**Assignment – Naïve Bayes**

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

#NAIVE BAYES CLASSIFICATION ON SMS DATA

#loading the data

sms <- sms\_raw\_NB

str(sms) #examining the data structure

#convert spam ham to factor

sms$type <- factor(sms$type)

View(sms)

#Building corpus using library(tm)

library(tm)

sms\_corp <- Corpus(VectorSource(sms$text))

#data cleaning using tm\_map function

clean\_corp <- tm\_map(sms\_corp, tolower)

clean\_corp <- tm\_map(clean\_corp, removePunctuation)

clean\_corp <- tm\_map(clean\_corp, removeNumbers)

clean\_corp <- tm\_map(clean\_corp, removeWords, stopwords())

clean\_corp <- tm\_map(clean\_corp, stripWhitespace)

inspect(clean\_corp)

#creating document term matrix

corp\_dtm <- DocumentTermMatrix(clean\_corp)

#creating training and test data

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

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

dtm\_train <- corp\_dtm[1:4169,]

dtm\_test <- corp\_dtm[4170:5559,]

clean\_corp\_train <- clean\_corp[1:4169]

clean\_corp\_test <- clean\_corp[4170:5559]

#checking the proportion of spam is similar

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

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

#wordcloud visualisation

library(wordcloud)

windows()

wordcloud(clean\_corp\_train, min.freq = 30, random.order = FALSE)

#creating subsets of spam and ham in the training data

spam\_sms <- subset(sms\_train, type == 'spam')

ham\_sms <- subset(sms\_train, type == 'ham')

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

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

#frequent words

dict\_sms <- findFreqTerms(dtm\_train, 5)

train\_sms <- DocumentTermMatrix(clean\_corp\_train, list(dictionary = dict\_sms))

test\_sms <- DocumentTermMatrix(clean\_corp\_test, list(dictionary = dict\_sms))

#convert counts to factor

count\_convert <- function(x) {

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

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

}

#apply count\_convert() to columns of train/test data

train\_sms <- apply(train\_sms, MARGIN = 2, count\_convert)

test\_sms <- apply(test\_sms,MARGIN = 2, count\_convert)

#model training

library(e1071)

classify\_sms <- naiveBayes(train\_sms, sms\_train$type)

#evaluating model performance

sms\_pred <- predict(classify\_sms, test\_sms)

library(gmodels)

CrossTable(sms\_pred, sms\_test$type, prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE, dnn = c('predicted', 'actual'))

#improving model efficiency

classify2\_sms <- naiveBayes(train\_sms, sms\_train$type, laplace = 1)

sms\_pred2 <- predict(classify2\_sms, test\_sms)

1. Prepare a classification model using Naive Bayes for salary data

#SALARY DATA

#NAIVE BAYES CLASSIFICATION ON SALARY DATA

sd\_train <- SalaryData\_Train

sd\_test <- SalaryData\_Test

sal\_classifier <- naiveBayes(sd\_train, sd\_train$Salary)

sal\_pred <- predict(sal\_classifier, sd\_test)

sal\_pred

CrossTable(sal\_pred, sd\_test$Salary, prop.r = F, prop.t = F, prop.chisq = F, dnn = c('predicted', 'actual'))

#improving the model

sal\_classifier2 <- naiveBayes(sd\_train, sd\_train$Salary, laplace = 1)

sal\_pred2 <- predict(sal\_classifier2, sd\_test)

CrossTable(sal\_pred2, sd\_test$Salary, prop.r = F, prop.t = F, prop.chisq = F, dnn = c('predicted', 'actual'))