

|  |  |
| --- | --- |
| **Student Name:** | Jayakarthi Boovendran |
| **Student ID**: | 19230487 |
| **Course:** | MBY – MSc Business Analytics |
| **Assignment Topic**: | Text Mining using R |

# *Part I*

# *Sentiment Analysis using Quanteda*

***R Code***

*#Package Installations*

install.packages("quanteda")

install.packages("wordcloud")

install.packages("RColorBrewer")

install.packages("dendextend")

*#Download & Extract Data*

data\_Url = "http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz"

download.file(data\_Url, "rt-polaritydata.tar.gz")

untar("rt-polaritydata.tar.gz")

*#Loading data into dataframes*

df\_neg <- data.frame(sentence = readLines("./rt-polaritydata/rt-polarity.neg"),stringsAsFactors = FALSE)

df\_neg['sentiment'] <- "neg" *#Assigning 'sentiment'='neg' for Negative Reviews*

head(df\_neg)

df\_pos <- data.frame(sentence = readLines("./rt-polaritydata/rt-polarity.pos"),stringsAsFactors = FALSE)

df\_pos['sentiment'] <- "pos" *#Assigning 'sentiment'='pos' for Positive Reviews*

head(df\_pos)

*#Creating corpus from dataframes*

library(quanteda)

corpus\_reviews <- corpus(rbind(df\_neg, df\_pos), text\_field='sentence')

token\_reviews <- tokens(corpus\_reviews) *#Extracting tokens from corpus*

head(token\_reviews)

*#Removing punctuation & English stop-words from the created tokens*

token\_reviewsnopunc <- tokens(token\_reviews, remove\_punct = TRUE)

head(token\_reviewsnopunc)

token\_reviewsnostop <- tokens\_select(token\_reviewsnopunc, pattern = stopwords('en'), selection = 'remove')

*#Creating Document-feature Matrix for the tokens*

review\_matrix <- dfm(token\_reviewsnostop)

head(review\_matrix)

*#a)Creating a frequency plot*

library(ggplot2)

ggplot(textstat\_frequency(review\_matrix, n = 15),

aes(x = reorder(feature, frequency),y = frequency)) +

geom\_point(size=2,colour='blue') +

coord\_flip() +

labs(title="Frequency Plot of Top 15 Words",x = "Words", y = "Frequency") +

theme\_minimal()

*#b)A word-cloud of 50 most common words*

library(wordcloud)

library(RColorBrewer)

textplot\_wordcloud(review\_matrix, max\_words = 50, random.order=TRUE, rot.per=0.35,

colors=brewer.pal(8, "Dark2"))

*#c)A grouped word-cloud of 50 most common words in positive and negative sentences*

names(docvars(token\_reviewsnostop))

docvars(token\_reviewsnostop, "Sentiments") <-

factor(ifelse(docvars(token\_reviewsnostop, "sentiment") == "pos", "Positive", "Negative"))

grouped\_corpus\_reviews <- dfm(token\_reviewsnostop, groups = "Sentiments") *#Grouping by Sentiments*

textplot\_wordcloud(grouped\_corpus\_reviews, comparison = TRUE, max\_words = 50, color=brewer.pal(8, "Dark2"))

*#d)A plot of lexical diversity of randomly selected 20 sentences*

lex\_diver <- dfm(token\_reviewsnostop)

test\_stat <- textstat\_lexdiv(lex\_diver)

rand\_t20 <- tstat[sample(nrow(test\_stat), 20, replace = FALSE, prob = NULL),] *#Sampling 20 sentences*

plot(rand\_t20$TTR, type = 'l', xaxt = 'n', xlab = NULL, ylab = "TTR")

grid()

axis(1, at = seq\_len(nrow(rand\_t20)), labels = lex\_diver$Reviews)

*#e)A dendogram of hierarchical clustering of randomly selected 20 sentences*

rand\_t20 <-dfm\_sample(review\_matrix,20, replace = FALSE)

dist\_matrix <- as.dist(textstat\_dist(rand\_t20)) *#Creating distance matrix*

cluster<- hclust(dist\_matrix) *#Creating hierarchical clustering*

dend\_obj <- as.dendrogram(cluster)

