

*# Importing the libraries*

library(caTools)

library(ggplot2)

library(MASS)

library(tm)

library(wordcloud)

library(caret)

library(e1071)

library(MLmetrics)

library(stringr)

*# Importing the dataset*

input\_data=input\_data<-read.csv("C:/Users/bvkka/Desktop/ISL-Deep Medhi/SMSSpamCollection.csv",

header = FALSE,

stringsAsFactors = FALSE)

str(input\_data)

*#Changing the name of the features/ columns*

colnames(input\_data) <- c("type", "text")

*#Converting the text to utf-8 format*

input\_data$text <- iconv(input\_data$text, to = "utf-8")

*#Type as factor*

input\_data$type <- factor(input\_data$type)

summary(input\_data)

table(input\_data$type)

prop.table(table(input\_data$type)) \* 100

set.seed(123)

*# Create a training set containing 80% of the data (with stratified sampling)*

trainIndex <- createDataPartition(input\_data$type, p = .8,

list = FALSE,

times = 1)

trainData <- input\_data[trainIndex,]

testData <- input\_data[-trainIndex,]

*# proportion in train dataset*

prop.table(table(trainData$type)) \* 100

*## ham spam*

*## 86.58591 13.41409*

*# proportion in test dataset*

prop.table(table(testData$type)) \* 100

*# Ham messages*

trainData\_ham <- trainData[trainData$type == "ham",]

head(trainData\_ham$text)

tail(trainData\_ham$text)

*# spam messages*

trainData\_spam <- trainData[trainData$type == "spam",]

head(trainData\_spam$text)

*# Removing the trainData\_ham and trainData\_spam*

trainData\_spam <- NULL

trainData\_ham <- NULL

*# create the corpus*

corpus <- Corpus(VectorSource(trainData$text))

*# basic info about the corpus*

print(corpus)

*#1. normalize to lowercase (not a standard tm transformation)*

corpus <- tm\_map(corpus, content\_transformer(tolower))

*#2. remove numbers*

corpus <- tm\_map(corpus, removeNumbers)

*#3. remove stopwords e.g. to, and, but, or (using predefined set of word in tm package)*

corpus <- tm\_map(corpus, removeWords, stopwords())

*#4. remove punctuation*

corpus <- tm\_map(corpus, removePunctuation)

*#5. normalize whitespaces*

corpus <- tm\_map(corpus, stripWhitespace)

*# Visualizing the data*

pal1 <- brewer.pal(9,"YlGn")

pal1 <- pal1[-(1:4)]

pal2 <- brewer.pal(9,"Reds")

pal2 <- pal2[-(1:4)]

*#min.freq initial settings -> around 10% of the number of docs in the corpus (40 times)*

par(mfrow = c(1,2))

wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors = pal1)

wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors = pal2)

*# Creation of the DTM considering terms with at least 2 chars*

sms\_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))

*# Basic information about the sparse matrix*

print(sms\_dtm)

inspect(sms\_dtm[1:10, 5:13])

sms\_features <- findFreqTerms(sms\_dtm, 5) *#find words that appears at least 5 times*

summary(sms\_features)

head(sms\_features)

sms\_dtm\_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms\_features))

print(sms\_dtm\_train)

convert\_counts <- function(x){

x <- ifelse(x > 0, 1, 0)

x <- factor(x, levels = c(0,1), labels = c("No", "Yes"))

return (x)

}

sms\_dtm\_train <- apply(sms\_dtm\_train, MARGIN = 2, convert\_counts)

head(sms\_dtm\_train[,1:5])

corpus <- Corpus(VectorSource(testData$text))

*#1. normalize to lowercase (not a standard tm transformation)*

corpus <- tm\_map(corpus, content\_transformer(tolower))

*#2. remove numbers*

corpus <- tm\_map(corpus, removeNumbers)

*#3. remove stopwords e.g. to, and, but, or (using predefined set of word in tm package)*

corpus <- tm\_map(corpus, removeWords, stopwords())

*#4. remove punctuation*

corpus <- tm\_map(corpus, removePunctuation)

*#5. normalize whitespaces*

corpus <- tm\_map(corpus, stripWhitespace)

sms\_dtm\_test <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms\_features))

*#print(sms\_dtm\_test)*

sms\_dtm\_test <- apply(sms\_dtm\_test, MARGIN = 2, convert\_counts)

sms\_dtm\_test[1:10, 5:12]

