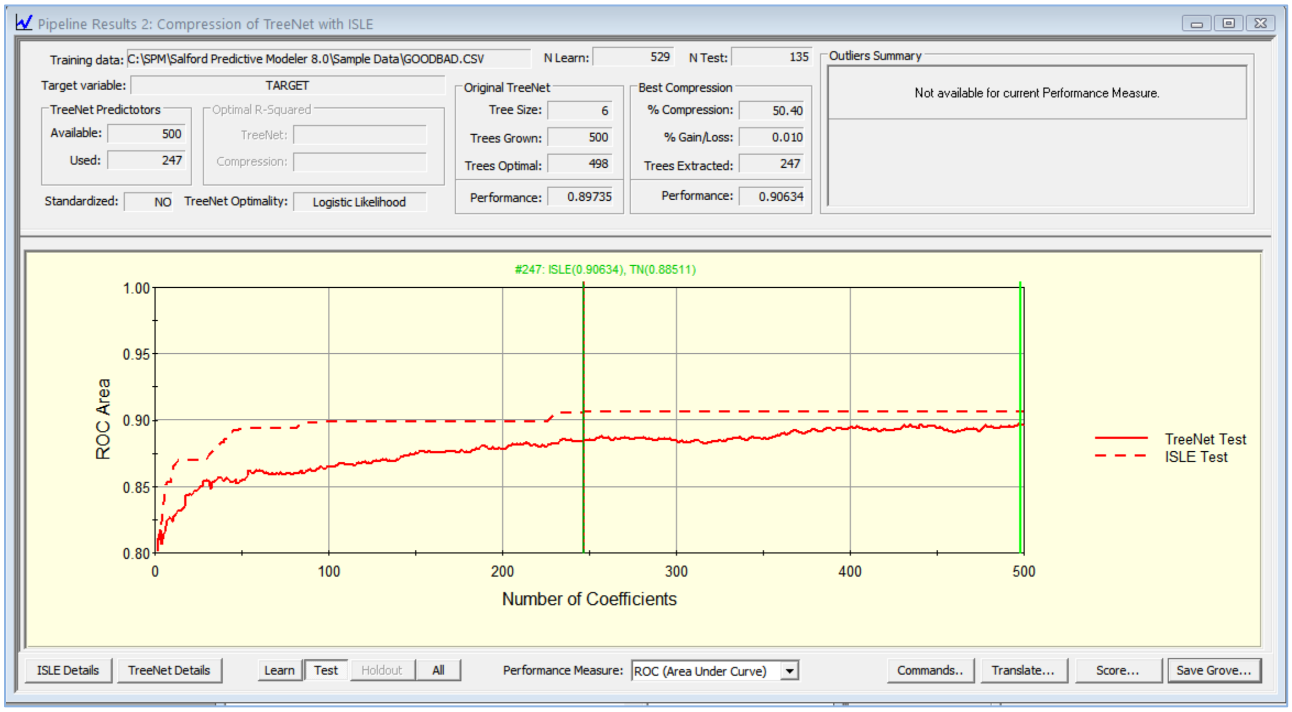
ISLE model translation only includes trees selected by the compression method (non-zero coefficients). Observe how the new updated ISLE coefficients are introduced directly into the model code.

✓ A similar code is produced when translating a RuleLearner model. However, because the RuleLearner model decomposition takes place at the node level, there is no simple way to reduce the model code by eliminating the redundant nodes. As the interim implementation, we now produce the full original TreeNet model code with the node-level adjustments augmented for all nodes selected by the RuleLearner process. Thus the code itself produces the correct RuleLearner scores but it also includes large parts that are unnecessary. This inconvenience has now been fully resolved in the latest release of the RuleLearner component.

Viewing ISLE Model Compression Details

Double click on the ISLE row in the **Pipeline Results: Models** window. This will open the **ISLE** window reporting details of the ISLE compression run.

✓ This window automatically opens up instead of the **Pipeline Results: Models** window when ISLE is the only compression method requested in the **ISLE** tab of the **Model Setup** window.


Pipeline Results 2: Compression of TreeNet with ISLE

Training data: C:\SPM\Salford Predictive Modeler 8.0\Sample Data\GOODBAD.CSV
N Learn: 529
N Test: 135
Target variable: TARGET

TreeNet Predictotors
Available: 500
Used: 247

Optimal R-Squared
TreeNet:
Compression:

Standardized: NO
TreeNet Optimality: Logistic Likelihood

Original TreeNet
Tree Size: 6
Trees Grown: 500
Trees Optimal: 498
Performance: 0.89735

Best Compression
% Compression: 50.40
% Gain/Loss: 0.010
Trees Extracted: 247
Performance: 0.90634

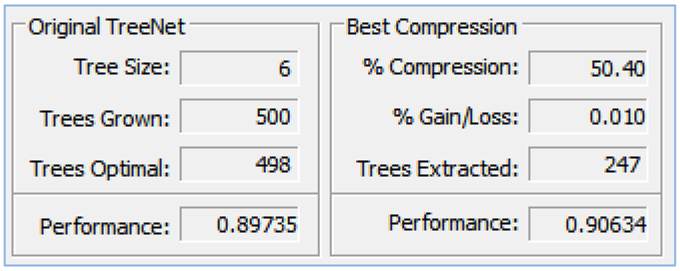
Outliers Summary
Not available for current Performance Measure.

Performance Measure: ROC (Area Under Curve)

This window shows detailed ISLE compression results indexed by model size in trees. The two test sample performance curves are aligned such that one can easily compare the performance of two models with identical size. Note that the 247-tree TreeNet model has test ROC 0.885 while the 247-tree ISLE compressed model has ROC 0.906. Even though both models have identical number of trees, they differ in the actual trees. The TreeNet model takes the first 247consecutive trees from the TreeNet sequence (all combined with the unit coefficients) while the ISLE model takes some specific subset of 247 trees from the original 500 trees available in the sequence, combined with ISLE specific coefficients.

✓ The actual subset of trees can be determined by looking into the ISLE Details or translating the model.

The top part of the window reports basic stats on the optimal TreeNet model versus optimal ISLE model in terms of ROC performance.



Original TreeNet		Best Compression	
Tree Size:	6	% Compression:	50.40
Trees Grown:	500	% Gain/Loss:	0.010
Trees Optimal:	498	Trees Extracted:	247
Performance:	0.89735	Performance:	0.90634

✓ You can switch to alternative performance measures using the Performance Measure: selection box.

Press the **[TreeNet Details]** button to open the usual **TreeNet Output** window described in the TreeNet engine chapter of this manual.

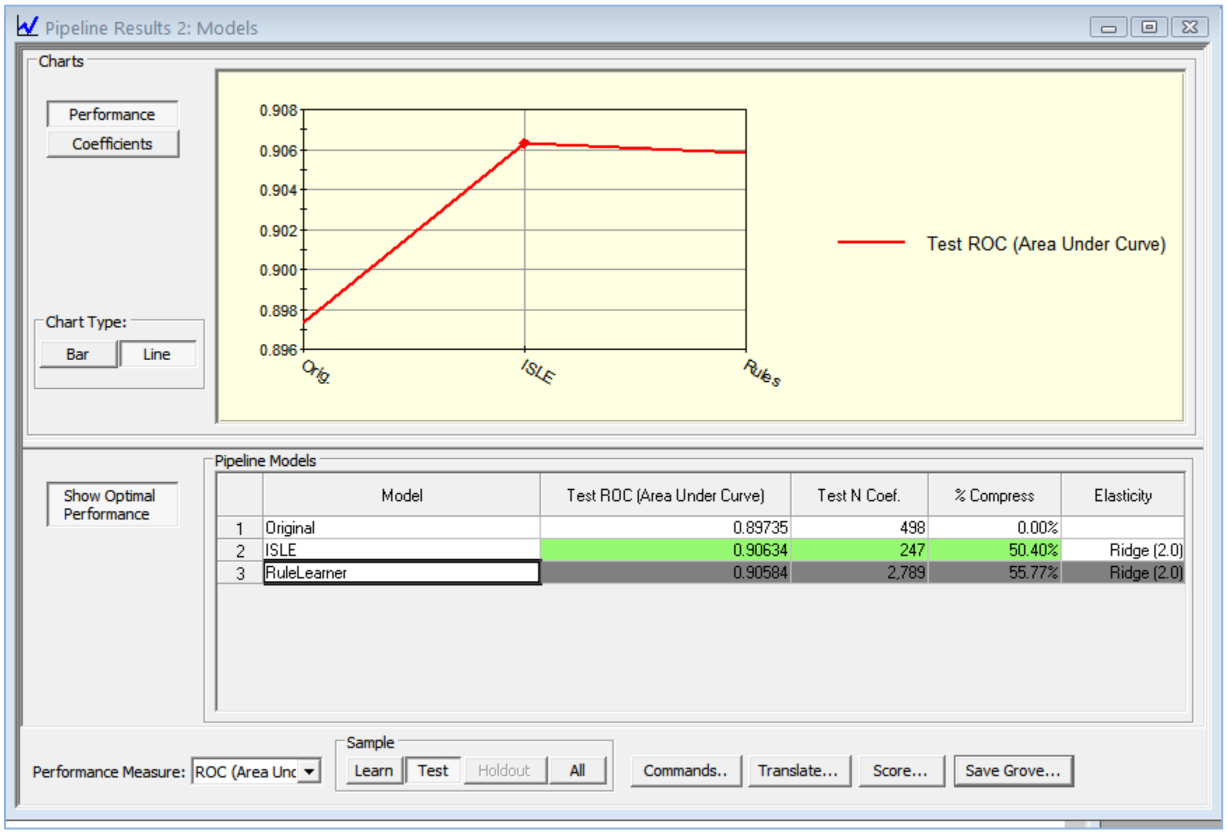
Press the **[ISLE Details]** button to open the usual **GPS Results** window described in the GPS engine chapter of this manual.

✓ Note that the GPS component of the ISLE pipeline uses trees as input variables. Therefore, all parts of the GPS output windows referencing the predictors (Coefficients, etc.) will now report names like Tree 1, Tree 2, Tree 3, etc.

The remaining control items are similar to what we have already described earlier.