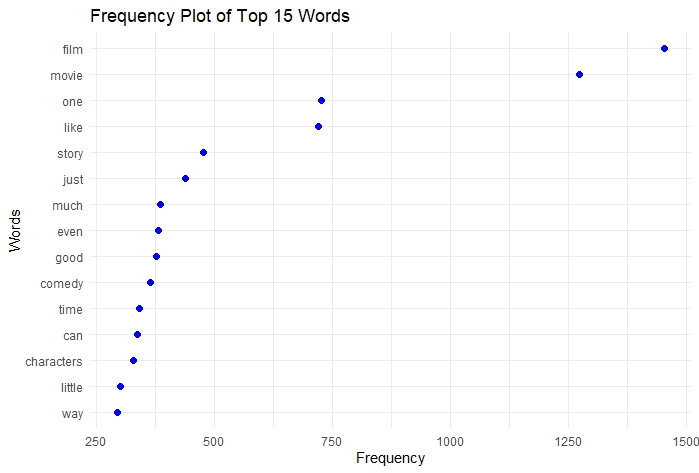
library(dendextend)

colored\_dend <- color\_branches(dend\_obj, height = 8)

plot(colored\_dend)

## *Outputs*

***a) Frequency plot of top 15 words***



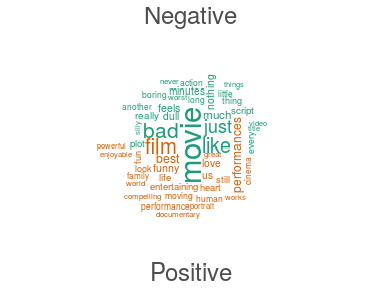
* *In general, frequency plots in text mining are used to identify and extract the most frequently used term in a particular context.*
* *Furthermore, frequency plots can help analyse the text trends over time. For instance, extracting #hashtags and plotting them gives the most used or trending hastags.The above-mentioned figure shows the top 15 frequently used words in the movie reviews.*

***b) Word-cloud of 50 most common words***



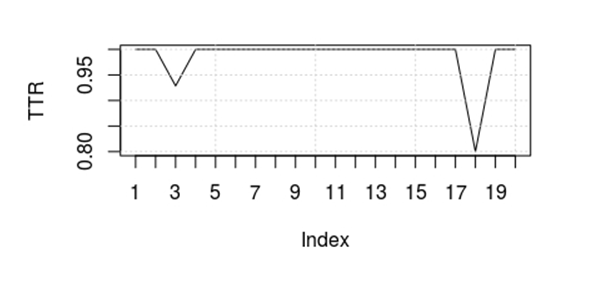
* *Word-clouds are used to highlight the frequently used terms as a cloud of words.*
* *A word-cloud is similar to that of a frequency plot, except the fact that words with higher frequency appears bigger than lower-frequency words. In this case, ‘film’ and ‘movie’ are clearly the most used terms.*

***c) Grouped word-cloud of 50 most common words in positive and negative sentences***

******

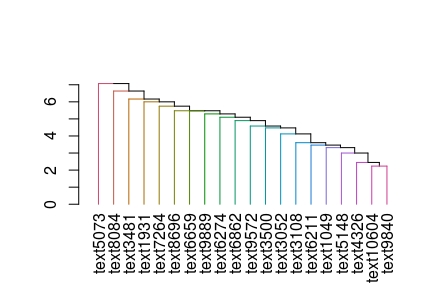
* *A grouped word-cloud is similar to that of a normal cloud, but the words are* ***categorised into groups*** *based on a condition.*
* *In this example, the top 50 words are split into two groups based on the type of* ***sentiment****.*
* *This kind of representations gives more value to the user than the regular clouds.*

***d) A plot of lexical diversity of randomly selected 20 sentences***



* *A lexical diversity plot represents the ‘****lexical richness’*** *of the tokens in a sentence. In other terms, lexical density is the ratio of content words to function words in a text.*
* *The higher the content words, the greater the lexical density. The above representation shows the lexical density of twenty randomly selected sentences.*

***e) A dendogram of hierarchical clustering of randomly selected 20 sentences***

******

* *A dendograms represents the hierarchial relationship between entities, in this case, sentences. Provided above is a dendogram of 20 randomly selected sentences.*
* *The cards that join together are more similar to that which don’t join. In this example, ‘****text9572****’ and ‘****text3500****’ are similar as they are joined at the same level.*

# *Part II*

***Sentiment Analysis using Lexicoder Sentiment Dictionary***

***I. Problem Statement***

To perform S**entiment Analysis** of movie reviews using ***Lexicoder Sentiment Dictionary*** (*quanteda - data\_dictionary\_LSD201*) and report on the performance metrics of the predicted results.

