Problem (R question). In this question, you will create a decision tree, a Naive Bayes classifier, a random forest and a boosting model for detecting e-mail spam on a publicly available spam dataset. You will test your classifier using 10-fold cross-validation. Download the Spambase dataset available from the UCI Machine Learning Repository. The Spambase data set consists of 4,601 e-mails, of which 1,813 are spam (39.4%). The data set archive contains a processed version of the e-mails wherein 57 real-valued features have been extracted and the spam/non-spam label has been assigned. You should work with this processed version of the data. The data set archive contains a description of the features extracted as well as some simple statistics over those features.

Solution

We imported the spambase dataset set and renamed the variables. Following are the details of the same:
It has 4601 observations and 58 variables.
Our Target variable is spam.

#Good-Bad Cases

```
> count(spambase, 'spam')
  spam freq
1    0 2788
2    1 1813
>
```

1813/4601 = 39.4% of the emails are actually spam

As we can see the error rate 39.4% of the emails were wrongly classified by the constant classifier.
We applied decision tree, a Naive Bayes classifier, a random forest and a boosting model for detecting e-mail spam

Results of Naïve Bayes

```
          Actual
Predicted    0    1
        0 1564   94
        1 1224 1719
```

Error= (1224+94)/4601= 0.2864

We now run k fold cross validation on Naïve Bayes Model.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1561   91
         1 1227 1722

               Accuracy : 0.7135
                 95% CI : (0.7002, 0.7266)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4594
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.5599
            Specificity : 0.9498
         Pos Pred Value : 0.9449
         Neg Pred Value : 0.5839
             Prevalence : 0.6060
         Detection Rate : 0.3393
   Detection Prevalence : 0.3591
      Balanced Accuracy : 0.7549

       'Positive' Class : 0
```

Error rate for Naïve Bayes CV : (1227+91)/4601 : 0.286459

Decision Tree without Pruning

```
> Test_Accuracy = predict(dt_spam_train, newdata = spamData, type="class")
> table(Test_Accuracy, spamData$spam, dnn = c("Predicted","Actual"))
         Actual
Predicted    0    1
        0 2786    1
        1    2 1812
```

Error rate: 3/4601 = 0.00065203

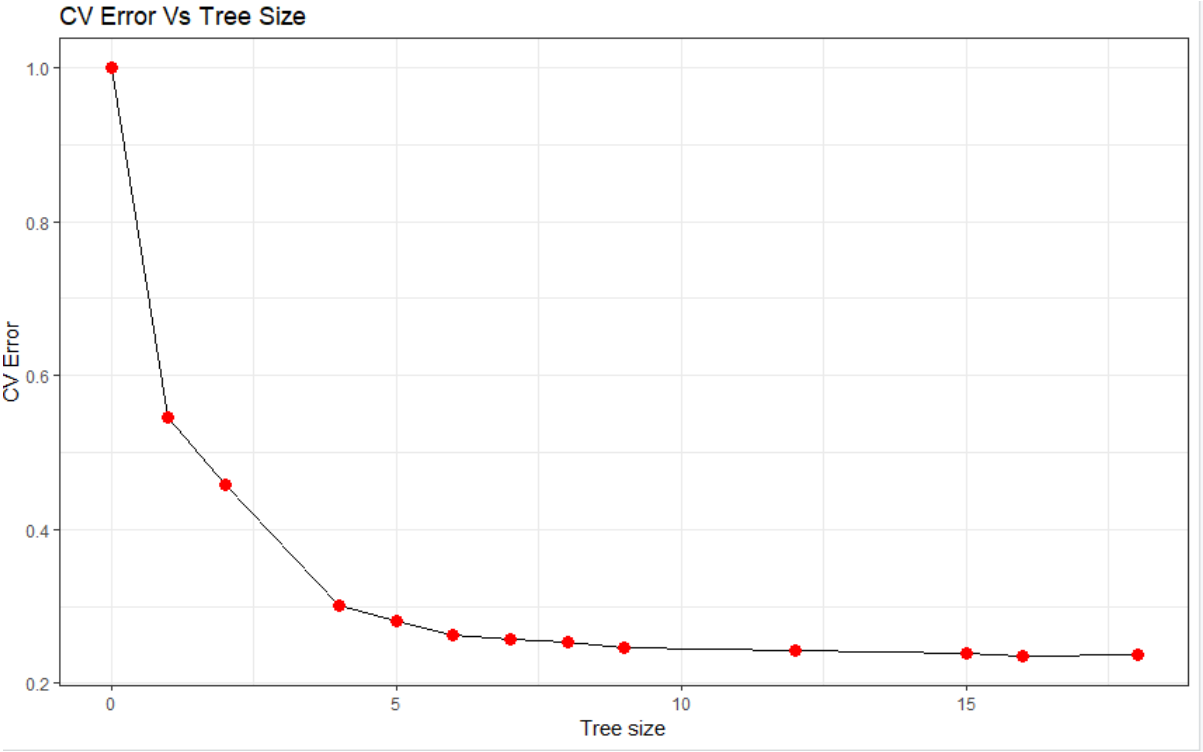
We clearly see a case of overfitting from above thus we now try pruning the tree to arrive at the best model possible.

Decision Tree Model after Pruning

