Configuration Manual

MSc Internship
Cyber Security

Tuvie Akpofure
Student ID: x18171028

School of Computing
National College of Ireland

Supervisor: Prof. Christos Grecos

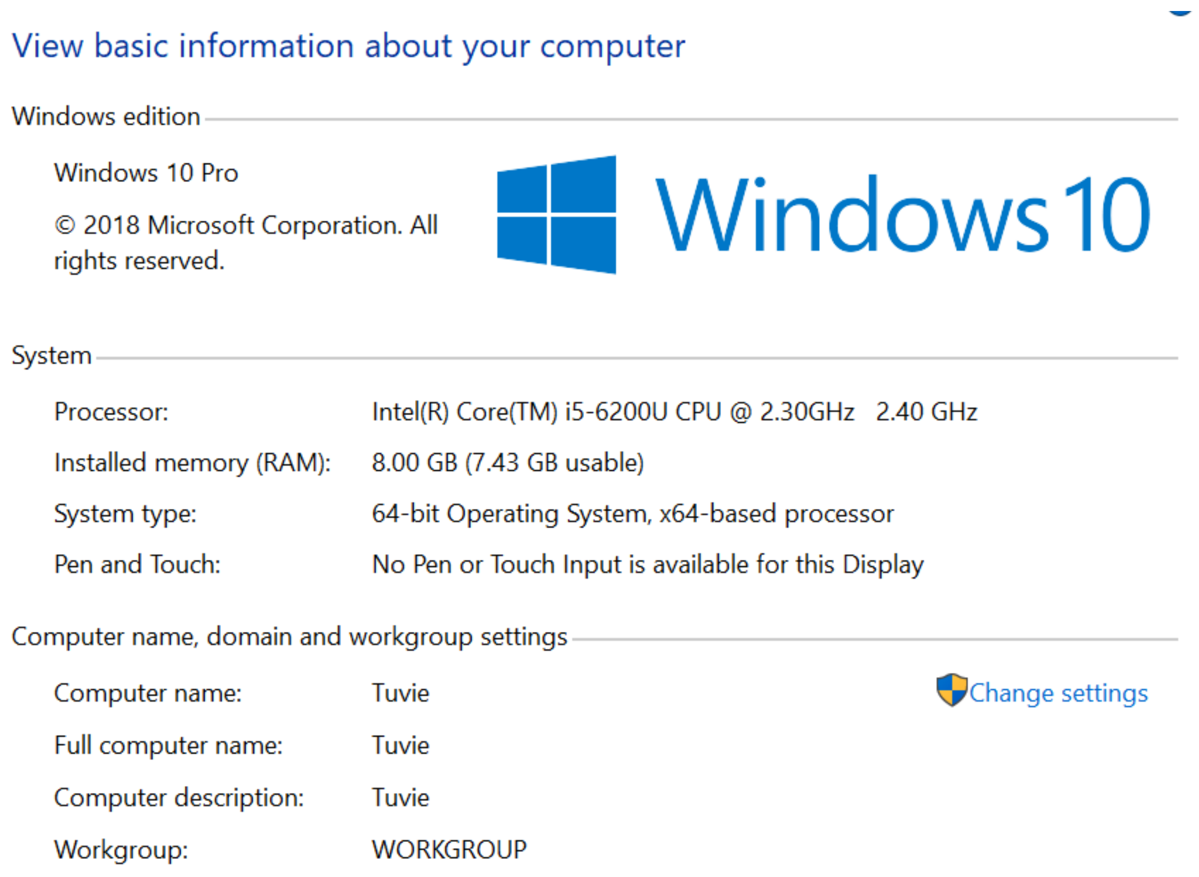
Configuration Manual

Tuvie Akpofure
Student ID: x18171028

The procedures outlined in this configuration manual are the steps that were taken in building an Automatic system for Identifying Hate Speech on Social media platforms using Machine Learning [1]. This manual can be used as a guide if this process is to be reproduced.

1 Hardware and Environment

The physical properties of the Hardware device used for this implementation include the following; Windows 10 operating system, 8G RAM, intel core i5 processor and 256 SSD. The image below shows the physical properties of the Hardware used for the implementation of the research work. It's advisable to use a hardware with a larger RAM size to increase the process.


View basic information about your computer

Windows edition

Windows 10 Pro

© 2018 Microsoft Corporation. All rights reserved.

