Random Forest Documentation
http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

The Random Forest description for the link above is clear.

The website also has many links to pursue. For example see
http://www.stat.berkeley.edu/~breiman/RandomForests/ENAR_files/frame.htm
Topics include
- Fast and then better imputation of missing values.
- Some data examples
- Weak learners and the power averaging independent learns
- Finding outliers via proximities

Much of text below is from document with the above link. I should add more quotes below. My own wording below may not be as good.
My good is tp draw more attention to their work, not take credit for it.

Most of graphics here use functions from the R randomForest package.
There are many ways to look at the results provided by Random Forests.

Overview: Tree Growth

RF mean Random Forests. The data is a training data set.

Case sampling for each tree

RF samples cases with replacement leaving out about one-third of the cases.
RF calls not-selected cases out-of-bag (oob) cases. These are use for:

1) unbiased estimates of classification errors as trees are added to the forest
2) variable importance

Variable sampling for each node

RF randomly selects m of the M variables in the data set
to assess the splitting of cases at each a node of the tree
RF selects the best split: both the variable and the value
The number m is constant for all trees and nodes.
R's RF function has an argument m with default values.
Classification: m ≈ sqrt(M), Regression: m ≈ M/3 (need to verify)

Tree growth

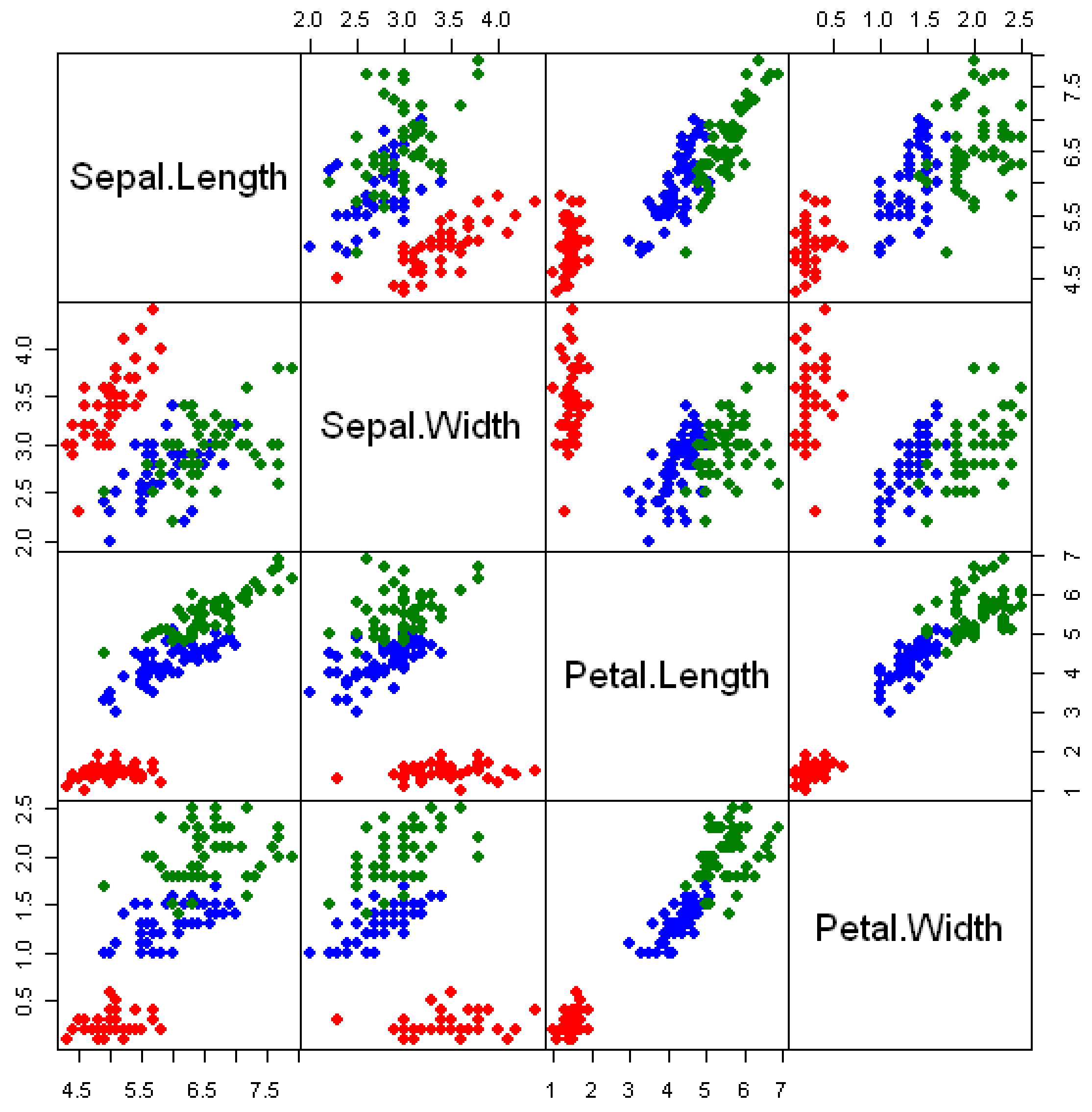
RF grows each classification tree to the largest extent possible.
*Regression trees have a default minimum leaf node size of 5.
There is no pruning.

