

**Homework 4**

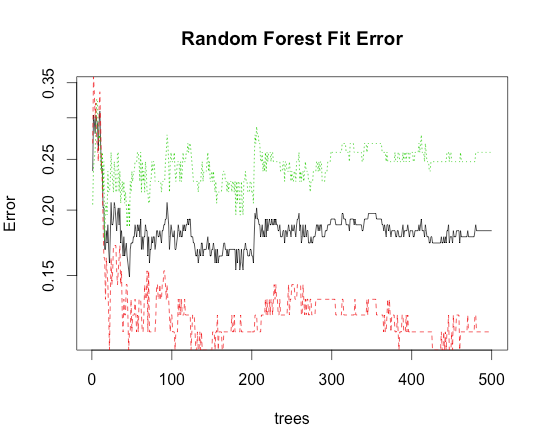
<Your Name Here>

**1) This problem uses two different ensemble methods to classify the sonar data.**

**1a) Use Random Forest to classify the sonar data.**

> testErr

[1] 0.1410256

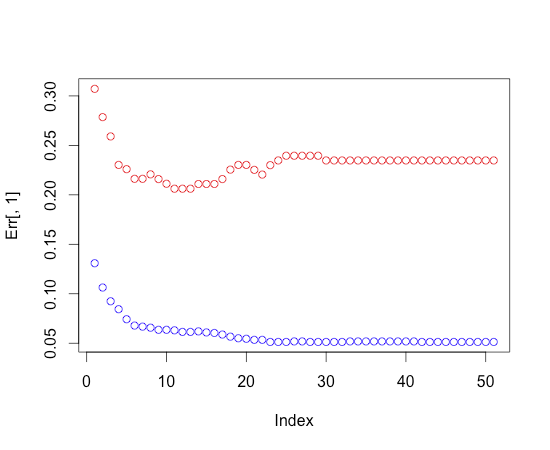




**1b) Use rpart to generate trees with a depth of two on randomly selected attributes. Use ridge regression to combine these trees to make predictions. Use cross validation to choose the best lambda and calculate the test error for this lambda.**

> min(Err[,1])

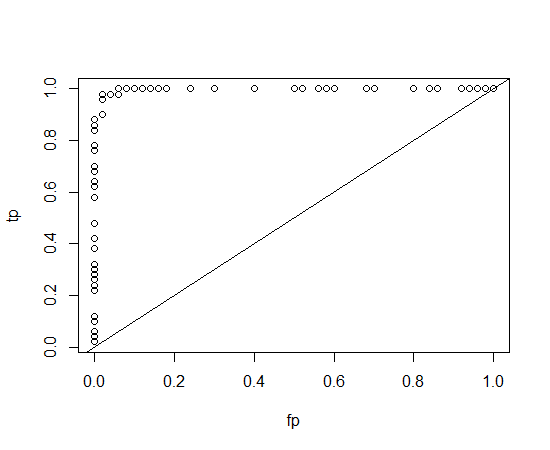
[1]0.2061905



**1c) Which model resulted in the smallest test error?**

Random Forest has the smallest test error of 0.1410256

**2) This problem uses the Iris Data Set. It only involves the Versicolor and Virginica species (rows 51 through 150). Use cross validated ridge regression to classify these two species. Create and plot a ROC curve for this classification method.**



**3) Classify the wine quality data using some Boosted method. Ada Boost won't work. What does? Hints for gradient boosted method: require(gbm) and gbm.perf(gbm1,method="cv")**

**3a) In Homework 3 Problem 4 ridge regression was performed on the wine quality data. Compare the results of using the Boosted method regression with the ridge method.**

dataWine<-read.csv("winequality-red.csv",sep=";",header=T)

dataTest<-dataWine[1:1400,];

dataTrain<-dataWine[1401:1599,];

# generalized boosting

names(dataWine)

