# Configuration Manual

MSc Research Project
Data Analytics

ADEDEJI ADEGOKE
Student ID: X18174582

School of Computing
National College of Ireland

Supervisor: Dr Catherine Mulwa

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | ADEDEJI ADEGOKE |
| **Student ID:** | X18174582 |
| **Programme:** | MSc DATA ANALYTICS **Year:** 2019 |
| **Module:** | RESEARCH PROJECT |
| **Lecturer:** | DR CATHERINE MULWA |
| **Submission Due Date:** | 12th DECEMBER 2019 |
| **Project Title:** | CONFIGURATION MANUAL |
| **Word Count:** | 2205 **Page Count:** 29 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** ………………………………………………………………………………………………………………

**Date:** ………………………………………………………………………………………………………………

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Detection of Depression among Nigerians using Machine Learning Techniques

ADEDEJI ADEGOKE

x18174582

## 1 Introduction

This research project adopts classification, which was used to develop depression detection models for classifying depressive and non-depressive tweets. This document explains the hardware and software configuration which was used for the implementation and execution of the research project. The extraction process of acquiring depressive tweets, exploratory data analysis, pre-processing stages of the datasets which was not fully reported in the technical report and codes used for the process were shown in Chapter 2 while Chapter 3 shows code used for the feature extraction and implementation process used in implementing the models, Chapter 4 shows the results of the implemented models and how the models were evaluated with adopted evaluation metrics from the literature and comparison of the existing work with developed models.

## 2 Environment

### 2.1 Hardware Configuration

A Hewlett Packard computer was used for the implementation of this research project and properties are shown in figure 1 below.

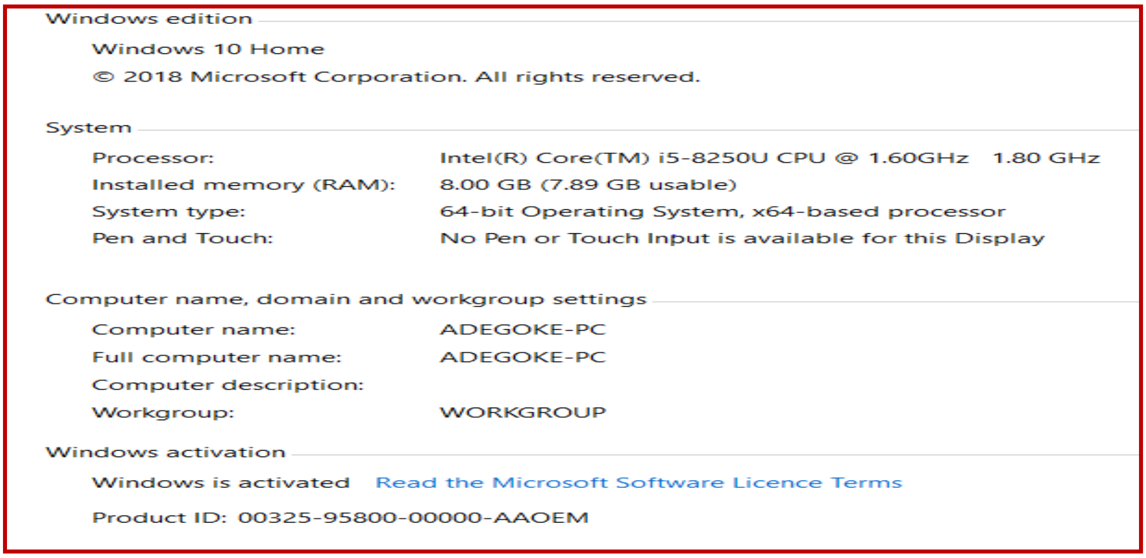
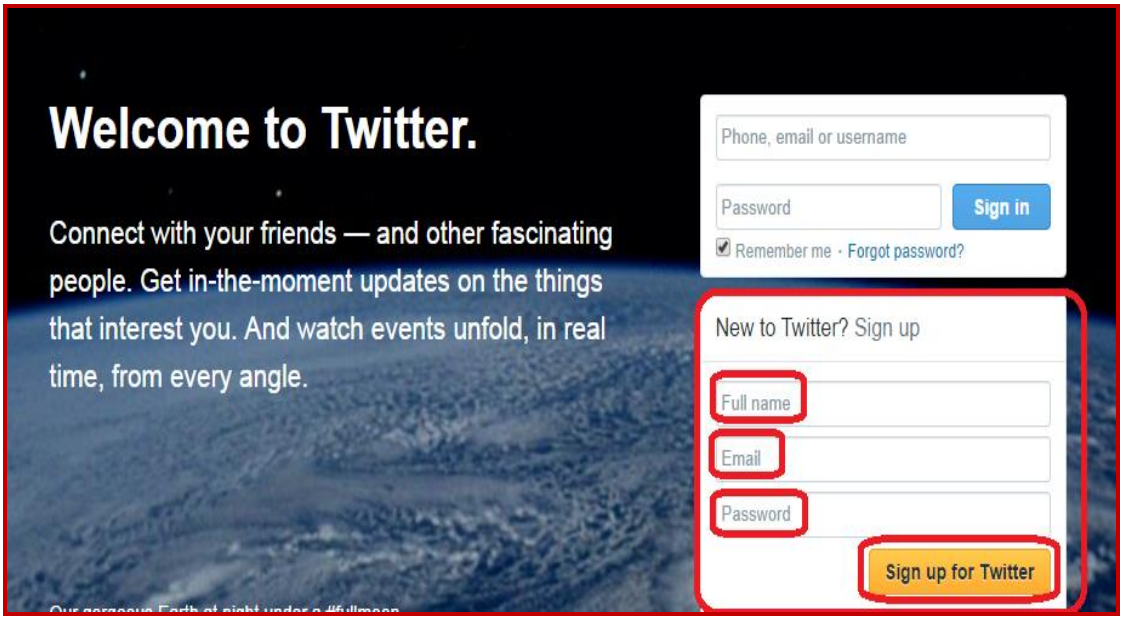



Figure 1. Personal computer (PC) configuration.

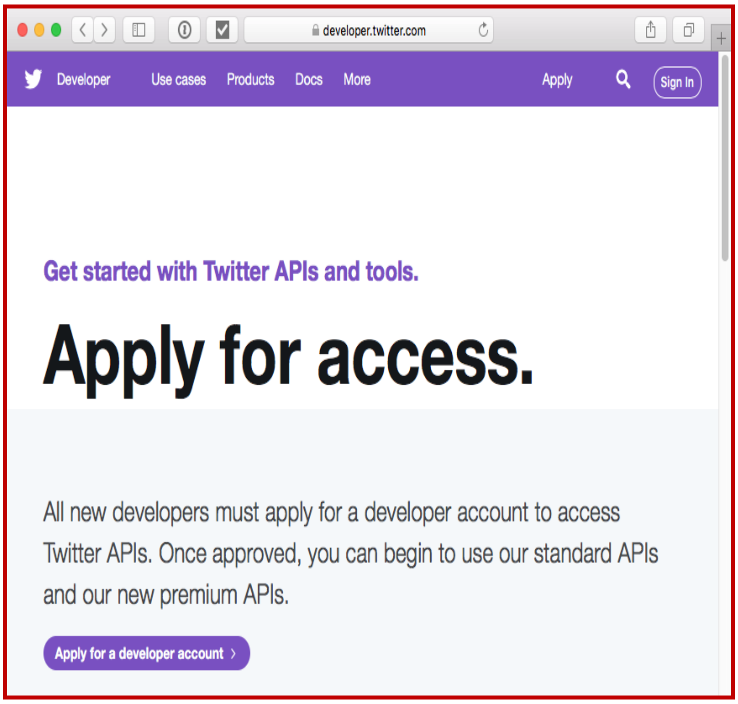
## 2.2 Software

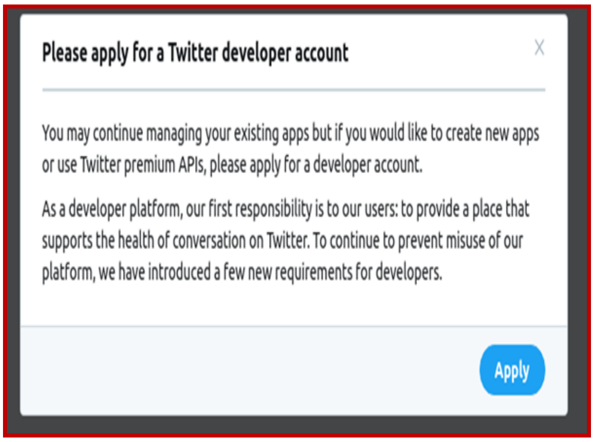
Authentication keys for extraction of tweets were acquired from Twitter developer account and the process are as follow:

Step 1: Open a twitter account

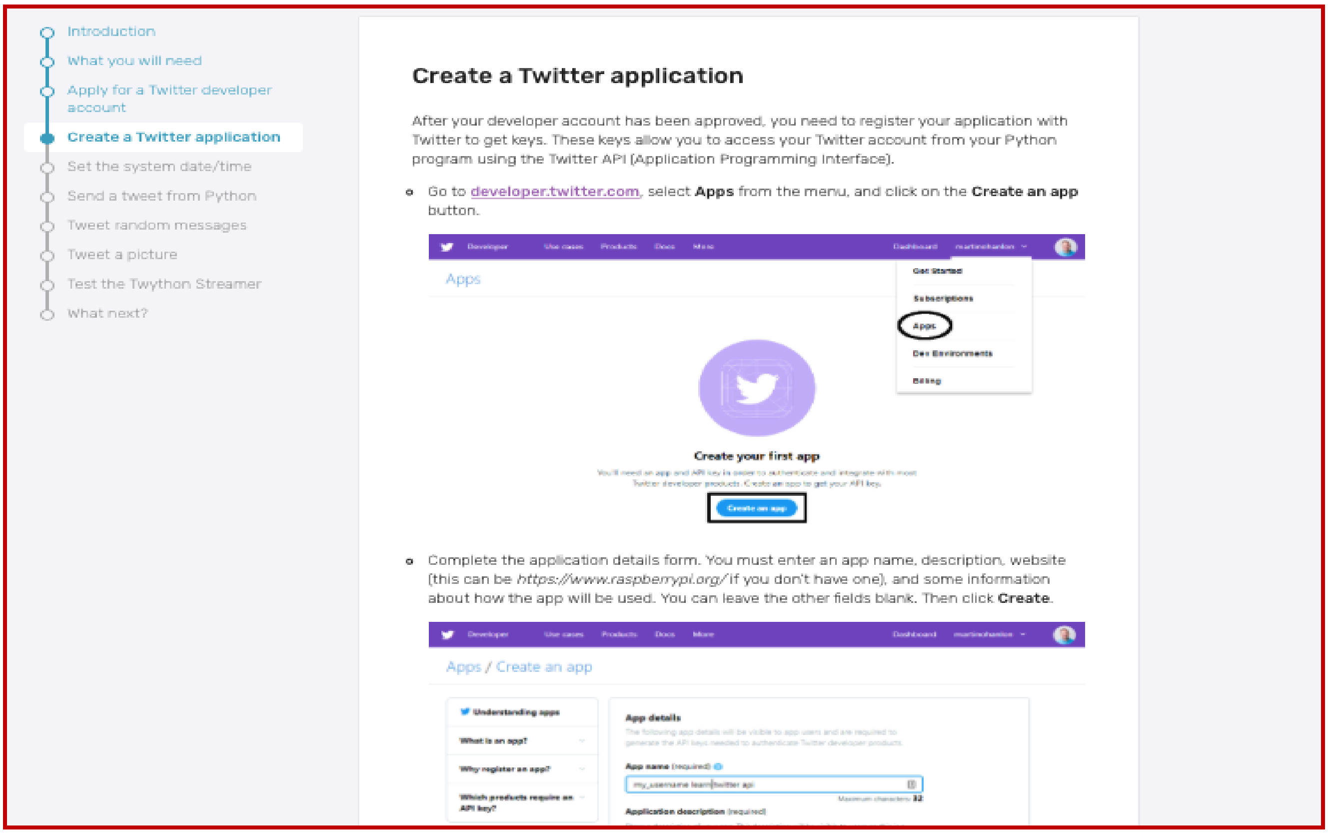



Step 2: Use the twitter account to sign into twitter developer account.

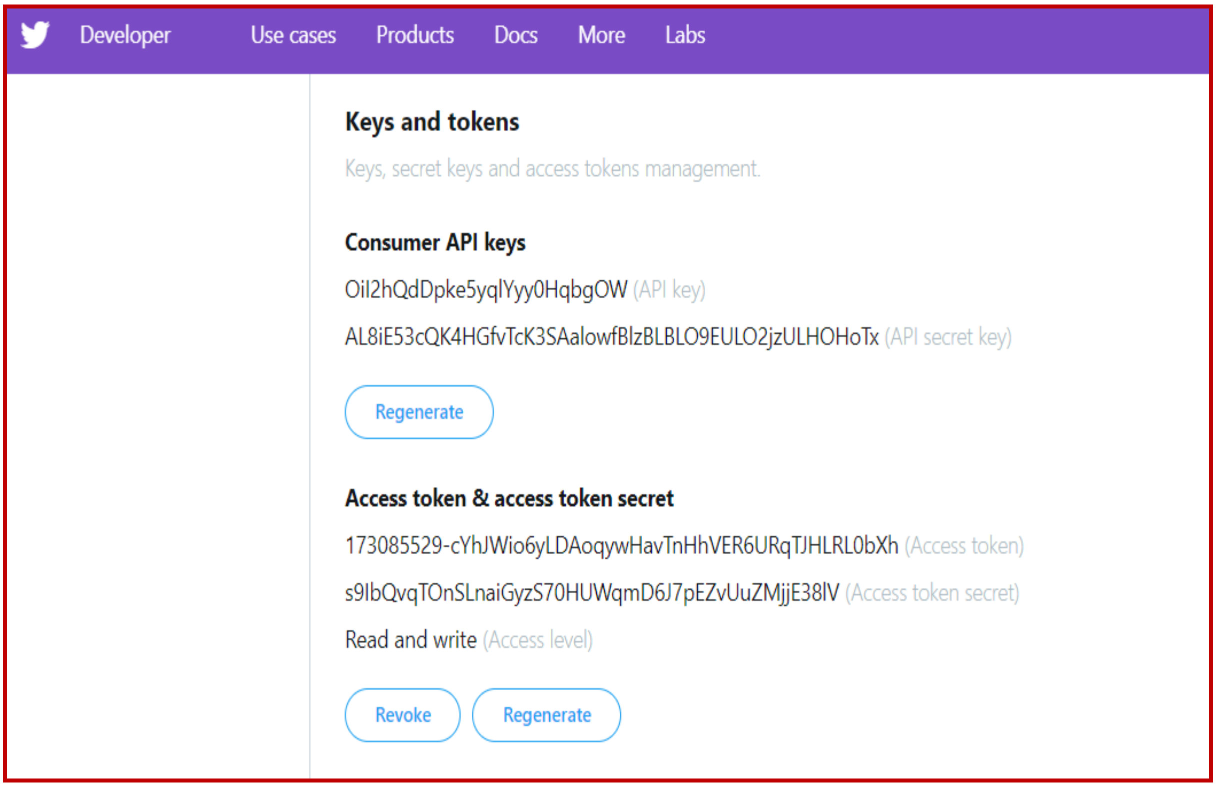



Step3: Create an application for the project in order to acquire the authentication keys for extraction.




Step 4: Obtain the authentication keys for extraction.



Implementation are done in R studio and Google colaboratory using python [1] using various libraries.

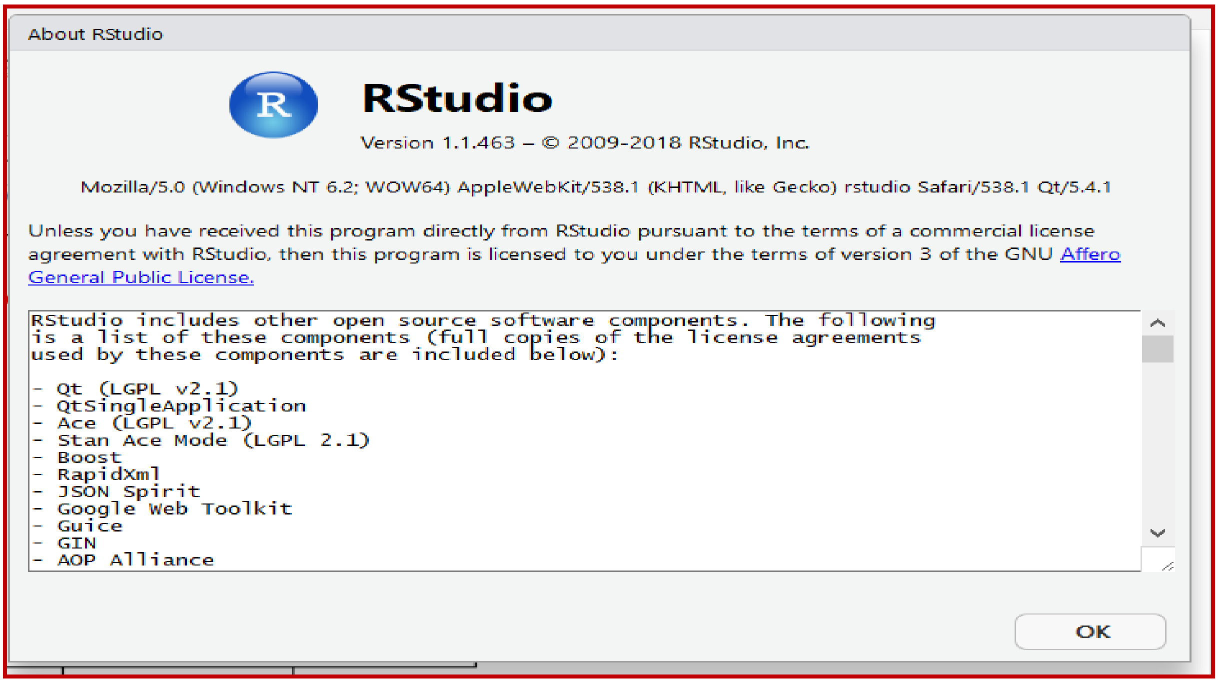



Figure 2. R Studio version used for the implementation

In order to access python on colaboratory, a google account was acquired to log into the colaboratory interface. Figure 3 shows the python notebook on colaboratory.