```
> CV_spam_tree <- do.call(rbind, CV_spam_tree)
>
> confusionMatrix(CV_spam_tree$preds, CV_spam_tree$real)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2670  293
         1  118 1520

               Accuracy : 0.9107
                 95% CI : (0.9021, 0.9188)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8097
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9577
            Specificity : 0.8384
         Pos Pred Value : 0.9011
         Neg Pred Value : 0.9280
             Prevalence : 0.6060
         Detection Rate : 0.5803
   Detection Prevalence : 0.6440
      Balanced Accuracy : 0.8980

       'Positive' Class : 0
```

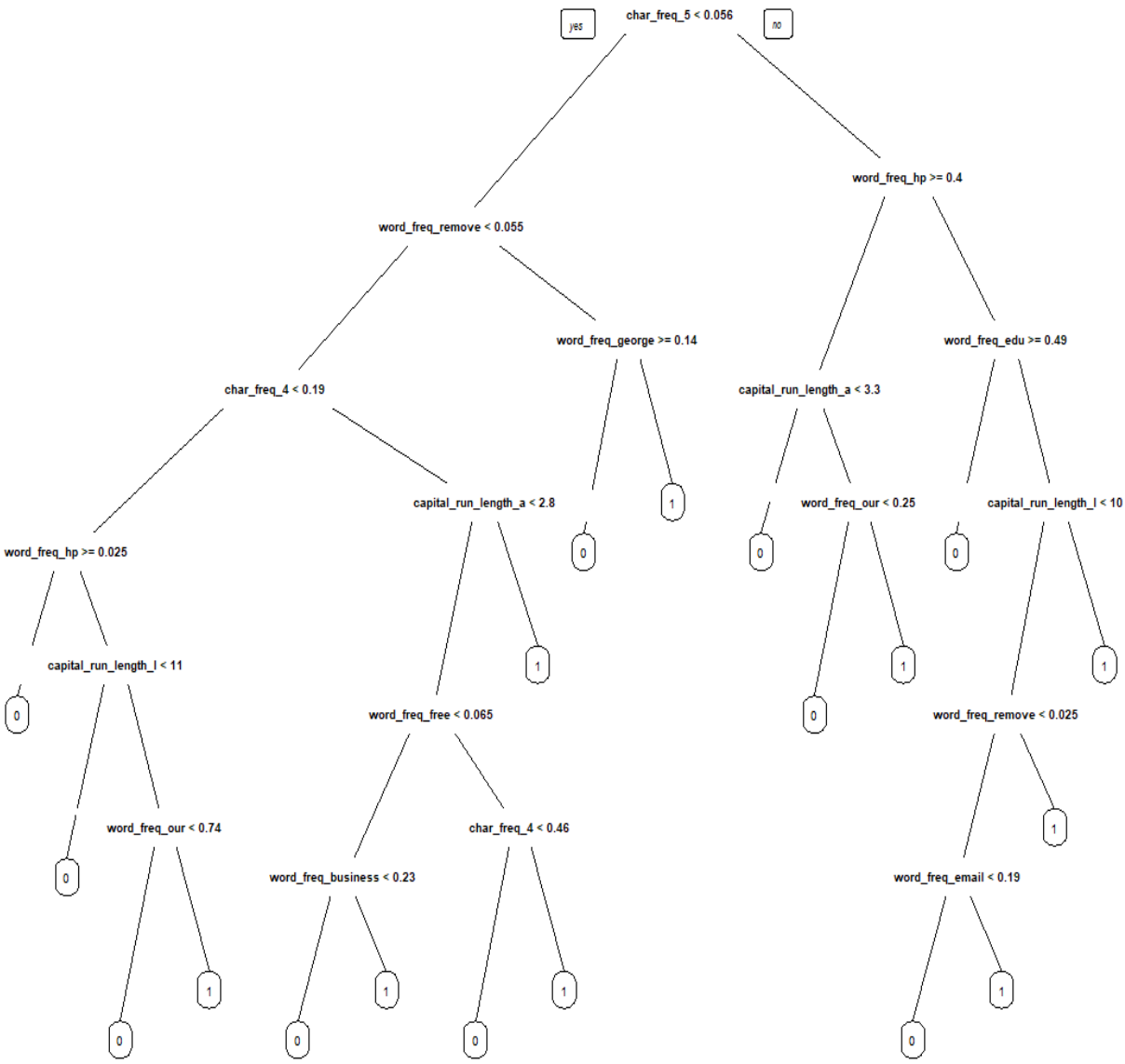
Testing Error : (118+293)/4601: 0.089328

Plot of Cv Error Vs Tree Size



As we can see from the above plot, the CV error stabilizes after around tree size of 6-7, thus we have finalized our best model with a max depth of 6 though there might be other fit but that might increase the tree size (maxdepth = 6)

Plot of the final model



Following are the variables that appear in the tree:

```
> print(dt_spam_train)
n= 4601

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 4601 1813 0 (0.605955227 0.394044773)
    2) char_freq_5< 0.0555 3471  816 0 (0.764909248 0.235090752)
      4) word_freq_remove< 0.055 3141  516 0 (0.835721108 0.164278892)
        8) char_freq_4< 0.191 2524  209 0 (0.917194929 0.082805071)
         16) word_freq_hp>=0.025 905    7 0 (0.992265193 0.007734807) *
         17) word_freq_hp< 0.025 1619  202 0 (0.875231624 0.124768376)
           34) capital_run_length_longest< 10.5 1047   54 0 (0.948424069 0.051575931) *
           35) capital_run_length_longest>=10.5 572  148 0 (0.741258741 0.258741259)
             70) word_freq_our< 0.74 523  108 0 (0.793499044 0.206500956) *
             71) word_freq_our>=0.74 49    9 1 (0.183673469 0.816326531) *
        9) char_freq_4>=0.191 617  307 0 (0.502431118 0.497568882)
         18) capital_run_length_average< 2.7655 371  108 0 (0.708894879 0.291105121)
           36) word_freq_free< 0.065 299   56 0 (0.812709030 0.187290970)
             72) word_freq_business< 0.225 277   38 0 (0.862815884 0.137184116) *
             73) word_freq_business>=0.225 22    4 1 (0.181818182 0.818181818) *
           37) word_freq_free>=0.065 72   20 1 (0.277777778 0.722222222)
             74) char_freq_4< 0.4605 30   13 0 (0.566666667 0.433333333) *
             75) char_freq_4>=0.4605 42    3 1 (0.071428571 0.928571429) *
