Question 3:

a) First, do the entire steps discussed in https://rpubs.com/pparacch/237109 to do naive Bayes classification on a dataset consisting of SMS messages. The data set on SMS messages is discussed at http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/ and can be downloaded from http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/smsspamcollection.zip

Solution:

Source code:

```
require(caret)
require(tm)
require(wordcloud)
require(e1071)
require(MLmetrics)
rawData <-  read.csv("C:\\Users\\sudheesha\\Documents\\R\\output_SMS_file.txt",
                header = FALSE,
                stringsAsFactors = FALSE)
#Changing the name of the features/ columns
colnames(rawData) <- c("type", "text")
#Converting the text to utf-8 format
rawData$text <- iconv(rawData$text, to = "utf-8")
#Type as factor
rawData$type <- factor(rawData$type)
a=1
rep=100
accuracy=dim(rep)
precision=dim(rep)
recall=dim(rep)

for (k in 1:rep)
{
 trainIndex <- createDataPartition(rawData$type, p = .8,
                           list = FALSE,
                           times = 1)
 trainData <- rawData[trainIndex,]
 testData <- rawData[-trainIndex,]
 corpus <- Corpus(VectorSource(trainData$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)
 #Creation of the DTM considering terms with at least 2 chars
```

```
  sms_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))
  sms_features <- findFreqTerms(sms_dtm, 5) #find words that appears at least 5 times
  sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary =
sms_features))
  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)
  sms_classifier <- naiveBayes(sms_dtm_train, trainData$type)
  corpus <- Corpus(VectorSource(testData$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))
  sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts)
  sms_test_pred <- predict(sms_classifier, sms_dtm_test)
  tablin = table(testData$type, sms_test_pred)
  accuracy[k] = (tablin[1,1]+tablin[2,2])/(sum(tablin))
  precision[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1])
  recall[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2])
  if(a==1)
  {
    a=a+1
    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)
  }
}
cat("Accuracy: ",mean(accuracy))
cat("Precision :  ",mean(precision))
cat("Recall: ",mean(recall))
```

Output:
Accuracy: 0.8077558
Precision: 0.8820229
Recall: 0.8983731

Visualization:

b) Now you are to consider a subset of 500 SMS messages from the original dataset using your last 4 digits of your student ID as the seed (set.seed(nnnn), where nnnn is the last 4 digits of your student ID) through sampling, using 'sample'. On this 500 SMS message in your collection, you'll then do 80/20-rule for training set/test data set split from YOUR data set. And repeat the above work performed in a) above. Report on how the results for your set varies from the original dataset (be sure to include the wordcloud figure for your dataset alongside the original data set for visual comparison).

Solution:

Source code:

```
require(caret)
require(tm)
require(wordcloud)
require(e1071)
require(MLmetrics)
rawData <-  read.csv("C:\\Users\\shankar pentyala\\Documents\\R\\output_SMS_file.txt",
                     header = FALSE,
                     stringsAsFactors = FALSE)
#Changing the name of the features/ columns
colnames(rawData) <- c("type", "text")
```

```
#Converting the text to utf-8 format
rawData$text <- iconv(rawData$text, to = "utf-8")

#Type as factor
rawData$type <- factor(rawData$type)
summary(rawData)
a=1

n=5574
nt=500
rep=100
accuracy=dim(rep)
precision=dim(rep)
recall=dim(rep)
for (k in 1:rep)
{
 set.seed(1536)
  Index <- sample(1:n,nt)
  rawData500 <- rawData[Index,]

  trainIndex <- createDataPartition(rawData500$type, p = .8, list = FALSE, times = 1)

  trainData <- rawData500[trainIndex,]
  testData <- rawData500[-trainIndex,]


  corpus <- Corpus(VectorSource(trainData$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)


  #Creation of the DTM considering terms with at least 2 chars
  sms_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))

  sms_features <- findFreqTerms(sms_dtm, 5) #find words that appears at least 5 times

  sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary =
sms_features))

  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)

  sms_classifier <- naiveBayes(sms_dtm_train, trainData$type)

  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))

  sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts)

  sms_test_pred <- predict(sms_classifier, sms_dtm_test)

  tablin = table(testData$type, sms_test_pred)
  accuracy[k] = (tablin[1,1]+tablin[2,2])/(sum(tablin))
  precision[k] = (tablin[1,1])/(tablin[1,1]+tablin[2,1])
  recall[k]=(tablin[1,1])/(tablin[1,1]+tablin[1,2])
  if(a==1)
  {
  a=a+1
  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)
  }


 }
```

```
cat("Accuracy: ",mean(accuracy))
cat("Precision :  ",mean(precision))
cat("Recall: ",mean(recall))
```