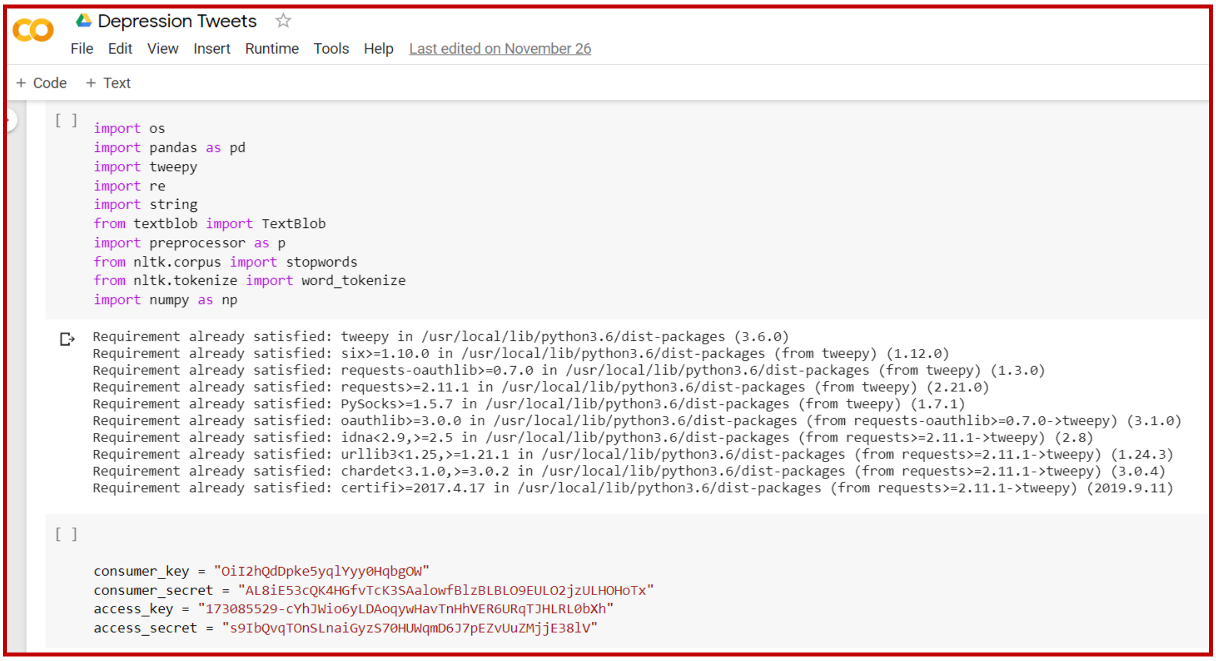
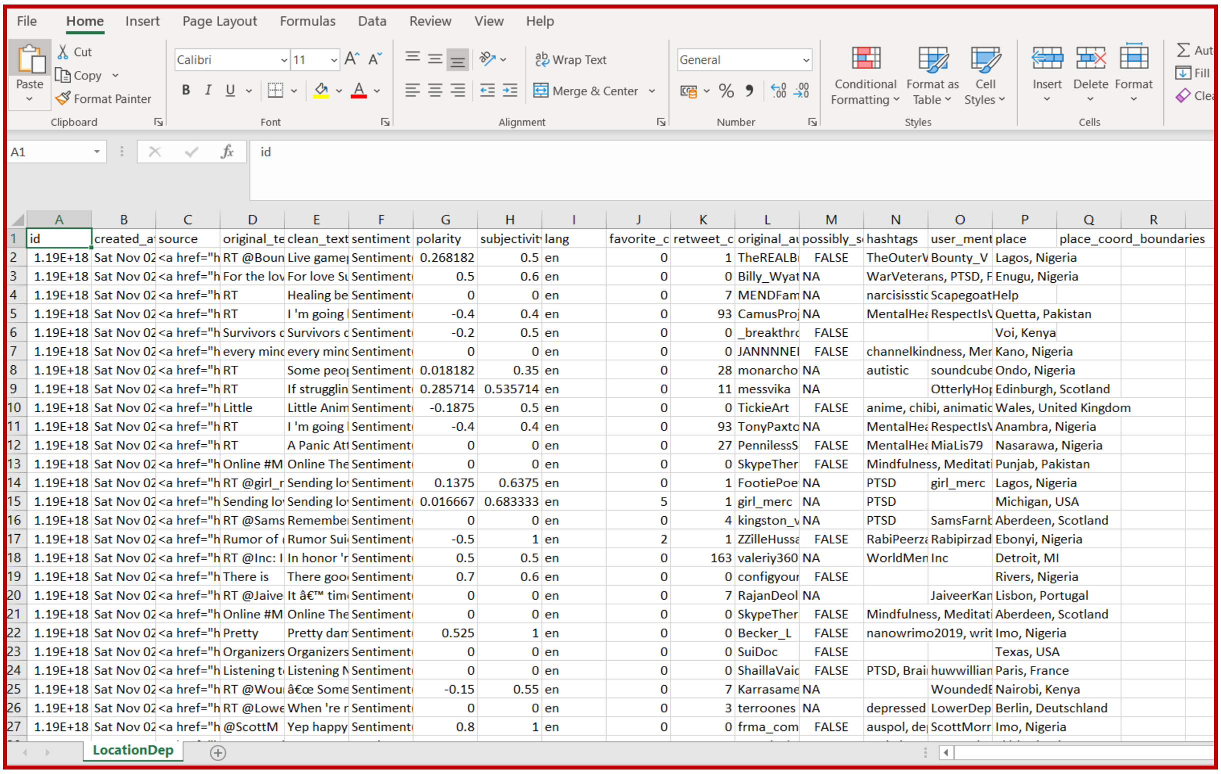



Figure 3. Google colaboratory interface for python

[1] https://colab.research.google.com/drive/1JzH1hnTpvbOuwUunUVJFqQDNI58peGVc#scrollTo=L9K1LTlqVfMm

The extracted tweets were extracted into excel as a comma delimited csv file. Figure 4 shows the extracted tweets in csv format.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | id | created_a | source | original_te | clean_text | sentiment | polarity | subjectivit | lang | favorite_c | retweet_c | original_au | possibly_s | hashtags | user_ment | place | place_coord_boundaries | |
| 2 | 1.19E+18 | Sat Nov 02 | <a href="h | RT @Boun | Live gamej | Sentiment | 0.268182 | 0.5 | en | 0 | 1 | TheREALB | FALSE | TheOuterV | Bounty_V | Lagos, Nigeria | | |
| 3 | 1.19E+18 | Sat Nov 02 | <a href="h | For the lov | For love Su | Sentiment | 0.5 | 0.6 | en | 0 | 0 | Billy_Wyat | NA | WarVeterans, PTSD, F | | Enugu, Nigeria | | |
| 4 | 1.19E+18 | Sat Nov 02 | <a href="h | RT | Healing be | Sentiment | 0 | 0 | en | 0 | 7 | MENDFam | NA | narcisisstic | ScapegoatHelp | | | |
| 5 | 1.19E+18 | Sat Nov 02 | <a href="h | RT | I 'm going | Sentiment | -0.4 | 0.4 | en | 0 | 93 | CamusProj | NA | MentalHea | RespectIs\ | Quetta, Pakistan | | |
| 6 | 1.19E+18 | Sat Nov 02 | <a href="h | Survivors c | Survivors c | Sentiment | -0.2 | 0.5 | en | 0 | 0 | _breakthro | FALSE | | | Voi, Kenya | | |
| 7 | 1.19E+18 | Sat Nov 02 | <a href="h | every minc | every minc | Sentiment | 0 | 0 | en | 0 | 0 | JANNNNEI | FALSE | channelkindness, Mer | | Kano, Nigeria | | |
| 8 | 1.19E+18 | Sat Nov 02 | <a href="h | RT | Some peoj | Sentiment | 0.018182 | 0.35 | en | 0 | 28 | monarcho | NA | autistic | soundcube | Ondo, Nigeria | | |
| 9 | 1.19E+18 | Sat Nov 02 | <a href="h | RT | If strugglin | Sentiment | 0.285714 | 0.535714 | en | 0 | 11 | messvika | NA | | OtterlyHoj | Edinburgh, Scotland | | |
| 10 | 1.19E+18 | Sat Nov 02 | <a href="h | Little | Little Anim | Sentiment | -0.1875 | 0.5 | en | 0 | 0 | TickieArt | FALSE | anime, chibi, animatic | | Wales, United Kingdom | | |
| 11 | 1.19E+18 | Sat Nov 02 | <a href="h | RT | I 'm going | Sentiment | -0.4 | 0.4 | en | 0 | 93 | TonyPaxto | NA | MentalHea | RespectIs\ | Anambra, Nigeria | | |
| 12 | 1.19E+18 | Sat Nov 02 | <a href="h | RT | A Panic Att | Sentiment | 0 | 0 | en | 0 | 27 | PennilessS | FALSE | MentalHea | MiaLis79 | Nasarawa, Nigeria | | |
| 13 | 1.19E+18 | Sat Nov 02 | <a href="h | Online #M | Online The | Sentiment | 0 | 0 | en | 0 | 0 | SkypeTher | FALSE | Mindfulness, Meditati | | Punjab, Pakistan | | |
| 14 | 1.19E+18 | Sat Nov 02 | <a href="h | RT @girl_r | Sending lo | Sentiment | 0.1375 | 0.6375 | en | 0 | 1 | FootiePoe | NA | PTSD | girl_merc | Lagos, Nigeria | | |
| 15 | 1.19E+18 | Sat Nov 02 | <a href="h | Sending lo | Sending lo | Sentiment | 0.016667 | 0.683333 | en | 5 | 1 | girl_merc | NA | PTSD | | Michigan, USA | | |
| 16 | 1.19E+18 | Sat Nov 02 | <a href="h | RT @Sams | Remembe | Sentiment | 0 | 0 | en | 0 | 4 | kingston_v | NA | PTSD | SamsFarnb | Aberdeen, Scotland | | |
| 17 | 1.19E+18 | Sat Nov 02 | <a href="h | Rumor of | Rumor Sui | Sentiment | -0.5 | 1 | en | 2 | 1 | ZZilleHussa | FALSE | RabiPeerza | Rabipirzad | Ebonyi, Nigeria | | |
| 18 | 1.19E+18 | Sat Nov 02 | <a href="h | RT @Inc: I | In honor 'r | Sentiment | 0.5 | 0.5 | en | 0 | 163 | valeriy360 | NA | WorldMen | Inc | Detroit, MI | | |
| 19 | 1.19E+18 | Sat Nov 02 | <a href="h | There is | There goo | Sentiment | 0.7 | 0.6 | en | 0 | 0 | configyour | FALSE | | | Rivers, Nigeria | | |
| 20 | 1.19E+18 | Sat Nov 02 | <a href="h | RT @Jaive | It â€™ tim | Sentiment | 0 | 0 | en | 0 | 7 | RajanDeol | NA | | JaiveerKan | Lisbon, Portugal | | |
| 21 | 1.19E+18 | Sat Nov 02 | <a href="h | Online #M | Online The | Sentiment | 0 | 0 | en | 0 | 0 | SkypeTher | FALSE | Mindfulness, Meditati | | Aberdeen, Scotland | | |
| 22 | 1.19E+18 | Sat Nov 02 | <a href="h | Pretty | Pretty dam | Sentiment | 0.525 | 1 | en | 0 | 0 | Becker_L | FALSE | nanowrimo2019, writ | | Imo, Nigeria | | |
| 23 | 1.19E+18 | Sat Nov 02 | <a href="h | Organizers | Organizers | Sentiment | 0 | 0 | en | 0 | 0 | SuiDoc | FALSE | | | Texas, USA | | |
| 24 | 1.19E+18 | Sat Nov 02 | <a href="h | Listening t | Listening N | Sentiment | 0 | 0 | en | 0 | 0 | ShaillaVaic | FALSE | PTSD, Brai | huwwillian | Paris, France | | |
| 25 | 1.19E+18 | Sat Nov 02 | <a href="h | RT @Wou | â€œ Some | Sentiment | -0.15 | 0.55 | en | 0 | 7 | Karrasame | NA | | WoundedE | Nairobi, Kenya | | |
| 26 | 1.19E+18 | Sat Nov 02 | <a href="h | RT @Lowe | When 're r | Sentiment | 0 | 0 | en | 0 | 3 | terroones | NA | depressed | LowerDep | Berlin, Deutschland | | |
| 27 | 1.19E+18 | Sat Nov 02 | <a href="h | @ScottM | Yep happy | Sentiment | 0.8 | 1 | en | 0 | 0 | frma_com | FALSE | auspol, de | ScottMorr | Imo, Nigeria | | |

Figure 4. Excel csv format of the extracted tweets.

