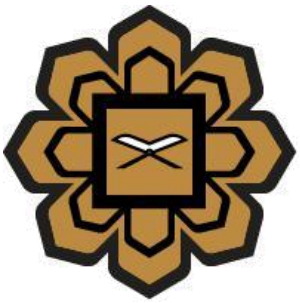

(Company No. 101067-P)

الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يونيورسيتي إسلام انتارابغسا ملسيا

Garden of Knowledge and Virtue

KULLIYYAH OF INFORMATION AND COMMUNICATION TECHNOLOGY

CSC 3304 MACHINE LEARNING SEMESTER 2, 2020/2021

SECTION 1

PROJECT TITLE:

Building Spam Filter Using Machine Learning Model in R

PREPARED BY:

NAME	MATRIC NO.
BILLAH SYED MASHKUR	1723387
HUDA MD NAJMUL	1627521
KM ZUBAIR	1722931

SUPERVISED BY:

ASSOC. PROF. DR. AMELIA RITAHANI BT ISMAIL

Table of Contents

1.0 Introduction

Spam, or unnecessary commercial mass communications, has been a major concern on the internet in recent years. The spammer is the one who sends out the spam messages. This person collects email addresses from a variety of sources, including blogs, chat rooms, and viruses. Spam prohibits users from getting the most out of their time, computing space, and network bandwidth. The main aim to develop this project is to filter the messages and tracking out the spam messages explores some of the most relevant spam filtering principles, attempts, performance, and research trends.

According to one survey, spam accounts for almost 70% of all business emails sent and received. The rapid increase of spam email has also been discovered to have implications such as overflowing user mailboxes, use of bandwidth and storage space, compromise of essential emails, and challenges for the user in terms of using their time to filter through all of the spam email. Because of the complexity brought by spammers, there is increasing interest in the topic of spam categorization at the moment. This is due to the fact that it is becoming increasingly difficult to discern between spam mails (unsolicited email) and ham email messages (legitimate emails).

Machine learning algorithms are employed to analyse all the authorized SMS or emails and report the suspicious ones. These reports are investigated by professionals who contact the cardholders to confirm if the transaction was genuine or fraudulent. The investigators provide feedback to the automated system which is used to train and update the algorithm to eventually improve the spam-detection performance over time.

Some of the currently used approaches to detection of such spams are:

- Deep Learning
- Naïve Bayes
- Support Vector Machines
- Neural Networks
- K-Nearest Neighbour
- Rough sets
- Random Forests

In this project, spam detection will be done using machine learning. However, for our project, **Random Forest** algorithm will be applied and we will also do a **comparison using XGBoost**. Dataset has been collected from Kaggle.

1.1 Project Objective

- To build a spam filter that can effectively categorize an incoming mail or text message as either spam or ham.
- To detect unsolicited, unwanted, and virus-infected email.
- To obtain a very accurate classification rule.
- To establish a machine learning model with the appropriate dataset.
- To test the model to check its accuracy and precision rate.
- To validate the machine learning model with a new dataset that hasn't been recorded.

1.2 Model Objective

- To train the model with existing dataset.
- To maximize the accuracy of prediction of spam SMS or emails with machine learning algorithms as much as possible with the given data set.

1.3 Expected Output

Based on the results of the model, it will allow us to help companies and users to detect possible spam mails or SMS. Therefore, the model will predict the user's potential risk based on the previous studies and target information. So, it will decrease the possibility of spams.

1.4 Dataset

This dataset contains 5573 rows for two variables which indicates spam and ham. The first variable is the class, and the second variable is the SMS information. We have 2 different results and it could be either "spam" or "ham." We'll make things more formal by using a classifier for the text messages from the email. There is some ambiguous information with the data contents i.e. extra spacing, capital letters or stop words which are not necessarily needed and also the column names are misleading that would need some cleanings as well.

2.0 Literature Review

Spam categorization has been a hard subject for research, as spammers discover ways to manipulate spam word information by adding complexity. Different machine learning classifiers have been employed to address similar difficulties in study. This section discusses the description of literature relating to this research.

Bayesian approaches become more prominent and are increasingly used in text and spam categorization applications. It has advantages in a cost-sensitive assessment since it can give a better level of categorization certainty. A naive Bayes classificator and a word representation bag for email datasets have been employed for Sahami et al. investigation. This document has showed an enhanced performance level by adding complicated features (e.g., FREE!, f r 3 3), domain name features, and some non-alphabetic features in the bag.

A collection of classifiers is also part of the project. Sakkis et al. have devised a combined technique to construct a superior classification ensemble. This design contains NB and kNN categories in order to produce decent results. In a recent work by Trivedi and Dey, they employed an ensemble of classification approaches in relation to probabilistic classifications and showed that this strategy helped to improve the performance of probabilistic classifications even with a tiny subgroup of information.

Vector Machine Support (SVM) is also popular in this field. Drucker et al. carried out a comparative SVM investigation with several machine learner classifications and found the speed and accuracy of SVM and boosted decision tree as the promising classifiers. The training pace of the SVM was nevertheless quicker than the boosted decision tree. Trivedi and Dey evaluated the influence of several kernel functions on the performance of SVM and discovered that normalised poly kernels were the best ways of enhancing SVM's learning capacity.

2.1 Introduction to Spam

Using the definition of spam as unsolicited bulk messages, spamming may be defined as the act of sending these messages, and a spammer may be described as someone who engages in the practise of spamming. The majority of spam is commercial in form, and though it is bothersome, it is not always criminal or fraudulent in origin. It is inconvenient, it is usually promotional in nature, it is sent to a big number of individuals, and it arrives whether or not you requested it. The fact that you joined up for a marketing mailing and then became sick of it is terrible, but it is not spam.

The term "spam" refers to a Monty Python cartoon that was used to describe this type of aggressive blanket-messaging in the 1990s. In it, a group of diners (who happen to be dressed in Viking clothing, no less) asserts over and over that everyone must eat Spam, regardless of whether they want it. Similar to how an email spammer may inundate your inbox with unsolicited messages, this is a form of cyberbullying. When written with a capital S, the canned pork dish that the aforementioned Vikings like is referred to as "Spam." Lowercase S is used to refer to the never-ending deluge of emails and other messages that you never asked for or asked for in the first place.

2.2 Causes and Implications of Spamming

The unfortunate reality is that many businesses profit by selling your email address and other personal information to third parties. This issue has become so serious that the EU passed the General Data Protection Regulation (GDPR) in 2018, a set of rules aimed at limiting what firms may do with your personal information. Spammers employ spam because it is a low-cost method of communication. Spammers may send emails to everyone they can locate for very little money. The spammer will readily see a return on their investment if only a few people reply positively to the campaign. It's tough to hold spammers accountable because they utilise spoofing to hide their identities from receivers and internet service providers. Spam is an appealing alternative for less-than-scrupulous advertising and marketers because of the cheap risks and expenses.

Spam messages inflict serious harm by infecting users' computers with dangerous software capable of damaging systems and stealing personal information, in addition to the frustration and time lost sorting through unsolicited communications. It can potentially deplete network resources.

2.3 Algorithms Used

In this model Random Forest Regression machine algorithm will be applied. It is a supervised learning algorithm which uses the approach of ensemble learning for classification and regression. There are no inter-relations between these trees while constructing the trees. Later we used xgboost along with random forest for better accuracy.

How Does Random Forest Work?

It works by building a multitude of decision trees at training time and gives output which is mode of classification and regression. It can combine multiple predictions. I have decided to use the Random Forest method because compared to other algorithms Random Forest gives more accuracy in my dataset. Decision trees are sensitive to work with my dataset. In my case, the results of the decision tree can be quite different if the training data is altered and the predictions can also be slightly different in turn. In addition, it is also costly to train computationally, bear a great risk of overfitting, and appear to find local optima because after splitting it cannot be undone. (Chakure 2019 P.4), On the other hand, the major limitation of logistic regression is more than Random Forest and logistic regression can only be used to predict the discrete function. In my case it will give less accuracy.