System

Processor:	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
Installed memory (RAM):	8.00 GB (7.43 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Computer name, domain and workgroup settings

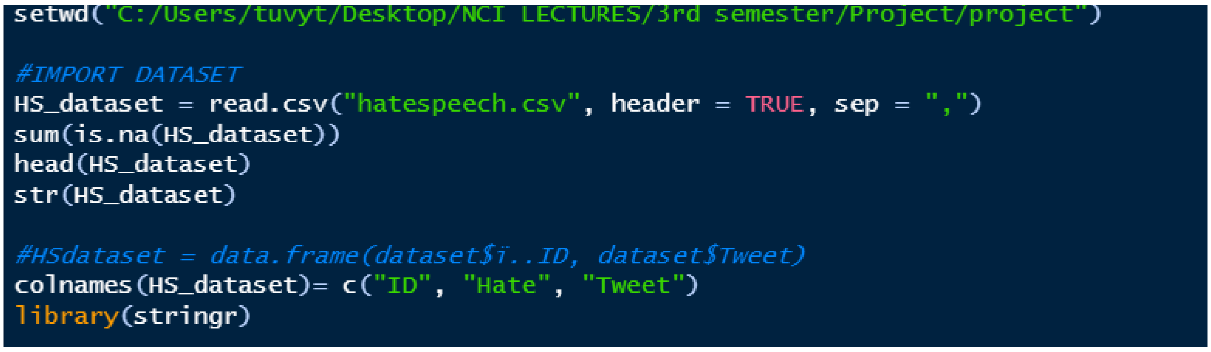
Computer name:	Tuvie
Full computer name:	Tuvie
Computer description:	Tuvie
Workgroup:	WORKGROUP

Fig 1: Hardware properties

Environment: We used R programming language to perform all steps and processes for this work. RStudio version 1.2.5019 was used as the environment for the execution of our work. RStudio and R programming provide packages and library that help in running each step. The necessary packages and libraries needed for all steps were installed and activated.

2 Data Collection

When carrying out classification problems, a dataset with information related to the problem to be solved should be acquired. For building our Hate speech detection model , we collected a dataset named Twitter Hate speech from Kaggle. It contained user tweets and labels for the tweet. After putting the dataset in the chosen folder and setting the working directory on RStudio, we used the read.csv function to import our dataset to RStudio since it is a .csv file. The lines of codes used for this process are below.



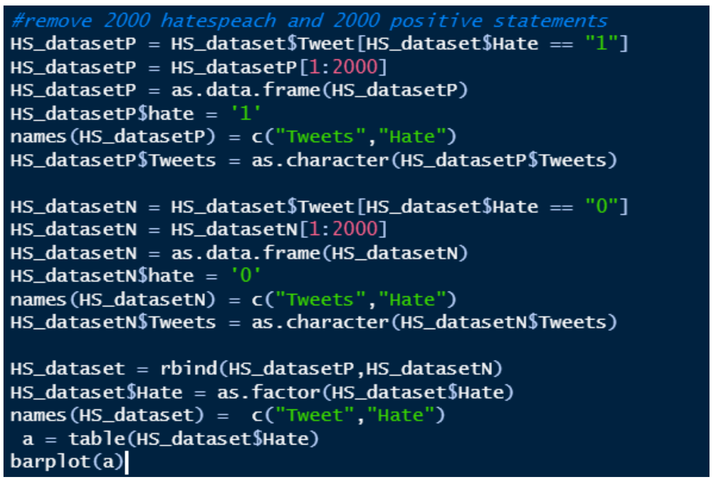
```
setwd("C:/Users/tuvyt/Desktop/NCI LECTURES/3rd semester/Project/project")

#IMPORT DATASET
HS_dataset = read.csv("hatespeech.csv", header = TRUE, sep = ",")
sum(is.na(HS_dataset))
head(HS_dataset)
str(HS_dataset)

#HSdataset = data.frame(dataset$ï..ID, dataset$Tweet)
colnames(HS_dataset)= c("ID", "Hate", "Tweet")
library(stringr)
```

Fig 2: Data importation to RStudio

From the figure above, assign the dataset to a variable name, then check for missing values in the dataset using the `sum(is.na()),` which checks and removes the missing values. You also use the `head()` and `str()` function to check if the dataset loaded correctly with the right properties. The `colnames() = c()` should be used to set the column names to ID, Hate and Tweet or any title of your choice.
The Hardware device used for this work kept running into errors when we were working with a larger portion of the dataset. Therefore, we randomly selected 4000 positive and negative tweets from the dataset to help our analysis run smoothly. It is advisable to use a system and environment with greater properties.



```
#remove 2000 hatespeach and 2000 positive statements
HS_datasetP = HS_dataset$Tweet[HS_dataset$Hate == "1"]
HS_datasetP = HS_datasetP[1:2000]
HS_datasetP = as.data.frame(HS_datasetP)
HS_datasetP$hate = '1'
names(HS_datasetP) = c("Tweets","Hate")
HS_datasetP$Tweets = as.character(HS_datasetP$Tweets)

HS_datasetN = HS_dataset$Tweet[HS_dataset$Hate == "0"]
HS_datasetN = HS_datasetN[1:2000]
HS_datasetN = as.data.frame(HS_datasetN)
HS_datasetN$hate = '0'
names(HS_datasetN) = c("Tweets","Hate")
HS_datasetN$Tweets = as.character(HS_datasetN$Tweets)

HS_dataset = rbind(HS_datasetP,HS_datasetN)
HS_dataset$Hate = as.factor(HS_dataset$Hate)
names(HS_dataset) =  c("Tweet","Hate")
 a = table(HS_dataset$Hate)
barplot(a)
```