The visualization of the results to the practitioners and comparison of the results were done using tableau. Figure 5 shows version of tableau used for this research work.


Figure 5. Tableau desktop version

# 3 Implementation

The implementation stage consists of the extraction of tweets, the exploratory data analysis, the pre-processing of the datasets, the merging the of the datasets, the extraction of features, the development of the models.

## 3.1 Data Extraction

The tweet extraction was done in python using google colaboratory and partial pre-processing was conducted on the extracted tweet in order to have clean text to generate the sentiment and calculate the polarity score of each tweet using the "NLP" package in python,

Step 1. All libraries in the "NLP" package were loaded as shown in figure 6

```
[ ] !pip install tweet-preprocessor

Requirement already satisfied: tweet-preprocessor in /usr/local/lib/python3.6/dist-packages (0.5.0)

[ ] !pip install tweepy

    import os
    import pandas as pd
    import tweepy
    import re
    import string
    from textblob import TextBlob
    import preprocessor as p
    from nltk.corpus import stopwords
    from nltk.tokenize import word_tokenize
    import numpy as np
```

Figure 6. Installation of the libraries

Step 2. The acquired authentication keys were parsed to the integrated development environment (IDE) in order to connect python to twitter database as shown in figure 7.

```
[ ]

    consumer_key = "OiI2hQdDpke5yqlYyy0HqbgOW"
    consumer_secret = "AL8iE53cQK4HGfvTcK3SAalowfBlzBLBLO9EULO2jzULHOHoTx"
    access_key = "173085529-cYhJWio6yLDAoqywHavTnHhVER6URqTJHLRL0bXh"
    access_secret = "s9IbQvqTOnSLnaiGyzS70HUWqmD6J7pEZvUuZMjjE38lV"

[ ]
    #pass twitter credentials to tweepy
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)
```

Figure 7. Integration of twitter authentication keys to python

Step 3. Columns were created for the tweets in order in which it will pulled into a csv format as shown in figure 8.

```
#columns of the csv file
COLS = ['id', 'created_at', 'source', 'original_text','clean_text', 'sentiment','polarity','subjectivity', 'lang',
        'favorite_count', 'retweet_count', 'original_author', 'possibly_sensitive', 'hashtags',
        'user_mentions', 'place', 'place_coord_boundaries']

#set two date variables for date range
start_date = '2019-10-30'
end_date = '2019-11-05'

# Happy Emoticons
emoticons_happy = set([
    ':-)', ':)', ';)', ':o)', ':]', ':3', ':c)', ':>', '=]', '8)', '=)', ':}',
    ':^)', ':-D', ':D', '8-D', '8D', 'x-D', 'xD', 'X-D', 'XD', '=-D', '=D',
    '=-3', '=3', ':-))', ":'-)", ":')", ':*', ':^*', '>:P', ':-P', ':P', 'X-P',
    'x-p', 'xp', 'XP', ':-p', ':p', '=p', ':-b', ':b', '>:)', '>;)', '>:-)',
    '<3'
    ])

# Sad Emoticons
emoticons_sad = set([
    ':L', ':-/', '>:/', ':S', '>:[', ':@', ':-(', ':[', ':-||', '=L', ':<',
    ':-[', ':-<', '=\\', '=/', '>:(', ':(', '>.<', ":'-(", ":'(", ':\\', ':-c',
    ':c', ':{', '>:\\', ';('
    ])

#Emoji patterns
emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
```

Figure 8. The set column, date range and emoji pattern of the tweets

Step 4. The location and origination of the tweets were acquired so as to know the origin of the tweets and the coordinates as shown in figure 9.

```
        #get location of the tweet
        try:
            location = status['user']['location']
        except TypeError:
            location = ''
        new_entry.append(location)

        try:
            coordinates = [coord for loc in status['place']['bounding_box']['coordinates'] for coord in loc]
        except TypeError:
            coordinates = None
        new_entry.append(coordinates)

        single_tweet_df = pd.DataFrame([new_entry], columns=COLS)
        df = df.append(single_tweet_df, ignore_index=True)
        csvFile = open(file, 'a' ,encoding='utf-8')
    df.to_csv(csvFile, mode='a', columns=COLS, index=False, encoding="utf-8")
```

Figure 9. The location of each tweets.

Step 5. In order to generate the sentiment and polarity of each tweets, libraries for natural language processing were used and partial pre-processing stages as shown in figure 10.

```
[ ] import nltk
    nltk.download("stopwords")

    nltk.download('punkt')
```

```
[ ] # clean_tweets()
    def clean_tweets(tweet):
        stop_words = set(stopwords.words('english'))
        word_tokens = word_tokenize(tweet)

        #Tweepy preprocessing the column, removing mentions
        #or RT sign in the beginning of the tweet
        tweet = re.sub(r':', '', tweet)
        tweet = re.sub(r',Ä¶', '', tweet)
        #replace consecutive non-ASCII characters with a space
        tweet = re.sub(r'[^\x00-\x7F]+',' ', tweet)


        #remove emojis from tweet
        tweet = emoji_pattern.sub(r'', tweet)

        #filter using NLTK library append it to a string
        filtered_tweet = [w for w in word_tokens if not w in stop_words]
        filtered_tweet = []

        #looping through conditions
        for w in word_tokens:
            #check tokens against stop words , emoticons and punctuations
            if w not in stop_words and w not in emoticons and w not in string.punctuation:
                filtered_tweet.append(w)
        return ' '.join(filtered_tweet)
        #print(word_tokens)
        #print(filtered_sentence)
```

Figure 10. Partial pre-processing for clean text.

Step 6. The sentiment, subjectivity and polarity of each text were calculated using library "textblob" in python as shown in figure 11.

```
#pass textBlob method for sentiment calculations
blob = TextBlob(filtered_tweet)
Sentiment = blob.sentiment

#seperate polarity and subjectivity in to two variables
polarity = Sentiment.polarity
subjectivity = Sentiment.subjectivity

#new entry append
new_entry += [status['id'], status['created_at'],
              status['source'], status['text'],filtered_tweet, Sentiment,polarity,subjectivity, status['lang'],
              status['favorite_count'], status['retweet_count']]

#to append original author of the tweet
new_entry.append(status['user']['screen_name'])

try:
    is_sensitive = status['possibly_sensitive']
except KeyError:
    is_sensitive = None
new_entry.append(is_sensitive)

# hashtags and mentions are saved using comma separted
hashtags = ", ".join([hashtag_item['text'] for hashtag_item in status['entities']['hashtags']])
new_entry.append(hashtags)
mentions = ", ".join([mention['screen_name'] for mention in status['entities']['user_mentions']])
new_entry.append(mentions)
```

Figure 11. Sentiment, subjectivity and polarity calculation of each tweets

Step 7. The tweets were harvested using hashtags relating to depression then saved out of the google colaboratory in a csv format as shown in figure 12.

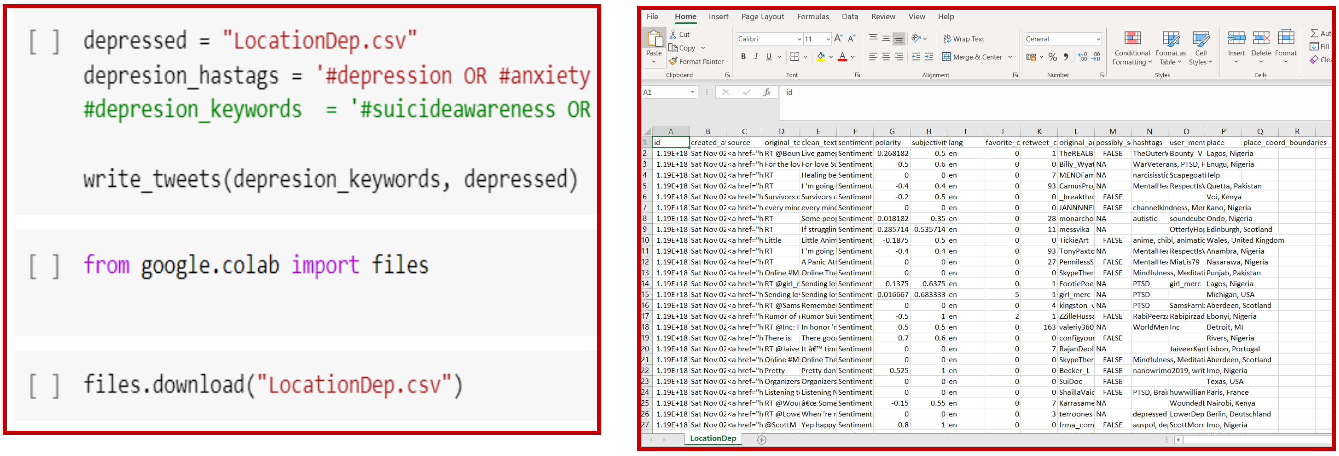
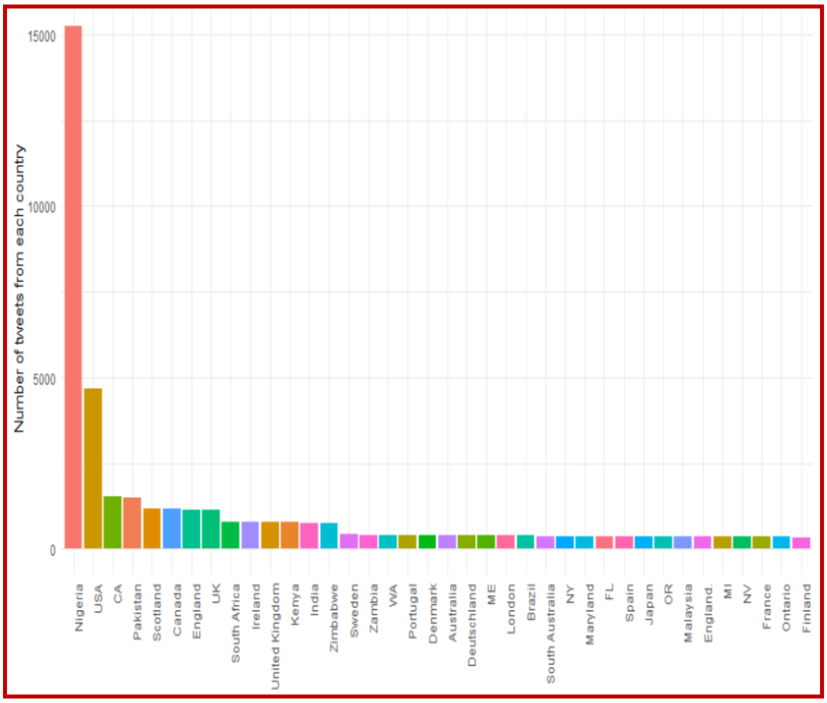


