Illinois Institute of Technology

CS-584 Machine Learning

Twitter Hate Speech Recognition

Final Project Report

Submitted By

Tanmay Akhil Devikar (A20517094)
tdevikar@hawk.iit.edu

Sai Reshma Guntimadugu (A20521853)
sguntimadugu@hawk.iit.edu

Aditya Sai Kolluru (A20516766)
akolluru@hawk.iit.edu

1. Introduction

We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompt and also more general (since the number of users who tweet are substantially more than those who write web blogs on a daily basis).

Sentiment analysis of the public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analysing overall public sentiment towards that firm with respect to time and using economic tools for finding the correlation between public sentiment and the firm's stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market it is having a favourable response and in which a negative response (since twitter allows us to download a stream of geo-tagged tweets for particular locations. If firms can get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis.

One such study was conducted by Tumasjan et al. in Germany for predicting the outcome of federal elections which concluded that twitter is a good reflection of offline sentiment .

2. Problem Description

Twitter is a popular social networking website where members create and interact with messages known as "tweets". This serves as a means for individuals to express their thoughts or feelings about different subjects. Various different parties such as consumers and marketers have done sentiment analysis on such tweets to gather insights into products or to conduct market analysis. Furthermore, with the recent advancements in machine learning algorithms, we are able improve the accuracy of our sentiment analysis predictions.

In this report, we will attempt to conduct sentiment analysis on "tweets" using various different machine learning algorithms. We attempt to classify the polarity of the tweet where it is either positive or negative. If the tweet has both positive and negative elements, the more dominant sentiment should be picked as the final label.

We use the data set from **Kaggle** which was crawled and labeled positive/negative. The data provided comes with emoticons, usernames and hashtags which are required to be processed and converted into a standard form. We also need to extract useful features from the text such as uni-grams and bi-grams which is a form of representation of the "tweet".

We use various machine learning algorithms to conduct sentiment analysis using the extracted features. However, just relying on individual models did not give a high accuracy so we picked the top few models to generate a model ensemble. Ensembling is a form of meta learning algorithm technique where we combine different classifiers in order to improve the prediction accuracy. Finally, we report our experimental results and findings at the end.

3. Dataset Description

Dataset used in this project is Hate Speech and Offensive Language Dataset from kaggle.The data given is in the form of a comma-separated values file with tweets and their corresponding sentiments. The training dataset is a csv file of type tweet_id, sentiment, tweet where the tweet_id is a unique integer identifying the tweet, sentiment is either 1 (positive) or 0 (negative), and tweet is the tweet enclosed in "". Similarly, the test dataset is a csv file of type tweet_id, tweet.

The dataset is a mixture of words, emoticons, symbols, URLs and references to people.

	Total	**Unique**	**Average**	**Max**	**Positive**	**Negative**
Tweets	80000	-	-	-	400312	399688
User	393392	-	0.4917	12	-	-
Mentions	-	-	-	-	-	-
Emoticons	6797		0.0085	5	5807	990
URLs	38698	-	0.0484	5	-	-
Unigrams	9823554	181232	12.279	40	-	-
Bigrams	9025707	1954953	11.28	-	-	-

Table 1: Statistics of preprocessed train dataset

	Total	**Unique**	**Average**	**Max**	**Positive**	**Negative**
Tweets	20000	-	-	-	-	-
User	97887	-	0.0.4894	11	-	-
Mentions	-	-	-	-	-	-
Emoticons	1700	-	0.0.0085	10	1472	228
URLs	9553	-	0.0478	5	-	-
Unigrams	2456216	78282	12.286	36	-	-
Bigrams	2257751	686530	11.29	-	-	-

Table 2: Statistics of preprocessed test dataset

Words and emoticons contribute to predicting the sentiment, but URLs and references to people don't. Therefore, URLs and references can be ignored. The words are also a mixture of misspelled words, extra punctuation's, and words with many repeated letters. The tweets, therefore, have to be pre-processed to standardize the dataset.

The provided training and test dataset have 5000 and 1000 tweets respectively. Preliminary statistical analysis of the contents of datasets, after preprocessing is shown in tables 1 and 2.

