EMAIL SPAM DETECTION

SUDHARANI RANGAPALE

INTERNATIONAL SCHOOL OF ENGINEERING

NOVEMBER 10,2017

TABLE OF CONTENTS

Chapter 1

ABSTRACT

Machine learning is a branch of artificial intelligence concerned with the creation and study of systems that can learn from data. A machine learning system could be trained to distinguish between spam and non-spam (ham) emails. We aim to analyze current methods in machine learning to identify the best techniques to use in content-based spam filtering.

1. **Introduction**

   Email has become one of the most important forms of communication. In 2014, there are estimated to be 4.1 billion email accounts worldwide, and about 196 billion emails are sent each day worldwide. Spam is one of the major threats posed to email users. In 2013, 69.6% of all email flows were spam. Links in spam emails may lead to users to websites with malware or phishing schemes, which can access and disrupt the receiver's computer system. These sites can also gather sensitive information. Additionally, spam costs businesses around $2000 per employee per year due to decreased productivity. Therefore, an effective spam filtering technology is a significant contribution to the sustainability of the cyberspace and to our society.

   There are currently different approaches to spam detection. These approaches include blacklisting, detecting bulk emails, scanning message headings, grey listing, and content-based filtering

   • Blacklisting is a technique that identifies IP addresses that send large amounts of spam. These IP addresses are added to a Domain Name System-Based Black hole List and future email from IP addresses on the list are rejected. However, spammers are circumventing these lists by using larger numbers of IP addresses.

   • Detecting bulk emails is another way to filter spam. This method uses the number of recipients to determine if an email is spam or not. However, many legitimate emails can have high traffic volumes.

   • Scanning message headings is a fairly reliable way to detect spam. Programs written by spammers generate headings of emails. Sometimes, these headings have errors that cause them to not fit standard heading regulations. When these headings have errors, it is a sign that the email is probably spam. However, spammers are learning from their errors and making these mistakes less often.

   • Greylisting is a method that involves rejecting the email and sending an error message back to the sender. Spam programs will ignore this and not resend the email, while humans are more likely to resend the email. However, this process is annoying

to humans and is not an ideal solution. Current spam techniques could be paired with content-based spam filtering methods to increase effectiveness. Content-based methods analyze the content of the email to determine if the email is spam. The goal of our project was to analyze machine learning algorithms and determine their effectiveness as content-based spam filters.

DATA DESCRIPTION

The last column of 'spambase.data' denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail. Most of the attributes indicate whether a particular word or character was frequently occurring in the e-mail. The run-length attributes (55-57) measure the length of sequences of consecutive capital letters. For the statistical measures of each attribute, see the end of this file. Here are the definitions of the attributes:

48 continuous real [0,100] attributes of type word_freq_WORD
= percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.

6 continuous real [0,100] attributes of type char_freq_CHAR]
= percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail

1 continuous real [1,...] attribute of type capital_run_length_average
= average length of uninterrupted sequences of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_longest
= length of longest uninterrupted sequence of capital letters

1 continuous integer [1,...] attribute of type capital_run_length_total
= sum of length of uninterrupted sequences of capital letters
= total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam
= denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

