Using Machine Learning to identify hate speech and offensive language on Twitter.

Mayara Lorens
CCT College Dublin

Thayene Lorens
CCT College Dublin

Module Title:	Problem Solving for Industry
Assessment Title:	Capstone Pair Project
Lecturer Name:	Dr. Muhammad Iqbal
Student Full Name:	Mayara Lorens; Thayene Lorens
Student Number:	2020292, 2020293
Assessment Due Date:	17/05/2024
Date of Submission:	22/05/2024

Declaration

Summary

Index

Group Members

Members Info:

- Mayara Lorens (2020292)
- Thayene Lorens (2020293)

(Twins Group members)

GitHub Link: https://github.com/Thayene-CCT/Twins_capstoneProject.git

Wordcount: Around 2.746 words.

Final number of GitHub commits: 71 commits.

Tools

- Python
- Jupyter Notebook
- Anaconda

For our project, we've opted to use Python as our programming language for coding within the Jupyter Notebook web platform. We made this choice due to our familiarity with Python, particularly in comparison to the R programming language and because Python is object-oriented, cross platform compatible with a high-level syntax, making it an intuitive and accessible language.

Abstract

The central theme of this project is the application of Machine Learning to identify both hate speech and offensive language on Twitter. We chose this topic for its ethical relevance in the technological environment and its business potential. This topic raises concerns such as cyberbullying and the existence of a hostile environment for users. For this reason, we sought to implement four different models to create an automated system capable of identifying and categorizing whether specific content is offensive, non-offensive or neutral.

Key words: Hate Speech, Offensive Language, Cyberbullying, Online Content, Machine Learning, Social Media, Python.

Warning: This report may contain racist, sexist, homophobic, and offensive content. They have been retrieved solely for studying purposes from online resources and do not reflect the authors' opinion.

Introduction

The introduction of social media like Twitter, Facebook, and Instagram has significantly made way for people to express their opinions, ideas, feelings, and thoughts. It is crucial to differentiate the constructive use of these platforms from their misuse for spreading hateful and offensive speech.

Before exploring the subject in depth, it is important to first point out the definition of hate speech in our digital age. According to the Council of Europe (as cited in Ramírez-García et al[1]., 2022), the term 'hate speech' encompasses "forms of expression that propagate, incite, promote, or justify rational hatred, xenophobia, anti-Semitism, and all other forms of hatred based on intolerance, including aggressive nationalism, ethnocentrism, discrimination, and hostility towards immigrant minorities".

Similarly, the Cambridge Dictionary defines hate speech as "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation."

Offensive language, on the other hand, is when a person uses pejorative words but without the intent of inciting hate or influencing others. RightsforPeace.org[2] explains that offensive language is discourse that poses no risk to others.

The Citizen's Information of Ireland [3] mentions that hate speech encompasses a range of intolerant behaviours across different online platforms. Our project focuses on written texts from Twitter, due to its extensive user base and prevalence in contemporary discourse.

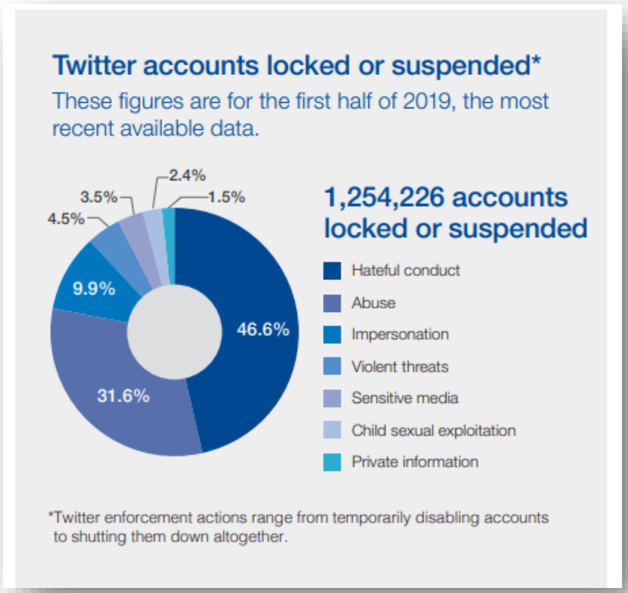
[1] Ramírez-García, A. et al. (2022) 'Interdisciplinarity of Scientific Production on Hate Speech and Social Media: A Bibliometric Analysis', Comunicar: Media Education Research Journal, 30(72), pp. 123–134. Available at: https://research.ebsco.com/linkprocessor/plink?id=529e6fa7-98d8-3129-9bcf-1953609ca289 (Accessed: 15 Mar. 2024).

[2] Rights for Peace. (n.d.). What is Hate Speech? [online] Available at: https://www.rightsforpeace.org/hate-speech#:~:text=Speech%20that%20is%20simply%20offensive [Accessed 23 May 2024].

[3] Citizensinformation.ie (n.d.). The law on hate speech. [online] www.citizensinformation.ie. Available at: https://www.citizensinformation.ie/en/justice/criminal-law/criminal-offences/law-on-hate-speech/#208370 [Accessed 15 Mar. 2024].

According to the NYU Stern report, as cited in 'Who Moderates the Social Media Giants? A call to end outsourcing' by Paul M. Barrett (2020), for the first half of 2019, 46.6% of Twitter accounts were locked or suspended due to hateful conduct as displayed in Figure 1.

Figure 1: 46.6% of Twitter accounts were locked or suspended due to hateful conduct.



Source: *Who Moderates the Social Media Giants? A call to end outsourcing*

Our project aims to develop a system that can analyse and detect online hateful and offensive text-based content using Machine Learning algorithms. By identifying and classifying cyber offensive language on Twitter, we aim to mitigate the harmful effects of hate speech and help companies promote a safer online environment. The large amount of daily data makes manual moderation difficult, highlighting the importance of automated solutions.

Literature Review

Previous studies have explored several methods for detecting hate speech using text mining and machine learning and this project builds on past research to develop a robust system for Twitter.

Sossi Alaoui, Safae & Farhaoui, Yousef & Aksasse, B. (2022)[4] explored hate speech detection using text mining and machine learning techniques, employing a Naive Bayes classifier to achieve an accuracy of 87.23%. Toktarova et al. (2023)[5] used both machine learning and deep learning methods, with SVM achieving an accuracy of 87.3% and NB 87.4%. Ona de Gibert et al.[6] used a dataset sourced from a white supremacy forum, with LSTM classifiers often performing successfully.

Ethical and Legal Aspects

Natural Language Processing (NLP) relies on text as its primary source of information. As data is collected for processing, the text undergoes analysis to determine its emotional tone. Consequently, concerns regarding privacy issues may arise, especially considering that our data is sourced from tweets. The collection process of this type of dataset involves using content posted by real users who probably do not want to be identified, so it is imperative to take responsibility in handling others' personal information, focusing on the importance of data transparency. Moreover, NLP models and sentiment analysis algorithms may sometimes reflect biases present in the data they are trained on, potentially causing discrimination against certain groups.

[4] Sossi Alaoui, Safae & Farhaoui, Yousef & Aksasse, B.. (2022). Hate Speech Detection Using Text Mining and Machine Learning. International Journal of Decision Support System Technology. 14. 1-20. 10.4018/IJDSST.286680. [Accessed 23.Mar 2024].

[5] Toktarova, Aigerim & Syrlybay, Dariga & Myrzakhmetova, Bayan & Anuarbekova, Gulzat & Rakhimbayeva, Gulbarshin & Zhylanbaeva, Balkiya & Suieuova, Nabat & Kerimbekov, Mukhtar. (2023). Hate Speech Detection in Social Networks using Machine Learning and Deep Learning Methods. International Journal of Advanced Computer Science and Applications. 14. 10.14569/IJACSA.2023.0140542. [Accessed 23.Mar 2024].

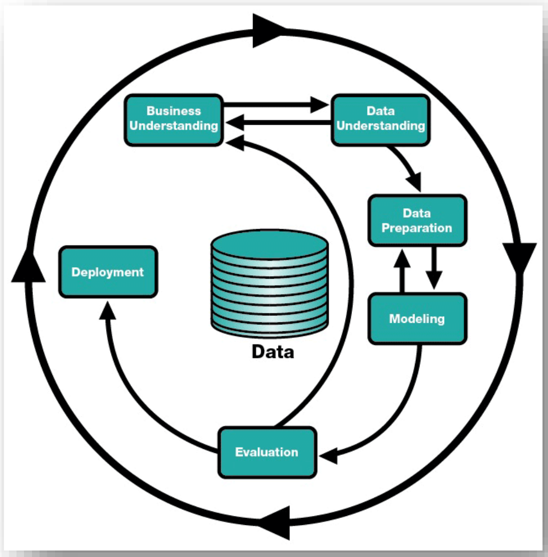
[6] de Gibert Bonet, Ona & Perez, Naiara & García-Pablos, Aitor & Cuadros, Montse. (2018). Hate Speech Dataset from a White Supremacy Forum. [Accessed 27.Mar 2024].

Methodology

This project follows the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, which includes the following steps:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modelling
5. Evaluation
6. Deployment

Figure 2: CRISP-DM Process Flow Diagram.



Source: Google images

1. Business Understanding

Over the past few decades, the digital space has experienced a remarkable expansion, especially when it comes to social medias, where the volume of data generated on a daily basis is massive, therefore, to manually detect hate speech or offensive language online can be a very time-consuming task, mostly for human moderators, who are limited by factors such as fatigue, reading abilities, and personal biases regarding what is considered to be censurable content. Kung (2020)[7] also mentions that human content moderation exploits people by consistently traumatizing and underpaying them based on a 2019 ‘The Verge’ article.

Data Understanding

To build our model, we were required to source suitable data to work with. We adopted a single dataset called ‘**labeled.csv**’ that we renamed as ‘**mean_tweets.csv**’. This dataset consists of American English text-based tweets or comments from Twitter users. This collection encompasses hate speech, offensive language, and non-offensive language. It was downloaded from the website *data.world.com*[8], but it was originally created by Davidson et al.[9] from Cornell University in 2017, who used Twitter API and ended up extracting 85.4 million tweets from thousands of users.

The labelled dataset consists of seven columns: ‘Unnamed: 0', 'count', 'hate_speech', 'offensive_language', ‘neither’ 'class' and 'tweet'. It was annotated automatically by CrowdFlower staff to avoid human annotation bias.

[7] GitHub. (n.d.). twitter_hate_speech_detection/final_notebook.ipynb at master · sidneykung/twitter_hate_speech_detection. [online] Available at: https://github.com/sidneykung/twitter_hate_speech_detection/blob/master/final_notebook.ipynb [Accessed 23. Mar 2024].

[8] data.world. (n.d.). Hate Speech Identification - dataset by crowdflower. [online] Available at: https://data.world/crowdflower/hate-speech-identification.

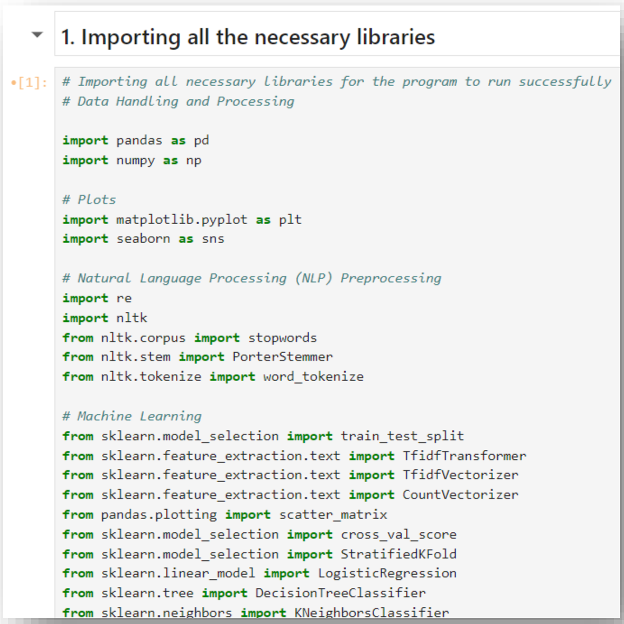
[9] t-davidson (2017). hate-speech-and-offensive-language/src/Automated Hate Speech Detection and the Problem of Offensive Language.ipynb at master · t-davidson/hate-speech-and-offensive-language. [online] GitHub. Available at: https://github.com/t-davidson/hate-speech-and-offensive-language/blob/master/src/Automated%20Hate%20Speech%20Detection%20and%20the%20Problem%20of%20Offensive%20Language.ipynb [Accessed 27. Mar 2024].

