Spam\_Detection

# The following analysis aims to buid a machine learning model to detect spam SMS.  
# The data set is available in UCI Machine Learning repository.  
#The Data consist of 5574 SMS Messages(observation) with output(spam/ham)  
  
# Loading Libraries  
library(tm)

## Warning: package 'tm' was built under R version 4.0.4

## Loading required package: NLP

library(e1071)

## Warning: package 'e1071' was built under R version 4.0.5

library(caret)

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

## Loading required package: ggplot2

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':  
##   
## annotate

# Reading Data  
sms\_data = read.csv("C:/Users/sachi/Documents/R\_datafiles/SMSSpamCollection", sep="\t", header = TRUE, stringsAsFactors = FALSE)

## Warning in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, :  
## EOF within quoted string

colnames(sms\_data)= c("Class", "Messages")  
  
# Building the Corpus  
corpus = Corpus(VectorSource(sms\_data$Messages))  
  
# Preparing the data by applying transformations  
cleaning\_sms\_data = function(data)  
{  
 # convert Lowercase  
 data = tm\_map(data, tolower)  
 # Remove stop-words words like: pronouns, articles, prepositions etc. which do not add much meaning  
 data = tm\_map(data, removeWords, stopwords("english"))  
 # strip whitespace  
 data = tm\_map(data, stripWhitespace)   
 # Remove Punctuations  
 data = tm\_map(data, removePunctuation)  
   
}  
transformed\_data = cleaning\_sms\_data(corpus)

## Warning in tm\_map.SimpleCorpus(data, tolower): transformation drops documents

## Warning in tm\_map.SimpleCorpus(data, removeWords, stopwords("english")):  
## transformation drops documents

## Warning in tm\_map.SimpleCorpus(data, stripWhitespace): transformation drops  
## documents

## Warning in tm\_map.SimpleCorpus(data, removePunctuation): transformation drops  
## documents

# Building Document Term matrix  
dtm\_data = DocumentTermMatrix(transformed\_data)  
  
  
# Find frequency terms  
new\_data = findFreqTerms(dtm\_data,lowfreq = 10)  
  
sparse = removeSparseTerms(dtm\_data, 0.99)  
sparse

## <<DocumentTermMatrix (documents: 3183, terms: 150)>>  
## Non-/sparse entries: 10285/467165  
## Sparsity : 98%  
## Maximal term length: 10  
## Weighting : term frequency (tf)

sms\_sparse <- as.data.frame(data.matrix(sparse))  
# correcting the name of sparse data set  
colnames(sms\_sparse) = make.names(colnames(sms\_sparse))  
  
sms\_sparse$class = sms\_data$Class  
  
sms\_sparse$class=as.factor(sms\_sparse$class)  
  
  
# Splitting data into train and test  
set.seed(123)  
index = createDataPartition(sms\_sparse$class, p=0.8, list = FALSE)  
sms\_train = sms\_sparse[index,]  
sms\_test = sms\_sparse[-index,]  
  
# Building SVM Model  
svm\_model <- svm(class~., data=sms\_train, scale=FALSE, kernel='linear',type="C" )  
predict\_train=predict(svm\_model, sms\_train)  
pred\_linear <- predict(svm\_model, sms\_test)  
train\_accuracy = confusionMatrix(as.factor(predict\_train), as.factor(sms\_train$class))  
test\_accuracy <- confusionMatrix(as.factor(pred\_linear),as.factor(sms\_test$class))  
train\_accuracy

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction ham spam  
## ham 2185 55  
## spam 11 296  
##   
## Accuracy : 0.9741   
## 95% CI : (0.9671, 0.9799)  
## No Information Rate : 0.8622   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8849   
##   
## Mcnemar's Test P-Value : 1.204e-07   
##   
## Sensitivity : 0.9950   
## Specificity : 0.8433   
## Pos Pred Value : 0.9754   
## Neg Pred Value : 0.9642   
## Prevalence : 0.8622   
## Detection Rate : 0.8579   
## Detection Prevalence : 0.8795   
## Balanced Accuracy : 0.9191   
##   
## 'Positive' Class : ham   
##

test\_accuracy

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction ham spam  
## ham 543 23  
## spam 6 64  
##   
## Accuracy : 0.9544   
## 95% CI : (0.9352, 0.9693)  
## No Information Rate : 0.8632   
## P-Value [Acc > NIR] : 2.47e-14   
##   
## Kappa : 0.7896   
##   
## Mcnemar's Test P-Value : 0.002967   
##   
## Sensitivity : 0.9891   
## Specificity : 0.7356   
## Pos Pred Value : 0.9594   
## Neg Pred Value : 0.9143   
## Prevalence : 0.8632   
## Detection Rate : 0.8538   
## Detection Prevalence : 0.8899   
## Balanced Accuracy : 0.8624   
##   
## 'Positive' Class : ham   
##