Learning algorithms

- Logistic Regression
- Naïve Bayes
- Decision Tree
- SVM
- Random Forest

- # PLOTS

#Numbers of not spam vs. Numbers of Spam in DataSet

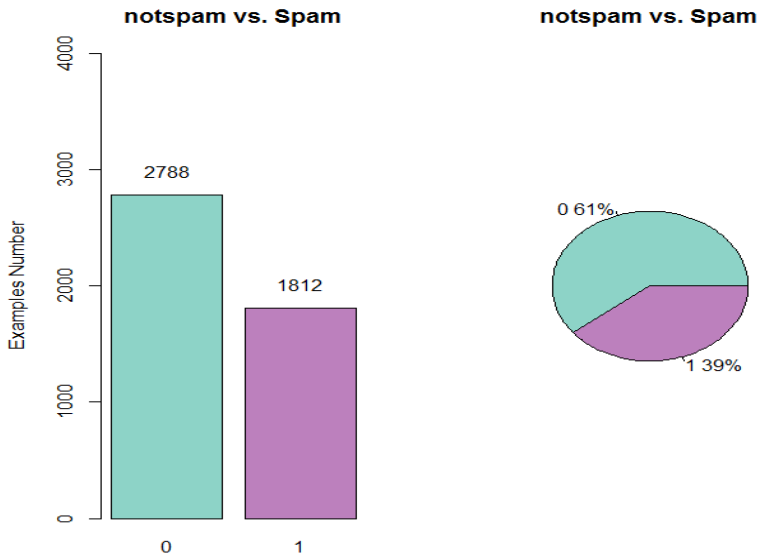


figure 1

nonspam vs. Spam in train data

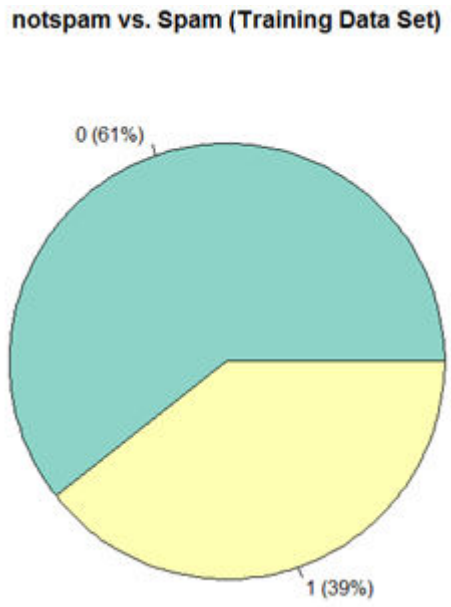


figure 2

Figure 1 and 2 explain Numbers of not spam vs. Numbers of Spam and non spam vs. Spam in train Dataset.

Chapter 2

Model Description

Machine Learning Basics

When we choose to approach spam filtering from a machine learning perspective, we view the problem as a classification problem. That is, we aim to classify an email as spam or not spam (ham) depending on its feature values. An example data point might be (x,y) where x is a d dimensional vector < x(1),x(2),….x(d)> describing the feature values and y has a value of 1 or 0, which denotes spam or not spam.

Machine learning systems can be trained to classify emails. As shown in Figure 1, a machine learning system operates in two modes: training and testing.

Training

During training, the machine learning system is given labelled data from a training data set. In our project, the labelled training data are a large set of emails that are labelled spam or non-spam. During the training process, the classifier (the part of the machine learning system that actually predicts labels of future emails) learns from the training data by determining the connections between the features of an email and its label.

Testing (Classification)

During testing, the machine learning system is given unlabeled data. In our case, these data are emails without the spam/non spam label. Depending on the features of an email, the classifier predicts whether the email is spam or non spam. This classification is compared to the true value of spam/non spam to measure performance.

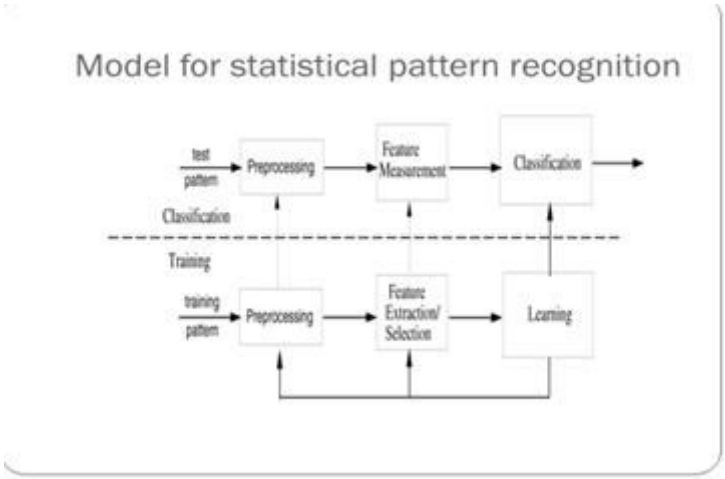


Figure 3 explains Basic Model for statistical pattern Recognition

Classifiers

As mentioned before, a machine learning system can be trained to classify emails as spam or non spam. To classify emails, the machine learning system must use some criteria to make its decision. The different algorithms that we describe below are different ways of deciding how to make the spam or non spam classification.

1.Logistic Regression

Logistic Regression is another way to determine a class label, depending on the features. Logistic regression takes features that can be continuous (for example, the count of words in an email) and translate them to discrete values (spam or non spam).