*#Evaluating the Model*

*#sms\_classifier <- naiveBayes(sms\_dtm\_train, trainData$type)*

sms\_classifier <- train(sms\_dtm\_train, trainData$type, method = "nb", trControl = trainControl(method = "cv", number = 10)) *#k fold cross validation*

sms\_classifier[[2]][1:5]

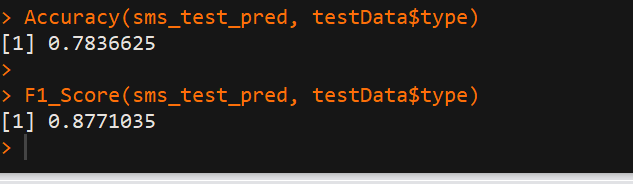
sms\_test\_pred <- predict(sms\_classifier$finalModel, sms\_dtm\_test)$class

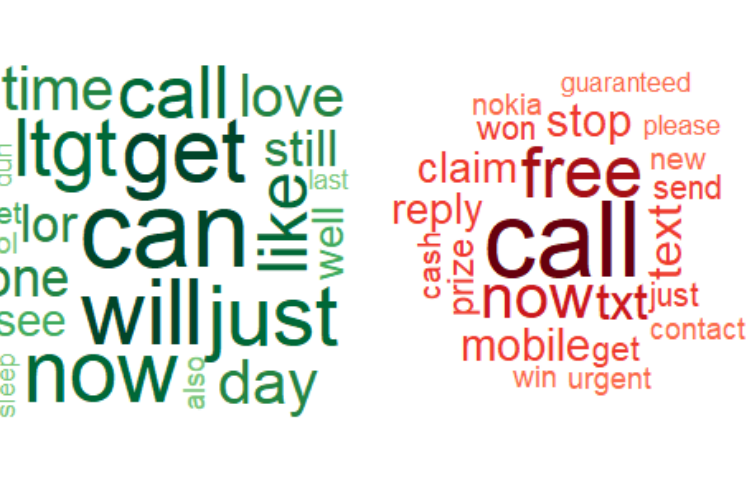
*#table actual (row) vs. predicted (col): confusion matrix*

Accuracy(sms\_test\_pred, testData$type)

F1\_Score(sms\_test\_pred, testData$type)

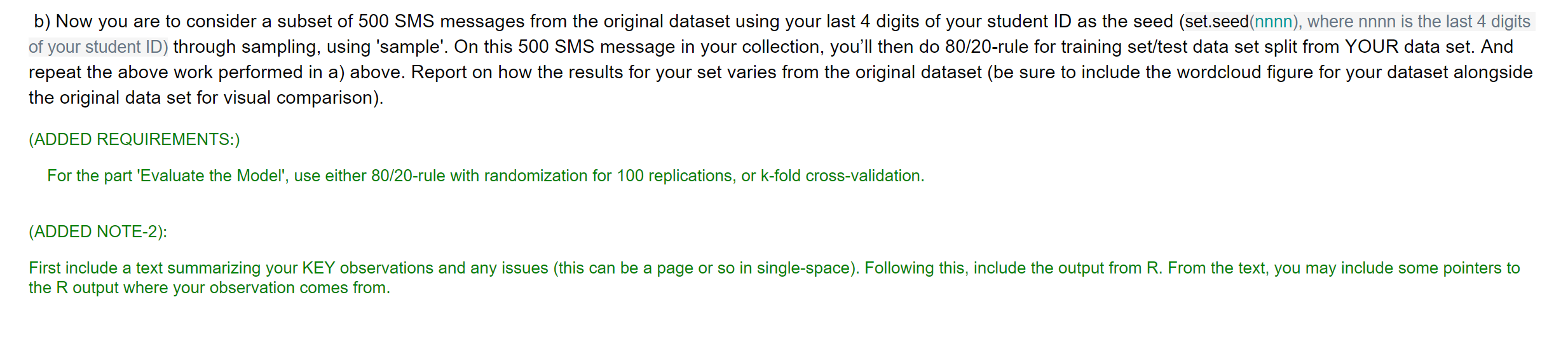
##\*\*Accuracy and Score\*\*\*###





**#####\*\*\*\*\*\*\*TEXT SUMMARIZATION\*\*\*\*\*\*\*\*######**

1. **We Observe that the world cloud figure is different from the URL source. The first one I attached is after 10 iterations and the second image I attached is after 100 iterations. We can see the difference how the words which are most frequently seen under spam and ham groups.**
2. **Note that the Accuracy and F1\_score obtained is different from the URL source. I took the mean of all 100 accuracy values and stored it under acc variable. Similarly, I took mean of all 100 F1\_Score values and stored it under F1\_Score variable.**