Iris Data Set

The iris data set has 150 cases.
It has 5 variables, 4 predictors and the iris species variable with 3 classes

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
51	7.0	3.2	4.7	1.4	versicolor
101	6.3	3.3	6.0	2.5	virginica

Red = setosa, Green=Versicolor, Blue = Virginica



Example R script for a Classification Model

In R’s RF function we specify the predictors variables using the x argument.
We specify the class variable with the y argument.

```
irisRf <- randomForest( x = iris[,  -5], y = iris[, 5],
              keepForest = TRUE, proximity = TRUE)
```

As indicated above the class variable is the 5th variable in the iris data matrix.

I chose to keep the forest to show the first tree as an example below.

I fairly often choose to keep the proximity matrix for later use.

Other useful options are to obtain variable and local variable importance.

One Iris Tree Table of the 500

1= Root Node	
Leaf Nodes	🟩

Node ID	Left Child	Right Child	Split Var	Split point	Status -1 = Leaf	Predicted Class
1	2	3	4	0.80	1	0
2	0	0	0	0.00	-1	1
3	4	5	4	1.65	1	0
4	6	7	4	1.35	1	0
5	8	9	1	6.05	1	0
6	0	0	0	0.00	-1	2
7	10	11	3	4.95	1	0
8	12	13	3	4.85	1	0
9	0	0	0	0.00	-1	3
10	0	0	0	0.00	-1	2
11	0	0	0	0.00	-1	3
12	14	15	1	5.40	1	2
13	0	0	0	0.00	-1	3
14	0	0	0	0.00	-1	3
15	0	0	0	0.00	-1	2

Variables: 1 = Sepal.L, 2 = Sepal.W, 3 = Petal.L, 4 = Petal.W
Classes: 1 = Setosa, 2 = Versicolor, 3= Virginia

The out-of-bag (oob) error estimate

RF puts each oob case for the kth tree down the kth tree to get a its classification .
RF will classify each case in about one-third of the trees
For each case let j be the class that gets most votes when the case is oob.

OOB error rate estimate is the proportion cases with j not equal to the true class
RF also computes the class.error rates using the class specific cases

“This has proven to be unbiased in many tests.”

Below rows are the training data classes and columns are predicted classes

Iris Data, 500 trees

OOB estimate of error rate: 4.67%

Confusion matrix:

	setosa	versicolor	virginica	class.error
setosa	50	0	0	0.00
versicolor	0	47	**3**	0.06
virginica	0	**4**	46	0.08

Classification Variable Importance

Classification Accuracy

In every tree grown in the forest, put down the oob cases and count the number of votes cast for the correct class.

Now randomly permute the values of **variable m** in the oob cases and put these cases down the tree.

Subtract the number of votes for the correct class in the **variable-m-permuted** oob data from the number of votes for the correct class in the untouched oob data.

The average of this number over all trees in the forest is the raw importance score for **variable m**.

Gini impurity

Every time a split of a node is made on **variable m** the Gini impurity criterion for the two descendent nodes is less than the parent node.

Adding up the Gini decreases for each individual variable over all trees in the forest gives a fast assessment variable importance

Gini Impurity and Categorical Variable Deviance

The decision tree learning Gini impurity measure

We will not discuss the general Gini index used for other purposes

We can interpret the Gini impurity as the probability of being wrong if we randomly predict a case classes based on the fraction of cases in each class.

For each class i, let Pi be the fraction of cases in class i.
The probability being class in i is Pi and the probability of being wrong is 1-Pi
The expect error rates is $\sum$ Pi*(1-Pi) = 1 - $\sum$ Pi**2
If there is only 1 class the Gini impurity is 0.

Some tree models use the deviance as a splitting criterion

Deviance is the -2 log likelihood ratio comparing two models, the model and the split model.

The models can be based on the multinomial distribution.
The key part of the log likelihood appears on the next page along with its connection to entropy.

Three Categorical Splitting Criteria

Suppose the node has people in three classes: Hostile, Unstable, and Safe

Label:	Hostile	Unstable	Safe	
Ni	20	80	900	N=1000
Proportion	.02	.08	.900	1
Pi	$P_1 = .02$	$P_2 = .08$	$P_3 = .90$	

Three Criteria

RF Gini Index: 1 – ∑ Pi **2 = 1 - (.02**2 + .08**2 + .9**2) = 0.1832

Multinomial log likelihood: ∑ Ni*log(Pi)
Entropy ∑ Pi*log(Pi)

Note 1: Terms with Ni or Pi = 0 are excluded from the computation
Note 2: Since Pi = Ni/N the last two criteria yield equivalent results