I. Model plot

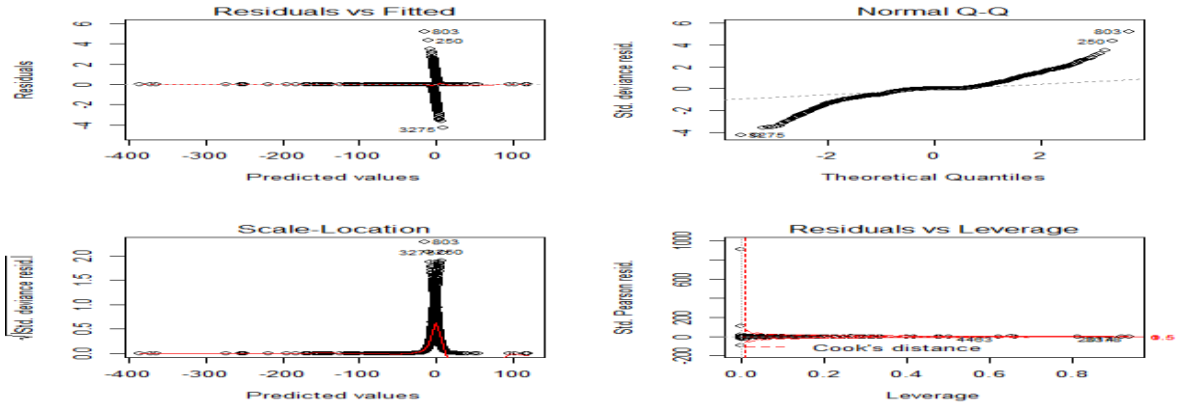


figure 4 explains the error plot of the model

II. ROC Curve

Here gives a sample of ROC curves in Figure 5. Where X-axis, called Specificity, is `False Positive rate = FP / [FP + TN]`; Y-axis, called Sensitivity, is `True Positive rate = TP / [TP + FN]`.

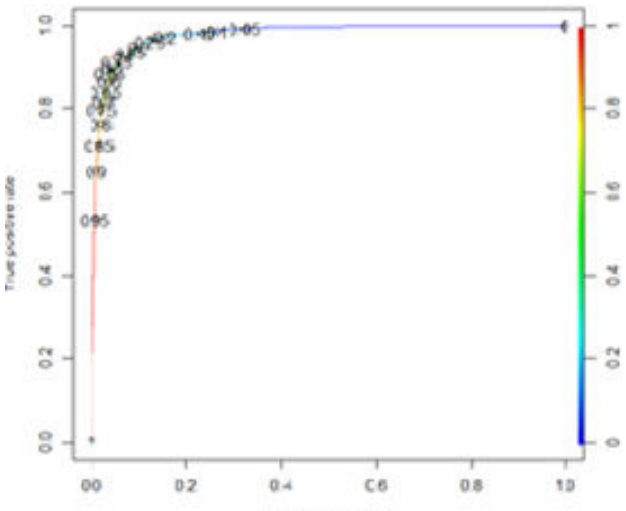


Figure 5: Sample of ROC curves

Output of Logistic Model

Output of Train model

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 522  35
         1  40 322

               Accuracy : 0.9184
                 95% CI : (0.8988, 0.93
    No Information Rate : 0.6115
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8287
 Mcnemar's Test P-Value : 0.6442

            Sensitivity : 0.9020
            Specificity : 0.9288
         Pos Pred Value : 0.8895
         Neg Pred Value : 0.9372
             Prevalence : 0.3885
         Detection Rate : 0.3504
   Detection Prevalence : 0.3939
      Balanced Accuracy : 0.9154

