**STA 545 ASSIGNMENT #4**

**SURUCHI JAIKUMAR AHUJA**

1. **Question 2**

The wine data set contains the results of a chemical analysis of 178 wines grown over the decade 1970-1979 in the same region of Italy, but derived from three different cultivars (Barolo, Grignolino, and Barbera). The Babera wines were predominately from a period that was much later than that of the Barolo and Grignolino wines. The analysis determined the quantities MalicAcid, Ash, AlcAsh, Mg, Phenols, Proa, Color, Hue, OD, and Proline. There are 50 Barolo wines, 71 Grignolino wines, and 48 Barbera wines.

**OUTPUT:**

**Full Tree**

Node number 1: 142 observations, complexity param=0.5058824

predicted class=2 expected loss=0.5985915 P(node) =1

left son=2 (54 obs) right son=3 (88 obs)

Node number 2: 54 observations, complexity param=0.05882353

predicted class=1 expected loss=0.1481481 P(node) =0.3802817

left son=4 (48 obs) right son=5 (6 obs)

Node number 3: 88 observations, complexity param=0.3411765

predicted class=2 expected loss=0.3863636 P(node) =0.6197183

left son=6 (51 obs) right son=7 (37 obs)

Node number 4: 48 observations, complexity param=0.02352941

predicted class=1 expected loss=0.04166667 P(node) =0.3380282

left son=8 (46 obs) right son=9 (2 obs)

Node number 5: 6 observations, complexity param=0.01176471

predicted class=3 expected loss=0.1666667 P(node) =0.04225352

left son=10 (1 obs) right son=11 (5 obs)

Node number 6: 51 observations

predicted class=2 expected loss=0.01960784 P(node) =0.3591549

Node number 7: 37 observations, complexity param=0.03529412

predicted class=3 expected loss=0.1081081 P(node) =0.2605634

left son=14 (3 obs) right son=15 (34 obs)

Node number 8: 46 observations

predicted class=1 expected loss=0 P(node) =0.3239437

Node number 9: 2 observations

predicted class=2 expected loss=0 P(node) =0.01408451

Node number 10: 1 observations

predicted class=2 expected loss=0 P(node) =0.007042254

Node number 11: 5 observations

predicted class=3 expected loss=0 P(node) =0.03521127

Node number 14: 3 observations

predicted class=2 expected loss=0 P(node) =0.02112676

Node number 15: 34 observations, complexity param=0.005882353

predicted class=3 expected loss=0.02941176 P(node) =0.2394366

left son=30 (3 obs) right son=31 (31 obs)

Node number 30: 3 observations, complexity param=0.005882353

predicted class=3 expected loss=0.3333333 P(node) =0.02112676

left son=60 (1 obs) right son=61 (2 obs)

Node number 31: 31 observations

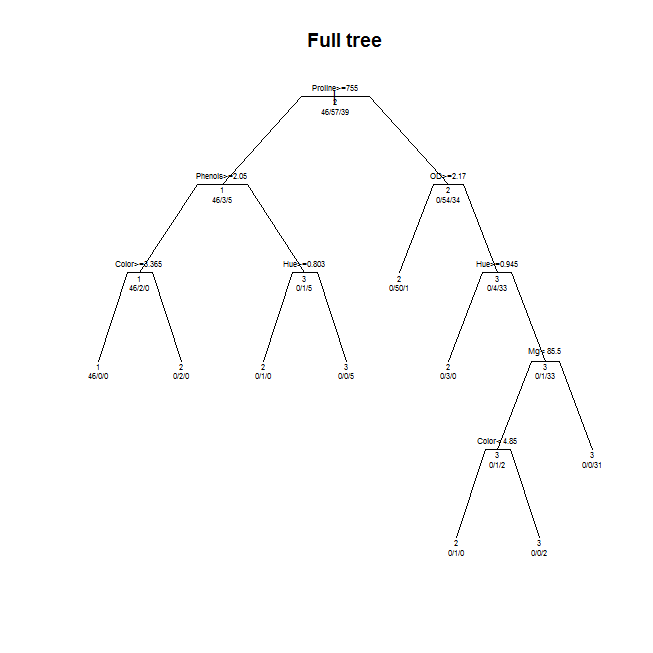
predicted class=3 expected loss=0 P(node) =0.2183099

