SDM_Assignment5_5

Sri Balaji Muruganandam

8/12/2021

Setting Working Directory

```
rm(list = ls())
setwd("G:\\SDM_Sem01\\Assignment5")
```

Importing necessary libraries

```
options(warn=-1)
library(rpart)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(geneplotter)
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
## Loading required package: lattice
```

```
## Loading required package: annotate
```

```
## Loading required package: AnnotationDbi
```

```
## Loading required package: stats4
```

```
## Loading required package: IRanges
```

```
## Loading required package: S4Vectors
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

```
##
## Attaching package: 'IRanges'
```

```
## The following object is masked from 'package:grDevices':
##
##     windows
```

```
## Loading required package: XML
```

```
options(warn=0)
```

Loading the Spam dataset

```
load("spam.RData")
dim(spam)
```

```
## [1] 4601   58
```

Exploring the high level overview of the data

```
names(spam)
```

```
##  [1] "A.1"  "A.2"  "A.3"  "A.4"  "A.5"  "A.6"  "A.7"  "A.8"  "A.9"  "A.10"
## [11] "A.11" "A.12" "A.13" "A.14" "A.15" "A.16" "A.17" "A.18" "A.19" "A.20"
## [21] "A.21" "A.22" "A.23" "A.24" "A.25" "A.26" "A.27" "A.28" "A.29" "A.30"
## [31] "A.31" "A.32" "A.33" "A.34" "A.35" "A.36" "A.37" "A.38" "A.39" "A.40"
## [41] "A.41" "A.42" "A.43" "A.44" "A.45" "A.46" "A.47" "A.48" "A.49" "A.50"
## [51] "A.51" "A.52" "A.53" "A.54" "A.55" "A.56" "A.57" "spam"
```

```
head(spam,5)
```

```
##    A.1  A.2  A.3 A.4  A.5  A.6  A.7  A.8  A.9 A.10 A.11 A.12 A.13 A.14 A.15
## 1 0.00 0.64 0.64   0 0.32 0.00 0.00 0.00 0.00 0.00 0.00 0.64 0.00 0.00 0.00
## 2 0.21 0.28 0.50   0 0.14 0.28 0.21 0.07 0.00 0.94 0.21 0.79 0.65 0.21 0.14
## 3 0.06 0.00 0.71   0 1.23 0.19 0.19 0.12 0.64 0.25 0.38 0.45 0.12 0.00 1.75
## 4 0.00 0.00 0.00   0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00
## 5 0.00 0.00 0.00   0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00
##   A.16 A.17 A.18 A.19 A.20 A.21 A.22 A.23 A.24 A.25 A.26 A.27 A.28 A.29 A.30
## 1 0.32 0.00 1.29 1.93 0.00 0.96    0 0.00 0.00    0    0    0    0    0    0
## 2 0.14 0.07 0.28 3.47 0.00 1.59    0 0.43 0.43    0    0    0    0    0    0
## 3 0.06 0.06 1.03 1.36 0.32 0.51    0 1.16 0.06    0    0    0    0    0    0
## 4 0.31 0.00 0.00 3.18 0.00 0.31    0 0.00 0.00    0    0    0    0    0    0
## 5 0.31 0.00 0.00 3.18 0.00 0.31    0 0.00 0.00    0    0    0    0    0    0
##   A.31 A.32 A.33 A.34 A.35 A.36 A.37 A.38 A.39 A.40 A.41 A.42 A.43 A.44 A.45
## 1    0    0    0    0    0    0 0.00    0    0 0.00    0    0 0.00    0 0.00
## 2    0    0    0    0    0    0 0.07    0    0 0.00    0    0 0.00    0 0.00
## 3    0    0    0    0    0    0 0.00    0    0 0.06    0    0 0.12    0 0.06
## 4    0    0    0    0    0    0 0.00    0    0 0.00    0    0 0.00    0 0.00
## 5    0    0    0    0    0    0 0.00    0    0 0.00    0    0 0.00    0 0.00
##   A.46 A.47 A.48 A.49  A.50 A.51  A.52  A.53  A.54  A.55 A.56 A.57 spam
## 1 0.00    0    0 0.00 0.000    0 0.778 0.000 0.000 3.756   61  278 spam
## 2 0.00    0    0 0.00 0.132    0 0.372 0.180 0.048 5.114  101 1028 spam
## 3 0.06    0    0 0.01 0.143    0 0.276 0.184 0.010 9.821  485 2259 spam
## 4 0.00    0    0 0.00 0.137    0 0.137 0.000 0.000 3.537   40  191 spam
## 5 0.00    0    0 0.00 0.135    0 0.135 0.000 0.000 3.537   40  191 spam
```

Splitting the data into training and the test data