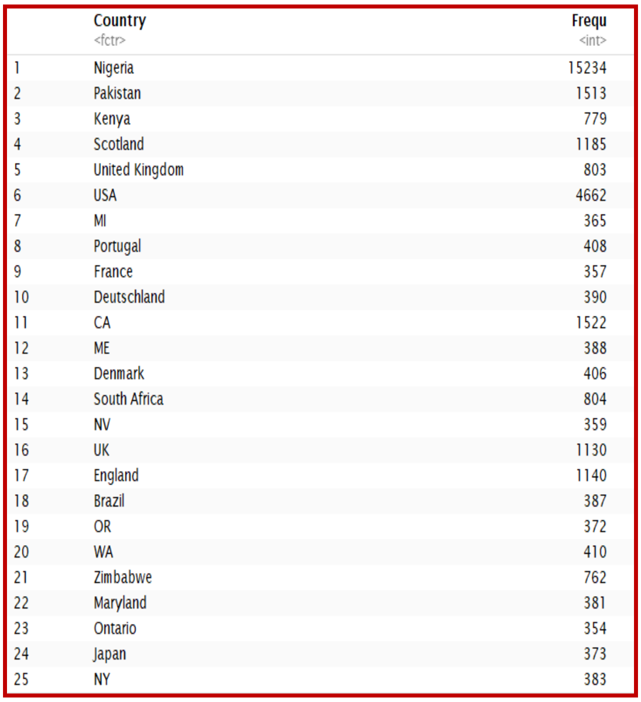

Figure 12. Hashtags and saving process.

The downloaded data set was transferred to R-studio for remaining process such as exploratory data analysis, data pre-processing, data merging and model implementation.

## 3.2 Exploratory Data Analysis

The data was visualized in its original format in order to what it contains and its relevance, and explanatory data analysis was performed to understand the countries where the tweets originated from as shown in the figure 13.



| | Country <fctr> | Frequ <int> |
|---|---|---|
| 1 | Nigeria | 15234 |
| 2 | Pakistan | 1513 |
| 3 | Kenya | 779 |
| 4 | Scotland | 1185 |
| 5 | United Kingdom | 803 |
| 6 | USA | 4662 |
| 7 | MI | 365 |
| 8 | Portugal | 408 |
| 9 | France | 357 |
| 10 | Deutschland | 390 |
| 11 | CA | 1522 |
| 12 | ME | 388 |
| 13 | Denmark | 406 |
| 14 | South Africa | 804 |
| 15 | NV | 359 |
| 16 | UK | 1130 |
| 17 | England | 1140 |
| 18 | Brazil | 387 |
| 19 | OR | 372 |
| 20 | WA | 410 |
| 21 | Zimbabwe | 762 |
| 22 | Maryland | 381 |
| 23 | Ontario | 354 |
| 24 | Japan | 373 |
| 25 | NY | 383 |

Figure 13. Visualization of the origination of tweet

The dataset was converted into a corpus in order to do more EDA especially word cloud and frequency of words in the tweet which has been reported in the technical report as well as class imbalance in the tweet which prompted an introduction of  new dataset called sentiment 140 which positive tweet was extracted from it to balance the negative tweets from the extracted dataset instead of a synthetic minority oversampling technique (SMOTE). Figure 14 shows the class imbalance in the extracted tweets and the balanced one with positive tweets from sentiment 140 dataset.

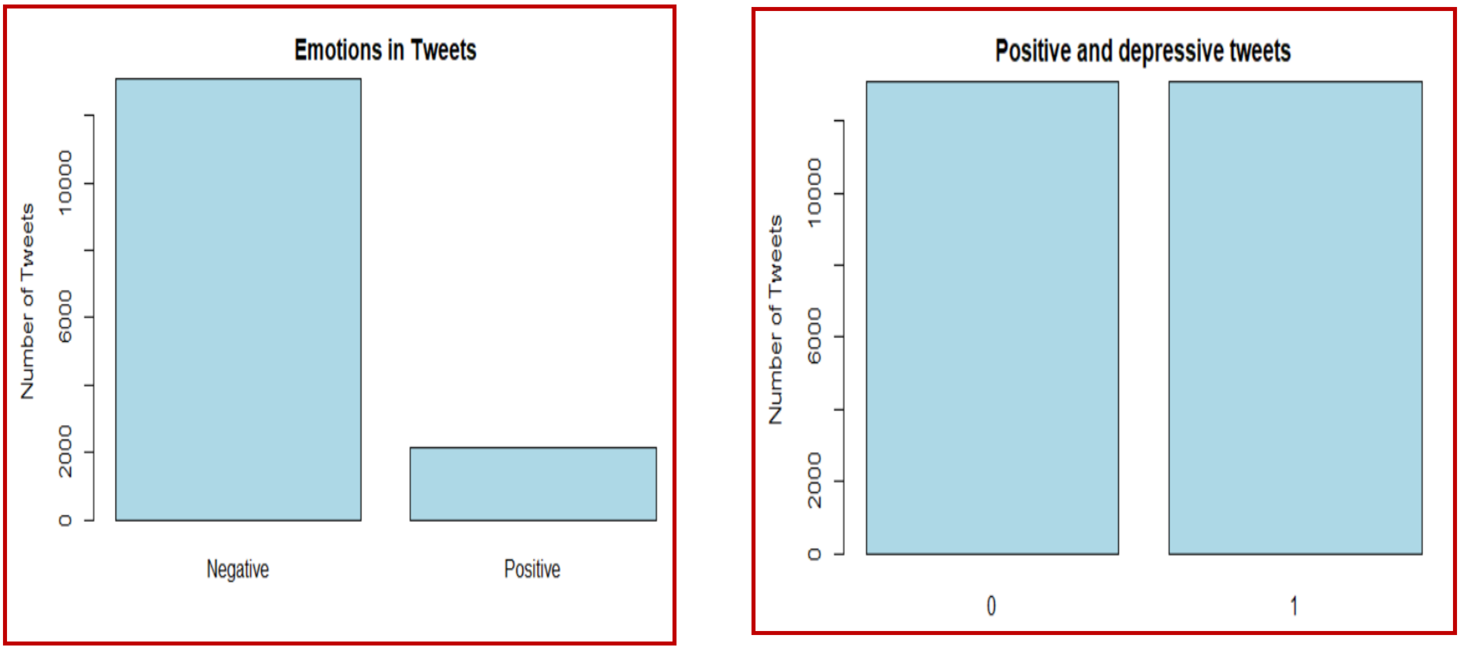



Figure 14. Class imbalance and balancing of the tweets

## 3.3 Data Pre-Processing

After having a balanced dataset, proper pre-processing was conducted on the dataset which involves the following process as presented in Figure 15.

- Converting the text into a corpus
- Converting all text to lower case
- Removing URLs in the text
- Removing punctuation
- Removing stop words
- Removing whitespace
- Conversion to document term matrix

```
Preprocessing proper
```{r}
library(tm)
TweetsTablepre <- str_replace_all(string=TweetsTable$Tweets, pattern= "[&Ã¢â·¬Â¦â·¢Ã°Å¸Â¥Ã¯Â·â··  Â¸Â·  Â¤Â·Â·]"
replacement= "")
# build a corpus
TweetsTableCorpus <- Corpus(VectorSource(TweetsTablepre))
# convert to lower case
TweetsTableCorpus <- tm_map(TweetsTableCorpus, content_transformer(tolower))
# remove URLs
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
TweetsTableCorpus <- tm_map(TweetsTableCorpus, content_transformer(removeURL))
# remove anything other than English letters or space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
TweetsTableCorpus <- tm_map(TweetsTableCorpus, content_transformer(removeNumPunct))
# remove stopwords
myStopwords <- c(setdiff(stopwords('english'), c("r", "big")),
"use", "see", "used", "via", "amp")
TweetsTableCorpus <- tm_map(TweetsTableCorpus, removeWords, myStopwords)
# remove extra whitespace
TweetsTableCorpus <- tm_map(TweetsTableCorpus, stripWhitespace)


# converting corpus to data frame
TweetsTablepos = data.frame(Tweets = sapply(TweetsTableCorpus, as.character), stringsAsFactors = FALSE)
TweetsTablepos = as.data.frame(cbind(TweetsTable$polarity, TweetsTablepos$Tweets))
TweetsTablepos$ID <- seq.int(nrow(TweetsTablepos))
TweetsTablepost = as.data.frame(TweetsTablepos)
colnames(TweetsTablepost) = c("Polarity","Tweets","ID")
TweetsTablepost$Tweets = as.character(TweetsTablepost$Tweets)
```

Figure 15. Pre-processing of the dataset

## 3.4 Modelling

The DTM contain sparse data i.e words that appeared 0.5% of the whole documents which were removed and sparsed data were splitted in 80:20 ratio of the train data and test data which were use to train and test seven algorithm as presented in figure 16.

