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

##Importing the data set ##

sms <- read.csv(file.choose())

str(sms)

## Two factor variable

## so we should convert the text variable into the character variable

sms$text <- as.character(sms$text)

str(sms)

##now variable are one in factor and one in character

prop.table(table(sms$type))

##ham=.8656233,spam=0.1343

library(tm)

sms\_corpus <- VCorpus(VectorSource(sms$text))## converting

## we can inspect the corpus

inspect(sms\_corpus[1:2])

##to view the data in corpus

as.character(sms\_corpus[[1]])

##[1] "Hope you are having a good week. Just checking in"

##cleaning the text file in the corpus

sms\_corpus\_clean <- tm\_map(sms\_corpus, content\_transformer(tolower))##coverting every letter into lower case

sms\_corpus\_clean <- tm\_map(sms\_corpus\_clean, removeNumbers)##removing the number from the text

sms\_corpus\_clean <- tm\_map(sms\_corpus\_clean, removeWords, stopwords())##removing stopwords

sms\_corpus\_clean <- tm\_map(sms\_corpus\_clean, removePunctuation)##removing punctuations

## for stemming purpose the snowball package is used

library(SnowballC)

sms\_corpus\_clean <- tm\_map(sms\_corpus\_clean, stemDocument)

sms\_corpus\_clean <- tm\_map(sms\_corpus\_clean, stripWhitespace)

##splitting the document in to words

sms\_dtm <- DocumentTermMatrix(sms\_corpus\_clean)

##splitting the data into testinting and training set

# creating training and test datasets

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

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

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

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

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

sms\_corpus\_test <- sms\_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))

# indicator features for frequent words

sms\_dict<-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))

sms\_dict

inspect(sms\_corpus\_train[1:100])

list(sms\_dict[1:100])

# 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 <- as.data.frame(apply(sms\_test, MARGIN = 2, convert\_counts))

View(sms\_train)

View(sms\_test)

## Training a model on the data ----

library(e1071)

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

## Evaluating model performance ----

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

class(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'))

##1355/1390=0.97

##improving the model

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'))

# 1357/1390=.976

mean(sms\_test\_pred2==sms\_raw\_test$type)

# 0.97625 %

SALARY DATA

import pandas as pd

import numpy as np

salary\_train = pd.read\_csv("SalaryData\_Train.csv")

salary\_test = pd.read\_csv("SalaryData\_Test.csv")

string\_columns=["workclass","education","maritalstatus","occupation","relationship","race","sex","native"]

from sklearn import preprocessing

number = preprocessing.LabelEncoder()

for i in string\_columns:

salary\_train[i] = number.fit\_transform(salary\_train[i])

salary\_test[i] = number.fit\_transform(salary\_test[i])

salary\_train.mean()

salary\_test.mean()

salary\_train.median()

salary\_test.median()

salary\_train.std()

salary\_train.skew()

salary\_train.hist()

##variables like native,race left skewed

## variables like capitalgain,caapitalloss,workclass are rightly skewed

## The variable like capital gain and capital loss are widely spread from the mean

colnames = salary\_train.columns

len(colnames[0:13])

trainX = salary\_train[colnames[0:13]]

trainY = salary\_train[colnames[13]]

testX = salary\_test[colnames[0:13]]

testY = salary\_test[colnames[13]]

from sklearn.naive\_bayes import GaussianNB

from sklearn.naive\_bayes import MultinomialNB

sgnb = GaussianNB()

smnb = MultinomialNB()

from sklearn.metrics import classification\_report,confusion\_matrix

## model building and evaluation of model

spred\_gnb = sgnb.fit(trainX,trainY).predict(testX)

confusion\_matrix(testY,spred\_gnb)

print ("Accuracy",(10759+1209)/(10759+601+2491+1209)) # 80%

spred\_mnb = smnb.fit(trainX,trainY).predict(testX)

confusion\_matrix(testY,spred\_mnb)

print("Accuracy",(10891+780)/(10891+780+2920+780)) # 75%