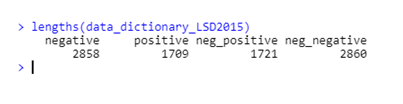
***II. Dataset Description***

The dataset contains two files of 5331 positive and 5331 negative sentences from movie reviews in Rotten Tomatoes website. In the given exercise, the data are processed in the form of a corpus. For the same, we make use of the ‘***quanteda’*** application package.

***III. Dictionary Analysis***

Lexicoder Sentiment Dictionary is intended to capture ***sentiment*** in political texts. According to R Documentation [4], the dictionary uses four keys.

* ***negative*** *- 2,858 word patterns indicating negative sentiment*
* ***positive****- 1,709 word patterns indicating positive sentiment*
* ***neg\_positive****- 1,721 word patterns indicating a positive word preceded by a negation*
* ***neg\_negative****- 2,860 word patterns indicating a negative word preceded by a negation*



*Fig 1: Four Keys of Lexicoder Dictionary*

***IV. Implementation***

## *R Script*

*#Package Installations*

install.packages("quanteda")

install.packages("devtools")

devtools::install\_github("quanteda/quanteda.corpora")

install.packages("readtext")

install.packages("caret")

install.packages("e1071")

*#Loading Packages*

library(quanteda)

library(quanteda.corpora)

library(caret)

*#Download & Extract Data*

data\_Url = "http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz"

download.file(data\_Url, "rt-polaritydata.tar.gz")

untar("rt-polaritydata.tar.gz")

*#Loading data into dataframes*

df\_neg <- data.frame(sentence = readLines("./rt-polaritydata/rt-polarity.neg"),stringsAsFactors = FALSE)

df\_neg['sentiment'] <- "neg" *#Assigning 'sentiment'='neg' for Negative Reviews*

head(df\_neg)

df\_pos <- data.frame(sentence = readLines("./rt-polaritydata/rt-polarity.pos"),stringsAsFactors = FALSE)

df\_pos['sentiment'] <- "pos" *#Assigning 'sentiment'='pos' for Positive Reviews*

head(df\_pos)

*#Creating a data frame with Postive and Negative Reviews*

combined\_reviews<-rbind(df\_neg,df\_pos)

*#Initial Setup*

corpus\_reviews <- corpus(rbind(df\_neg, df\_pos), text\_field='sentence') *#Creating Corpus*

token\_reviews <- tokens(corpus\_reviews) *#Extracting tokens from corpus*

token\_reviewsnopunc <- tokens(token\_reviews, remove\_punct = TRUE) *#Removing punctuation from tokens*

*#Removing English stop-words*

token\_reviewsnostop <- tokens\_select(token\_reviewsnopunc, pattern = stopwords('en'), selection = 'remove')

*#Lookup using Dictionary Lexicoder Sentiment Dictionary LSD2015*

lengths(data\_dictionary\_LSD2015)

review\_lookup\_data<- tokens\_lookup(token\_reviewsnostop, dictionary = data\_dictionary\_LSD2015[1:2])

lookup\_matrix<- dfm(review\_lookup\_data)

df\_lookup<-convert(lookup\_matrix, to = "data.frame")

*#Appending No. of positive & Negative tokens to combined review dataframe*

combined\_reviews['neg\_tokens']<-df\_lookup['negative']

combined\_reviews['pos\_tokens']<-df\_lookup['positive']

*#Assigning sentiemnt based on the max tokens*

combined\_reviews['predicted\_sentiment']<-ifelse(df\_lookup['negative']>=df\_lookup['positive'],'neg','pos')

head(combined\_reviews)