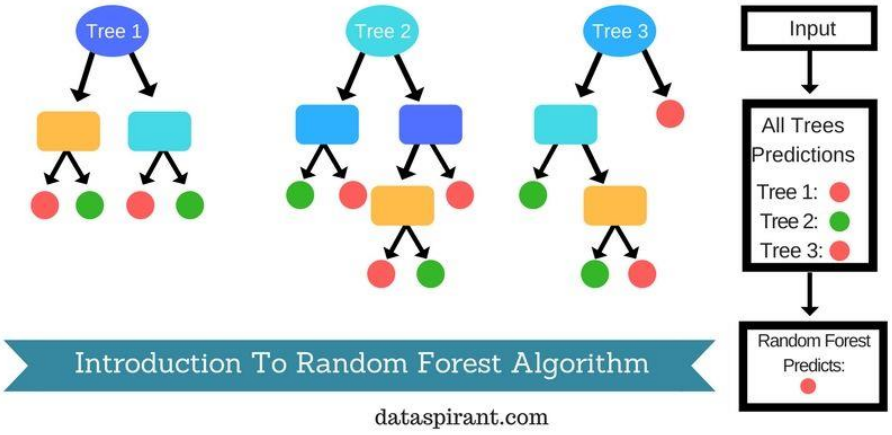

Fig 1: Random Forest (Source)

How Does XGBOOST Work?

Extreme Gradient Boosting Algorithm, or XGBoost, is another ensemble approach that operates by boosting trees. Gradient Boosting is the name given to XGboost since it uses a gradient descent technique. It is a gradient boosting-based decision-tree-based ensemble Machine Learning technique. Both XGBoost and gradient boosting machines are ensemble tree approaches that employ the gradient descent architecture to boost weak learners. XGBoost, on the other hand, enhances the underlying gradient boosting framework through system optimization and algorithmic improvements. Such as:

- Hardware optimization

- Parallelized tree building
- Fast handling of missing data
- Tree pruning utilising a "depth-first" strategy
- Built-in cross-validation capabilities are just a few examples (at each iteration)
- For preventing overfitting, LASSO (L1) and Ridge (L2) regularisation are used.

A regularised (L1 and L2) objective function with a function of convex loss and a model complexity penalty term is minimised using XGBoost. The training proceeds iteratively to create the final forecast, adding new trees that anticipate the residuals or mistakes of prior trees, which are then blended with prior trees. It's termed gradient boosting because it utilises a gradient descent technique to minimise loss while adding new models.

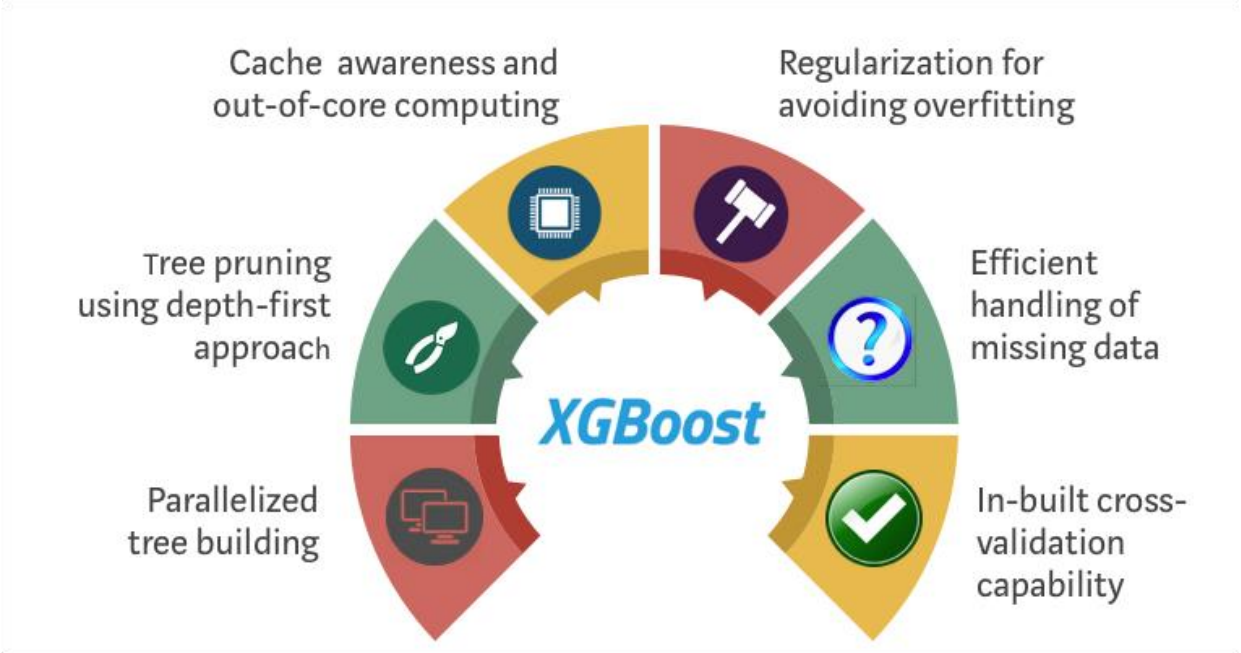

Fig 2: XGBoost (Source)

3.0 Experimental Setup

3.1 Dataset Description

This dataset spam.csv contains the information messages with a flag being spam or ham.This file has 5573 observations while having 2 attributes or variables.

The first variable is the email information, and the second variable is the goal variable, which is the expected class. We have 2 different results and it could be either "spam" or "ham." We'll make this more formal by including text messages from the email. The abstract of the dataset is shown below in a table to give a short idea:

spam.csv		
Variable name	**Defined as**	**Remarks**
V1	Class	It is basically referring to the categories of the class to identify a message as spam or ham. This will be gone through some pre-processings as we move on with the dataset and analysis more.
V2	SMS/Messages	These are the messages, or the texts inputted as strings which shows the SMS only. More cleaning is needed in order to process with these strings as we move on to the further steps ahead.

3.2 Data Pre-Processing

In this phase, we proceed with the process of taking raw data and converting it into usable information. In order to do this, we move forward taking each point and note it in a step-by-step approach. As the spam.csv data is not pre-processed well, we try to gather, filter, sort, process, analyse, and store them before being displayed in an usable way. These are the steps that we followed while processing the data:

3.2.1 Data Cleaning

After reviewing the data, we discovered several confusing column names in "spam.csv." We re-named V1 column to Class and V2 column to SMS to make the columns easier to identify. In addition, the Class column must be converted from Character strings to factor as there are only two values used as decisions. In our dataset, we also need to determine the percentage of ham to spam.
Such as-

```
#Rename columns

colnames(myData)

colnames(myData) <- c("Class", "SMS") #lets change to Class and SMS
colnames(myData) #lets check

myData$Class <- factor(myData$Class) #convert string classes to factor
prop.table(table(myData$Class)) #get the proportion of spam and ham
```

It has also been encountered that after importing the spam.csv dataset, some unusual and irrelevant columns pop out (e.g. ‘X’,’X.1’,’X.2’). Data frequently arrives from many sources and, most of the time, does not arrive in the correct format for the computer to handle. As these variable carry no meanings and have empty data inside, we removed those columns as well as the figure shows below:

```
#As there are some extra unused column, we shall drop the last columns
myData <- subset (myData, select = -c(X,X.1,X.2))
```

After renaming columns and dealing with some unusual variables, we came up with these results shown in the table below:

Column Name	**Changed to**
V1	Class
V2	SMS
‘X’,’X.1’,’X.2’	N/A

Now, we checked if our datasets have any missing values(NA) or empty strings using lapply() defined function and we found that our spam.csv dataset had absolutely no columns with null values or empty strings and our observations count remains the same with 5573 values.

```
> lapply(myData,function(x) { length(which(is.na(x)))})
$Class
[1] 0

$SMS
[1] 0
```

So, after having this done, now we are going to create a corpus for storing the texts or messages inside of it using VCorpus() function and deal with some necessary adjustments with the data:

- First of all, we needed to lowercase the words in text mining because in the further steps we need to find the count of every single word, identify them and group them together.
- We also eliminated stop words because we are not going to count them as words and moreover these don't contribute any sense to the model.
- We removed the numbers and punctuations from the texts may help in the next steps as numerical representations do not create any effect on making the bag of words.
- In addition, the stopwords are of no use while creating the model to identify spam or ham. So we can easily ignore it.
- As we do not want any white spaces in the texts or SMSs, therefore whitespaces are negligible for the analysis.

```
#put the words in lowercase, remove stops words and white spaces
corpus = tm_map(corpus, content_transformer(tolower))
corpus = tm_map(corpus, removeNumbers)
corpus = tm_map(corpus, removePunctuation)
corpus = tm_map(corpus, removeWords, stopwords("english"))
corpus = tm_map(corpus, stemDocument)
corpus = tm_map(corpus, stripWhitespace)
```

On the next step, as we get the texts stored in a corpus in a cleaned form without any symbols other than only the words, we create a bag of words for the model.