4. System Modules

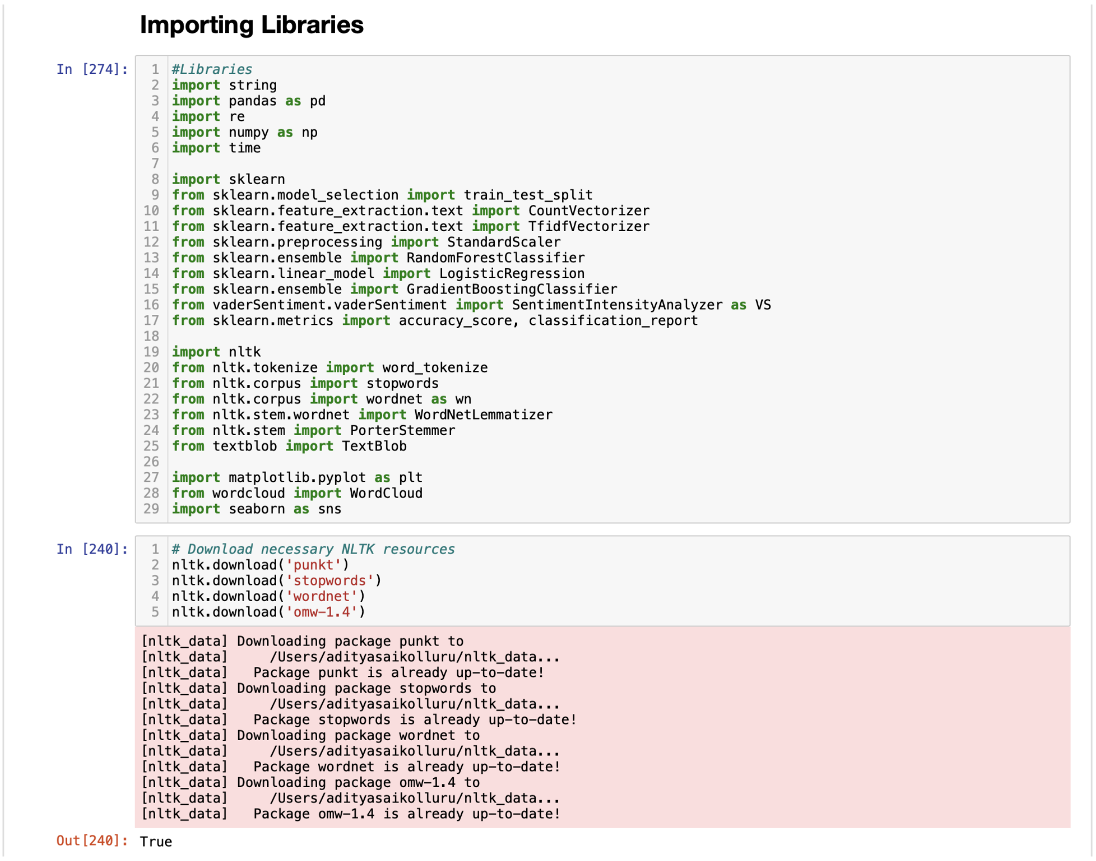
The steps to be followed are :

1. Importing Models
2. Importing Data set
3. Pre-processing the Dataset
4. Model Training
5. Model Evaluation

4.1. Importing the Libraries:

The libraries which we are using :

- Pandas: To load dataset.
- Numpy : For matrix computing
- SkLearn: For train-test split and to import the modules for model evaluation.
- Matplotlib ,Seaborn, WordCloud: For data visualization.
- Re: For regular expressions during data pre-processing stage.
- Nltk: To support classification task.



Importing Libraries

In [274]:
```
#Libraries
import string
import pandas as pd
import re
import numpy as np
import time

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer as VS
from sklearn.metrics import accuracy_score, classification_report

import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem import PorterStemmer
from textblob import TextBlob

import matplotlib.pyplot as plt
from wordcloud import WordCloud
import seaborn as sns
```

In [240]:
```
# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

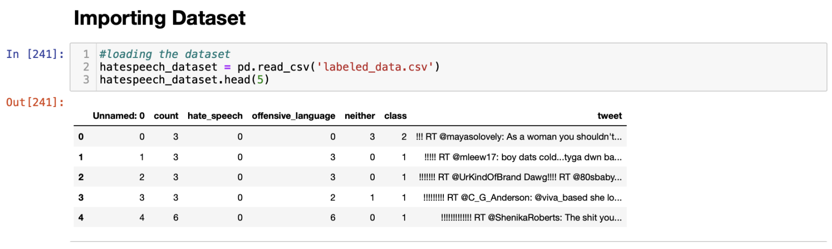
```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/adityasaikolluru/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/adityasaikolluru/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/adityasaikolluru/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     /Users/adityasaikolluru/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[240]:
```
True
```

Fig.1 Importing Libraries

4.2. Importing the Data Set:

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a Data Frame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be accessed using the data frame. Any missing value or NaN value has to be cleaned.



Importing Dataset

In [241]:
```
#loading the dataset
hatespeech_dataset = pd.read_csv('labeled_data.csv')
hatespeech_dataset.head(5)
```

