**Capstone Project – Kinjal Purohit (Dave)**

**# Installation of necessary libraries**

install.packages("readtext")

install.packages("tm")

install.packages("wordcloud")

install.packages("RWeka")

install.packages("kernlab")

install.packages("party")

install.packages("FSelector")

install.packages("rJava",type = "source")

install.packages("syuzhet")

install.packages("tidytext")

**#calling all the libraries**

library(readtext)

library(wordcloud)

library(RWeka)

library(kernlab)

library(SnowballC)

library(tm)

library(syuzhet)

library(caret)

library(rminer)

library(party)

library(FSelector)

library(tidytext)

library(RSentiment)

library(sentimentr)

\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Phase 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**# Loading of files with for loop**

# For training Pos

txt.files <- list.files("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/train/pos/", full.names = TRUE)

for(i in 1:(length(txt.files)))

{

if(i==1){

my.data <- data.frame(NULL)

}

cur.file <- readtext(file = txt.files[i])

my.data <- rbind(my.data, cur.file)

}

# for training neg

txt.filestrainneg <- list.files("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/train/neg", full.names = TRUE)

for(i in 1:(length(txt.filestrainneg)))

{

if(i==1){

my.datatrainneg <- data.frame(NULL)

}

cur.file1 <- readtext(file = txt.filestrainneg[i])

my.datatrainneg <- rbind(my.datatrainneg, cur.file1)

}

# for the testing pos

txt.filestestpos <- list.files("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/test/pos", full.names = TRUE)

for(i in 1:(length(txt.filestestpos)))

{

if(i==1){

my.datatestpos <- data.frame(NULL)

}

cur.file2 <- readtext(file = txt.filestestpos[i])

my.datatestpos <- rbind(my.datatestpos, cur.file2)

}

# for the testing neg

txt.filestestneg <- list.files("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/test/neg", full.names = TRUE)

for(i in 1:(length(txt.filestestneg)))

{

if(i==1){

my.datatestneg <- data.frame(NULL)

}

cur.file3 <- readtext(file = txt.filestestneg[i])

my.datatestneg <- rbind(my.datatestneg, cur.file3)

}

**# Alternate method of loading files**

#Reading the text file

readtrainpos <-readtext::readtext("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/train/pos")

readtrainneg <-readtext::readtext("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/train/neg")

readtestpos <-readtext::readtext("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/test/pos")

readtestneg <-readtext::readtext("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/test/neg")

# combing the test and train datasets

train\_df <- rbind(readtrainpos,readtestneg)

test\_df <- rbind(readtestpos,readtestneg)

# adding the labels

colnames(train\_df) <- c("id", "text")

colnames(test\_df) <- c("id", "text")

# removing the id column

train\_df <- train\_df$text

test\_df <- test\_df$text

# converting into dataframe

train\_data <- as.data.frame(train\_df)

test\_data <- as.data.frame(test\_df)

#checking for complete cases

which(!complete.cases(train\_data))

**# creating corpus object**

doc1 <- Corpus(VectorSource(train\_df))

mycorpus <- tm\_map(doc1, content\_transformer(tolower))

mycorpus <- tm\_map(mycorpus, content\_transformer(removePunctuation))

mycorpus <- tm\_map(mycorpus, content\_transformer(stripWhitespace))

mycorpus <- tm\_map(mycorpus, content\_transformer(removeNumbers))

mycorpus <- tm\_map(mycorpus, removeWords, stopwords("english"))

**# stemming and Lemmatization**

mycorpuscopy <- mycorpus

mycorpus <- tm\_map(mycorpus, stemDocument)

for(i in 1:2){

cat(paste("[[", i, " ]]", sep = ""))

writeLines(strwrap(mycorpus[[i]],width=73))

}

inspect(mycorpus[1:2])

library(textstem)

mycorpus <- tm\_map(mycorpus, stemCompletion, dictionary = mycorpuscopy)

lemmatize\_words(mycorpus)

inspect(mycorpus[1:2])

lemmatize\_strings(mycorpus)