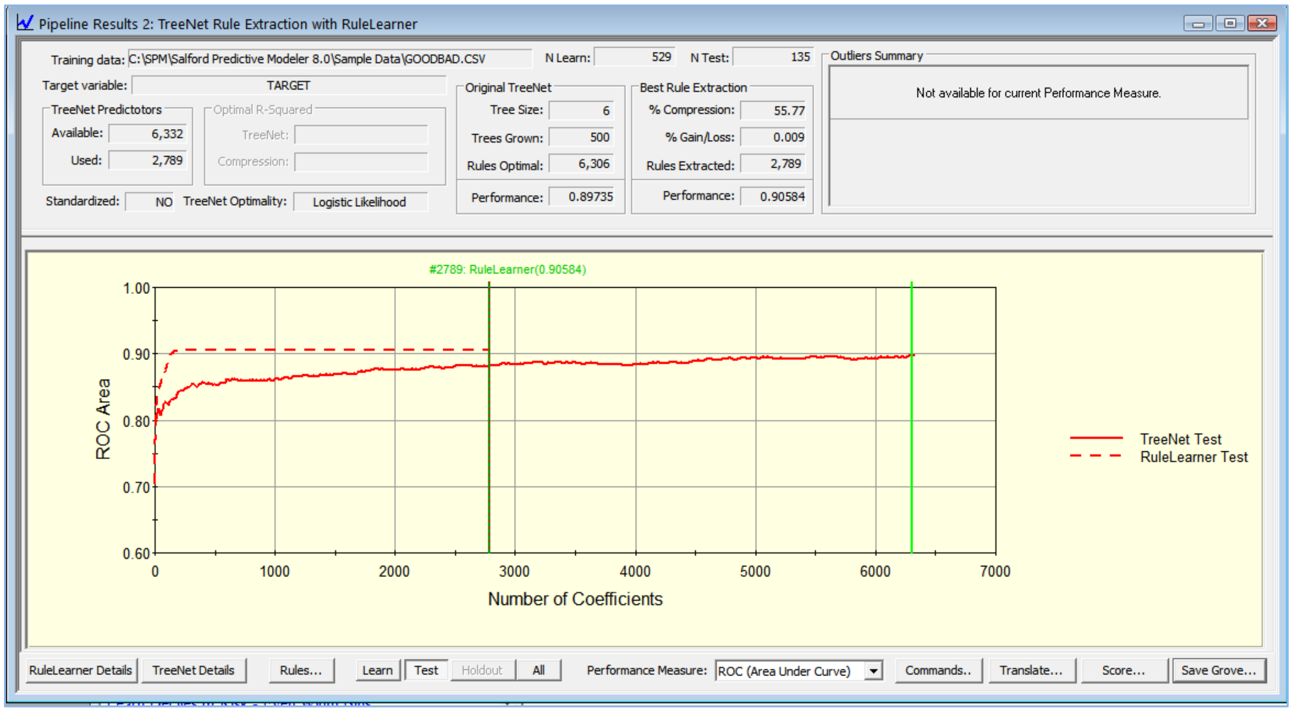
Viewing RuleLearner Model Compression Details

Double click on the RuleLearner row in the **Pipeline Results: Models** window. This will open the **RuleLearner Results: RuleLearner** window reporting details of the RuleLearner compression run.



	Model	Test ROC (Area Under Curve)	Test N Coef.	% Compress	Elasticity
1	Original	0.89735	498	0.00%	
2	ISLE	0.90634	247	50.40%	Ridge (2.0)
3	RuleLearner	0.90584	2,789	55.77%	Ridge (2.0)

✓ This window automatically opens up instead of the **Pipeline Results: Models** window when RuleLearner is the only compression method requested in the **RuleLearner** tab of the **Model Setup** window.


Pipeline Results 2: TreeNet Rule Extraction with RuleLearner

Training data: C:\SPM\Salford Predictive Modeler 8.0\Sample Data\GOODBAD.CSV   N Learn: 529   N Test: 135

Target variable: TARGET

TreeNet Predictotors — Available: 6,332; Used: 2,789

Optimal R-Squared — TreeNet: ; Compression:

Standardized: NO   TreeNet Optimality: Logistic Likelihood

Original TreeNet — Tree Size: 6; Trees Grown: 500; Rules Optimal: 6,306; Performance: 0.89735

Best Rule Extraction — % Compression: 55.77; % Gain/Loss: 0.009; Rules Extracted: 2,789; Performance: 0.90584

Outliers Summary — Not available for current Performance Measure.

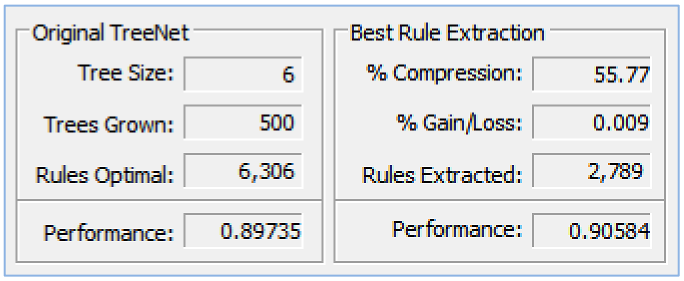
Performance Measure: ROC (Area Under Curve)

This window shows detailed RuleLearner compression results indexed by model size in nodes. The two test sample performance curves are aligned such that one can easily compare the performance of two models with identical size.

Note that the 2789-node TreeNet model has test ROC 0.875 while the 2789-node RuleLearner compressed model has ROC 0.90584. Even though both models have identical number of nodes, they differ in the actual trees selected. The TreeNet model takes the first consecutive trees from the TreeNet sequence (all combined with the unit coefficients and the original node updates) while the RuleLearner model takes some specific subset of 2789 nodes from the original complete set available in the sequence. Furthermore, each selected node has a new associated update.

✓ The actual subset of nodes can be determined by looking into the RuleLearner Details or translating the model.

The top part of the window reports basic stats on the optimal TreeNet model versus optimal RuleLearner model in terms of ROC performance.


Original TreeNet		Best Rule Extraction	
Tree Size:	6	% Compression:	55.77
Trees Grown:	500	% Gain/Loss:	0.009
Rules Optimal:	6,306	Rules Extracted:	2,789
Performance:	0.89735	Performance:	0.90584

✓ You can switch to alternative performance measures using the Performance Measure: selection box.

Press the **[TreeNet Details]** button to open the usual **TreeNet Output** window described in the TreeNet engine chapter of this manual.

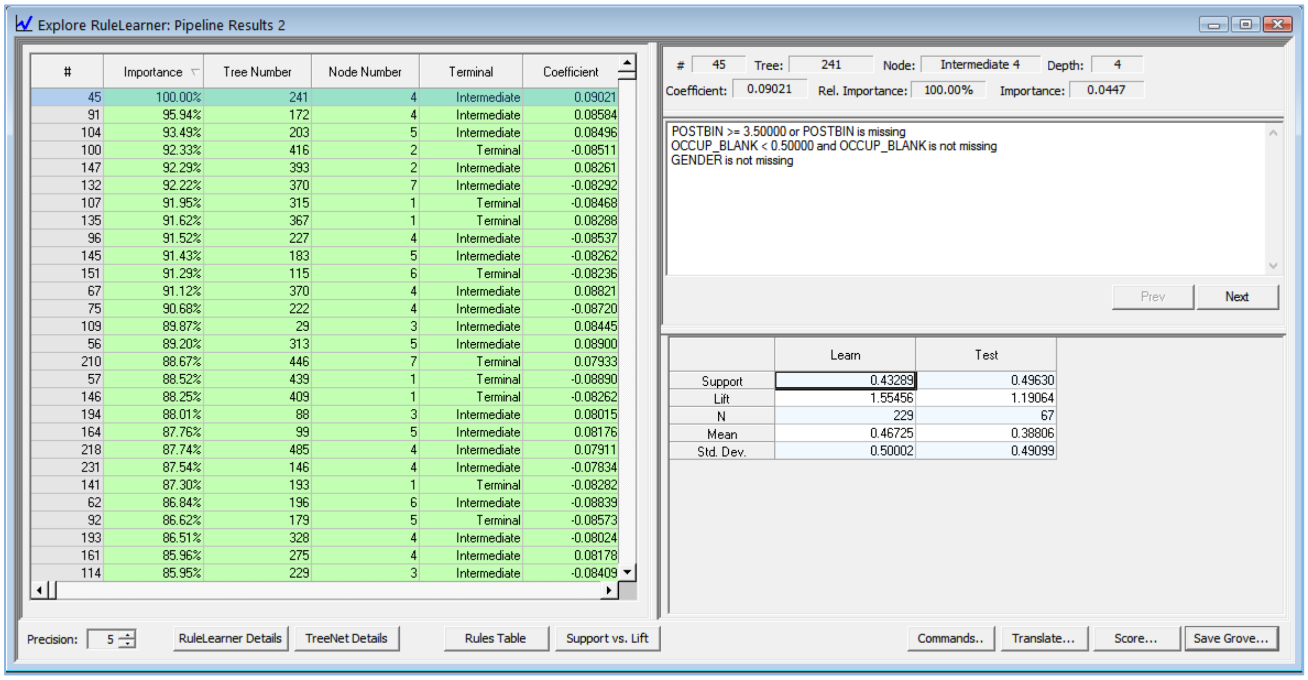
Press the **[RuleLearner Details]** button to open the usual **GPS Results** window described in the GPS engine chapter of this manual.

✓ Note that the GPS component of the RuleLearner pipeline uses nodes as input variables. Therefore, all parts of the GPS output windows referencing the predictors (Coefficients, etc.) will now report names like Tree 1 Node 1, Tree 1 TNode 1, Tree 2 Node 1, etc.

The remaining control items are similar to what we have already described earlier except for the **[Rules…]** button which opens up a RuleLearner specific **Explore** window.

Exploring the Rules

Press the **[Rules…]** button in the **Pipeline Results: RuleLearner** window to open the **Explore RuleLearner** window shown below.



