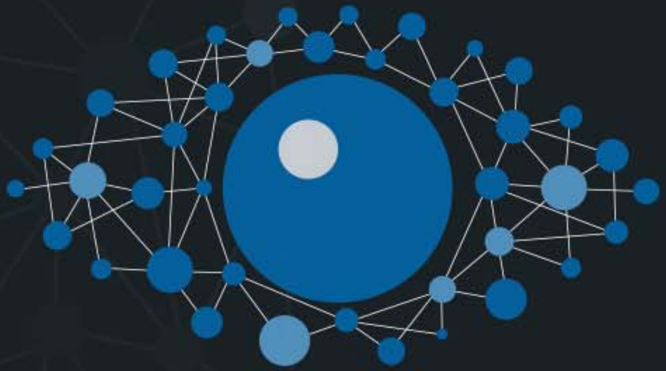
The Evolution of Data Mining Tools and Machine Learning Models
透視機器學習預測模型的發展



Bruno Scibilia
Technical Training Specialist 技术及业务开发经理, Minitab

Improve your Classification & Regression Models

CART DECISION TREES, GRADIENT BOOSTING, AND RANDOM FORESTS

When should we use Salford Predictive Modeler (SPM) ?

The curse of dimensionality…..

- **Very large number of predictors** : In this context, traditional statistical modeling tools may become less efficient. Risk of having many spurious effects wrongly identified as real effects, even when such effects are exclusively caused by random fluctuations.
- **Very large sample size** (Power vs. Practical significance) : In this context, traditional statistical techniques tend to become over-sensitive to small effects leading to a very complex final model. Even though some terms are statistically significant, most of them may actually have little practical significance..
- Complexity due to **Non linear effects, Complex interactions, Missing values** and **Outliers** …..

Suite of Solutions

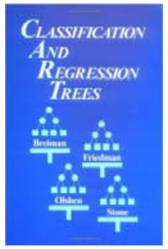
Time- and market-tested predictive modeling tools including everything from market-leading decision tree and classification engines to advanced interaction detection and automation to state-of-the-art machine learning capabilities.

SPM Software Suite						
CART	MARS	Random Forests	TreeNet	RuleLearner	ISLE	GPS
Decision trees	Nonlinear regression	Data ensemble bagging	Gradient boosting	Rule ensemble	Model compression	Regularized regression

CART: Introduction

Classification **A**nd **R**egression **T**rees

Breiman, Friedman, Olshen, and Stone (1984)



CART is a decision tree algorithm and can be used for both regression and classification problems

CART is available exclusively in the SPM®8 Software Suite and was developed in close consultation with the original authors

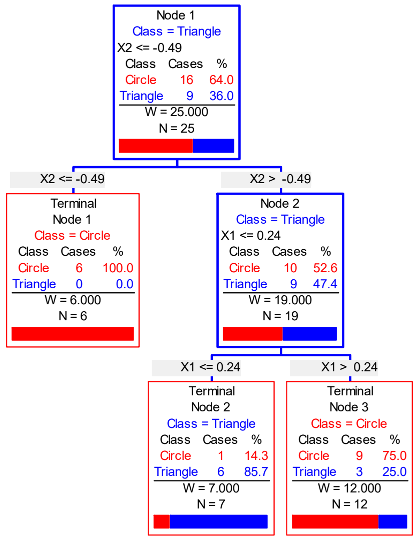
CART & Tree-Based Machine Learning

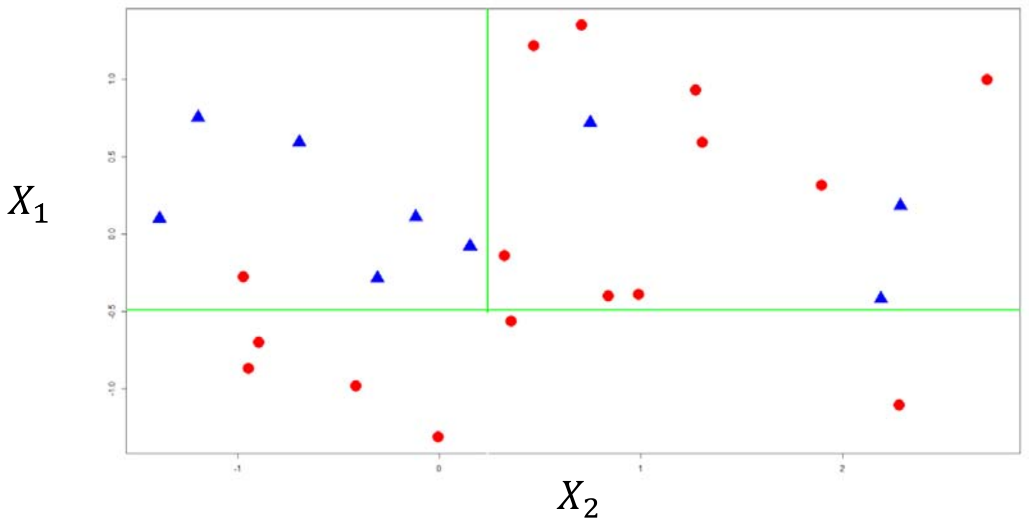
Understanding CART is essential to understanding other decision tree-based machine learning algorithms like Stochastic Gradient Boosting, Random Forests, MARS regression splines, RuleLearner rule ensembles, and ISLE model compression.

CART: Introduction

Main Idea: divide the data (often people say “partition” instead of “divide”) into different regions so that the dependent variable (also called the “target variable”) can be predicted more accurately.

Example: classify an observations as either a triangle or a circle

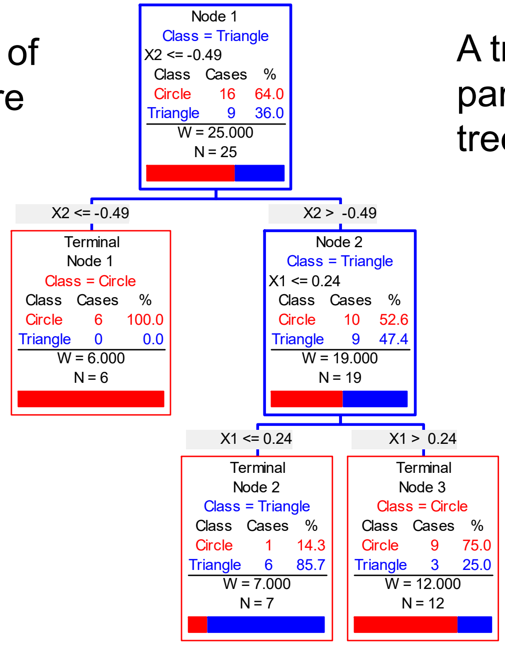




CART: Terminology

The node at the top of the tree is called the ***root node***

Nodes are the building blocks of CART decision trees. There are 5 nodes in this CART tree.



A tree ***split*** occurs when the data are partitioned (more on this later). This tree has two splits:

1. X2 ≤ -.49
2. X1 ≤ .24

The **predicted value** in a CART tree for classification is a class prediction or a probability

Example: if $X_2 = -.2$ and $X_1 = 0$ then the predicted class is a triangle (the probability of a triangle is $\frac{6}{7} = 85.7\%$)

Each split divides one node into two "child" nodes. This is called a *binary split.*

A node that has no sub-branch is a ***terminal node***

This tree has three terminal nodes

A tree node is **pure** if it contains only a single class (see Terminal Node 1)

CART: Algorithm

Grow a large tree

This is done for you automatically

All variables are considered at each split in the tree

Each split is made using one variable and a specific value or set of values.