```
set.seed(23)
random_index = sample(c(1:nrow(spam)), size = round(8/10 * nrow(spam)), replace = FALSE)
train_data <- spam[random_index,]
test_data <- spam[-random_index,]
#train_data = data.frame(train_data)
#test_data = data.frame(test_data)
data1 = data.frame(spam)

y_train_data <- as.numeric(train_data$spam)-1
y_test_data <- as.numeric(test_data$spam)-1
dim(train_data)
```

```
## [1] 3681   58
```

```
dim(test_data)
```

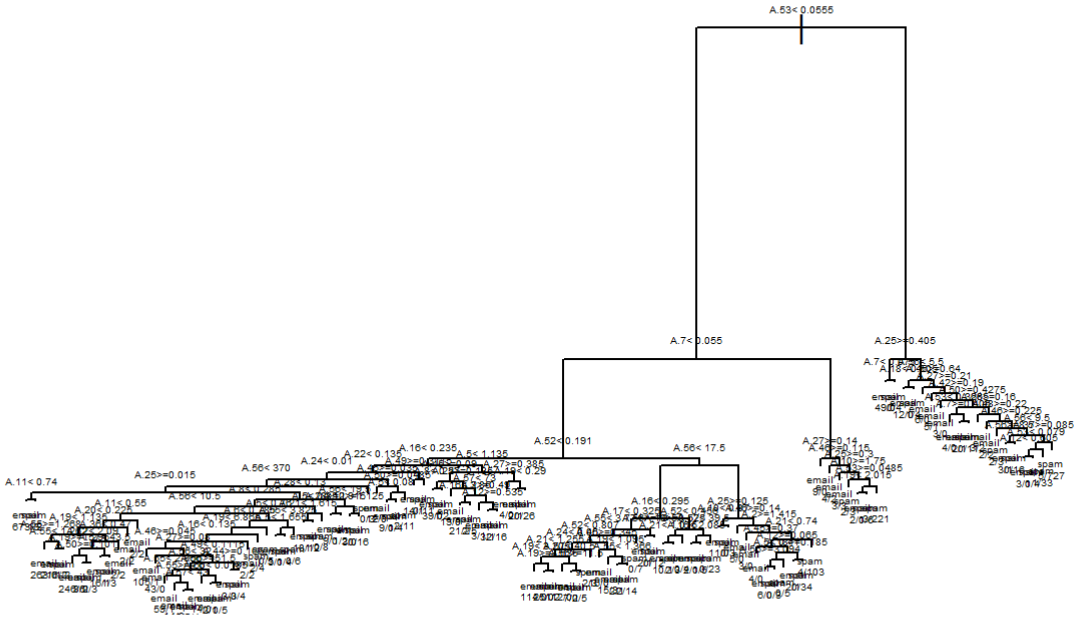
```
## [1] 920  58
```

Getting a Single Tree