*#Creating Confusion Matrix using the actual and predicted sentiments*

conf\_matrx<-confusionMatrix(table(combined\_reviews[c('sentiment','predicted\_sentiment')]))

conf\_matrx

*#Alternate Method to compute Classification statistics*

*#Retrieve confusion matrix*

review\_conf\_matrix<-conf\_matrx$table

review\_conf\_matrix

*#Compute values for TP, TN, (TP+TN+FP+FN)*

n = sum(review\_conf\_matrix) *# TP+TN+FP+FN*

nc = nrow(review\_conf\_matrix) *# Number of classes = 2 (binary classification)*

true\_predictions = diag(review\_conf\_matrix) *# Number of True Predictions (diagonal values: TP+TN)*

row\_sums = apply(review\_conf\_matrix, 1, sum) *# Number of instances per class*

col\_sums = apply(review\_conf\_matrix, 2, sum) *# Number of predictions per class*

p = row\_sums / n *# Distribution of instances over the actual classes*

q = col\_sums / n *# Distribution of instances over the predicted classes*

*#Accuracy calculation*

accuracy = sum(true\_predictions) / n *# Accuracy = (TP+TN)/(TP+TN+FP+FN)*

accuracy

*#Precision & Recall calculation*

precision = true\_predictions / col\_sums *#Precision = TP / (TP + FP)*

recall = true\_predictions / row\_sums *#Recall = TP / (TP + FN)*

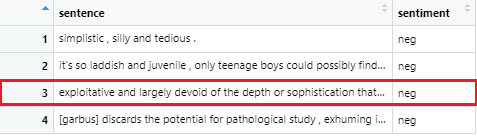
*#F Score Calculation*

f1 = 2 \* precision \* recall / (precision + recall) #*F-score = (2\*Recall\*Precision)/(Recall +Presision)*

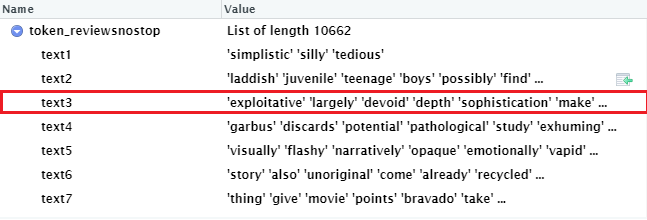
data.frame(precision, recall, f1)

## *V. Sample Output*

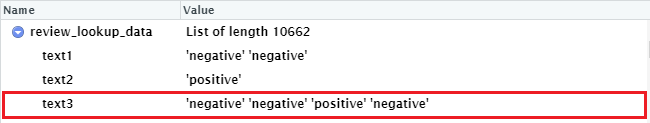
* ***Original sentence***

******

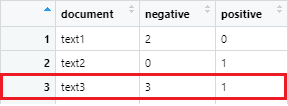
* ***Tokens after removing stop-words and punctuation***

**

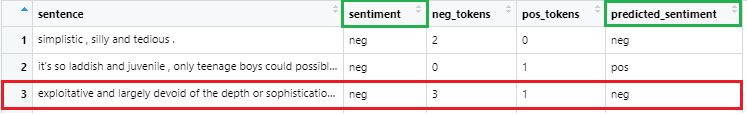
* ***Sentiment of tokens using “tokens\_lookup()” with dictionary***

**

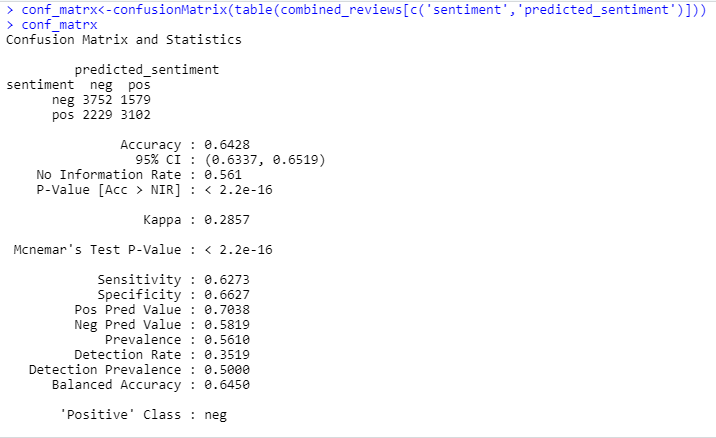
* ***Count of positive and negative tokens***

