ISL Assignment -2-Q3

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:

```
> tmp2 <- read.csv("~/R/tmp2.txt")
>   View(tmp2)
> rawData<-tmp2
> str(rawData)
'data.frame':  5573 obs. of  2 variables:
 $ ham                  : Factor w/ 2 levels "ham","spam": 1 2 1 1 2 1 1 2 2 1 ...
 $ Go.until.jurong.point: Factor w/ 4967 levels "","'AH POOR BABY!HOPE URFEELING BETTERSN LUV! PRO
BTHAT OVERDOSE OF WORK HEY GO CAREFUL SPK 2 U SN LOTS OF LOVEJEN XXX.'",..: 3137 1033 4112 2797 10
58 967 437 4591 1250 1761 ...
> colnames(rawData) <- c("type", "text")
> rawData$text <- iconv(rawData$text, to = "utf-8")
> rawData$type <- factor(rawData$type)
> summary(rawData)
   type          text
 ham :4826   Length:5573
 spam: 747   Class :character
             Mode  :character
> table(rawData$type)

 ham spam
4826  747
> prop.table(table(rawData$type)) * 100

     ham     spam
86.59609 13.40391
> set.seed(0695)
> trainIndex <- createDataPartition(rawData$type, p = .80,
+                                   list = FALSE,
+                                   times = 1)
> trainData <- rawData[trainIndex,]
> testData <- rawData[-trainIndex,]
> prop.table(table(trainData$type)) * 100

     ham     spam
86.58892 13.41108
> prop.table(table(testData$type)) * 100

     ham     spam
86.62478 13.37522
> trainData_ham <- trainData[trainData$type == "ham",]
> head(trainData_ham$text)
[1] "Ok lar... Joking wif u oni..."
[2] "U dun say so early hor... U c already then say..."
[3] "Nah I don't think he goes to usf"
[4] "Even my brother is not like to speak with me. They treat me like aids patent."
[5] "I'm gonna be home soon and i don't want to talk about this stuff anymore tonight"
[6] "I've been searching for the right words to thank you for this breather. I promise i wont take
your help for granted and will fulfil my promise. You have been wonderful and a blessing at all ti
mes."
> tail(trainData_ham$text)
[1] "Ok lor... Sony ericsson salesman... I ask shuhui then she say quite gd 2 use so i considering
..."
[2] "Ard 6 like dat lor."
[3] "Huh y lei..."
[4] "Will ÃƒÂ¼ b going to esplanade fr home?"
[5] "Pity"
[6] "The guy did some bitching but I acted like i'd be interested in buying something else next we
ek and he gave it to us for free"
> trainData_spam <- trainData[trainData$type == "spam",]
> head(trainData_spam$text)
[1] "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to recei
ve entry question(std txt rate)T&C's apply 08452810075over18's"
[2] "FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up f
or it still? Tb ok! XxX std chgs to send"
[3] "WINNER!! As a valued network customer you have been selected to receivea Ã,Â£900 prize reward
! To claim call 09061701461. Claim code KL341. Valid 12 hours only."
[4] "Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with c
amera for Free! Call The Mobile Update Co FREE on 08002986030"
[5] "SIX chances to win CASH! From 100 to 20"
[6] "URGENT! You have won a 1 week FREE membership in our Ã,Â£100"
> tail(trainData_spam$text)
[1] "SMS SERVICES. for your inclusive text credits"
[2] "You are awarded a SiPix Digital Camera! call 09061221061 from landline. Delivery within 28day
s. T Cs Box177. M221BP. 2yr warranty. 150ppm. 16 . p pÃ,Â£3.99"
[3] "Want explicit SEX in 30 secs? Ring 02073162414 now! Costs 20p/min Gsex POBOX 2667 WC1N 3XX"
[4] "ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE MINS. INDIA CUST SERVs SED YES. L8ER GOT MEGA B
ILL. 3 DONT GIV A SHIT. BAILIFF DUE IN DAYS. I O Ã,Â£250 3 WANT Ã,Â£800"