```
#get the data as a corpus
dataTweetsTable = Corpus(VectorSource(TweetsTablepost$Tweets))

#a document term matrix
dataTweetsTabledtm <- DocumentTermMatrix(dataTweetsTable)

#remove sparse data
sparseData <- removeSparseTerms(dataTweetsTabledtm, sparse=0.995)


# Convert to data frame
sparsedf <- as.data.frame(as.matrix(sparseData))

#add the dependent variable
sparsedf$Polarity = TweetsTablepost$Polarity

colnames(sparsedf) <- make.names(colnames(sparsedf))

#split the dataset
split <- sample.split(sparsedf$Polarity, SplitRatio = 0.8)
trainSparse <- subset(sparsedf, split==TRUE)
testSparse <- subset(sparsedf, split==FALSE)
```

Figure 16. Sparsity and Split ratio
```

The algorithm used on the sparse data are linear discriminant analysis (LDA), random forest (RF), classification and regression tree (CART), extreme gradient boosting (XGboost), adaptive boosting (AdaBoost), regularized generalized linear model (GLMNET and c5.0 decision tree (C50) as presented in Figure 17.

```
#build models
control <- trainControl(method='cv', number=4)
metric <- 'Accuracy'

#CART
set.seed(101)
tweet.cart <- train(Polarity~., data= trainSparse, method='rpart',trControl=control, metric=metric)

# LDA
set.seed(101)
tweet.lda <- train(Polarity ~ ., data=trainSparse, method='lda',
                   trControl=control, metric=metric)

# RF
set.seed(101)
tweet.rf <- train(Polarity~., data=trainSparse, method='ranger',
                   trControl=control, metric=metric)
#C5.0
tweet.C50 <- train(Polarity~., data=trainSparse, method='C5.0',
                   trControl=control, metric=metric)

#xgboost
tweet.xgboost <- train(Polarity~., data=trainSparse, method='xgbTree',
                   trControl=control, metric=metric)
#adaboost
tweet.adaboost <- train(Polarity~., data=trainSparse, method='AdaBag',
                   trControl=control, metric=metric)

#glm_net
tweet.glmnet <- train(Polarity~., data=trainSparse, method='glmnet',
                   trControl=control, metric=metric)

tweet.results <- resamples(list(lda=tweet.lda, Xgboost=tweet.xgboost, cart=tweet.cart, rf=tweet.rf,AdaBag
=tweet.adaboost,glmnet =tweet.glmnet, c50=tweet.C50))

summary(tweet.results)
```

Figure 17. Seven algorithms used on sparse data

The DTM was tokenized to further extract features such as term frequency-inverse document frequency, n-grams and hash which were used to train regularized generalized linear model. Figure 18 shows how the tweets were tokenized.

```
Tokenization
```{r}
#Tokenization
library(text2vec)
library(glmnet)
library(tidyverse)
tok_fun = word_tokenizer

it_train = itoken(tweets_train$Tweets,
                  tokenizer = tok_fun,
                  progressbar = TRUE)

it_test = itoken(tweets_test$Tweets,
                  tokenizer = tok_fun,
                  progressbar = TRUE)
```
```

Figure 18. Tokenization

Term frequency inverse document frequency was extracted from the tokenized tweets and was used to train a glmnet as presented in Figure 19 and 20

```
Vocab and DTM
```{r}
#Vocab and DTM
library(text2vec)
tweetsvocab = create_vocabulary(it_train)
tweetsvectorizer = vocab_vectorizer(tweetsvocab)
tweets_traindtm = create_dtm(it_train, tweetsvectorizer)

#fit the TF-IDF to the train data
tfidf <- TfIdf$new()
# fit the model to the train data and transform it with the fitted model
tweets_traintfidf <- fit_transform(tweets_traindtm, tfidf)
# apply pre-trained tf-idf transformation to test data
tweets_testtfidf  <- create_dtm(it_test, tweetsvectorizer) %>%
  transform(tfidf)
```
```

Figure 19. Term frequency-inverse document frequency

```
Train classifier GLM_NET with TF_IDF
```{r}
#Train classifier
#cross validation
n_folds = 10

glmnet_classifier = glmnet::cv.glmnet(
  x = tweets_traintfidf,
  y = tweets_train[["Polarity"]],
  # set binary classification
  family = 'binomial',
  # L1 penalty
  alpha = 1,
  # interested in the area under ROC curve
  type.measure = "auc",
  # 4-fold cross-validation
  nfolds = n_folds,
  # high value is less accurate, but has faster training
  thresh = 1e-3,
  # again lower number of iterations for faster training
  maxit = 1e3
  )

print(paste("max AUC(Area under the curve) =", round(max(glmnet_classifier$cvm),4)))
library(caret)
#install.packages("pROC")
library(pROC)

#predict the test set
glmpred <- predict(glmnet_classifier, tweets_testtfidf, type = 'response')[ ,1]
auc(as.numeric(tweets_test$Polarity), glmpred)

#predict using class labels
class_labels = predict(glmnet_classifier,tweets_testtfidf , type = 'class') %>%
  as.factor()
```

Figure 20. Glmnet classifier with Term frequency-inverse document frequency
```

N-gram was extracted from the tokenized tweets and was used to train a glmnet as presented in Figure 21 and 22.

```
GLM_NET with NGRAMS
```{r}
vocabNgrams = create_vocabulary(it_train, ngram = c(1L, 2L))
vocabNgrams = prune_vocabulary(vocabNgrams, term_count_min = 10,doc_proportion_max = 0.5)

bigram_vectorizer = vocab_vectorizer(vocabNgrams)

Ngrams_train = create_dtm(it_train, bigram_vectorizer)
```

Figure 21. Vocabulary n-gram

```
Nglmnet_classifier = cv.glmnet(x = Ngrams_train, y = tweets_train[["Polarity"]],
                    family = 'binomial',
                    alpha = 1,
                    type.measure = "auc",
                    nfolds = n_folds,
                    thresh = 1e-3,
                    maxit = 1e3)
print(paste("max AUC =", round(max(Nglmnet_classifier$cvm),4)))

#Test dataset
Ngrams_test = create_dtm(it_test, bigram_vectorizer)

#predict using class labels
Nclass_labels = predict(Nglmnet_classifier,Ngrams_test , type = 'class') %>%
  as.factor()
```

Figure 22. Glmnet classifier with vocabulary n-gram

Hash was extracted from the tokenized tweets and was used to train a glmnet as presented in Figure 23 and 24.

```
GLM_NET with HSL
```{r}
HSLvectorizer = hash_vectorizer(hash_size = 2 ^ 14, ngram = c(1L, 2L))

HSL_train = create_dtm(it_train, HSLvectorizer)
```

Figure 23. Hash feature

```
Hglmnet_classifier = cv.glmnet(x = HSL_train, y = tweets_train[["Polarity"]],
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "auc",
                               nfolds = n_folds,
                               thresh = 1e-3,
                               maxit = 1e3)
print(paste("max AUC =", round(max(Hglmnet_classifier$cvm),4)))

#Test dataset
HSL_test = create_dtm(it_test, HSLvectorizer)

#predict using class labels
Hclass_labels = predict(Hglmnet_classifier,HSL_test , type = 'class') %>%
  as.factor()
```

Figure 24. Glmnet classifier with hash

The tokenized dataset was then normalized in order to increase the precision, recall and f-measure of the model, Figure 25 presents the normalization process.

```
GLM_Net with Normalization of DTM_train and test
```{r}
TFIDFNorm_train = normalize(tweets_traintfidf, "l1")
NgramsNorm_train = normalize(Ngrams_train,"l1")
HSLNorm_train = normalize(HSL_train,"l1")
```

Figure 25. Normalization of the tokenized tweets

Glmnet with all the feature were then trained on the normalized-tokenized tweets as shown in figure 26, 27 and 28.

```
#HSL
NHglmnet_classifier = cv.glmnet(x = HSLNorm_train, y = tweets_train[["Polarity"]],
                               family = 'binomial',
                               alpha = 1,
                               type.measure = "auc",
                               nfolds = n_folds,
                               thresh = 1e-3,
                               maxit = 1e3)
print(paste("max AUC =", round(max(NHglmnet_classifier$cvm),4)))


#predict using class labels
NHclass_labels = predict(NHglmnet_classifier,HSL_test , type = 'class') %>%
  as.factor()
```

Figure 26. Glmnet classifier with hash on normalized-tokenized tweets
```

```
#Ngrams
NNglmnet_classifier = cv.glmnet(x = NgramsNorm_train, y = tweets_train[["Polarity"]],
                  family = 'binomial',
                  alpha = 1,
                  type.measure = "auc",
                  nfolds = n_folds,
                  thresh = 1e-3,
                  maxit = 1e3)
print(paste("max AUC =", round(max(NNglmnet_classifier$cvm),4)))


#predict using class labels
NNclass_labels = predict(NNglmnet_classifier,Ngrams_test , type = 'class') %>%
  as.factor()
```

Figure 27. Glmnet classifier with n-gram on normalized-tokenized tweets

```
#TFIDF
NTFglmnet_classifier = glmnet::cv.glmnet(
  x = TFIDFNorm_train,
  y = tweets_train[["Polarity"]],
  family = 'binomial',
  alpha = 1,
  type.measure = "auc",
  nfolds = n_folds,
  thresh = 1e-3,
  maxit = 1e3
  )
print(paste("max AUC(Area under the curve) =", round(max(NTFglmnet_classifier$cvm),4)))

#predict using class labels
NTFclass_labels = predict(NTFglmnet_classifier,tweets_testtfidf , type = 'class') %>%
  as.factor()
```

Figure 28. Glmnet classifier with n-gram on normalized-tokenized tweets

The above implementation further gave results which are evaluated and validated by the evaluation metrics adopted from the literature. The next chapter shows the results from the implementation.

# 4 Results

The result of the implementation of the seven algorithms on sparse data and regularised generalized linear model on tokenized and normalized tweets using precision, recall and f-measure utilizing library “caret” in R-studio for the confusion matrix and 10 fold cross validation was used to validate the result so that the result are not by chance.#

## 4.1 Results on Sparse Tweet

### Classification and Regression Tree (CART)

The result of the implementation of Classification and Regression Tree (CART) and its best tune parameter is presented in figure 29.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1  469   27
         2 2150 2592

               Accuracy : 0.5844
                 95% CI : (0.5709, 0.5978)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1688

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.94556
                 Recall : 0.17908
                     F1 : 0.30112
             Prevalence : 0.50000
         Detection Rate : 0.08954
   Detection Prevalence : 0.09469
      Balanced Accuracy : 0.58438

       'Positive' Class : 1
```

| | trials | model | winnow |
|---|---|---|---|
| 1 | 1 | tree | FALSE |

Figure 29. Classification and Regression Tree and its best tune parameter

## Linear Discriminant Analysis (LDA)

The result of the implementation of Linear Discriminant Analysis (LDA) and its best tune parameter is presented in figure 30.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2262  728
         2  357 1891

               Accuracy : 0.7929
                 95% CI : (0.7816, 0.8038)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5857

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.7565
                 Recall : 0.8637
                     F1 : 0.8066
             Prevalence : 0.5000
         Detection Rate : 0.4318
   Detection Prevalence : 0.5708
      Balanced Accuracy : 0.7929

       'Positive' Class : 1
```

| | parameter |
|---|---|
| 1 | none |

Figure 30. Linear Discriminant Analysis Tree and its best tune parameter

## Random Forest (RF)

The result of the implementation of Random Forest (RF) and its best tune parameter is presented in figure 31.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2310  615
         2  309 2004