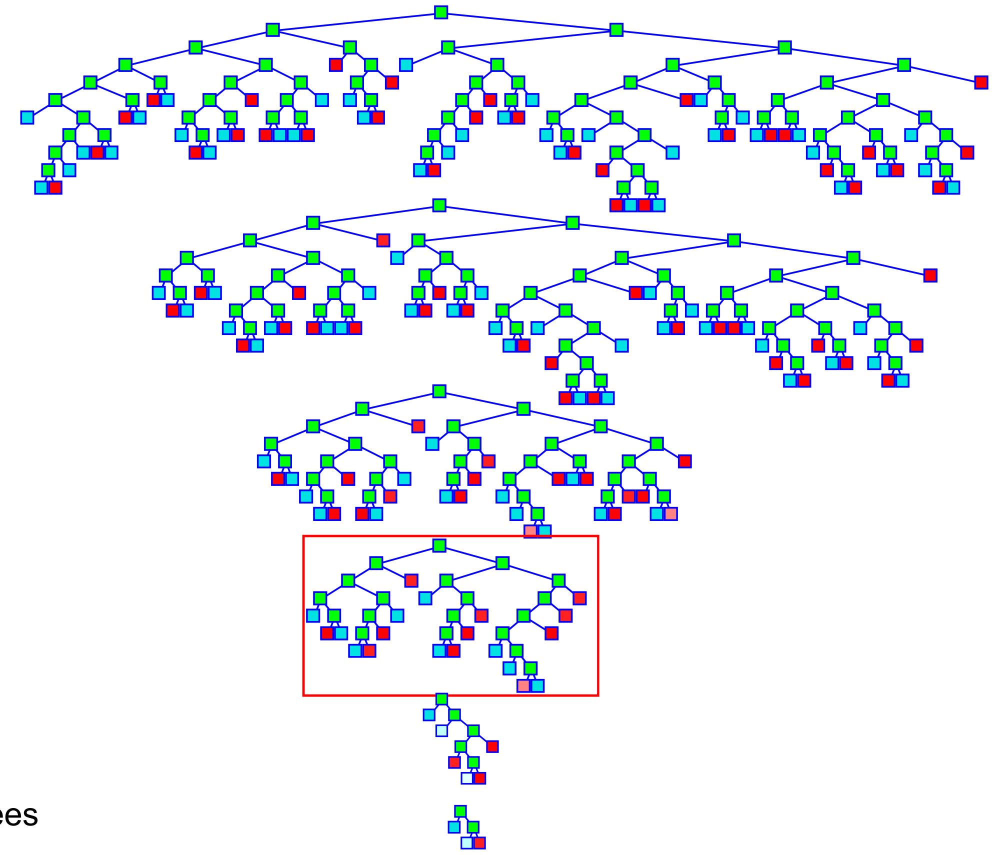
Splits are chosen so as to maximize a splitting criterion

Grow the tree until either a
user-specified criterion is met or
until the tree cannot be grown further

Prune the large tree upward

This is done for you automatically

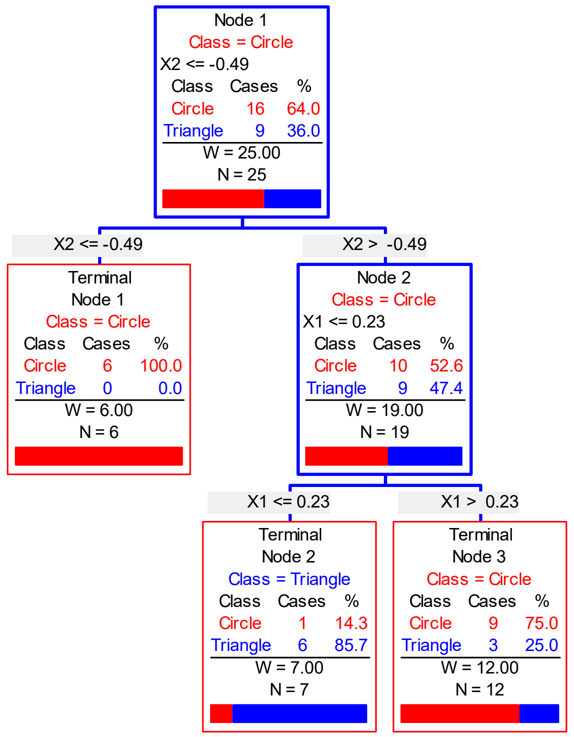
Use either a test sample or cross validation to prune subtrees



CART: Splitting Process

How exactly did we get this tree?

X1	X2	SHAPE
2.29	0.18	Triangle
-1.20	0.75	Triangle
-0.69	0.59	Triangle
-0.41	-0.98	Circle
-0.97	-0.28	Circle
-0.95	-0.87	Circle
0.75	0.72	Triangle
-0.12	0.11	Triangle
0.15	-0.08	Triangle
2.19	-0.42	Triangle
0.36	-0.56	Circle
2.72	1.00	Circle
2.28	-1.11	Circle
0.32	-0.14	Circle
1.90	0.31	Circle
0.47	1.22	Circle
-0.89	-0.70	Circle
-0.31	-0.29	Triangle
0.00	-1.31	Circle
0.99	-0.39	Circle
0.84	-0.40	Circle
0.71	1.35	Circle
1.31	0.59	Circle
-1.39	0.10	Triangle
1.27	0.93	Circle



CART: Splitting Process

Step 1: Find the best split point for the variable X_1

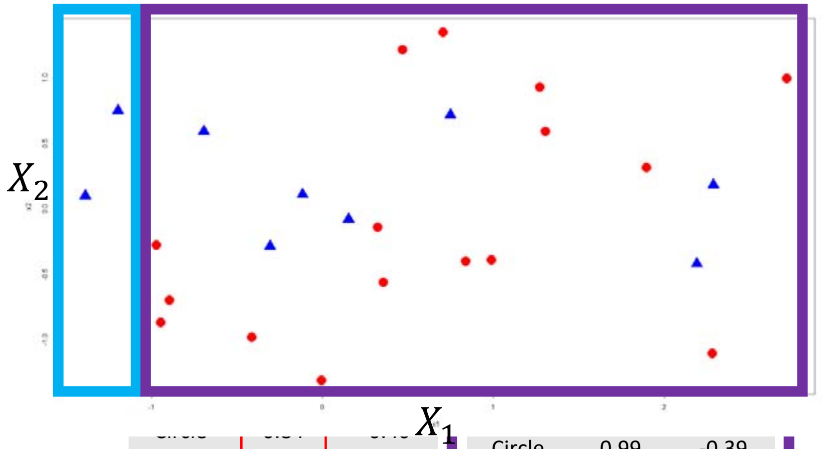
- Sort the variable X_1
- Compute the split improvement for each possible split point.
- Best split for X_1:
  - $X_1 \leq -1.08$

The split improvement is a measure of how much the split separates the target variable classes (i.e. triangles and circles).

