Build a naive Bayes model on the data set for classifying the ham and spam

**Ans :**

**R Code :**

## Naive Bayes

########## Sms Spam Data Set #########

## Filtering Spam SMS messages

# read the sms data into the sms data frame

sms\_raw <- read.csv('D:\\Data Science\\Excelr\\R\\Datasets\\sms\_spam.csv', stringsAsFactors = FALSE)

# examine the structure of the sms data

str(sms\_raw)

# convert spam/ham to factor.

sms\_raw$type <- factor(sms\_raw$type)

# build a corpus using the text mining (tm) package

library(tm)

sms\_corpus <- Corpus(VectorSource(sms\_raw$text))

# clean up the corpus using tm\_map()

corpus\_clean <- tm\_map(sms\_corpus, tolower)

corpus\_clean <- tm\_map(corpus\_clean, removeNumbers)

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

corpus\_clean <- tm\_map(corpus\_clean, removePunctuation)

corpus\_clean <- tm\_map(corpus\_clean, stripWhitespace)

inspect(corpus\_clean)

# create a document-term sparse matrix

sms\_dtm <- DocumentTermMatrix(corpus\_clean)

# creating training and test datasets

sms\_raw\_train <- sms\_raw[1:4169, ]

sms\_raw\_test <- sms\_raw[4170:5559, ]

sms\_dtm\_train <- sms\_dtm[1:4169, ]

sms\_dtm\_test <- sms\_dtm[4170:5559, ]

sms\_corpus\_train <- corpus\_clean[1:4169]

sms\_corpus\_test <- corpus\_clean[4170:5559]

# check that the proportion of spam is similar

prop.table(table(sms\_raw\_train$type))

prop.table(table(sms\_raw\_test$type))

# word cloud visualization

library(wordcloud)

wordcloud(sms\_corpus\_train, min.freq = 30, random.order = FALSE)

# subset the training data into spam and ham groups

spam <- subset(sms\_raw\_train, type == "spam")

ham <- subset(sms\_raw\_train, type == "ham")

wordcloud(spam$text, max.words = 40, scale = c(3, 0.5),colors = 'blue')

wordcloud(ham$text, max.words = 40, scale = c(3, 0.5))

# indicator features for frequent words

sms\_dict<-findFreqTerms(sms\_dtm\_train, 3)

#sms\_dict <- Dictionary(findFreqTerms(sms\_dtm\_train, 5))

sms\_train <- DocumentTermMatrix(sms\_corpus\_train, list(dictionary = sms\_dict))

sms\_test <- DocumentTermMatrix(sms\_corpus\_test, list(dictionary = sms\_dict))

# convert counts to a factor

convert\_counts <- function(x) {

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

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

}

# apply() convert\_counts() to columns of train/test data

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

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

## Step 3: Training a model on the data ----

library(e1071)

sms\_classifier <- naiveBayes(sms\_train, sms\_raw\_train$type)

## Step 4: Evaluating model performance ----

sms\_test\_pred <- predict(sms\_classifier, sms\_test)

library(gmodels)

CrossTable(sms\_test\_pred, sms\_raw\_test$type,

prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,

dnn = c('predicted', 'actual'))

## Step 5: Improving model performance ----

sms\_classifier2 <- naiveBayes(sms\_train, sms\_raw\_train$type, laplace = 1)

sms\_test\_pred2 <- predict(sms\_classifier2, sms\_test)

CrossTable(sms\_test\_pred2, sms\_raw\_test$type,

prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,

dnn = c('predicted', 'actual'))

**Results :**

> CrossTable(sms\_test\_pred, sms\_raw\_test$type,

+ prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,

+ dnn = c('predicted', 'actual'))

Cell Contents

|-------------------------|

| N |

| N / Col Total |

|-------------------------|

Total Observations in Table: 1390

| actual

predicted | ham | spam | Row Total |

-------------|-----------|-----------|-----------|

ham | 1203 | 28 | 1231 |

| 0.997 | 0.153 | |

-------------|-----------|-----------|-----------|

spam | 4 | 155 | 159 |

| 0.003 | 0.847 | |

-------------|-----------|-----------|-----------|

Column Total | 1207 | 183 | 1390 |

| 0.868 | 0.132 | |

-------------|-----------|-----------|-----------|

> ## Step 5: Improving model performance ----

> sms\_classifier2 <- naiveBayes(sms\_train, sms\_raw\_train$type, laplace = 1)

> sms\_test\_pred2 <- predict(sms\_classifier2, sms\_test)

> CrossTable(sms\_test\_pred2, sms\_raw\_test$type,

+ prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,

+ dnn = c('predicted', 'actual'))