       'Positive' Class : 1
```

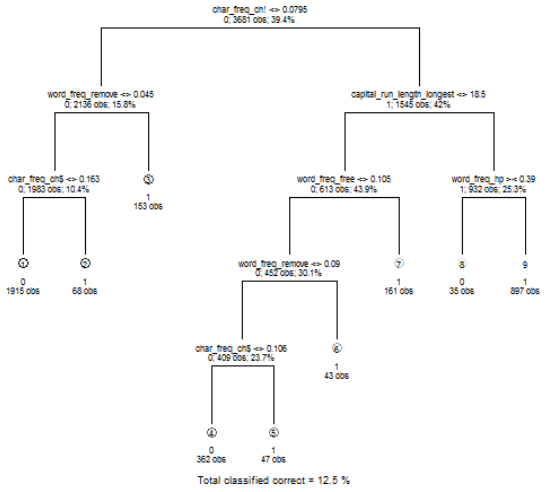
output of Test Model

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2139   92
         1  141 1309

               Accuracy : 0.9367
                 95% CI : (0.9283, 0.9444)
    No Information Rate : 0.6194
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8666
 Mcnemar's Test P-Value : 0.001663

            Sensitivity : 0.9343
            Specificity : 0.9382
         Pos Pred Value : 0.9028
         Neg Pred Value : 0.9588
             Prevalence : 0.3806
         Detection Rate : 0.3556
   Detection Prevalence : 0.3939
      Balanced Accuracy : 0.9362

       'Positive' Class : 1
```

This the output of model built using all the variables

Decision Tree

The decision tree algorithm aims to build a tree structure according to a set of rules from the training dataset, which can be used to classify unlabeled data.



Output of Decision Tree

Train model output

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2145  272
         1   86 1178

               Accuracy : 0.9027
                 95% CI : (0.8927, 0.9121
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7916
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8124
            Specificity : 0.9615
         Pos Pred Value : 0.9320
         Neg Pred Value : 0.8875
             Prevalence : 0.3939
         Detection Rate : 0.3200
   Detection Prevalence : 0.3434
      Balanced Accuracy : 0.8869

       'Positive' Class : 1
```

Test Model Output

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 531  27
         1  26 335

               Accuracy : 0.9423
                 95% CI : (0.9252, 0.9565)
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8792
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9254
            Specificity : 0.9533
         Pos Pred Value : 0.9280
         Neg Pred Value : 0.9516
             Prevalence : 0.3939
         Detection Rate : 0.3645
   Detection Prevalence : 0.3928
      Balanced Accuracy : 0.9394

       'Positive' Class : 1
```

This is the output on model built using all the variables

Naïve Bayesian

The legitimate e-mails a user receives will tend to be different. For example, in a corporate environment, the company name and the names of clients or customers will be mentioned often. The filter will assign a lower spam probability to emails containing those names.

The word probabilities are unique to each user and can evolve over time with corrective training whenever the filter incorrectly classifies an email. As a result, Bayesian spam filtering accuracy after training is often superior to pre-defined rules.

It can perform particularly well in avoiding false positives,where legitimate email is incorrectly classified as spam. For example, if the email contains the word "Nigeria", which is frequently used in Advance fee fraud spam, a pre-defined rules filter might reject it outright. A Bayesian filter would mark the word "Nigeria" as a probable spam word, but would take into account other important words that usually indicate legitimate e-mail. For example, the name of a spouse may strongly indicate the e-mail is not spam, which could overcome the use of the word "Nigeria."

NOTE: When Naïve Bayes model was built considering all the variables,the error metric(specificity) values was least compared to other models and below output is of model using all the variables.

Built model using only important variables (dollar,word_freq_remove,word_freq_000,word_freq_money,capital_run_length_longest,exclamatory,word_freq_credit,word_freq_receive,word_freq_hp,capital_run_length_average,capital_run_length_total,word_freq_free,word_freq_hpl,word_freq_telnet,word_freq_your,word_freq_our,word_freq_order,word_freq_650,word_freq_lab,word_freq_project,word_freq_3d, word_freq_business,word_freq_internet) and found specificity changed to 0.5567 to 0.9139 on trained data and 0.9102 from 0.5530 on test data.

Model plot

The Naive Bayesian classifier takes its roots in the famous Bayes Theorem:

P(H|e)=(P(e|H)P(H))/P(e).

P(s|w)=(P(w|s)*P(s))/(P(w|s)*P(s))+(P(w|nonspam).P(nonspam)

One of the main advantages of Bayesian spam filtering is that it can be trained on a per-user basis.

The spam that a user receives is often related to the online user's activities. For example, a user may have been subscribed to an online newsletter that the user considers to be spam. This online newsletter is likely to contain words that are common to all newsletters, such as the name of the newsletter and its originating email address. A Bayesian spam filter will eventually assign a higher probability based on the user's specific patterns.