SHAPE	X1	X2
Triangle	-1.39	0.10
Triangle	-1.20	0.75

SHAPE	X1	X2
Circle	-0.97	-0.28

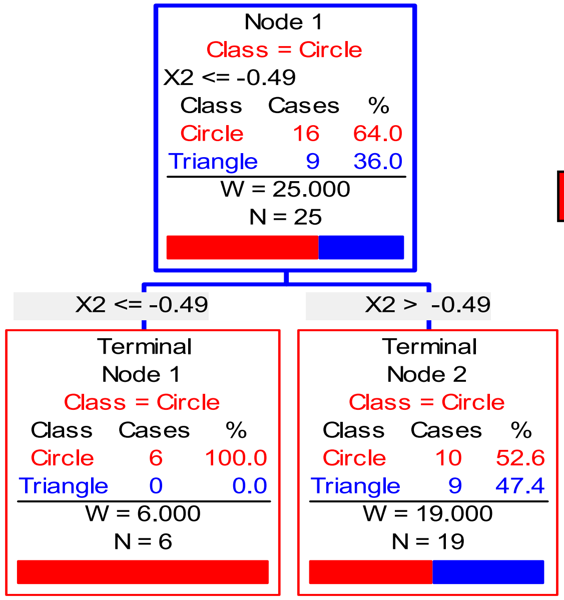
SHAPE	X1	X2
Triangle	-1.39	0.10
Triangle	-1.20	0.75
Circle	0.99	-0.39
Circle	1.27	0.93
Circle	1.31	0.59
Circle	1.90	0.31
Triangle	2.19	-0.42
Circle	2.28	-1.11
Triangle	2.29	0.18
Circle	2.72	1.00

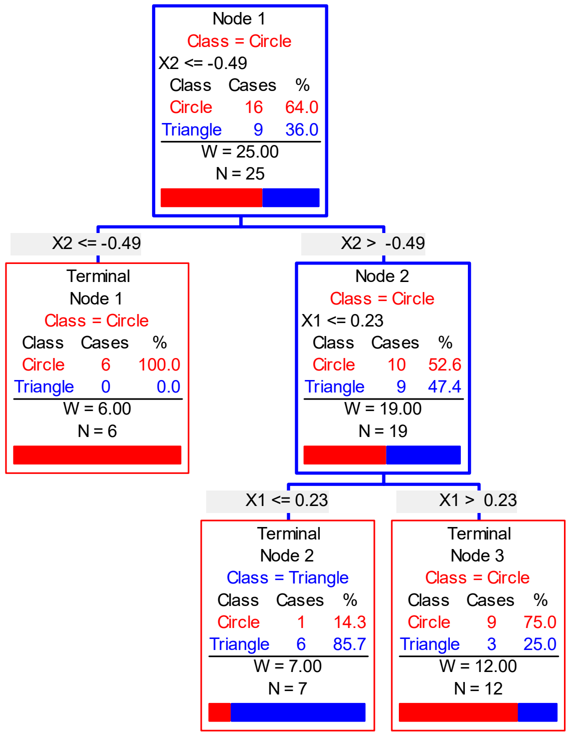


Circle	0.99	-0.39
Circle	1.27	0.93
Circle	1.31	0.59
Circle	1.90	0.31
Triangle	2.19	-0.42
Circle	2.28	-1.11
Triangle	2.29	0.18
Circle	2.72	1.00

CART: Splitting Procedure

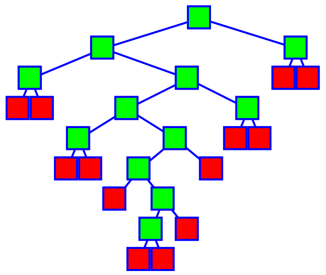
So how do we get to our final tree?

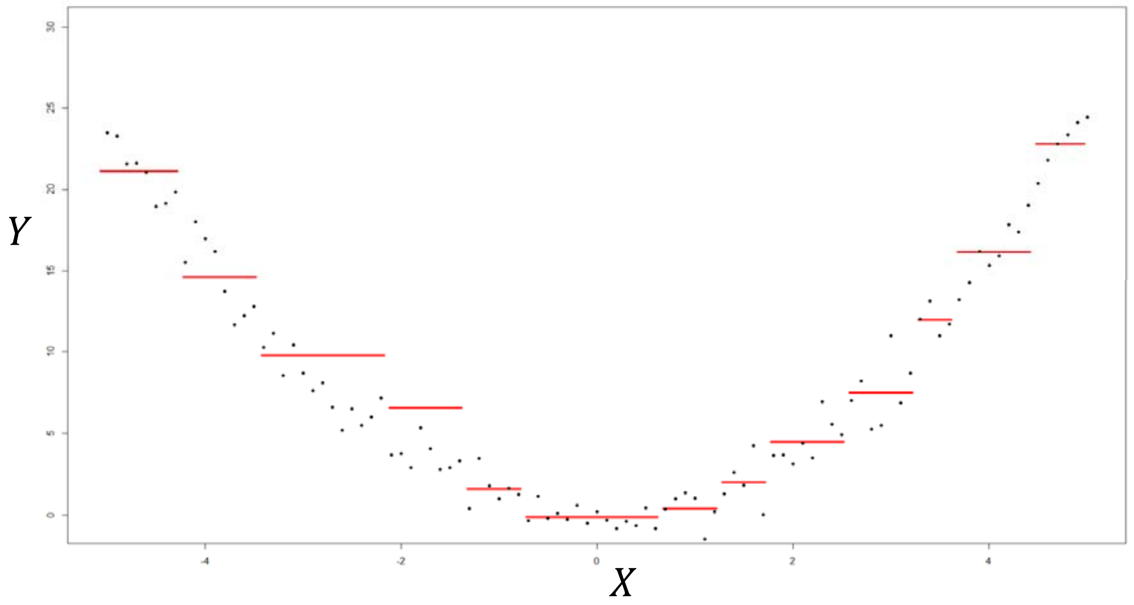




CART: Automatic Nonlinear Modeling

Nonlinear functions (and linear) are approximated via step functions, **so in practice you do not need to worry about adding terms like x^2 or $ln(x)$ to capture nonlinear relationships.** The picture below is the CART fit to $Y = X^2$ + noise. CART modeled this data automatically. No data pre-processing. Just CART.





CART: Outliers in the Target Variable

Two types of outliers are

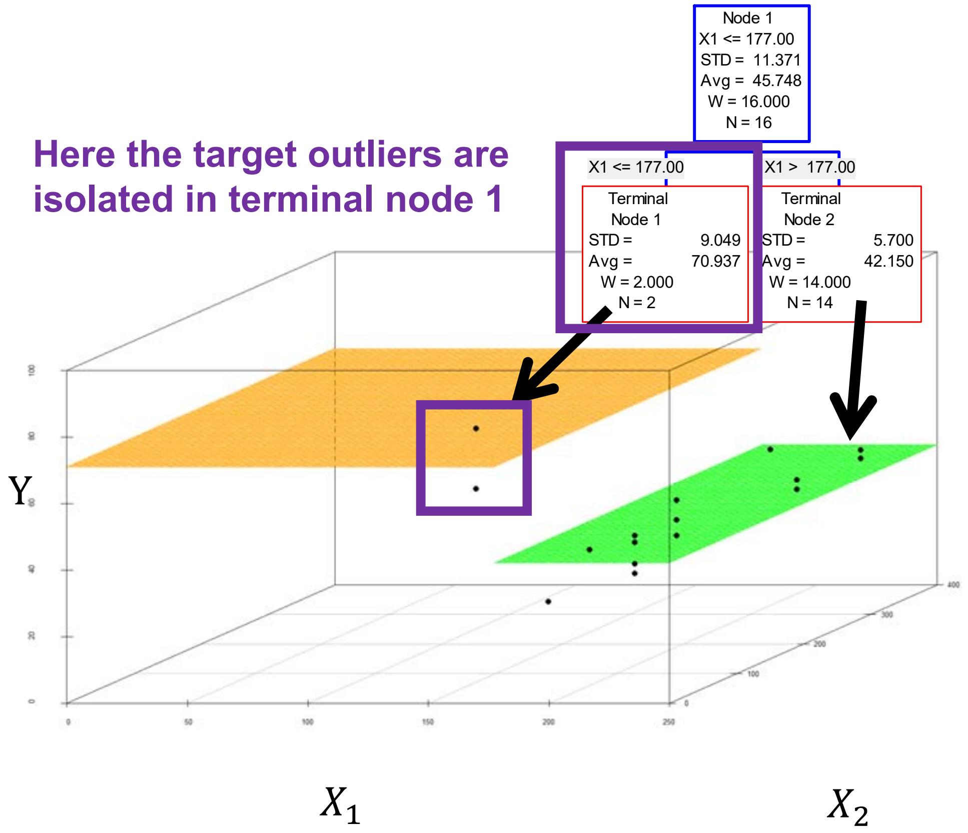
1. Outliers in the target variable (i.e. “Y”)
2. Outliers in the predictor variable (i.e. “x”)