```
model_control = rpart.control(minsplit = 8, xval = 10, cp = 0)
model_fit = rpart(spam~., data = train_data, method = "class", control = model_control)
```

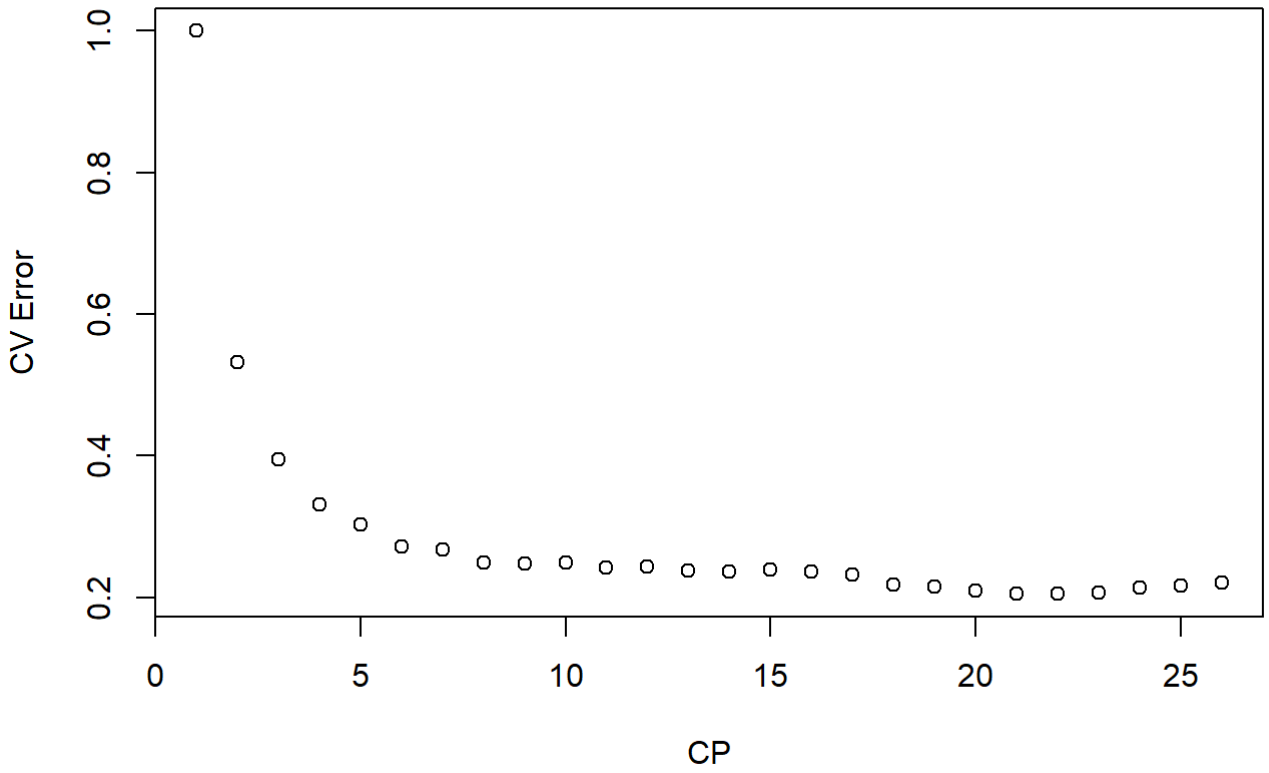
Plotting the tree

```
plot(model_fit)
text(model_fit, use.n = TRUE, cex = 0.3)
```



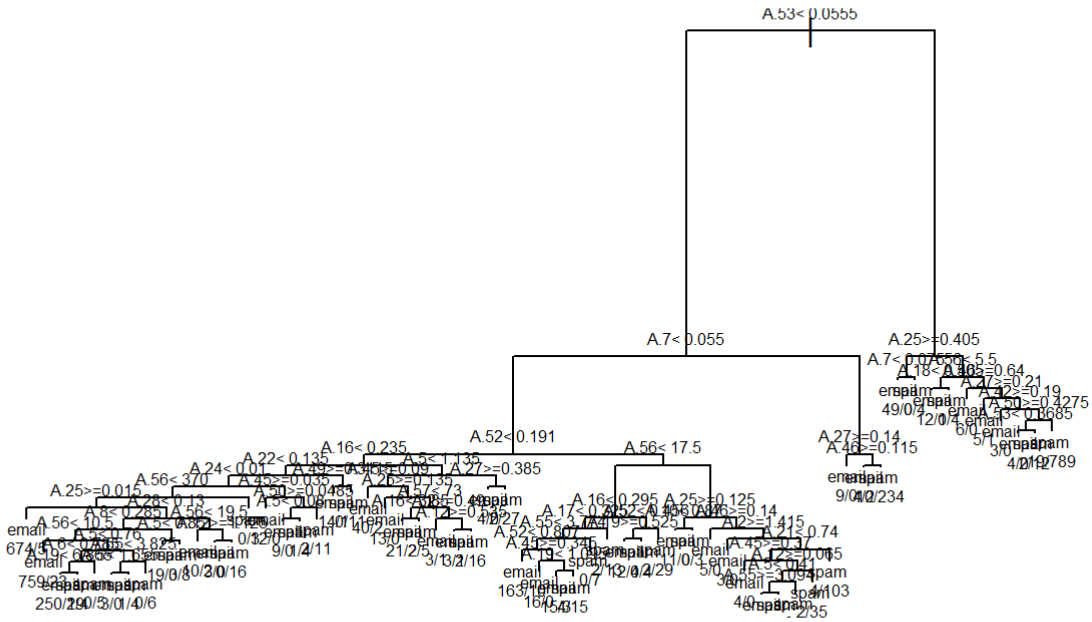
Pruning the tree