#	Importance	Tree Number	Node Number	Terminal	Coefficient
45	100.00%	241	4	Intermediate	0.09021
91	95.94%	172	4	Intermediate	0.08584
104	93.49%	203	5	Intermediate	0.08496
100	92.33%	416	2	Terminal	-0.08511
147	92.29%	393	2	Intermediate	0.08261
132	92.22%	370	7	Intermediate	-0.08292
107	91.95%	315	1	Terminal	-0.08468
135	91.62%	367	1	Terminal	0.08288
96	91.52%	227	4	Intermediate	-0.08537
145	91.43%	183	5	Intermediate	-0.08262
151	91.29%	115	6	Terminal	-0.08236
67	91.12%	370	4	Intermediate	0.08821
75	90.68%	222	4	Intermediate	-0.08720
109	89.87%	29	3	Intermediate	0.08445
56	89.20%	313	5	Intermediate	0.08900
210	88.67%	446	7	Terminal	0.07933
57	88.52%	439	1	Terminal	-0.08890
146	88.25%	409	1	Terminal	-0.08262
194	88.01%	88	3	Intermediate	0.08015
164	87.76%	99	5	Intermediate	0.08176
218	87.74%	485	4	Intermediate	0.07911
231	87.54%	146	4	Intermediate	-0.07834
141	87.30%	193	1	Terminal	-0.08282
62	86.84%	196	6	Intermediate	-0.08839
92	86.62%	179	5	Terminal	-0.08573
193	86.51%	328	4	Intermediate	-0.08024
161	85.96%	275	4	Intermediate	0.08178
114	85.95%	229	3	Intermediate	-0.08409

#: 45 Tree: 241 Node: Intermediate 4 Depth: 4
Coefficient: 0.09021 Rel. Importance: 100.00% Importance: 0.0447

```
POSTBIN >= 3.50000 or POSTBIN is missing
OCCUP_BLANK < 0.50000 and OCCUP_BLANK is not missing
GENDER is not missing
```

	Learn	Test
Support	0.43289	0.49630
Lift	1.55456	1.19064
N	229	67
Mean	0.46725	0.38806
Std. Dev.	0.50002	0.49099

Recall that the RuleLearner compression method decomposes a TreeNet model into a collection of individual nodes (internal and terminal) and then harvests individual nodes using GPS. Each node represents a segment of the original data described by a collection of rules leading into this node. Hence, analyzing the rules associated with the top selected nodes might provide additional modeling insights.

The nodes are listed on the left part of the **Explore** window according to the order of their importance (as reported by the GPS component). The actual rules associated with the currently selected node (segment) are shown on the top right part while the basic stats about the node (segment) are shown on the bottom right part of the window.

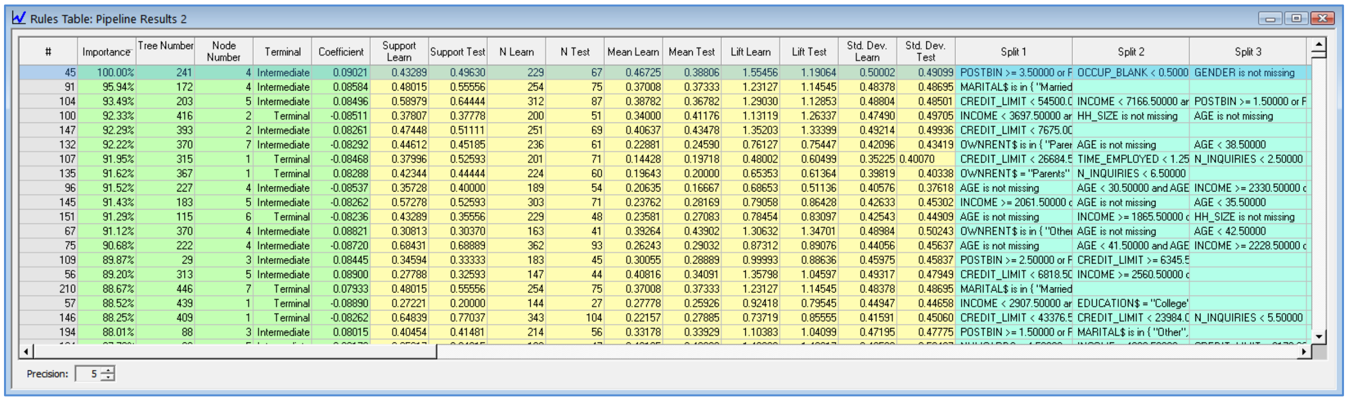
Important Rule Metrics

Note the use of the following terms generally accepted in the associative rules research community:

- **Support** – fraction of the sample (either learn or test) covered by the segment.
- **Lift** – segment mean of the target divided by the overall sample mean of the target.
- ✓ In the case of binary target, this is the same as segment fraction of 1s divided by the sample fraction of 1s.
- **Mean, Std. Dev.** – the usual statistical definitions of the mean and standard deviation in the segment of interest.

Click on the [Rules Table] button to open up even more detailed information about each node. The resulting table can be sorted by clicking on any of its columns. This allows quick and easy identification of the potential rules of interest. For example, you may focus on finding the segments having the highest test sample lift or segments having the highest support and so on.

For example, internal node 4 of tree 241 is at the top of the importance list, it covers almost 50% of the available data (very large support) and is described by a joint collection of rules involving POST_BIN, OCCUP_BLANK, and GENDER.

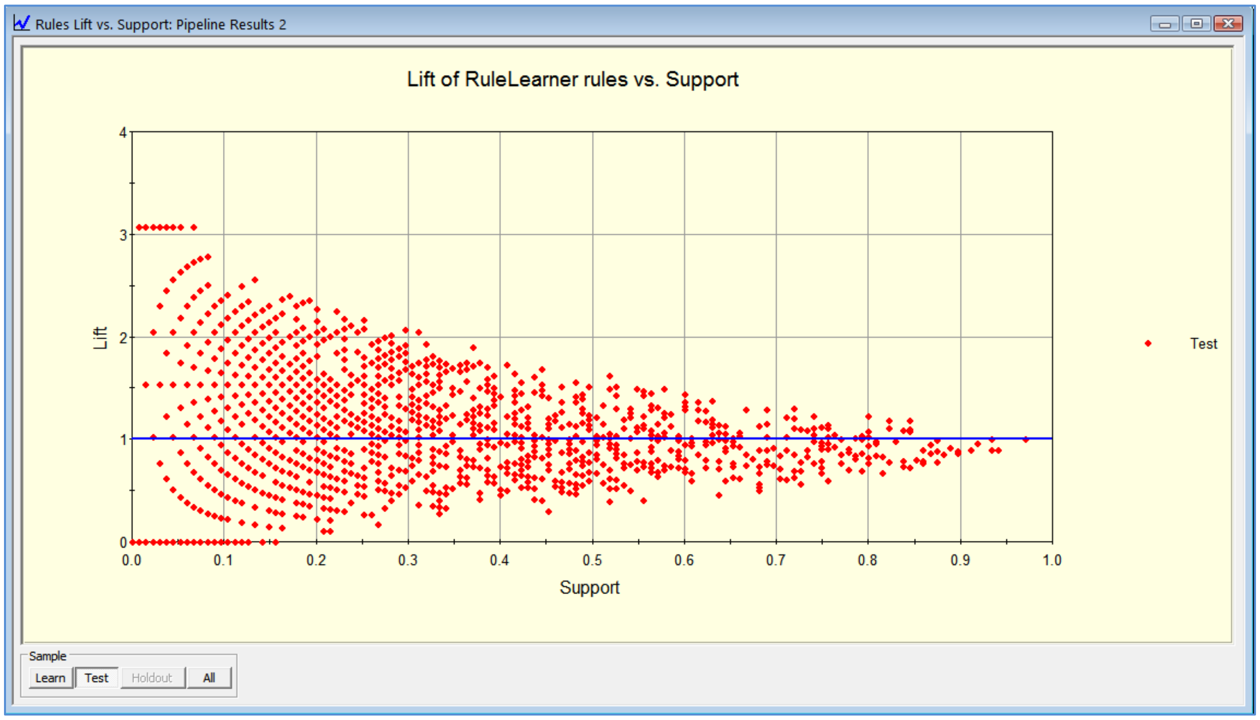


Rules Table: Pipeline Results 2

#	Importance	Tree Number	Node Number	Terminal	Coefficient	Support Learn	Support Test	N Learn	N Test	Mean Learn	Mean Test	Lift Learn	Lift Test	Std. Dev. Learn	Std. Dev. Test	Split 1	Split 2	Split 3
45	100.00%	241	4	Intermediate	0.09021	0.43289	0.49630	229	67	0.46725	0.38806	1.55456	1.19064	0.50002	0.49099	POSTBIN >= 3.50000 or F	OCCUP_BLANK < 0.5000	GENDER is not missing
91	95.94%	172	4	Intermediate	0.08584	0.48015	0.55556	254	75	0.37008	0.37333	1.23127	1.14545	0.48378	0.48695	MARITAL$ is in { "Married		
104	93.49%	203	5	Intermediate	0.08496	0.58979	0.64444	312	87	0.38782	0.36782	1.29030	1.12853	0.48804	0.48501	CREDIT_LIMIT < 54500.0	INCOME < 7166.50000 ar	POSTBIN >= 1.50000 or F
100	92.33%	416	2	Terminal	-0.08511	0.37807	0.37778	200	51	0.34000	0.41176	1.13119	1.26337	0.47490	0.49705	INCOME < 3697.50000 ar	HH_SIZE is not missing	AGE is not missing
147	92.29%	393	2	Intermediate	0.08261	0.47448	0.51111	251	69	0.40637	0.43478	1.35203	1.33399	0.49214	0.49936	CREDIT_LIMIT < 7675.00		
132	92.22%	370	7	Intermediate	-0.08292	0.44612	0.45185	236	61	0.22881	0.24590	0.76127	0.75447	0.42096	0.43419	OWNRENT$ is in { "Parer	AGE is not missing	AGE < 38.50000
107	91.95%	315	1	Terminal	-0.08468	0.37996	0.52593	201	71	0.14428	0.19718	0.48002	0.60499	0.35225	0.40070	CREDIT_LIMIT < 26684.5	TIME_EMPLOYED < 1.25	N_INQUIRIES < 2.50000
135	91.62%	367	1	Terminal	0.08288	0.42344	0.44444	224	60	0.19643	0.20000	0.65353	0.61364	0.39819	0.40338	OWNRENT$ = "Parents"	N_INQUIRIES < 6.50000	
96	91.52%	227	4	Intermediate	-0.08537	0.35728	0.40000	189	54	0.20635	0.16667	0.68653	0.51136	0.40576	0.37618	AGE is not missing	AGE < 30.50000 and AGE	INCOME >= 2330.50000 c
145	91.43%	183	5	Intermediate	-0.08262	0.57278	0.52593	303	71	0.23762	0.28169	0.79058	0.86428	0.42633	0.45302	INCOME >= 2061.50000 c	AGE is not missing	AGE < 35.50000
151	91.29%	115	6	Terminal	-0.08236	0.43289	0.35556	229	48	0.23581	0.27083	0.78454	0.83097	0.42543	0.44909	AGE is not missing	INCOME >= 1865.50000 c	HH_SIZE is not missing
67	91.12%	370	4	Intermediate	0.08821	0.30813	0.30370	163	41	0.39264	0.43902	1.30632	1.34701	0.48984	0.50243	OWNRENT$ is in { "Other	AGE is not missing	AGE < 42.50000
75	90.68%	222	4	Intermediate	-0.08720	0.68431	0.68889	362	93	0.26243	0.29032	0.87312	0.89076	0.44056	0.45637	AGE is not missing	AGE < 41.50000 and AGE	INCOME >= 2228.50000 c
109	89.87%	29	3	Intermediate	0.08445	0.34594	0.33333	183	45	0.30055	0.28889	0.99993	0.88636	0.45975	0.45837	POSTBIN >= 2.50000 or F	CREDIT_LIMIT >= 6345.5	
56	89.20%	313	5	Intermediate	0.08900	0.27788	0.32593	147	44	0.40816	0.34091	1.35798	1.04597	0.49317	0.47949	CREDIT_LIMIT < 6818.50	INCOME >= 2560.50000 c	
210	88.67%	446	7	Terminal	0.07933	0.48015	0.55556	254	75	0.37008	0.37333	1.23127	1.14545	0.48378	0.48695	MARITAL$ is in { "Married		
57	88.52%	439	1	Terminal	-0.08890	0.27221	0.20000	144	27	0.27778	0.25926	0.92418	0.79545	0.44947	0.44658	INCOME < 2907.50000 ar	EDUCATION$ = "College'	
146	88.25%	409	1	Terminal	-0.08262	0.64839	0.77037	343	104	0.22157	0.27885	0.73719	0.85555	0.41591	0.45060	CREDIT_LIMIT < 43376.5	CREDIT_LIMIT < 23984.0	N_INQUIRIES < 5.50000
194	88.01%	88	3	Intermediate	0.08015	0.40454	0.41481	214	56	0.33178	0.33929	1.10383	1.04099	0.47195	0.47775	POSTBIN >= 1.50000 or F	MARITAL$ is in { "Other",	

