> #Read the DataSet

> tweets<-read.csv("tweets.csv",stringsAsFactors = FALSE)

>#Build Corpus

> library(tm)

> tweets.corpus<-Corpus(VectorSource(tweets$text))

> #Data Transformation

> tweets.corpus<-tm\_map(tweets.corpus,tolower)

Warning message:

In tm\_map.SimpleCorpus(tweets.corpus, tolower) :

transformation drops documents

> tweets.corpus<-tm\_map(tweets.corpus,stripWhitespace)

Warning message:

In tm\_map.SimpleCorpus(tweets.corpus, stripWhitespace) :

transformation drops documents

> tweets.corpus<-tm\_map(tweets.corpus,removePunctuation)

Warning message:

In tm\_map.SimpleCorpus(tweets.corpus, removePunctuation) :

transformation drops documents

> tweets.corpus<-tm\_map(tweets.corpus,removeNumbers)

Warning message:

In tm\_map.SimpleCorpus(tweets.corpus, removeNumbers) :

transformation drops documents

> mystopwords<-c(stopwords('english'),'available')

> tweets.corpus<-tm\_map(tweets.corpus,removeWords,mystopwords)

Warning message:

In tm\_map.SimpleCorpus(tweets.corpus, removeWords, mystopwords) :

transformation drops documents

> pos=scan('positive-words.txt',what='character',comment.char = ';')

> neg=scan('negative-words.txt',what='character',comment.char = ';')

> #Jeffreybrean Algorithm to map sentiments

> score.sentiment = function(sentences, pos.words, neg.words, .progress='none')

+ {

+ require(plyr)

+ require(stringr)

+

+ #we got a vector of sentences. plyr will handle a list

+ #or a vector as an "l" for us

+ #we want a simple array of scores back, so we use

+ #"l" + "a" + "ply" = "laply":

+ scores = laply(sentences, function(sentence, pos.words, neg.words) {

+

+ #clean up sentences with R's regex-driven global substitute, gsub():

+ sentence = gsub('[[:punct:]]', '', sentence) #removes punctuations

+ sentence = gsub('[[:cntrl:]]', '', sentence) #removes control characters

+ sentence = gsub('\\d+', '', sentence) #removes digits

+

+ #and convert to lower case:

+ sentence = tolower(sentence)

+

+ #split sentences into words. str\_split is in the stringr package

+ word.list = str\_split(sentence, '\\s+')

+

+ #sometimes a list() is one level of hierarchy too much

+ words = unlist(word.list)

+

+ #compare our words to the dictionaries of positive & negative terms

+ pos.matches = match(words, pos.words)

+ neg.matches = match(words, neg.words)

+

+ #match() returns the position of the matched term or NA

+ #we just want a TRUE/FALSE:

+ pos.matches = !is.na(pos.matches)

+ neg.matches = !is.na(neg.matches)

+

+ #and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():

+ score = sum(pos.matches) - sum(neg.matches)

+

+ return(score)

+ }, pos.words, neg.words, .progress=.progress )

+

+ scores.df = data.frame(score=scores, text=sentences)

+ return(scores.df)

+ }

> analysis<-score.sentiment(tweets.text, pos, neg, .progress="text")

|======================================================================================| 100%

> View(analysis)

> table(analysis$score)

-5 -4 -3 -2 -1 0 1 2 3 4 5 6

1 4 27 146 556 1306 782 230 59 20 3 1

> hist(analysis$score)

> analysis$text<-as.character(analysis$text)

> analysis$sentiment<-ifelse(analysis$score>0,"positive",

+ ifelse(analysis$score<0,"negative","neutral"))

> table(analysis$sentiment)

negative neutral positive

734 1306 1095

> #Cleaning the data again

> analysis$text = gsub('[[:punct:]]', '', analysis$text)

> str(analysis)

'data.frame': 3135 obs. of 3 variables:

$ score : int -2 -1 -1 -1 -1 0 0 -3 1 -2 ...

$ text : chr "Bad news was Dad has cancer and is dying Good news new business started and I am now a life coach practising"| \_\_truncated\_\_ "im lonely keep me company 22 female new york" "Sad about Kutner being killed off my fav show House " "is going to priceline city tomorrow but lost her must haves list " ...

$ sentiment: chr "negative" "negative" "negative" "negative" ...

> head(analysis,5)

Score

1 -2

2 -1

3 -1

4 -1

5 -1

text

1 Bad news was Dad has cancer and is dying Good news new business started and I am now a life coach practising holistic weight management

2 im lonely keep me company 22 female new york

3 Sad about Kutner being killed off my fav show House

4 is going to priceline city tomorrow but lost her must haves list

5 Difficulties with GTalk Closing the Division for the day Later everyone

sentiment

1 negative

2 negative

3 negative

4 negative

5 negative

**Required Packages**

> install.packages("caret")

Error in install.packages : Updating loaded packages

Restarting R session...