Cell Contents

|-------------------------|

| N |

| N / Col Total |

|-------------------------|

Total Observations in Table: 1390

| actual

predicted | ham | spam | Row Total |

-------------|-----------|-----------|-----------|

ham | 1205 | 30 | 1235 |

| 0.998 | 0.164 | |

-------------|-----------|-----------|-----------|

spam | 2 | 153 | 155 |

| 0.002 | 0.836 | |

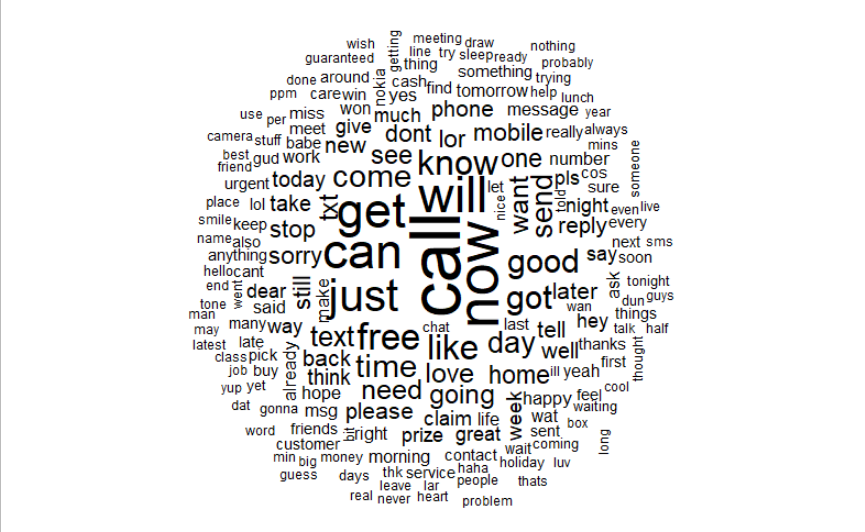
-------------|-----------|-----------|-----------|

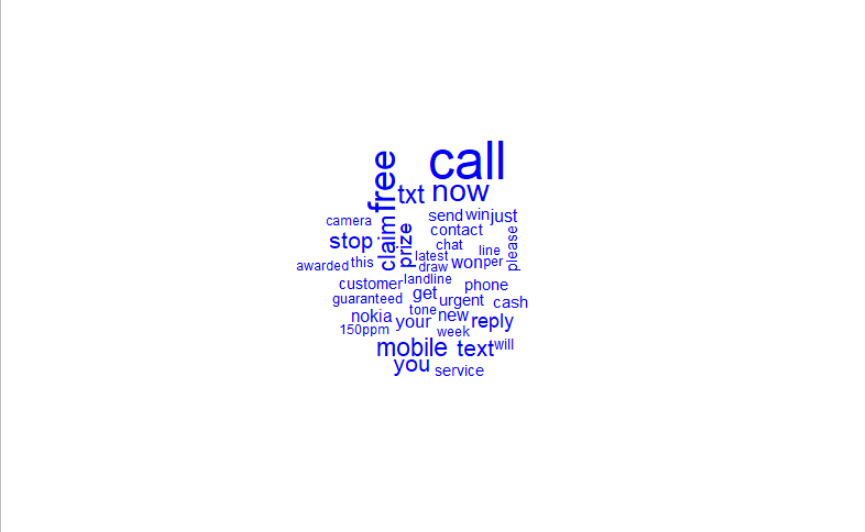
Column Total | 1207 | 183 | 1390 |

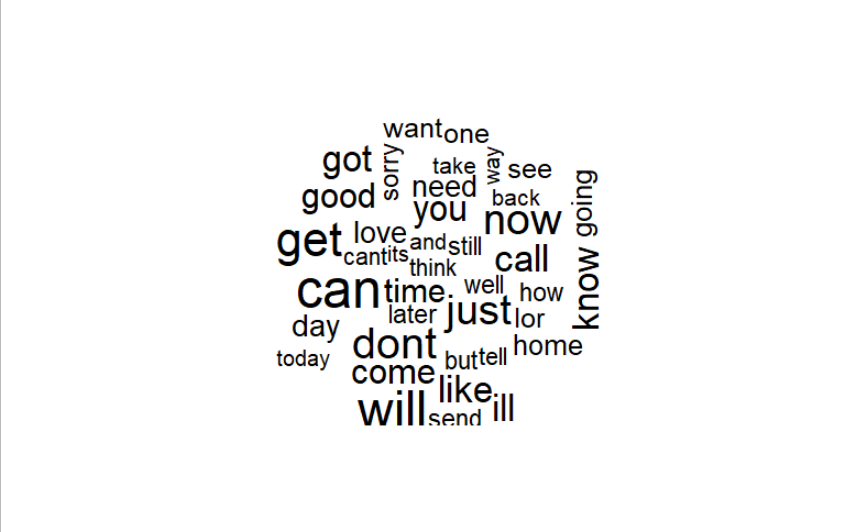
| 0.868 | 0.132 | |

-------------|-----------|-----------|-----------|

**Plots :**







**Inference :**

Getting little better results by adding laplace term in algorithm.