[5] "REMINDER FROM O2: To get 2.50 pounds free call credit and details of great offers pls reply 2
this text with your valid name"
[6] "This is the 2nd time we have tried 2 contact u. U have won the Ã,Â£750 Pound prize. 2 claim i
s easy"
> trainData_spam <- NULL
> trainData_ham <- NULL
> corpus <- Corpus(VectorSource(trainData$text))
> print(corpus)
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
```

```
Content:  documents: 4459
> corpus[[1]]$content
[1] "Ok lar... Joking wif u oni..."
> corpus[[2]]$content
[1] "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to recei
ve entry question(std txt rate)T&C's apply 08452810075over18's"
> corpus[[50]]$content
[1] "Ha ha ha good joke. Girls are situation seekers."
> corpus[[100]]$content
[1] "ÃfÅ“ predict wat time ÃfÂ¼'ll finish buying?"
> corpus <- tm_map(corpus, content_transformer(tolower))
> corpus <- tm_map(corpus, removeNumbers)
> corpus <- tm_map(corpus, removeWords, stopwords())
> corpus <- tm_map(corpus, removePunctuation)
> corpus <- tm_map(corpus, stripWhitespace)
> corpus[[1]]$content
[1] "ok lar joking wif u oni"
> corpus[[2]]$content
[1] "free entry wkly comp win fa cup final tkts st may text fa receive entry questionstd txt ratet
cs apply s"
> corpus[[50]]$content
[1] "ha ha ha good joke girls situation seekers"
> corpus[[100]]$content
[1] "ãfå“ predict wat time ãfâll finish buying"
> pal1 <- brewer.pal(9,"YlGn")
> pal1 <- pal1[-(1:4)]
> pal2 <- brewer.pal(9,"Reds")
> pal2 <- pal2[-(1:4)]
> par(mfrow = c(1,2))
> wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors = pal1)
There were 49 warnings (use warnings() to see them)
> wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors = pal2)
> sms_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))
> print(sms_dtm)
<<DocumentTermMatrix (documents: 4459, terms: 6027)>>
Non-/sparse entries: 27792/26846601
Sparsity           : 100%
Maximal term length: 40
Weighting          : term frequency (tf)
> inspect(sms_dtm[1:10, 5:13])
<<DocumentTermMatrix (documents: 10, terms: 9)>>
Non-/sparse entries: 10/80
Sparsity           : 89%
Maximal term length: 11
Weighting          : term frequency (tf)
Sample             :
    Terms
Docs apply comp cup entry final free may questionstd ratetcs
  1      0    0   0     0     0    0   0            0       0
  10     0    0   0     0     0    0   0            0       0
  2      1    1   1     2     1    1   1            1       1
  3      0    0   0     0     0    0   0            0       0
  4      0    0   0     0     0    0   0            0       0
  5      0    0   0     0     0    0   0            0       0
  6      0    0   0     0     0    0   0            0       0
  7      0    0   0     0     0    0   0            0       0
  8      0    0   0     0     0    2   0            0       0
  9      0    0   0     0     0    0   0            0       0
> sms_features <- findFreqTerms(sms_dtm, 5)
> summary(sms_features)
   Length     Class      Mode
     1112 character character
> head(sms_features)
[1] "joking" "lar"    "wif"    "apply"  "comp"   "cup"
> sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
> print(sms_dtm_train)
<<DocumentTermMatrix (documents: 4459, terms: 1112)>>
Non-/sparse entries: 20536/4937872
Sparsity           : 100%
Maximal term length: 15
Weighting          : term frequency (tf)
> 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])
    Terms
Docs joking lar   wif   apply comp
   1 "Yes"  "Yes" "Yes" "No"  "No"