3.2.2 Bag of Words

In text mining, it's critical to gain a sense for the terms that indicate whether a text message is spam or ham. What percentage of the time do you hear each of these words? Which of the following words occurs the most? Thus we're going to bebuilding a DocumentTermMatrix to keep track of all these terms in order to answer this question.

Bag-of-words is a well-known and commonly used method for encoding textual data in documents for analysis. It is simple to learn and put into practise.
A textual document is represented as a bag-of-words, which essentially implies a word frequency table or an unordered set of words that preserve their frequency but ignore their place in the document.

It is a document matrix with respect to all the words present in the corpus we created earlier. It is a technique for extracting functionality from text records. These characteristics can be used for teaching machine learning algorithms. Below is the matrix:

	‰û÷ll	‰û÷s	‰ûò	abiola	abl	abt	accept	access	account	across	activ	actual	add	address
1214	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1215	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1216	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1217	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1218	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1219	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1220	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1221	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1222	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1223	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1224	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1225	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1226	No	No	No	No	No	No	No	No	Yes	No	No	No	No	No
1227	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes
1228	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1229	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1230	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1231	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1232	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1233	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1234	No	No	No	No	No	No	No	No	No	No	No	No	No	No
1235	No	No	No	No	No	No	No	No	No	No	No	No	No	No

From the above figure, we can see that the columns represent the words which we get from the corpus derived from the messages. On the other hand, the rows represent the position number of the text message/sms while showing the occurrence of every specific word with value ‘yes’ or ‘no’. For example, the sms at line 1226 has the word ‘account’ labeled with ‘yes’ which shows that there is an occurrence of the word ‘account’ in that message. In this way, we can get the idea on how frequently every word is occured that makes the training of the model easier ahead.

Later, after building the bag of words we can now include the Class variable to this matrix for proceeding with the model.

```
> #Adding the Class variable to the Dataset
> dataset$Class = myData$Class
> str(dataset$Class)
 Factor w/ 2 levels "ham","spam": 1 1 2 1 1 2 1 1 2 2 ...
```

3.2.3 Feature Extraction

In pattern recognition and machine learning, a feature is an individual characteristic or property of a phenomena being observed. It is the process of turning textual material into a certain format so that the key material may be analysed. For text categorization, there are numerous feature extraction approaches available, such as Bag of Words which we used for extracting features.

The process of extracting features entails minimising the amount of resources necessary to explain a huge amount of data. One of the primary issues with analysing complicated data is the large number of variables involved. A high number of variables necessitates a huge amount of memory and compute capacity, and it may also lead a classification method to overfit to training examples and generalise poorly to new examples. Feature extraction is a broad word encompassing ways of building variable combinations that avoid these issues while still accurately representing the data. In our case, if we had a large collection of sms and the keywords contained in these texts, then a feature extraction process could find correlations among the various keywords in the bag of words (i.e. using the word frequency in the matrix). Once we have done this, we can then associate certain combinations of words or phrases as spam and automatically out-filter these sms(s).

As we do not have any variables to remove or ignore in the dataset, we can simply ignore the feature selection part having all the variables used as main features in the model.

In our study, we have to show that spam messages can be identified with the right set of features that catch various characteristics of valid messages in order to separate them from spam messages. After creating the bag of words matrix and adding the class column to the matrix, we can select every variable/word as a feature and analyse the frequency of it which helps to decide whether the text will be spam or ham. The function of feature extraction is to train and test the classifier. By combining the Class variable, we produce a feature matrix in the bag of words and feature vectors as input of the algorithm that will be used in the next step. Hence, at the time of training data, this process extracts the frequency of words in the SMS texts.

3.3 Data Visualization

Data visualisation is the information and data graphical representation. Visual components such as graphs, plots, maps, charts may be used to view and get some ideas on the data. Visualization of data is one of the processes in the process of data science which indicates that it needs to be visualised for inferences to be formed once it has been gathered, processed and modelled. Data visualisation is also a facet of the wider DPA discipline, with the purpose of identifying, locating, processing, formatting and delivering data as efficiently as feasible. Below are the data visualizations that we came up with:

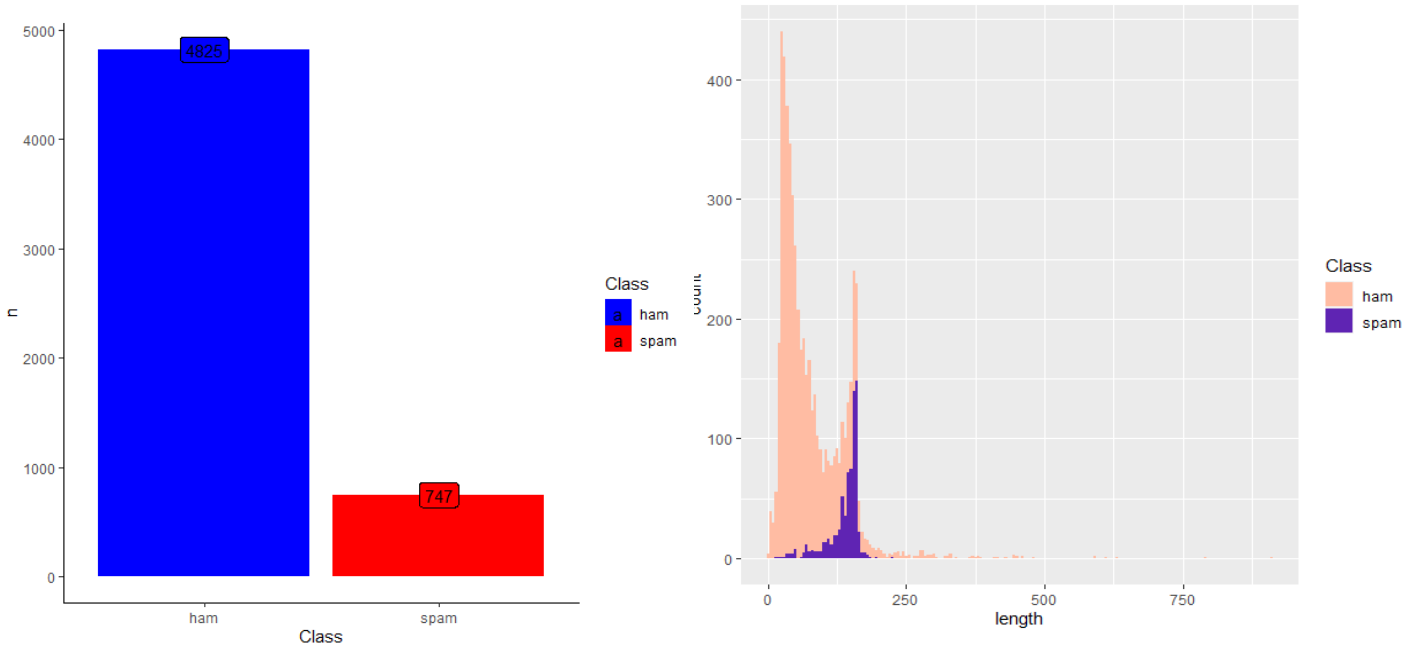

Figure 3: Plotting of Spam Vs Ham in the dataset

From the figure above, in the barplot it is clearly seen that the amount of ham is more compared to the amount of spam. It is calculated that around 8% of the texts or sms(s) are spam whereas around 92% of the texts remain intact as ham. We can also conclude that among a total of 5573 observations, only 747 observations remain spam.