Out[241]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

Fig.2 Loading Data

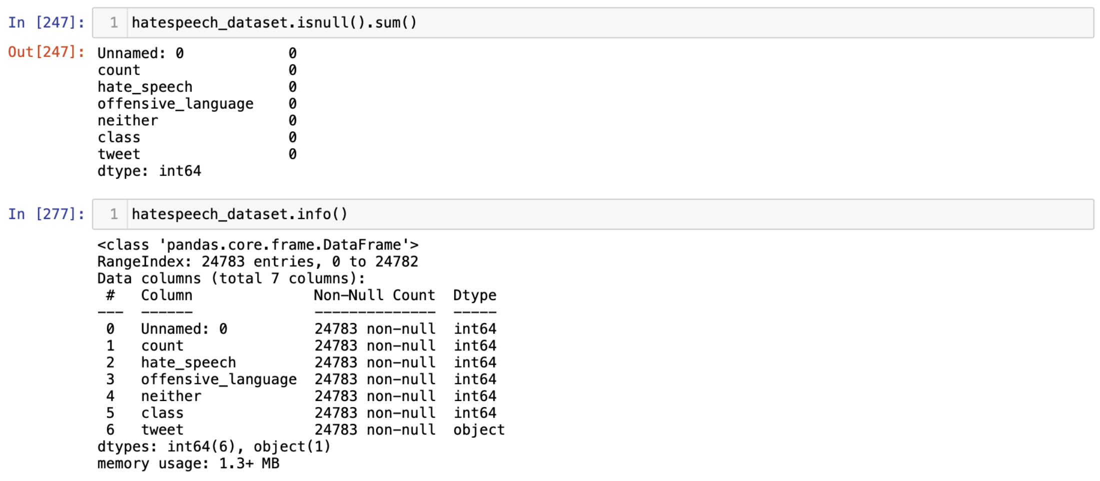
4.3. Data Cleaning and Preprocessing the Dataset

Recommenders are very much garbage-in-garbage-out systems so it’s worth investing time in developing a suitable data collection and processing component. The inner workings of this component are very much use case specific but it’s common to have some data cleansing and normalisation steps plus some feature generation and selection capabilities. As the recommender model has an upstream dependency on this data, the quality of recommendations generated is constrained by the quality of the input data

Data Cleaning is one of the a crucial steps in the Data Pre-processing phase . Data quality issues such as duplication, mislabelling, irrelevant symbols, URLs, and inaccuracies can significantly impact the accuracy of the results produced by any algorithm, even if the outputs may appear to be correct. Therefore, it is important to handle these issues appropriately to ensure that our model produces accurate results and ensure that businesses can make informed decisions based on accurate and reliable analysis of data.

4.3.1. Removing Missing Data

We first check for null values in each column.



```
In [247]: hatespeech_dataset.isnull().sum()
Out[247]: Unnamed: 0            0
          count                 0
          hate_speech           0
          offensive_language    0
          neither               0
          class                 0
          tweet                 0
          dtype: int64

In [277]: hatespeech_dataset.info()
          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 24783 entries, 0 to 24782
          Data columns (total 7 columns):
           #   Column              Non-Null Count  Dtype
          ---  ------              --------------  -----
           0   Unnamed: 0          24783 non-null  int64
           1   count               24783 non-null  int64
           2   hate_speech         24783 non-null  int64
           3   offensive_language  24783 non-null  int64
           4   neither             24783 non-null  int64
           5   class               24783 non-null  int64
           6   tweet               24783 non-null  object
          dtypes: int64(6), object(1)
          memory usage: 1.3+ MB
```

Fig.3.1. Removing Missing Data

After analysing the dataset, we can conclude that there are no null values and the data is complete. Therefore, there is no need to handle missing values.

4.3.2. Text Cleaning

Text cleaning is the process of preparing raw text for machines to understand human language.Clean text refers to human language that has been rearranged into a machine-readable format. Text cleaning typically involves using Python code to eliminate stopwords, remove non-ASCII characters, and reduce complex words to their root form using techniques like lemmatization and stemming.

We need to perform the three most used text cleaning techniques on this query:

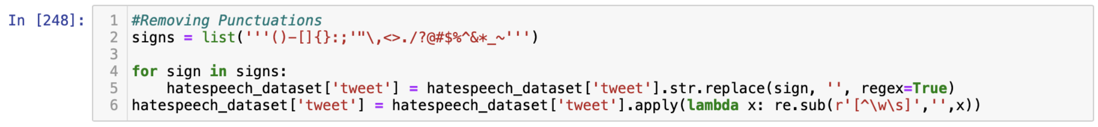
1. Normalizing Text

In order to make the text analysis process easier for machines, we need to perform text normalization techniques:

- that convert words to their basic or root form (Lemmatization)
- removing special characters, HTML tags
- removing punctuation’s
- standardize the letter case to lower case
- @mentions, URLs, and emojis are also converted into Unicode

This step ensures that the text is uniform and standardized, and we can reduce the complexity of the text while retaining the important information.

For example, we can convert ”running”, ”runs”, ”ran” to ”run”, which is the root form of the word. Similar, ”Amazon” and ”AMAZON” to ”amazon” to ensure consistency. Additionally, we can remove special characters like ”@” and punctuation’s like ”.”, ”,”, ”?” which may not provide much value for text analysis.