```
minimum_cp = which.min(model_fit$cptable[,4])
plot(model_fit$cptable[,4], xlab = "CP", ylab = "CV Error" )
```



Prune Fit

```
pruned_model_fit = prune(model_fit, cp = model_fit$cptable[minimum_cp, 1])
plot(pruned_model_fit)
text(pruned_model_fit, use.n = TRUE, cex = .5)
```



Predicting for new values - For Test Data

```
model_predict = predict(pruned_model_fit, newdata = test_data, type = "class")
y_predict = as.numeric(model_predict)-1
```

Calculating Error

```
prediction_error = sum(abs(y_predict - y_test_data))/length(y_test_data)
print(prediction_error)
```

```
## [1] 0.07826087
```

Modelling Random Forest for the spam data

For Random Forest the Y column should be a factor. So Converting the Spam column to factor again

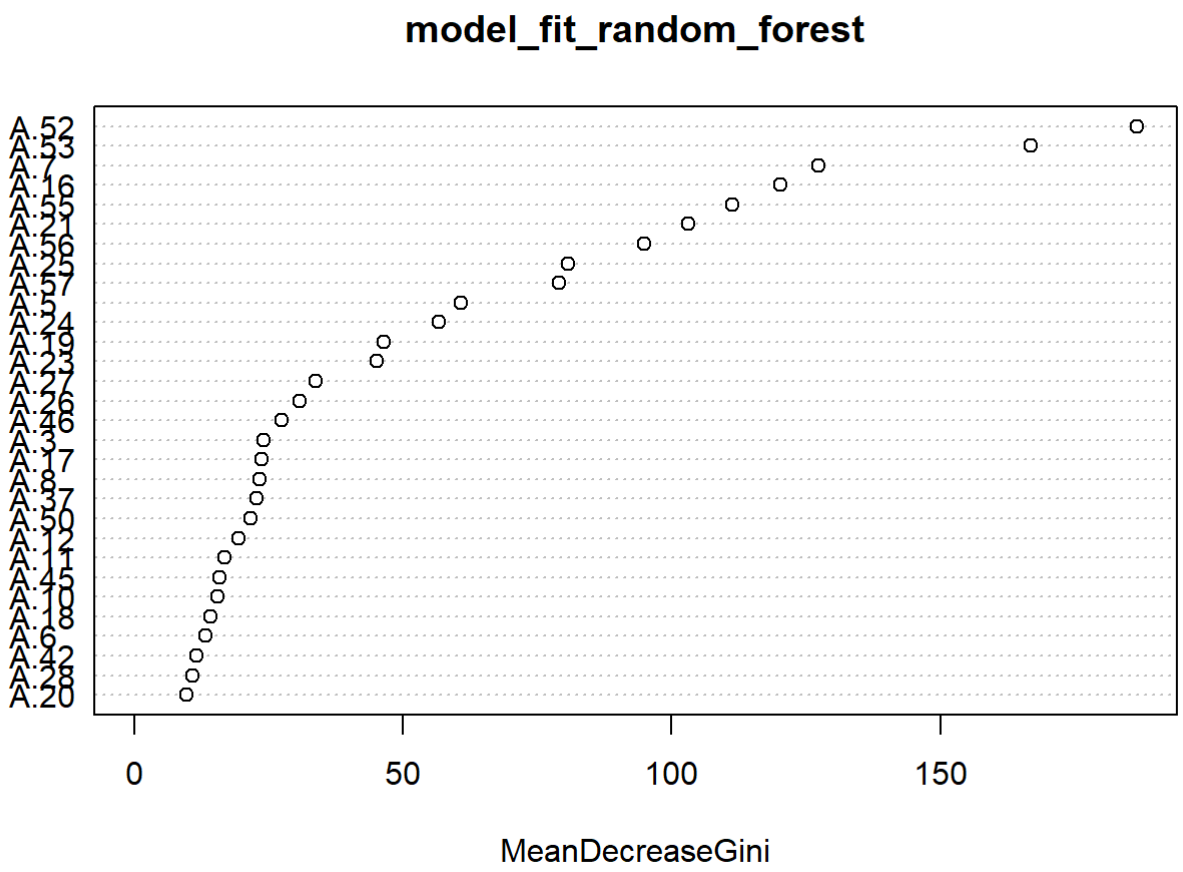
```
train_data$spam = as.factor(train_data$spam)
model_fit_random_forest = randomForest(spam~., data = train_data, n.tree = 1000)
```

Plotting the model parameters