*# Importing the libraries*

library(caTools)

library(ggplot2)

library(MASS)

library(tm)

library(wordcloud)

library(caret)

library(e1071)

library(MLmetrics)

library(stringr)

*# Importing the libraries*

library(caTools)

library(ggplot2)

library(MASS)

library(tm)

library(wordcloud)

library(caret)

library(e1071)

library(MLmetrics)

library(stringr)

*# Importing the dataset*

input\_data=input\_data<-read.csv("C:/Users/bvkka/Desktop/ISL-Deep Medhi/SMSSpamCollection.csv",

header = FALSE,

stringsAsFactors = FALSE)

str(input\_data)

*#Changing the name of the features/ columns*

colnames(input\_data) <- c("type", "text")

*#Converting the text to utf-8 format*

input\_data$text <- iconv(input\_data$text, to = "utf-8")

*#Type as factor*

input\_data$type <- factor(input\_data$type)

*# Cleaning the raw data by removing the na's*

input\_data = na.omit(input\_data)

*# Taking only 500 messages from the main dataset*

data = input\_data[sample(nrow(input\_data), 500), ]

summary(data)

table(data$type)

prop.table(table(data$type)) \* 100

set.seed(0698) *###\*\*\*\*MY STUDENT ID LAST FOUR DIGITS\*\*\*\*###*

*# Create a training set containing 80% of the data (with stratified sampling)*

trainIndex <- createDataPartition(data$type, p = .8,

list = FALSE,

times = 1)

trainData <- data[trainIndex,]

testData <- data[-trainIndex,]

*# proportion in train dataset*

prop.table(table(trainData$type)) \* 100

*## ham spam*

*## 86.58591 13.41409*

*# proportion in test dataset*

prop.table(table(testData$type)) \* 100

*# Ham messages*

trainData\_ham <- trainData[trainData$type == "ham",]

head(trainData\_ham$text)

tail(trainData\_ham$text)

*# spam messages*

trainData\_spam <- trainData[trainData$type == "spam",]

head(trainData\_spam$text)

*# Removing the trainData\_ham and trainData\_spam*

trainData\_spam <- NULL

trainData\_ham <- NULL

*# create the corpus*

corpus <- Corpus(VectorSource(trainData$text))

*# basic info about the corpus*

print(corpus)

*#1. normalize to lowercase (not a standard tm transformation)*

corpus <- tm\_map(corpus, content\_transformer(tolower))

*#2. remove numbers*

corpus <- tm\_map(corpus, removeNumbers)

*#3. remove stopwords e.g. to, and, but, or (using predefined set of word in tm package)*

corpus <- tm\_map(corpus, removeWords, stopwords())

*#4. remove punctuation*

corpus <- tm\_map(corpus, removePunctuation)

*#5. normalize whitespaces*

corpus <- tm\_map(corpus, stripWhitespace)

*# Visualizing the data*

pal1 <- brewer.pal(9,"YlGn")

pal1 <- pal1[-(1:4)]

pal2 <- brewer.pal(9,"Reds")

pal2 <- pal2[-(1:4)]

*#min.freq initial settings -> around 10% of the number of docs in the corpus (40 times)*

par(mfrow = c(1,2))

wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors = pal1)

wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors = pal2)

*# Creation of the DTM considering terms with at least 2 chars*

sms\_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))

*# Basic information about the sparse matrix*

print(sms\_dtm)

inspect(sms\_dtm[1:10, 5:13])

sms\_features <- findFreqTerms(sms\_dtm, 5) *#find words that appears at least 5 times*

summary(sms\_features)

head(sms\_features)

sms\_dtm\_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms\_features))

print(sms\_dtm\_train)

convert\_counts <- function(x){

x <- ifelse(x > 0, 1, 0)

x <- factor(x, levels = c(0,1), labels = c("No", "Yes"))

return (x)