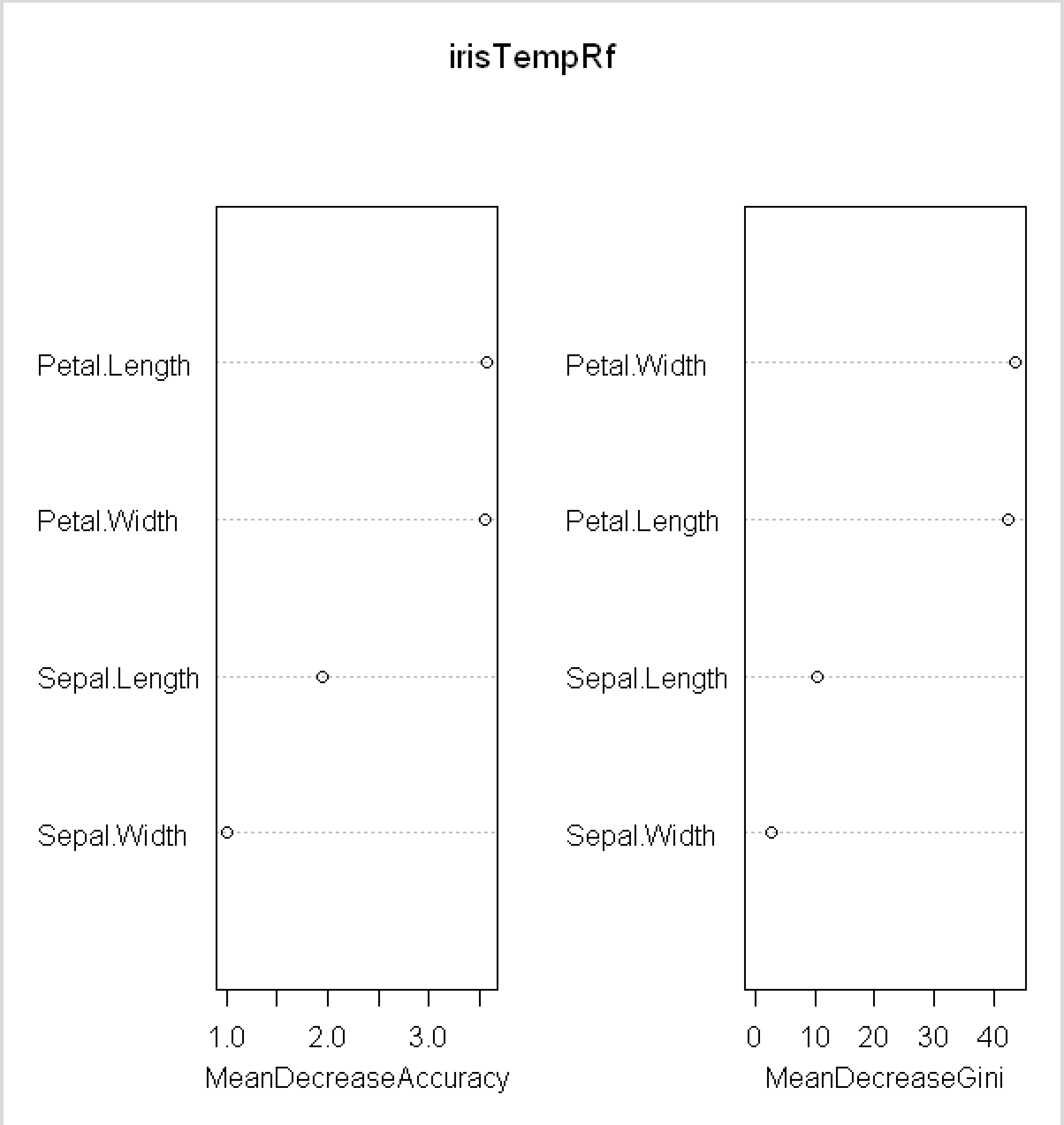
Gini Impurity Example: Iris Data

Node ID	Left Child	Right Child	Split Var	Split point	Status -1 = Leaf	Predicted Class
1	2	3	4	0.80	1	0
2	0	0	0	0.00	-1	1
3	4	5	4	1.65	1	0

```
Gini <- function(x) 1- (x/sum(x))**2
class <- iris[, 5]

# Node 1
tab <- table(class)
tab
setosa versicolor  virginica
      50         50         50
Gini(tab)  # 0.6666667

# Node 2: left child
# table  value .80  was rounded
g1 <- iris[, 4] <= .801
table( class[g1])
setosa versicolor  virginica
      50          0         0
# Gini = 0
```

```
# Node 3: right child
tab <- table( class[!g1])
 setosa  versicolor  virginica
      0          50          50
Gini(tab)
 0.5
#  0.5 + 0 < .6666667

#  Node 4: left  child of Node 3
g3  <- !g1 & iris[,4]  <= 1.65
tab <- table( class[g3] )
tab
   setosa versicolor  virginica
        0        48          4
Gini(tab)
 0.142
```

Classification: Iris Variable Importance



Proximities

Proximities are one of the most useful tools in random forests.