Annotations and manual labelling were managed by a diverse team of individuals. To gain insight into the workflow, it's helpful to understand Crowdflower's operational model. They tackle sizable projects by breaking them down into manageable tasks and rely on human input. Each contributor operates as a freelancer, earning income from their collaboration with the company.

In this subsection are the primary steps taken to understand the dataset.

1. Importing all the necessary libraries for our program to run smoothly.

Figure 3: Importing all the necessary libraries.

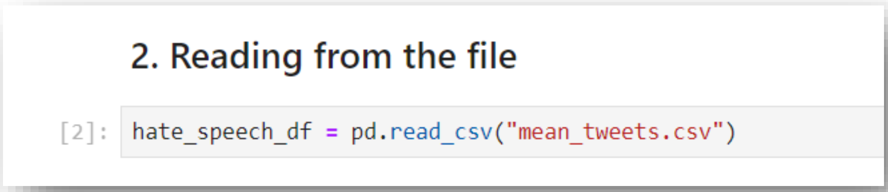


1. Importing all the necessary libraries

```
[1]: # Importing all necessary libraries for the program to run successfully
# Data Handling and Processing

import pandas as pd
import numpy as np

# Plots
import matplotlib.pyplot as plt
import seaborn as sns

# Natural Language Processing (NLP) Preprocessing
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# Machine Learning
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from pandas.plotting import scatter_matrix
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

2. Loading the data.

Figure 4: Data loading.

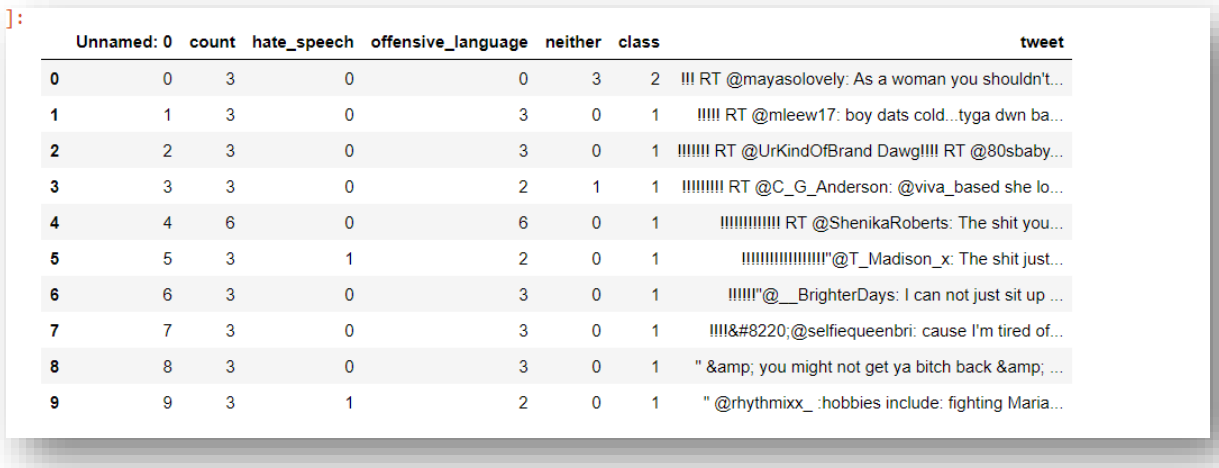


2. Reading from the file

```
[2]: hate_speech_df = pd.read_csv("mean_tweets.csv")
```

3. Displaying the first ten rows of the data.

Figure 5: Displaying the first 10 rows of our data.



]:

	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...
5	5	3	1	2	0	1	!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...
6	6	3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up ...
7	7	3	0	3	0	1	!!!!&#8220;@selfiequeenbri: cause I'm tired of...
8	8	3	0	3	0	1	" &amp; you might not get ya bitch back &amp; ...
9	9	3	1	2	0	1	" @rhythmixx_ :hobbies include: fighting Maria...

- **Unnamed:** Index.
- **Count:** It represents the number of users who coded each tweet.
- **Hate_speech:** How many users believed the tweet to be related to hate speech.
- **Offensive_language:** How many users believed the tweet to be offensive.
- **Neither:** How many users believed the tweet to be neither offensive nor hate speech.
- **Class:**

  > 0: hate speech.

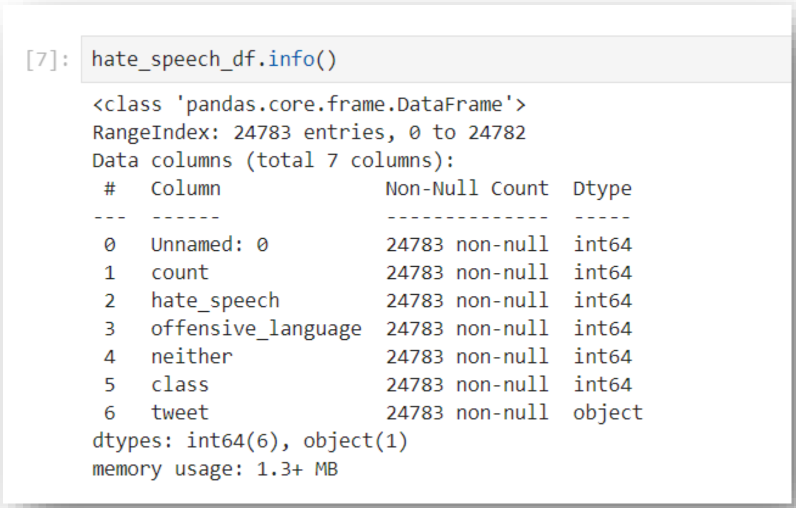
  > 1: offensive language.

  > 2: neither.

- **Tweet:** It represents the tweet posted in Twitter.

4. Displaying the hate_speech_df information to understand the structure of the DataFrame.

Figure 6: Displaying the info of the hate_speech_df dataset.

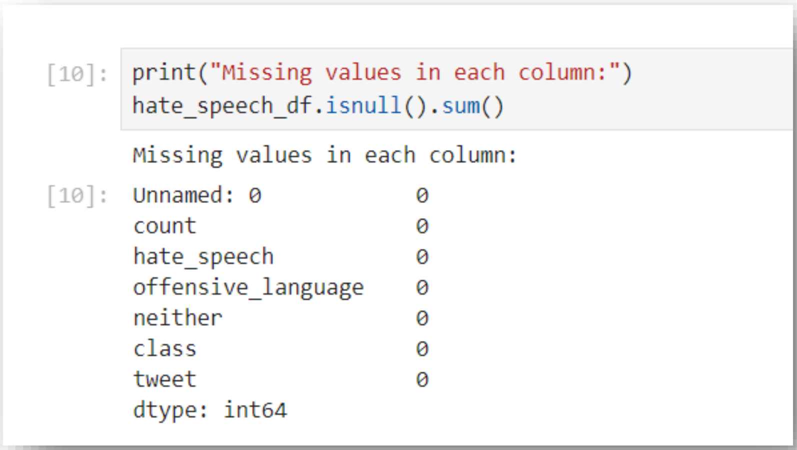


```
[7]: hate_speech_df.info()

     <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
```

- **Number of rows:** 24783.
- **Number of object types:** 1.
- **Number of int types**: 6.

5. Checking for missing values.

Figure 7: Displaying the sum of missing values.

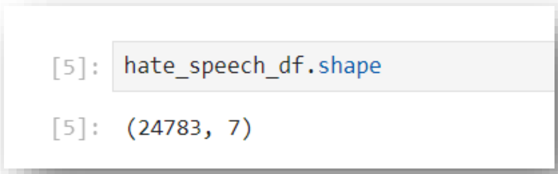


```
[10]: print("Missing values in each column:")
      hate_speech_df.isnull().sum()

      Missing values in each column:
[10]: Unnamed: 0            0
      count                 0
      hate_speech           0
      offensive_language    0
      neither               0
      class                 0
      tweet                 0
      dtype: int64
```

There are no missing values in our dataset.

6. Checking the total number of rows and columns of the dataset using the shape method.

Figure 8: Displaying number of rows and columns of our dataset.

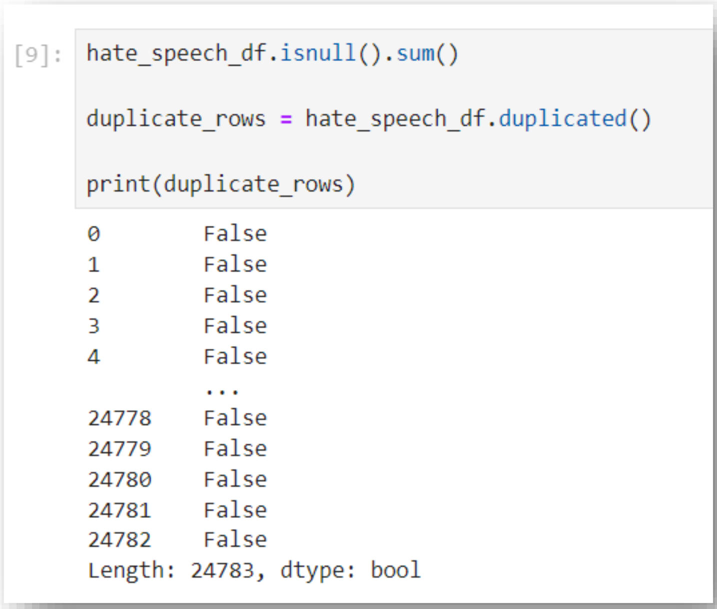


```
[5]: hate_speech_df.shape

[5]: (24783, 7)
```

- **Number of rows:** 24783.
- **Number of columns:** 7.

7. Checking for duplicate entries.

Figure 9: No duplicates.

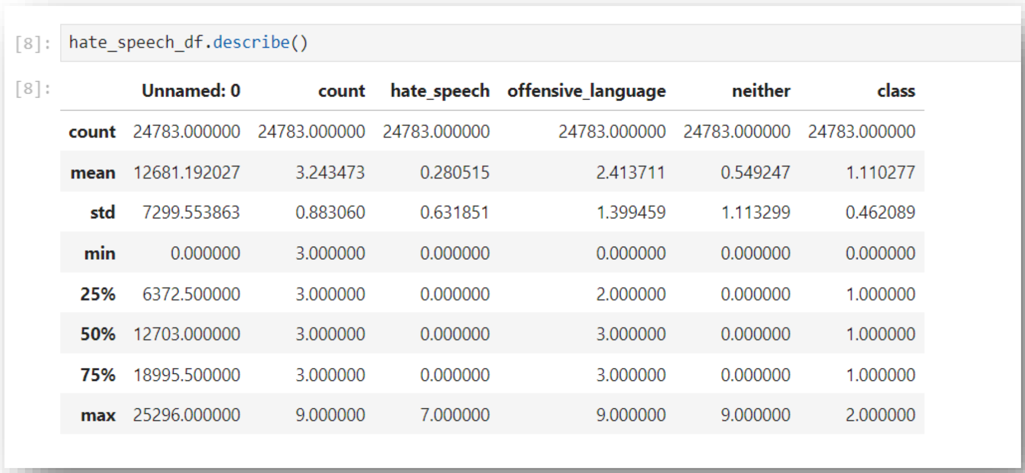