         19) capital_run_length_average>=2.7655 246   47 1 (0.191056911 0.808943089) *
      5) word_freq_remove>=0.055 330   30 1 (0.090909091 0.909090909)
       10) word_freq_george>=0.14 13    0 0 (1.000000000 0.000000000) *
       11) word_freq_george< 0.14 317   17 1 (0.053627760 0.946372240) *
    3) char_freq_5>=0.0555 1130  133 1 (0.117699115 0.882300885)
      6) word_freq_hp>=0.4 70    7 0 (0.900000000 0.100000000)
       12) capital_run_length_average< 3.296 48    0 0 (1.000000000 0.000000000) *
       13) capital_run_length_average>=3.296 22    7 0 (0.681818182 0.318181818)
         26) word_freq_our< 0.245 12    0 0 (1.000000000 0.000000000) *
         27) word_freq_our>=0.245 10    3 1 (0.300000000 0.700000000) *
      7) word_freq_hp< 0.4 1060   70 1 (0.066037736 0.933962264)
       14) word_freq_edu>=0.49 15    0 0 (1.000000000 0.000000000) *
       15) word_freq_edu< 0.49 1045   55 1 (0.052631579 0.947368421)
         30) capital_run_length_longest< 9.5 46   21 1 (0.456521739 0.543478261)
           60) word_freq_remove< 0.025 33   12 0 (0.636363636 0.363636364)
            120) word_freq_email< 0.19 22    3 0 (0.863636364 0.136363636) *
            121) word_freq_email>=0.19 11    2 1 (0.181818182 0.818181818) *
           61) word_freq_remove>=0.025 13    0 1 (0.000000000 1.000000000) *
         31) capital_run_length_longest>=9.5 999   34 1 (0.034034034 0.965965966) *
>
```

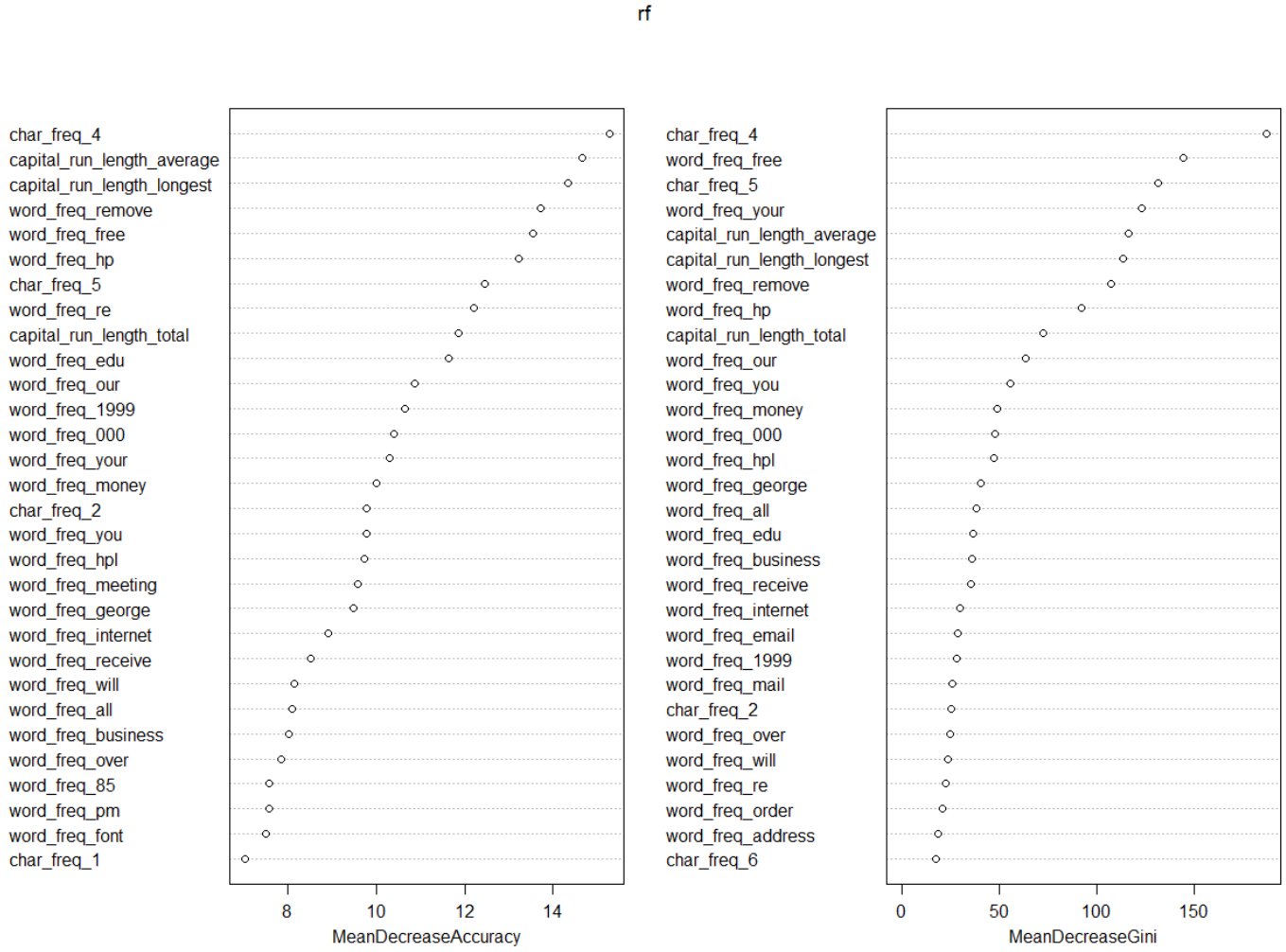
Random Forest Model