**# Feature Selection – PCA**

# PCA

tdm <- DocumentTermMatrix(mycorpus)

tdm\_tfxid <- weightTfIdf(tdm)

tdm\_tfxid <- removeSparseTerms(tdm\_tfxid, 0.99)

tdm.pca <- prcomp(tdm\_tfxid, center = TRUE, scale. = TRUE)

#tdm.pca<- preProcess(as.matrix(tdm\_tfxid), method = c("pca"), thresh = 0.95)

plot(tdm.pca, type = 'l') # from the plot we can conclude that is there no significane change after PCA7. Hence we will take till PCA7

# Prediction with PCA7

pcapredict <- predict(tdm.pca, newdata = tdm\_tfxid)

adtm.m <- as.matrix(pcapredict)

adtm.df <- as.data.frame(adtm.m)

pcaprdictexp <- data.frame(adtm.df,pcapredict)

\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Phase 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**#Count of frequency**

tdm <- TermDocumentMatrix(mycorpus, control = list(wordLengths = c(1, Inf)))

tdm

freq.terms <- findFreqTerms(tdm, lowfreq = 50)

termfreq <- rowSums(as.matrix(removeSparseTerms(tdm, 0.99)))

termfreq <- subset(termfreq, termfreq >= 15)

df <- data.frame(term = names(termfreq), freq = termfreq)

**# finding the min and max freq**

findFreqTerms(tdm, lowfreq = df$freq)

max(df$freq)

min(df$freq)

length(df$freq)

tail(table(df$term),256)

freq <- sort(termfreq, decreasing = TRUE)

df1 <- data.frame(term = names(freq), freq=freq)

head(df1)

**# plotting the frequency**

library(ggplot2)

ggplot(df1, aes(x = reorder(term,-freq), y = freq))+

geom\_bar(stat = 'identity', position = "dodge") +

theme(axis.text.x = element\_text(angle = 45, hjust = 1, vjust = .5)) +

scale\_x\_discrete()

**# finding association with movie**

findAssocs(tdm, "movie", .25)

findAssocs(tdm, "good", corlimit = 0.10)

**#ploting word frequncies**

barplot(df[1:10,]$freq, las = 2, names.arg = df[1:10,]$term,

col = "lightblue", main = "Most frequent words",

ylab = "word Frequencies")

**# world cloud**

set.seed(1234)

pal <- brewer.pal(8, "Dark2")

wordcloud(words = df$term, freq = df$freq, min.freq = 500, max.words = Inf, width = 1000, height = 1000,

random.order = FALSE, rot.per = 0.35, colors = pal)

**# Reading the unsupdata and applying clustering on it.**

**# reading the unlabeled data**

unsup\_df <- readtext("H:/Rlatest/Kinjal R/Capstone project/aclImdb\_v1/aclImdb/train/unsup")

unsup\_dfchar <- unsup\_df$text

unsup\_df <- as.data.frame(unsup\_df$text, stringsAsFactors=FALSE)

**# cleaning the unsup data**

doc2 <- Corpus(VectorSource(unsup\_dfchar))

unsupcorpus <- tm\_map(doc2, content\_transformer(tolower))

unsupcorpus <- tm\_map(unsupcorpus, content\_transformer(removePunctuation))

unsupcorpus <- tm\_map(unsupcorpus, content\_transformer(stripWhitespace))

unsupcorpus <- tm\_map(unsupcorpus, content\_transformer(removeNumbers))

unsupcorpus <- tm\_map(unsupcorpus, removeWords, stopwords("english"))

**#stemming**

unsupcorpuscopy <- unsupcorpus

unsupcorpus <- tm\_map(unsupcorpus, stemDocument)

for(i in unsupcorpus){

cat(paste("[[", i, "]]", sep = ""))

writeLines(strwrap(unsupcorpus[[i]],width=73))

}

#inspect(mycorpus[1:2])

#mycorpus <- tm\_map(mycorpus, stemCompletion, dictionary = mycorpuscopy)