**

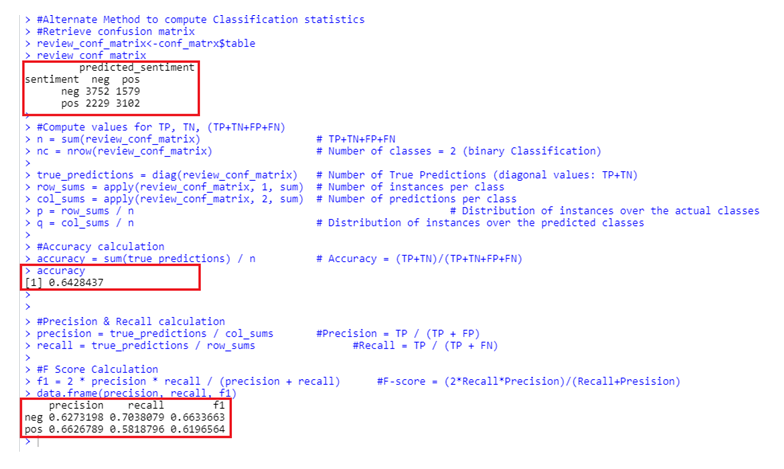
* ***Sentiment of sentence based on sentiment of most tokens***

**

* ***Confusion Matrix and Statistics***



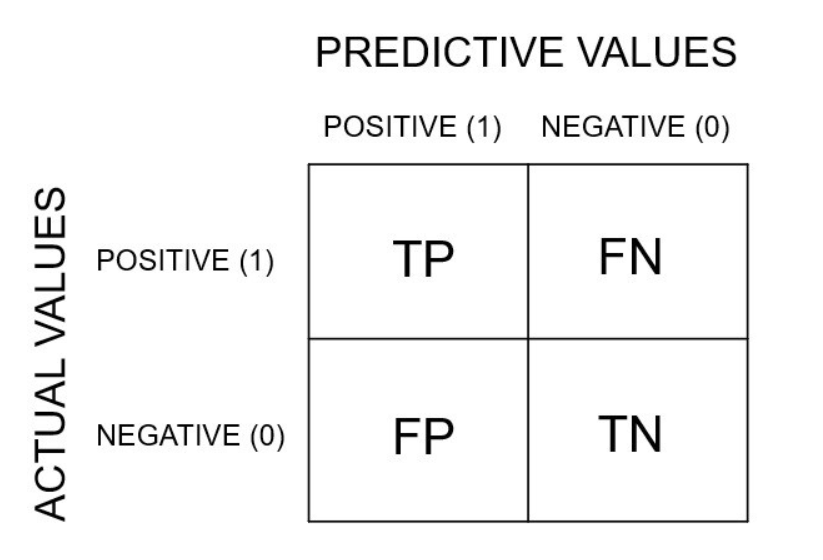
* ***Performance metrics calculated using formulae***



***VI. Performance Evaluation***

***Confusion Matrix***

A confusion matrix plays a significant role in ***assessing the classification model*** for its correctness and the types of errors it is making. A confusion matrix checks the performance of a classification model with a test data for which the actual values are known. These performance metrics include accuracy, precision, recall, F-score, and specificity [1].



*Fig 2: Confusion matrix for binary classification problem*

***Definition of the Terms:***

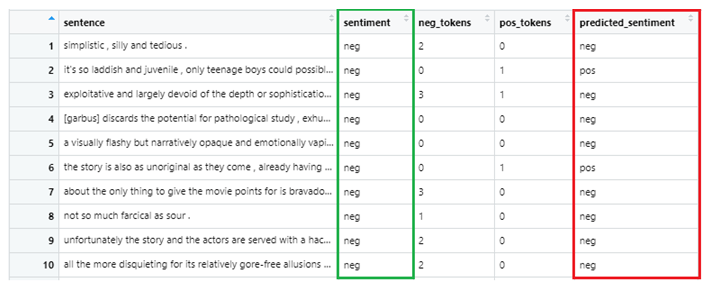
A confusion matrix presents a summary of the predictive results and actual results in a classification problem. Following are the common terms used in a confusion matrix [2].