   2 "No"   "No"  "No"  "Yes" "Yes"
   3 "No"   "No"  "No"  "No"  "No"
   4 "No"   "No"  "No"  "No"  "No"
   5 "No"   "No"  "No"  "No"  "No"
   6 "No"   "No"  "No"  "No"  "No"
> sms_classifier <- naiveBayes(sms_dtm_train, trainData$type)
> sms_classifier[[2]][1:5]
$joking
              joking
trainData$type          No         Yes
          ham  0.998704999 0.001295001
          spam 1.000000000 0.000000000

$lar
              lar
trainData$type          No         Yes
          ham  0.992229992 0.007770008
          spam 1.000000000 0.000000000

$wif
              wif
trainData$type          No         Yes
          ham  0.994560995 0.005439005
          spam 1.000000000 0.000000000

$apply
```

```
              apply
trainData$type           No          Yes
          ham  0.9997409997 0.0002590003
          spam 0.9816053512 0.0183946488

$comp
              comp
trainData$type           No          Yes
          ham  0.9997409997 0.0002590003
          spam 0.9866220736 0.0133779264

> corpus <- Corpus(VectorSource(testData$text))
> corpus <- tm_map(corpus, content_transformer(tolower))
> corpus <- tm_map(corpus, removeNumbers)
> corpus <- tm_map(corpus, removeWords, stopwords())
> corpus <- tm_map(corpus, removePunctuation)
> corpus <- tm_map(corpus, stripWhitespace)
> sms_dtm_test <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
> print(sms_dtm_test)
<<DocumentTermMatrix (documents: 1114, terms: 1112)>>
Non-/sparse entries: 4930/1233838
Sparsity           : 100%
Maximal term length: 15
Weighting          : term frequency (tf)
> sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts)
> sms_dtm_test[1:10, 5:12]
    Terms
Docs comp  cup   entry final free  may   receive text
  1  "Yes" "Yes" "Yes" "No"  "No"  "No"  "No"    "No"
  2  "No"  "No"  "No"  "Yes" "Yes" "Yes" "Yes"   "No"
  3  "No"  "No"  "No"  "No"  "No"  "No"  "No"    "Yes"
  4  "No"  "No"  "No"  "No"  "No"  "No"  "No"    "No"
  5  "No"  "No"  "No"  "No"  "No"  "No"  "No"    "No"
  6  "No"  "No"  "No"  "No"  "No"  "No"  "No"    "No"
  7  "No"  "No"  "No"  "No"  "No"  "No"  "No"    "No"
  8  "No"  "No"  "No"  "No"  "No"  "No"  "No"    "No"
  9  "No"  "No"  "No"  "No"  "No"  "No"  "No"    "No"
  10 "No"  "No"  "No"  "No"  "No"  "No"  "No"    "No"
> sms_test_pred <- predict(sms_classifier, sms_dtm_test)
> table(testData$type, sms_test_pred)
      sms_test_pred
       ham spam
  ham  826  139
  spam 129   20
> ConfusionMatrix(sms_test_pred, testData$type)
      y_pred
y_true ham spam
  ham  826  139
  spam 129   20
> tablin<-table(testData$type, sms_test_pred)
> tablin
      sms_test_pred
       ham spam
  ham  826  139
  spam 129   20
> confusionMatrix(tablin)
Confusion Matrix and Statistics

      sms_test_pred
       ham spam
  ham  826  139
  spam 129   20

               Accuracy : 0.7594
                 95% CI : (0.7332, 0.7843)
    No Information Rate : 0.8573
    P-Value [Acc > NIR] : 1.0000

                  Kappa : -0.0095
 Mcnemar's Test P-Value : 0.5825

            Sensitivity : 0.8649
            Specificity : 0.1258
         Pos Pred Value : 0.8560
         Neg Pred Value : 0.1342
             Prevalence : 0.8573
         Detection Rate : 0.7415
   Detection Prevalence : 0.8662
      Balanced Accuracy : 0.4954

       'Positive' Class : ham
> Accuracy(sms_test_pred, testData$type)
[1] 0.7594255
> Precision(sms_test_pred, testData$type)
[1] 0.8649214
> Recall(sms_test_pred, testData$type)
[1] 0.8559585

> F1_Score(sms_test_pred, testData$type)
[1] 0.8604167
```