```
names(model_fit_random_forest)
```

```
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
```

```
varImpPlot(model_fit_random_forest)
```




```
importance(model_fit_random_forest)
```

```
##      MeanDecreaseGini
## A.1         6.5487395
## A.2         9.0241241
## A.3        24.1054233
## A.4         1.1787321
## A.5        60.7352648
## A.6        13.1365500
## A.7       127.3302798
## A.8        23.2624302
## A.9         7.7719906
## A.10       15.4601590
## A.11       16.8192330
## A.12       19.4215626
## A.13        6.8171441
## A.14        4.3135499
## A.15        2.5732475
## A.16      120.1415323
## A.17       23.6848305
## A.18       14.1989147
## A.19       46.3569349
## A.20        9.5869179
## A.21      103.1532783
## A.22        4.7218879
## A.23       45.1082409
## A.24       56.7377711
## A.25       80.7133567
## A.26       30.6533304
## A.27       33.7291262
## A.28       10.7171339
## A.29        2.7582915
## A.30        9.0960436
## A.31        3.0878247
## A.32        1.1864956
## A.33        4.8243275
## A.34        0.9009200
## A.35        6.9420750
## A.36        5.8594604
## A.37       22.7183334
## A.38        0.9569010
## A.39        5.6530567
## A.40        2.1822098
## A.41        1.9779749
## A.42       11.5256830
## A.43        2.4151668
## A.44        3.7361652
## A.45       15.8521451
## A.46       27.4333288
## A.47        0.4353338
## A.48        2.1191562
## A.49        9.0611884
## A.50       21.6685519
## A.51        3.3328728
## A.52      186.5767867
## A.53      166.9179877
## A.54        6.4210520
```

```
## A.55      111.2556131
## A.56       94.8801490
## A.57       79.0318883
```

Predicting for new set of values - Test data

```
y_predict = predict(model_fit_random_forest, newdata = test_data, type = "response")
y_predict = as.numeric(y_predict)-1
```

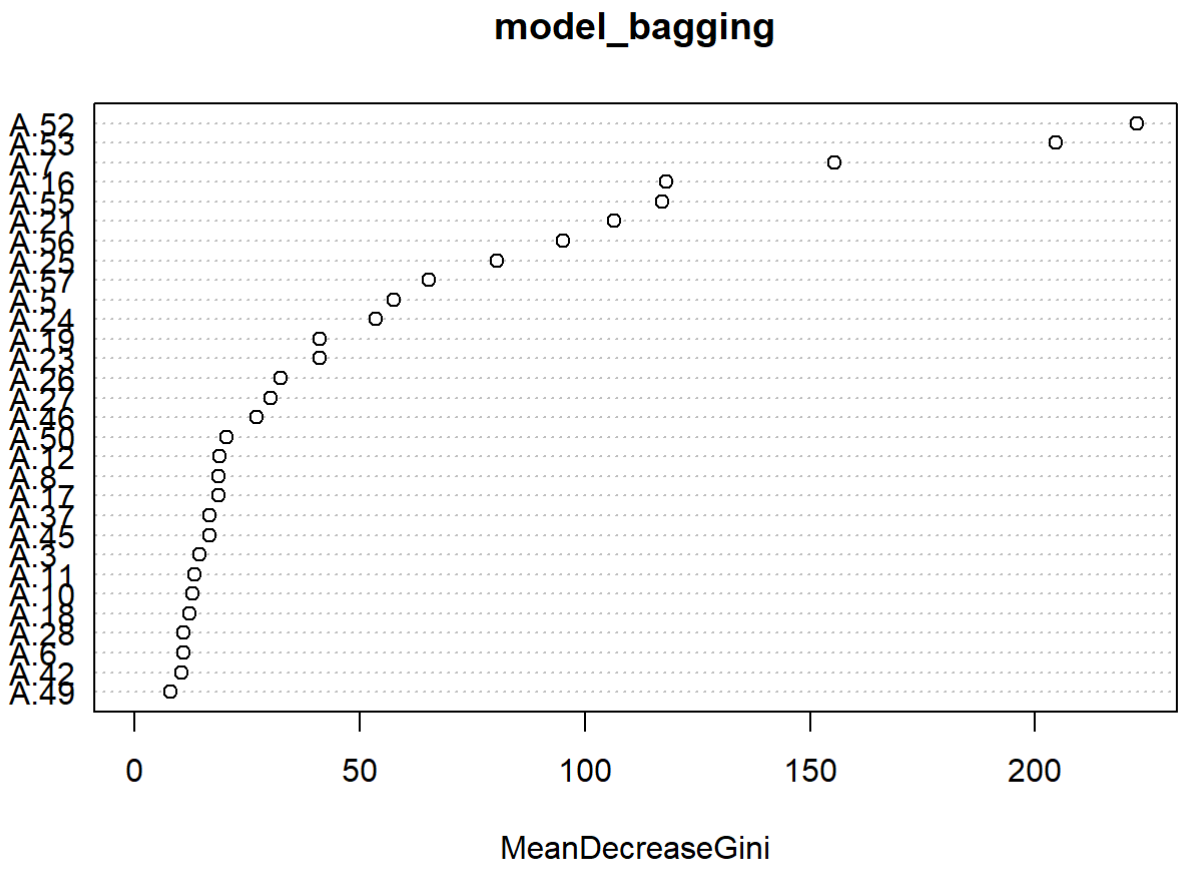
Calculating Prediction error for Random Forest

