```
api_key<-"WkqhCMmTBJnVbzAuZXLRfl2NJ"
secret_key<-"pmLDqRo1ghV1dF1CzdVRHqnAhIkeKqsYdXWsnh1Wlrszh8H765"
access_token<-"964916186256322561-0KkxuDeTp3KBW6RF3mzJa7eETX1KZAG"
access_token_secret="tkOApc0D2SyT1xFv1FRaEVYVAVsTNr5FpvCIfooDx8hJ9"
setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret)

positiveText<-read.csv("E:/Project Twiiter Analysis/sentiment analysis positive word
list.csv",header=FALSE,stringsAsFactors=FALSE)
str(positiveText)
positiveText<-positiveText$V1
positiveText<-unlist(lapply(positiveText, function(x){ str_split(x,"\n")}))
negativeText=read.csv("E:/Project Twiiter Analysis/sentiment analysis Negative word
list.csv",header=FALSE,stringsAsFactors=FALSE)
str(negativeText)
negativeText=negativeText$V1
negativeText<-unlist(lapply(negativeText, function(x){ str_split(x,"\n")}))
pos.words=c(positiveText,"upgrade")
neg.words=c(negativeText,"wtf","wait","waiting","epicfail","mechanical")
Amazon_tweets=searchTwitter('@Amazon',n=1000)
flipkart_tweets=searchTwitteR('@flipkart',n=1000)
Myntra_tweets=searchTwitter('@Myntra',n=1000)
Amazon_text=sapply(Amazon_tweets,function(t) t$getText())
flipkart_text=sapply(flipkart_tweets,function(t) t$getText())
Myntra_text=sapply(Myntra_tweets,function(t) t$getText())
no_of_tweets=c(length(Amazon_text),length(flipkart_text),length(Myntra_text))
Shopping_Site<-c(Amazon_text,flipkart_text,Myntra_text)

score.sentiment=function(sentences, pos.words,neg.words){

  sent.score=sapply(sentences, function(sentence,pos.words,neg.words){
    # removing punctuations
    sentences=gsub("[[:punct:]]","",sentence)
    # removing control charaters
    sentences=gsub("[[:cntrl:]]","",sentence)
    # removing digits
    sentences=gsub("\\d+","",sentence)
    # error handling function when trying to convert lower case
    tryTolower=function(x){
      y=NA
      try_error=tryCatch(tolower(x),error=function(e) e)
      if(!inherits(try_error,"error")){
        y=tolower(x)
      }
```

```
    return(y)
  }
  sentence=sapply(sentence,tryTolower)
  # split sentence into words with str_split (stringr package)
  word.list = str_split(sentences, "\\s+")
  words = unlist(word.list)
  # compare words to the dictionaries of positive & negative terms
  pos.matches = match(words, pos.words)
  neg.matches = match(words, neg.words)
  # get 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)
  # final score
  score = sum(pos.matches) - sum(neg.matches)
  return(score)
 }, pos.words, neg.words )

 sent.scores.datafrm = data.frame(text=sentences, score=sent.score)
 return(sent.scores.datafrm)
}


sent.scores = score.sentiment(Shopping_Site, pos.words,neg.words)
sent.scores$Shopping_Site = factor(rep(c("Amazon","flipkart","Myntra"), no_of_tweets))
sent.scores$positive <- as.numeric(sent.scores$score >0)
sent.scores$negative <- as.numeric(sent.scores$score <0)
sent.scores$neutral <- as.numeric(sent.scores$score==0)

Amazon_Shopping_Site <- subset(sent.scores, sent.scores$Shopping_Site=="Amazon")
Flipkart_Shopping_Site <- subset(sent.scores,sent.scores$Shopping_Site=="flipkart")
Myntra_Shopping_Site <- subset(sent.scores,sent.scores$Shopping_Site=="Myntra")
Amazon_Shopping_Site$polarity <- ifelse(Amazon_Shopping_Site$score
>0,"positive",ifelse(Amazon_Shopping_Site$score <
0,"negative",ifelse(Amazon_Shopping_Site$score==0,"Neutral",0)))
Flipkart_Shopping_Site$polarity <- ifelse(Flipkart_Shopping_Site$score
>0,"positive",ifelse(Flipkart_Shopping_Site$score <
0,"negative",ifelse(Flipkart_Shopping_Site$score==0,"Neutral",0)))
Myntra_Shopping_Site$polarity <- ifelse(Myntra_Shopping_Site$score
>0,"positive",ifelse(Myntra_Shopping_Site$score <
0,"negative",ifelse(Myntra_Shopping_Site$score==0,"Neutral",0)))

qplot(factor(polarity), data=Amazon_Shopping_Site, geom="bar",
```

```
fill=factor(polarity))+xlab("Polarity Categories") +
ylab("Frequency") + ggtitle("Customer Sentiments - Amazon Shopping Site")

qplot(factor(score), data=Amazon_Shopping_Site, geom="bar",
fill=factor(score))+xlab("Sentiment Score") + ylab("Frequency") + ggtitle("Customer Sentiment
Scores - Amazon Shopping Site")
qplot(factor(polarity), data=Flipkart_Shopping_Site, geom="bar",