Node number 60: 1 observations

predicted class=2 expected loss=0 P(node) =0.007042254

Node number 61: 2 observations

predicted class=3 expected loss=0 P(node) =0.01408451



**Pruned Tree:**

n= 142

Node number 1: 142 observations, complexity param=0.5058824

predicted class=2 expected loss=0.5985915 P(node) =1

left son=2 (54 obs) right son=3 (88 obs)

Node number 2: 54 observations, complexity param=0.05882353

predicted class=1 expected loss=0.1481481 P(node) =0.3802817

left son=4 (48 obs) right son=5 (6 obs)

Node number 3: 88 observations, complexity param=0.3411765

predicted class=2 expected loss=0.3863636 P(node) =0.6197183

left son=6 (51 obs) right son=7 (37 obs)

Node number 4: 48 observations, complexity param=0.02352941

predicted class=1 expected loss=0.04166667 P(node) =0.3380282

left son=8 (46 obs) right son=9 (2 obs)

Node number 5: 6 observations

predicted class=3 expected loss=0.1666667 P(node) =0.04225352

Node number 6: 51 observations

predicted class=2 expected loss=0.01960784 P(node) =0.3591549

Node number 7: 37 observations, complexity param=0.03529412

predicted class=3 expected loss=0.1081081 P(node) =0.2605634

left son=14 (3 obs) right son=15 (34 obs)

Node number 8: 46 observations

predicted class=1 expected loss=0 P(node) =0.3239437

Node number 9: 2 observations

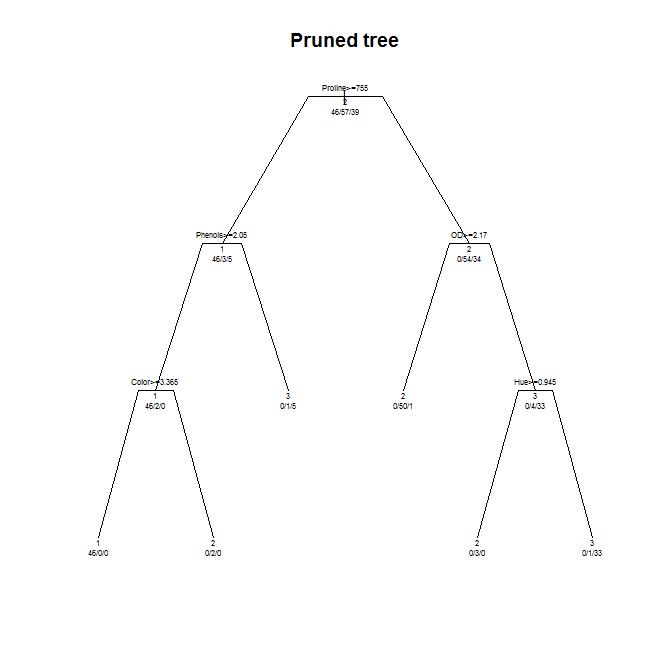
predicted class=2 expected loss=0 P(node) =0.01408451

Node number 14: 3 observations

predicted class=2 expected loss=0 P(node) =0.02112676

Node number 15: 34 observations

predicted class=3 expected loss=0.02941176 P(node) =0.2394366



1. **Question 3**

The Boston dataset was taken.

Bagging, Boosting and Random forests was applied to the Boston dataset. The models were fit on the training data set and evaluated on the test set. Also logistic regression was performed on the dataset for the comparisons of results.

Ensemble methods use multiple learning algorithms to obtain better [predictive performance](https://en.wikipedia.org/wiki/Predictive_inference) than could be obtained from any of the constituent learning algorithms. Ensemble methods tend to yield better results when there is a significant diversity among the models. Hence Ensemble methods results are more accurate that non ensemble methods as ensemble methods can be trained and then be used to make predictions.