```
prediction_error_random_forest = sum(abs(y_predict - y_test_data))/length(y_test_data)
print(prediction_error_random_forest)
```

```
## [1] 0.03695652
```

Using Bagging for m = 10

```
model_bagging <- randomForest(spam~., data = train_data, n.tree = 1000, mtry = 10)

varImpPlot(model_bagging)
```




```
importance(model_bagging)
```

```
##      MeanDecreaseGini
## A.1         5.1433838
## A.2         7.1254429
## A.3        14.4084645
## A.4         1.0631271
## A.5        57.6578304
## A.6        10.8608414
## A.7       155.4370466
## A.8        18.6697212
## A.9         5.2868437
## A.10       12.8498462
## A.11       13.2526479
## A.12       18.8231590
## A.13        5.5513838
## A.14        3.9195981
## A.15        2.0245258
## A.16      118.0920700
## A.17       18.5835462
## A.18       12.2399664
## A.19       41.1402680
## A.20        6.8120483
## A.21      106.4925108
## A.22        4.4535938
## A.23       41.0999107
## A.24       53.5900146
## A.25       80.4501066
## A.26       32.3963206
## A.27       30.1208852
## A.28       10.8631327
## A.29        2.2225182
## A.30        6.6128534
## A.31        2.4879604
## A.32        0.6411734
## A.33        4.0277990
## A.34        0.8066795
## A.35        5.0396985
## A.36        5.7210148
## A.37       16.6696038
## A.38        0.7423280
## A.39        5.3152742
## A.40        1.7942311
## A.41        1.6468307
## A.42       10.4995402
## A.43        2.1614815
## A.44        3.1315939
## A.45       16.5758651
## A.46       27.1492362
## A.47        0.3728222
## A.48        2.0675302
## A.49        7.9770635
## A.50       20.4362803
## A.51        2.7365561
## A.52      222.5439157
## A.53      204.4963582
## A.54        5.5070102
```

```
## A.55      117.2227086
## A.56       95.2553231
## A.57       65.4479318
```

Predicting with bagging model

```
y_predict = predict(model_bagging, newdata = test_data, type = "response")
y_predict = as.numeric(y_predict)-1
```

Prediction Error for Bagging model

```
prediction_error_bagging = sum(abs(y_predict - y_test_data))/length(y_test_data)
print(prediction_error_bagging)
```