As far as the ggplot of the spam vs ham relation is concerned, it gives us the idea on how the two types of classes are related with each other according to their respective lengths. The x-axis of the ggplot graph is the length of every string or texts in the dataset and based on the length of each sms string, it can be noticed that the ham messages have texts with larger lengths than that of spam bearing in mind that the count of ham is generally something uncatchable when it comes for spam count. This is also the reason why hams have a bigger length of texts than that of spams.

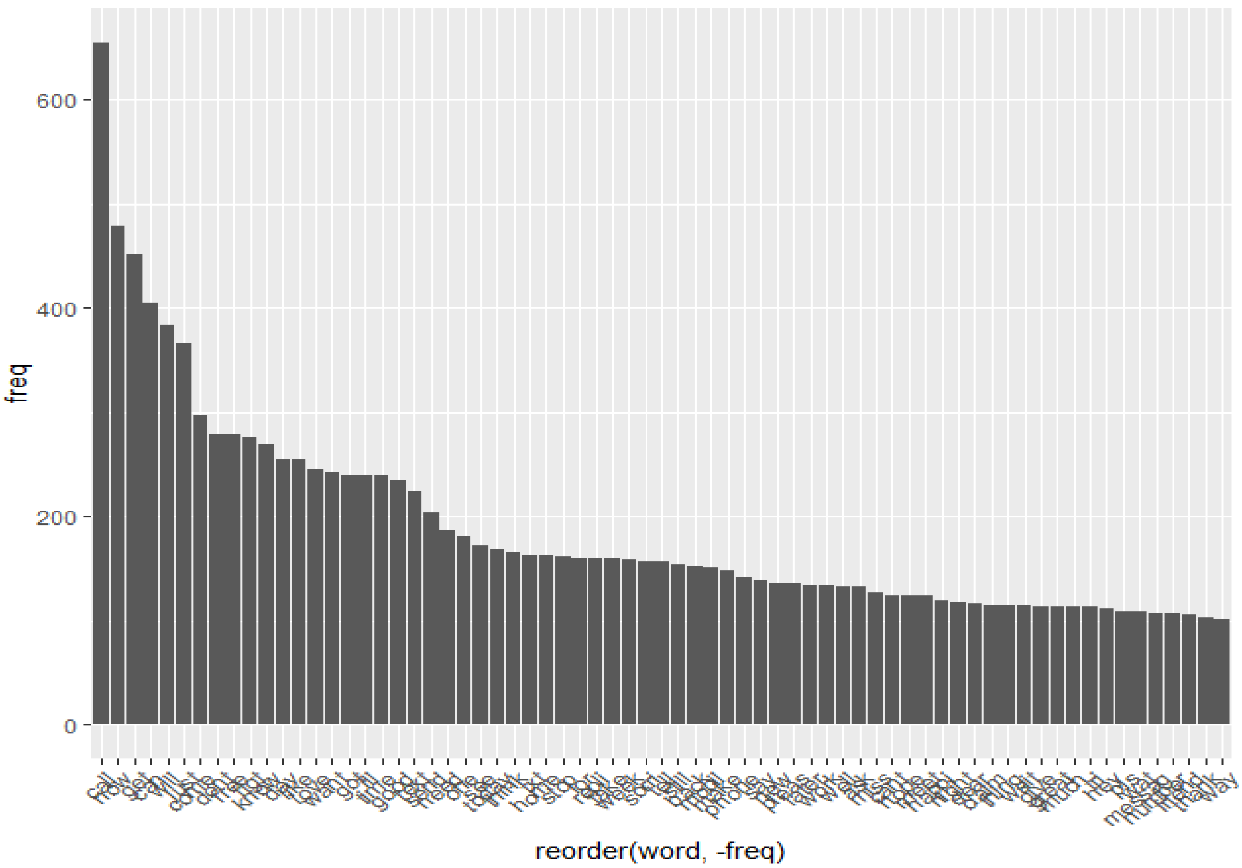

Figure 4: Graph of the frequencies of every word

The figure above is a bar plot showing the amount of time each word occured in every sentence or SMS throughout the whole dataset. A significant point to be noted is that we already have reduced the number of features by having taken the number of frequencies more than 100 for each word in the previous Data Pre-Processing phase. This is why, here is a representation of the words' frequencies that occur more than 60 times and we can visualize that the word 'call' has been used the most out of all.

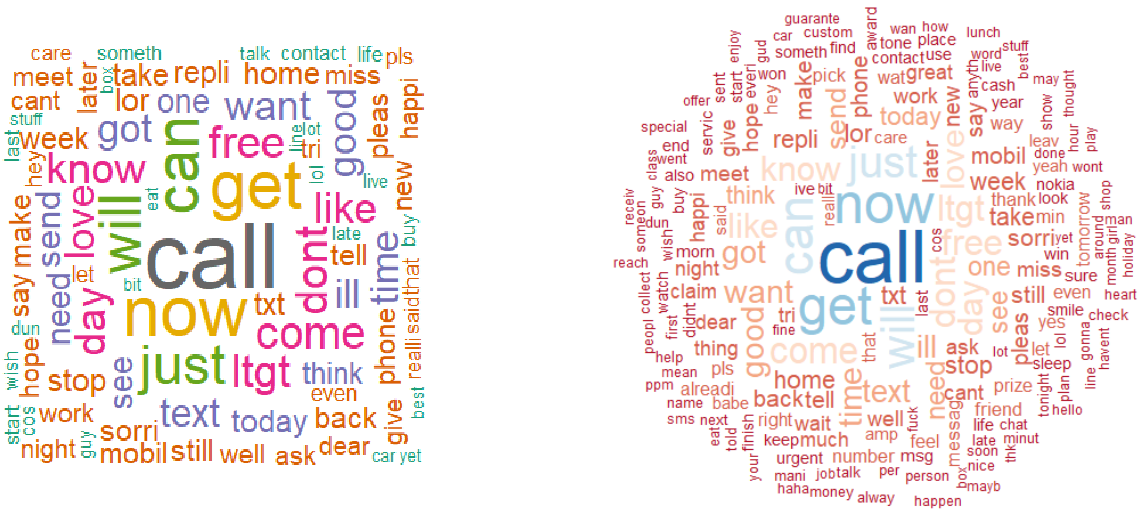

Figure 5: Word cloud representation (including spam and ham)

Setting a seed of 1234 and a max.words limit to 200(left) or more than 200(right), we get a visualization of word clouds like the above stated. It has been noticed that how high the number of max.words is set, the most frequent word remains the same, which is ‘call’. Moreover the words like ‘just’, ‘now’, ‘can’, ‘get’, ‘dont’, ‘like’, ‘love’, ‘day’ etc. are some honorable mentions of a frequent occurrence.