}

sms\_dtm\_train <- apply(sms\_dtm\_train, MARGIN = 2, convert\_counts)

head(sms\_dtm\_train[,1:5])

corpus <- Corpus(VectorSource(testData$text))

*#1. normalize to lowercase (not a standard tm transformation)*

corpus <- tm\_map(corpus, content\_transformer(tolower))

*#2. remove numbers*

corpus <- tm\_map(corpus, removeNumbers)

*#3. remove stopwords e.g. to, and, but, or (using predefined set of word in tm package)*

corpus <- tm\_map(corpus, removeWords, stopwords())

*#4. remove punctuation*

corpus <- tm\_map(corpus, removePunctuation)

*#5. normalize whitespaces*

corpus <- tm\_map(corpus, stripWhitespace)

sms\_dtm\_test <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms\_features))

*#print(sms\_dtm\_test)*

sms\_dtm\_test <- apply(sms\_dtm\_test, MARGIN = 2, convert\_counts)

sms\_dtm\_test[1:10, 5:12]

*#Evaluating the Model*

*#sms\_classifier <- naiveBayes(sms\_dtm\_train, trainData$type)*

sms\_classifier <- train(sms\_dtm\_train, trainData$type, method = "nb", trControl = trainControl(method = "cv", number = 10)) ***# K fold Cross Validation where K=10***

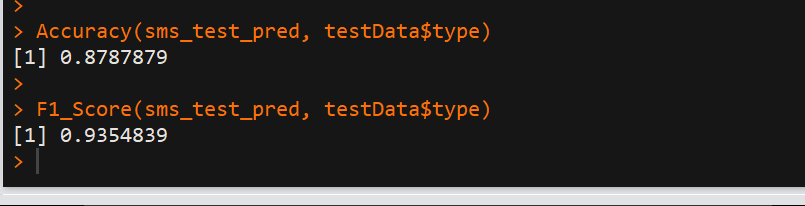
sms\_classifier[[2]][1:5]

sms\_test\_pred <- predict(sms\_classifier$finalModel, sms\_dtm\_test)$class

*#table actual (row) vs. predicted (col): confusion matrix*

Accuracy(sms\_test\_pred, testData$type)

F1\_Score(sms\_test\_pred, testData$type)





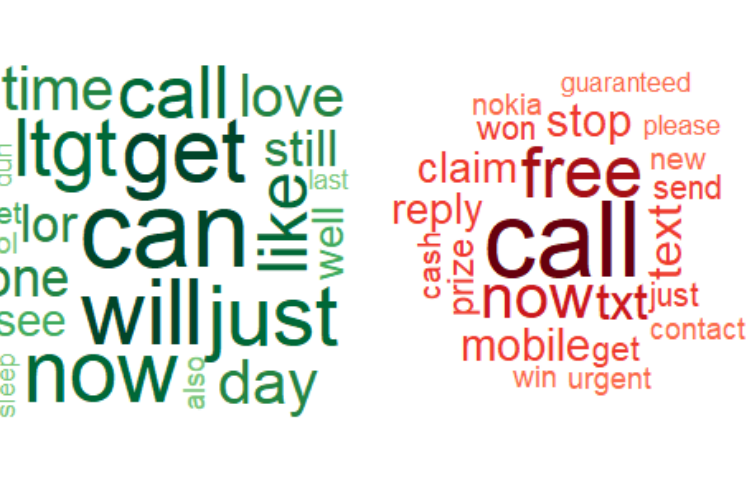
|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **F1\_Score** |
| **Whole data set** | 0.7836625 | 0.8771035 |
| **500 data subset points and seed value set to 0698** | 0.8787879 | 0.9354 |

**Text Summarization:**

**I USED K FOLD CROSS VALIDATION WHERE K=10**

The data set when considered completely and applied k fold cross validation with training as 80% and testing as 20% of data has more accuracy and F1\_Score values when compared to the model where 500 points are sampled and applied k fold cross validation with training as 80% and testing as 20% of data and seed value set. Hence, whole data set fit model is the best fit and good classifier model.

**#####\*\*\*\*\*\*COMPARING TWO WORDCLOUD FIGURE\*\*\*\*####**

PART A PART B

# we notice that the number of most frequent words in ham and spam class of Part A is less compared to the number of most frequent words in ham and spam class of Part B