* ***True Positive****: You predicted positive, and it is true.*
* ***True Negative****: You predicted negative, and it is true.*
* ***False Positive*** *(Type 1 Error): You predicted positive, and it is false.*
* ***False Negative*** *(Type 2 Error): You predicted negative, and it is false.*

***Explanation***

In the given example, we are computing confusion matrix using the actual and predicted values of sentiments. The actual values of sentiments are assigned manually based on the type of review. Whereas, the predicted values are obtained through lookup, using the dictionary (*data\_dictionary\_LSD201*)

The ***ConfusionMatrix( )*** function offered by the caret package takes in the actual and predicted values as a table and generates a confusion matrix alongside a few pre-computed statistics and performance classifiers.

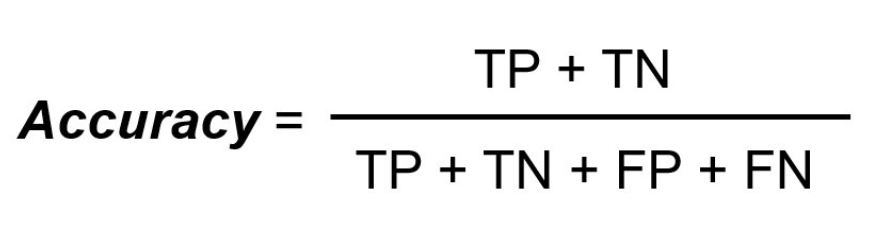


*Fig 3: Actual and Predicted sentiments*

***Performance Metrics***

* ***Classification Accuracy***

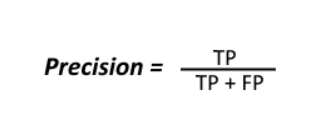
Accuracy can be defined as the number of correctly predicted cases out of all the classes. In any classification model, the accuracy should be as high as possible. The Classification Accuracy can be calculated using the following formula.



In our example, we obtained an accuracy of **0.6428** or an accuracy rate of **64.28%.** Therefore, the Classification model should be adjusted to provide more accuracy.

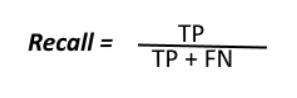
* ***Precision***

Precision is the number of actual positive cases out of all the predicted positive classes. It is essential that the precision value should be as high as possible. In our case, we obtained a precision value of **0.6273** (*Sensitivity*). Precision can be obtained using the following formula,



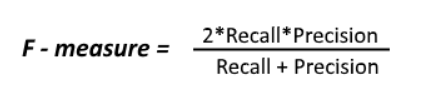
* ***Recall***

Recall is the correctly predicted classes in all the positive classes. It should be as high as possible. In our scenario, the recall value is **0.7038** (*Pos. Pred. Value*). Recall can be calculated using the below formula,



* ***F-Score***

F-Score allows comparison of two models with low precision and high recall or vice versa. In the given example, the obtained value of F1 is **0.6633.** F-Score can be calculated using the formula,



***VII. Conclusion***

The final output shows that the classification model built using Lexicoder Sentiment Dictionary can predict whether the movie review is positive or negative, with an ***accuracy of approximately 64%***. Therefore, using a prediction model or a more efficient dictionary for lookup will yield better results.

***Part III***

***Sentiment Analysis using Naive Bayes Classification Algorithm***

***I. Problem Statement***

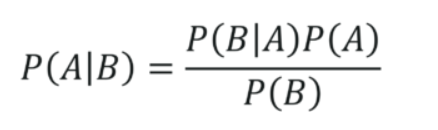
To train a ***Naïve Bayes classifier*** model to perform Sentimental Analysis on text data and investigate the performance metrics of the prediction model.

***II. Dataset Description***

The dataset contains two files of 5331 positive and 5331 negative sentences from movie reviews in Rotten Tomatoes website. In the given exercise, the data are processed in the form of a corpus. For the same, we make use of the ‘***quanteda’*** application package.

***III. Model Analysis***

Naive Bayes is a Supervised Machine Learning algorithm based on the Bayes Theorem. The algorithm solves classification problems using a probabilistic approach*.* Following is the Bayes Rule*,*



*In the above equation [3]:*

* *P(A|B): Conditional probability of event A occurring, given the event B*
* *P(A): Probability of event A occurring*
* *P(B): Probability of event B occurring*
* *P(B|A): Conditional probability of event B occurring, given the event A*

***IV. Model Implementation***

* ***R Script***

*#Step 1- Import and load required packages*

install.packages("quanteda")

install.packages("devtools")

devtools::install\_github("quanteda/quanteda.corpora")

install.packages("readtext")

install.packages("caret")

install.packages("e1071")

install.packages("quanteda.textmodels")

library(quanteda)

library(quanteda.corpora)

library(caret)