CART is more sensitive to outliers with respect to the target variable

1. More severe in a regression context than a classification context
2. CART may treat target variable outliers by isolating them in small terminal nodes which can limit their effect

Reference: Pages 197-200 and 253 in Breiman, Friedman, Olshen, and Stone (1984)

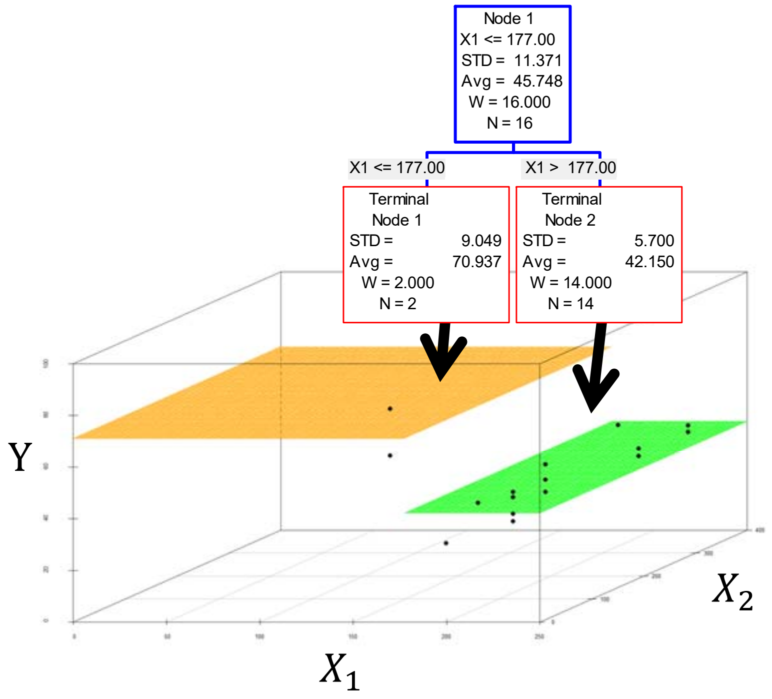
Here the target outliers are isolated in terminal node 1

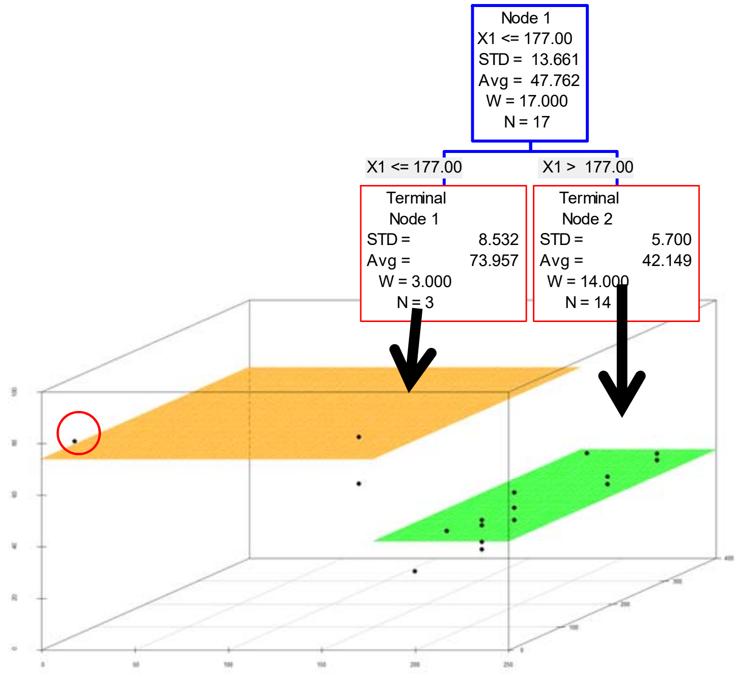


CART: Outliers in the Predictor Variables

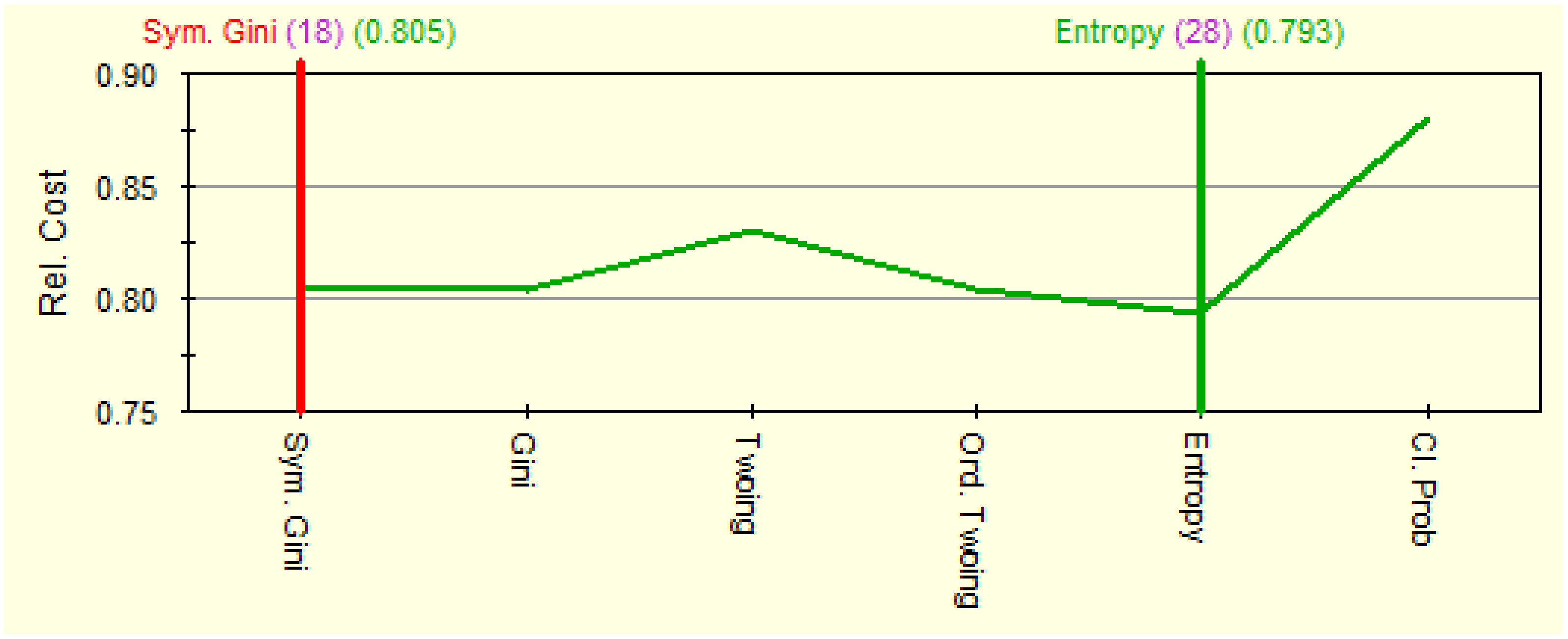
CART is more robust to outliers in the predictor variables partly due to nature of the splitting process