> library(caret)

Error in lapply(x[ind], paste, collapse = ".") :

reached elapsed time limit

Loading required package: lattice

Loading required package: ggplot2

Warning message:

package ‘caret’ was built under R version 3.5.3

> install.packages("e1071")

Installing package into ‘C:/Users/VARUN/Documents/R/win-library/3.5’

(as ‘lib’ is unspecified)

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/e1071\_1.7-1.zip'

Content type 'application/zip' length 1016018 bytes (992 KB)

downloaded 992 KB

package ‘e1071’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\VARUN\AppData\Local\Temp\RtmpSW8wiF\downloaded\_packages

> library(e1071)

Warning message:

package ‘e1071’ was built under R version 3.5.3

> train\_tweets$type="train"

> test\_tweets$type="test"

> tweets<-rbind(train\_tweets,test\_tweets)

> head(tweets)

**Building Corpus**

> #Build Corpus

> library(tm)

> tweets.corpus<-Corpus(VectorSource(tweets$text))

**Data Transformation**

> tweets.corpus<-tm\_map(tweets.corpus,tolower)

> tweets.corpus<-tm\_map(tweets.corpus,stripWhitespace)

> tweets.corpus<-tm\_map(tweets.corpus,removePunctuation)

> tweets.corpus<-tm\_map(tweets.corpus,removeNumbers)

> mystopwords<-c(stopwords('english'),'available')

> tweets.corpus<-tm\_map(tweets.corpus,removeWords,mystopwords)

**Document Term Matrix**

> #Document Term Matrix

> tweets\_dtm<-DocumentTermMatrix(tweets.corpus)

> tweets\_dtm

<<DocumentTermMatrix (documents: 3135, terms: 8014)>>

Non-/sparse entries: 22928/25100962

Sparsity : 100%

Error in nchar(Terms(x), type = "chars") :

invalid multibyte string, element 674

> #Remove Sparse Terms

> tweets\_dtm

<<DocumentTermMatrix (documents: 3135, terms: 8014)>>

Non-/sparse entries: 22928/25100962

Sparsity : 100%

Error in nchar(Terms(x), type = "chars") :

invalid multibyte string, element 674

> tweets\_dtm<-removeSparseTerms(tweets\_dtm,0.97)

> tweets\_dtm

<<DocumentTermMatrix (documents: 3135, terms: 18)>>

Non-/sparse entries: 2356/54074

Sparsity : 96%

Maximal term length: 5

Weighting : term frequency (tf)

> tweets\_dtm\_matrix<-as.matrix(tweets\_dtm)

>View(tweets\_dtm\_matrix)

> dtm<-as.data.frame(tweets\_dtm\_matrix)

> dtm$class<-analysis$sentiment

> head(dtm)

going will work day just now still got today time dont cant good like love see

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

5 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

6 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

get back class

1 0 0 negative

2 0 0 negative

3 0 0 negative

4 0 0 negative

5 0 0 negative

6 0 0 neutral

**Sampling Data**

> #Train and Test Dataset

> sampling<-sort(sample(nrow(analysis),nrow(analysis)\*0.7))

> train= dtm[sampling,]

> test = dtm[-sampling,]

> dim(train)

[1] 2194 19

> dim(test)

[1] 941 19

> #To Check Distribution of Sentiment in Train and Test

> prop.table(table(train$class))

negative neutral positive

0.2324521 0.4111212 0.3564266

> prop.table(table(test$class))

negative neutral positive

0.2380446 0.4293305 0.3326249

> library(caret)

> library(rpart)

**Model Building**

> which(names(train)=="class")

[1] 19

> #(1) Classification Tree

> tree<-train(train[,-19],as.factor(train$class),method="rpart")

> predTree=predict(tree$finalModel,newdata=test,type="class")

> #Accuracy

> sum(diag(table(test$class,predTree)))/nrow(test)

[1] 0.4293305

> #(2) Bagged Cart

> treebag<-train(train[,-19],as.factor(train$class),method="treebag")

> predictTreebag<-predict(treebag$finalModel,newdata=test,type="class")

> sum(diag(table(test$class,predictTreebag)))/nrow(test)

[1] 0.3772582

> #(3) Random Forest

> rf<-train(train[,-19],as.factor(train$class),method="rf")

> predictRf<-predict(rf$finalModel,newdata=test,type="class")

> sum(diag(table(test$class,predictRf)))/nrow(test)

[1] 0.41339

Out of the Three Models Classification Tree has the highest accuracy: 0.4299