*#Data Preparation*

*#Step 2 – Download & Extract data*

data\_Url = "http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz"

download.file(data\_Url, "rt-polaritydata.tar.gz")

untar("rt-polaritydata.tar.gz")

*#Step 3 – Load data into data frames*

df\_neg <- data.frame(sentence = readLines("./rt-polaritydata/rt-polarity.neg"),

stringsAsFactors = FALSE)

*#Assigning 'sentiment'='neg' for Negative Reviews*

df\_neg['sentiment'] <- "neg"

df\_pos <- data.frame(sentence = readLines("./rt-polaritydata/rt-polarity.pos"),

stringsAsFactors = FALSE)

*#Assigning 'sentiment'='pos' for Positive Reviews*

df\_pos['sentiment'] <- "pos"

*#Step 4 – Create a corpus using the loaded data*

corpus\_reviews <- corpus(rbind(df\_neg, df\_pos), text\_field='sentence') #Creating Corpus

*#Data Modelling*

*#Step 5 – Split data into training set and test sets*

set.seed(300)

*#Splitting 70% data to Training set*

train\_set <- sample(1:10662, size = round(0.7\*10662) , replace = FALSE)

*#Splitting 30% data to Test set*

test\_set <- sample(1:10662, size = round(0.3\*10662) , replace = FALSE)

corpus\_reviews$id\_numeric <- 1:ndoc(corpus\_reviews)

*#Step 6 – Create corpus subsets by removing stop-words*

dfmat\_testdata <- corpus\_subset(corpus\_reviews, !id\_numeric %in% test\_set) %>%

dfm(stem = TRUE) %>% dfm(remove = stopwords("english"), stem = TRUE)

dfmat\_trainingdata <- corpus\_subset(corpus\_reviews, id\_numeric %in% train\_set) %>%

dfm(stem = TRUE) %>% dfm(remove = stopwords("english"), stem = TRUE)

*#Step 7 – Build, train and predict using Naïve model*

require(quanteda.textmodels)

tmod\_nb <- textmodel\_nb(dfmat\_trainingdata, docvars(dfmat\_trainingdata, "sentiment"))

dfmat\_matched <- dfm\_match(dfmat\_testdata, features = featnames(dfmat\_trainingdata))

actual\_class <- docvars(dfmat\_matched, "sentiment")

predicted\_class <- predict(tmod\_nb, dfmat\_matched)

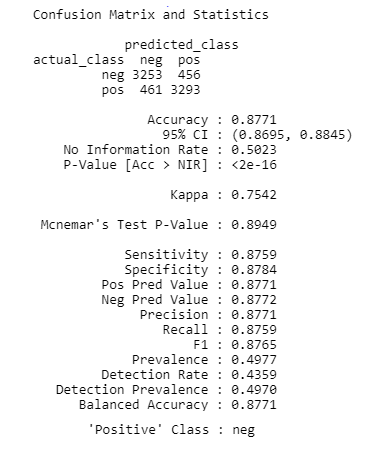
tab\_class <- table(actual\_class, predicted\_class)

*#Model Evaluation*

*#Step 8 – Create a confusion matrix to evaluate the performance metrics*

confusionMatrix(tab\_class, mode = "everything")