Visualization of word cloud in R:

3.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:

```
> require(caret)
> require(tm)
> require(wordcloud)
> require(e1071)
> require(MLmetrics)
> sampleData <-head(rawData, 500)
> str(sampleData)
'data.frame':  500 obs. of  2 variables:
 $ type: Factor w/ 2 levels "ham","spam": 1 2 1 1 2 1 1 2 2 1 ...
 $ text: chr  "Ok lar... Joking wif u oni..." "Free entry in 2 a wkly comp to win FA Cup final tkts 21st Ma
y 2005. Text FA to 87121 to receive entry question("| __truncated__ "U dun say so early hor... U c already
then say..." "Nah I don't think he goes to usf" ...
> colnames(sampleData) <- c("type", "text")
> sampleData$text <- iconv(sampleData$text, to = "utf-8")
> sampleData$type <- factor(sampleData$type)
> summary(sampleData)
   type         text
 ham :429   Length:500
 spam: 71   Class :character
            Mode  :character
> table(sampleData$type)

 ham spam
 429   71
> prop.table(table(sampleData$type)) * 100

 ham spam
85.8 14.2
> set.seed(0695)
> rep=100
> trainIndex <- createDataPartition(sampleData$type, p = .80,
+                                   list = FALSE,
+                                   times = 1)
> trainData <- sampleData[trainIndex,]
> testData <- sampleData[-trainIndex,]
> for (k in 1:rep) {
+ prop.table(table(trainData$type)) * 100
+ prop.table(table(testData$type)) * 100
+ trainData_ham <- trainData[trainData$type == "ham",]
+ head(trainData_ham$text)
+ tail(trainData_ham$text)
+ trainData_spam <- trainData[trainData$type == "spam",]
+ head(trainData_spam$text)
+ tail(trainData_spam$text)
+ trainData_spam <- NULL
+ trainData_ham <- NULL
+ corpus <- Corpus(VectorSource(trainData$text))
+ print(corpus)
+ }

> corpus[[1]]$content
[1] "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry
question(std txt rate)T&C's apply 08452810075over18's"
> corpus[[2]]$content
[1] "Nah I don't think he goes to usf"
> corpus[[50]]$content
[1] "Urgent UR awarded a complimentary trip to EuroDisinc Trav"
> corpus[[100]]$content
[1] "Dear"
> corpus <- tm_map(corpus, content_transformer(tolower))
> corpus <- tm_map(corpus, removeNumbers)
> corpus <- tm_map(corpus, removeWords, stopwords())
> corpus <- tm_map(corpus, removePunctuation)
> corpus <- tm_map(corpus, stripWhitespace)
> corpus[[1]]$content
[1] "free entry wkly comp win fa cup final tkts st may text fa receive entry questionstd txt ratetcs apply
s"
> corpus[[2]]$content
[1] "nah think goes usf"
> corpus[[50]]$content
[1] "urgent ur awarded complimentary trip eurodisinc trav"
> corpus[[100]]$content
[1] "dear"
```

```
> pal1 <- brewer.pal(9,"YlGn")
> pal1 <- pal1[-(1:4)]
> pal2 <- brewer.pal(9,"Reds")
> pal2 <- pal2[-(1:4)]
> par(mfrow = c(1,2))
> wordcloud(corpus[trainData$type == "ham"], min.freq = 40, random.order = FALSE, colors = pal1)
There were 50 or more warnings (use warnings() to see the first 50)
> wordcloud(corpus[trainData$type == "spam"], min.freq = 40, random.order = FALSE, colors = pal2)
There were 50 or more warnings (use warnings() to see the first 50)
> sms_dtm <- DocumentTermMatrix(corpus, control = list(global = c(2, Inf)))
> print(sms_dtm)
<<DocumentTermMatrix (documents: 401, terms: 1322)>>
Non-/sparse entries: 2560/527562
Sparsity           : 100%
Maximal term length: 21
Weighting          : term frequency (tf)
> inspect(sms_dtm[1:10, 5:13])
<<DocumentTermMatrix (documents: 10, terms: 9)>>
Non-/sparse entries: 11/79
Sparsity           : 88%
Maximal term length: 11
Weighting          : term frequency (tf)
Sample             :
    Terms
Docs final free may questionstd ratetcs receive text tkts txt
  1     1    1   1           1       1       1    1    1   1
  10    0    0   0           0       0       0    0    0   0
  2     0    0   0           0       0       0    0    0   0
  3     0    0   0           0       0       0    0    0   0
  4     0    0   0           0       0       0    0    0   0
  5     0    2   0           0       0       0    0    0   0
  6     0    0   0           0       0       0    0    0   0
  7     0    1   0           0       0       0    0    0   0
  8     0    0   0           0       0       0    0    0   0
  9     0    0   0           0       0       0    0    0   0
> sms_features <- findFreqTerms(sms_dtm, 5)
> summary(sms_features)
   Length     Class      Mode
      107 character character
> head(sms_features)
[1] "entry"   "free"    "receive" "text"    "txt"     "win"
> sms_dtm_train <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
> print(sms_dtm_train)
<<DocumentTermMatrix (documents: 401, terms: 107)>>
Non-/sparse entries: 876/42031
Sparsity           : 98%
Maximal term length: 10
Weighting          : term frequency (tf)
> 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])
    Terms
Docs entry free  receive text  txt
   1 "Yes" "Yes" "Yes"   "Yes" "Yes"
   2 "No"  "No"  "No"    "No"  "No"
   3 "No"  "No"  "No"    "No"  "No"
   4 "No"  "No"  "No"    "No"  "No"
   5 "No"  "Yes" "No"    "No"  "No"
   6 "No"  "No"  "No"    "No"  "No"
> sms_classifier <- naiveBayes(sms_dtm_train, trainData$type)
> sms_classifier[[2]][1:5]
$entry
             entry
trainData$type         No        Yes
          ham  1.00000000 0.00000000
          spam 0.92982456 0.07017544

$free
             free
trainData$type         No        Yes
          ham  0.99127907 0.00872093
          spam 0.71929825 0.28070175

$receive
             receive
trainData$type          No         Yes
          ham  0.997093023 0.002906977
          spam 0.912280702 0.087719298

$text
             text
trainData$type         No        Yes
          ham  0.98546512 0.01453488
          spam 0.87719298 0.12280702

$txt
             txt
trainData$type         No         Yes
          ham  0.994186047 0.005813953
          spam 0.859649123 0.140350877

> corpus <- Corpus(VectorSource(testData$text))
> corpus <- tm_map(corpus, content_transformer(tolower))
> corpus <- tm_map(corpus, removeNumbers)
> corpus <- tm_map(corpus, removeWords, stopwords())
> corpus <- tm_map(corpus, removePunctuation)
> corpus <- tm_map(corpus, stripWhitespace)
> sms_dtm_test <- DocumentTermMatrix(corpus, list(global = c(2, Inf), dictionary = sms_features))
> print(sms_dtm_test)
<<DocumentTermMatrix (documents: 99, terms: 107)>>
Non-/sparse entries: 176/10417
Sparsity           : 98%
Maximal term length: 10
Weighting          : term frequency (tf)
```