```
[9]: hate_speech_df.isnull().sum()

     duplicate_rows = hate_speech_df.duplicated()

     print(duplicate_rows)

     0        False
     1        False
     2        False
     3        False
     4        False
              ...
     24778    False
     24779    False
     24780    False
     24781    False
     24782    False
     Length: 24783, dtype: bool
```

No duplicated entries in the dataset.

8. Checking the description of the data.

Figure 10: Numerical description of the data.



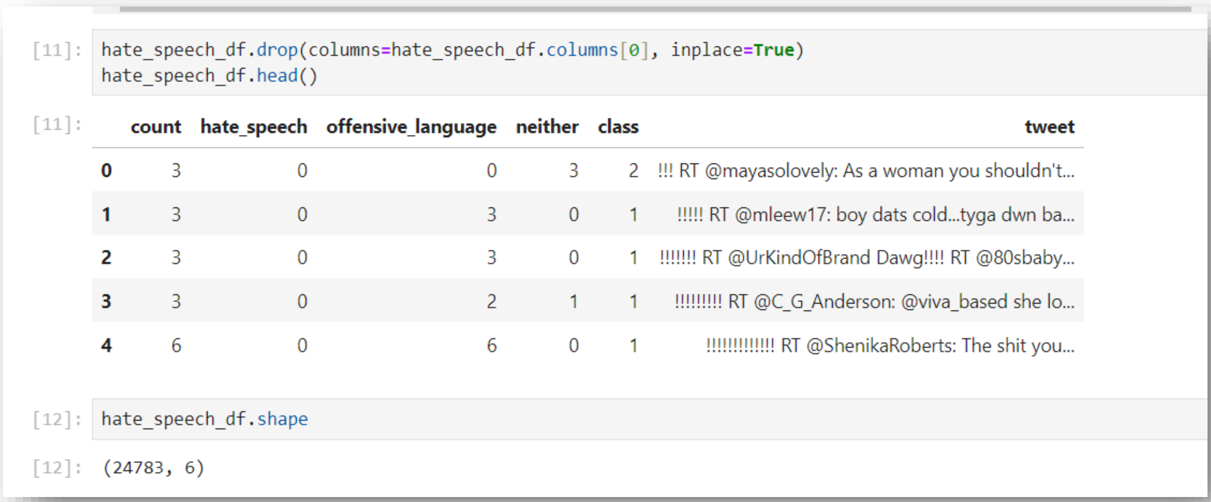
```
[8]: hate_speech_df.describe()
```

[8]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class
count	24783.000000	24783.000000	24783.000000	24783.000000	24783.000000	24783.000000
mean	12681.192027	3.243473	0.280515	2.413711	0.549247	1.110277
std	7299.553863	0.883060	0.631851	1.399459	1.113299	0.462089
min	0.000000	3.000000	0.000000	0.000000	0.000000	0.000000
25%	6372.500000	3.000000	0.000000	2.000000	0.000000	1.000000
50%	12703.000000	3.000000	0.000000	3.000000	0.000000	1.000000
75%	18995.500000	3.000000	0.000000	3.000000	0.000000	1.000000
max	25296.000000	9.000000	7.000000	9.000000	9.000000	2.000000

9. Dropping unnecessary columns.

Figure 11: Dropping the ‘unnamed’ column.



```
[11]: hate_speech_df.drop(columns=hate_speech_df.columns[0], inplace=True)
      hate_speech_df.head()
```

[11]:

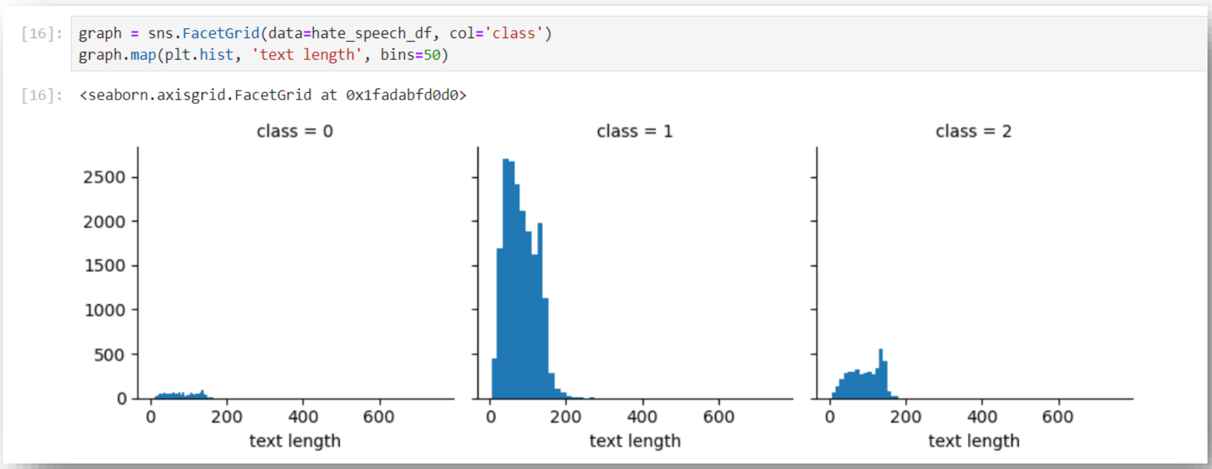
	count	hate_speech	offensive_language	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...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

```
[12]: hate_speech_df.shape

[12]: (24783, 6)
```

We decided to drop the ‘Unnamed’ column because it was unnecessary for this study case.

10. Checking the text length of the tweets according to their labels.

Figure 12: Text length of the tweets.

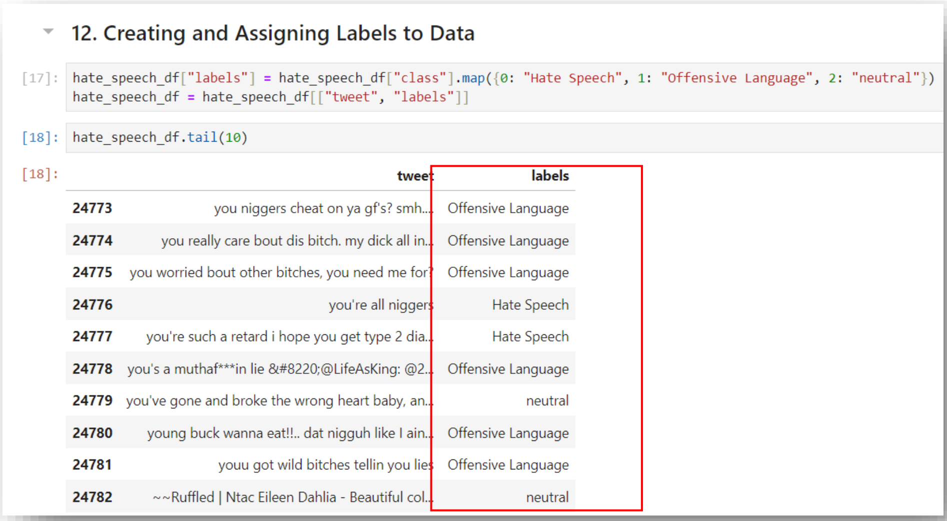


```
[16]: graph = sns.FacetGrid(data=hate_speech_df, col='class')
      graph.map(plt.hist, 'text length', bins=50)

[16]: <seaborn.axisgrid.FacetGrid at 0x1fadabfd0d0>
```

We analysed the text lengths in the 'tweet' column and found that users tend to write longer texts when expressing offensive language.

11. Renaming class labels and plotting their distribution.

Figure 13: New class labels.



12. Creating and Assigning Labels to Data

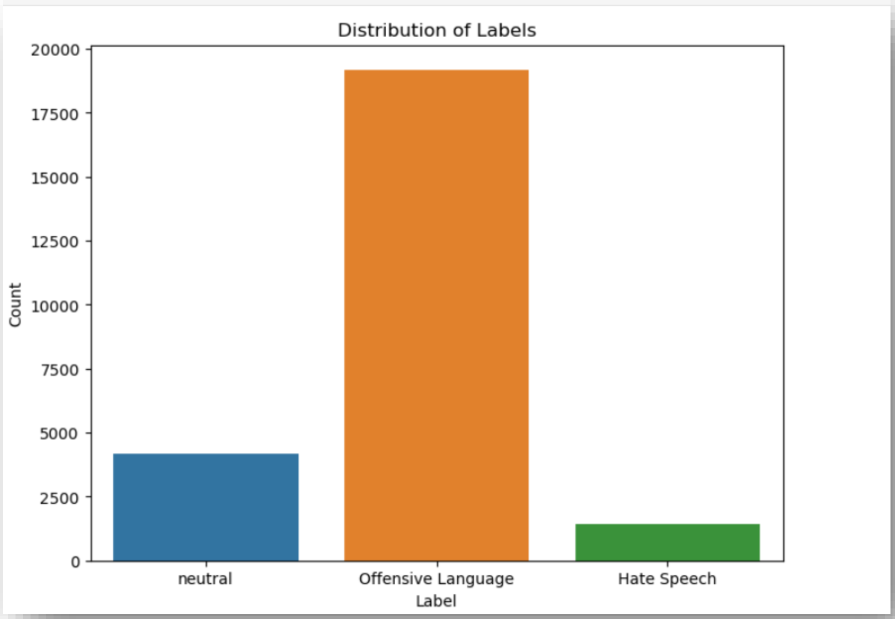
```
[17]: hate_speech_df["labels"] = hate_speech_df["class"].map({0: "Hate Speech", 1: "Offensive Language", 2: "neutral"})
      hate_speech_df = hate_speech_df[["tweet", "labels"]]