A committee machine is a type of [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) in which the responses of multiple neural networks are combined into a single response. It has two types – Static structures and dynamic structures. The advantages of this method are by averaging, the average error is reduced, over training risk is minimized and the effect of one (or more) of the experts training to a local minima is minimized. One of the drawback is Randomness of the noise in the data means the network with the best validation set performance will not necessarily have the best test set performance.

**OUTPUT:**

**Bagging:**

bag.fit <- randomForest(High~., data = training, n.tree= 10000, mtry = 13)

importance(bag.fit)

MeanDecreaseGini

zn 0.0000000000

indus 0.0000000000

chas 0.3160987076

nox 1.9170613605

rm 2.3538387313

age 5.8681326663

dis 4.1321816254

rad 125.8535206567

tax 0.7766051540

ptratio 0.0000000000

black 1.0472731984

lstat 1.5709999423

medv 2.5472805501

Bagging Test error : > misclass\_bag

[1] 0.0297029703

**Boosting :**

Here the adaboost model was fit with two different shrinkages (0.1 and 0.6)

These are the names in both the models.

[1] "initF" "fit" "train.error" "valid.error" "oobag.improve"

[6] "trees" "c.splits" "bag.fraction" "distribution" "interaction.depth"

[11] "n.minobsinnode" "num.classes" "n.trees" "nTrain" "train.fraction"

[16] "response.name" "shrinkage" "var.levels" "var.monotone" "var.names"

[21] "var.type" "verbose" "data" "Terms" "cv.folds"

[26] "call" "m"

|  |
| --- |
| > summary(boost.fit)  var rel.inf  black black 89.3376586198763  rad rad 8.7190763499962  dis dis 0.6891822860581  medv medv 0.3672127731498  nox nox 0.3185869031642  lstat lstat 0.3132726252798  age age 0.1088038182130  rm rm 0.0752286494064  tax tax 0.0383984127711  indus indus 0.0290381369946  chas chas 0.0032058302299  ptratio ptratio 0.0003355948606  zn zn 0.0000000000000  > summary(boost.fit2)  var rel.inf  rad rad 55.422784217498496  nox nox 20.127585906056346  dis dis 16.018208271380065  lstat lstat 2.718946484372274  chas chas 1.744403413423311  black black 1.667652066649364  medv medv 0.941377413881352  indus indus 0.597237466699339  age age 0.380665821889704  rm rm 0.365117497470965  tax tax 0.016014326076618  ptratio ptratio 0.000007114602165  zn zn 0.000000000000000 |

|  |
| --- |
|  |
|  |

**Boosting Test Error :**

First model with shrinkage 0.1 – misclass\_boost

[1] 0.03094878958

Second model with shrinkage 0.6 - > misclass\_boost2

[1] 0.04966662233

**Random forests :**

> rf.fit <- randomForest(High~., data = training, n.tree= 10000)

> varImpPlot(rf.fit)

> importance(rf.fit)