The proximities originally formed a N x N matrix.

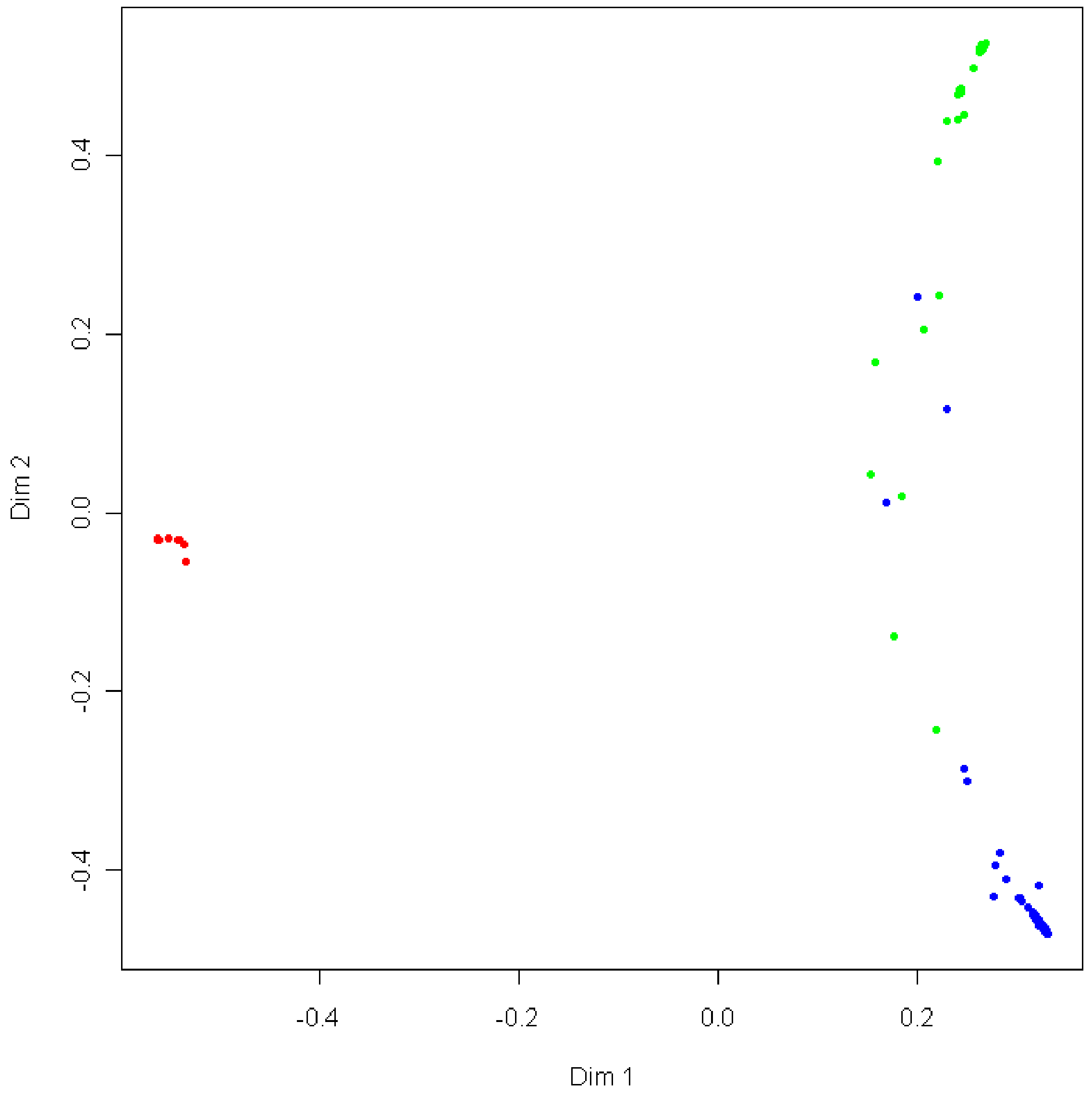
After a tree is grown, put all of the data, both training and oob, down the tree. If cases k and n are in the same terminal node increase their proximity by one. At the end, normalize the proximities by dividing by the number of trees.