Reference: Pages 197-200 and 253 in Breiman, Friedman, Olshen, and Stone (1984)





CART: **Rules** Automate



This automate automatically tries all six alternative splitting rules.
According to the output above, Entropy, Gini and Sym.
Gini splitting rules resulted to be the most accurate models on the given data.

Introduction to MARS

MARS- Multivariate Adaptive Regression Splines

Creator: Jerome Friedman

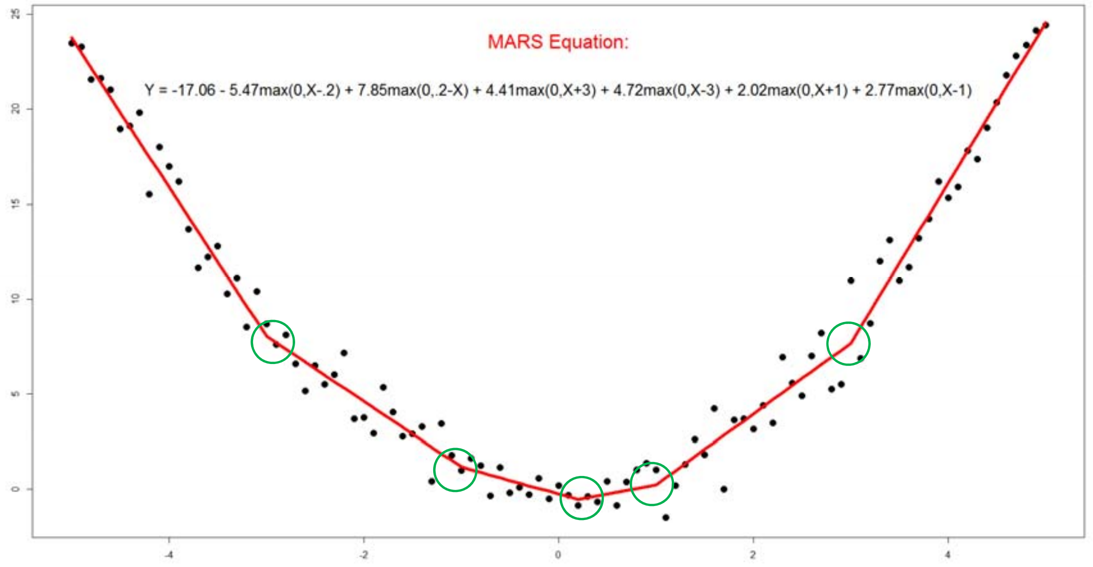
Co-Created CART decision trees, created gradient boosting, and Co-Created RuleLearner rule ensembles and ISLE® Model Compression

Primary References:

1. Multivariate Adaptive Regression Splines (Friedman, 1991)
2. Fast MARS (Friedman, 1993)
3. Estimating Functions of Mixed Ordinal and Categorical Variables Using Adaptive Splines (Friedman, 1991)
4. Recursive Partitioning (Zhang and Singer, 2010; see Chapter 10)

Introduction to MARS

A MARS model is a regression model that is automatically constructed using an adaptive spline algorithm



$$Y = -17.06 - 5.47\max(0, X-2) + 7.85\max(0, 2-X) + 4.41\max(0, X+3) + 4.72\max(0, X-3) + 2.02\max(0, X+1) + 2.77\max(0, X-1)$$

Introduction to MARS Basis Functions

The transformed terms used in a MARS model are referred to as “basis functions”

In MARS, basis functions are added to the model in pairs and take the following form

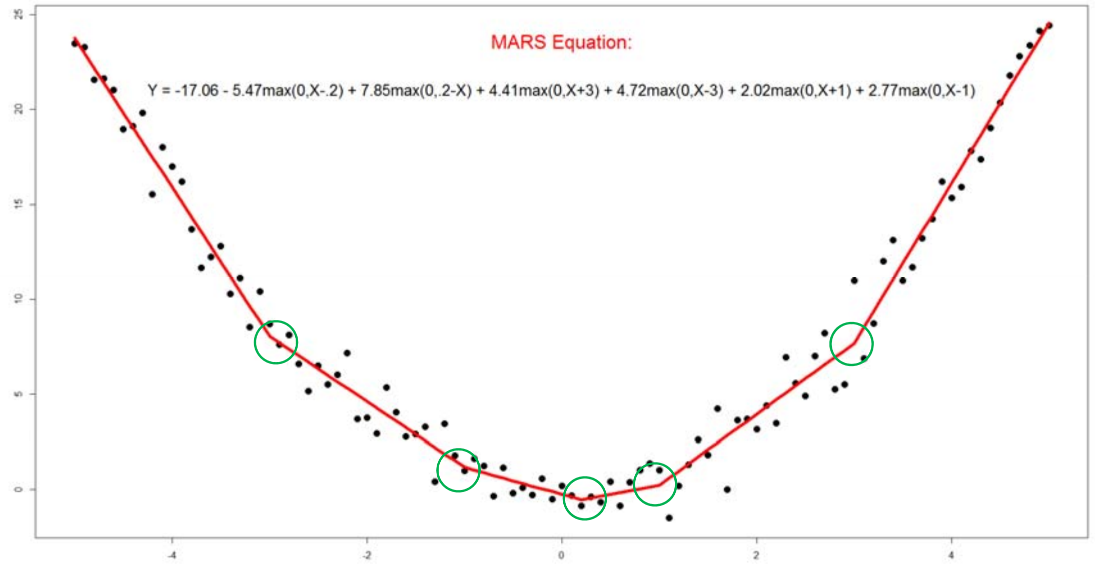
1. max(0, X-K)
2. max(0, K-X)

Where X is a predictor variable and K is a “knot”

The basis functions allow MARS to model both nonlinear and linear relationships in the data

Nonlinear: MARS places knots that allow the model to “turn” when necessary

Linear: MARS places the knot at the minimum value for a specific variable



$$Y = -17.06 \ - 5.47\max(0, X-2) + 7.85\max(0, 2-X) + 4.41\max(0, X+3) + 4.72\max(0, X-3) + 2.02\max(0, X+1) + 2.77\max(0, X-1)$$

Constructing a MARS Model

MARS is actually an adaptation of CART that allows for additive terms to be entered into the model (this gives MARS an advantage if there are additive effects in the data)

Remember that in order to construct a CART decision tree we have two main steps

1. Grow a large tree
2. Prune the large tree

The process for constructing a MARS model is similar and involves two steps:

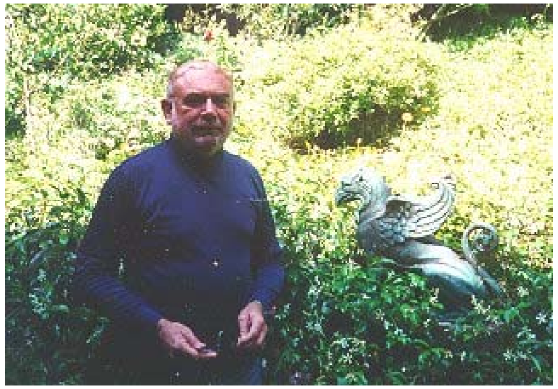
1. Forward Step (add terms to the model)
2. Backwards Step (delete terms from the model)

Introduction to Random Forests

Main Idea: fit multiple CART trees to independent "bootstrap" samples of the data and then combine the predictions

Leo Breiman, one of the co-creators of CART, also created Random Forests and published a paper on this method in 2001

Our RandomForests® software was developed in close consultation with Breiman himself



What is a bootstrap sample?