[18]: hate_speech_df.tail(10)
```

[18]:

	tweet	labels
24773	you niggers cheat on ya gf's? smh...	Offensive Language
24774	you really care bout dis bitch. my dick all in...	Offensive Language
24775	you worried bout other bitches, you need me for?	Offensive Language
24776	you're all niggers	Hate Speech
24777	you're such a retard i hope you get type 2 dia...	Hate Speech
24778	you's a muthaf***in lie &#8220;@LifeAsKing: @2...	Offensive Language
24779	you've gone and broke the wrong heart baby, an...	neutral
24780	young buck wanna eat!!.. dat nigguh like I ain...	Offensive Language
24781	youu got wild bitches tellin you lies	Offensive Language
24782	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...	neutral

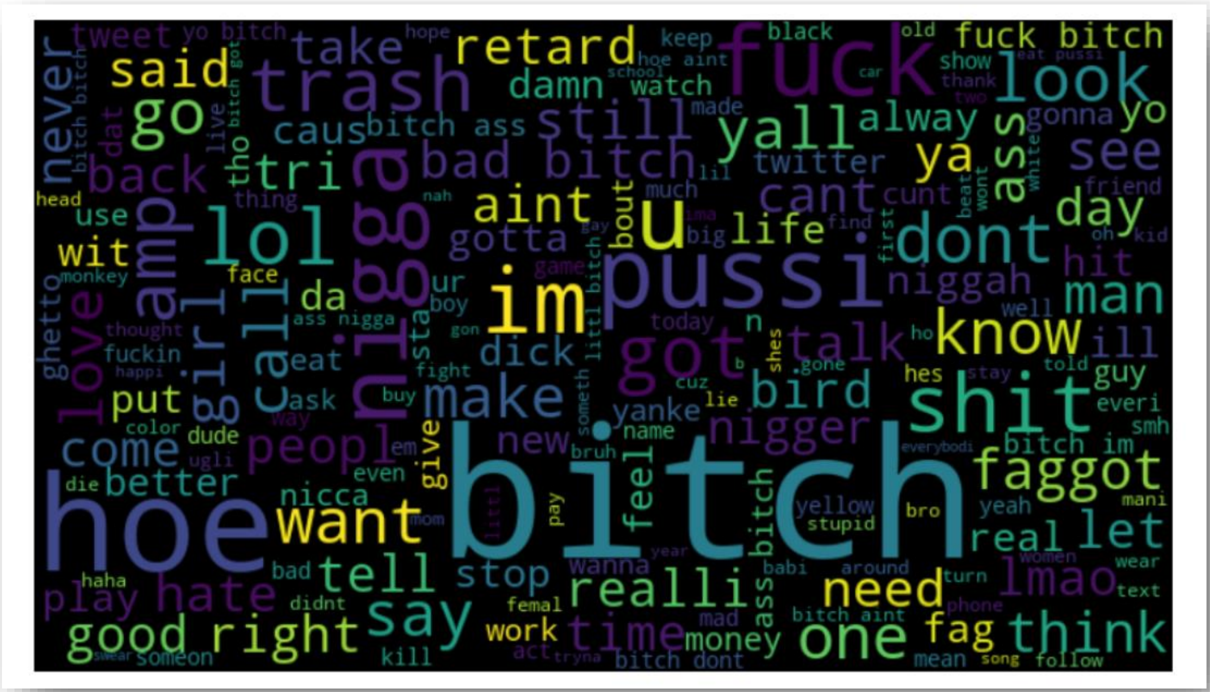
12. Class distribution

Figure 14: Class distribution.



13. WordCloud

Figure 15: Wordcloud under a dark background.



We wanted to generate a word cloud to have a nice visualization of the most common words from the 'tweet' column in our dataset. Some words are larger because they appear more frequently or because they are deemed more significant. We wanted to create a wordcloud for each class, however we lacked the knowledge to so. It can be seen a mixture of neutral, good and very bad words.

Data Processing

To ensure the dataset is ready for analysis, several preprocessing steps were performed. This subsection will explore all the preparation for the data.

These were the steps performed during this phase:

- Stemming
- Stopword Removal
- Removing noise - text cleaning
- Lowercasing tweets

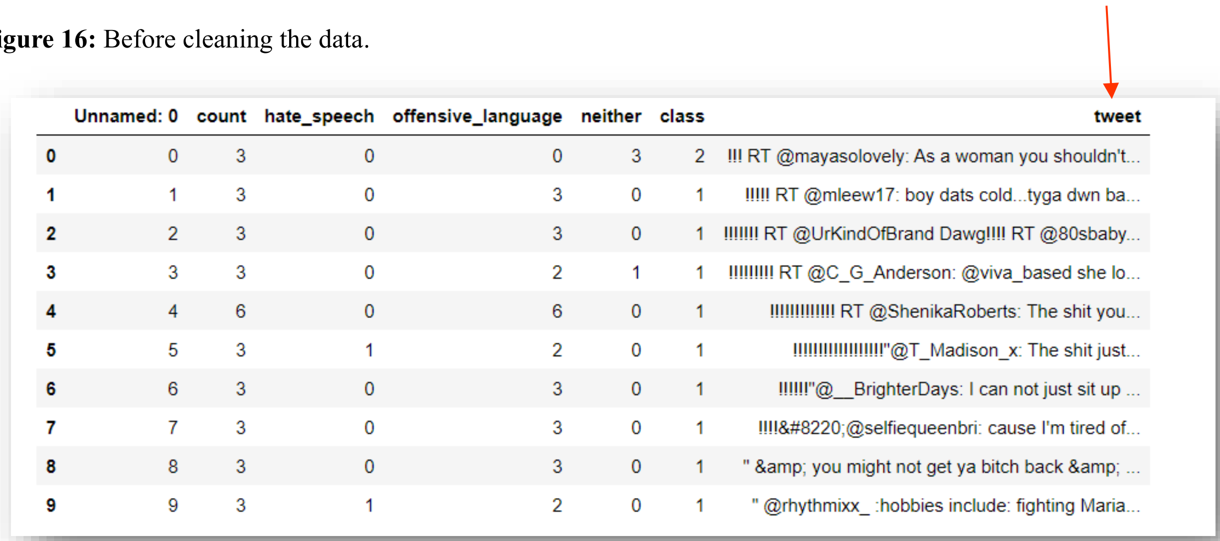
1. Data Cleaning & Stemming

Firstly, our focus was on data cleaning, where we addressed various noise sources present in the text data. We used techniques such as punctuation removal, numeric character elimination, and the elimination of 'RT' (retweet) tags. This preprocessing step ensures that our analysis is conducted on standardized, noise-free data.

- Remove punctuation marks to eliminate unnecessary noise.
- Converting to lowercase for uniformity.
- Eliminate numeric characters if they don't contribute to the analysis.
- Remove specific tags or patterns such as 'RT' (retweet) tags to ensure data consistency.
- Remove Stopwords and apply Stemming.

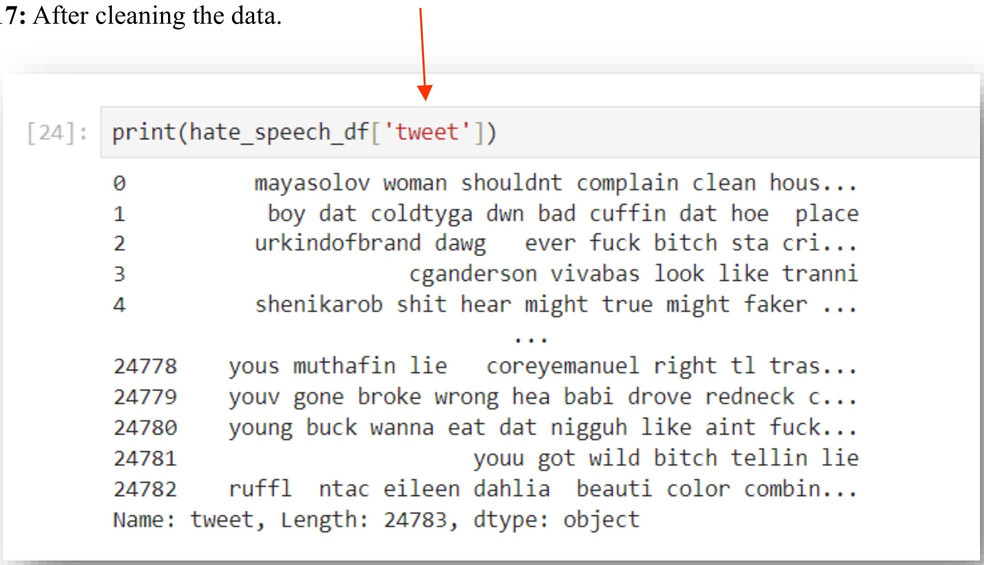
We decided to use the method ‘text.split ()’, instead of using **tokenization** to split the sentences into words. We chose this method due to its simplicity, even though tokenization would have been a more elegant choice.

Figure 16: Before cleaning the data.



	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...
5	5	3	1	2	0	1	!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...
6	6	3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up ...
7	7	3	0	3	0	1	!!!!&#8220;@selfiequeenbri: cause I'm tired of...
8	8	3	0	3	0	1	" &amp; you might not get ya bitch back &amp; ...
9	9	3	1	2	0	1	" @rhythmixx_ :hobbies include: fighting Maria...

Figure 17: After cleaning the data.

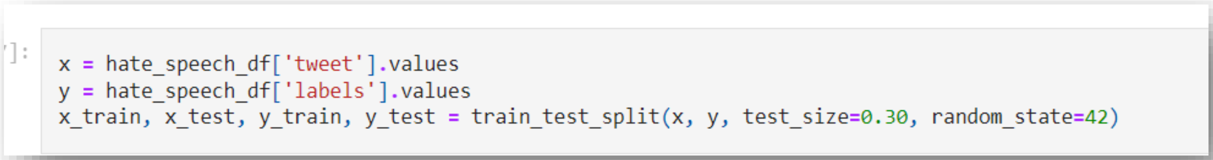


```
[24]: print(hate_speech_df['tweet'])

      0          mayasolov woman shouldnt complain clean hous...
      1           boy dat coldtyga dwn bad cuffin dat hoe  place
      2          urkindofbrand dawg   ever fuck bitch sta cri...
      3                       cganderson vivabas look like tranni
      4          shenikarob shit hear might true might faker ...
                                     ...
      24778    yous muthafin lie   coreyemanuel right tl tras...
      24779    youv gone broke wrong hea babi drove redneck c...
      24780    young buck wanna eat dat nigguh like aint fuck...
      24781                        youu got wild bitch tellin lie
      24782    ruffl  ntac eileen dahlia  beauti color combin...
      Name: tweet, Length: 24783, dtype: object
```

Data Preparation

1. Data Splitting

Figure 18: After cleaning the data.



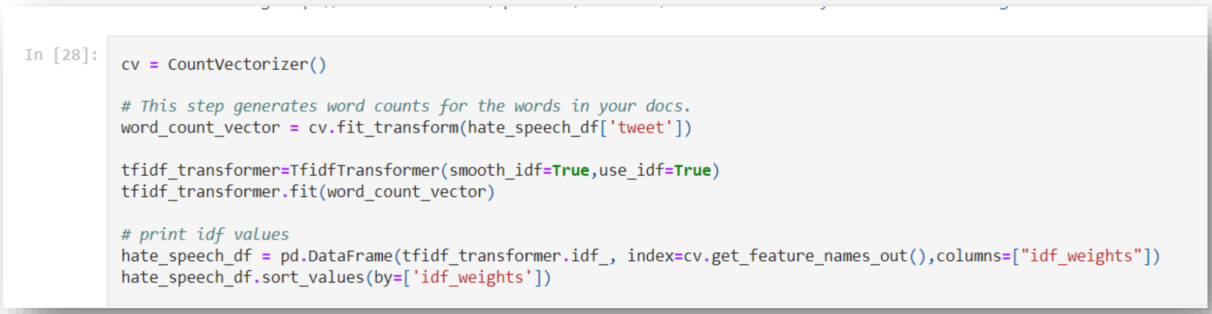
```
]:
    x = hate_speech_df['tweet'].values
    y = hate_speech_df['labels'].values
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=42)
```

30% used for test and 70% used for training.

- x: tweet column
- y: labels column.

2. Count Vectorization & TF-IDF Transformation

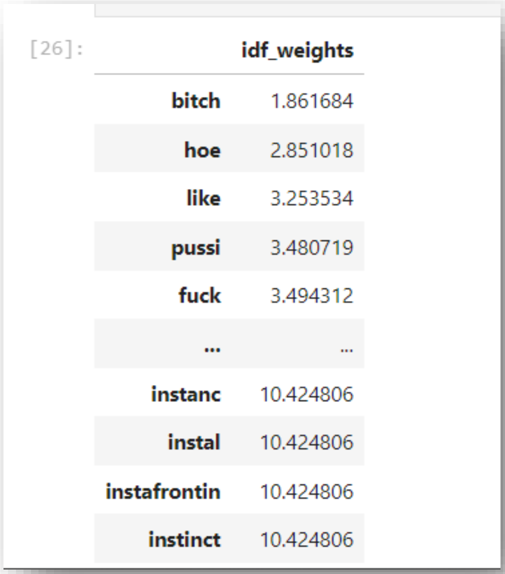
Figure 19: Applying count vectorization and TF-IDF Transformation.