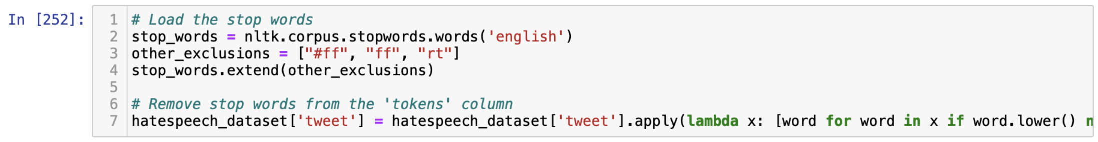
```
In [248]:
#Removing Punctuations
signs = list('''()-[]{}:;'"\,<>./?@#$%^&*_~''')

for sign in signs:
    hatespeech_dataset['tweet'] = hatespeech_dataset['tweet'].str.replace(sign, '', regex=True)
hatespeech_dataset['tweet'] = hatespeech_dataset['tweet'].apply(lambda x: re.sub(r'[^\w\s]','',x))
```

Fig.3.2. Removing Punctuation’s

2. Removing StopWords

After removing punctuation’s and converting the text to lowercase, we still may have some words that do not add any meaning to the text and can be safely removed without affecting the context of the text. These words are known as ”stopwords”. To eliminate these words, we can use pre-existing lists of stopwords for various languages, including English. By removing these words, we can simplify the text and improve the efficiency of our analysis.



```
In [252]:
# Load the stop words
stop_words = nltk.corpus.stopwords.words('english')
other_exclusions = ["#ff", "ff", "rt"]
stop_words.extend(other_exclusions)

# Remove stop words from the 'tokens' column
hatespeech_dataset['tweet'] = hatespeech_dataset['tweet'].apply(lambda x: [word for word in x if word.lower() n
```

Fig.3.3. Removing StopWords

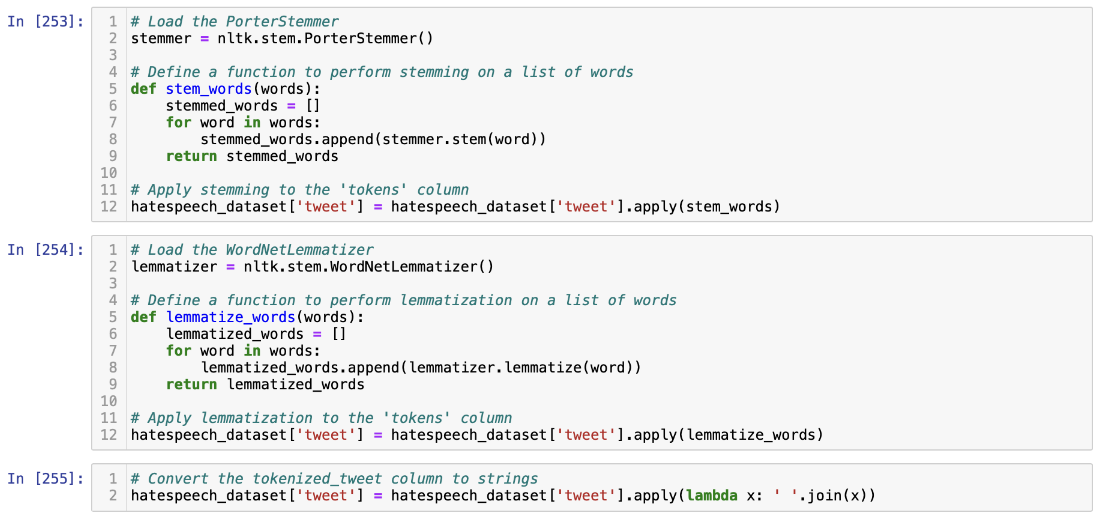
3. Stemming and Lemmitization

Stemming and lemmatization are two text preprocessing techniques used in natural language processing to reduce words to their basic forms.

Stemming involves removing suffixes from words in order to extract their root form. For example, the stem of the word ”running” is ”run”, and the stem of the word ”cats” is ”cat”. Stemming is useful for reducing the number of unique words in a corpus and for improving the efficiency of certain natural language processing tasks, such as information retrieval and search engines.

Lemmatization, on the other hand, involves reducing words to their canonical form, which is the form found in a dictionary or vocabulary. This typically involves removing suffixes and determining the base form of a word based on its part of speech. For example, the lemma of the word ”am” is ”be”, and the lemma of the word ”ran” is ”run”. Lemmatization is more complex than stemming but can provide more accurate results for certain natural language processing tasks, such as sentiment analysis and machine translation.

Both stemming and lemmatization can be useful for improving the efficiency and accuracy of natural language processing tasks.



```
In [253]:
# Load the PorterStemmer
stemmer = nltk.stem.PorterStemmer()

# Define a function to perform stemming on a list of words
def stem_words(words):
    stemmed_words = []
    for word in words:
        stemmed_words.append(stemmer.stem(word))
    return stemmed_words

# Apply stemming to the 'tokens' column
hatespeech_dataset['tweet'] = hatespeech_dataset['tweet'].apply(stem_words)
```

```
In [254]:
# Load the WordNetLemmatizer
lemmatizer = nltk.stem.WordNetLemmatizer()

# Define a function to perform lemmatization on a list of words
def lemmatize_words(words):
    lemmatized_words = []
    for word in words:
        lemmatized_words.append(lemmatizer.lemmatize(word))
    return lemmatized_words

# Apply lemmatization to the 'tokens' column
hatespeech_dataset['tweet'] = hatespeech_dataset['tweet'].apply(lemmatize_words)
```

```
In [255]:
# Convert the tokenized_tweet column to strings
hatespeech_dataset['tweet'] = hatespeech_dataset['tweet'].apply(lambda x: ' '.join(x))
```

Fig.3.4. Stemming and Lemmatization

4.4. Data Visualizations

Data visualization is the graphical representation of information and data. Visualizations are a powerful tool for communicating complex ideas, patterns, and trends in a simple and intuitive way. By using visualizations, we can explore data, identify patterns and outliers, and communicate our findings to others. We did few visualizations using Word Cloud.