Output of Naïve Bayes Model

Train model

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1261   68
         1  970 1382

               Accuracy : 0.718
                 95% CI : (0.7032, 0.7325)
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4674
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9531
            Specificity : 0.5652
         Pos Pred Value : 0.5876
         Neg Pred Value : 0.9488
             Prevalence : 0.3939
         Detection Rate : 0.3754
   Detection Prevalence : 0.6390
      Balanced Accuracy : 0.7592

       'Positive' Class : 1
```

Test Model

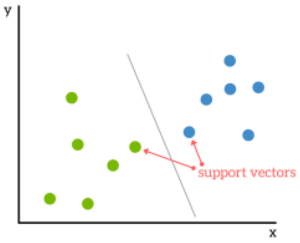
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 329  27
         1 228 335

               Accuracy : 0.7225
                 95% CI : (0.6923, 0.7513)
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : 9.59e-14

                  Kappa : 0.4704
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9254
            Specificity : 0.5907
         Pos Pred Value : 0.5950
         Neg Pred Value : 0.9242
             Prevalence : 0.3939
         Detection Rate : 0.3645
   Detection Prevalence : 0.6126
      Balanced Accuracy : 0.7580

       'Positive' Class : 1
```

SVM

A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be employed for both classification and regression purposes. SVMs are more commonly used in classification problems and as such, this is what we will focus on in this post.

SVMs are based on the idea of finding a hyper plane that best divides a dataset into two classes, as shown in the image below(figure 7).

Figure 7



Gamma

Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’. Higher the value of gamma, will try to exact fit the as per training data set i.e. generalization error and cause over-fitting problem.

In this section, support vector machine is applied to the dataset.

linear kernel gains better performance compared to other mappings.

Using the polynomial kernel and increasing the degree of the polynomial from two to three shows improvement in error rates.

However the error rate does not improve when the degree is increased further. Radial basis function (RBF) is another kernel applied here to the dataset.

One option we can explore in this case is reducing the number of features.

Output of SVM model

Train Model

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2138  219
         1   93 1231

               Accuracy : 0.9152
                 95% CI : (0.9058, 0.924)
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8197
 Mcnemar's Test P-Value : 1.476e-12

            Sensitivity : 0.8490
            Specificity : 0.9583
         Pos Pred Value : 0.9298
         Neg Pred Value : 0.9071
             Prevalence : 0.3939
         Detection Rate : 0.3344
   Detection Prevalence : 0.3597
      Balanced Accuracy : 0.9036

       'Positive' Class : 1
```

Test Model

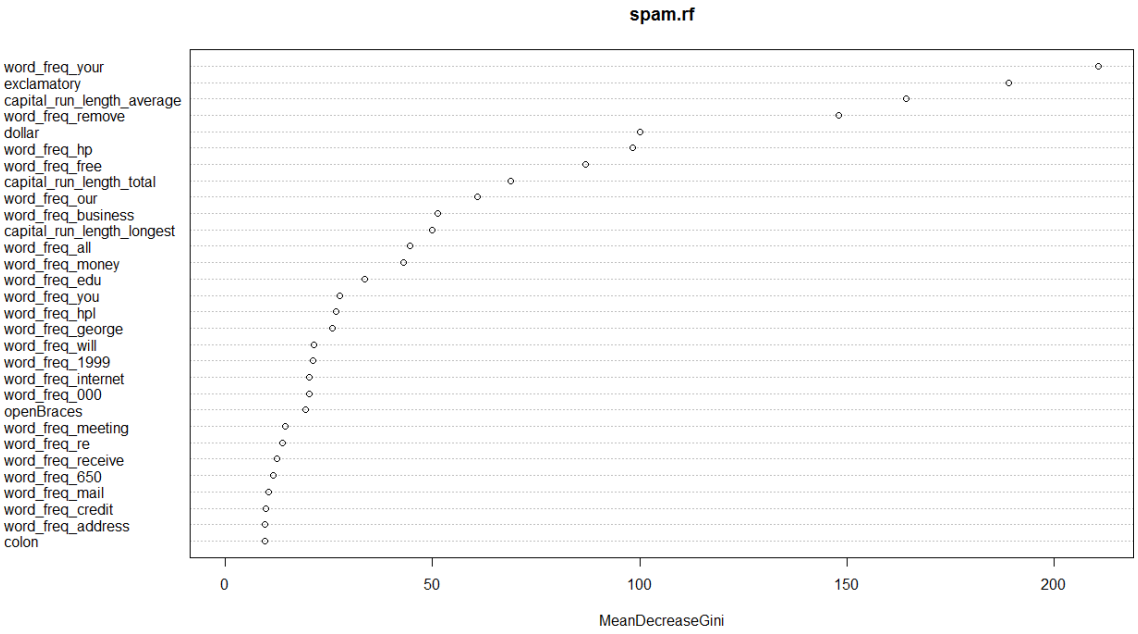
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 529  58
         1  28 304

               Accuracy : 0.9064
                 95% CI : (0.8857, 0.9245)
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8011
 Mcnemar's Test P-Value : 0.001765

            Sensitivity : 0.8398
            Specificity : 0.9497
         Pos Pred Value : 0.9157
         Neg Pred Value : 0.9012
             Prevalence : 0.3939
         Detection Rate : 0.3308
   Detection Prevalence : 0.3613
      Balanced Accuracy : 0.8948

       'Positive' Class : 1
```