Fig 3: Reduced Dataset

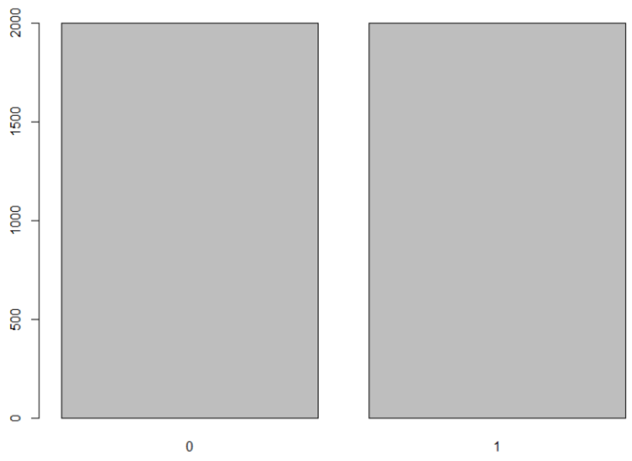
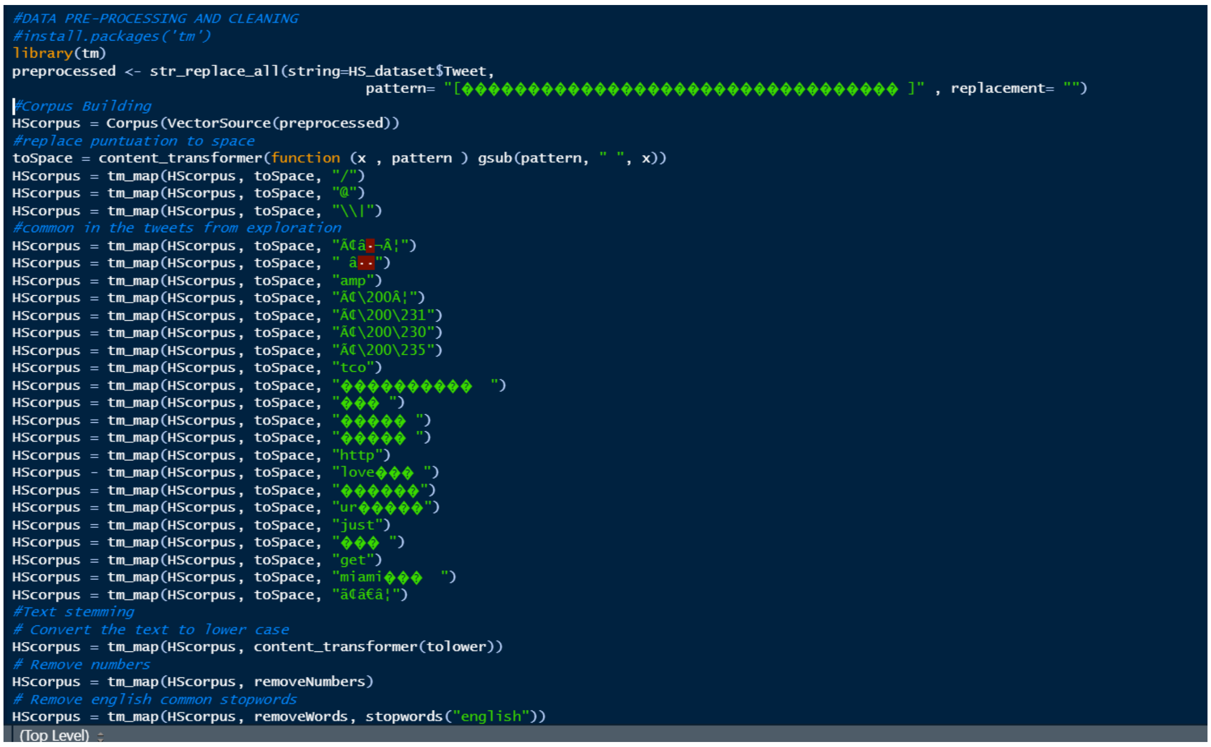


Fig 4: Barplot showing dataset balance

3 Data Pre-processing and Cleaning

In accordance to our flow of plan data pre-processing is the next step to perform for our implementation. Data pre-processing is the processing in which certain irrelevant features found in the dataset are removed to help a classification model run easily [2]. It converts a dataset to the accepted format in which machine learning models understand. The figure below shows how the pre-processing and cleaning on the dataset was done.