```
## [1] 0.03478261
```

Modelling Random Forest for different values of m

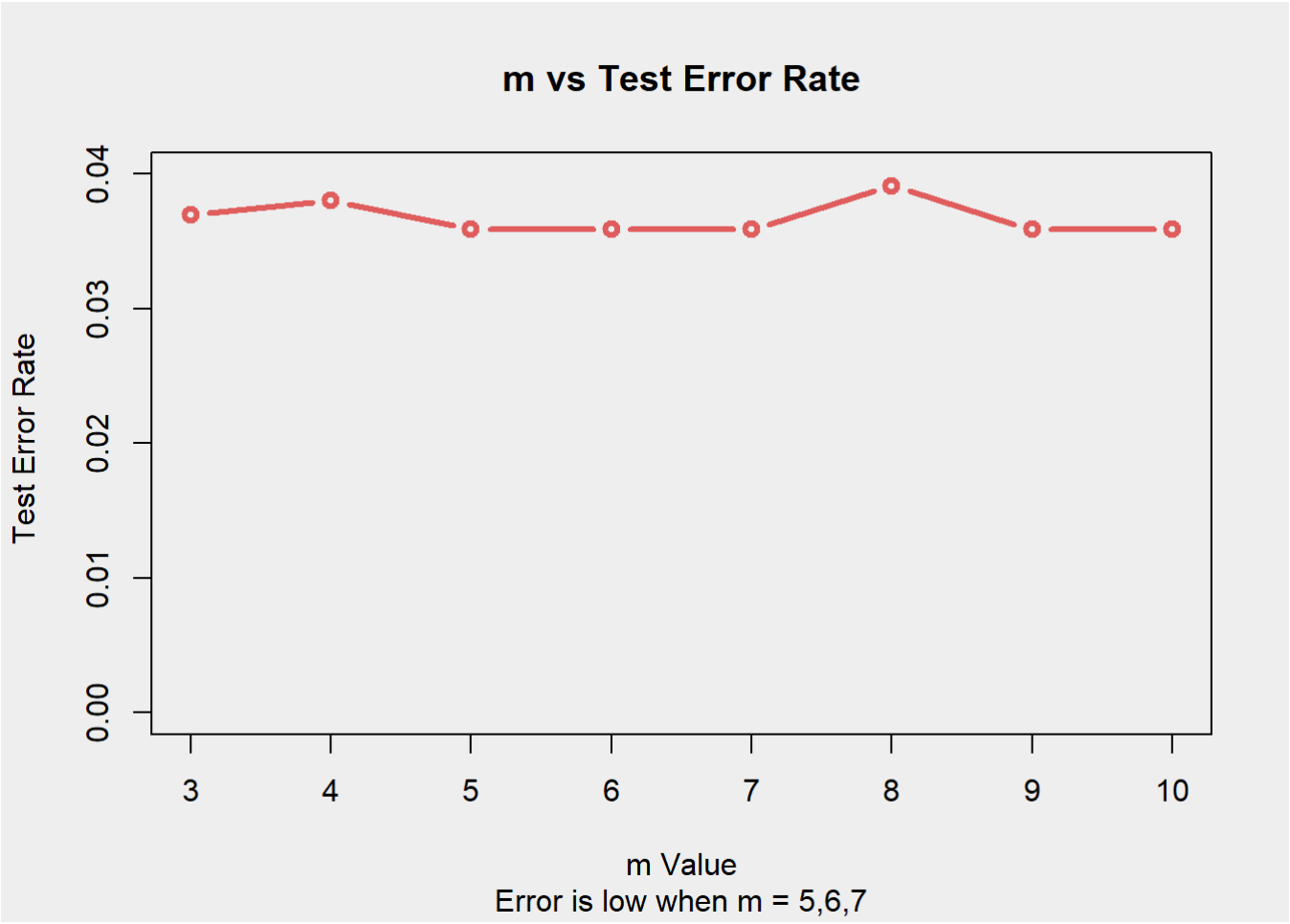
m = 3,4,5,6,7,8,9,10

```
m = c(3,4,5,6,7,8,9,10)
error_rate = c()
for (i in 1:8){
model_bag_rf <- randomForest(spam~., data = train_data, n.tree = 350, mtry = m[i])

y_predict = predict(model_bag_rf, newdata = test_data, type = "response")
y_predict = as.numeric(y_predict)-1

predict_err_rf = sum(abs(y_predict - y_test_data))/length(y_test_data)
error_rate[i] = predict_err_rf
if(m[i]==3){
  m3 = model_bag_rf$err.rate[,1]
}
if(m[i]==5){
  m5 = model_bag_rf$err.rate[,1]
}
if(m[i]==7){
  m7 = model_bag_rf$err.rate[,1]
}
if(m[i]==9){
  m9 = model_bag_rf$err.rate[,1]
}
}
print(error_rate)
```