??stemCompletion

# Stem completion

stemCompletion\_mod <- function(x,unsupcorpuscopy) {

PlainTextDocument(stripWhitespace(paste(stemCompletion(unlist(strsplit(as.character(x)," ")), dictionary = unsupcorpuscopy, type = "shortest"), sep = "", collapse = " ")))

}

unsupcorpus <- sapply(unsupcorpus, stemCompletion\_mod)

**# making unsup tdm**

unsuptdm <- TermDocumentMatrix(unsupcorpus, control = list(wordLengths = c(1, Inf)))

unsuptdm <- removeSparseTerms(unsuptdm,0.99)

unsupmat <- rowSums(as.matrix(unsuptdm))

unsupmtdm\_df <- data.frame(term = names(unsupmat), freq = unsupmat)

**# getting pos and neg words from unsup data**

unsupresult <- score.sentiment(unsupmtdm\_df$term,pos.words,neg.words)

max(unsupresult$score)

min(unsupresult$score)

unsupresult$pos <- as.numeric(unsupresult$score >=1)

unsupresult$neg <- as.numeric(unsupresult$score <=-1)

library(sqldf)

unsupposwords <- sqldf("select text from unsupresult where pos ==1")

unsupnegwords <- sqldf("select text from unsupresult where neg ==1")

**# k means clustering – Please note clustering below is done on random sample of 500 due to memory issue.**

# combining unsup pos and neg words into one and making the tdm on it

unsuptdm1 <- TermDocumentMatrix(unsupcorpus,control = list(wordLengths = c(1, Inf)))

unsuptdm2 <- removeSparseTerms(unsuptdm1, 0.99)

m2 <- as.matrix(unsuptdm2)

distmatrix <- dist(scale(m2))

fit <- hclust(distmatrix, method = 'ward')

plot(fit)

rect.hclust(fit, k =6)

m3 <- t(m2)

set.seed(122)

k <- 6

kmeansresult <- kmeans(m3, k)

round(kmeansresult$centers, digits = 3)

for (i in 1:k){

cat(paste("cluster ", i, ": ", sep = ""))

s <- sort(kmeansresult$centers[i, ], decreasing = T)

cat(names(s)[1:5],"\n")

}

library(fpc)

d <- dist(t(unsuptdm), method = "euclidian")

kfit <- kmeans(d,20)

clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)

\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Phase 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**# getting sentiments**

mysentiment <- get\_nrc\_sentiment(as.character(train\_df))

sentimentscores <- data.frame(colSums(mysentiment[,]))

names(sentimentscores) <- "score"

sentimentscores <- cbind("sentiment" = rownames(sentimentscores), sentimentscores)

rownames(sentimentscores) <- NULL

ggplot(data = sentimentscores, aes(x=sentiment, y = score)) +

geom\_bar(aes(fill = sentiment), stat = "identity") +

theme(legend.position = "none")+

xlab("sentiment") + ylab("score") + ggtitle("total sentiment score based on reviews")

setwd("H:/Rlatest/Kinjal R/Capstone project")

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

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

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

{

require(plyr);

require(stringr);

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

sentence = gsub('[^A-z ]','', sentence)

sentence = tolower(sentence);

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

words = unlist(word.list);

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

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

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

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

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);

}

result=score.sentiment(train\_df,pos.words,neg.words)

result$pos <- as.numeric(result$score >=2)

result$neg <- as.numeric(result$score <=-2)

numpos = sum(result$pos)

numneg = sum(result$neg)

rm(numpos, numneg)

library(dplyr)

library(sqldf)

postext <- sqldf("select text from result where pos== 1")

negtext <- sqldf("select text from result where neg == 1")

corpusdf <- data.frame(text = sapply(mycorpus, as.character), stringsAsFactors = FALSE)

resultc <- score.sentiment(df1$term,pos.words,neg.words)

max(resultc$score)

min(resultc$score)

resultc$pos <- as.numeric(resultc$score >=1)

resultc$neg <- as.numeric(resultc$score <=-1)

poswords <- sqldf("select text from resultc where pos== 1")

