```
#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)



#Import the data and look at the rows
myData <- read.csv(file = 'C:/Users/mashk/Desktop/ML/spam.csv')
head(myData)

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

#cheking null values
lapply(myData,function(x) { length(which(is.na(x)))})

#lets check
head(myData)
```

```r
#Data Wrangling

#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

#Data Cleaning
# Cleaning the SMS
corpus = VCorpus(VectorSource(myData$SMS)) #getting corpus
as.character(corpus[[1]])

#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)

#now lets check
as.character(corpus[[1]])
```

```
#Bag of Words
#In SMS mining get the frequency of each of the words in SMS
dtm = DocumentTermMatrix(corpus)
dtm


dtm = removeSparseTerms(dtm, 0.999)
dim(dtm)
inspect(dtm[40:50, 10:15])


#Converting the word frequencies to Yes and No Labels
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}



# Apply the convert_count function to get final training and testing DTMs
datasetNB <- apply(dtm, 2, convert_count)

dataset = as.data.frame(as.matrix(datasetNB))




#Descriptive Analysis of Data
#lets build word frequency

#We are preserving terms that appeared more than 60 times in the sample due to the large number of terms in the dataset.
```

```
freq<- sort(colSums(as.matrix(dtm)), decreasing=TRUE)

tail(freq, 10)

findFreqTerms(dtm, lowfreq=60)


#lets ggplot word frequency

wf<- data.frame(word=names(freq), freq=freq)

head(wf)


fp <- ggplot(subset(wf, freq>100), aes(x=reorder(word, -freq), y =freq)) +

  geom_bar(stat = "identity") +

  theme(axis.text.x=element_text(angle=45, hjust=1))

fp #we can see that the word 'call' is mostly frequent




#word cloud

set.seed(1234)

wordcloud(words = wf$word, freq = wf$freq, min.freq = 1,

          max.words=200, random.order=FALSE, rot.per=0.35,

          colors=brewer.pal(8, "RdBu"))


#creating word cloud for spam and ham

spam <- subset(myData, Class == "spam")

ham <- subset(myData, Class == "ham")


wordcloud(spam$SMS, max.words = 70, scale = c(3, 1))

wordcloud(ham$SMS, max.words = 70, scale = c(3, 1))


#Adding the Class variable to the Dataset
```

```
dataset$Class = myData$Class
str(dataset$Class)


#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 #99.8% accuracy
con_m

#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")



#Some Visualizations
#amount of spam and ham
myData %>%
  group_by(Class) %>%
  count() %>%
  ggplot(aes(Class, n, fill = Class))+
  geom_col()+
  geom_label(aes(label = n))+
  theme_classic()+
  scale_fill_manual(values = c('blue','red'))
#Another ggplot visualization of ham and spam amount
myData$length <- str_length(myData$SMS)
myData$Class = factor(myData$Class)

ggplot(myData,aes(x=length,fill=Class))+geom_histogram(binwidth=5)+scale_fill_manual(values=c("#ffbca3","#5f24b3"))+labs("Distribution of SMS length")
```

```
#error analysis of random forest
plot(rf_classifier, main ="Evolution of the error of RF")
legend('topright', colnames(rf_classifier$err.rate), col=1:3, fill=1:3)

##ROC
rf_pred <- predict(rf_classifier,test_set, type = 'prob')
ROC <- roc(test_set$Class, rf_pred[,2], auc = TRUE)
plot.roc(ROC, print.auc = TRUE)
```

```
#error analysis of xgbTree
plot(spam.xgb, main ="Evolution of the error of xgBoostTree")
##ROC
xgb_predict <- predict(spam.xgb, test_set, type = 'prob')
ROC <- roc(test_set$Class, xgb_predict[,2], auc = TRUE)
plot.roc(ROC, print.auc = TRUE)
```