               Accuracy : 0.8236
                 95% CI : (0.813, 0.8338)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6472

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.7897
                 Recall : 0.8820
                     F1 : 0.8333
             Prevalence : 0.5000
         Detection Rate : 0.4410
   Detection Prevalence : 0.5584
      Balanced Accuracy : 0.8236

       'Positive' Class : 1
```

| | mtry | splitrule | min.node.size |
|---|---|---|---|
| 4 | 113 | extratrees | 1 |

Figure 31. Random Forest and its best tune parameter

### C5.0 Decision Tree (c50)

The result of the implementation of C5.0 Decision Tree (c50)and its best tune parameter is presented in figure 32.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2301  753
         2  318 1866

               Accuracy : 0.7955
                 95% CI : (0.7844, 0.8064)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5911

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.7534
                 Recall : 0.8786
                     F1 : 0.8112
             Prevalence : 0.5000
         Detection Rate : 0.4393
   Detection Prevalence : 0.5830
      Balanced Accuracy : 0.7955

       'Positive' Class : 1
```

| | trials | model | winnow |
|---|---|---|---|
| **1** | 1 | tree | FALSE |

Figure 32. C5.0 Decision Tree and its best tune parameter

### Regularized Generalized Linear Model (GLMNET)

The result of the implementation of Regularized Generalized Linear Model and its best tune parameter is presented in figure 33.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2256  739
         2  363 1880

               Accuracy : 0.7896
                 95% CI : (0.7783, 0.8006)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5792

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.7533
                 Recall : 0.8614
                     F1 : 0.8037
             Prevalence : 0.5000
         Detection Rate : 0.4307
   Detection Prevalence : 0.5718
      Balanced Accuracy : 0.7896

       'Positive' Class : 1
```

| | alpha | lambda |
|---|---|---|
| 2 | 0.1 | 0.00199113 |

Figure 33. Regularized Generalized Linear Model and its best tune parameter

## Adaptive Boosting (AdaBoost)

The result of the implementation of Adaptive Boosting (AdaBoost) and its best tune parameter is presented in figure 34.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1  468   25
         2 2151 2594

               Accuracy : 0.5846
                 95% CI : (0.5711, 0.598)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1691

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.94929
                 Recall : 0.17869
                     F1 : 0.30077
             Prevalence : 0.50000
         Detection Rate : 0.08935
   Detection Prevalence : 0.09412
      Balanced Accuracy : 0.58457

       'Positive' Class : 1
```

| | mfinal | maxdepth |
|---|---|---|
| 9 | 150 | 3 |

Figure 34. Adaptive Boosting and its best tune parameter

## Extreme Gradient Boosting (XGBoost)

The result of the implementation of Extreme Gradient Boosting (XGBoost) and its best tune parameter is presented in figure 35.

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2315  753
         2  304 1866

               Accuracy : 0.7982
                 95% CI : (0.7871, 0.809)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5964

 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.7546
                 Recall : 0.8839
                     F1 : 0.8141
             Prevalence : 0.5000
         Detection Rate : 0.4420
   Detection Prevalence : 0.5857
      Balanced Accuracy : 0.7982

       'Positive' Class : 1
```

| | nrounds | max_depth | eta | gamma | colsample_bytree | min_child_weight | subsample |
|---|---|---|---|---|---|---|---|
| 108 | 150 | 3 | 0.4 | 0 | 0.8 | 1 | 1 |

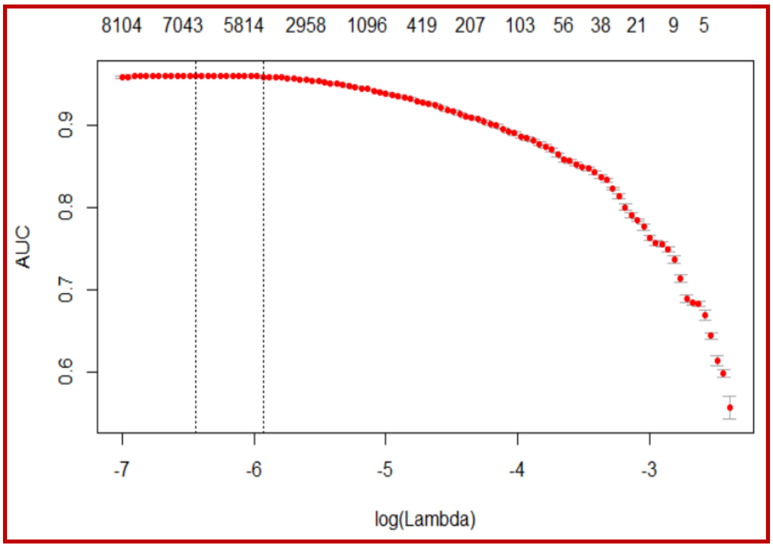
Figure 35. Extreme Gradient Boosting and its best tune parameter

## 4.2 Results on Tokenized and Normalized Tweets

The result of the implementation of Regularized Generalized Linear Model (GLMNET) on tokenized tweets and normalized tweets with extraction of term frequency-inverse document frequency (tfidf), vocabulary n-gram (n-gram), hashing (hsl).

**GLMNET with tokenized tweet using tfidf**

The result of is GLMNET with tokenized tweet using tfidf presented in figure 36.




```
[1] "max AUC(Area under the curve) = 0.9596"
NaNs producedNaNs produced[1] NaN
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2310  255
         2  308 2363

               Accuracy : 0.8925
                 95% CI : (0.8838, 0.9007)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.785

 Mcnemar's Test P-Value : 0.02841

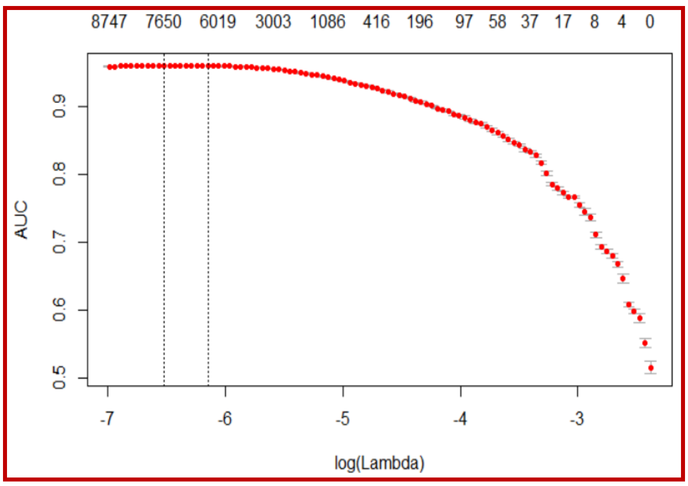
              Precision : 0.9006
                 Recall : 0.8824
                     F1 : 0.8914
             Prevalence : 0.5000
         Detection Rate : 0.4412
   Detection Prevalence : 0.4899
      Balanced Accuracy : 0.8925

       'Positive' Class : 1
```

Figure 36. Result of GLMNET with tokenized tweet using tfidf

**GLMNET with tokenized and normalized tweet using tfidf**

The result of is GLMNET with tokenized and normalized tweet using tfidf presented in figure 37.




```
[1] "max AUC(Area under the curve) = 0.9602"
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2364  332
         2  254 2286

               Accuracy : 0.8881
                 95% CI : (0.8792, 0.8965)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7762

 Mcnemar's Test P-Value : 0.001468

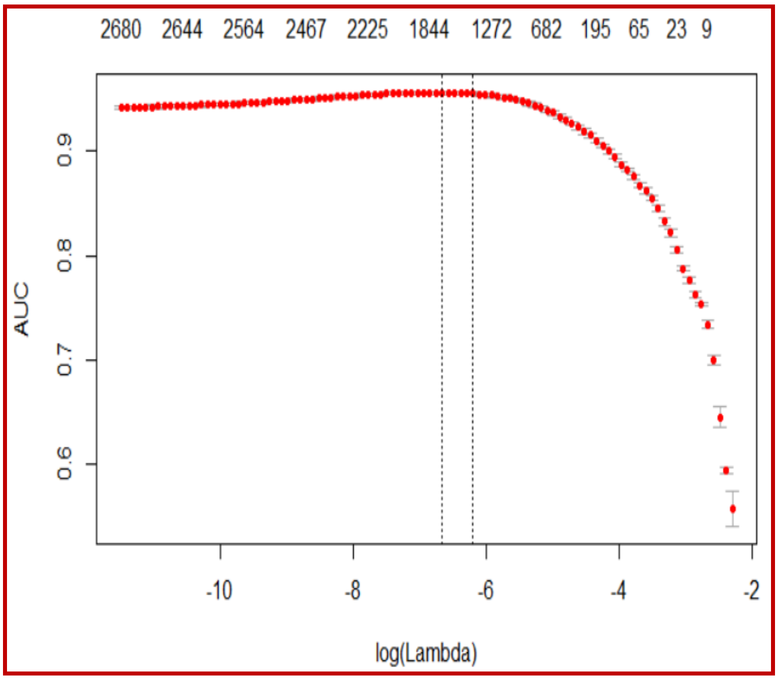
              Precision : 0.8769
                 Recall : 0.9030
                     F1 : 0.8897
             Prevalence : 0.5000
         Detection Rate : 0.4515
   Detection Prevalence : 0.5149
      Balanced Accuracy : 0.8881

       'Positive' Class : 1
```

Figure 37. Result of GLMNET with tokenized and normalized tweet using tfidf

**GLMNET with tokenized tweet using n-gram**

The result of is GLMNET with tokenized tweet using n-gram presented in figure 38.