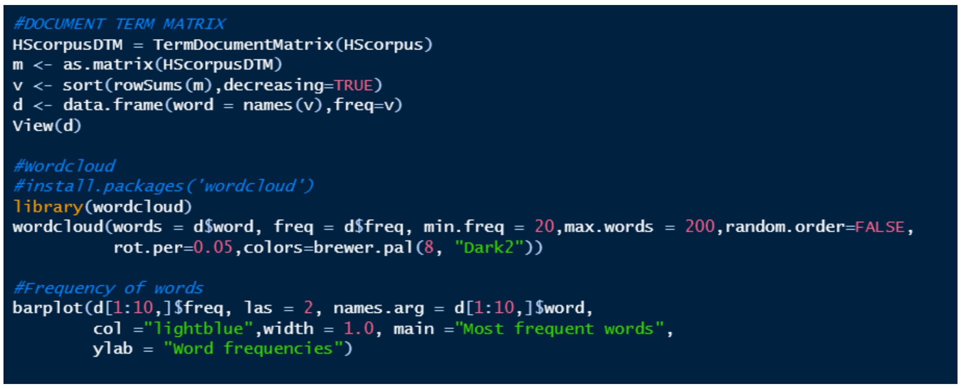
```
#DATA PRE-PROCESSING AND CLEANING
#install.packages('tm')
library(tm)
preprocessed <- str_replace_all(string=HS_dataset$Tweet,
                                             pattern= "[���������������������������������� ]" , replacement= "")
#Corpus Building
HScorpus = Corpus(VectorSource(preprocessed))
#replace puntuation to space
toSpace = content_transformer(function (x , pattern ) gsub(pattern, " ", x))
HScorpus = tm_map(HScorpus, toSpace, "/")
HScorpus = tm_map(HScorpus, toSpace, "@")
HScorpus = tm_map(HScorpus, toSpace, "\\|")
#common in the tweets from exploration
HScorpus = tm_map(HScorpus, toSpace, "Ã¢â·¬Â¦")
HScorpus = tm_map(HScorpus, toSpace, " â··")
HScorpus = tm_map(HScorpus, toSpace, "amp")
HScorpus = tm_map(HScorpus, toSpace, "Ã¢\200Â¦")
HScorpus = tm_map(HScorpus, toSpace, "Ã¢\200\231")
HScorpus = tm_map(HScorpus, toSpace, "Ã¢\200\230")
HScorpus = tm_map(HScorpus, toSpace, "Ã¢\200\235")
HScorpus = tm_map(HScorpus, toSpace, "tco")
HScorpus = tm_map(HScorpus, toSpace, "����������  ")
HScorpus = tm_map(HScorpus, toSpace, "��� ")
HScorpus = tm_map(HScorpus, toSpace, "����� ")
HScorpus = tm_map(HScorpus, toSpace, "����� ")
HScorpus = tm_map(HScorpus, toSpace, "http")
HScorpus - tm_map(HScorpus, toSpace, "love��� ")
HScorpus = tm_map(HScorpus, toSpace, "������")
HScorpus = tm_map(HScorpus, toSpace, "ur�����")
HScorpus = tm_map(HScorpus, toSpace, "just")
HScorpus = tm_map(HScorpus, toSpace, "��� ")
HScorpus = tm_map(HScorpus, toSpace, "get")
HScorpus = tm_map(HScorpus, toSpace, "miami���  ")
HScorpus = tm_map(HScorpus, toSpace, "ã¢â€â¦")
#Text stemming
# Convert the text to lower case
HScorpus = tm_map(HScorpus, content_transformer(tolower))
# Remove numbers
HScorpus = tm_map(HScorpus, removeNumbers)
# Remove english common stopwords
HScorpus = tm_map(HScorpus, removeWords, stopwords("english"))
```

Fig 5: Data Pre-processing and Cleaning

For the data pre-processing and cleaning phase, you installthe Text mining package (tm). The `install.packages()` function should be used to install all packages on R, then the `library()` function should be used to automatically activate the installed package on R. You start the pre-processing by replacing any special character which occurs a lot on your dataset with a white space. Then build a corpus and use the `content_transformer()` function to build a wrapper to define the contents of the dataset. The `tm_map()` function should be used to remove other unnecessary features such as, punctuations which can be replaced with space, numbers, special characters, convert the text to lower case, remove stopwords and remove excess white space from the dataset.

4 Feature Generation

We generated and extracted our features by first creating a Document Term Matrix (DTM) file which allocates rows and columns to comments and words respectively. This means each word in the dataset was given a column and then the number of times in which they occurred was recorded.



```
#DOCUMENT TERM MATRIX
HScorpusDTM = TermDocumentMatrix(HScorpus)
m <- as.matrix(HScorpusDTM)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
View(d)

#Wordcloud
#install.packages('wordcloud')
library(wordcloud)
wordcloud(words = d$word, freq = d$freq, min.freq = 20,max.words = 200,random.order=FALSE,
          rot.per=0.05,colors=brewer.pal(8, "Dark2"))

#Frequency of words
barplot(d[1:10,]$freq, las = 2, names.arg = d[1:10,]$word,
        col ="lightblue",width = 1.0, main ="Most frequent words",
        ylab = "Word frequencies")
```

Fig 6: Feature Generation

The DTM file should be converted into a matrix and sorted, it should also be put in a `data.frame()` which is used to display all the occurring words and the number of times they occur in the dataset. You can install the wordcloud package and activate the library for visual representation of the relevant features, also use the `barplot()` function to generate the first ten or more most occurring texts. The largely written words at the centre of the wordcloud are the most occurring features, the smaller the size of a word the fewer times it occurs.