Precision: 5

Finally, click on the **[Support vs. Lift]** button to see the graphical representation of the usual tradeoff involved. Note that the larger support segments tend to have smaller lift.



This concludes our discussion of the ISLE model compression approach now available in SPM.

RuleLearner Example

At this point it will suffice to say that setting up a model compression run using ISLE from a user standpoint is very straightforward. We are essentially creating a "pipeline", a sequence of two already introduced engines, first a TreeNet engine using the original predictors followed by a GPS engine that uses a decomposed TreeNet model as its own inputs. The decomposition itself is done automatically by SPM based on the user-selected method, which could be either ISLE or RuleLearner (as well as additional trivial modifications introduced below). In addition, it is assumed that each engine component of the pipeline (TreeNet and GPS) is configured using its own set of control parameters as usual.

The end result of the compression pipeline is the overall compression summary plus the usual output of the TreeNet engine (as if it were run by itself) as well as the standard GPS output of the follow-up compression phase.

The output displays are deliberately designed for ease of comparison between the original uncompressed model and the resulting compressed model. The user can choose any desired model for viewing, scoring, and translation using the "point and click" approach thus balancing the degree of compression versus model size.

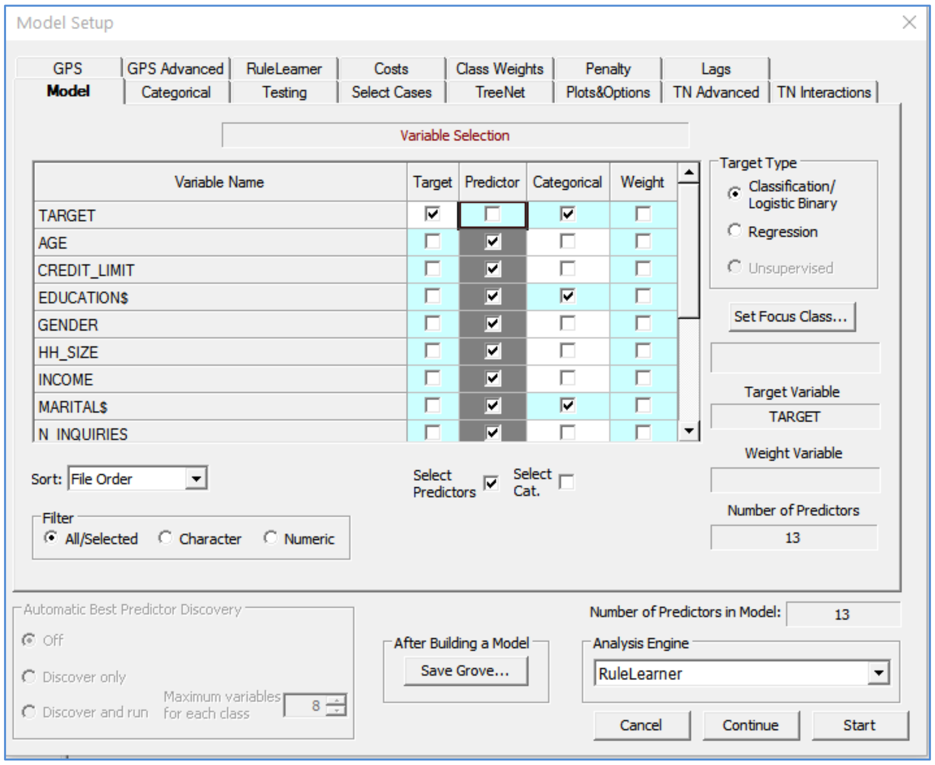
Setting up RuleLearner

Again, we start with the GOODBAD.CSV data set which is rather on the small side but runs very quickly and illustrates our main points well. The dataset comes from a credit default study and has 664 observations and 14 variables including a binary response variable named TARGET, with 1 indicating a bad credit standing account and 0 otherwise. The remaining predictors provide a nice mix of continuous and categorical measures which represent a typical modeling environment. The goal of the study is to build a predictive model for bad credit using the TreeNet approach and then possibly compress the model using either ISLE or RuleLearner approaches.

Use the **File/Open/Data File…** menu item to open the GOODBAD.CSV dataset, then click the **[Model…]** button in the resulting activity window to open the **Model Setup** window.

In the Model tab:

- Set **Analysis Engine** to **RuleLearner** – this activates the pipeline part of the SPM – the **RuleLearner** tab. Note that this mode activates the **TreeNet** tab and the **GPS** and **GPS Advanced** tabs thus giving you full control over other parts of the pipeline.
- Set **Sort:** to **File Order** – this sorts all variables according to the position in the dataset.
- Check TARGET variable as the **Target**.
- Check all remaining variables as predictors.
- Select **Target Type** to **Classification/Logistic Binary** to take advantage of the natural capability of both TreeNet and GPS engine to handle binary targets effectively.


Model Setup

GPS | GPS Advanced | RuleLearner | Costs | Class Weights | Penalty | Lags
Model | Categorical | Testing | Select Cases | TreeNet | Plots&Options | TN Advanced | TN Interactions

Variable Selection

Variable Name	Target	Predictor	Categorical	Weight
TARGET	✓	☐	✓	☐
AGE	☐	✓	☐	☐
CREDIT_LIMIT	☐	✓	☐	☐
EDUCATION$	☐	✓	✓	☐
GENDER	☐	✓	☐	☐
HH_SIZE	☐	✓	☐	☐
INCOME	☐	✓	☐	☐
MARITAL$	☐	✓	✓	☐
N_INQUIRIES	☐	✓	☐	☐

Sort: File Order

Select Predictors ✓   Select Cat. ☐

Filter: [● All/Selected] ○ Character ○ Numeric

Target Type: [● Classification/ Logistic Binary] ○ Regression ○ Unsupervised

Set Focus Class…

Target Variable: TARGET

Weight Variable:

Number of Predictors: 13

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: 13

Analysis Engine: RuleLearner

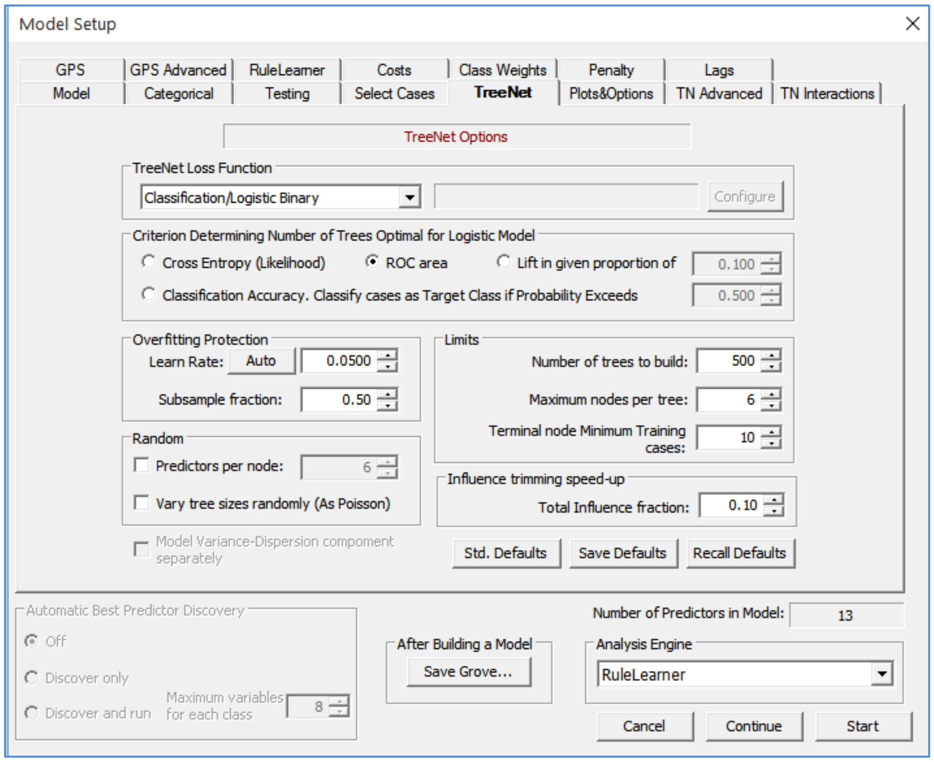
Cancel   Continue   Start

Switch to the Testing tab, and make sure that **Fraction of cases selected at random for testing** is set to 0.2.

We now need to set up the TreeNet component of the pipeline, as if you were setting up a stand-alone TreeNet model.

- Set the **Learn Rate** to 0.05.
- Set the **Number of trees** to 500.
- Set **the Optimal Logistic Model Selection Criterion** to **ROC area**.
- Leave the remaining controls at their defaults (consult with the screenshot below).