```
> rf

Call:
 randomForest(formula = spam ~ ., data = spamData, ntree = 100,      proximity = T, replace = T, importance = T, mtry = 3)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 3

        OOB estimate of  error rate: 5.19%
Confusion matrix:
     0    1 class.error
0 2709   79  0.02833572
1  160 1653  0.08825152
> |
```

```
> confusionMatrix(CV_spam_rf$preds, CV_spam_rf$real)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2715  156
         1   73 1657

               Accuracy : 0.9502
                 95% CI : (0.9435, 0.9563)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8949
 Mcnemar's Test P-Value : 6.003e-08

            Sensitivity : 0.9738
            Specificity : 0.9140
         Pos Pred Value : 0.9457
         Neg Pred Value : 0.9578
             Prevalence : 0.6060
         Detection Rate : 0.5901
   Detection Prevalence : 0.6240
      Balanced Accuracy : 0.9439

       'Positive' Class : 0
```

Error rate: (73+1657)/4601: 0.0497

Importance of variables

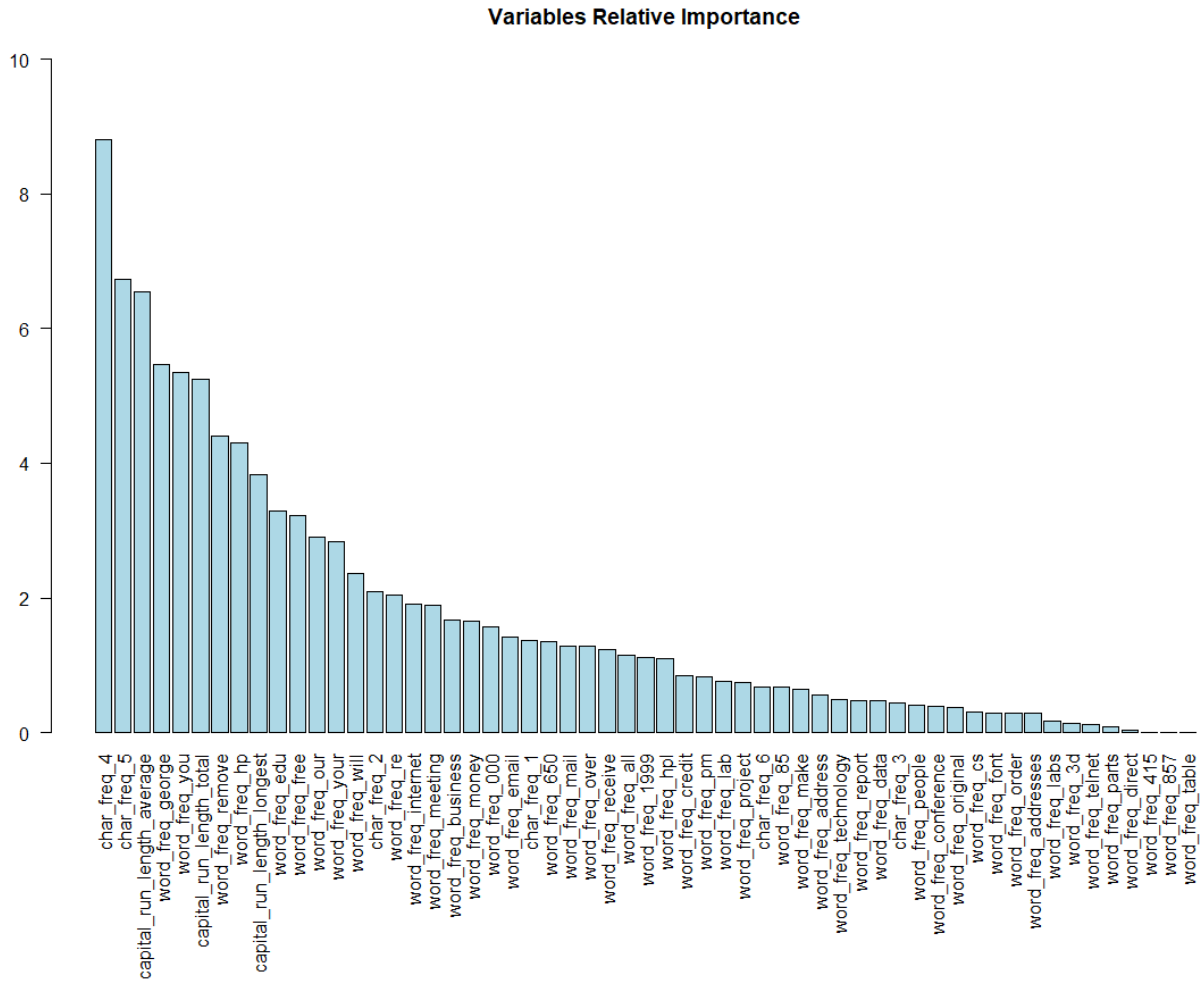


Ada Boost