MeanDecreaseGini

zn 0.07641727729

indus 18.12829370933

chas 0.33386978223

nox 14.57949255969

rm 2.15590229752

age 4.30089617677

dis 7.22685056204

rad 38.47672775296

tax 32.97653383917

ptratio 12.95674382444

black 2.58406814865

lstat 3.97905269996

medv 6.99145070254

**Random Forest Test error** : > misclass\_rf

[1] 0.0198019802

**Logistic Regression :**

**Logistic Regression Errors :**

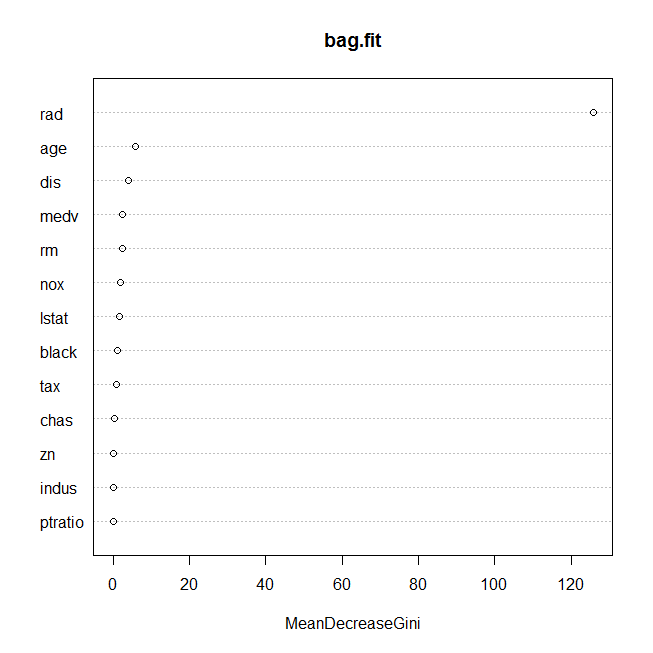
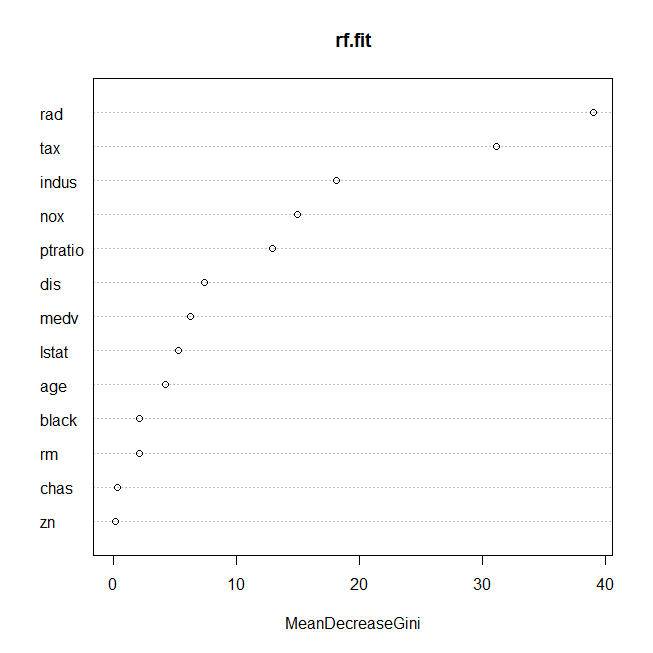
> train\_err

[1] 0.01975308642

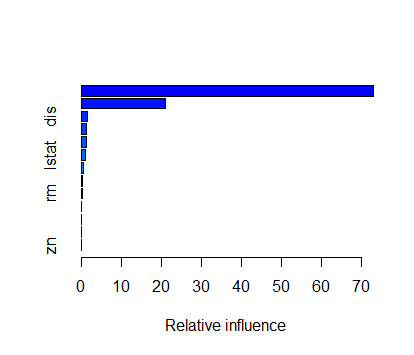
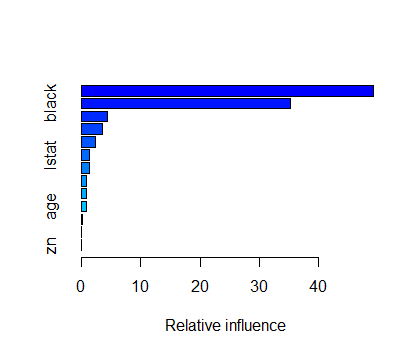
> test\_err

[1] 0.0297029703

Bagging Random Forests

Boost model -1 Boost model - 2

1. **Question 4**

Random forests is a general technique of random decision forest, that is an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis), that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees.

Here the spam data set was taken from the ElemStatPackage and a series of random forest classifiers was fitted. The sensitivity to m (the number of randomly selected inputs for each tree) was tested. Here the m values were taken as 1, 3, 5, and 8. As the m value increases the OOB error decreases and so does the misclassification error.