✓ See the TreeNet chapter for detailed description of all of the available controls.


Model Setup

GPS | GPS Advanced | RuleLearner | Costs | Class Weights | Penalty | Lags
Model | Categorical | Testing | Select Cases | TreeNet | Plots&Options | TN Advanced | TN Interactions

TreeNet Options

TreeNet Loss Function: Classification/Logistic Binary

Criterion Determining Number of Trees Optimal for Logistic Model: ( ) Cross Entropy (Likelihood) (●) ROC area ( ) Lift in given proportion of 0.100 ( ) Classification Accuracy. Classify cases as Target Class if Probability Exceeds 0.500

Overfitting Protection: Learn Rate: Auto 0.0500; Subsample fraction: 0.50

Limits: Number of trees to build: 500; Maximum nodes per tree: 6; Terminal node Minimum Training cases: 10

Random: [ ] Predictors per node: 6; [ ] Vary tree sizes randomly (As Poisson)

Influence trimming speed-up: Total Influence fraction: 0.10

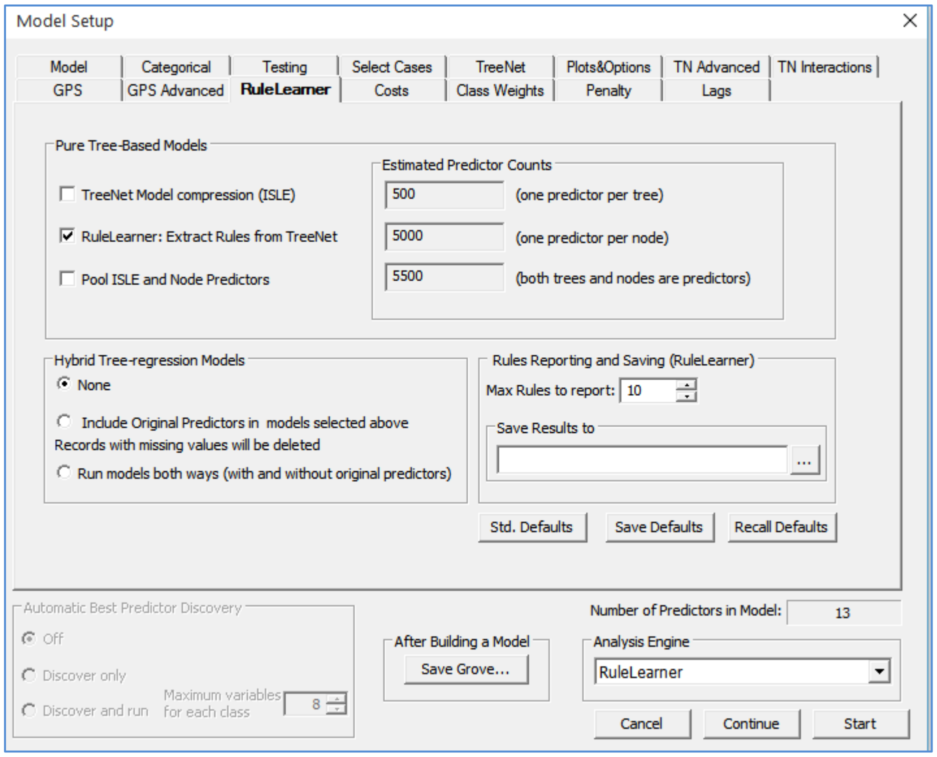
[ ] Model Variance-Dispersion component separately

Automatic Best Predictor Discovery: (●) Off ( ) Discover only ( ) Discover and run; Maximum variables for each class: 8

Number of Predictors in Model: 13

Analysis Engine: RuleLearner

Next you may want to set up the GPS component of the pipeline using the **GPS** and **GPS Advanced** tabs. The default settings in those tabs are good enough for our purposes, you may however, review the GPS part of the manual and later experiment with different number of steps, points, as well as elasticities and learn rates. Finally, switch to the ISLE tab to setup the actual pipeline "linkage" parameters, which could be as simple as specifying whether you want ISLE or RuleLearner compression or both (Pooled ISLE and Node Predictors).



Pure Tree-Based Models
- [ ] TreeNet Model compression (ISLE)
- [✓] RuleLearner: Extract Rules from TreeNet
- [ ] Pool ISLE and Node Predictors

Estimated Predictor Counts

500	(one predictor per tree)
5000	(one predictor per node)
5500	(both trees and nodes are predictors)

Hybrid Tree-regression Models
- [✓] None
- Include Original Predictors in models selected above Records with missing values will be deleted
- Run models both ways (with and without original predictors)

Rules Reporting and Saving (RuleLearner)
Max Rules to report: 10
Save Results to

Automatic Best Predictor Discovery
- [✓] Off
- Discover only
- Discover and run

Maximum variables for each class: 8

Number of Predictors in Model: 13

Analysis Engine: RuleLearner

Note that we have requested only the RuleLearner rule extraction feature which will trigger only one independent compression runs after the initial TreeNet model has been built: you could always try one run for the ISLE compression and the other to try the RuleLearner compression like what we showed earlier for the ISLE algorithm which would give you the convenience of not building the TreeNet model (the first stage of the pipeline) twice and thus saving the overall run time.

The control options on the **RuleLearner** tab under the Pure Tree-Based Models section are:

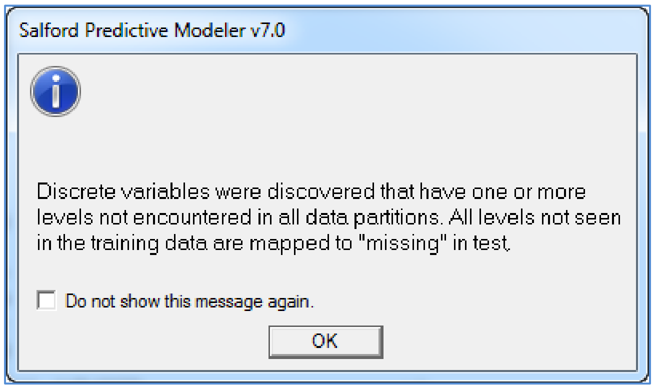
- **TreeNet Model Compression (ISLE)** – uses tree-based variables as intermediate inputs to the GPS compression stage.
- **RuleLearner: Extract Rules from TreeNet** – uses node-based variables as intermediate inputs to the GPS compression stage.
- **Pool ISLE and Node Predictors** – technically a hybrid approach, uses the combined set of both tree-based and node-based variables as intermediate inputs to the GPS compression stage. Note that the end result of this could be a compressed model that has both types of components.
- **Estimated Predictor Counts** – reports the total number of intermediate variables that will be created during the pipeline process. Note that the RuleLearner approach dramatically increase the memory and computational burden of the process, use it judiciously!
- **Hybrid Tree-regression Models** – allows to include the original predictors into the intermediate set of inputs to the GPS compression stage. This sometimes helps to compress the model even further in the presence of strong linear effects.

Keep in mind that each pipeline mode, especially when it involves the node-based variables (RuleLearner) may take significant amount of run time. It is best to start experimenting with a single ISLE or RuleLearner model first and then proceed with more elaborate setups.

- **Max Rules to Report** – sets the default number of the top important rules that will be reported and saved.
- **[Save Results to]** – click this button if you want to save the top important rules into a file.

Press the **[Start]** button to initiate the modeling pipeline.

Depending on your prior system settings, the following window may occur:



Salford Predictive Modeler v7.0

Discrete variables were discovered that have one or more levels not encountered in all data partitions. All levels not seen in the training data are mapped to "missing" in test.

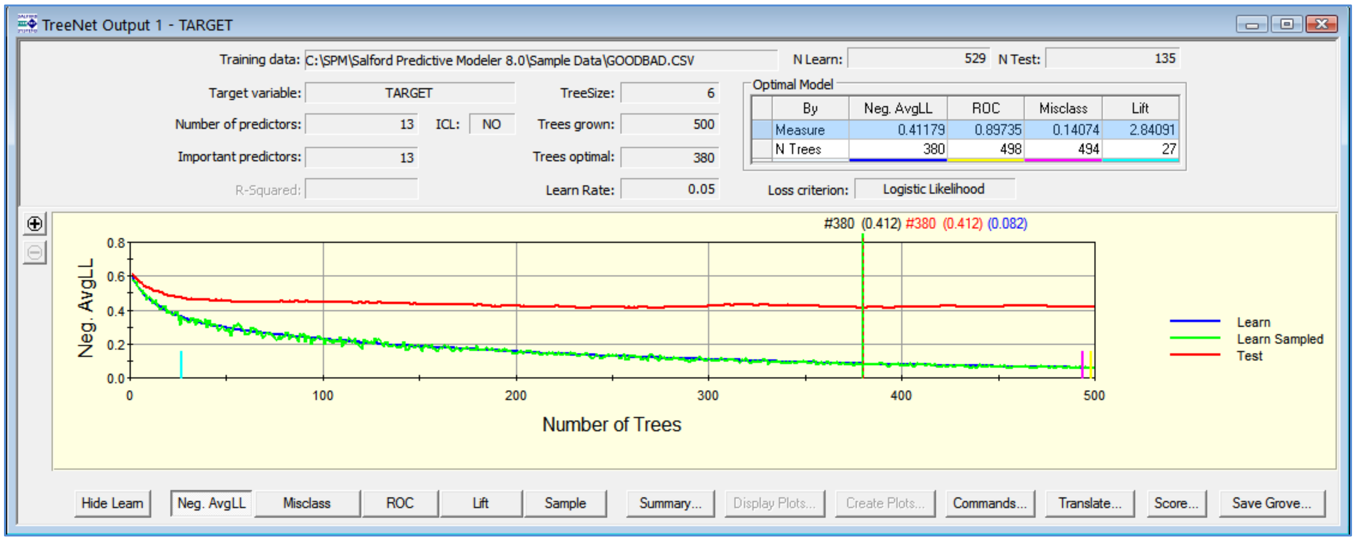
☐ Do not show this message again.

This indicates that some of the categorical predictors available in the dataset had under-represented categories that could not be assigned to both learn and test partitions. You may ignore this warming for now and press the **[OK]** button to proceed with the modeling run.

As the run develops, you will first see the TreeNet engine progress report followed by the GPS engine progress report for the ISLE run and then again for the RuleLearner run. This indicates that the two separate pipelines are being properly executed.