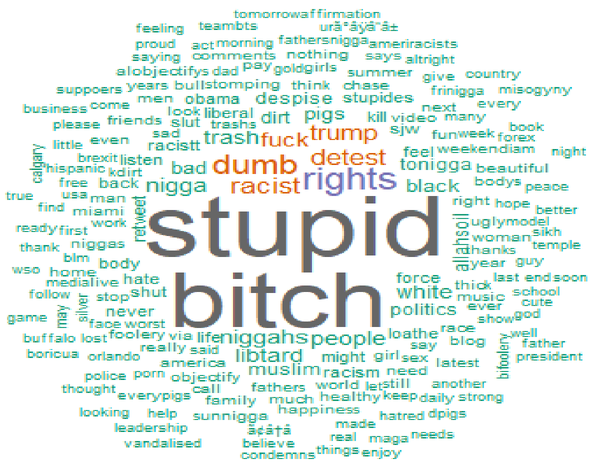
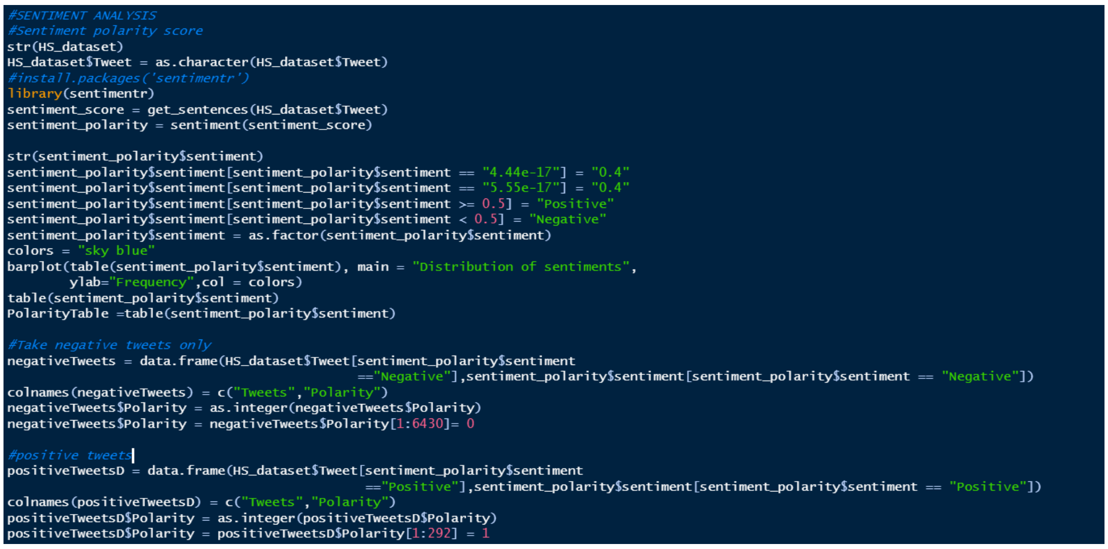


Fig 7: Wordcloud Representation of texts

5 Sentiment Analysis

Sentiment analysis is a section of Natural Language Processing (NLP) that helps check, identify and extract the underlying contexts in a selected text document or a document which contains speech [3]. We used sentiment polarity to check the polarity and assign labels to the dataset. Where tweets were termed negative and positive, the negative stands for tweets with hate then positive tweets represent the absence of hate in tweets. To perform the sentiment analysis, you install the `sentimentr` package, `syuzhet` package and activate their libraries.



```
#SENTIMENT ANALYSIS
#Sentiment polarity score
str(HS_dataset)
HS_dataset$Tweet = as.character(HS_dataset$Tweet)
#install.packages('sentimentr')
library(sentimentr)
sentiment_score = get_sentences(HS_dataset$Tweet)
sentiment_polarity = sentiment(sentiment_score)

str(sentiment_polarity$sentiment)
sentiment_polarity$sentiment[sentiment_polarity$sentiment == "4.44e-17"] = "0.4"
sentiment_polarity$sentiment[sentiment_polarity$sentiment == "5.55e-17"] = "0.4"
sentiment_polarity$sentiment[sentiment_polarity$sentiment >= 0.5] = "Positive"
sentiment_polarity$sentiment[sentiment_polarity$sentiment < 0.5] = "Negative"
sentiment_polarity$sentiment = as.factor(sentiment_polarity$sentiment)
colors = "sky blue"
barplot(table(sentiment_polarity$sentiment), main = "Distribution of sentiments",
        ylab="Frequency",col = colors)
table(sentiment_polarity$sentiment)
PolarityTable =table(sentiment_polarity$sentiment)

#Take negative tweets only
negativeTweets = data.frame(HS_dataset$Tweet[sentiment_polarity$sentiment
                                              =="Negative"],sentiment_polarity$sentiment[sentiment_polarity$sentiment == "Negative"])
colnames(negativeTweets) = c("Tweets","Polarity")
negativeTweets$Polarity = as.integer(negativeTweets$Polarity)
negativeTweets$Polarity = negativeTweets$Polarity[1:6430]= 0

#positive tweets
positiveTweetsD = data.frame(HS_dataset$Tweet[sentiment_polarity$sentiment
                                              =="Positive"],sentiment_polarity$sentiment[sentiment_polarity$sentiment == "Positive"])
colnames(positiveTweetsD) = c("Tweets","Polarity")
positiveTweetsD$Polarity = as.integer(positiveTweetsD$Polarity)
positiveTweetsD$Polarity = positiveTweetsD$Polarity[1:292] = 1
```