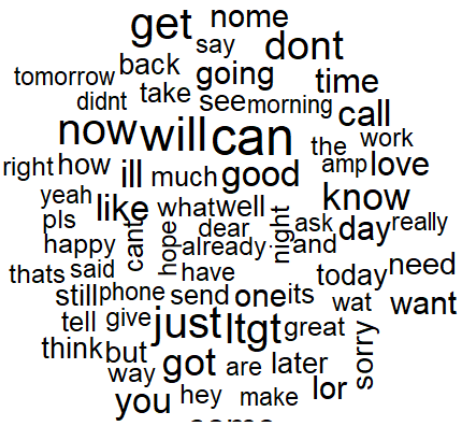

Figure 6: Ham word cloud

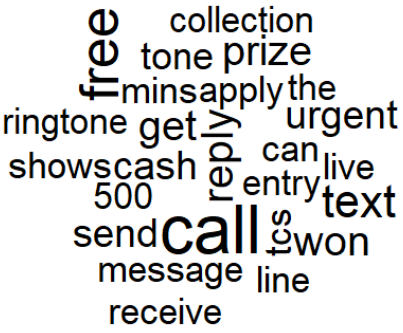

Figure 7: Spam word cloud

There are two figures above representing the visualization of the word cloud for ham(left) and span(right) as a comparison. Setting the max.words limit to 70 for both of the class type at a scale of 3 to 1, comparison between these figures show that:

- There are more and versatile sets of words in the ham category compared to the spam one.
- The words ‘text’ , ‘free’ and ‘call’ appeared too many times in the spam category.
- On the other hand, words like ‘now’, ‘will’, ‘get’, ‘dont’ etc. mark as a symbol of being a ham categorized word.