Viewing RuleLearner Results

A successfully finished compression run with one pipeline will produce the **Pipeline Results: TreeNet Rule Extraction with RuleLearner** Results**:** Model summary window shown below:


TreeNet Output 1 - TARGET

Training data: C:\SPM\Salford Predictive Modeler 8.0\Sample Data\GOODBAD.CSV
N Learn: 529 N Test: 135

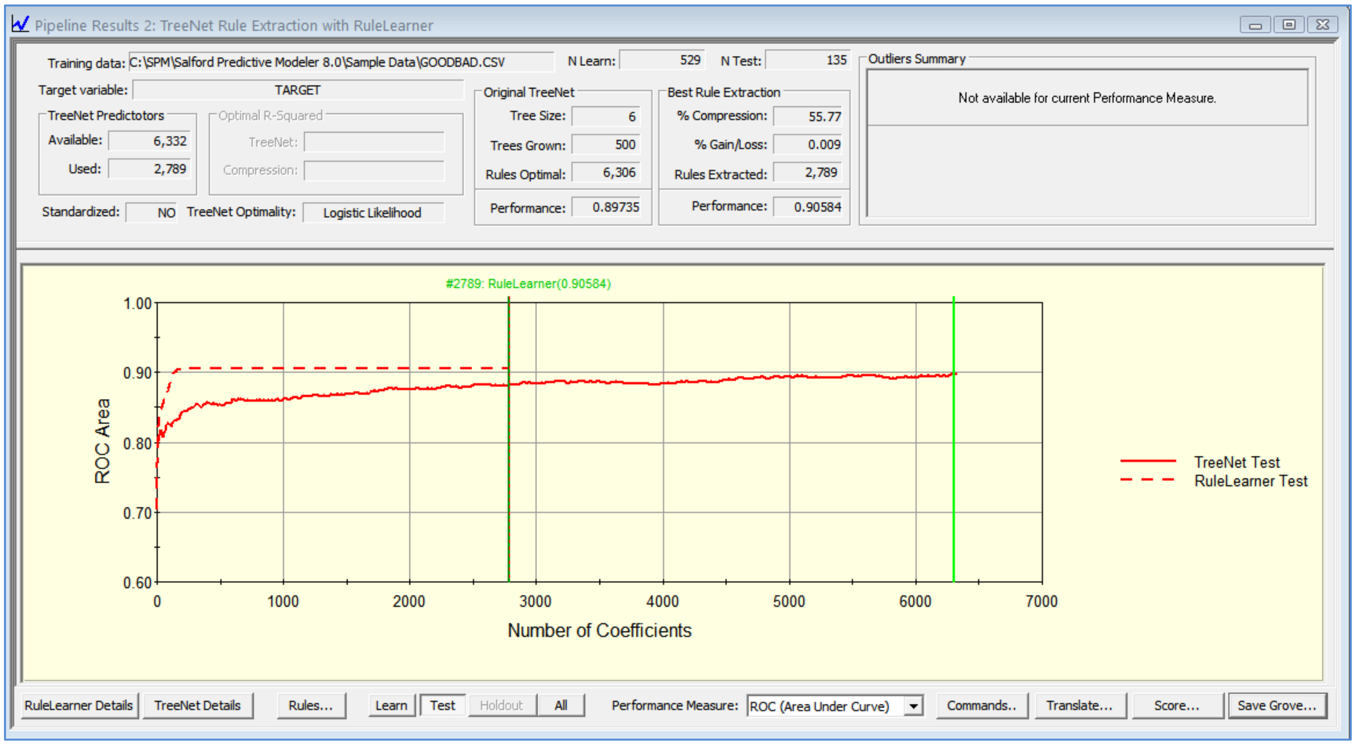
Target variable: TARGET
Number of predictors: 13 ICL: NO
Important predictors: 13
R-Squared:

TreeSize: 6
Trees grown: 500
Trees optimal: 380
Learn Rate: 0.05

Optimal Model

By	Neg. AvgLL	ROC	Misclass	Lift
Measure	0.41179	0.89735	0.14074	2.84091
N Trees	380	498	494	27

Loss criterion: Logistic Likelihood


Pipeline Results 2: TreeNet Rule Extraction with RuleLearner

Training data: C:\SPM\Salford Predictive Modeler 8.0\Sample Data\GOODBAD.CSV
N Learn: 529 N Test: 135
Target variable: TARGET

TreeNet Predictotors
Available: 6,332
Used: 2,789

Optimal R-Squared
TreeNet:
Compression:

Standardized: NO TreeNet Optimality: Logistic Likelihood

Original TreeNet
Tree Size: 6
Trees Grown: 500
Rules Optimal: 6,306
Performance: 0.89735

Best Rule Extraction
% Compression: 55.77
% Gain/Loss: 0.009
Rules Extracted: 2,789
Performance: 0.90584

Outliers Summary
Not available for current Performance Measure.

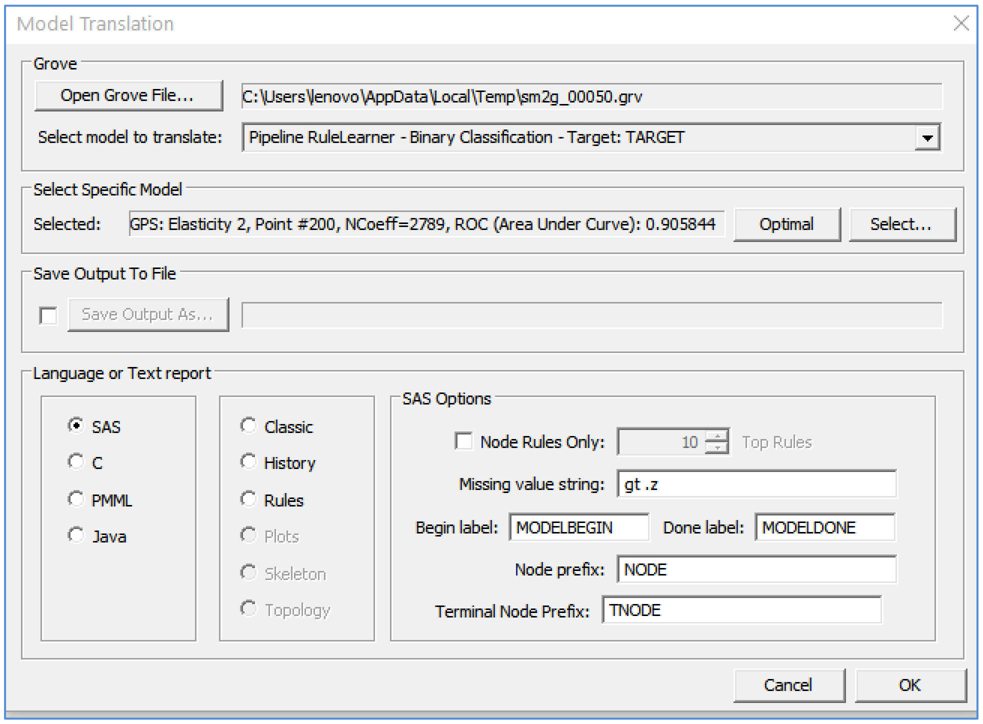
Performance Measure: ROC (Area Under Curve)

In our case, two different models are reported:

- **Original** – the original best TreeNet model before compression.
- **RuleLearner** – best model using RuleLearner (node-based variables) approach.

Translating RuleLearner

Press the **[Translate…]** button to activate the **RuleLearner Model Translation** control window.



Model Translation

Grove

Open Grove File… C:\Users\lenovo\AppData\Local\Temp\sm2g_00050.grv

Select model to translate: Pipeline RuleLearner - Binary Classification - Target: TARGET

Select Specific Model

Selected: GPS: Elasticity 2, Point #200, NCoeff=2789, ROC (Area Under Curve): 0.905844 Optimal Select…

Save Output To File

☐ Save Output As…

Language or Text report

[✓ SAS] C PMML Java

Classic History Rules Plots Skeleton Topology

SAS Options

☐ Node Rules Only: 10 Top Rules

Missing value string: gt .z

Begin label: MODELBEGIN Done label: MODELDONE

Node prefix: NODE

Terminal Node Prefix: TNODE

Cancel OK

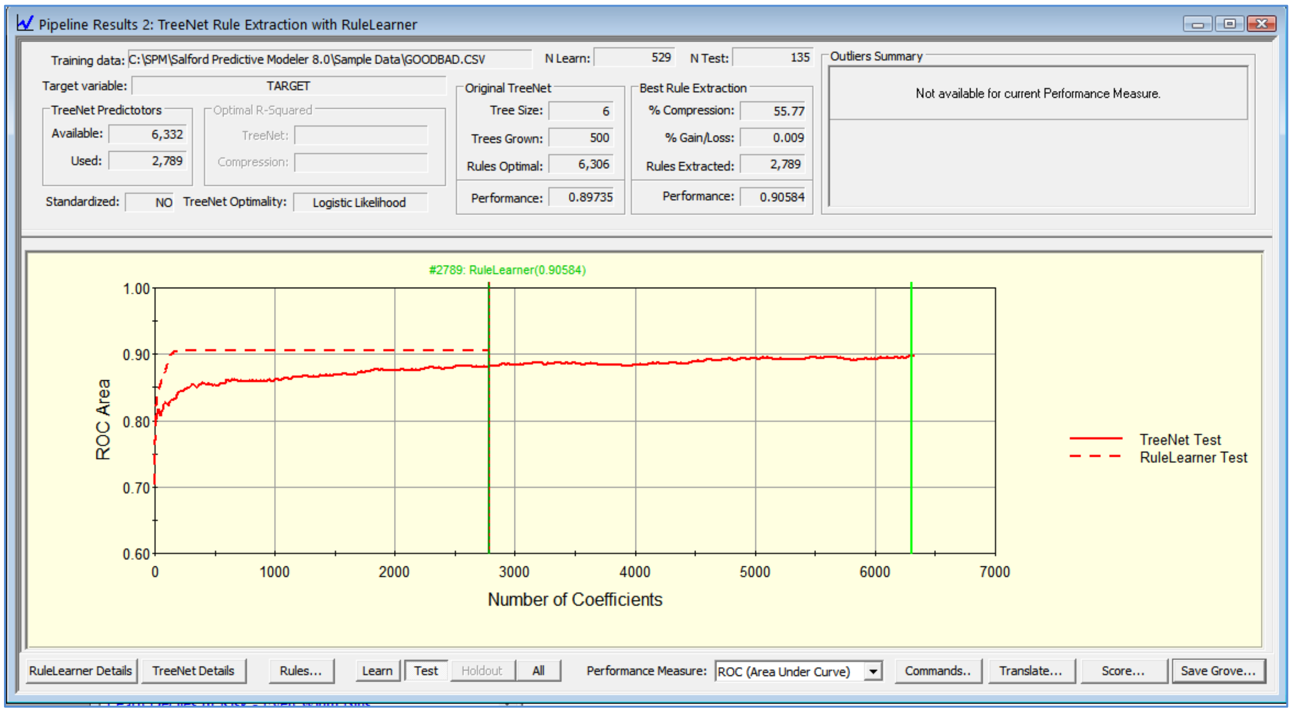
Use the **Select Specific Model** selection box to pick the model of interest (the Original, ISLE, or RuleLearner).