Visualizations are an essential tool in data analysis and can be used to uncover insights that would otherwise be difficult to see. They can also be used to communicate findings to a wider audience, making data more accessible and engaging. With the increasing amount of data being generated every day, the ability to create effective visualizations is becoming increasingly important.

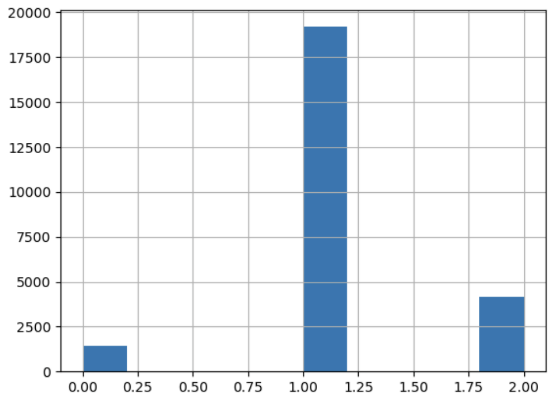


Fig.4.1. Dataset visualization

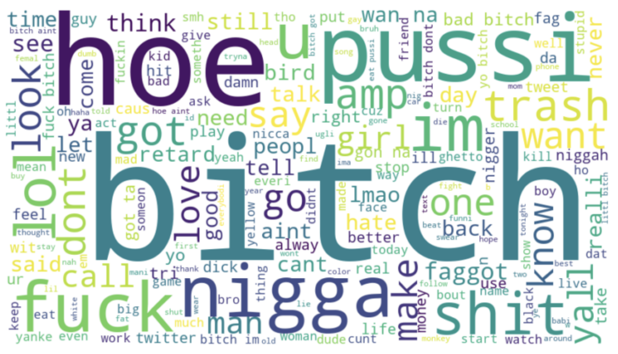


Fig.4.2. WordCloud visualization

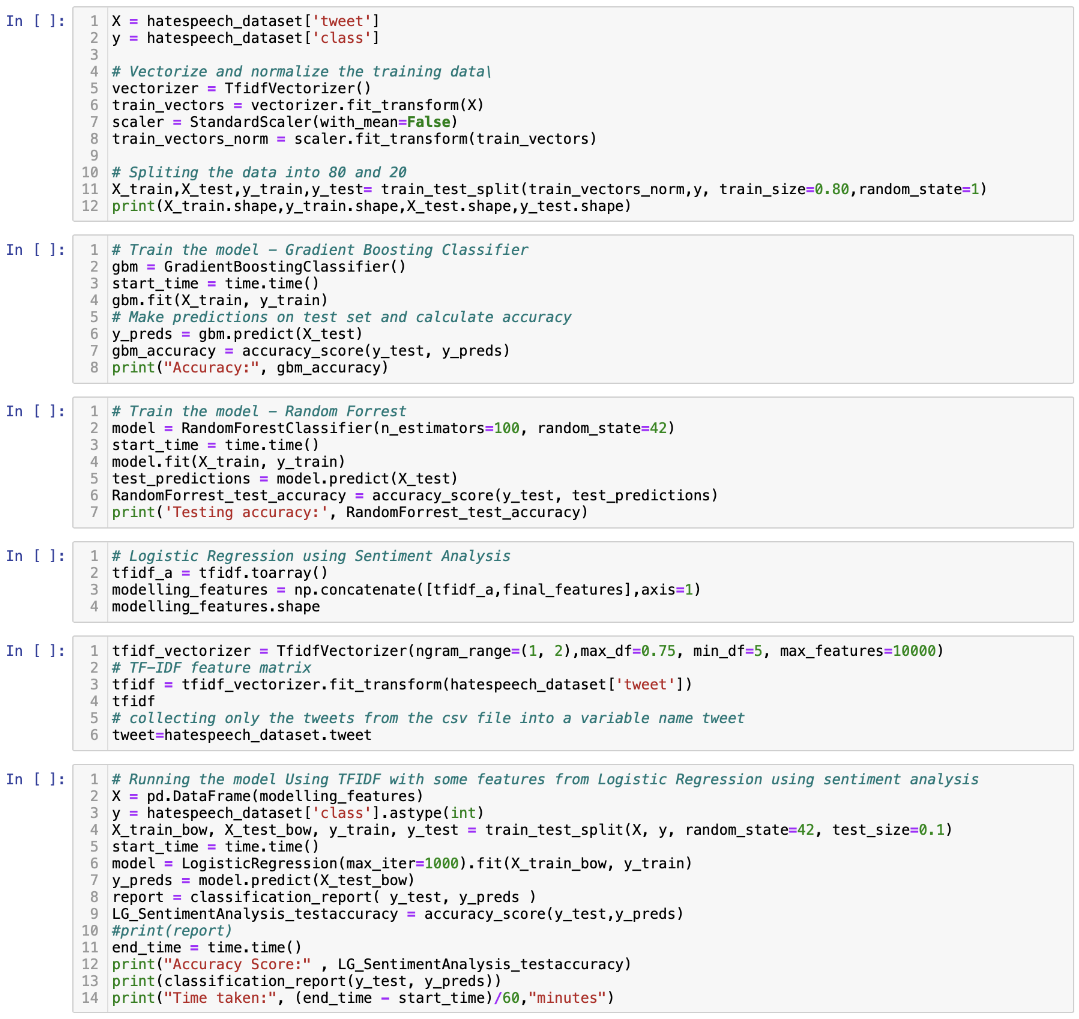
Histogram is used to visualize the distribution of a dataset.