Fig 8: Sentiment Analysis

The `syuzhet` package should be activated to help the `get_sentences()` function work. The sentiment score can be derived by using the `get_sentences()` function which is used to tokenize the tweets. Then use `sentiment_polarity` to categorize the tweets into negative and positive tweets. The negative tweets are tweets that fall below the 0.5 sentiment score while tweets that are equal to or greater than 0.5 are considered as positive tweets. Furthermore, the polarity of the negative tweets and positive tweets are converted into integer with the `as.integer()` function, whereby the negative tweets were denoted as "0" while positive were denoted as "1". The result of this phase should be used to train the models on the classification of hate and non-hate tweets. Install the `ggplot` and `ggrepel` packages to help plot a sentiment polarity graph showing the class where all tweets in the dataset belong.

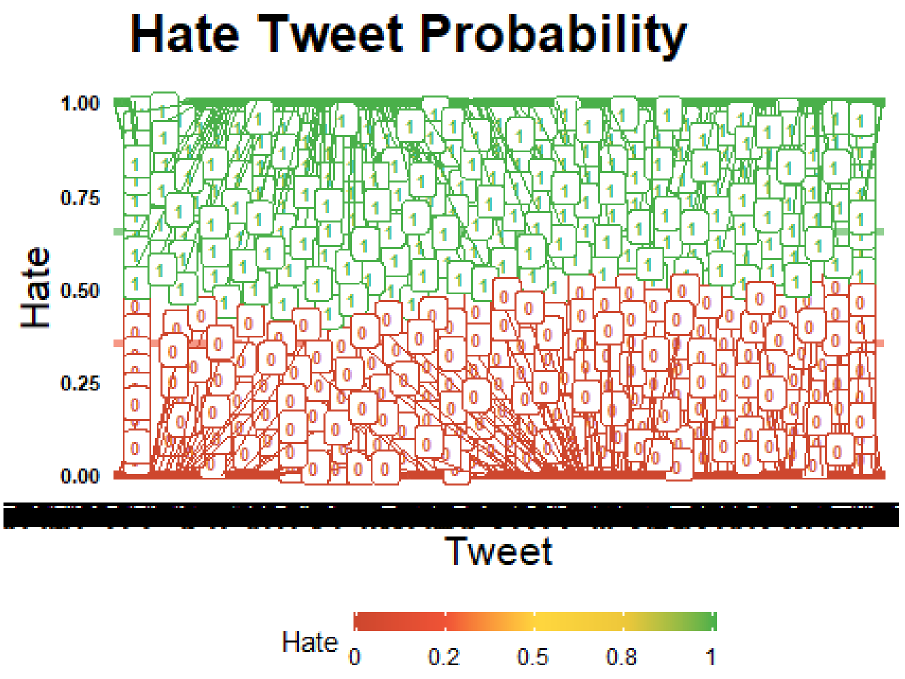
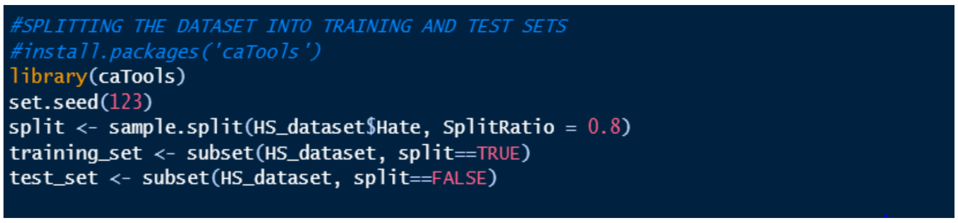


Fig 9: Hate Tweet Probability plot

6 Building of Models

6.1 Data Splitting

Split the dataset in two subsets which are training set and test set, where the training set is used to help the model learn the relevant features needed for the classification. While, the test set is used to confirm if the trained model can make accurate and precise predictions. The training set for our research was given 80% of our dataset while, the test set 20%. This was done in order for the classification model to have a lot of text features to train with, it was also done to help prevent overfitting. To split the dataset, the `caTools` package should be installed and the library activated. Use the `set.seed()` function to set any number of your choice as seed, this function helps to create random numbers or objects which can be replicated.