fill=factor(polarity))+xlab("Polarity Categories") +
ylab("Frequency") + ggtitle(" Customer Sentiments - Flipkart Shopping Site ")
qplot(factor(score), data=Flipkart_Shopping_Site,
geom="bar",fill=factor(score))+xlab("Sentiment Score") + ylab("Frequency")
+ ggtitle("Customer Sentiment Scores - Flipkart Shopping Site")
qplot(factor(polarity), data=Myntra_Shopping_Site, geom="bar",
fill=factor(polarity))+xlab("Polarity Categories") +
ylab("Frequency") + ggtitle("Customer Sentiments - Myntra Shopping Site")
qplot(factor(score), data=Myntra_Shopping_Site, geom="bar",
fill=factor(score))+xlab("Sentiment Score") + ylab("Frequency") + ggtitle("Customer Sentiment
Scores - Myntra Shopping Site ")

datafrm = ddply(sent.scores, c("Shopping_Site"), summarise,pos_count=sum( positive ),
neg_count=sum( negative ),neu_count=sum(neutral))
datafrm$total_count = datafrm$pos_count +datafrm$neg_count + datafrm$neu_count

datafrm$pos_percent_score = round( 100*datafrm$pos_count/datafrm$total_count )

datafrm$neg_percent_score = round( 100*datafrm$neg_count/datafrm$total_count )

datafrm$neu_percent_score = round( 100*datafrm$neu_count/datafrm$total_count )

attach(datafrm)
pie.chart.abc <-paste(datafrm$Shopping_Site,datafrm$pos_percent_score)
pie.chart.abc <- paste(pie.chart.abc,"percent",sep="")
pie(pos_percent_score, labels = pie.chart.abc, col = rainbow(length(pie.chart.abc)),
main = "Positive Comparative Analysis - Shopping Site")

pie.chart.abc<-paste(datafrm$Shopping_Site,datafrm$neg_percent_score)
pie.chart.abc <- paste(pie.chart.abc,"percent",sep="")
pie(neg_percent_score, labels = pie.chart.abc, col =
rainbow(length(pie.chart.abc)), main = " Negative
Comparative Analysis - Shopping Site")

pie.chart.abc <-paste(datafrm$Shopping_Site,datafrm$neu_percent_score)
pie.chart.abc <- paste(pie.chart.abc,"percent",sep="")
```

```
pie(neu_percent_score, labels = pie.chart.abc, col =
      rainbow(length(pie.chart.abc)), main = "Neutral Comparative
    Analysis - Shopping Site")

write.table(Amazon_Shopping_Site,"E:/Project Twiiter Analysis/Amazon_Shopping_Site.csv",
append=T, row.names=F, col.names=T,sep=",")
Amazon_Shopping_Site_csv <-read.csv("E:/Project Twiiter
Analysis/Amazon_Shopping_Site.csv", header = TRUE, encoding = "UCS-2LE")
View(Amazon_Shopping_Site_csv)
dataf$class <- as.factor(dataf$class)

write.table(Amazon_Shopping_Site,"E:/Project Twiiter Analysis/Amazon_Shopping_Site.csv",
append=T, row.names=F, col.names=T,sep=",",)
Amazon_Shopping_Site_csv <-read.csv("E:/Project Twiiter
Analysis/Amazon_Shopping_Site.csv", header = TRUE, encoding = "UCS-2LE")

dataf<- read.csv("E:/Project Twiiter Analysis/Amazon_Shopping_Site_classif2.csv",
stringsAsFactors = FALSE)
dataf<- read.csv("E:/Project Twiiter Analysis/Amazon_Shopping_Site_classif1.csv",
stringsAsFactors = FALSE)

head(dataf)
set.seed(1)
dataf <- dataf[sample(nrow(dataf)), ]
head(dataf)
str(dataf)
dataf$class<-as.factor(dataf$class)

corpuss <- VCorpus(VectorSource(dataf$text))
inspect(corpuss[1:3])

corpus.clean <- corpuss %>% tm_map(content_transformer(tolower)) %>%
tm_map(removePunctuation) %>% tm_map(removeNumbers) %>% tm_map(removeWords,
stopwords(kind="en")) %>% tm_map(stripWhitespace)
dtm <- DocumentTermMatrix(corpus.clean)
inspect(dtm[40:50, 10:15])
dataf.train <- dataf[1:1600,]
dataf.test <- dataf[1601:1992,]
dtm.train<-dtm[1:1600,]
dtm.test<-dtm[1601:1992,]
corpus.clean.train <- corpus.clean[1:1600]
corpus.clean.test <- corpus.clean[1601:1992]
dim(dtm.train)
```

```
fivefreq <- findFreqTerms(dtm.train, 5)
length((fivefreq))
dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary = fivefreq))
dim(dtm.train.nb)
dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary = fivefreq))
dim(dtm.test.nb)
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
train_NB <- apply(dtm.train.nb, 2, convert_count)
test_NB <- apply(dtm.test.nb, 2, convert_count)
system.time( classifier <- naiveBayes(train_NB, dataf.train$class,laplace = 1))
system.time( pred <- predict(classifier, newdata=test_NB))
table("Predictions"= pred, "Actual" = dataf.test$class )
conf_matrix <- confusionMatrix(pred, dataf.test$class)
```