Press the **[Select…]** button in the **Select Specific Model** section if you want to pick a different model on the GPS path for the current compression approach.

The rest of the translation controls have been described earlier in the manual.

Press the **[OK]** button to initiate the model translation process.

Viewing RuleLearner Details



Training data: C:\SPM\Salford Predictive Modeler 8.0\Sample Data\GOODBAD.CSV N Learn: 529 N Test: 135

Target variable: TARGET

TreeNet Predictotors: Available: 6,332 Used: 2,789

Optimal R-Squared: TreeNet: Compression:

Standardized: NO TreeNet Optimality: Logistic Likelihood

Original TreeNet: Tree Size: 6 Trees Grown: 500 Rules Optimal: 6,306 Performance: 0.89735

Best Rule Extraction: % Compression: 55.77 % Gain/Loss: 0.009 Rules Extracted: 2,789 Performance: 0.90584

Outliers Summary: Not available for current Performance Measure.

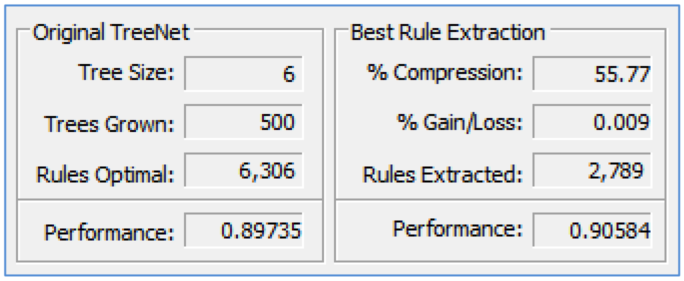
Performance Measure: ROC (Area Under Curve)

This window shows detailed RuleLearner compression results indexed by model size in nodes. The two test sample performance curves are aligned such that one can easily compare the performance of two models with identical size.

Note that the 2789-node TreeNet model has test ROC 0.875 while the 2789-node RuleLearner compressed model has ROC 0.90584. Even though both models have identical number of nodes, they differ in the actual trees selected. The TreeNet model takes the first consecutive trees from the TreeNet sequence (all combined with the unit coefficients and the original node updates) while the RuleLearner model takes some specific subset of 2789 nodes from the original complete set available in the sequence. Furthermore, each selected node has a new associated update.

✓ The actual subset of nodes can be determined by looking into the RuleLearner Details or translating the model.

The top part of the window reports basic stats on the optimal TreeNet model versus optimal RuleLearner model in terms of ROC performance.



Original TreeNet

Tree Size: 6

Trees Grown: 500

Rules Optimal: 6,306

Performance: 0.89735

Best Rule Extraction

% Compression: 55.77

% Gain/Loss: 0.009

Rules Extracted: 2,789

Performance: 0.90584

✓ You can switch to alternative performance measures using the Performance Measure: selection box.

Press the **[TreeNet Details]** button to open the usual **TreeNet Output** window described in the TreeNet engine chapter of this manual.

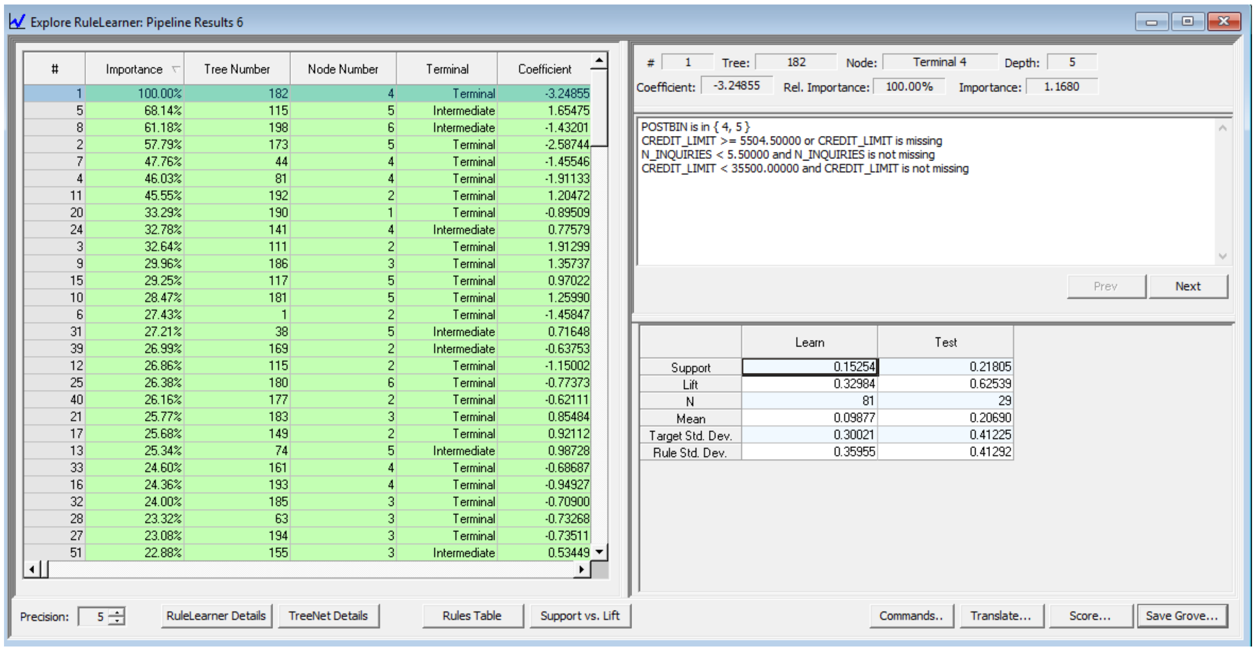
Press the **[RuleLearner Details]** button to open the usual **GPS Results** window described in the GPS engine chapter of this manual.

✓ Note that the GPS component of the RuleLearner pipeline uses nodes as input variables. Therefore, all parts of the GPS output windows referencing the predictors (Coefficients, etc.) will now report names like Tree 1 Node 1, Tree 1 TNode 1, Tree 2 Node 1, etc.

The remaining control items are similar to what we have already described earlier except for the **[Rules…]** button which opens up a RuleLearner specific **Explore** window.

Exploring the Rules

Press the **[Rules…]** button in the **Pipeline Results: RuleLearner** window to open the **Explore RuleLearner** window shown below.



#	Importance	Tree Number	Node Number	Terminal	Coefficient
1	100.00%	182	4	Terminal	-3.24855
5	68.14%	115	5	Intermediate	1.65475
8	61.18%	198	6	Intermediate	-1.43201
2	57.79%	173	5	Terminal	-2.58744
7	47.76%	44	4	Terminal	-1.45546
4	46.03%	81	4	Terminal	-1.91133
11	45.55%	192	2	Terminal	1.20472
20	33.29%	190	1	Terminal	-0.89509
24	32.78%	141	4	Intermediate	0.77579
3	32.64%	111	2	Terminal	1.91299
9	29.96%	186	3	Terminal	1.35737
15	29.25%	117	5	Terminal	0.97022
10	28.47%	181	5	Terminal	1.25990
6	27.43%	1	2	Terminal	-1.45847
31	27.21%	38	5	Intermediate	0.71648
39	26.99%	169	2	Intermediate	-0.63753
12	26.86%	115	2	Terminal	-1.15002
25	26.38%	180	6	Terminal	-0.77373
40	26.16%	177	2	Terminal	-0.62111
21	25.77%	183	3	Terminal	0.85484
17	25.68%	149	2	Terminal	0.92112
13	25.34%	74	5	Intermediate	0.98728
33	24.60%	161	4	Terminal	-0.68687
16	24.36%	193	4	Terminal	-0.94927
32	24.00%	185	3	Terminal	-0.70900
28	23.32%	63	3	Terminal	-0.73268
27	23.08%	194	3	Terminal	-0.73511
51	22.88%	155	3	Intermediate	0.53449

1 Tree: 182 Node: Terminal 4 Depth: 5
Coefficient: -3.24855 Rel. Importance: 100.00% Importance: 1.1680

```
POSTBIN is in { 4, 5 }
CREDIT_LIMIT >= 5504.50000 or CREDIT_LIMIT is missing
N_INQUIRIES < 5.50000 and N_INQUIRIES is not missing
CREDIT_LIMIT < 35500.00000 and CREDIT_LIMIT is not missing
```

	Learn	Test
Support	0.15254	0.21805
Lift	0.32984	0.62539
N	81	29
Mean	0.09877	0.20690
Target Std. Dev.	0.30021	0.41225
Rule Std. Dev.	0.35955	0.41292

Recall that the RuleLearner compression method decomposes a TreeNet model into a collection of individual nodes (internal and terminal) and then harvests individual nodes using GPS. Each node represents a segment of the original data described by a collection of rules leading into this node. Hence, analizing the rules asociated with the top selected nodes might provide additional modeling insights.

The nodes are listed on the left part of the **Explore** window according to the order of their importance (as reported by the GPS component). The actual rules associated with the currently selected node (segment) are shown on the top right part while the basic stats about the node (segment) are shown on the bottom right part of the window.

Note the use of the following terms generally accepted in the associative rules research community:

♦ **Support** – fraction of the sample (either learn or test) covered by the segment.