```
In [28]:  cv = CountVectorizer()

          # This step generates word counts for the words in your docs.
          word_count_vector = cv.fit_transform(hate_speech_df['tweet'])

          tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
          tfidf_transformer.fit(word_count_vector)

          # print idf values
          hate_speech_df = pd.DataFrame(tfidf_transformer.idf_, index=cv.get_feature_names_out(),columns=["idf_weights"])
          hate_speech_df.sort_values(by=['idf_weights'])
```

We encoded our data using Count Vectorization, a process known for transforming text data into numerical format. The next step was applying TF-IDF Transformation, which assigns weights to each term based on its frequency in the document and its importance in the corpus. This step adds significance to the encoded data by considering the relevance of each term. Finally, we presented the TF-IDF transformed data, showcasing key insights or top-weighted terms to facilitate interpretation and analysis.

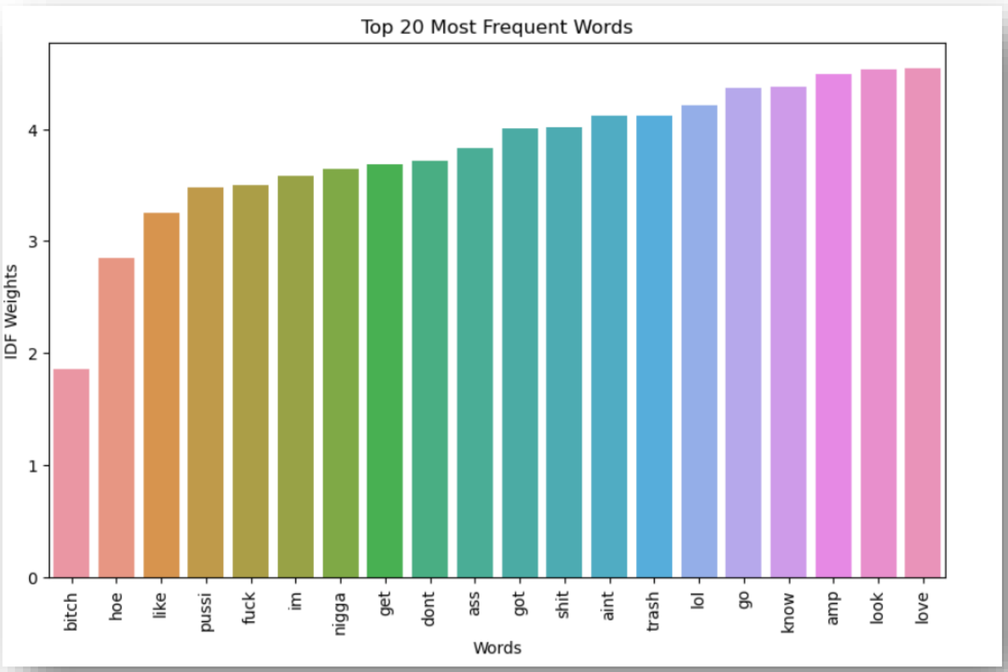
Figure 20: TF-IDF transformed data.



[26]:

	idf_weights
bitch	1.861684
hoe	2.851018
like	3.253534
pussi	3.480719
fuck	3.494312
...	...
instanc	10.424806
instal	10.424806
instafrontin	10.424806
instinct	10.424806

Figure 21: Plotting top 20 most frequent words with *Seaborn.*



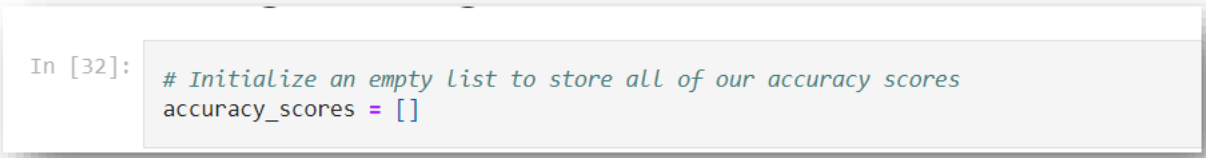
Modelling

For our modelling approach, we chose to use four different classifiers to ensure comprehensive analysis. These include:

1. Logistic Regression Model.
2. Naive Bayes.
3. Classification and Regression Tree.
4. K-Nearest Neighbour.

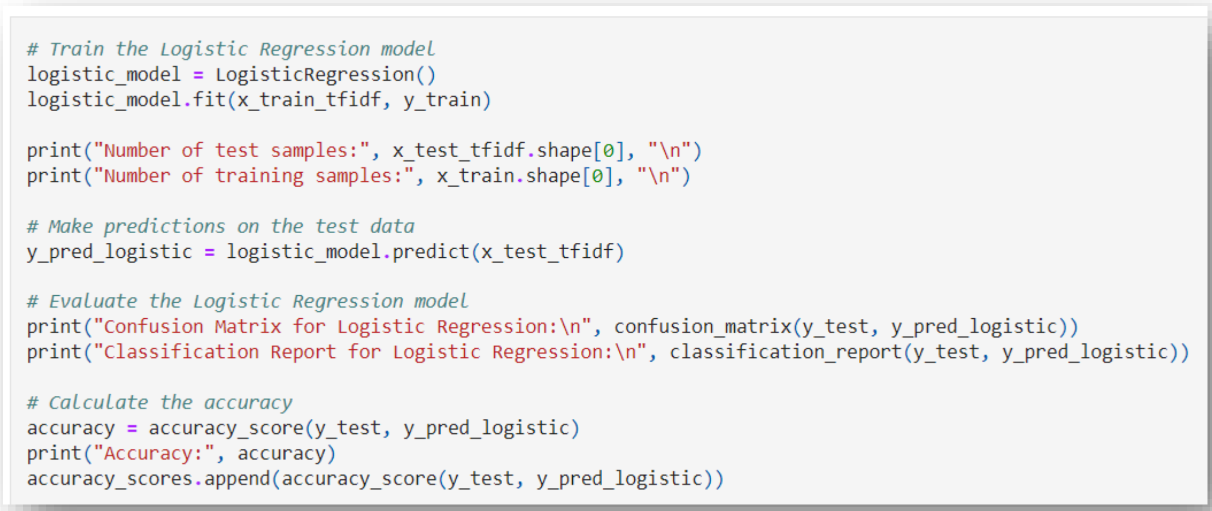
To facilitate comparison and visualization of their performance, we initialize an empty list to store the accuracy scores appended by each model.

Figure 22: Initializing an empty list to store all accuracy results.



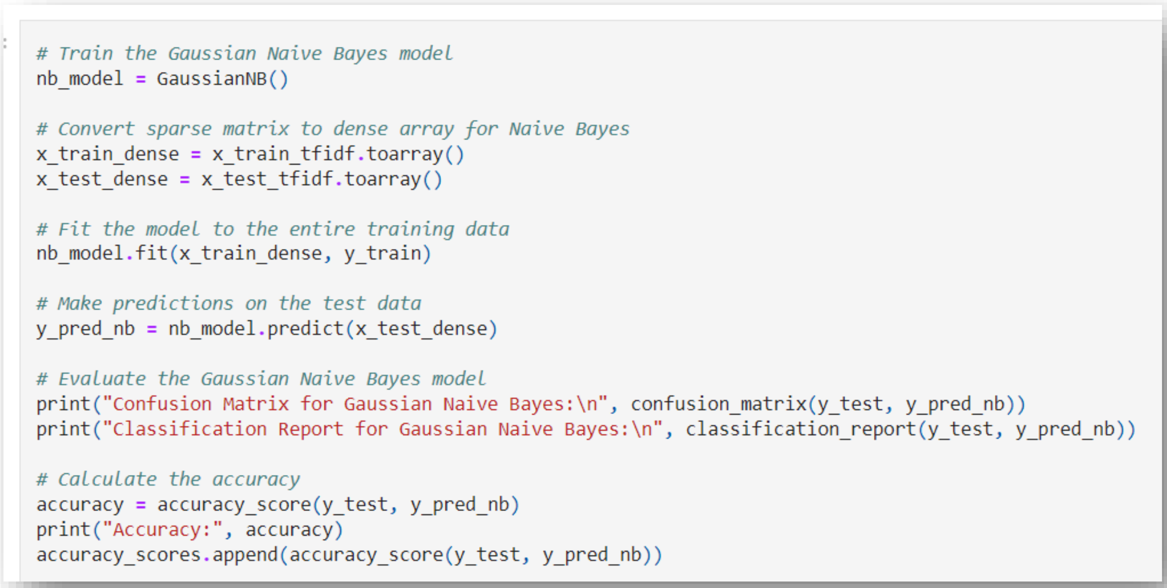
```
In [32]:  # Initialize an empty list to store all of our accuracy scores
          accuracy_scores = []
```

Figure 23: Logistic Regression Model Implementation.



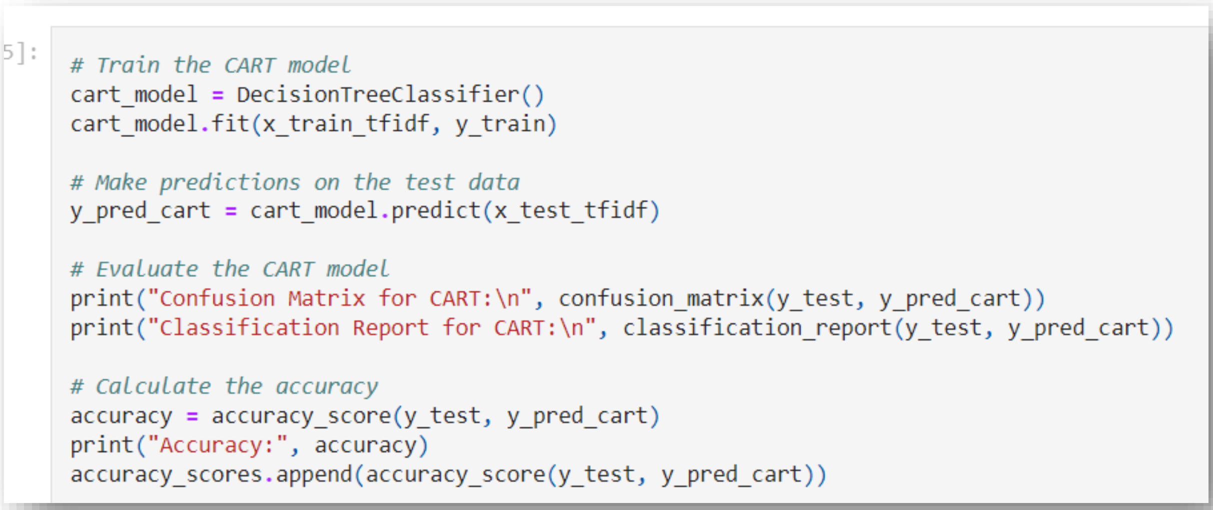
```python
# Train the Logistic Regression model
logistic_model = LogisticRegression()
logistic_model.fit(x_train_tfidf, y_train)

print("Number of test samples:", x_test_tfidf.shape[0], "\n")
print("Number of training samples:", x_train.shape[0], "\n")

# Make predictions on the test data
y_pred_logistic = logistic_model.predict(x_test_tfidf)

# Evaluate the Logistic Regression model
print("Confusion Matrix for Logistic Regression:\n", confusion_matrix(y_test, y_pred_logistic))
print("Classification Report for Logistic Regression:\n", classification_report(y_test, y_pred_logistic))