3.4 Model Development

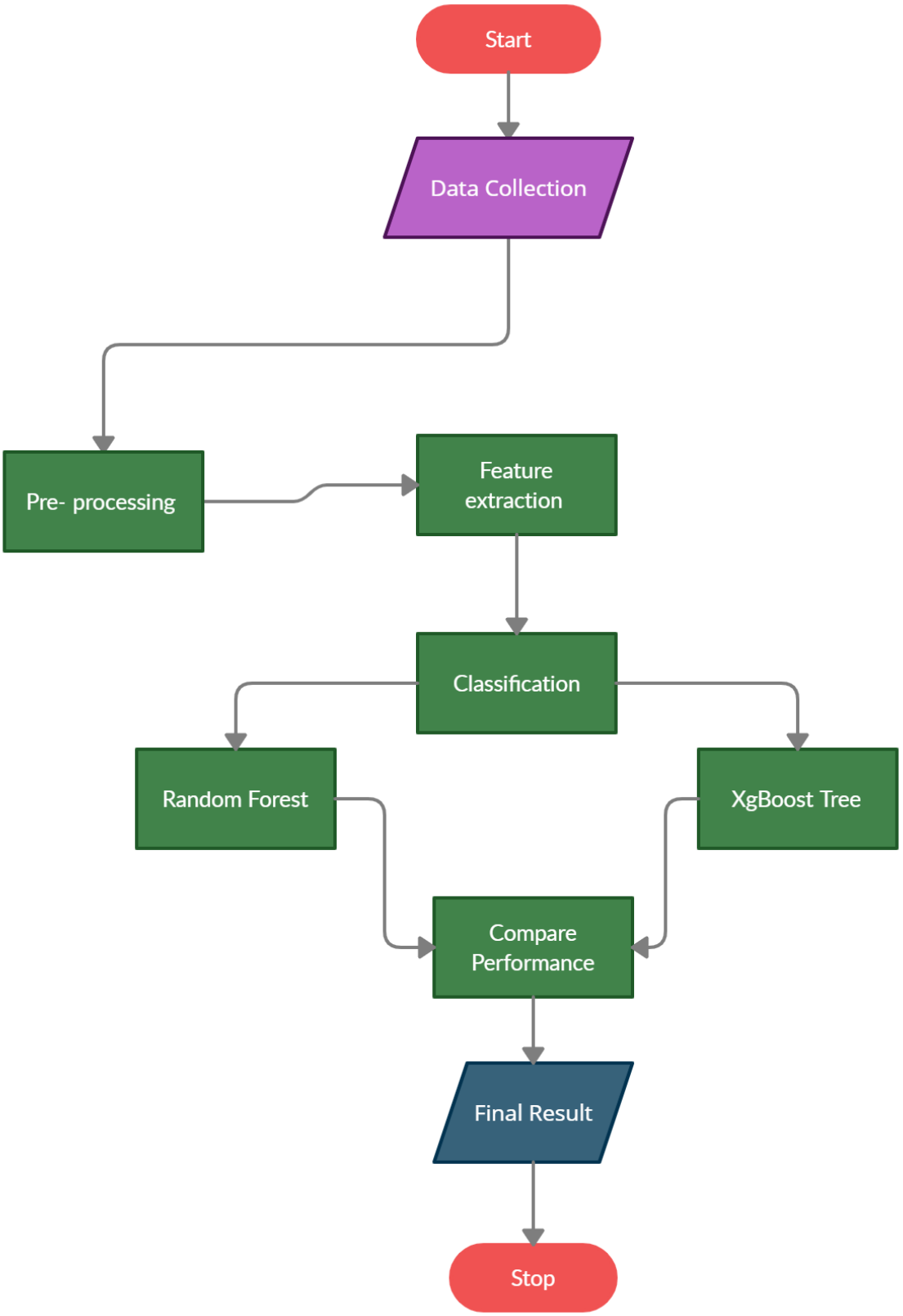

Figure 8: Flowchart of the model development

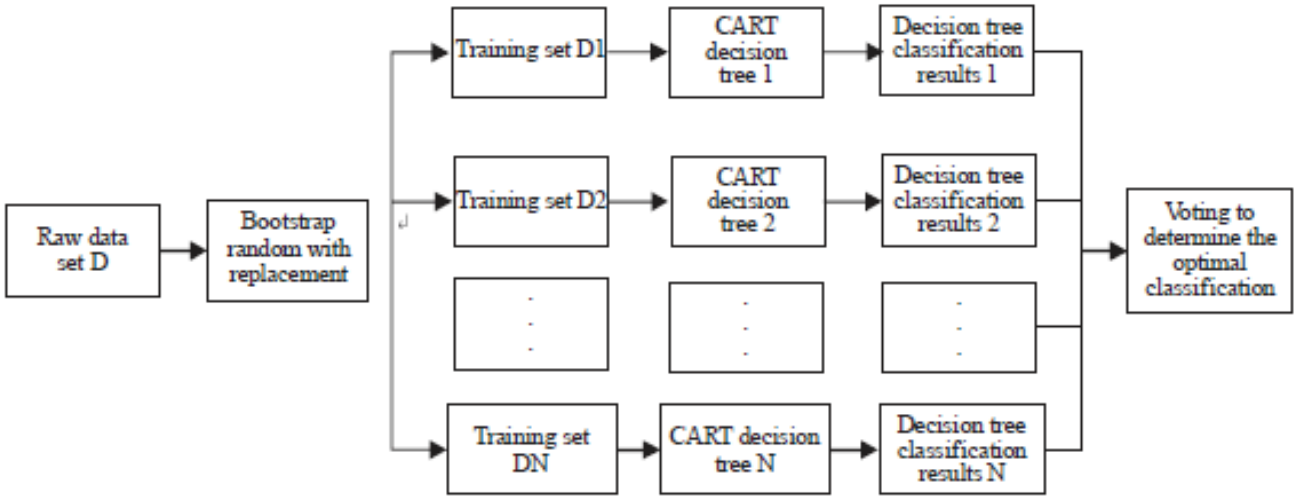

Figure 9: Flowchart of Random Forest

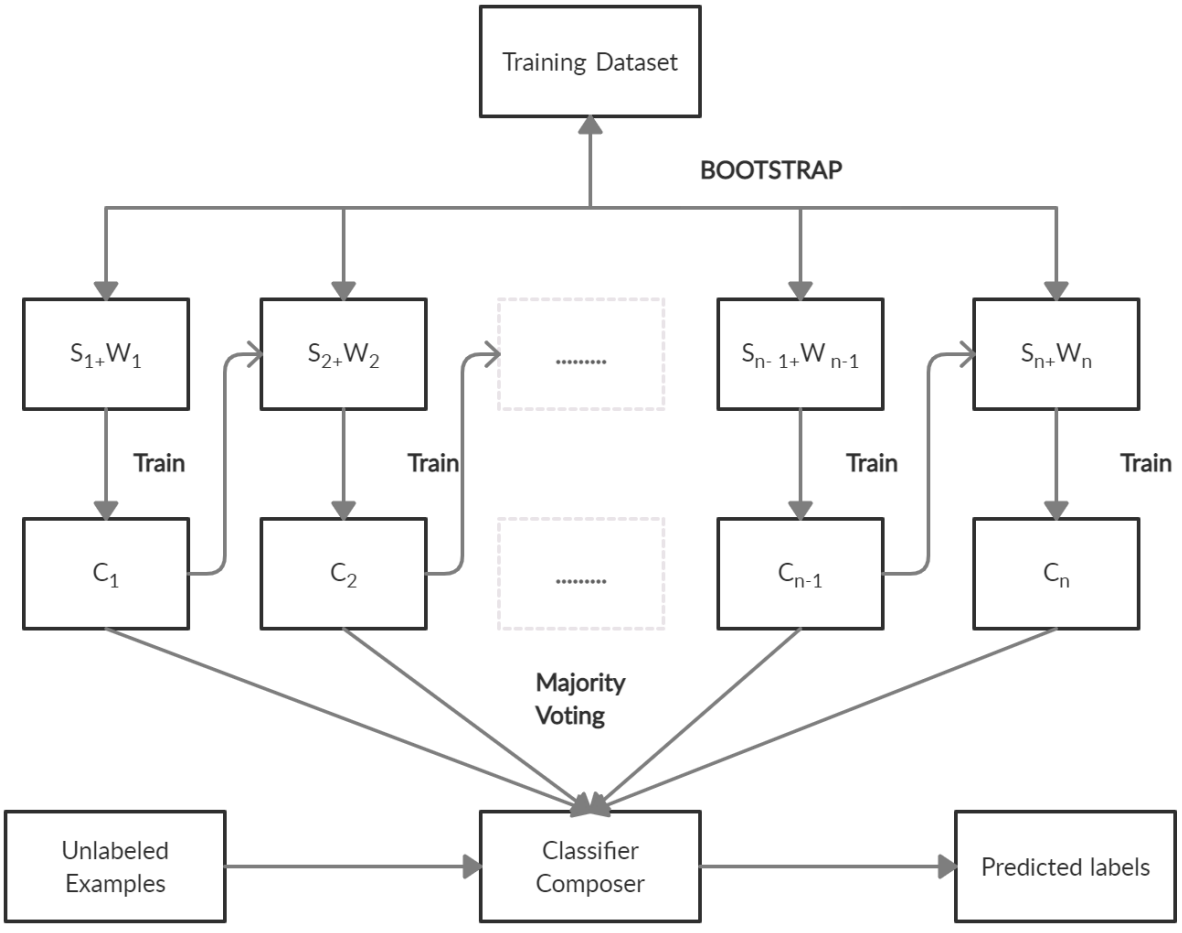

Figure 10: Flowchart of XgBoost tree

4.0 MODELING

To decide which machine performs more we will design our model based on two distinct machine learning methods, namely Random Forest and XgBoost tree. Both use decision trees but there is still a difference to show. When using XgBoosting Tree, we are encouraged to use shallow decision trees because of low variance and high bias but when using Random Forest, we are encouraged to use deep decision-making trees because of their high variance and their low bias. This could be an interesting comparison for us to work with our spam data.

4.1 Random Forest

Random forest is an algorithm of supervised learning that we used as our first model to fit. There is a hyperparameter for n estimators that is simply the number of trees that the algorithm creates before you take the whole vote or take the average of forecasts. In general, taller trees boost performance and stabilise the predictions, but also slow down calculations.

In our case, we will set the n_tree to 300 trees. This is a moderate approach in order to calculate the prediction in an accurate way with a shorter period of time.

Figure below shows the Random Forest implementation on the Spam and Ham dataset.

- Import the necessary libraries
- Import the data created for the modelling
- Build the model by splitting the data into 75%:25% ratio (Train:Test)
- Set the n-estimators to 300
- Make a proportion table for training and testing set for the class attribute
- Fit the Random Forest Classification algorithm and predict
- Construct confusion matrix and calculate accuracy

```
#import necessary libraries
library(tm)
library(SnowballC)
library(ggplot2)
library("wordcloud")
library("RColorBrewer")
library(randomForest)
library(caret)
library(cvms)
library(e1071)
library(magrittr) # needs to be run every time you start R and want to use %>%
library(dplyr)
library(pROC)
library(tidyverse)
```

```r
#Build Model
#Splitting the dataset into the Training set and Test set
set.seed(222)
split = sample(2,nrow(dataset),prob = c(0.75,0.25),replace = TRUE)
train_set = dataset[split == 1,]
test_set = dataset[split == 2,]
#proportion table
prop.table(table(train_set$Class))
prop.table(table(test_set$Class))




#Fit model no. 1 - Random Forest Classification
#We used 300 decision trees to build this model and made ntree=300
rf_classifier = randomForest(x = train_set[-1210],
                             y = train_set$Class,
                             ntree = 300)

rf_classifier #we have the class error of 0 which suggest that there is 100% accuracy

#lets check the actual accuracy by testing
# Predicting the Test set results
rf_pred = predict(rf_classifier, newdata = test_set[-1210])

# Confusion Matrix
con_m <- confusionMatrix(table(rf_pred,test_set$Class)) #99.8 accuracy
#plot
con_m_fig <- confusion_matrix(targets = test_set$Class,predictions = rf_pred)
plot_confusion_matrix(con_m_fig,add_row_percentages = TRUE,darkness = 0.7,
                      add_col_percentages = TRUE, palette = "Greens")
#Accuracy
accuracy1 = sum(rf_pred == test_set$Class)/length(rf_pred)*100 #100% accuracy
```

After applying the Random Forest algorithm we got an accuracy of 99.86% ~ 100%.

```
rf_pred  ham spam
   ham  1200    0
   spam    2  182

               Accuracy : 0.9986
                 95% CI : (0.9948, 0.9998)
    No Information Rate : 0.8685
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9937

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9983
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9891
             Prevalence : 0.8685
         Detection Rate : 0.8671
   Detection Prevalence : 0.8671
      Balanced Accuracy : 0.9992

       'Positive' Class : ham
```

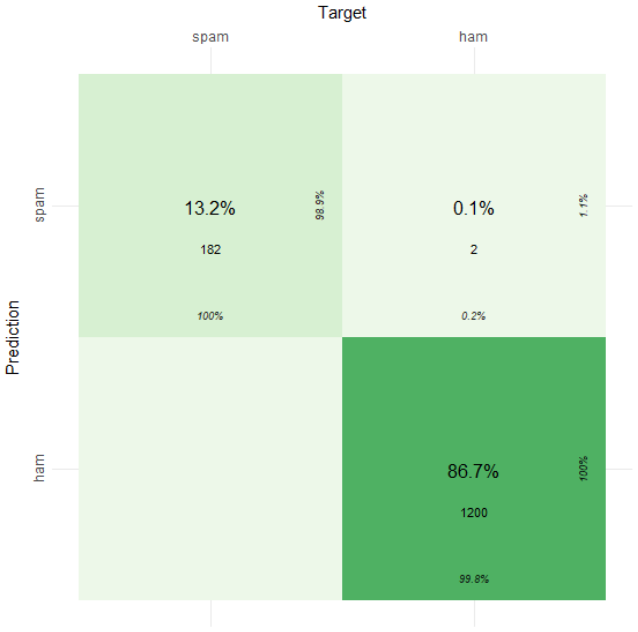

Figure 11: Confusion Matrix of Random Forest

We can evaluate that the random forest has done a brilliant job on predicting though it took a long time to run. This model has predicted all the spam values correctly without any fault whereas on the other hand the ham prediction was pretty much close to accurate while predicting only 2 ham texts wrongly as spam.