***Output***

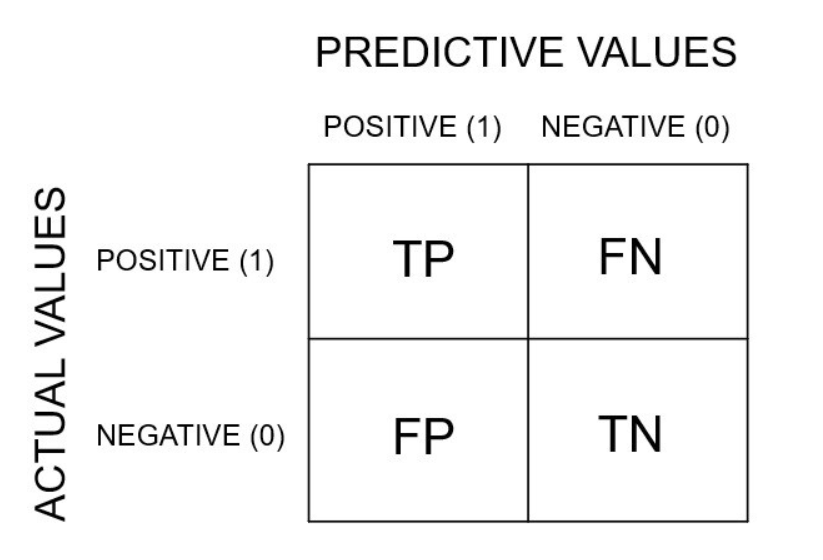
**

*Fig 4: Confusion Matrix & Classification Metrics*

***V. Model Evaluation***

* ***Confusion Matrix***

A confusion matrix plays a significant role in ***assessing the classification model*** for its correctness and the types of errors it is making. A confusion matrix checks the performance of a classification model with a test data for which the actual values are known. These performance metrics include ***accuracy, precision, recall, F-score, and specificity*** [1].



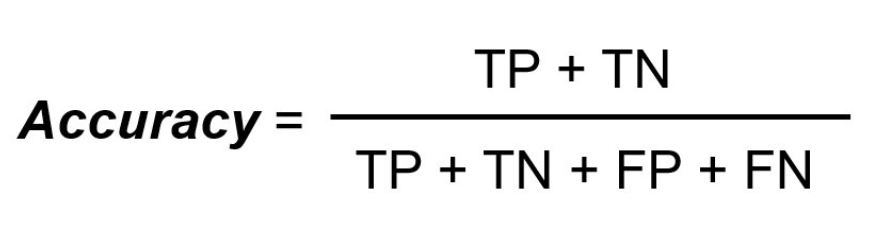
*Fig 5: Confusion matrix for binary classification problem*

* ***Definition of the Terms:***

A confusion matrix presents a summary of the predictive results and actual results in a classification problem. Following are the common terms used in a confusion matrix [2].

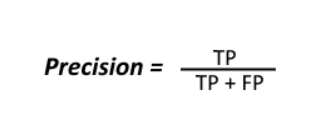
* ***True Positive****: You predicted positive, and it is true.*
* ***True Negative****: You predicted negative, and it is true.*
* ***False Positive*** *(Type 1 Error): You predicted positive, and it is false.*
* ***False Negative*** *(Type 2 Error): You predicted negative, and it is false.*
* ***Accuracy***

Accuracy can be defined as the number of correctly predicted cases out of all the classes. In any classification model, the accuracy should be as high as possible. In our example, we obtained an accuracy of **0.8777** or an accuracy rate of **87.77%.** The Classification accuracy can be calculated using the following formula.



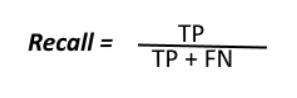
* ***Precision***

Precision is the number of actual positive cases out of all the predicted positive classes. It is essential that the precision value should be as high as possible. In our case, we obtained a precision value of **0.8665** (*Sensitivity*). Precision can be obtained using the following formula,



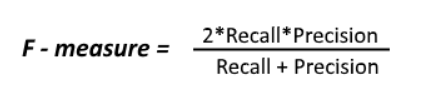
* ***Recall***

Recall is the correctly predicted classes in all the positive classes. It should be as high as possible. In our scenario, the recall value is **0.8911** *(Pos. Pred. Value).* Recall can be calculated using,



* ***F-Score***

F-Score allows comparison of two models with low precision and high recall or vice versa. In the given example, the obtained value of F1 is **0.6633.** F-Score can be calculated using the formula,



***VI. Conclusion***

The final output shows that the Naive Bayes classifier can predict whether the movie review is positive or negative, with an ***accuracy of approximately 88%***.

***References***

[1] <https://towardsdatascience.com/decoding-the-confusion-matrix-bb4801decbb>, Accessed On: 02 April 2020

[2] [https://towardsdatascience.com/understanding-confusion-matrix](https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62), Accessed On: 02 April 2020

[3] <https://www.edureka.co/blog/naive-bayes-in-r/>, Accessed On: 03 April 2020

[4] <http://search.r-project.org/library/quanteda/html/data_dictionary_LSD2015.html>, Accessed on : 2 April 2020