negwords <- sqldf("select text from resultc where neg ==1")

set.seed(123)

wordcloud(resultc$text, freq = resultc$pos, min.freq = 1, max.words = Inf, width = 1000, height = 1000,

random.order = FALSE, rot.per = 0.35, colors = brewer.pal(8, "Dark2"))

wordcloud(resultc$text, freq = resultc$neg,min.freq = 1, max.words = Inf, width = 1000, height = 1000,

random.order = FALSE, rot.per = 0.35, colors = brewer.pal(8, "Dark2"))

neutral <- length(which(resultc$score == 0))

positive <- length(which(resultc$score >0))

negative <- length(which(resultc$score <0))

Sentiment <- c("Negative","Neutral","Positive")

Count <- c(negative,neutral,positive)

result1 <- as.data.frame(cbind(Sentiment,Count))

qplot(Sentiment,Count,data=result1,geom = "histogram", fill=factor(Sentiment),

binwidth=1,stat="identity",main="positive and negative sentiments of movie")

histogram(count)

ggplot(data = resultc, aes(Sentiment))+

geom\_point(alpha = 1, size = 9)

histogram(Count,Sentiment)

qplot(result1,geom = "histogram")

barplot(Count)

qplot(data=result1$Count,geom = "histogram", fill=Sentiment,

binwidth=1,stat="identity",main="positive and negative sentiments of movie", scale\_x\_continuous())

subset(df1, freq>15000) %>%

ggplot(aes(term,freq))+

geom\_bar(stat="identity", fill="darkred", colour="darkgreen") +

theme(axis.text.x=element\_text(angle=45, hjust=1))

subset(resultc, score ==1)%>%

ggplot(aes(text,score))+

geom\_bar(stat="identity", fill="darkred", colour="darkgreen") +

theme(axis.text.x=element\_text(angle=45, hjust=1))+

coord\_cartesian(c(0,10))

subset(resultc, score ==-1)%>%

ggplot(aes(text,score))+

geom\_bar(stat="identity", fill="darkred", colour="darkgreen") +

theme(axis.text.x=element\_text(angle=45, hjust=1))+

coord\_cartesian(c(0,10))

\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Phase 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# chi square test

H0: There is no significant difference between sentiments (positive and negative) with text

H1: There is significant difference the between sentiments (positive and negative) with text

chisq.test(combineposneg$sentiments, combineposneg$text)

Output: We accept the null hypothesis, and conclude there is significant difference.

\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Phase 5 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# combining the train and test data

totaldata <- rbind(readtrainpos,readtrainneg,readtestpos,readtestneg)

totaldata <- as.data.frame(totaldata$text)

colnames(totaldata) <- c("text")

# converting to corpus

doc1 <- Corpus(VectorSource(totaldata$text))

#pre-processing the data

corpus <- DocumentTermMatrix(doc1, control = list(

tolower = T,

removePunctuation = T,

removeNumbers = T,

stopwords=T,

stripWhitespace = T,

stemDocument=T

))

result=score.sentiment(c(train\_df,test\_df),pos.words,neg.words)

result$pos <- as.numeric(result$score >=2)

result$neg <- as.numeric(result$score <=-2)

sum(result$pos)

sum(result$neg)

library(dplyr)

library(sqldf)

postext <- sqldf("select \* from result where pos== 1")

postext$score <- NULL

postext$neg <- NULL

negtext <- sqldf("select text,neg from result where neg == 1")

negtext$neg<- 0

posnegtext <- unique(merge(postext,negtext, by="text",all.x = TRUE, all.y = TRUE))

x <- cbind(posnegtext[1], sentiments = na.omit(unlist(posnegtext[-1])))

xcorpus <- Corpus(VectorSource(x$text))

review\_corpus = tm\_map(xcorpus, content\_transformer(tolower))

review\_corpus = tm\_map(xcorpus, removeNumbers)

review\_corpus = tm\_map(xcorpus, removePunctuation)

review\_corpus = tm\_map(xcorpus, removeWords, c("the", "and", "br", "</>", stopwords("english")))