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred_logistic)
print("Accuracy:", accuracy)
accuracy_scores.append(accuracy_score(y_test, y_pred_logistic))
```

Figure 24: Naïve Bayes Implementation.



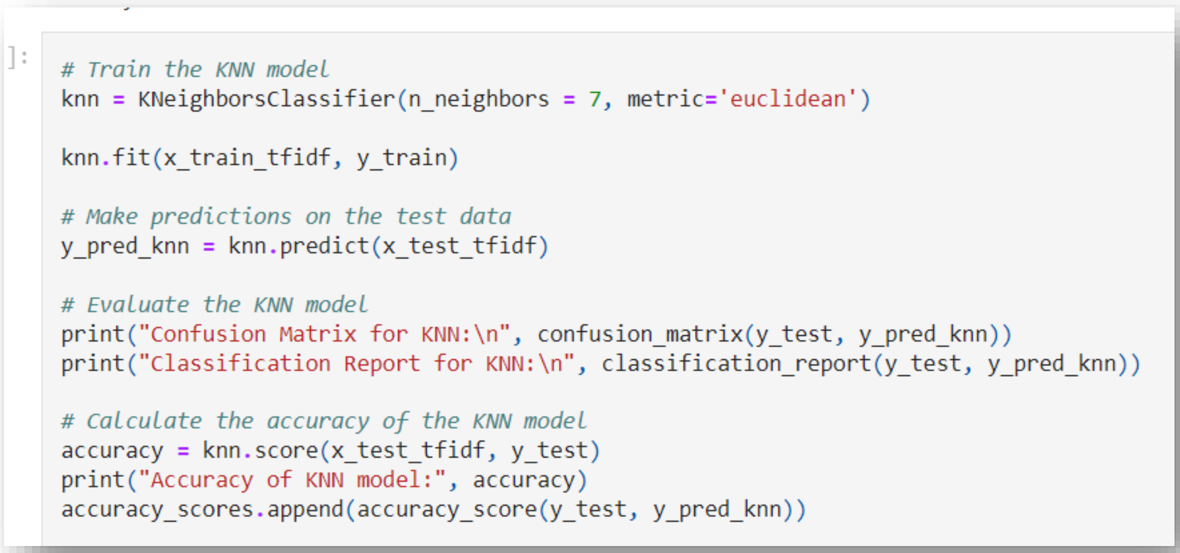
```python
# Train the Gaussian Naive Bayes model
nb_model = GaussianNB()

# Convert sparse matrix to dense array for Naive Bayes
x_train_dense = x_train_tfidf.toarray()
x_test_dense = x_test_tfidf.toarray()

# Fit the model to the entire training data
nb_model.fit(x_train_dense, y_train)

# Make predictions on the test data
y_pred_nb = nb_model.predict(x_test_dense)

# Evaluate the Gaussian Naive Bayes model
print("Confusion Matrix for Gaussian Naive Bayes:\n", confusion_matrix(y_test, y_pred_nb))
print("Classification Report for Gaussian Naive Bayes:\n", classification_report(y_test, y_pred_nb))

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred_nb)
print("Accuracy:", accuracy)
accuracy_scores.append(accuracy_score(y_test, y_pred_nb))
```

Figure 25: Classification and Regression Tree Implementation.



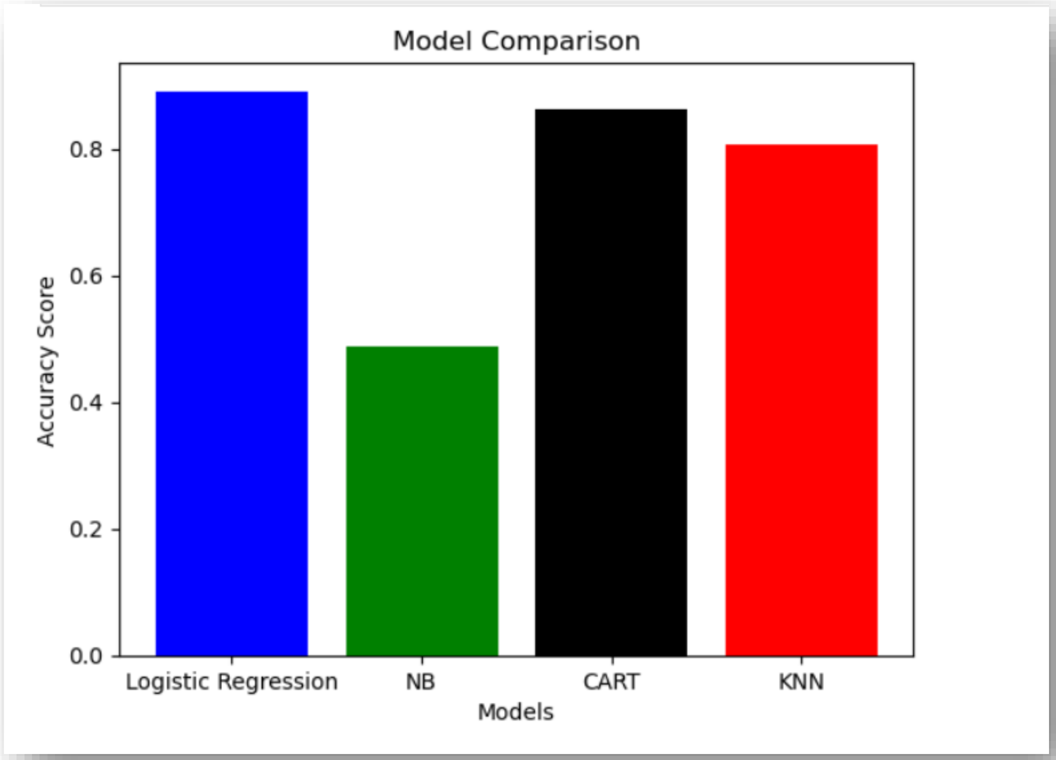
```
5]:  # Train the CART model
     cart_model = DecisionTreeClassifier()
     cart_model.fit(x_train_tfidf, y_train)

     # Make predictions on the test data
     y_pred_cart = cart_model.predict(x_test_tfidf)

     # Evaluate the CART model
     print("Confusion Matrix for CART:\n", confusion_matrix(y_test, y_pred_cart))
     print("Classification Report for CART:\n", classification_report(y_test, y_pred_cart))

     # Calculate the accuracy
     accuracy = accuracy_score(y_test, y_pred_cart)
     print("Accuracy:", accuracy)
     accuracy_scores.append(accuracy_score(y_test, y_pred_cart))
```

Figure 26: KNN Implementation.



```
]:  # Train the KNN model
    knn = KNeighborsClassifier(n_neighbors = 7, metric='euclidean')

    knn.fit(x_train_tfidf, y_train)

    # Make predictions on the test data
    y_pred_knn = knn.predict(x_test_tfidf)

    # Evaluate the KNN model
    print("Confusion Matrix for KNN:\n", confusion_matrix(y_test, y_pred_knn))
    print("Classification Report for KNN:\n", classification_report(y_test, y_pred_knn))