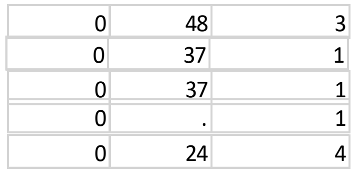
A bootstrap sample is a random sample conducted with replacement

<u>Steps:</u>

1. Randomly select an observation from the original data
2. “Write it down”
3. “Put it back” (i.e. any observation can be selected more than once)

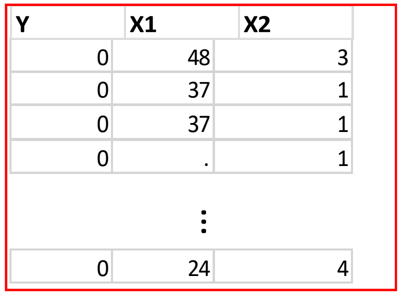
Repeat steps 1-3 N times; N is the number of observations in the original sample

FINAL RESULT: One “bootstrap sample” with N observations



0	48	3
0	37	1
0	37	1
0	.	1
0	24	4

Original Data



Y	X1	X2
0	48	3
0	37	1
0	37	1
0	.	1
	⋮	
0	24	4

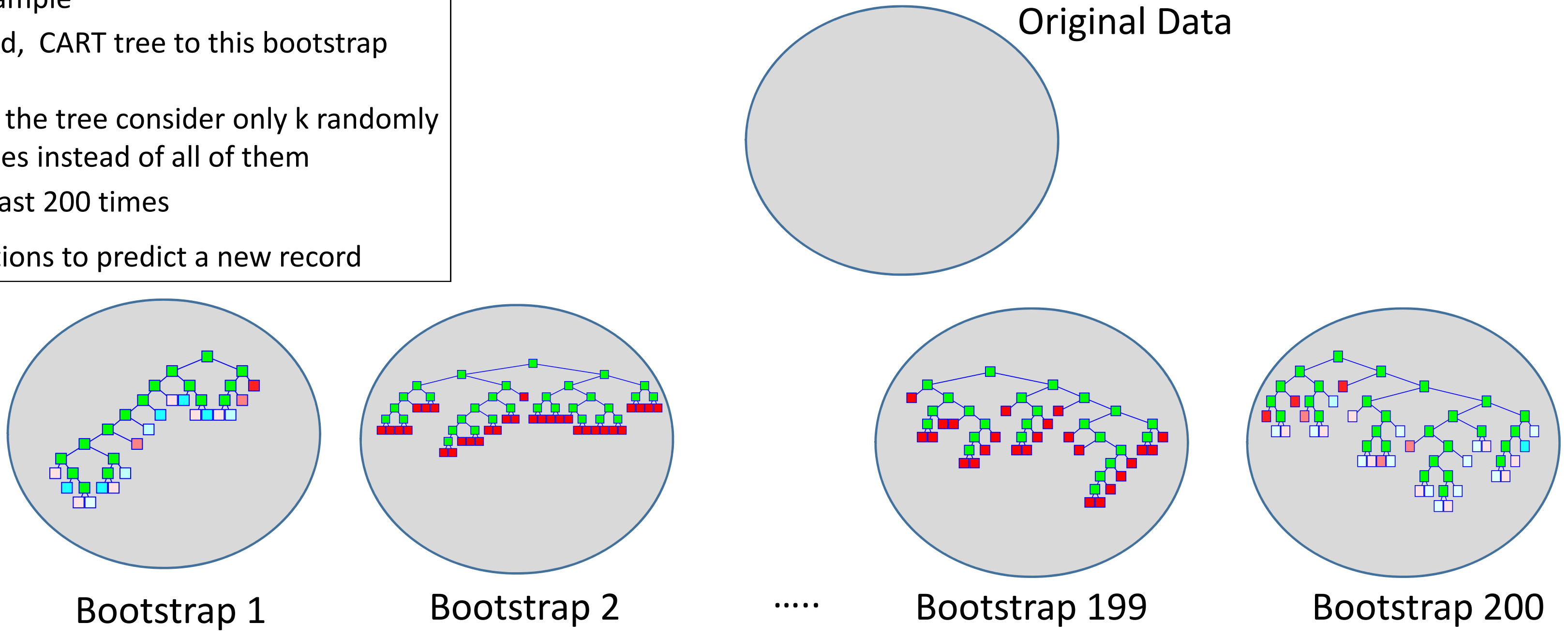
Bootstrap Sample

Random Forest Algorithm (Regression):

1. Draw a bootstrap sample
2. Fit a large, unpruned, CART tree to this bootstrap sample
   - -At each split in the tree consider only k randomly selected variables instead of all of them

Repeat Steps 1-2 at least 200 times

3. Average the predictions to predict a new record



Predict a New Record : run the record down each tree, each time computing a prediction

Tree 1: 10.5 Tree 2: 9.8 ..... Tree 199: 10.73 Tree 200: 12

Final Prediction for a New Record: take the average of the 200 individual predictions

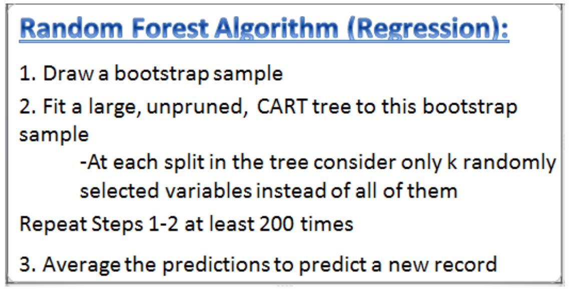
$$\frac{10.5 + 9.8 \ldots + 10.73 + 12}{200}$$

CART and Random Forests

When you build a Random Forest model just keep this picture in the back of your mind:

$$\text{Random Forests}^{®} = \frac{1}{B}\left[\text{CART}^{®}_{1} + \text{CART}^{®}_{2} + \text{CART}^{®}_{3} + \ldots + \text{CART}^{®}_{B}\right]$$

The reason is because a Random Forest is really just an average of CART trees constructed on bootstrap samples of the original data



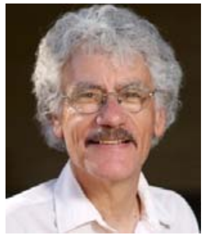
Random Forest Algorithm (Regression):

1. Draw a bootstrap sample
2. Fit a large, unpruned, CART tree to this bootstrap sample
   - -At each split in the tree consider only k randomly selected variables instead of all of them

