#Load necessary packages

library(twitteR)

library(ROAuth)

library(tidyverse)

library(purrrlyr)

library(text2vec)

library(caret)

library(glmnet)

library(ggrepel)

library(readr)

#Library for database connection

library(RODBC)

#Convert symbols

doConversion <- function(x) iconv(x, "latin1", "ASCII", "")

#Load tweets with column names

readTwitterData <- read\_csv('TrainingData.csv', col\_names = c('opinion', 'userno', 'datetime', 'alias', 'diction')) %>%

#Convert symbols and switch class values

dmap\_at('diction', doConversion) %>% mutate(opinion = ifelse(opinion == 0, 0, 1))

#Enable repeatable results, split, and train data set

set.seed(42)

startTraining <- createDataPartition(readTwitterData$opinion, p = 0.8, list = FALSE, times = 1)

trainTwitterFeeds <- readTwitterData[startTraining, ]

checkTwitterDatum <- readTwitterData[-startTraining, ]

#First pass process, clean up, and then tokenize

startWork <- tolower

simpleTokenizer <- word\_tokenizer

trainSet <- itoken(trainTwitterFeeds$diction, preprocessor = startWork, tokenizer = simpleTokenizer, empnum = trainTwitterFeeds$userno, progressbar = TRUE)

checkSet <- itoken(checkTwitterDatum$diction, preprocessor = startWork, tokenizer = simpleTokenizer, empnum = checkTwitterDatum$userno, progressbar = TRUE)

#Build dictionary of terms and DTM (Document Term Matrix)

dictionary <- create\_vocabulary(trainSet)

dictionaryVector <- vocab\_vectorizer(dictionary)

docTermMatrixTraining <- create\_dtm(trainSet, dictionaryVector)

docTermMatrixChecking <- create\_dtm(checkSet, dictionaryVector)

#tf-idf modelling

modelTfidf <- TfIdf$new()

#Align the data model to the set that was trained. Change the trained data with the model.

docTermMatrixTraining\_modelTfidf<- fit\_transform(docTermMatrixTraining, modelTfidf)

docTermMatrixChecking\_modelTfidf<- fit\_transform(docTermMatrixChecking, modelTfidf)

#Train the model. Check under the ROC curve. Use nfolds = 5 for faster training. Set maxit = 1e3 to keep iterations low for faster training.

modelTrainingWithGlmnet <- cv.glmnet(x = docTermMatrixTraining\_modelTfidf, y = trainTwitterFeeds[['opinion']], family = 'binomial', alpha = 1, type.measure = "auc", nfolds = 5, thresh = 1e-3, maxit = 1e3)

#Plot model and make prediction

plot(modelTrainingWithGlmnet)

makePrediction <- predict(modelTrainingWithGlmnet, docTermMatrixChecking\_modelTfidf, type = 'response')[ ,1]

auc(as.numeric(checkTwitterDatum$opinion), makePrediction)

#Reduce the time to rerun the model by saving it

saveRDS(modelTrainingWithGlmnet, 'modelTrainingWithGlmnet.RDS')

#Create database connection. Get data from the CheckText database in the DataToAnalyze table.

con <- odbcDriverConnect(connection="Driver={SQL Server Native Client 11.0};server=localhost;database=CheckText;trusted\_connection=yes;")

DataToAnalyzeFrDb <- sqlFetch(con,"DataToAnalyze",colnames = FALSE)

DataFromDatabase <- DataToAnalyzeFrDb %>% dmap\_at('text', doConversion)

#Tokenize and start processing

tweetDiction <- itoken(DataFromDatabase$text, preprocessor = startWork, tokenizer = simpleTokenizer, id = DataFromDatabase$userno, progressbar = TRUE)

tweetsDocTermMatrix <- create\_dtm(tweetDiction, dictionaryVector)

#use tf-idf to transform the data

tweetsDocTermMatrix\_modelTfidf<- fit\_transform(tweetsDocTermMatrix, modelTfidf)

#The model being classified is loaded below. Try to predict what is the probability that the statements will be positive. Append the column with the rates to the data from the database

modelTrainingWithGlmnet <- readRDS('modelTrainingWithGlmnet.RDS')

probabilityOfPositivity <- predict(modelTrainingWithGlmnet, tweetsDocTermMatrix\_modelTfidf, type = 'response')[ ,1]

DataFromDatabase$opinion <- probabilityOfPositivity

#Create graph with labels at the distribution points

attach(DataFromDatabase)

plot(ResponseDate, opinion, main="Probability of having positive Words in an opinion", xlab="Opinion Posted Date", ylab="Opinion Scores", col="red", pch=8)

abline(h =0.5, col="blue")

text(ResponseDate, opinion, rownames(DataFromDatabase), pos= 4 )