As we have included enough trees in the forest, the classifier won't overfit the model and it works well enough. However, the key restriction of random forests that we faced while performing the operation is that a large number of trees might make the algorithm too sluggish to forecast in real time. These algorithms are often rapid to train, but sluggish to predict once taught. More prediction calls for more trees, leading to a slower model.

4.2 XgBoost Tree

Figure below shows the XgBoost implementation on the Spam and Ham dataset.

- Import the data created for the modelling
- Build the model by using the 'cv' method
- Fit the XgBoost Classification algorithm by using the existing training and testing data
- Train the model setting metric to 'ROC'
- Predict using the model
- Construct confusion matrix and calculate accuracy

```r
#Fit model no. 2 - Xg boost Tree

set.seed(123)
fit_control <- trainControl(## cv
  method = "cv",
  number = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  allowParallel = TRUE)


##XgbTree
set.seed(123)
spam.xgb <- train(Class ~ .,
                  data = train_set,
                  method = "xgbTree",
                  metric = 'ROC',
                  trControl = fit_control)

xgb_predict <- predict(spam.xgb, test_set)
con_m2 = confusionMatrix(xgb_predict, test_set$Class) #97.8% accuracy
accuracy2 = sum(xgb_predict == test_set$Class)/length(xgb_predict)*100

con_m2_fig <- confusion_matrix(targets = test_set$Class,predictions = xgb_predict)
plot_confusion_matrix(con_m2_fig,add_row_percentages = TRUE,darkness = 0.7,
                      add_col_percentages = TRUE, palette = "Blues")
```

As a result, we got our statistics summary as:

```
          Reference
Prediction  ham spam
      ham  1195   23
      spam    7  159

               Accuracy : 0.9783
                 95% CI : (0.9692, 0.9853)
    No Information Rate : 0.8685
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9014

 Mcnemar's Test P-Value : 0.00617

            Sensitivity : 0.9942
            Specificity : 0.8736
         Pos Pred Value : 0.9811
         Neg Pred Value : 0.9578
             Prevalence : 0.8685
         Detection Rate : 0.8634
   Detection Prevalence : 0.8801
      Balanced Accuracy : 0.9339

       'Positive' Class : ham
```

This gives us an accuracy of 97.83% which can be rounded to 98%.

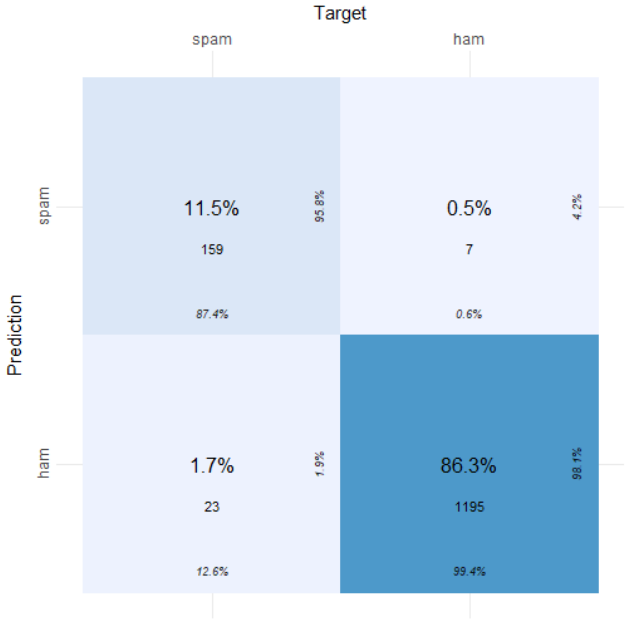

Figure 12: Confusion Matrix of XgBoost Tree

From the above figure, we can summarize that the XgBoost algorithm predicted the ham and spam values mostly in a correct way. Only 0.5% of hams have been identified wrong as spams whereas in the case of predicting spam, a percentage of 1.7 were predicted incorrectly as hams. Other than that, the rest of the 97.8% of the hams and spams are predicted correctly.

As we know that both XgBoost and Random Forest algorithms work pretty much the same way with decision trees but the model training of XgBoost is different as it is a boosting algorithm with shallow trees. This makes the model faster and even most of the times better than Random Forest if the data is not noisy. XGBoost's real advantages include its speed and ability to handle missing values.

Thus, we get a smoother and faster calculation speed while training the model but however we ended up with less accuracy than that of Random Forest because there are three parameters used in our XgBoost: the number of trees, the depth of trees and the learning rate whereas in RF, besides of using 'bagging', we used two primary parameters: the number of specified characteristics for each node and the number of decision-making processes.

5.0 RESULTS

Figures below show the comparison of the confusion matrix that was produced by the Random Forest and XgBoost algorithm. From this confusion matrix, we were able to conduct performance evaluation by calculating the accuracy, sensitivity, specificity, precision, recall, f-measure and error rate for the predictive model.

This is also to note that we took 25% of data as our test data which holds 1384 observations for our test data.

	Actual 0	Actual 1
Predicted 0	159	7
Predicted 1	23	1195

Figure: XgBoost Tree

	Actual 0	Actual 1
Predicted 0	182	2
Predicted 1	0	1200

Figure: Random Forest

Accuracy	**97.83%**
Sensitivity	**99.42%**
Specificity	**87.36%**
Precision	**98.11%**
Recall	**99.41%**
F-measure	**98.75%**
Error Rate	**0.02%**

Figure: Results of the evaluation performance of XgBoost

Accuracy	**99.86%**
Sensitivity	**99.83%**
Specificity	**100%**
Precision	**100%**
Recall	**99.83%**
F-measure	**99.91%**
Error Rate	**0%**

Figure: Results of the evaluation performance of RF

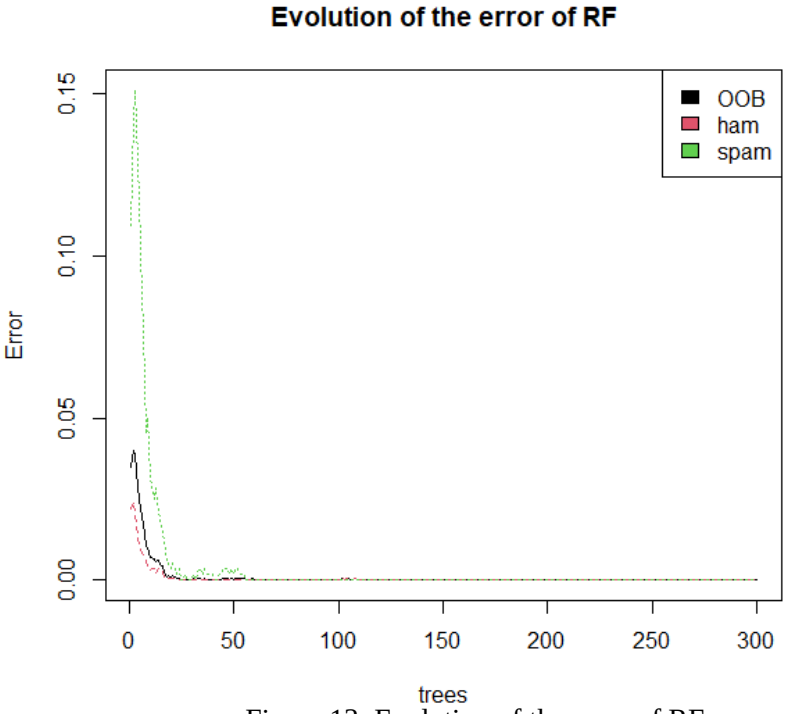

Figure 13: Evolution of the error of RF

From the above figure, we can notice that while training the RF model, the prediction of spam and ham becomes more accurate as the number of decision trees keep increasing from time to time, giving more accurate and approximately no errors in predictions which is excellent.