Repeat Steps 1-2 at least 200 times

3. Average the predictions to predict a new record

Introduction to Stochastic Gradient Boosting



Main Idea: iteratively fit CART trees to "*generalized residuals*" (much more on this later)

Creator: Jerome Friedman

Friedman also co-created CART decision trees, created MARS regression splines, and co-created RuleLearner™ rule ensembles

The SPM modeling engine that implements the Stochastic Gradient Boosting algorithm is called TreeNet

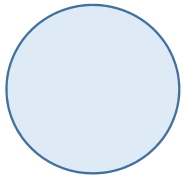
The TreeNet code was originally written by Friedman himself

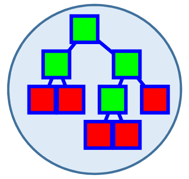
Gradient Boosting Algorithm Sketch

Make an initial prediction

1. Draw a random sample from the LEARN Data
2. Compute the “generalized residuals” for the records in the sample
3. Fit a CART tree to the generalized residuals for the records in the sample
4. Shrink the updates and add them to the current model to obtain an updated model

Repeat Steps 1-4 M times

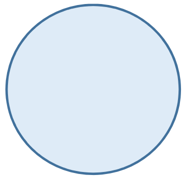

Sample


Sample

Gradient Boosting: Iteration 3 (Least Squares Loss)

Current Model: $f_2(x_i) = \bar{y} + \alpha CART_1 + \alpha CART_2$

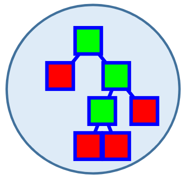
1. Draw a random sample S from the records in the learn data



2. Compute the generalized residual $\tilde{y}_i = y_i - f_2(x_i)$ for the records in S

$\tilde{y}_i$	x_1	x_3
-.3	2	12
-2	5	1
.3	7	7

3. Using $\tilde{y}_i$ as the target variable, fit a small, unpruned CART tree with J terminal nodes using the records in S



4. Multiply the predictions from Step 3 by the learning rate α = .01 and add them to the current model $f_2(x_i)$ to obtain an updated model $f_3(x_i)$.

$$f_3(x_i) = (f_2(x_i)) + \alpha CART_3$$

$$= (\bar{y} + \alpha CART_1 + \alpha CART_2) + \alpha CART_3$$

Least Squares Loss Function:

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

We use the tree to update the model:
for the least squares loss function, the updates are the target variable average for each terminal node.
Denote the updates for iteration 3 by $CART_3$

CART and Gradient Boosting

When you are fitting gradient boosted trees remember that you are sequentially fitting M number of CART trees to *generalized residuals* instead of the original target and that you are shrinking the updates by α

Keep this picture in the back of your mind

$$\text{TREENET}^{®} = f_0 + \alpha\,\text{CART}^{®}_1 + \ldots + \alpha\,\text{CART}^{®}_M$$

Learning Rate and the Number of Iterations

In gradient boosting the learning rates are set to be smaller. Smaller learning rates require more trees (iterations) for the model error to converge ("converge" means that the test error curve will be flat, that is, no longer decrease)

Intuition: Imagine that you want to walk to the middle of a crater

If you take small strides (i.e. smaller learning rate) then you will need more steps (i.e. more trees= more iterations) to get to the bottom of the crater (i.e. converge to the minimum error).

If you make a mistake (i.e. the model makes an error during one of the iterations) then the error is not as serious because your stride is smaller (i.e. it is easier for the model to recover in the next iteration)

If you take larger strides (i.e. larger learning rate) then you will need less steps (i.e. less trees = less iterations) to get to the bottom of the crater (i.e. converge to the minimum error)

If you make a mistake (i.e. the model makes an error during one of the iterations) then the error is more serious because your stride is larger(i.e. it is not as easy for the model to recover as compared to the situation where the learning rate is smaller)

Gradient Boosting: Least Squares Loss

How does TreeNet model this curve? It makes small improvements
(i.e. the learning rate is a small number that “**shrinks**” the model updates)

Tree 1

Tree 10

Tree 50

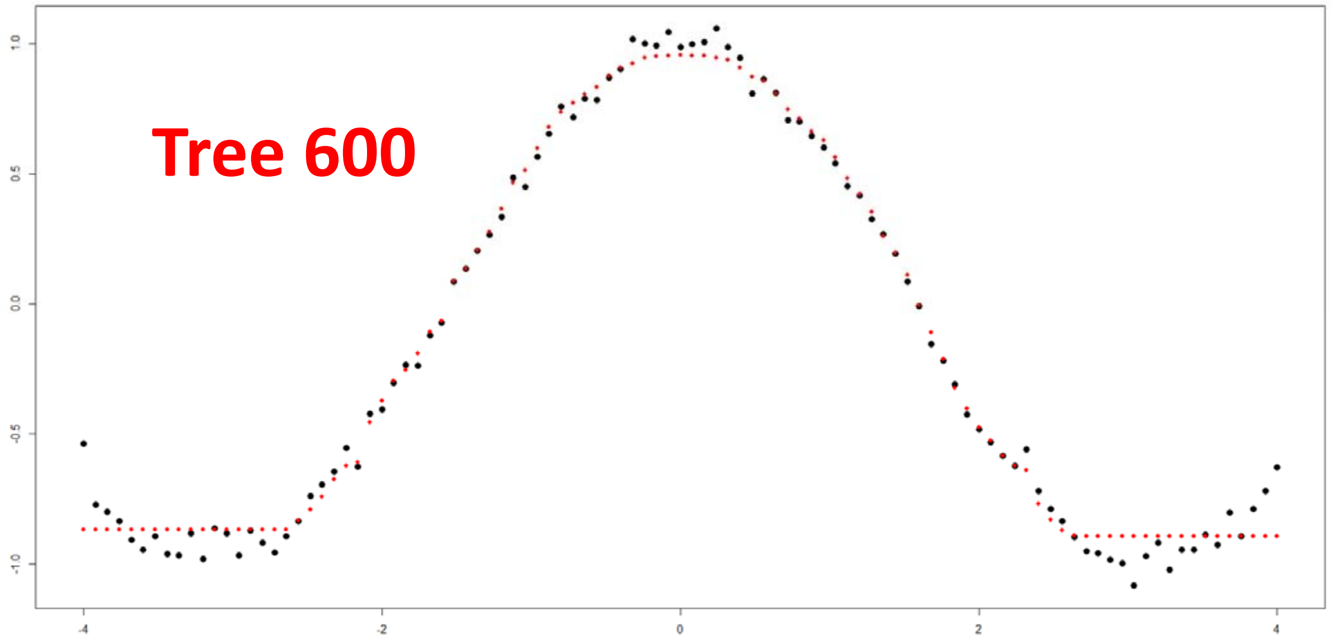
Tree 100

Tree 150

Tree 200

Tree 400

Tree 600

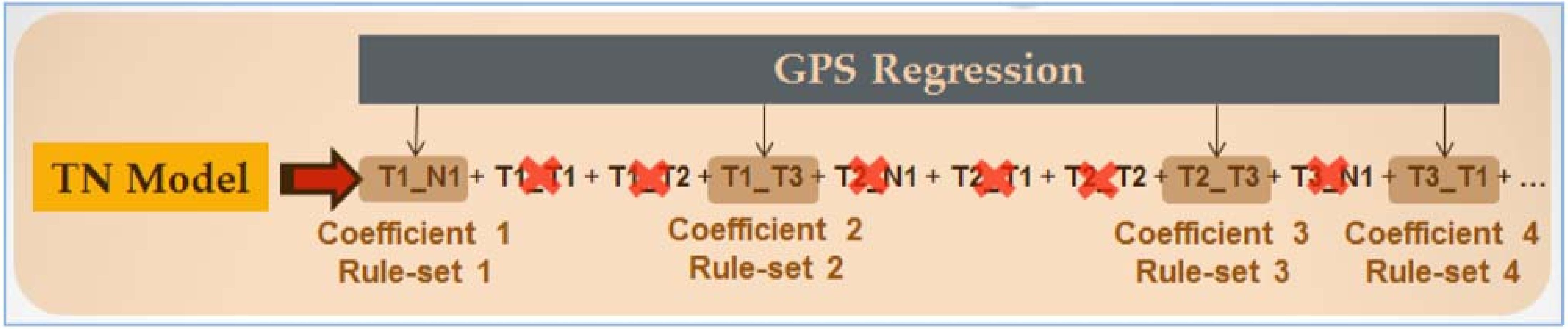


Note: Noise ~ N(0,1)