    # Calculate the accuracy of the KNN model
    accuracy = knn.score(x_test_tfidf, y_test)
    print("Accuracy of KNN model:", accuracy)
    accuracy_scores.append(accuracy_score(y_test, y_pred_knn))
```

Figure 27: Plotting models by their accuracy scores.



Logistic Regression and CART presented better results as it can be seen in the plot above. The scores will be further discussed in this section. Moreover, figures 28-33 showcase all the metrics generated by each classifier.

Breakdown of the results:

- **Accuracy**: Among all classifiers that were used, Logistic Regression and CART emerged as the top performers, displaying better results across different metrics, while Naïve Bayes did not outstand, failing to demonstrate effectiveness. While Logistic Regression showed an impressive accuracy of 89%, Naïve Bayes struggled, achieving only 48%. CART achieved 86% and KNN 81%.

Precision: In terms of precision, all models excelled in detecting 'Offensive Language' and neutral contexts, while struggling with 'Hate Speech' identification. CART led in precision for 'Offensive Language' at 92%, while Logistic Regression did well in terms of 'Hate Speech' and neutral context precision. On the other hand, Naive Bayes delivered the weakest results across all categories.

Recall: Considering recall, Logistic Regression achieved an outstanding 96% recall result for 'Offensive Language', showcasing its capacity of capturing true positive instances. However, Naive Bayes demonstrated a surprising strength in 'Hate Speech' recall at 35%, even though being behind in comparison to the other models.

F1 Score: The F1 score is a measure that considers both the precision and recall of the model. Logistic Regression stood as the best performer across all categories with F1 scores of 28% for 'Hate Speech', 94% for 'Offensive Language', and 82% for 'Neutral'.

Figure 28: All four confusion matrices with sub plotting for a nicer and concise visualization.

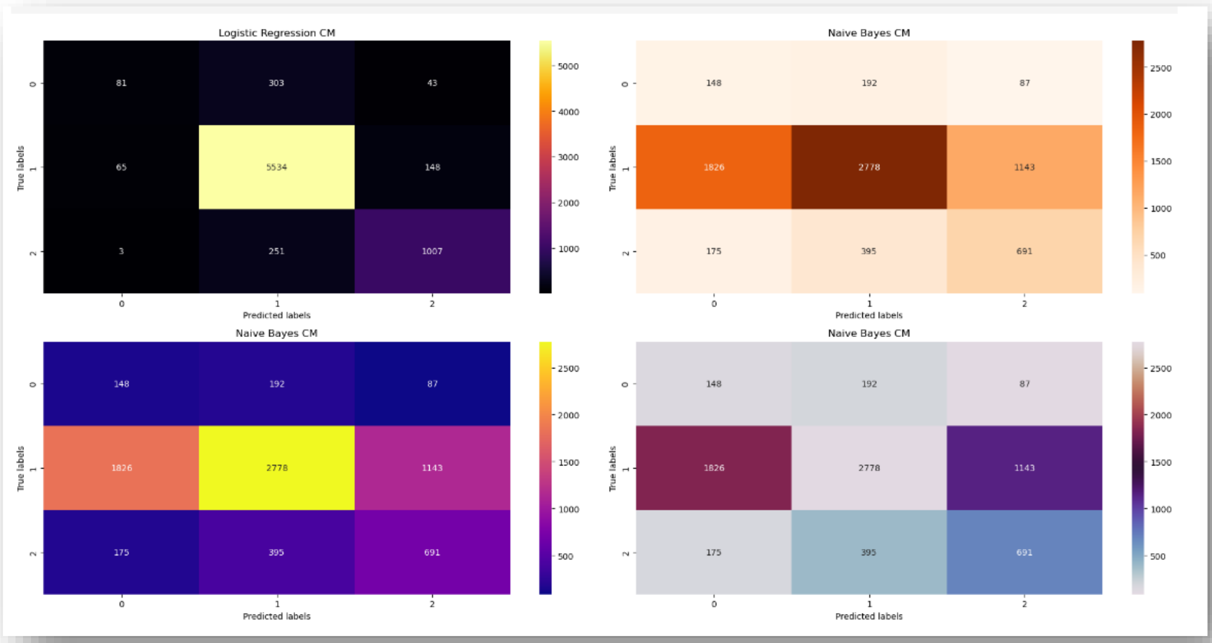
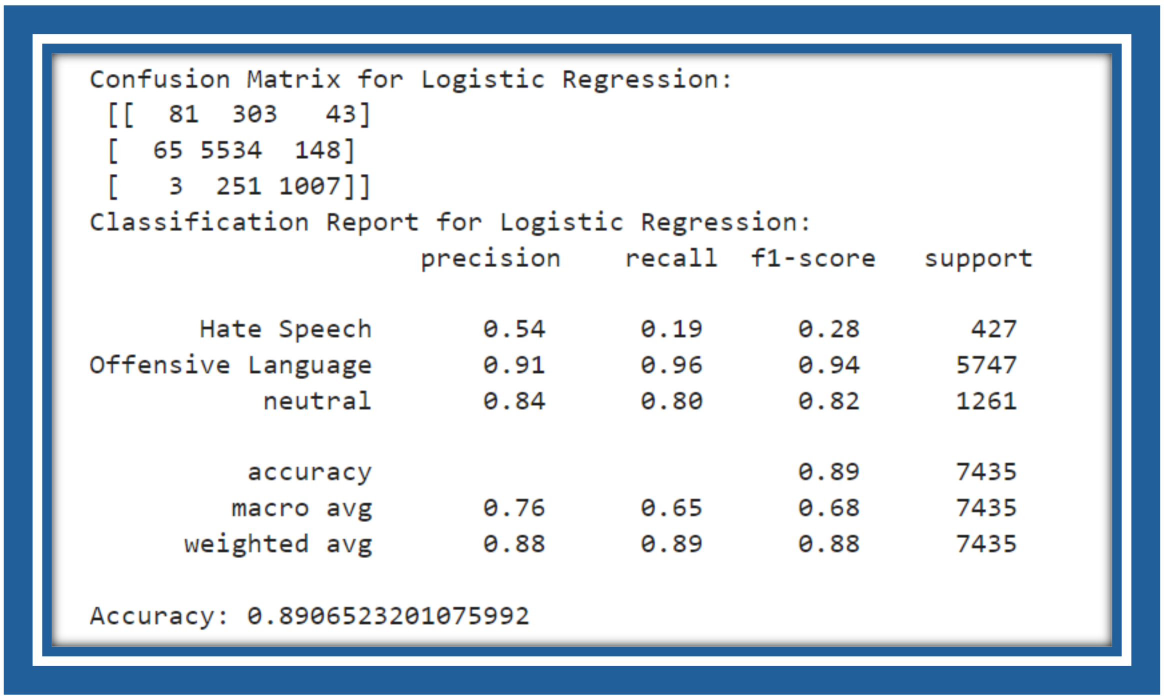
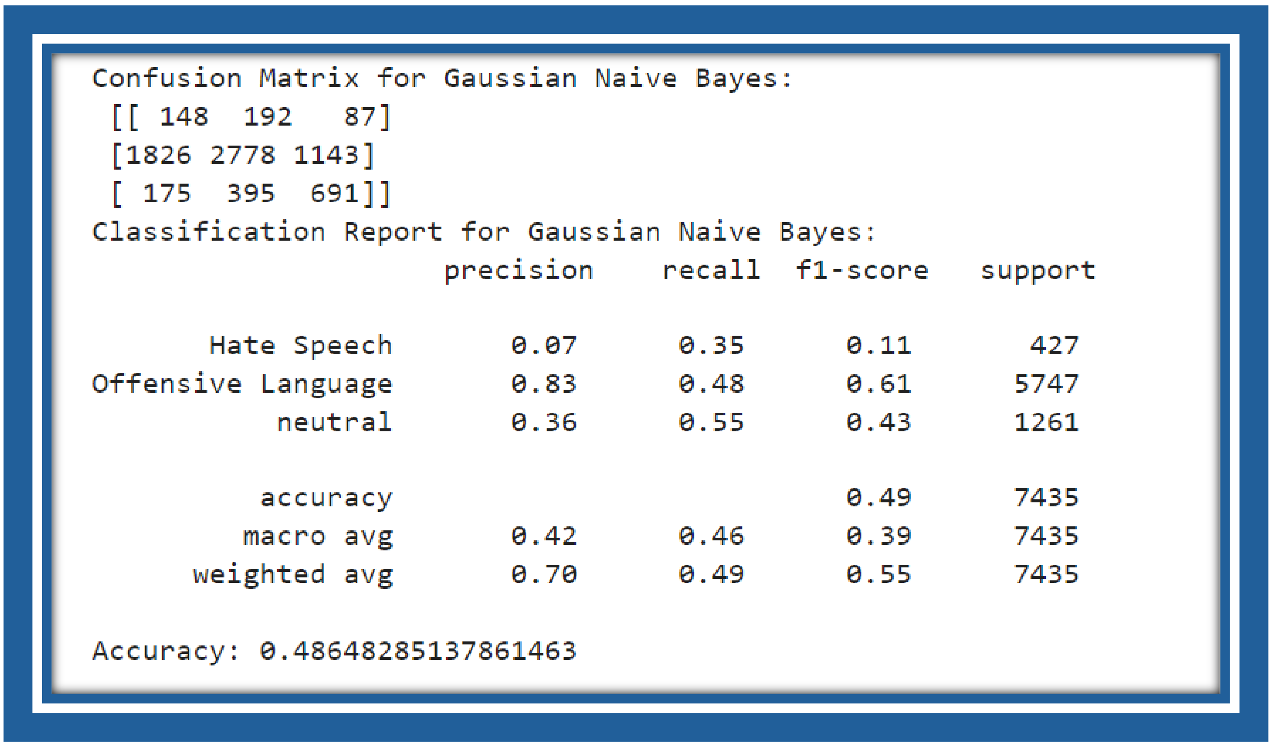


Figure 29: Linear Regression results.



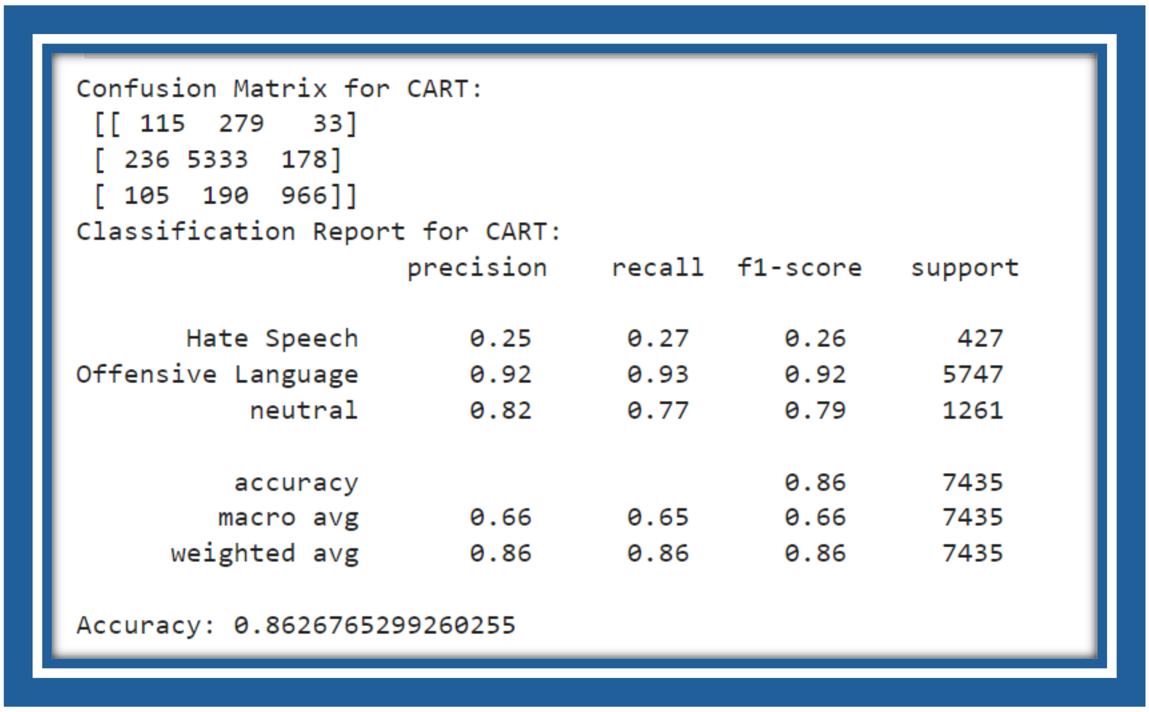
```
Confusion Matrix for Logistic Regression:
 [[  81  303   43]
 [  65 5534  148]
 [   3  251 1007]]
Classification Report for Logistic Regression:
                   precision    recall  f1-score   support

      Hate Speech       0.54      0.19      0.28       427
Offensive Language       0.91      0.96      0.94      5747
          neutral       0.84      0.80      0.82      1261

         accuracy                           0.89      7435
        macro avg       0.76      0.65      0.68      7435
     weighted avg       0.88      0.89      0.88      7435

Accuracy: 0.8906523201075992
```

Figure 30: Naïve Bayes results.



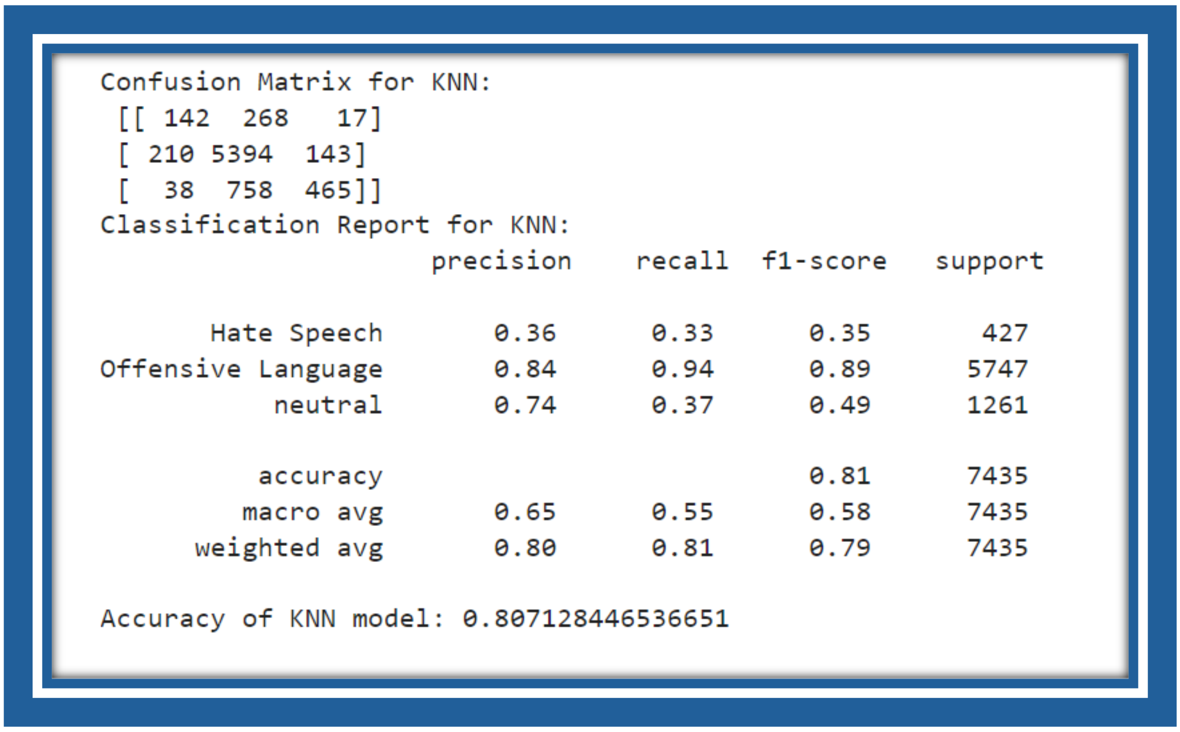
```
Confusion Matrix for Gaussian Naive Bayes:
 [[ 148  192   87]
 [1826 2778 1143]
 [ 175  395  691]]
Classification Report for Gaussian Naive Bayes:
                   precision    recall  f1-score   support

      Hate Speech       0.07      0.35      0.11       427
Offensive Language       0.83      0.48      0.61      5747
          neutral       0.36      0.55      0.43      1261

         accuracy                           0.49      7435
        macro avg       0.42      0.46      0.39      7435
     weighted avg       0.70      0.49      0.55      7435

Accuracy: 0.48648285137861463
```

Figure 31: CART results.