WordCloud is a type of visualization that represents the frequency of words in a text corpus using a cloud-like layout. The size of each word in the cloud corresponds to its frequency in the text, with more frequently occurring words displayed in larger font sizes. Word clouds are often used to get a quick understanding of the most important or relevant words in a text corpus.

4.5. Model Training

After loading the training data, we split it into training and validation sets. We split the training dataset into 80% training and 20% validation sets. We use ‘tweet’ and ‘class’ columns as the input variables and ‘label’ column as the output variable while training the data.

Vectorization is an essential step in natural language processing that involves converting textual data into numerical vectors. Once the text has been pre-processed and cleaned, vectorization can significantly enhance the efficiency and accuracy of machine learning models. There are various techniques available for vectorization, each with its own advantages and drawbacks. We use the Tfidf Vectorizer from scikit-learn to convert the text data into a matrix of numerical features.



```
In [ ]:
X = hatespeech_dataset['tweet']
y = hatespeech_dataset['class']

# Vectorize and normalize the training data\
vectorizer = TfidfVectorizer()
train_vectors = vectorizer.fit_transform(X)
scaler = StandardScaler(with_mean=False)
train_vectors_norm = scaler.fit_transform(train_vectors)

# Spliting the data into 80 and 20
X_train,X_test,y_train,y_test= train_test_split(train_vectors_norm,y, train_size=0.80,random_state=1)
print(X_train.shape,y_train.shape,X_test.shape,y_test.shape)
```

```
In [ ]:
# Train the model - Gradient Boosting Classifier
gbm = GradientBoostingClassifier()
start_time = time.time()
gbm.fit(X_train, y_train)
# Make predictions on test set and calculate accuracy
y_preds = gbm.predict(X_test)
gbm_accuracy = accuracy_score(y_test, y_preds)
print("Accuracy:", gbm_accuracy)
```

```
In [ ]:
# Train the model - Random Forrest
model = RandomForestClassifier(n_estimators=100, random_state=42)
start_time = time.time()
model.fit(X_train, y_train)
test_predictions = model.predict(X_test)
RandomForrest_test_accuracy = accuracy_score(y_test, test_predictions)
print('Testing accuracy:', RandomForrest_test_accuracy)
```

```
In [ ]:
# Logistic Regression using Sentiment Analysis
tfidf_a = tfidf.toarray()
modelling_features = np.concatenate([tfidf_a,final_features],axis=1)
modelling_features.shape
```

```
In [ ]:
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),max_df=0.75, min_df=5, max_features=10000)
# TF-IDF feature matrix
tfidf = tfidf_vectorizer.fit_transform(hatespeech_dataset['tweet'])
tfidf
# collecting only the tweets from the csv file into a variable name tweet
tweet=hatespeech_dataset.tweet
```

```
In [ ]:
# Running the model Using TFIDF with some features from Logistic Regression using sentiment analysis
X = pd.DataFrame(modelling_features)
y = hatespeech_dataset['class'].astype(int)
X_train_bow, X_test_bow, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)
start_time = time.time()
model = LogisticRegression(max_iter=1000).fit(X_train_bow, y_train)
y_preds = model.predict(X_test_bow)
report = classification_report( y_test, y_preds )
LG_SentimentAnalysis_testaccuracy = accuracy_score(y_test,y_preds)
#print(report)
end_time = time.time()
print("Accuracy Score:" , LG_SentimentAnalysis_testaccuracy)
print(classification_report(y_test, y_preds))
print("Time taken:", (end_time - start_time)/60,"minutes")
```

Fig.5.1. TFIDF Vectorizer and Models

After vectorization we use machine learning algorithms to train the model. The algorithms that we used are:

1. Gradient Boosting Classifier
2. Random Forests
3. Logistic Regression with Sentiment Analysis

These three algorithms are commonly used for text classification tasks. Each algorithm has its own strengths and weaknesses, and the best algorithm to use depends on the specific task, performance of the model and the characteristics of the data.