review\_corpus = tm\_map(xcorpus, stripWhitespace)

inspect(review\_corpus[1])

review\_dtm <- DocumentTermMatrix(review\_corpus)

review\_dtm

review\_dtm\_tfidf <- DocumentTermMatrix(review\_corpus, control = list(weighting = weightTfIdf))

review\_dtm\_tfidf = removeSparseTerms(review\_dtm\_tfidf, 0.95)

review\_dtm\_tfidf

m <- as.matrix(review\_dtm\_tfidf)

df <- data.frame(m)

df$polarity <- c(x$sentiments)

train\_dtm <- df[1:25000,]

test\_dtm <- df[25001:50000,]

tst\_tdm <- cbind(df$polarity,test\_dtm)

df$polarity <- as.factor(df$polarity)

svm.model <- ksvm(polarity ~., data = train\_dtm)

set.seed(123)

cpart <- createDataPartition(df$polarity, p=0.75, list = FALSE)

#Trainiing DTM

tr\_dtm <- train\_dtm [cpart, ]

dim(tr\_dtm)

table(tr\_dtm$polarity)

tr\_dtm$polarity <- as.factor(tr\_dtm$polarity)

#Validation DTM

val\_dtm <- train\_dtm [-cpart, ]

dim(tr\_dtm)

table(val\_dtm$polarity)

val\_dtm$polarity <- as.factor(val\_dtm$polarity)

**# SVM Model**

library (kernlab)

class\_svm <- ksvm(polarity ~ ., data = tr\_dtm)

summary(class\_svm)

# Doing prediction

pred\_svm <- predict(class\_svm, newdata = val\_dtm)

summary(pred\_svm)

#checking accuracy

tab\_svm <- table(pred\_svm, val\_dtm$polarity)

confusionMatrix(tab\_svm)

**## Random Forest Model**

library (randomForest)

class\_rf <- randomForest(as.factor(polarity) ~ ., data = tr\_dtm, ntree =500, importance = TRUE, mtry =3, na.action = na.exclude)

summary(class\_rf)

# Doing prediction

pred\_rf <- predict(class\_rf, newdata = val\_dtm)

summary(pred\_rf)

#checking accuracy

tab\_rf <- table(pred\_rf, val\_dtm$polarity)

confusionMatrix(tab\_rf)

**# Logistics model**

logic.model <- glm(polarity~., data = tr\_dtm, family = "binomial",na.action = na.exclude)

summary(logic.model)

pred\_logic <- predict(logic.model, newdata = val\_dtm, type = "response")

summary(pred\_logic)

fitted.results <- ifelse(pred\_logic >0.5,1,0)

#Calculating Accuracy

misClasificError <- mean(fitted.results != val\_dtm$polarity)

print(paste('Accuracy',1-misClasificError))

# Accuracy is 77%.

**# Clustering**

clus <- kmeans(df, 2, nstart = 100)

round(clus$centers, digits = 3)

set.seed(123)

k <- 2

for (i in 1:k){

cat(paste("cluster ", i, ": ", sep = ""))

s <- sort(clus$centers[i, ], decreasing = T)

cat(names(s)[1:2],"\n")

}

library(cluster)

clusplot(as.matrix(m), clus$cluster)

library(fpc)

d <- dist(t(m), method = "euclidian")

kfit <- kmeans(d,2)

clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)

**#PCA on total data**

review\_weight <- weightTfIdf(review\_dtm\_tfidf)

tdm.pca <- prcomp(train\_dtm, center = TRUE, scale. = TRUE)

plot(tdm.pca, type='l')

pcapredict1 <- predict(tdm.pca, newdata = test\_dtm)

adtm.m <- as.matrix(pcapredict1)

adtm.df <- as.data.frame(adtm.m)

#Output: Upto 8 PCA we can consider

***Observations: Sentiments and the text are co-related to eachother, and there is high significance between the two. For the prediction, model is accurate to 77%, hence we can predict positive and negative sentiments on the test data by 77%.***