# [1] "fixed.acidity" "volatile.acidity" "citric.acid"

# [4] "residual.sugar" "chlorides" "free.sulfur.dioxide"

# [7] "total.sulfur.dioxide" "density" "pH"

# [10] "sulphates" "alcohol" "quality"

gbmFit<-gbm(quality~.,data=dataTrain,distribution="gaussian",n.trees=3000, shrinkage=0.005, interaction.depth=3, bag.fraction=0.5, train.fraction=0.5,cv.folds=5)

> best.iter <- gbm.perf(gbmFit,method="cv")

> print(best.iter)

[1] 750

> #[1] 2775

>

> summary(gbmFit,n.trees=best.iter)

var rel.inf

1 sulphates 20.216169

2 volatile.acidity 16.677293

3 density 16.245751

4 chlorides 10.158842

5 pH 6.210874

6 fixed.acidity 6.133560

7 citric.acid 6.124639

8 free.sulfur.dioxide 5.769965

9 alcohol 5.685235

10 total.sulfur.dioxide 4.393993

11 residual.sugar 2.383679

> # var rel.inf

> #1 alcohol 24.558930

> #2 sulphates 14.463651

> #3 volatile.acidity 12.875163

> #4 total.sulfur.dioxide 11.050594

> #5 citric.acid 6.362643

> #6 pH 6.205398

> #7 chlorides 5.793746

> #8 residual.sugar 5.604007

> #9 fixed.acidity 5.497743

> #10 density 4.542594

> #11 free.sulfur.dioxide 3.045531

**3b) Use the Random Forest technique on the wine quality data.**

> library(randomForest)

> rfFit<-randomForest(quality~.,dataTrain,importance=TRUE,do.trace=100 )

| Out-of-bag |

Tree | MSE %Var(y) |

100 | 0.3163 48.87 |

200 | 0.3117 48.17 |

300 | 0.3117 48.17 |

400 | 0.3118 48.17 |

500 | 0.3096 47.84 |

> print(rfFit)

Call:

randomForest(formula = quality ~ ., data = dataTrain, importance = TRUE, do.trace = 100)

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 3

Mean of squared residuals: 0.309564

% Var explained: 52.16

> q.predict<-predict(rfFit,dataTest)

> q.predict<-round(q.predict)

> errTest<-1-sum(q.predict==dataTest$quality)/length(dataTest$quality)

> print(errTest)

[1] 0.4422111

**3c) Compare the results of Boosted method and Random Forest.**

the results using Boosted method and random forest are comparable.

**4) See if you can improve on regression-based classification of the iris data that we did in class. Classify the iris data set with second degree terms added using a ridge regression. (ie supplement the original 4 attributes x1, x2, x3, and x4 by including the 10 second degree terms ( x1\*x1, x1\*x2, x1\*x3, ... ) for a total of 14 attributes.) Use multiclass to classify the data and then compare the results with the results obtained in class.**

**It is fine to use brute force to add these attributes. For those who are adventurous, investigate the function mutate in the package plyr.**

10 attributes were added to the table which were terms such as x1\*x1, x1\*x2,....,x4\*x4.

First train on Y1 (iris-setosa) and use cross validated ridge regression to find the best lambda. Error matrix was as below :

Err

> Err

TrainErr TestErr lambda

[1,] 0 0 100.00000000

[2,] 0 0 63.09573445

[3,] 0 0 39.81071706

[4,] 0 0 25.11886432

[5,] 0 0 15.84893192

[6,] 0 0 10.00000000

[7,] 0 0 6.30957344

[8,] 0 0 3.98107171

[9,] 0 0 2.51188643

[10,] 0 0 1.58489319

[11,] 0 0 1.00000000

[12,] 0 0 0.63095734

[13,] 0 0 0.39810717

[14,] 0 0 0.25118864

[15,] 0 0 0.15848932

[16,] 0 0 0.10000000

[17,] 0 0 0.06309573

[18,] 0 0 0.03981072

[19,] 0 0 0.02511886

[20,] 0 0 0.01584893

[21,] 0 0 0.01000000

Both training and test errors were zero which means there was an accurate prediction.