```
$confusion
                Observed Class
Predicted Class    0    1
              0 2787    3
              1    1 1810

$error
[1] 0.0008693762
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2689   94
         1   99 1719

               Accuracy : 0.9581
                 95% CI : (0.9519, 0.9637)
    No Information Rate : 0.606
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9122
 Mcnemar's Test P-Value : 0.7734
```

Testing Error: (94+99)/4601: 0.0419



Error Table of various models across 10 folds for comparison

Folds	Naïve Bayes		
	False Positive	False Negative	Overall Error Rate
Fold1	0.368	0.041	0.2304
Fold2	0.517	0.046	0.3152
Fold3	0.471	0.043	0.3000
Fold4	0.442	0.047	0.2783
Fold5	0.446	0.046	0.2935
Fold6	0.441	0.076	0.3059
Fold7	0.441	0.059	0.3000
Fold8	0.429	0.072	0.2891
Fold9	0.423	0.040	0.2761
Fold10	0.425	0.034	0.2761
Average Error Rate across all folds	**0.29**		

Folds	Random Forest		
	False Positive	False Negative	Overall Error Rate
Fold1	0.011	0.108	0.0522
Fold2	0.042	0.081	0.0587
Fold3	0.014	0.092	0.0457
Fold4	0.026	0.089	0.0522
Fold5	0.018	0.086	0.0435
Fold6	0.025	0.137	0.0674
Fold7	0.035	0.063	0.0457
Fold8	0.029	0.072	0.0457
Fold9	0.028	0.100	0.0543