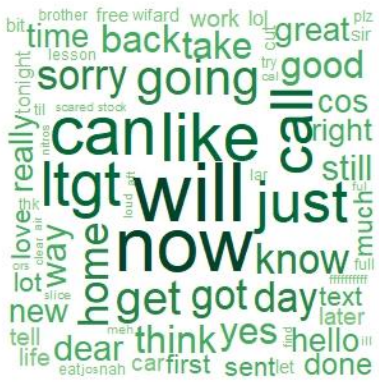
```
> sms_dtm_test <- apply(sms_dtm_test, MARGIN = 2, convert_counts)
> sms_dtm_test[1:10, 5:12]
    Terms
Docs txt   win   think back  hey   like  now   send
  1  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  2  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  3  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  4  "Yes" "Yes" "Yes" "No"  "No"  "No"  "No"  "No"
  5  "No"  "No"  "No"  "Yes" "Yes" "No"  "No"  "No"
  6  "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"
  7  "No"  "No"  "Yes" "No"  "No"  "No"  "Yes" "Yes"
  8  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  9  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
  10 "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"
> sms_test_pred <- predict(sms_classifier, sms_dtm_test)
> table(testData$type, sms_test_pred)
      sms_test_pred
       ham spam
  ham   78    7
  spam  10    4
> ConfusionMatrix(sms_test_pred, testData$type)
      y_pred
y_true ham spam
  ham   78    7
  spam  10    4
> confusionMatrix(tablin)
Confusion Matrix and Statistics

      sms_test_pred
       ham spam
  ham   78    7
  spam  10    4

               Accuracy : 0.8283
                 95% CI : (0.7394, 0.8967)
    No Information Rate : 0.8889
    P-Value [Acc > NIR] : 0.9757

                  Kappa : 0.2234
 Mcnemar's Test P-Value : 0.6276

            Sensitivity : 0.8864
            Specificity : 0.3636
         Pos Pred Value : 0.9176
         Neg Pred Value : 0.2857
             Prevalence : 0.8889
         Detection Rate : 0.7879
   Detection Prevalence : 0.8586
      Balanced Accuracy : 0.6250

       'Positive' Class : ham


> Accuracy(sms_test_pred, testData$type)
[1] 0.8282828
> Precision(sms_test_pred, testData$type)
[1] 0.8863636
> Recall(sms_test_pred, testData$type)
[1] 0.9176470

> F1_Score(sms_test_pred, testData$type)
[1] 0.9017341
```

Visualization of the word cloud subset (500 data subset):

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

Original Data set :

Subset:



Summary:

	Accuracy	Precision	Recall	KAPPA values	F Score
Whole data set	0.7594255	0.8649214	0.8559585	Kappa: -0.0095	0.86
500 data subset points and seed value set to 0695	0.8282828	0.8863636	0.9176470	Kappa: 0.2234	0.90

Analysis Summary:

1. The data set when considered completely and replicated for 100 times with training as 80% and testing as 20% of data has less 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, subset model is the best fit and good classifier model from above values.

2. **Kappa statistic**: The Kappa statistic (or value) is a metric that compares an Observed Accuracy with an Expected Accuracy (random chance). And taking to the consideration of the KAPPA values of both models, the more the kappa value the better the model fit. So, the from above statistics can be conclude that the 500 sample-data subset has more KAPPA value than the whole dataset. Thus, the data subset has better fit.

3. F **Score**: F score is the harmonic mean of precision and recall. It generally lies between 0 and 1 .Higher the value the better the model is . So, the data subset is a better model