**#Sentiment analysis using the twitter data**

**#classifying responsive e-mails of Enron Company for investigation**

# Unit 5 - Twitter

# VIDEO 5

# Read in the data

tweets = read.csv("tweets.csv", stringsAsFactors=FALSE)

str(tweets)

# Create dependent variable

tweets$Negative = as.factor(tweets$Avg <= -1)

table(tweets$Negative)

# Install new packages

install.packages("tm")

library(tm)

# If it doesn’t work install “slam” and “NLP” first using R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"

install.packages("SnowballC")

library(SnowballC)

# Create corpus

corpus = Corpus(VectorSource(tweets$Tweet))

# Look at corpus

corpus

corpus[[1]]

# Convert to lower-case

corpus = tm\_map(corpus, tolower)

corpus[[1]]

# IMPORTANT NOTE: If you are using the latest version of the tm package, you will need to run the following line before continuing (it converts corpus to a Plain Text Document). This is a recent change having to do with the tolower #function that occurred after this video was recorded.

#required for R version 3.3.1

corpus = tm\_map(corpus, PlainTextDocument)

# Remove punctuation

corpus = tm\_map(corpus, removePunctuation)

corpus[[1]]

# Look at stop words

stopwords("english")[1:10]

# Remove stopwords and apple

corpus = tm\_map(corpus, removeWords, c("apple", stopwords("english")))

corpus[[1]]

# Stem document

corpus = tm\_map(corpus, stemDocument)

corpus[[1]]

# Video 6

# Create matrix

frequencies = DocumentTermMatrix(corpus)

frequencies

# Look at matrix

inspect(frequencies[1000:1005,505:515])

# Check for sparsity

findFreqTerms(frequencies, lowfreq=20)

# Remove sparse terms

sparse = removeSparseTerms(frequencies, 0.995)

sparse

# Convert to a data frame

tweetsSparse = as.data.frame(as.matrix(sparse))

# Make all variable names R-friendly

colnames(tweetsSparse) = make.names(colnames(tweetsSparse))

# Add dependent variable

tweetsSparse$Negative = tweets$Negative

# Split the data

library(caTools)

set.seed(123)

split = sample.split(tweetsSparse$Negative, SplitRatio = 0.7)

trainSparse = subset(tweetsSparse, split==TRUE)

testSparse = subset(tweetsSparse, split==FALSE)

# Video 7

# Build a CART model

library(rpart)

library(rpart.plot)

tweetCART = rpart(Negative ~ ., data=trainSparse, method="class")

prp(tweetCART)

# Evaluate the performance of the model

predictCART = predict(tweetCART, newdata=testSparse, type="class")

table(testSparse$Negative, predictCART)

# Compute accuracy

(294+18)/(294+6+37+18)

# Baseline accuracy

table(testSparse$Negative)

300/(300+55)

# Random forest model

library(randomForest)

set.seed(123)

tweetRF = randomForest(Negative ~ ., data=trainSparse)

# Make predictions:

predictRF = predict(tweetRF, newdata=testSparse)

table(testSparse$Negative, predictRF)

# Accuracy:

(293+21)/(293+7+34+21)

##################################\*

#classifying responsive e-mails of Enron company for investigation

##################################\*

# Unit 5 - Recitation

# Video 2

# Load the dataset

emails = read.csv("energy\_bids.csv", stringsAsFactors=FALSE)

str(emails)

# Look at emails

emails$email[1]

emails$responsive[1]

emails$email[2]

emails$responsive[2]

# Responsive emails

table(emails$responsive)

# Video 3

# Load tm package

library(tm)

# Create corpus

corpus = Corpus(VectorSource(emails$email))

corpus[[1]]

# Pre-process data

corpus = tm\_map(corpus, tolower)

# IMPORTANT NOTE: If you are using the latest version of the tm package, you will need to run the following line before continuing (it converts corpus to a Plain Text Document). This is a recent change having to do with the tolower function that occurred after this video was recorded.

corpus = tm\_map(corpus, PlainTextDocument)

corpus = tm\_map(corpus, removePunctuation)

corpus = tm\_map(corpus, removeWords, stopwords("english"))

corpus = tm\_map(corpus, stemDocument)

# Look at first email

corpus[[1]]

# Video 4

# Create matrix

dtm = DocumentTermMatrix(corpus)

dtm

# Remove sparse terms

dtm = removeSparseTerms(dtm, 0.97)

dtm

# Create data frame

labeledTerms = as.data.frame(as.matrix(dtm))

# Add in the outcome variable

labeledTerms$responsive = emails$responsive

str(labeledTerms)

# Video 5

# Split the data

library(caTools)

set.seed(144)

spl = sample.split(labeledTerms$responsive, 0.7)

train = subset(labeledTerms, spl == TRUE)

test = subset(labeledTerms, spl == FALSE)

# Build a CART model

library(rpart)

library(rpart.plot)

emailCART = rpart(responsive~., data=train, method="class")

prp(emailCART)

# Video 6

# Make predictions on the test set

pred = predict(emailCART, newdata=test)

pred[1:10,]

pred.prob = pred[,2]

# Compute accuracy

table(test$responsive, pred.prob >= 0.5)

(195+25)/(195+25+17+20)

# Baseline model accuracy

table(test$responsive)

215/(215+42)

# Video 7

# ROC curve

library(ROCR)

predROCR = prediction(pred.prob, test$responsive)

perfROCR = performance(predROCR, "tpr", "fpr")

plot(perfROCR, colorize=TRUE)

# Compute AUC

performance(predROCR, "auc")@y.values