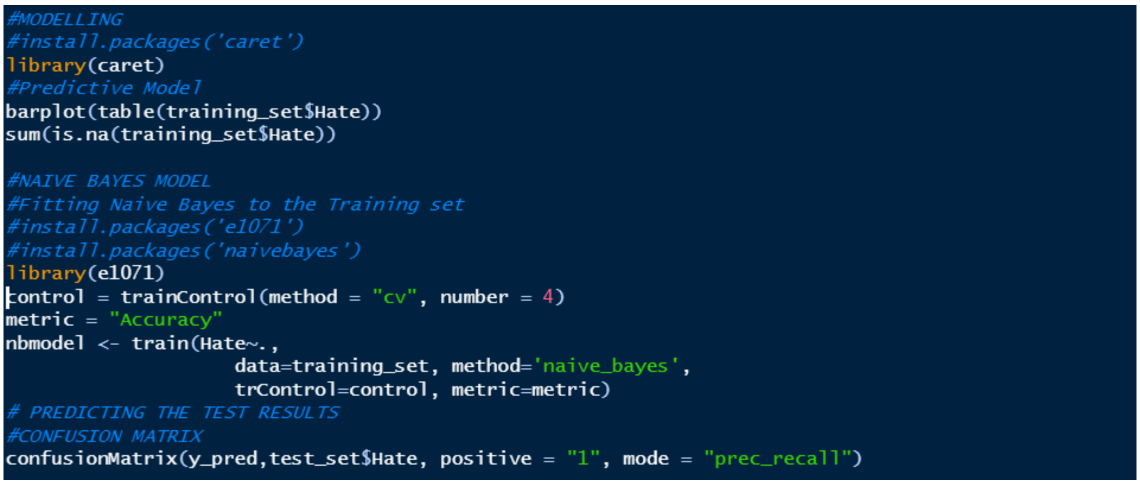


```
#SPLITTING THE DATASET INTO TRAINING AND TEST SETS
#install.packages('caTools')
library(caTools)
set.seed(123)
split <- sample.split(HS_dataset$Hate, SplitRatio = 0.8)
training_set <- subset(HS_dataset, split==TRUE)
test_set <- subset(HS_dataset, split==FALSE)
```

Fig 10: Dataset Splitting

6.2 Model Building

We chose to compare six classification algorithms for this research, they algorithms include; Naïve Bayes, Gaussian Support Vector Machine, K-Nearest Neighbors, Logistic Regression and Decision Tree. For building the models, we installed the `caret` package and activated the library. This package helps in detecting important features needed by algorithms easily. It offers a grid-search method for checking for features and then it has several methods which can be used to estimate the performance of a particular model. The various packages needed to run each model should be installed and their libraries activated. The figure below shows the steps in which our final classification model was performed. The `e1071` package should be installed and the library activated. Use the `trainControl()` function to run this model, as well as set the `cross validation (cv)` to any number of your choice. The model should be first trained on the training set data before it can be used to predict features on the test set data. Use `confusionMatrix()` function to generate the false positives, false negatives, accuracy, precision, recall and F1 scores.



```
#MODELLING
#install.packages('caret')
library(caret)
#Predictive Model
barplot(table(training_set$Hate))
sum(is.na(training_set$Hate))

#NAIVE BAYES MODEL
#Fitting Naive Bayes to the Training set
#install.packages('e1071')
#install.packages('naivebayes')
library(e1071)
control = trainControl(method = "cv", number = 4)
metric = "Accuracy"
nbmodel <- train(Hate~.,
                 data=training_set, method='naive_bayes',
                 trControl=control, metric=metric)
# PREDICTING THE TEST RESULTS
#CONFUSION MATRIX
confusionMatrix(y_pred,test_set$Hate, positive = "1", mode = "prec_recall")
```

Fig: 11: Model Building

7 Results and Evaluation

The metric used for training our models are as follows; accuracy, precision, recall and F1 score. The models seemed to favour precision over accuracy, therefore, we used the precision, recall and F1 scores to compare the models. The graph below shows the metric values gotten from the models when they were used to predict the test set. Also, from the graph below it was observed that our models made a lot mis-classifications and predictions. Where the number of false positives and false negative were massive leading to low accuracy, precision, recall and F1 score.