Prototypes

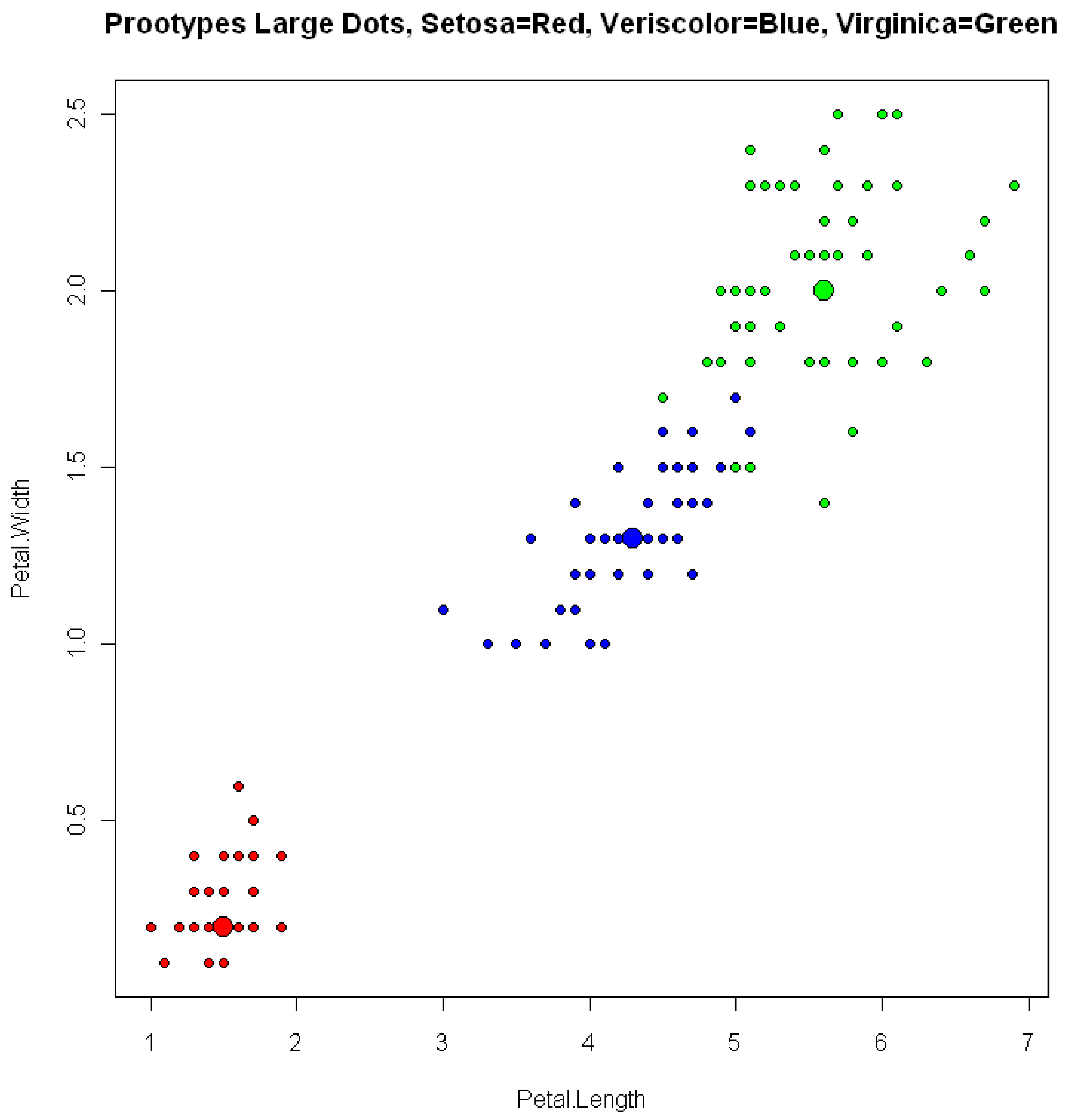
Prototypes are a way of getting a picture of how the variables relate to the classification.

For the j^{th} class, find the case that has the largest number of class j cases among its k nearest neighbors, determined using the proximities. Among these k cases we find the median, 25th percentile, and 75th percentile for each variable. The medians are the prototype for class j and the quartiles give an estimate of its stability.

Multidimensional scaling of points into 2D based proximity matrix. Many red points are overplotted