```
## [1] 0.03695652 0.03804348 0.03586957 0.03586957 0.03586957 0.03913043 0.03586957
## [8] 0.03586957
```

Plotting m and the test error value

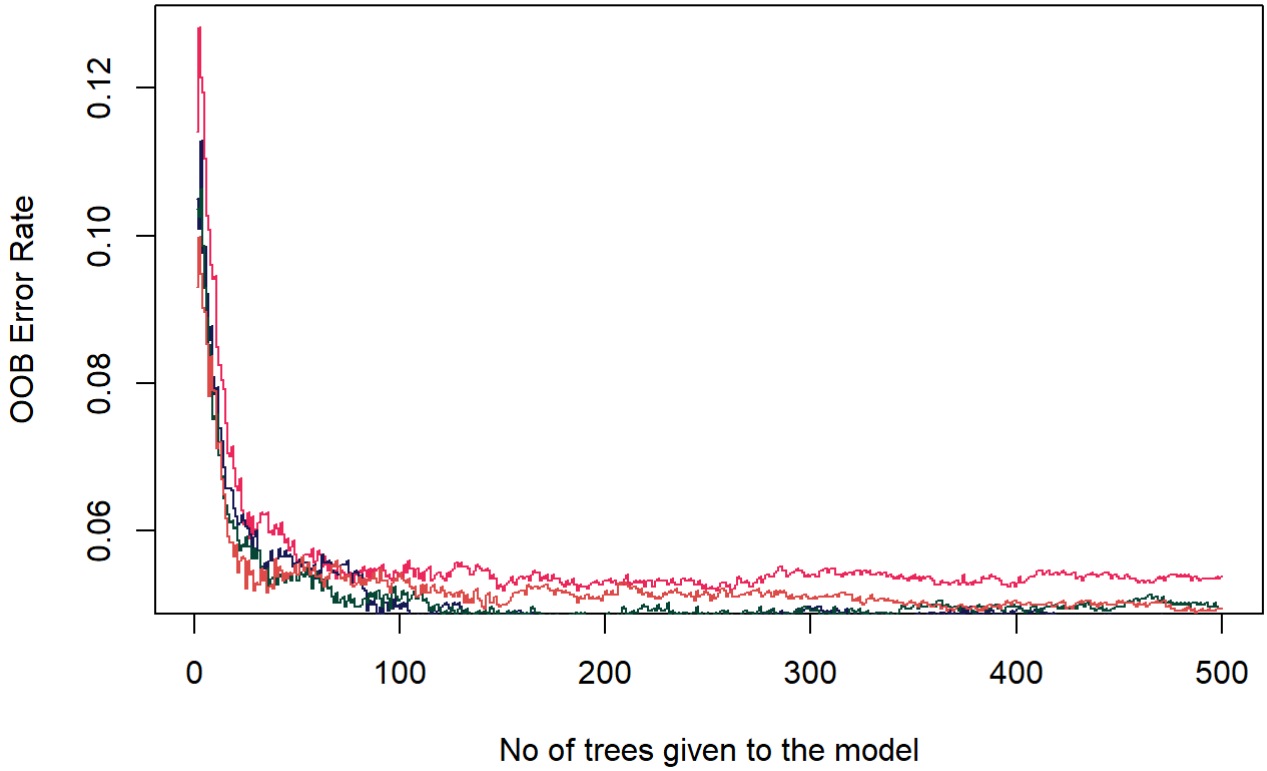
```
par(bg = '#EEEEEE')
plot(m,error_rate, col="#E05D5D", type = "b", xlab = "m Value", ylab = " Test Error Rate", yl
im = c(0,0.04), main = "m vs Test Error Rate", sub="Error is low when m = 5,6,7", lwd = 3.0)
```



Plotting OOB Error rate and m

Plotting for m values 3,5,7,9

```
x_axis = c(1:length(m3))
plot(x_axis, m3, col = "#EC255A", type = "s", xlab = "No of trees given to the model", ylab
= "OOB Error Rate")
lines(x_axis, m5, col = "#161853", type = "s")
lines(x_axis, m7, col = "#064635", type = "s")
lines(x_axis, m9, col = "#DD4A48", type = "s")
```



The Error is low when the value of m = 5,6,7