- **Gradient Boosting Classifier** is a type of ensemble machine learning algorithm that is used for both classification and regression tasks.In this, each decision tree is trained on the errors made by the previous tree in the series. The algorithm gradually reduces the errors in the training data by optimizing the loss function, which measures the difference between the predicted and actual values. It is known for its ability to handle complex and non-linear relationships between the features and the target variable.
- **Random Forests** are also well-suited for text classification tasks. They can handle both categorical and continuous data, can easily handle missing data, and can handle non-linear relationships between features. Decision trees are also interpretable, which can be useful for understanding the decision-making process of the model. However, decision trees can be sensitive to overfitting, which is when the model learns the training data too well and does not generalize well to new data. But whereas, random forests can be used to reduce the risk of overfitting and improve the generalization performance of the model.
- **Logistic Regression** is a statistical method used to analyze and predict the likelihood of a binary outcome (eg: positive or negative sentiment) based on one or more input variables (eg: text features). Sentiment analysis is a type of natural language processing that involves identifying and categorizing opinions and emotions expressed in text. By applying logistic regression to sentiment analysis, we can build a model that predicts the probability of a given piece of text having a hate, offensive or neutral sentiment based on certain text features.

4.6 Model Evaluation

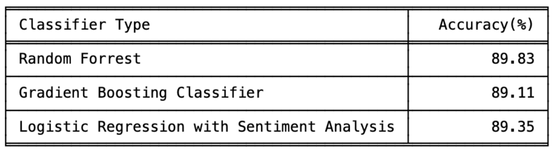
Model evaluation is the process of assessing how well a machine learning model performs on new data. It involves measuring various metrics that indicate how well the model is able to predict the target variable. The goal of model evaluation is to determine the generalization performance of the model and identify best algorithm/model for predicting test data.

There are several metrics that can be used to evaluate the performance of a machine learning model, depending on the type of problem and the nature of the data. Here are some common metrics for classification problems:

- **Accuracy** is the proportion of correct predictions out of the total number of predictions. It is a simple and intuitive metric, but it can be misleading if the data is imbalanced.
- **Precision** is the proportion of true positives (correctly predicted positive instances) out of the total number of predicted positive instances. It is a measure of how accurate the model is at predicting positive instances.
- **Recall** is the proportion of true positives out of the total number of actual positive instances. It is a measure of how complete the model is at predicting positive instances.
- **F1 Score** is the harmonic mean of precision and recall, which gives equal weight to both metrics. It is a more comprehensive metric than accuracy, as it takes into account both the accuracy and completeness of the model.

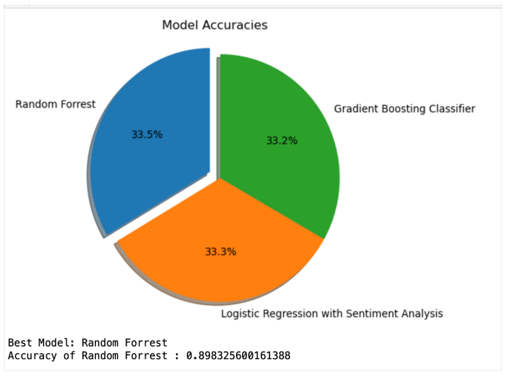
It is also important to note that no single metric can provide a complete picture of the performance of a machine learning model. It is often helpful to look at multiple metrics to get a better understanding of the model's performance.

Accuracies for selected classifier models:



Classifier Type	Accuracy(%)
Random Forrest	89.83
Gradient Boosting Classifier	89.11
Logistic Regression with Sentiment Analysis	89.35

Fig.6.1. Tabulate Representation



```
Best Model: Random Forrest
Accuracy of Random Forrest : 0.898325600161388
```

Fig.6.2. Pie chart - Accuracies Representation

In our case the best accuracy is calculated for Random Forest Classifier.

A confusion matrix is a table used to evaluate the performance of a machine learning algorithm by comparing the predicted labels to the actual labels of a set of data. It is also known as an error matrix or a contingency table.

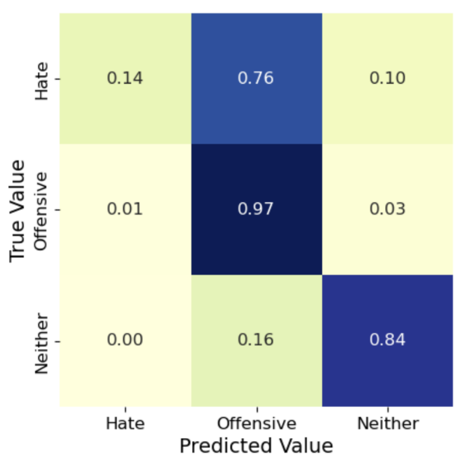
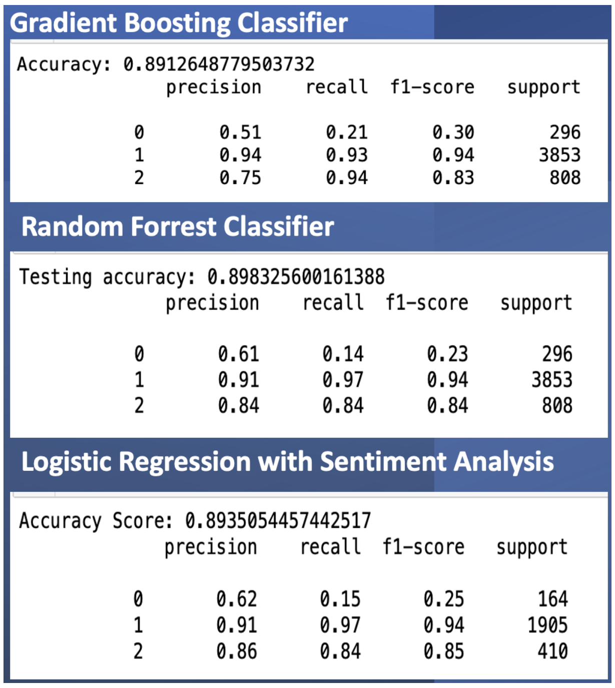