Random Forest

The Random forest is a meta-learner which consists of many individual trees. Each tree votes on an overall classification for the given set of data and the random forest algorithm chooses the individual classification with the most votes. Each decision tree is built from a random subset of the training dataset, using what is called replacement, in performing this sampling. That is, some entities will be included more than once in the sample, and others won't appear at all. In building each decision tree, a model based on a different random subset of the training dataset and a random subset of the available variables is used to choose how best to partition the dataset at each node. Each decision tree is built to its maximum size, with no pruning performed. Together, the resulting decision tree models of the Random forest represent the final ensemble model where each decision tree votes for the result and the majority wins.

This will increase the bias of a single model, but the averaging reduces the variance and can compensate for increase in bias too. Consequently, a better model is built.

We observe that comparing to the naive Bayes algorithm, although the complexity of the model is increased, yet the performance does not show any improvement.



Output of Random forest

Train Model

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2222   23
         1    9 1427

               Accuracy : 0.9913
                 95% CI : (0.9877, 0.994)
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9818
 Mcnemar's Test P-Value : 0.02156

            Sensitivity : 0.9841
            Specificity : 0.9960
         Pos Pred Value : 0.9937
         Neg Pred Value : 0.9898
             Prevalence : 0.3939
         Detection Rate : 0.3877
   Detection Prevalence : 0.3901
      Balanced Accuracy : 0.9901

       'Positive' Class : 1
```

Test Model

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 531  27
         1  26 335

               Accuracy : 0.9423
                 95% CI : (0.9252, 0.9565)
    No Information Rate : 0.6061
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8792
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9254
            Specificity : 0.9533
         Pos Pred Value : 0.9280
         Neg Pred Value : 0.9516
             Prevalence : 0.3939
         Detection Rate : 0.3645
   Detection Prevalence : 0.3928
      Balanced Accuracy : 0.9394

       'Positive' Class : 1
```

Implementation

The results were based on a training sample of 800 emails and a testing sample of 200 emails. This means our classifiers learned from the data in the training set of 800 emails. Then, we determined the performance of the classifiers by observing how the 200 emails in the testing set were classified. All programming was implemented in R.

Results

We used the following performance metrics to evaluate our classifiers:

• Accuracy: % of predictions that were correct

• Recall: % of spam emails that were predicted correctly

• Precision: % of emails classified as spam that were actually spam

Conclusion

The sensitivity indicates the ability of the classifier to identify positive instances correctly, the specificity indicates the ability of the classifier to identify negative instances correctly and accuracy indicates the percentage of correct classification of both positive class as well as negative class instances. The Random Forest performs better than other classifiers with sensitivity, specificity and accuracy values 0.9478, 0.9088 and 0.9478 respectively.

Models		Accuracy	Specificity	sensitivity
logistic	Train	0.9277	0.9292	0.9253
	Test	0.9325	0.9350	0.9286
Svm	Train	0.9133	0.9516	0.8545
	Test	0.914	0.9569	0.8481
Decision tree	Train	0.8998	0.9355	0.8448
	Test	0.9042	0.8564	0.9354
Naïve Bayes	Train	0.7101	0.5567	0.9462
	Test	0.7116	0.5530	0.9558
Random forest	Train	0.991	0.9969	0.9821
	Test	0.9478	0.9088	0.9731