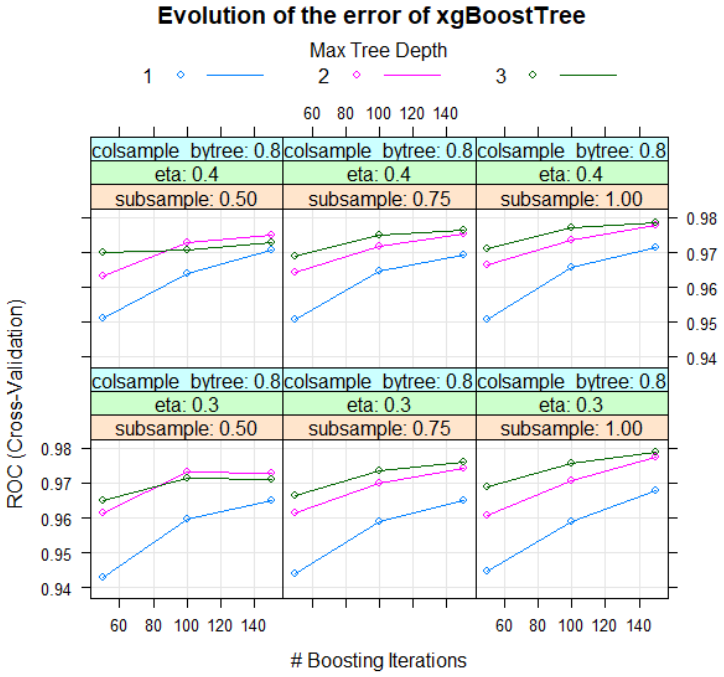

Figure 14: Evolution of the error of xgb Tree

The xgBoost Tree also shows improvement in every sequence of learning as the boosting iterations keep increasing, making it almost a low error rate of the model.

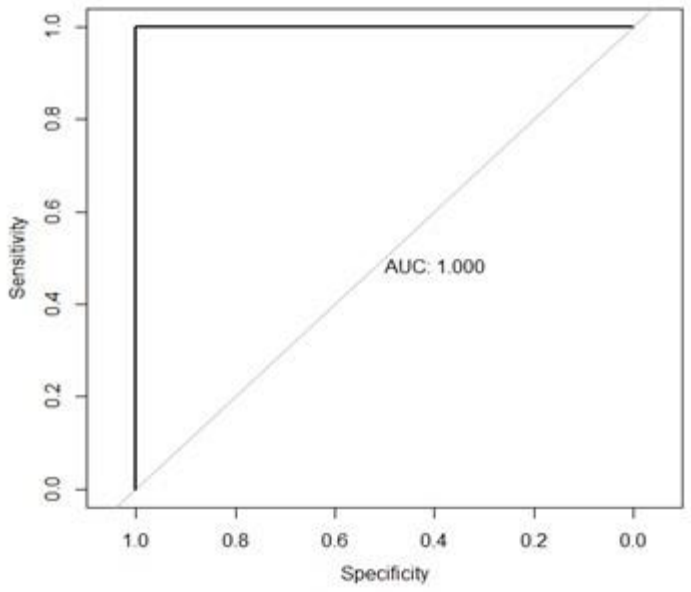

Figure 15: ROC Curve (RF)

We know that the higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. The above figure shows the ROC curve where the AUC rate is 100% for Random Forest.

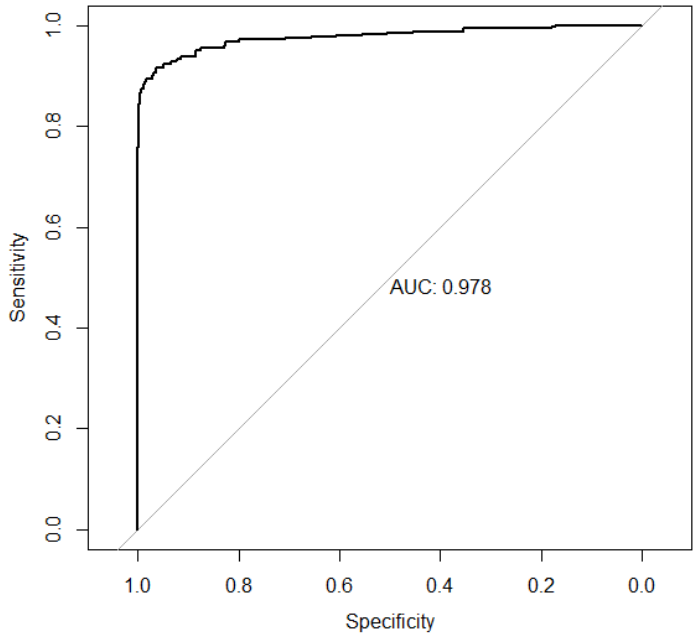

Figure 16: ROC Curve (XgbTree)

While looking at the ROC Curve of Xgbtree, there is slightly an error to the prediction where it does not completely make the 90 degree ROC angle, leaving the AUC to be at 97.8%.

Figure below shows the difference between some measures of the RF and XgBoost Tree.

Test Model	Accuracy	Sensitivity	Specificity
RF	99.86%	99.83%	100%
XgbTree	97.83%	99.42%	87.36%

To put it all in a nutshell, as it is clearly noticed that, although both models give excellent accuracy and less error rate, Random Forest is still unbeatable to its neighbor XgbTree with an accuracy of 99.9% and AUC of 100%. However, it is still a matter of fact that this model training takes more time and is slower than XgbTree. Therefore, keeping in mind that XgbTree can perform much faster with an enriched boosting algorithm, one can also go with this one as the difference between these two models' accuracy is very little, which is 2.03% only.

References:

- Jiang, Changjun et al. "Credit Card Fraud Detection: A Novel Approach Using Aggregation Strategy and Feedback Mechanism." IEEE Internet of Things Journal 5 (2018): 3637-3647.
- S. K. Trivedi, "A study of machine learning classifiers for spam detection," 2016 4th International Symposium on Computational and Business Intelligence (ISCBI), 2016, pp. 176-180, doi: 10.1109/ISCBI.2016.7743279.
- Spark, C. (2018, March 24). Getting started with XGBoost - Cambridge Spark. Medium. https://blog.cambridgespark.com/getting-started-with-xgboost-3ba1488bb7d4
- HOME. IDRE Stats. (n.d.). https://stats.idre.ucla.edu/r/faq/how-does-r-handle-date-values/.
- Statistical Data Cleaning with Applications in R *Journal* of Official Statistics, 19, 383–402. de Waal, T., Pannekoek, J., and ... G.R. (2017) daff: Diff, Patch and *Merge* for *Data.frames*,R Package Version 0.3.0.
- Differentiating between good credits and bad credits using neuro-fuzzy systems. (2002, January 1). ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0377221701000522
- Sheng-TunLia, a, WeissorShiueb, b, Meng-HuahHuangc, c, & AbstractThe commencement of the Basel II requirement. (2005, August 25). The evaluation of consumer loans using support vector machines. Expert Systems with Applications. https://reader.elsevier.com/reader/sd/pii/S0957417405001739?token=C5699A8A8A0278FBE35FA51D+17DD33D39F78352D8F14918BC37C2942E31FABD43D7FD6B410A3DA24775510745E95F1CA.
- Vapnik, V., 1995. *The nature of statistical learning theory*.Springer, New York.
- Haykin, S. (1999). *Neural networks: A comprehensive foundation*.Prentice Hall.

- Brownlee, J. (2020, August 20). How to Choose a Feature Selection Method For Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/
- Wikipedia contributors. (2021, May 13). Confusion matrix. Wikipedia. https://en.wikipedia.org/wiki/Confusion_matrix
- Y. (2018). ymattu/MlBayesOpt. GitHub. https://github.com/ymattu/MlBayesOpt
- J. (2017, July 11). Formulating the Support Vector Machine Optimization Problem. Math ∩ Programming. https://jeremykun.com/2017/06/05/formulating-the-support-vector-machine-optimization-problem/
- Brownlee, J. (2020a, August 18). How to Perform Feature Selection with Categorical Data. Machine Learning Mastery. https://machinelearningmastery.com/feature-selection-with-categorical-data/