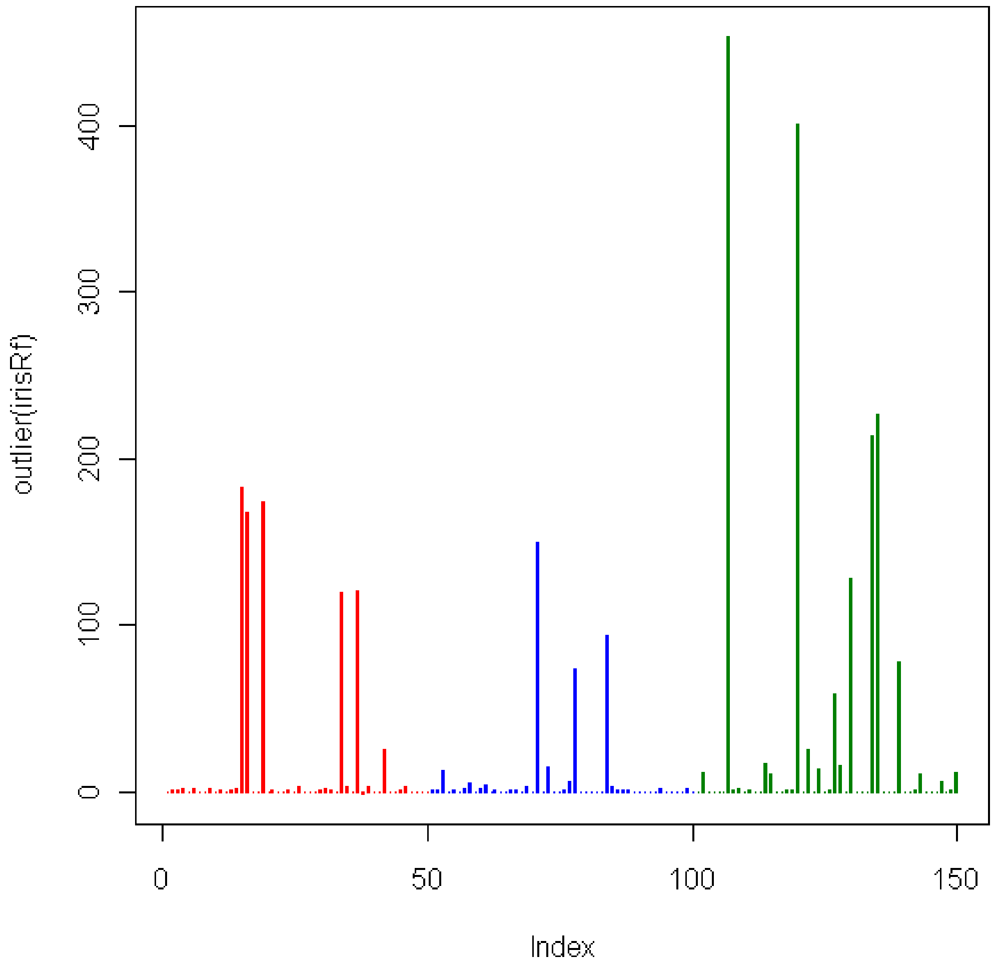


Prototype Example



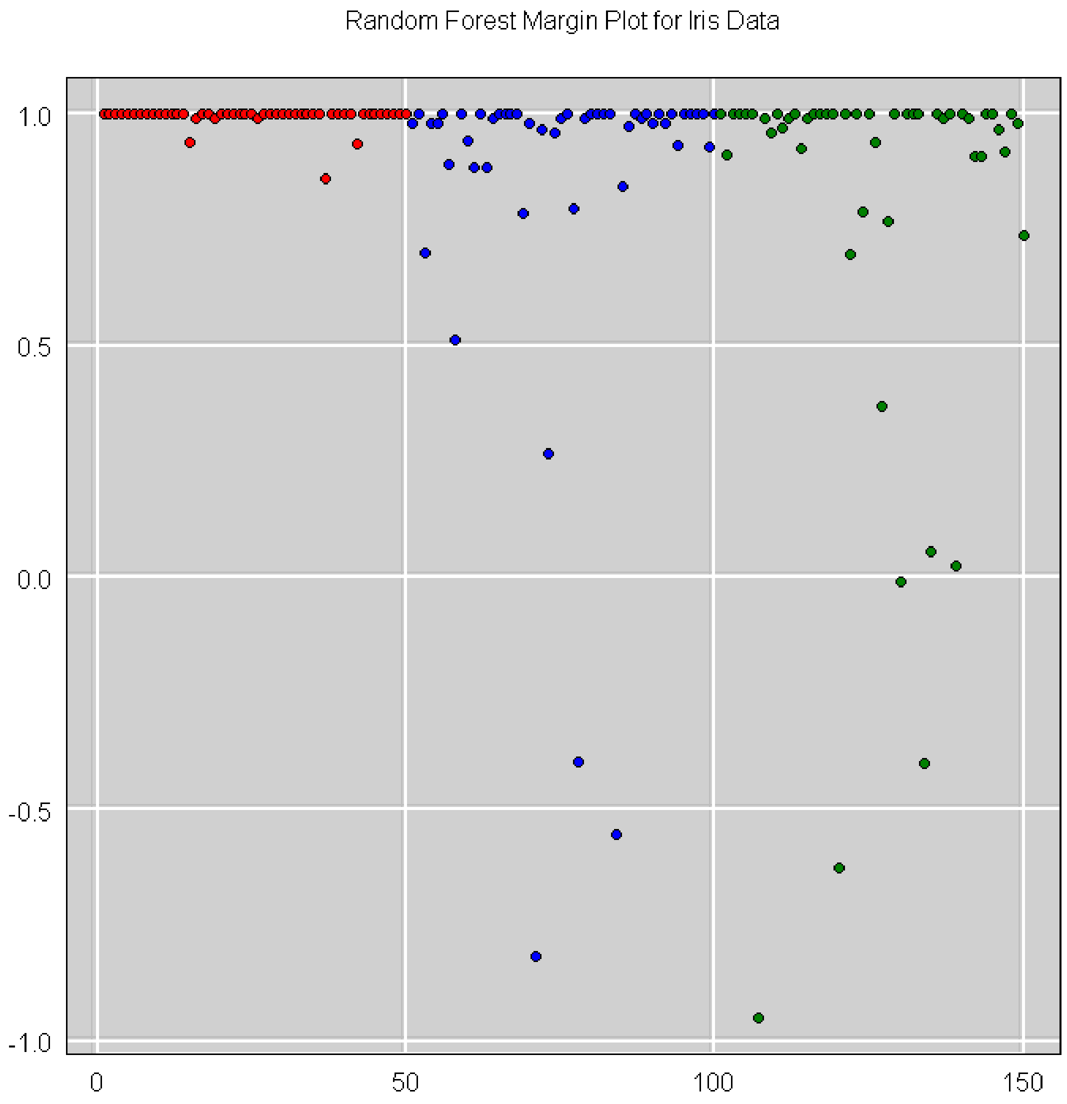
Outlier Measure

The outlier measure for a case is computed as n/sum(squared proximity), normalized by subtracting the median and divided by the MAD, within each class.