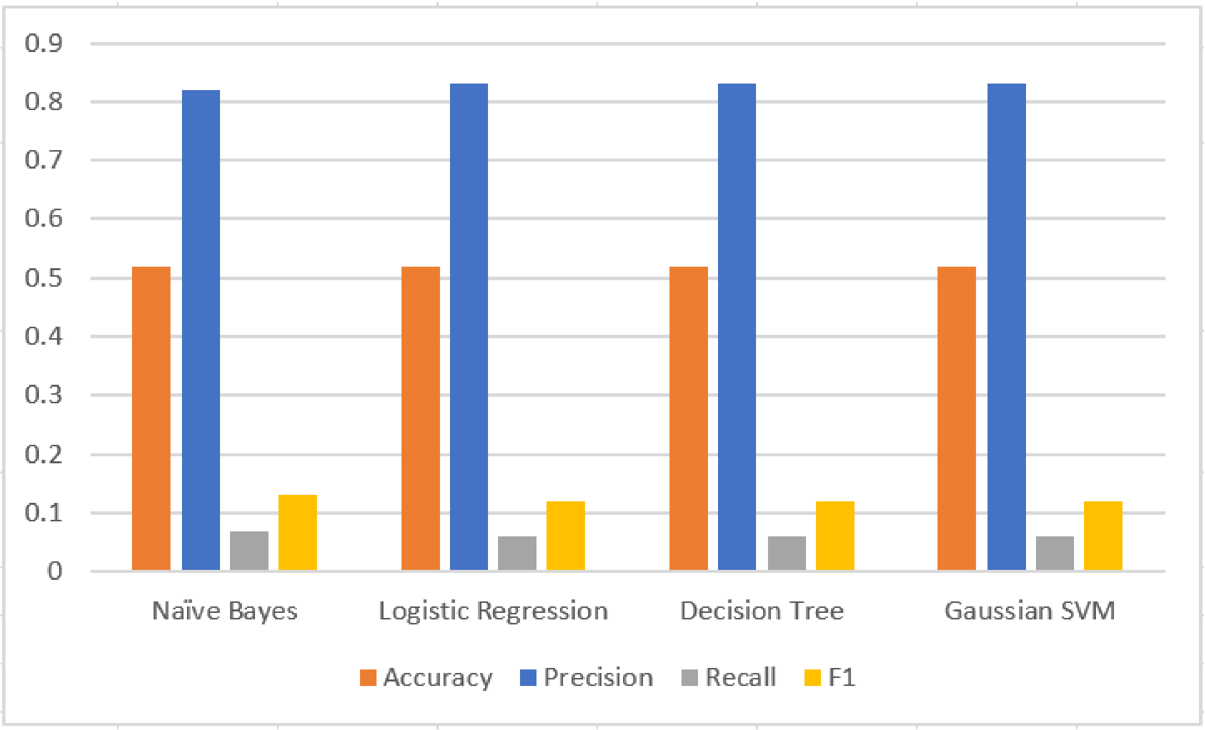


Fig 12: Metrics

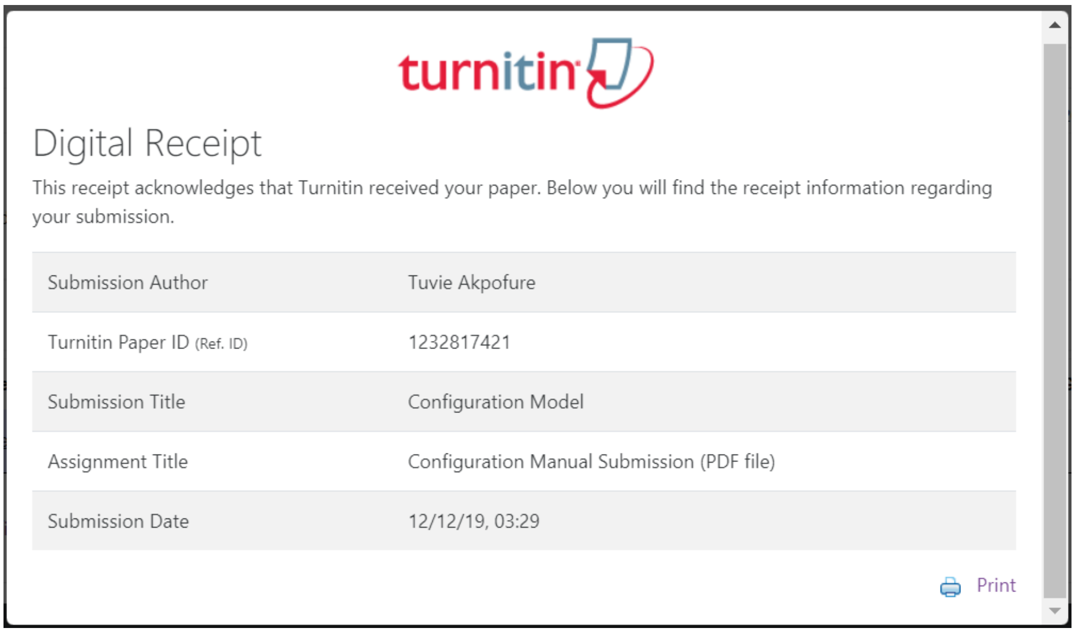
8 References

[1] Tuvie Akpofure, "Automatic Identification of Hate Speech on Social Media platforms using Machine Learning," Dublin, Ireland, 2019.

[2] Kotsiantis S. B., Kanellopoulos D. and Pintelas P. E, "Data Preprocessing for Supervised Leaning," *IJSC,* vol. 1, no. 11306, pp. 111 - 117, 2006.

[3] Sujata R., Parteek K., "A Sentiment Analysis System to Improve Teaching and Learning," *Computer, IEEE,* vol. 50, no. 5, pp. 36 -43, 10 May 2017.

Submission receipt



Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

Submission Author	Tuvie Akpofure
Turnitin Paper ID (Ref. ID)	1232817421
Submission Title	Configuration Model
Assignment Title	Configuration Manual Submission (PDF file)
Submission Date	12/12/19, 03:29