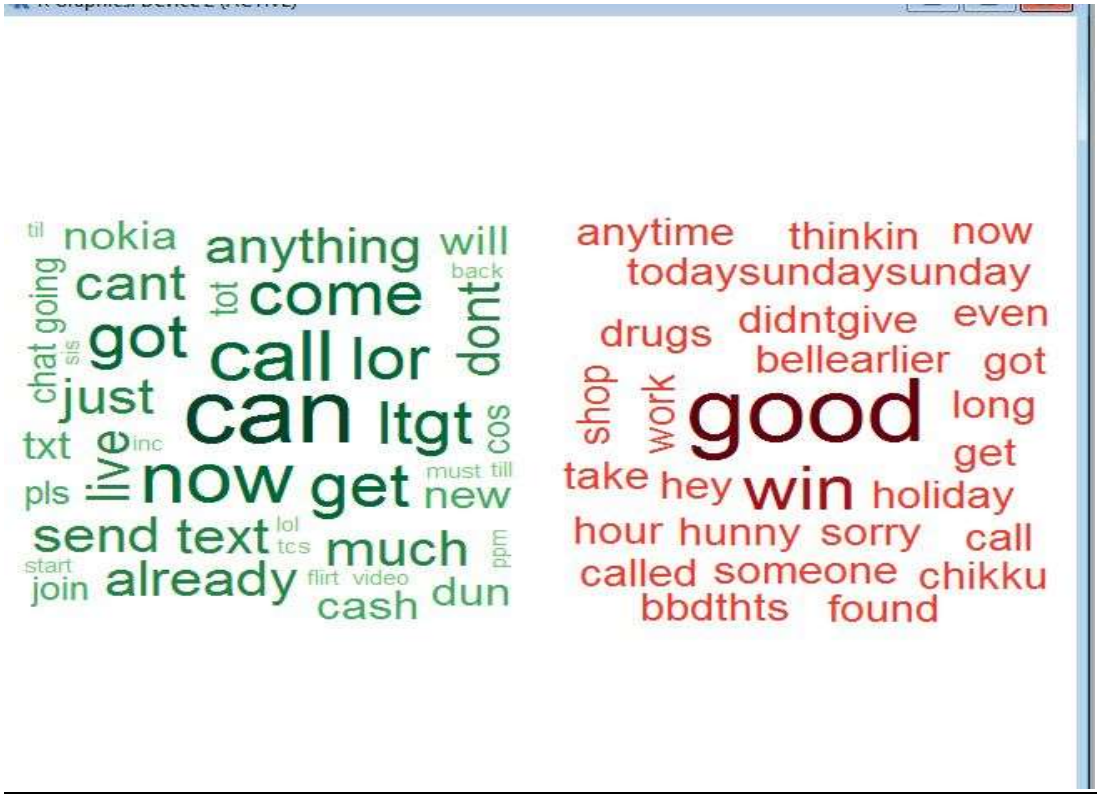
Output:

Accuracy: 0.7575758
Precision: 0.8554217
Recall: 0.8554217

Visualization of Original data set vs data subset of 500 samples:

Original data set:

Subset Visualization:



Summary:

	Accuracy	**Precision**	**Recall**
Whole data set	0.8077558	0.8820229	0.8983731
500 data subset points and seed value set to 1536	0.7575758	0.8554217	0.8554217

Text Summarization:

1. The data set when considered completely and replicated for 100 times with training as 80% and testing as 20% of data has more accuracy, precision and recall values when compared to the model where 500 points are sampled for every iteration and 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.
2. The computation power required for building whole data set model is very high and it took roughly more than an hour for building the model whereas the sampled data set of 500 points took less than 2 minutes for building the model. Both these models are built in the local system.
3. Since, the seed value is set while building the model for 500 sample data points the model even built once or replicated 100 times results in the same model.