Fold10	0.034	0.029	0.0325
Average Error Rate across all folds	**0.05**		

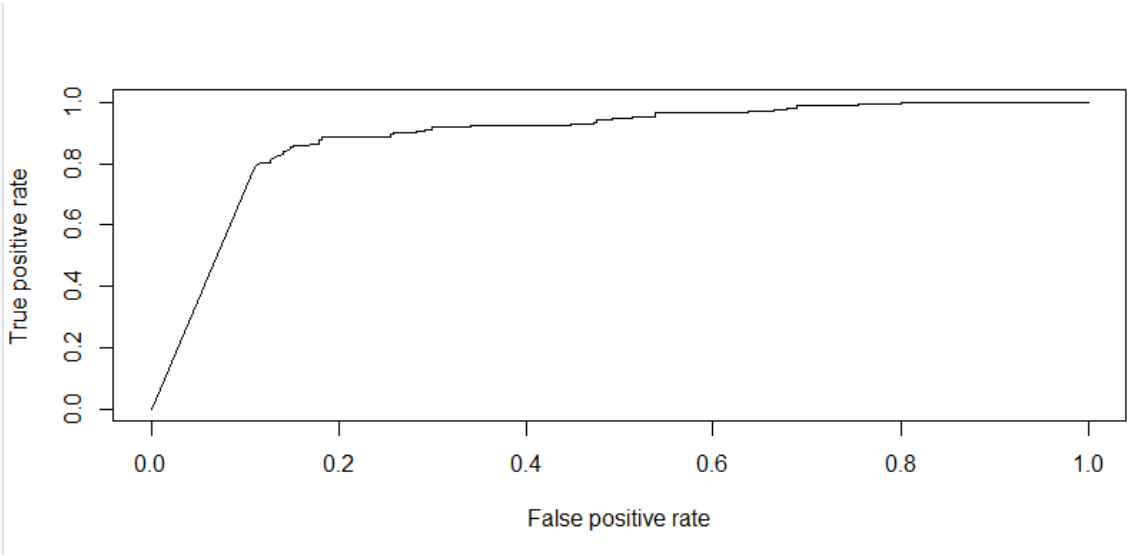
Folds	Decision Tree		
	False Positive	False Negative	Overall Error Rate
Fold1	0.049	0.170	0.100
Fold2	0.034	0.147	0.083
Fold3	0.043	0.196	0.104
Fold4	0.037	0.152	0.085
Fold5	0.021	0.217	0.096
Fold6	0.032	0.177	0.087
Fold7	0.077	0.125	0.096
Fold8	0.043	0.156	0.087
Fold9	0.034	0.141	0.074
Fold10	0.052	0.135	0.082
Average Error Rate across all folds	**0.089**		

Folds	Boosting		
	False Positive	False Negative	Overall Error Rate
Fold1	0.041	0.023	0.035
Fold2	0.021	0.053	0.033
Fold3	0.039	0.028	0.035
Fold4	0.035	0.040	0.037
Fold5	0.039	0.063	0.048
Fold6	0.046	0.080	0.059
Fold7	0.037	0.047	0.041
Fold8	0.018	0.071	0.039
Fold9	0.053	0.061	0.057
Fold10	0.026	0.052	0.037
Average Error Rate across all folds	**0.042**		

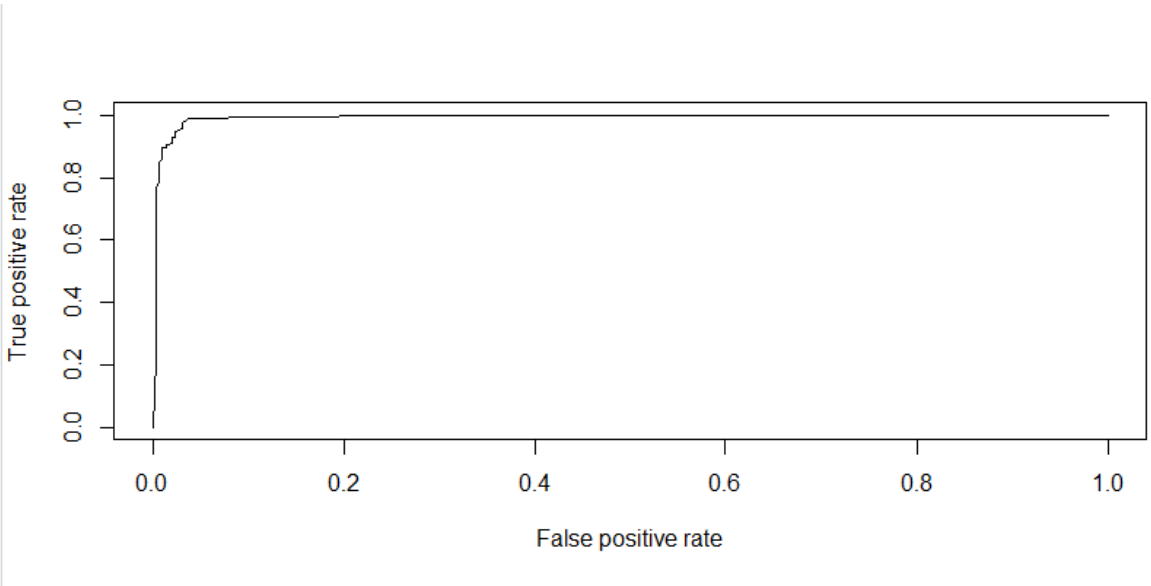
ROC Curve:

We have generated ROC curve for various models as follows for Fold1:

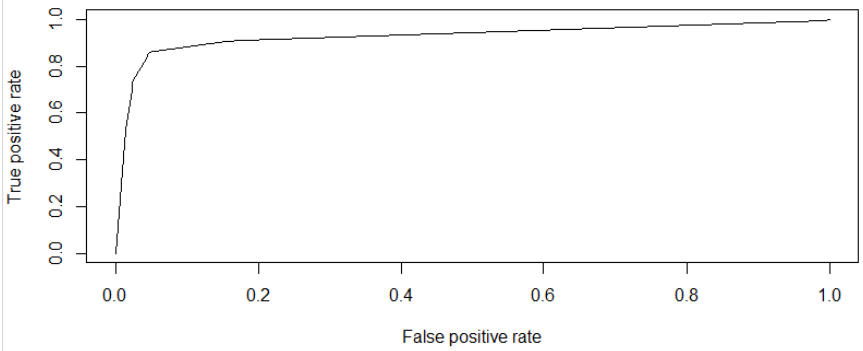
Naïve Bayes



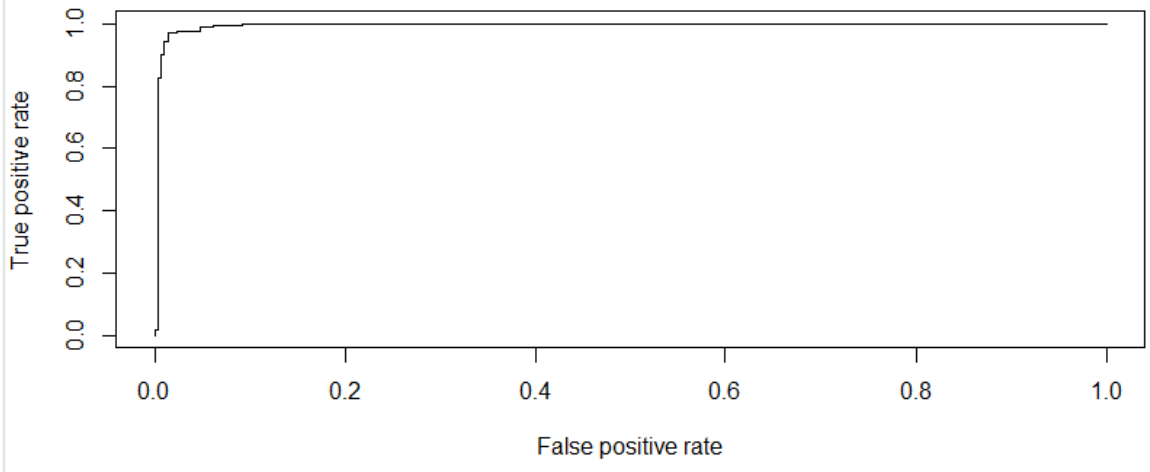
Random Forest



Decision Tree



Boosting



Following are the summary and findings of the performance of various models:

1. As it is evident from the error table as well as ROC, Naïve Bayes shows a trade off between False Positive and False Negative. In this example higher false positive is likely to be riskier as the chances that a legitimate e-mail is misclassified as spam can be a threat in missing out important information. Also the overall error rate is higher. Thus, Naïve Bayes is not a good model to predict the spam/non-spam classifiers.
2. Now out of the other models, we can see that Boosting performs the best with an average overall error of ~4% slightly lower than Random Forest with an error rate of 5%. However, when we look at the False Positive and False Negative rate, Random Forest comparatively performs better than Boosting with a slightly lower False positive rate across all the folds in our cross validation thus making it better for this prediction.
3. For this dataset, the decision tree though has a lower false positive rate across the folds but compared the Random Forest and Boosting, its performance is at a lower end with a higher overall testing error as well.

Thus, we can conclude that Random forest or Boosting would be a preferred model that helps in better prediction of spam and non-spam e-mails.