**RF1 – for m = 1**

Call:

randomForest(formula = as.factor(spam) ~ ., data = spam\_train, n.tree = 1000, mtry = 1, keep.inbag = T, oob.prox = T, importance = T, main = "random forest for m=10")

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 1

OOB estimate of error rate: 7.69%

Confusion matrix:

0 1 class.error

0 2165 58 0.02609087

1 225 1233 0.15432099

misclass1

[1] 0.07065217

**RF2 – for m = 3**

Call:

randomForest(formula = as.factor(spam) ~ ., data = spam\_train, n.tree = 1000, mtry = 3, keep.inbag = T, oob.prox = T, importance = T)

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 3

OOB estimate of error rate: 5.41%

Confusion matrix:

0 1 class.error

0 2146 77 0.03463788

1 122 1336 0.08367627

misclass2

[1] 0.04673913

**RF3 – for m =5**

Call:

randomForest(formula = as.factor(spam) ~ ., data = spam\_train, n.tree = 1000, mtry = 5, keep.inbag = T, oob.prox = T, importance = T)

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 5

OOB estimate of error rate: 5.08%

Confusion matrix:

0 1 class.error

0 2155 68 0.03058929

1 119 1339 0.08161866

misclass3

[1] 0.0423913

**RF4 for m = 8**

Call:

randomForest(formula = as.factor(spam) ~ ., data = spam\_train, n.tree = 1000, mtry = 8, keep.inbag = T, oob.prox = T, importance = T)

Type of random forest: classification

Number of trees: 500

No. of variables tried at each split: 8

OOB estimate of error rate: 4.81%

Confusion matrix:

0 1 class.error

0 2145 78 0.03508772

1 99 1359 0.06790123

> misclass4

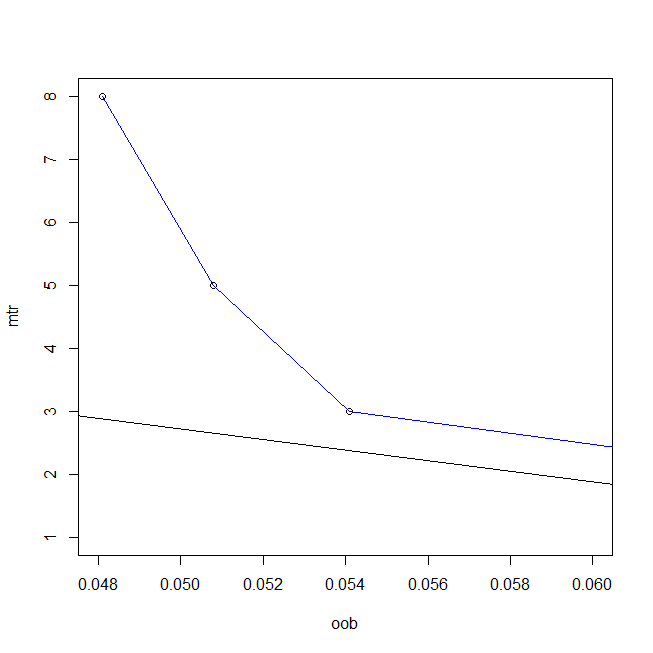
[1] 0.04565217

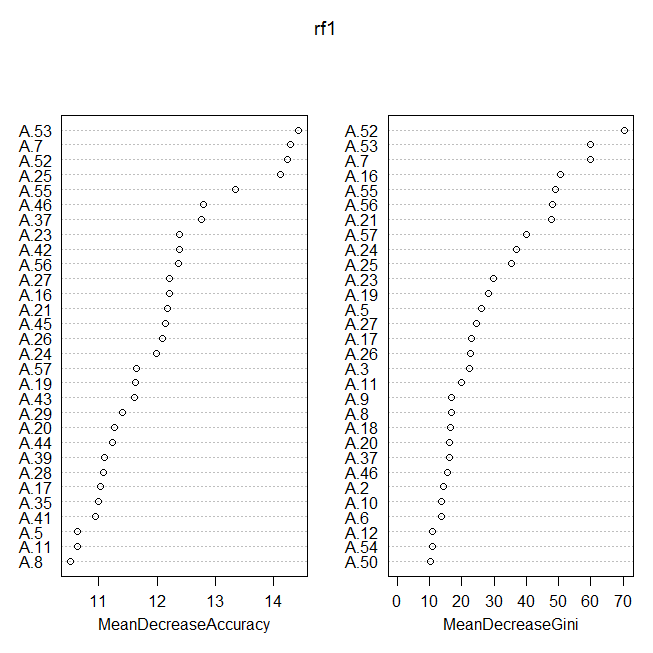
RF1 - OOB estimate of error rate: 7.69%

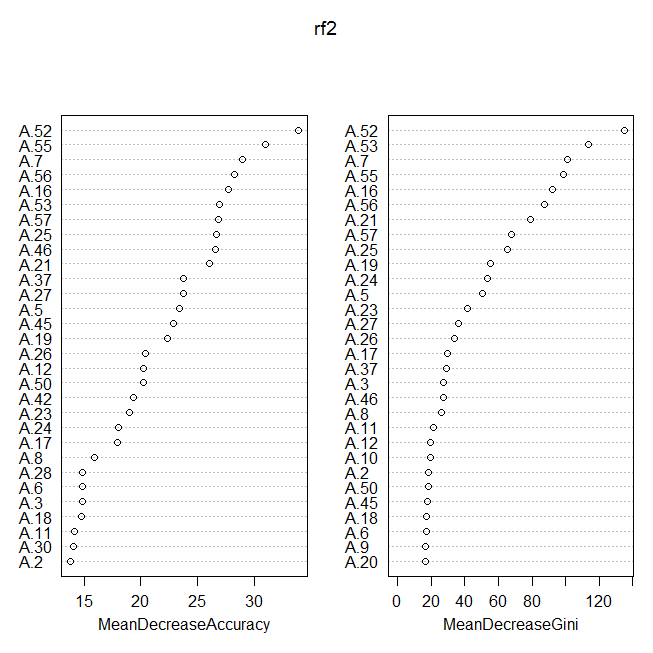
RF2 - OOB estimate of error rate: 5.41%

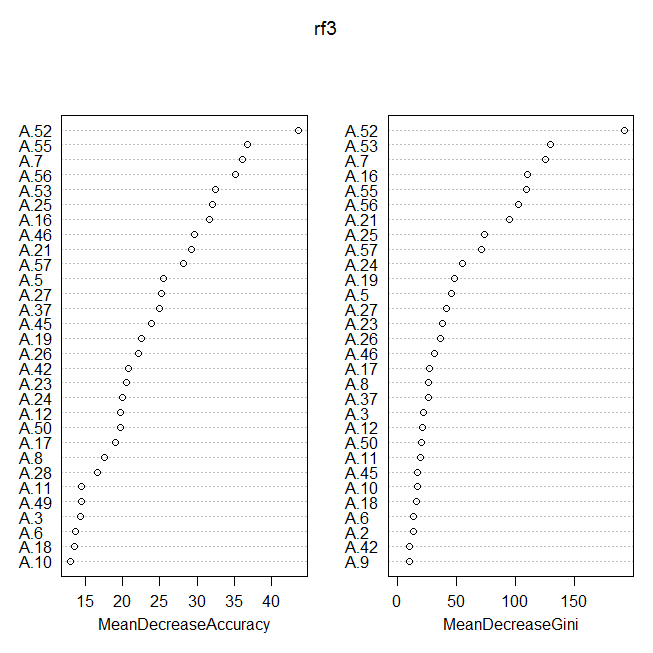
RF3 - OOB estimate of error rate: 5.08%

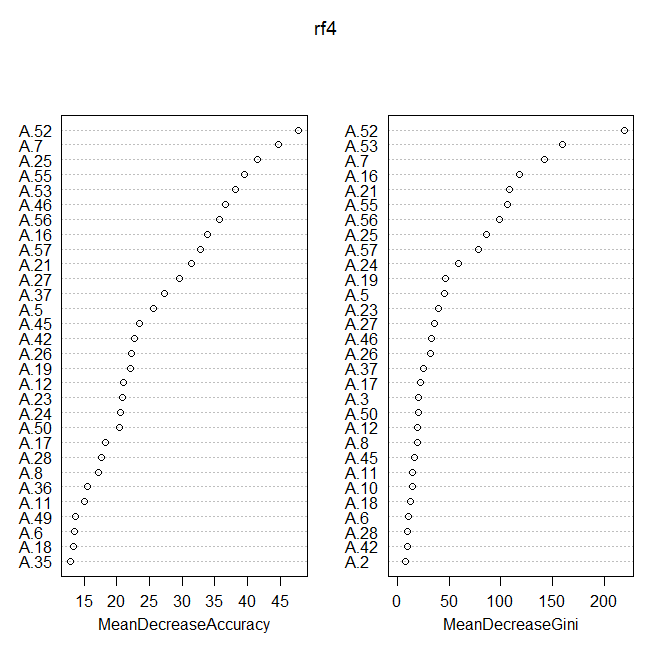
RF4 - OOB estimate of error rate: 4.81%











1. **Question 5**

A neural network is a two-stage regression or classiﬁcation model, typically represented by a network diagram. Here a neural network was fitted on the spam data which is available through the ElemStatLearn Package. The hold out method was used to calculate number of neurons used in the layer. In the spam dataset when a neural network was fitted, 2 neurons were found to be used in the layer. Logistic regression and artificial neural networks shave common roots in statistical pattern recognition.

For this dataset the logistic regression has a better model interpretation when compared to neural networks. The predictive accuracy is almost similar in both the cases.

**OUTPUT:**

names(nn)

[1] "call" "response" "covariate" "model.list"

[5] "err.fct" "act.fct" "linear.output" "data"

[9] "net.result" "weights" "startweights" "generalized.weights"

[13] "result.matrix"

> error\_train

[1] 0.04673913043

> error\_test

[1] 0.02713956522

> summary(new.output)

Length Class Mode

neurons 2 -none- list

net.result 921 -none- numeric

1. **Question 7**