Model Compression via Rule Learner

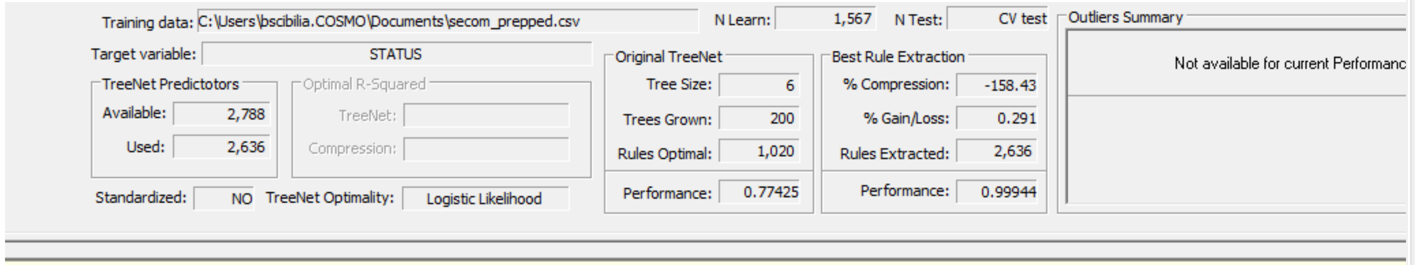
Use modern regression techniques to post-process a complex TREENET model to create a radically simplified version. The post-processed simplified model might perform even better than the complex model.

Extracting a collection of informative rule-sets from the TREENET model :



Rule Learner

Achieve higher performance compared to the original TreeNet model while compressing the model.



Training data: C:\Users\bscibilia.COSMO\Documents\secom_prepped.csv   N Learn: 1,567   N Test: CV test

Target variable: STATUS

TreeNet Predictotors
Available: 2,788
Used: 2,636

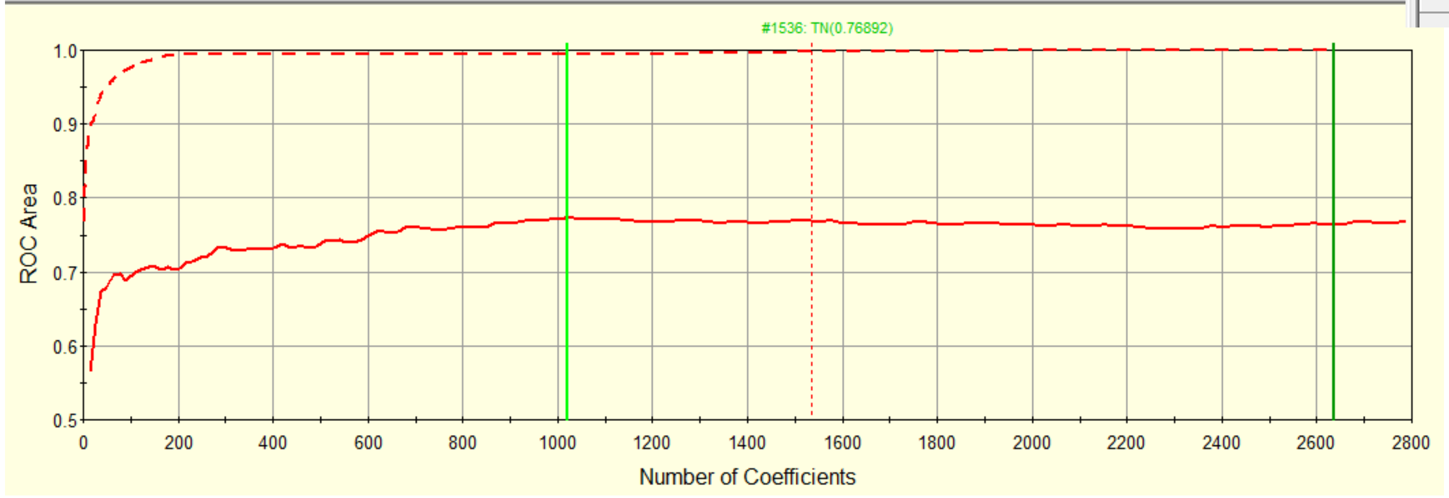
Optimal R-Squared
TreeNet:
Compression:

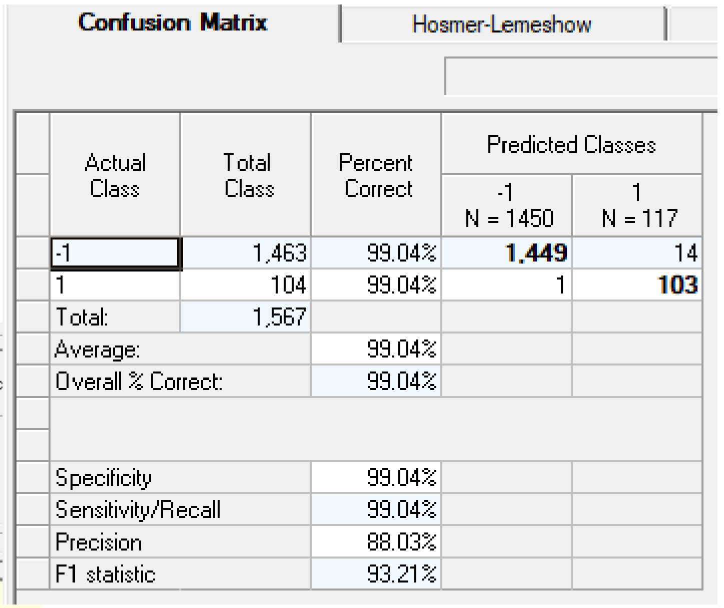
Standardized: NO   TreeNet Optimality: Logistic Likelihood

Original TreeNet
Tree Size: 6
Trees Grown: 200
Rules Optimal: 1,020
Performance: 0.77425

Best Rule Extraction
% Compression: -158.43
% Gain/Loss: 0.291
Rules Extracted: 2,636
Performance: 0.99944

Outliers Summary
Not available for current Performanc





Actual Class	Total Class	Percent Correct	Predicted Classes -1 N = 1450	Predicted Classes 1 N = 117
-1	1,463	99.04%	**1,449**	14
1	104	99.04%	1	**103**
Total:	1,567			
Average:		99.04%		
Overall % Correct:		99.04%		
Specificity		99.04%		
Sensitivity/Recall		99.04%		
Precision		88.03%		
F1 statistic		93.21%		