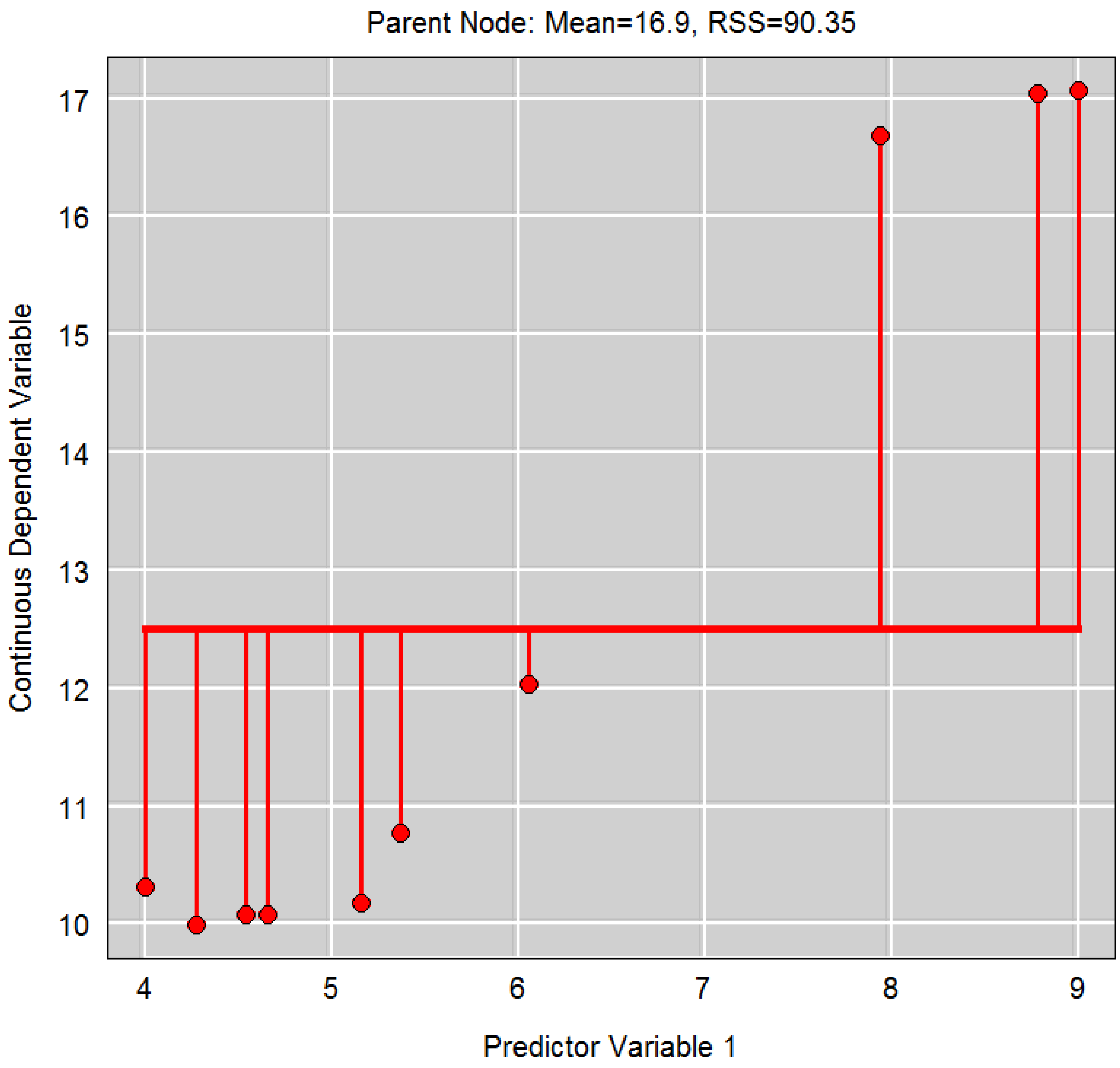


Margin Plot for Cases

Fraction of votes for Correct Class – Fraction for 2nd Highest Class



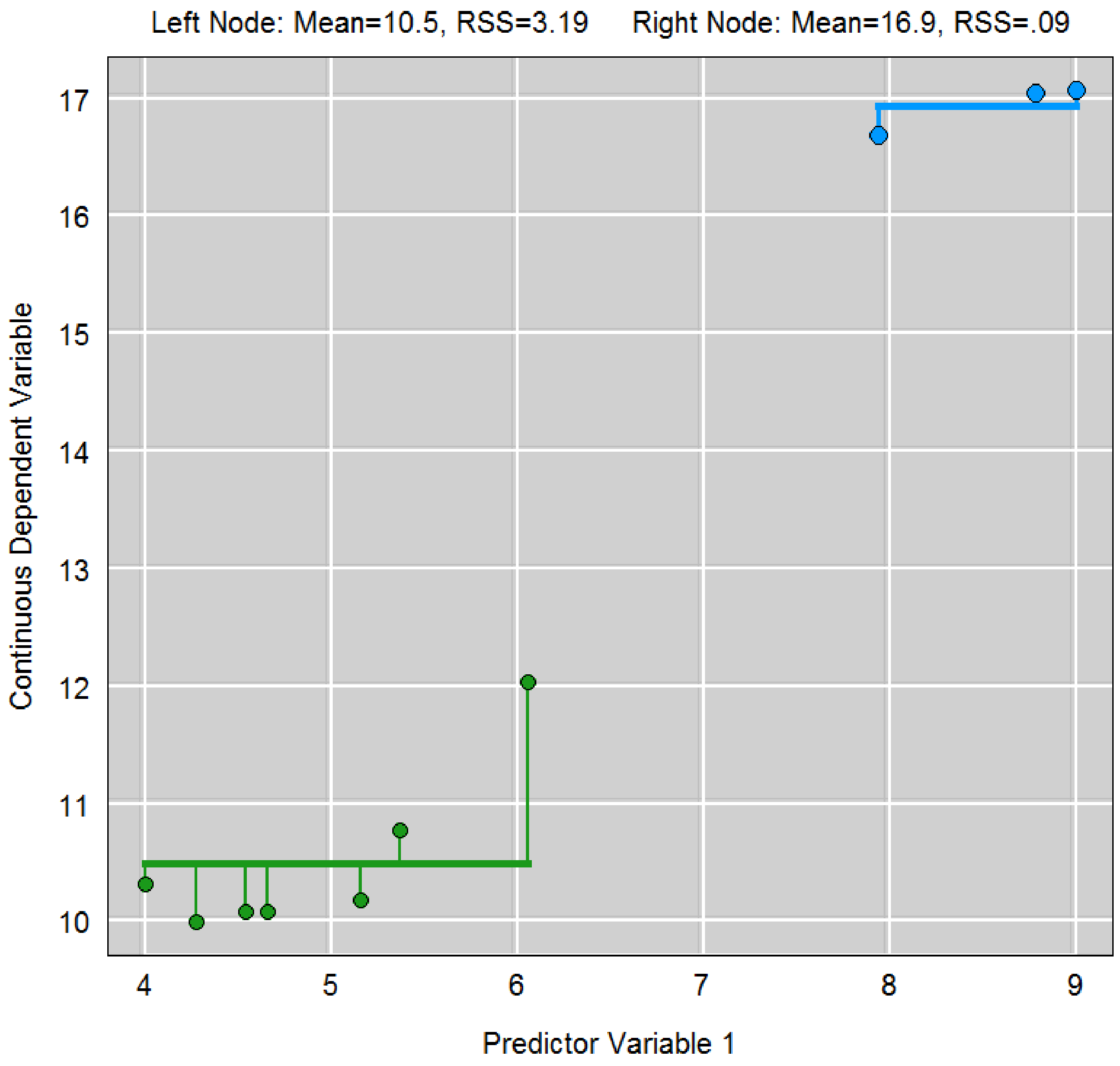
Regression Example: Node Predicted Value Is the Mean



Regression: Node Splitting Based on Residual Sum of Squares

Left Pts	Left Mean	Left RSS	Right Mean	Right RSS	Total RSS
1	10.3	.00	12.7	85.39	85.39
1:2	10.1	.05	13.0	77.38	77.43
1:3	10.1	.05	13.4	67.67	67.72
1:4	10.1	.06	14.0	54.76	54.82
1:5	10.1	.06	14.7	37.55	37.61
1:6	10.2	.40	15.7	18.04	18.44
1:7	10.5	3.19	16.9	.09	3.29
1:8	11.3	36.71	17.1	.00	36.71
1:9	11.9	66.35	17.1	.00	66.35

Regression: Node Splitting Based on Residual Sum of Squares



Regression Variable Importance

Mean square error

"For each tree, the prediction error on the out-of-bag portion of the data is recorded.
Then the same is done after permuting each predictor variable. "

For each predictor variable difference between permuted and original prediction errors are averaged over all trees, and normalized by the standard deviation of the differences.

(If the standard deviation of the differences is equal to 0 for a variable, the division is not done but the average is almost always equal to 0 in that case).

Residual sum of squares

This used total decrease in node impurities from splitting on the variable, averaged over all trees. For regression, impuritity is the **residual sum of squares**.