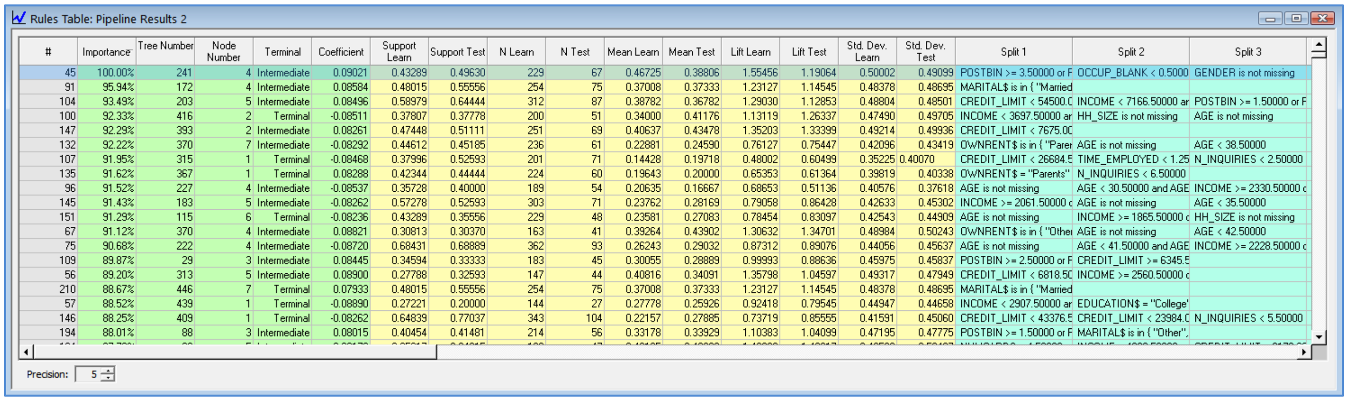
♦ **Lift** – segment mean of the target divided by the overall sample mean of the target.

✓ In the case of binary target, this is the same as segment fraction of 1s divided by the sample fraction of 1s.

- **Mean-** the average of the target variable for records that satisfy the rule of interest
- **Target Std. Dev.** – the usual standard deviation of the target variable for records that satisfy the rule of interest
- **Rule Std. Dev.:** also referred to as the "scale" of a rule and is given by $\sqrt{support * (1 - support)}$ where support is the support for the rule of interest
- **Importance = Rule Std. Dev. * |Coefficient|**

Click on the [Rules Table] button to open up even more detailed information about each node. The resulting table can be sorted by clicking on the any of its columns. This allows quick and easy identification of the potential rules of interest. For example, you may focus on finding the segments having the highest test sample lift or segments having the highest support and so on.

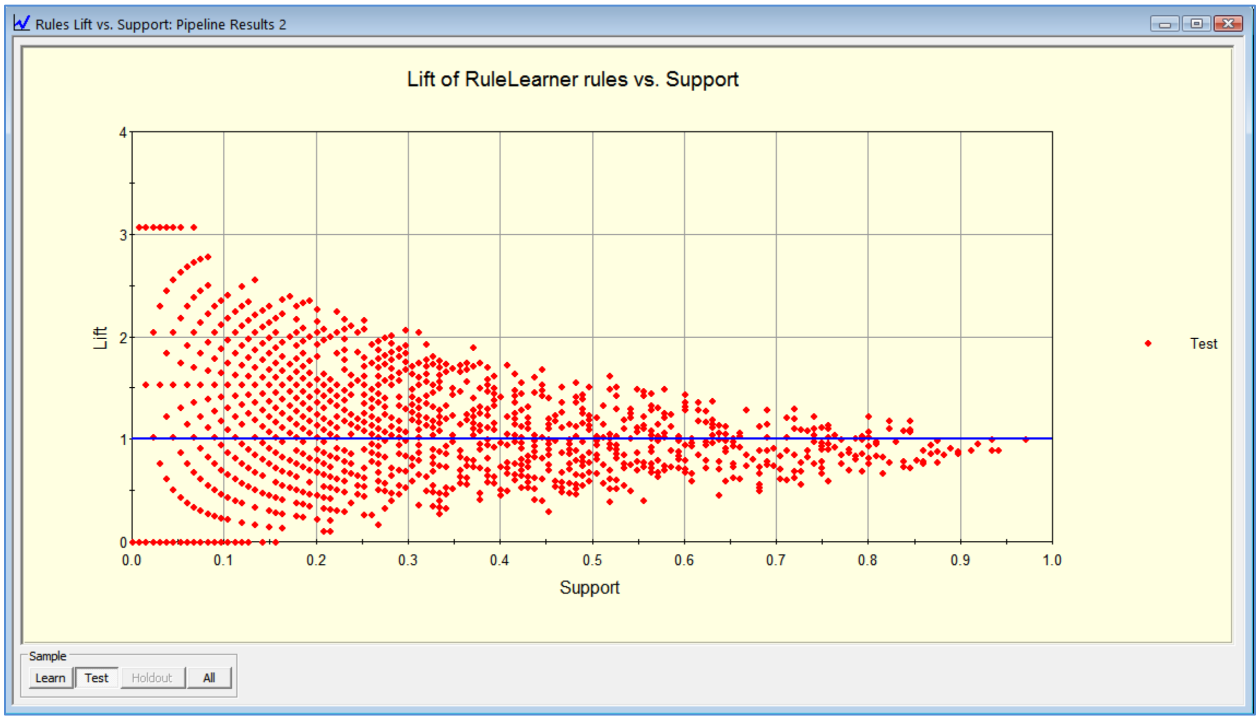
For example, internal node 4 of tree 241 is at the top of the importance list, it covers more than 50% of the available data (very large support) and is described by a joint collection of rules involving POST_BIN, OCCUP_BLANK, and GENDER.


Rules Table: Pipeline Results 2

#	Importance	Tree Number	Node Number	Terminal	Coefficient	Support Learn	Support Test	N Learn	N Test	Mean Learn	Mean Test	Lift Learn	Lift Test	Std. Dev. Learn	Std. Dev. Test	Split 1	Split 2	Split 3
45	100.00%	241	4	Intermediate	0.09021	0.43289	0.49630	229	67	0.46725	0.38806	1.55456	1.19064	0.50002	0.49099	POSTBIN >= 3.50000 or F	OCCUP_BLANK < 0.5000	GENDER is not missing
91	95.94%	172	4	Intermediate	0.08584	0.48015	0.55556	254	75	0.37008	0.37333	1.23127	1.14545	0.48378	0.48695	MARITAL$ is in { "Married		
104	93.49%	203	5	Intermediate	0.08496	0.58979	0.64444	312	87	0.38782	0.36782	1.29030	1.12853	0.48804	0.48501	CREDIT_LIMIT < 54500.0	INCOME < 7166.50000 ar	POSTBIN >= 1.50000 or F
100	92.33%	416	2	Terminal	-0.08511	0.37807	0.37778	200	51	0.34000	0.41176	1.13119	1.26337	0.47490	0.49705	INCOME < 3697.50000 ar	HH_SIZE is not missing	AGE is not missing
147	92.29%	393	2	Intermediate	0.08261	0.47448	0.51111	251	69	0.40637	0.43478	1.35203	1.33399	0.49214	0.49936	CREDIT_LIMIT < 7675.00		
132	92.22%	370	7	Intermediate	-0.08292	0.44612	0.45185	236	61	0.22881	0.24590	0.76127	0.75447	0.42096	0.43419	OWNRENT$ is in { "Parer	AGE is not missing	AGE < 38.50000
107	91.95%	315	1	Terminal	-0.08468	0.37996	0.52593	201	71	0.14428	0.19718	0.48002	0.60499	0.35225	0.40070	CREDIT_LIMIT < 26684.5	TIME_EMPLOYED < 1.25	N_INQUIRIES < 2.50000
135	91.62%	367	1	Terminal	0.08288	0.42344	0.44444	224	60	0.19643	0.20000	0.65353	0.61364	0.39819	0.40338	OWNRENT$ = "Parents"	N_INQUIRIES < 6.50000	
96	91.52%	227	4	Intermediate	-0.08537	0.35728	0.40000	189	54	0.20635	0.16667	0.68653	0.51136	0.40576	0.37618	AGE is not missing	AGE < 30.50000 and AGE	INCOME >= 2330.50000 c
145	91.43%	183	5	Intermediate	-0.08262	0.57278	0.52593	303	71	0.23762	0.28169	0.79058	0.86428	0.42633	0.45302	INCOME >= 2061.50000 c	AGE is not missing	AGE < 35.50000
151	91.29%	115	6	Terminal	-0.08236	0.43289	0.35556	229	48	0.23581	0.27083	0.78454	0.83097	0.42543	0.44909	AGE is not missing	INCOME >= 1865.50000 c	HH_SIZE is not missing
67	91.12%	370	4	Intermediate	0.08821	0.30813	0.30370	163	41	0.39264	0.43902	1.30632	1.34701	0.48984	0.50243	OWNRENT$ is in { "Other	AGE is not missing	AGE < 42.50000
75	90.68%	222	4	Intermediate	-0.08720	0.68431	0.68889	362	93	0.26243	0.29032	0.87312	0.89076	0.44056	0.45637	AGE is not missing	AGE < 41.50000 and AGE	INCOME >= 2228.50000 c
109	89.87%	29	3	Intermediate	0.08445	0.34594	0.33333	183	45	0.30055	0.28889	0.99993	0.88636	0.45975	0.45837	POSTBIN >= 2.50000 or F	CREDIT_LIMIT >= 6345.5	
56	89.20%	313	5	Intermediate	0.08900	0.27788	0.32593	147	44	0.40816	0.34091	1.35798	1.04597	0.49317	0.47949	CREDIT_LIMIT < 6818.50	INCOME >= 2560.50000 c	
210	88.67%	446	7	Terminal	0.07933	0.48015	0.55556	254	75	0.37008	0.37333	1.23127	1.14545	0.48378	0.48695	MARITAL$ is in { "Married		
57	88.52%	439	1	Terminal	-0.08890	0.27221	0.20000	144	27	0.27778	0.25926	0.92418	0.79545	0.44947	0.44658	INCOME < 2907.50000 ar	EDUCATION$ = "College"	
146	88.25%	409	1	Terminal	-0.08262	0.64839	0.77037	343	104	0.22157	0.27885	0.73719	0.85555	0.41591	0.45060	CREDIT_LIMIT < 43376.5	CREDIT_LIMIT < 23984.0	N_INQUIRIES < 5.50000
194	88.01%	88	3	Intermediate	0.08015	0.40454	0.41481	214	56	0.33178	0.33929	1.10383	1.04099	0.47195	0.47775	POSTBIN >= 1.50000 or F	MARITAL$ is in { "Other",	

Precision: 5

Finally, click on the **[Support vs. Lift]** button to see the graphical representation of the usual tradeoff involved. Note that the larger support segments tend to have smaller lift.


Rules Lift vs. Support: Pipeline Results 2
Lift of RuleLearner rules vs. Support
Lift
0
1
2
3
4
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Support
Test
Sample
Learn
Test
Holdout
All

This concludes our discussion of the new model compression approaches now available in SPM.