**a)**

Support vector machines (SVM) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data and recognize patterns, used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis).

The linear support vector machine (svm) was fit with varying cost parameters. The training and test errors was plot across the spectrum of cost parameters over the range of 0.01 to 10.

Best parameter:

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 10

gamma: 0.05555556

Number of Support Vectors: 357

> train\_error

[1] 0.8364486 0.8329439 0.8352804 0.8341121 0.8352804 0.8352804 0.8376168 0.8341121 0.8352804 0.8352804

[11] 0.8364486

> test\_error

[1] 0.8271028 0.8364486 0.8364486 0.8271028 0.8271028 0.8271028 0.8317757 0.8317757 0.8317757 0.8317757

[11] 0.8317757

**b)**

The radial support vector machine (svm) was fit with varying cost parameters. The training and test errors was plot across the spectrum of cost parameters over the range of 0.01 to 10.

Best parameter:

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 10

gamma: 0.05555556

Number of Support Vectors: 358

> test\_error

[1] 0.6448598 0.8084112 0.8317757 0.8364486 0.8457944 0.8224299 0.8317757 0.8411215 0.8411215 0.8177570

[11] 0.8271028

> train\_error

[1] 0.6016355 0.8165888 0.8294393 0.8294393 0.8364486 0.8422897 0.8422897 0.8457944 0.8434579 0.8469626

[11] 0.8504673

**c)**

The polynomial support vector machine (svm) with kernel =2, was fit with varying cost parameters. The training and test errors was plot across the spectrum of cost parameters over the range of 0.01 to 10.

Best parameter:

Parameters:

SVM-Type: C-classification

SVM-Kernel: polynomial

cost: 10

degree: 2

gamma: 0.05555556

coef.0: 0

Number of Support Vectors: 375

> test\_error

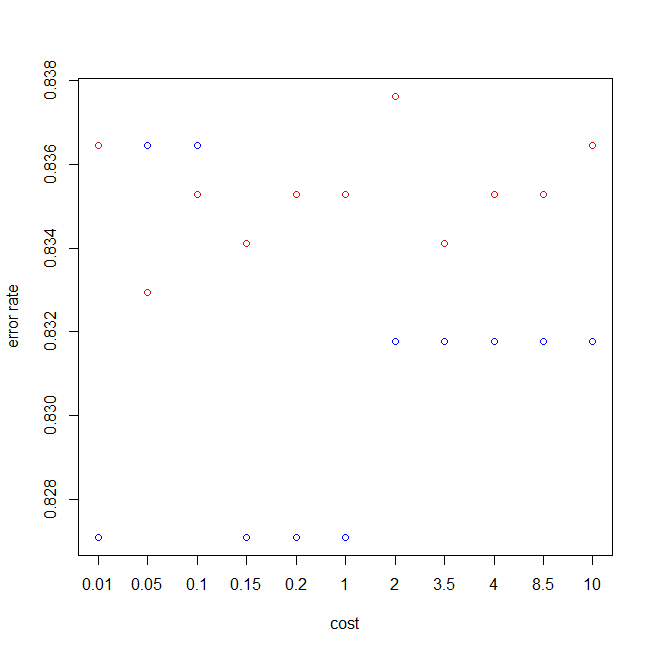
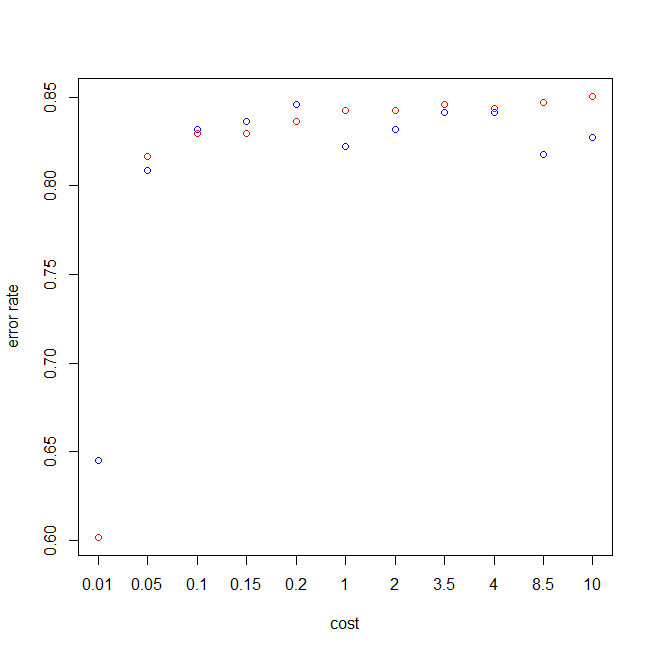
[1] 0.6682243 0.6869159 0.7009346 0.7663551 0.8084112 0.8130841 0.8084112 0.8271028 0.8271028 0.8457944

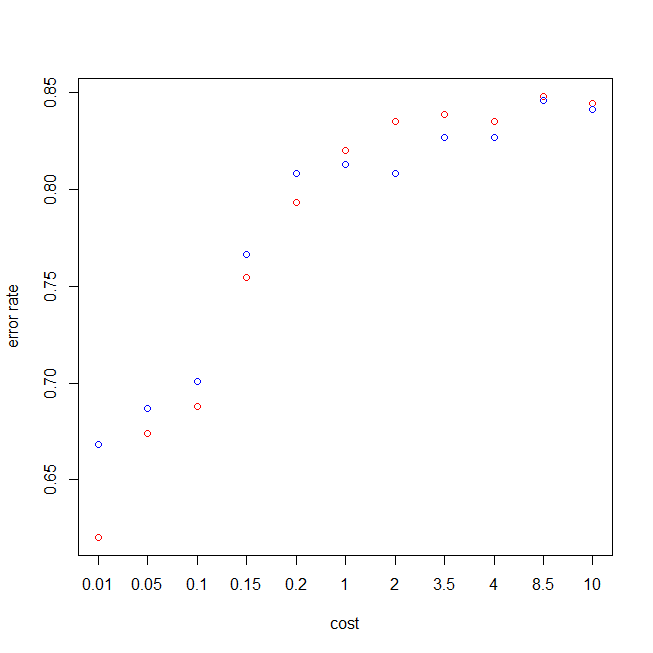
[11] 0.8411215

> train\_error

[1] 0.6203271 0.6740654 0.6880841 0.7546729 0.7932243 0.8200935 0.8352804 0.8387850 0.8352804 0.8481308

[11] 0.8446262

The figures below show the training and test errors for a linear, radial, polynomial svm model. The red dots show training error and blue dots show test error in the linear svm model. 



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THE END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***