```
100%[1] "max AUC = 0.956"

|==========================================
100%Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2219  276
         2  399 2342

               Accuracy : 0.8711
                 95% CI : (0.8617, 0.88)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7422

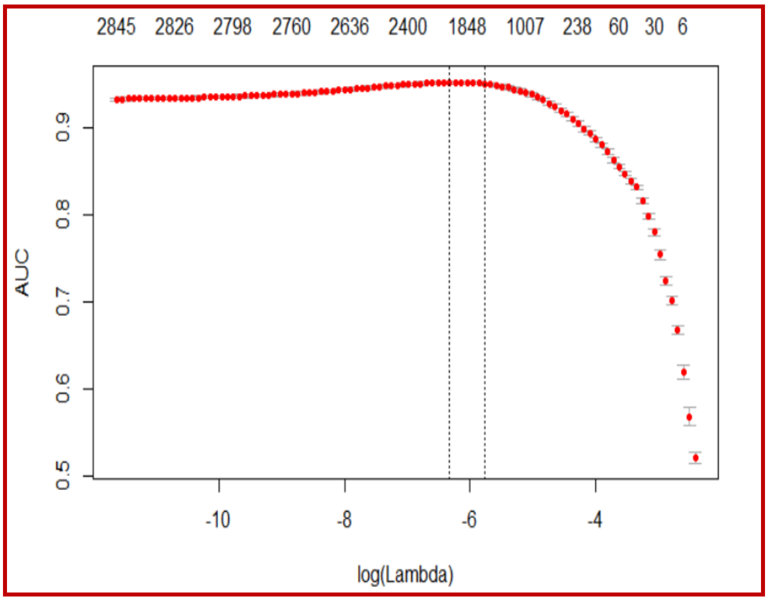
 Mcnemar's Test P-Value : 2.656e-06

              Precision : 0.8894
                 Recall : 0.8476
                     F1 : 0.8680
             Prevalence : 0.5000
         Detection Rate : 0.4238
   Detection Prevalence : 0.4765
      Balanced Accuracy : 0.8711
```

Figure 38. Result of GLMNET with tokenized tweet using n-gram

**GLMNET with tokenized and normalized tweet using n-gram**

The result of is GLMNET with tokenized tweet using n-gram presented in figure 39.




```
[1] "max AUC = 0.9529"
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2312  341
         2  306 2277

               Accuracy : 0.8764
                 95% CI : (0.8672, 0.8852)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7529

 Mcnemar's Test P-Value : 0.1813

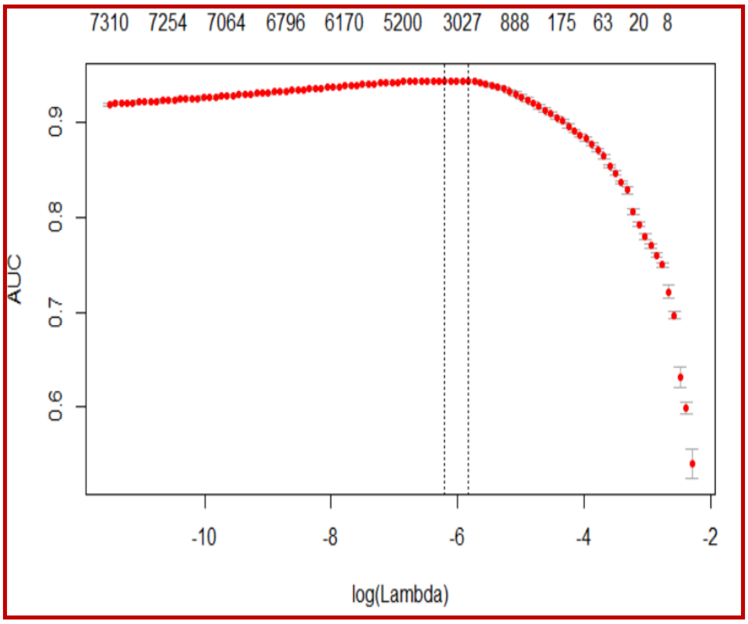
              Precision : 0.8715
                 Recall : 0.8831
                     F1 : 0.8773
             Prevalence : 0.5000
         Detection Rate : 0.4416
   Detection Prevalence : 0.5067
      Balanced Accuracy : 0.8764

       'Positive' Class : 1
```

Figure 39. Result of GLMNET with tokenized and normalized tweet using n-gram

### GLMNET with tokenized tweet using hsl

The result of is GLMNET with tokenized tweet using n-gram presented in figure 40.




```
100%Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2238  328
         2  380 2290

               Accuracy : 0.8648
                 95% CI : (0.8552, 0.8739)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7296

 Mcnemar's Test P-Value : 0.05528

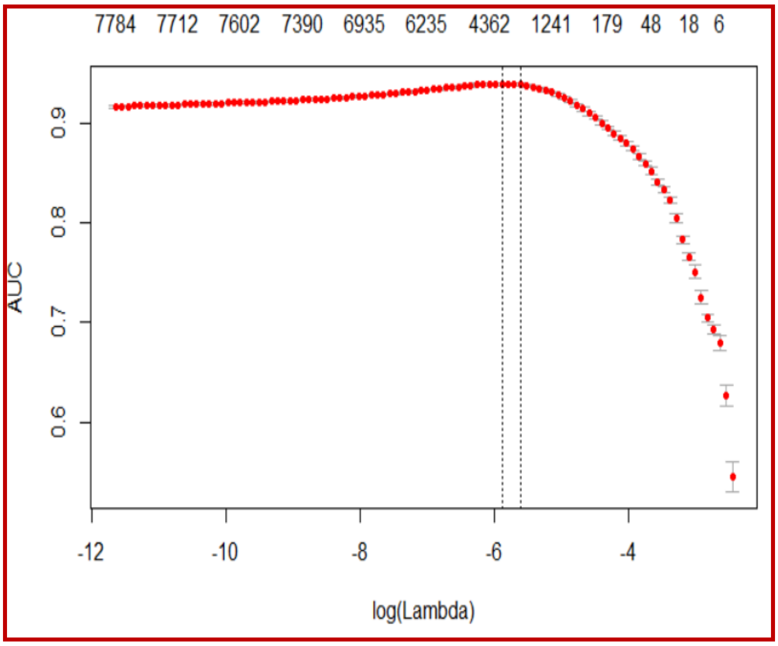
              Precision : 0.8722
                 Recall : 0.8549
                     F1 : 0.8634
             Prevalence : 0.5000
         Detection Rate : 0.4274
   Detection Prevalence : 0.4901
      Balanced Accuracy : 0.8648

       'Positive' Class : 1
```

Figure 40. Result of GLMNET with tokenized tweet using hsl

### GLMNET with tokenized and normalized tweet using hsl

The result of is GLMNET with tokenized tweet using n-gram presented in figure 40.




```
[1] "max AUC = 0.9397"
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 2272  353
         2  346 2265

               Accuracy : 0.8665
                 95% CI : (0.857, 0.8756)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.733

 Mcnemar's Test P-Value : 0.8205

              Precision : 0.8655
                 Recall : 0.8678
                     F1 : 0.8667
             Prevalence : 0.5000
         Detection Rate : 0.4339
   Detection Prevalence : 0.5013
      Balanced Accuracy : 0.8665

       'Positive' Class : 1
```

Figure 41. Result of GLMNET with tokenized and normalized tweet using hsl

## 4.3 Comparison of the Results with Existing Results

The best performing results from the implemented model were compared to the existing literature results and Table 1 and 2 shows the comparison.

Table 1. Existing results from Literature

| Author | Sample Size | Platform | Results |
|---|---|---|---|
| Tsugawa et.al (2015) | 209 | Twitter | Recall = 0.69 |
| Islam et.al (2018) | 7145 | Facebook | Recall = 0.98 |
| De Choudhry et.al (2013) | 476 | Twitter | Recall = 0.72 |

Table 2. Best model results from the implementation

| Best Models | Sample size | Result |
|---|---|---|
| Glmnet_TFIDF | 15,234 | Recall = 0.89 |
| Glmnet_TFIDF_Norm | 15,234 | Recall = 0.91 |
| Glmnet_Ngram_Norm | 15,234 | Recall = 0.89 |

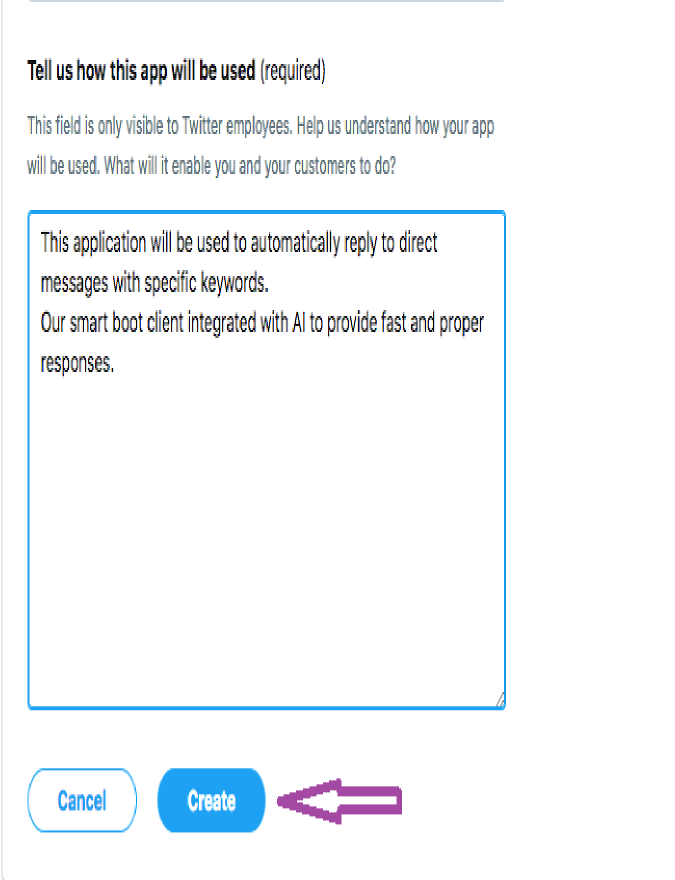
# References

De Choudhury, M., Gamon, M., Counts, S., and Horvitz, E. (2013) 'Predicting depression via social media', In ICWSM, page 2.

Google colaboratory (2019), Python 3 Notebook, Available at: https://colab.research.google.com/drive/1JzH1hnTpvbOuwUunUVJFqQDNI58peGVc#scrollTo=L9K1LTlqVfMm [Accessed 21 October 2019]

Islam, M.R., Kabir, M.A., Ahmed, A., Kamal, A.R.M., Wang, H. and Ulhaq, A. (2018) 'Depression detection from social network data using machine learning techniques', Health information science and systems, 6(1), p.8.

Tsugawa, S., Kikuchi, Y., Kishino, F., Nakajima, K., Itoh, Y., and Ohsaki, H. (2015) 'Recognizing depression from twitter activity', In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI '15, pages 3187–3196, New York, NY, USA. ACM.


Tell us how this app will be used (required)
This field is only visible to Twitter employees. Help us understand how your app will be used. What will it enable you and your customers to do?
This application will be used to automatically reply to direct messages with specific keywords.
Our smart boot client integrated with AI to provide fast and proper responses.
Cancel
Create