Second train on Y2(iris-versicolor) and repeat the same procedure. Error matrix was as below :

Err

> Err

[,1] [,2] [,3]

[1,] 0.24370370 0.25333333 100.00000000

[2,] 0.22148148 0.22666667 63.09573445

[3,] 0.17555556 0.18666667 39.81071706

[4,] 0.14666667 0.15333333 25.11886432

[5,] 0.09259259 0.10000000 15.84893192

[6,] 0.07037037 0.06666667 10.00000000

[7,] 0.04370370 0.05333333 6.30957344

[8,] 0.03333333 0.04000000 3.98107171

[9,] 0.03407407 0.03333333 2.51188643

[10,] 0.03037037 0.04000000 1.58489319

[11,] 0.03185185 0.04666667 1.00000000

[12,] 0.03407407 0.06000000 0.63095734

[13,] 0.03555556 0.06666667 0.39810717

[14,] 0.03481481 0.06666667 0.25118864

[15,] 0.03703704 0.05333333 0.15848932

[16,] 0.03555556 0.04666667 0.10000000

[17,] 0.03481481 0.04666667 0.06309573

[18,] 0.02962963 0.04666667 0.03981072

[19,] 0.02740741 0.04666667 0.02511886

[20,] 0.02518519 0.05333333 0.01584893

[21,] 0.02296296 0.05333333 0.01000000

Third, train on Y3(iris-virginica) and repeat the same procedure. Error matrix was as below :

Err

> Err

[,1] [,2] [,3]

[1,] 0.04888889 0.05333333 100.00000000

[2,] 0.04814815 0.05333333 63.09573445

[3,] 0.04370370 0.05333333 39.81071706

[4,] 0.03703704 0.04000000 25.11886432

[5,] 0.03259259 0.04000000 15.84893192

[6,] 0.02888889 0.03333333 10.00000000

[7,] 0.03111111 0.03333333 6.30957344

[8,] 0.02666667 0.04000000 3.98107171

[9,] 0.02666667 0.04000000 2.51188643

[10,] 0.02296296 0.03333333 1.58489319

[11,] 0.01925926 0.04000000 1.00000000

[12,] 0.01703704 0.04000000 0.63095734

[13,] 0.01629630 0.04000000 0.39810717

[14,] 0.01555556 0.04000000 0.25118864

[15,] 0.01555556 0.04000000 0.15848932

[16,] 0.01555556 0.03333333 0.10000000

[17,] 0.01481481 0.03333333 0.06309573

[18,] 0.01407407 0.03333333 0.03981072

[19,] 0.01407407 0.03333333 0.02511886

[20,] 0.01407407 0.03333333 0.01584893

[21,] 0.01407407 0.03333333 0.01000000

Now, we can proceed to computing the misclassification error. Basically, the number of correct positives in each class is given by

sum(Y1>0 & Y1h)

sum(Y2>0 & Y2h)

sum(Y3>0 & Y3h)

Choose the best lambda that corresponds to the lowest training and test errors and generate the ridge regression models. Now, compute the actual predictions and then calculate the number of correct positives.

These values for the original code in the R file was 50, 34 and 43 respectively. In the current solution code, these values were 50, 49 and 48(i.e) there was a significant improvement in the classification of Y2 and Y3 by adding more attributes to the data which are higher order terms.

**5) This is a multi-class problem. Consider the Glass Identification Data Set from the UC Irvine Data Repository. The Data is located at the web site:**

**http://archive.ics.uci.edu/ml/datasets/Glass%2BIdentification**

**This problem will only work with building and vehicle window glass (classes 1,2 and 3), so it only uses the first 163 rows of data. (Ignore rows 164 through 214) With this set up this is a three class problem. Use ridge regression to classify this data into the three classes: building windows float processed, building windows non float processed, and vehicle windows float processed.**

Below are the lowest error values for all classes and the lambdas that achieved them. Class 3 had the lowest error value and class 2 did really poorly.

> ErrC1[1,]

Err Class Lambda

43 0.2576687 1 0.1584893

> ErrC2[1,]

Err Class Lambda

23 0.5337423 2 3.981072

> ErrC3[1,]

Err Class Lambda

3 0.1104294 3 100