```
Confusion Matrix for CART:
 [[ 115  279   33]
 [ 236 5333  178]
 [ 105  190  966]]
Classification Report for CART:
                    precision    recall  f1-score   support

       Hate Speech       0.25      0.27      0.26       427
Offensive Language       0.92      0.93      0.92      5747
           neutral       0.82      0.77      0.79      1261

          accuracy                           0.86      7435
         macro avg       0.66      0.65      0.66      7435
      weighted avg       0.86      0.86      0.86      7435

Accuracy: 0.8626765299260255
```

Figure 32: KNN results.



```
Confusion Matrix for KNN:
 [[ 142  268   17]
 [ 210 5394  143]
 [  38  758  465]]
Classification Report for KNN:
                    precision    recall  f1-score   support

       Hate Speech       0.36      0.33      0.35       427
Offensive Language       0.84      0.94      0.89      5747
           neutral       0.74      0.37      0.49      1261

          accuracy                           0.81      7435
         macro avg       0.65      0.55      0.58      7435
      weighted avg       0.80      0.81      0.79      7435

Accuracy of KNN model: 0.807128446536651
```

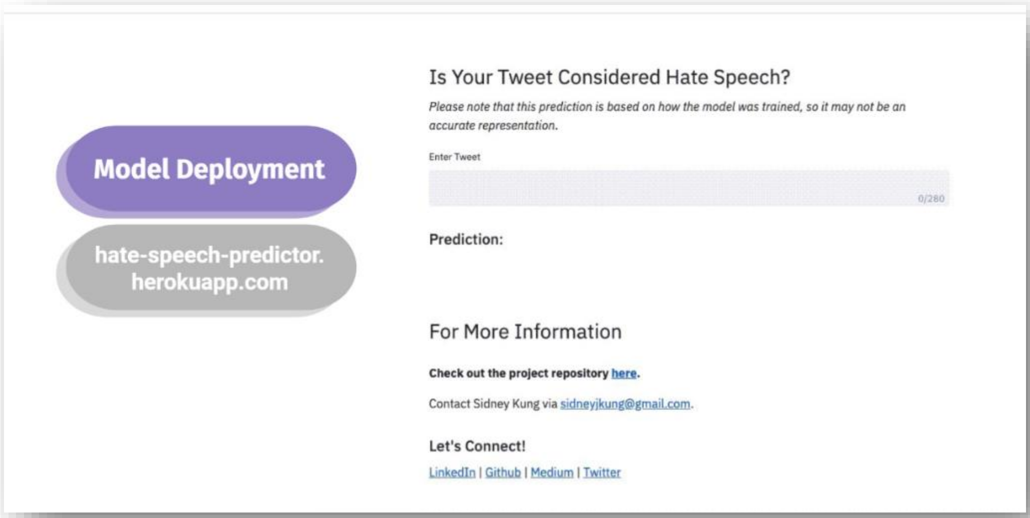
Deployment

Machine learning serves as the cornerstone of our project as it empowers computer systems to learn from data and to refine their performance from experience over time. Our goal for business would be to develop a real-time detection and scalable system to provide significant market opportunities by collaborating with social media companies and potentially law enforcement agencies.

In this phase, we would prioritize security and compliance with ethical considerations (discussed in the 'Ethical and legal aspects' section), ensuring responsible handling of data and transparent communication of model limitations and biases. By deploying an efficient detection system, we aim to contribute towards mitigating the harmful effects of hate speech and build a more inclusive online community.

The model deployment below by Kung (2020) is a great example of what we'd like to build in the future.

Figure 33: KNN results.



Is Your Tweet Considered Hate Speech?

Please note that this prediction is based on how the model was trained, so it may not be an accurate representation.

Enter Tweet

Prediction:

For More Information

Check out the project repository here.

Contact Sidney Kung via sidneyjkung@gmail.com.

Let's Connect!

LinkedIn | Github | Medium | Twitter

Source: GitHub. (n.d.). Twitter hate speech detection. [online] Available at: https://github.com/sidneykung/twitter_hate_speech_detection/blob/master/final_notebook.ipynb [Accessed 23. Mar 2024].

Individual Report Thayene

I encountered no issues with our task distribution because I chose my partner wisely. Working with my twin sister, who is in the same class, made it easy to split the workload evenly. My responsibilities included:

- Poster creation.
- Word file formatting.
- Report writing (50/50).
- Coding (collaborative).
- Video production (collaborative).
- Dataset and topic selection (collaborative).
- Data understanding and exploratory data analysis (EDA).
- Modelling: Logistic Regression & Naïve Bayes.
- Evaluation (Confusion Matrix Plotting) and accuracy scores plotting.
- Part of markdown documentation.

I am pleased with the final look of our project, despite encountering several obstacles along the way. I am grateful for the support from Muhammad and Ken, and I appreciate that machine learning was a requirement for the capstone project. At times, I strongly feared we wouldn't be able to finish this project due to being behind in comparison to our peers, but despite everything, we managed to meet all assignment requirements.

Individual Report Mayara

Our commits on GitHub were very well balanced, though we faced challenges with GitHub Desktop, including file corruption that required recommitting from scratch. To mitigate this, we kept drafts saved and occasionally would have to recommit them. Living together certainly facilitated our collaboration on this assignment. Although it took some time to choose a topic, we were satisfied with the project's outcome. My responsibilities included:

- PowerPoint slide creation.
- Report writing (50/50).
- Coding (collaborative).
- Video production (collaborative).
- Dataset and topic selection (collaborative).
- Data processing.
- Data preparation.
- Modelling: CART & KNN.
- Evaluation.
- Part of markdown documentation.

Working with categorical data was particularly challenging for me. In future projects, I might prefer to focus on numerical data instead. At times, I felt like giving up on the whole project, but the support from the IT faculty, along with my colleagues and friends, motivated me to persevere and complete this assignment. Having my sister with me was the best part, because we tend to think alike and that made it easier for me.

Final Considerations

This project proved to be very rewarding for both of us as it helped us to further refine our Python and Machine Learning skills. However, if given the chance, we would have liked to have had more time for the topic selection.

Undoubtedly, the biggest challenge in this assignment was sourcing a wide variety of datasets because of the type of content. While we encountered a few datasets available online, many were in languages other than English, which did not align with the focus of our project.

Moreover, the absence of datasets reflecting specific regional contexts, such as Irish data, was a challenge we faced throughout this project, since we were unable to locate an Irish dataset available for download.

References

Ramírez-García, A. et al. (2022) 'Interdisciplinarity of Scientific Production on Hate Speech and Social Media: A Bibliometric Analysis', Comunicar: Media Education Research Journal, 30(72), pp. 123–134. Available at: https://research.ebsco.com/linkprocessor/plink?id=529e6fa7-98d8-3129-9bcf-1953609ca289 [Accessed: 15 Mar. 2024].

Cambridge Dictionary (2019). HATE SPEECH | meaning in the Cambridge English Dictionary. [online] Cambridge.org. [Accessed: 15 March 2024]. Available at: https://dictionary.cambridge.org/dictionary/english/hate-speech.

Citizensinformation.ie (n.d.). *The law on hate speech*. [online] www.citizensinformation.ie. Available at: https://www.citizensinformation.ie/en/justice/criminal-law/criminal-offences/law-on-hate-speech/#208370 [Accessed 15 Mar. 2024].

Davidson, T.; Warmsley, D.; Macy, M.; Weber, I. Automated hate speech detection and the problem of offensive language. In Proceedings of the Eleventh International AAAI Conference on Web and Social Media, Montreal, QC, Canada, 15–18 May 2017. [Accessed 15.Mar 2024].

Sossi Alaoui, Safae & Farhaoui, Yousef & Aksasse, B.. (2022). Hate Speech Detection Using Text Mining and Machine Learning. International Journal of Decision Support System Technology. 14. 1-20. 10.4018/IJDSST.286680. [Accessed 23.Mar 2024].

Toktarova, Aigerim & Syrlybay, Dariga & Myrzakhmetova, Bayan & Anuarbekova, Gulzat & Rakhimbayeva, Gulbarshin & Zhylanbaeva, Balkiya & Suieuova, Nabat & Kerimbekov, Mukhtar. (2023). Hate Speech Detection in Social Networks using Machine Learning and Deep

Learning Methods. International Journal of Advanced Computer Science and Applications. 14. 10.14569/IJACSA.2023.0140542. [Accessed 23.Mar 2024].

de Gibert Bonet, Ona & Perez, Naiara & García-Pablos, Aitor & Cuadros, Montse. (2018). Hate Speech Dataset from a White Supremacy Forum. [Accessed 27.Mar 2024].

Kumar, A. (2020). Twitter Sentiment Analysis. [online] Analytics Vidhya. Available at: https://medium.com/analytics-vidhya/twitter-sentiment-analysis-b9a12dbb2043. [Accessed 01.Apr 2024].

Coursesteach (2024). Hate Speech Detection with Machine Learning. [online] Medium. Available at: https://medium.com/@Coursesteach/hate-speech-detection-with-machine-learning-c7d12a217e69 [Accessed 02.Apr 2024].

www.justthink.ai. (n.d.). The Persistent Challenges of Content Moderation at Scale - Just Think AI. [online] Available at: https://www.justthink.ai/blog/the-persistent-challenges-of-content-moderation-at-scale#:~:text=Moderating%20user%2Dgenerated%20content%20is [Accessed 6 May 2024].

Kane, C. (2022). The Real Challenge in Content Moderation. [online] VISUA. Available at: https://visua.com/challenge-in-content-moderation. [Accessed 15.Mar 2024]

Dataset downloaded from: data.world. (n.d.). Hate Speech Identification - dataset by crowdflower. [online] Available at: https://data.world/crowdflower/hate-speech-identification.

kaggle.com. (n.d.). Text Data Cleaning - tweets analysis. [online] Available at: https://www.kaggle.com/code/ragnisah/text-data-cleaning-tweets-analysis [Accessed 6 May 2024].

GitHub. (n.d.). twitter_hate_speech_detection/final_notebook.ipynb at master · sidneykung/twitter_hate_speech_detection. [online] Available at: https://github.com/sidneykung/twitter_hate_speech_detection/blob/master/final_notebook.ipynb [Accessed 23. Mar 2024].

Dr. Ernesto Lee (2024). Use Machine Learning to Combat Hate Speech: A Python Approach that Creates an API. [online] Medium. Available at: https://drlee.io/use-machine-learning-to-combat-hate-speech-a-python-approach-that-creates-an-api-ae73737cc427 [Accessed 23. Mar 2024].

GitHub. (n.d.). Hate-Speech-Detection-in-Social-Media-using-Python/final_customization.ipynb at master · NakulLakhotia/Hate-Speech-Detection-in-Social-Media-using-Python. [online] Available at: https://github.com/NakulLakhotia/Hate-Speech-Detection-in-Social-Media-using-Python/blob/master/final_customization.ipynb [Accessed 27. Mar 2024].

t-davidson (2017). hate-speech-and-offensive-language/src/Automated Hate Speech Detection and the Problem of Offensive Language.ipynb at master · t-davidson/hate-speech-and-offensive-language. [online] GitHub. Available at: https://github.com/t-davidson/hate-speech-and-offensive-language/blob/master/src/Automated%20Hate%20Speech%20Detection%20and%20the%20Problem%20of%20Offensive%20Language.ipynb [Accessed 27. Mar 2024].

KumarAnand11 (2024). INFO-7390-Project/Twitter Sentiment Analysis.ipynb at master · KumarAnand11/INFO-7390-Project. [online] GitHub. Available at: https://github.com/KumarAnand11/INFO-7390-Project/blob/master/Twitter%20Sentiment%20Analysis.ipynb [Accessed 01 Apr 2024].

Rights for Peace. (n.d.). What is Hate Speech? [online] Available at: https://www.rightsforpeace.org/hate-speech#:~:text=Speech%20that%20is%20simply%20offensive [Accessed 23 May 2024].