Fig.6.3. Confusion Matrix of Random Forrest

It is crucial to select appropriate metrics for evaluating the performance of a machine learning model based on the specific problem and dataset. In situations where the data is imbalanced, accuracy may not be the only metric that can accurately assess the performance of a model. This is because a model can achieve high accuracy by simply predicting the majority class, which may not be useful in identifying patterns in the minority class.

Therefore, in such cases, it is important to use additional metrics such as precision, recall, and F1-score in conjunction with accuracy to select the best model. These metrics provide a more comprehensive understanding of the model's performance by taking into account the model's ability to correctly identify both the majority and minority classes. By using multiple metrics, we can better evaluate the model's performance and ensure that we select the best model for our specific problem and dataset.



Gradient Boosting Classifier

```
Accuracy: 0.8912648779503732
              precision    recall  f1-score   support

           0       0.51      0.21      0.30       296
           1       0.94      0.93      0.94      3853
           2       0.75      0.94      0.83       808
```

Random Forrest Classifier

```
Testing accuracy: 0.898325600161388
              precision    recall  f1-score   support

           0       0.61      0.14      0.23       296
           1       0.91      0.97      0.94      3853
           2       0.84      0.84      0.84       808
```

Logistic Regression with Sentiment Analysis

```
Accuracy Score: 0.8935054457442517
              precision    recall  f1-score   support

           0       0.62      0.15      0.25       164
           1       0.91      0.97      0.94      1905
           2       0.86      0.84      0.85       410
```

Fig.6.4. Metrics

Based on the above metric analysis, we can observe that Random Forest has higher precision and recall values compared to the other models. This means that Random Forest has a better balance between correctly identifying positive cases (precision) and capturing all positive cases (recall) compared to other models.

Therefore, it can be concluded that Random Forest is a better performing model for the given problem, taking into account both precision and recall metrics.

5. Conclusion:

This project is addressing the issue of Hate and offensive speech on Social Media platforms like Twitter. By developing a classification model that can accurately identify and classify Hate, offensive and neutral speech tweets , we can minimize the negative impact of these hate speech on social media. After testing multiple models over the training data, and analysis on the metrics on them we finalized Random Forest as the best model with **89.83%** accuracy.

The results obtained from this project can serve as a foundation for future work in the field of Hate speech recognition, ultimately leading to a safer and more inclusive environment where everyone feels valued and respected.

6. References:

[1] Albert Biffet and Eibe Frank. Sentiment Knowledge Discovery in Twitter Streaming Data. Discovery Science, Lecture Notes in Computer Science, 2010, Volume 6332/2010, 1-15, DOI: 10.1007/978-3-642-16184-1_1

[2] Alec Go, Richa Bhayani and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Project Technical Report, Stanford University, 2009.

[3] Ricgard O. Duda, Peter E. Hart David G. Stork: Pattern Classification.

[4] https://monkeylearn.com/blog/text-cleaning/

[5] https://heartbeat.comet.ml/vectorization-in-machine-learning-2e3bdce7dbe

[6] https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error- metrics/

[7] https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm- f10ba6e38234

[8] NumPy. (2021). NumPy: A fundamental package for scientific computing with Python. Retrieved from https://numpy.org/

[9] McKinney, W., others. (2011). pandas: a foundational Python library for data analysis and statistics. Python for High Performance and Scientific Computing, 14, 1-9. Retrieved from https://pandas.pydata.org/docs/

[10] Scikit-learn. (2021). Scikit-learn: Machine learning in Python. Retrieved from https://scikit-learn.org/stable/

[11] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Retrieved from https://matplotlib.org/

[12] Waskom, M. (2021). seaborn: statistical data visualization. Journal of Open Source Software,6(60), 3021. Retrieved from https://joss.theoj.org/papers/10.21105/joss.03021

[13] Van Rossum, G., Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace. Retrieved from https://www.python.org/doc/3/libra

[14] Bird, S., Loper, E., Klein, E. (2009). Natural language processing with Python. O'Reilly Media,Inc. Retrieved from https://www.nltk.org/book/

[15] https://github.com/AdityakolluruHawk/CS584Group51TwitterHateSpeechRecognition