Artificial intelligence (CSE422)
Project Name: Hate speech and offensive language detection from a Twitter dataset

Group :
Group members:

Showmick kar (20301177)
Sajidul Islam Khandaker (20301190)
Soumik Deb Niloy(20301207)
Md Ashiqur Rahman()

Submitted to :

Md. Asif Talukdar
&
Syed Zamil Hasan

INTRODUCTION

It is 2022. Since the dawn of civilization, our technology has come a long way. Now anyone with a smartphone device and an internet connection can connect with each other. This is very positive but having many people access the social world leads to mixed opinions and words thrown at each other. It is easier to type something cruel than to say it in someone's face. This leads to people commenting or talking negatively with cruel comments, harsh words, racist comments, insulting expressions, and many more. This is called hate speech. The barrier that can divide people from expressing hatred through comments and texts can be hate speech or offensive language detectors and ban them.

This is why we come up with hate speech detectors that could silently erase the hatred and make the social world a little bit cleaner.

METHODOLOGY

Dataset Description: The 'Hate speech and offensive language dataset' is used for this project and 6000 rows from the dataset are used to train where 80% rows are labeled as training dataset and 20% rows are labeled as testing dataset. Each row in the dataset is labeled as hate speech and offensive language.

Libraries Used: Following libraries would be used for this project

- **Pandas** - for reading the CSV file
- **Numpy**- for arrays
- **Matplotlib** - for graph plotting
- **Sklearn** - for implementing machine learning models
- **Seaborn** - for making statistical graphics
- **NLTK** - for text data preprocessing

PROJECT ARCHITECTURE:

Our project is a combination of several subtasks and processes. Starting with loading the dataset into the pandas dataframe, we first oversampled the dataset to combat discrimination among the different classes. Then we applied text preprocessing to get rid of unhelpful and noisy data. Moving on, we applied feature engineering to vectorize the raw text data. Following this, we split the dataset into train and test sets and fed the train data into our models. Finally, we ran accuracy metrics to measure and compare the performance of all three algorithms.

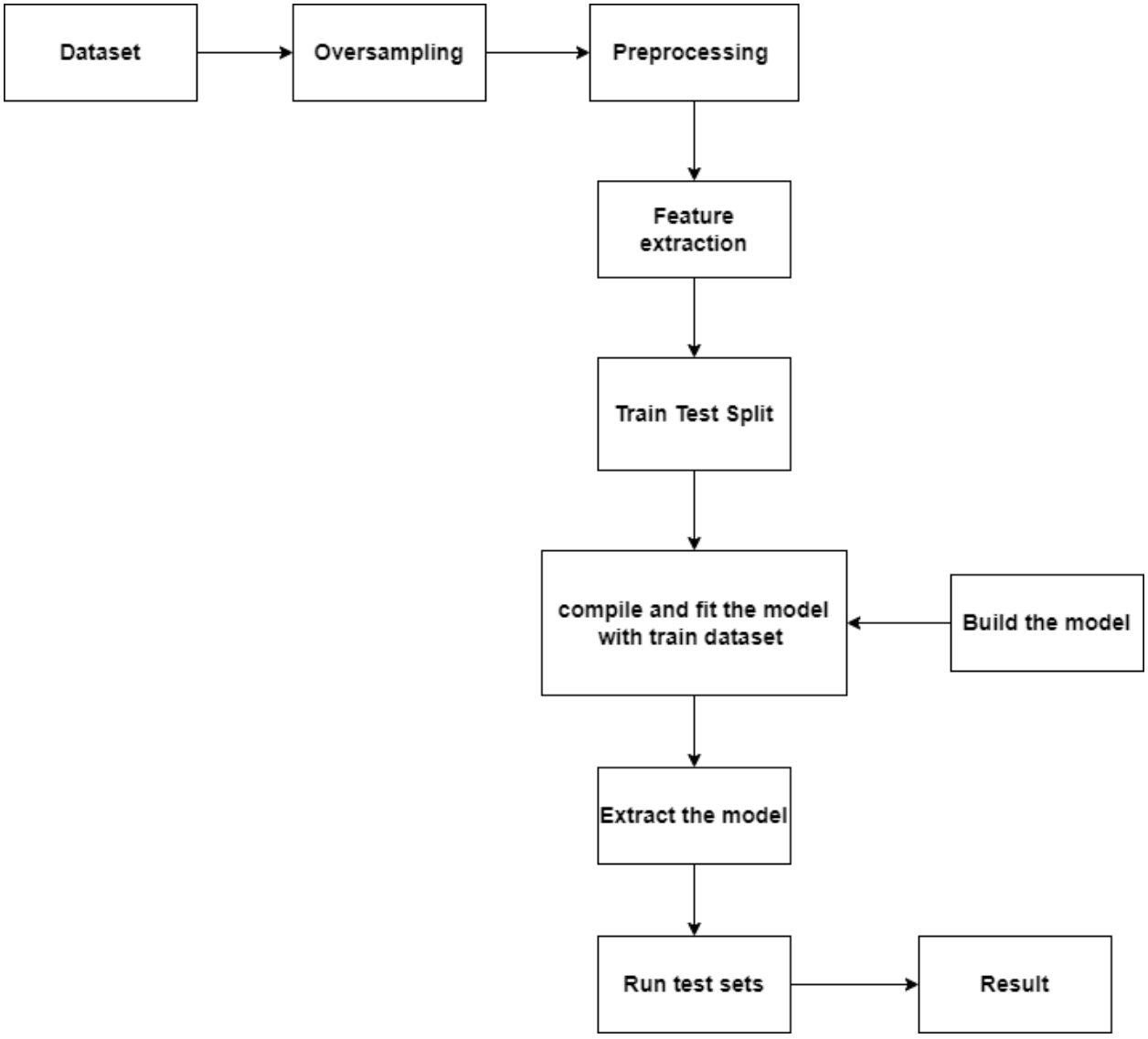

Figure 1- Project Architecture

PRE-PROCESSING TECHNIQUES APPLIED

Many research studies have said that using text preprocessing makes better classification results. In this project, we've used different preprocessing techniques, some of which are generalized ML techniques to reduce noise, and others are directly related to NLP tasks.

Firstly, from figure 1, it is visible that our dataset is imbalanced and the outputs are unevenly distributed for different classes. This imbalance can have adverse effects on overall how our models perform in terms of precision, recall, and f1 scores. To overcome this, we've used data oversampling to even the dataset throughout all three classes.

Then, we've used some preprocessing techniques which are more specific to NLP tasks. These techniques have been used to filter unnecessary and non-informative features from the dataset. The techniques are listed below.

1. Removal of Punctuations
2. Lower Casing
3. Tokenization
4. Removal of Stop Words
5. Lemmatization

FEATURE EXTRACTION

In order for our raw text data to be fed into the Machine Learning models, they first needed to be transformed into numerical vectorized form without losing any information. For this, we've used the Bag-of-Words model using monograms. In this representation, each tweet has been transformed into a vector that contains the frequency of each word that occurred within it.

MODELS APPLIED

After the completion of count vectorization using bag-of-words, our dataset is finally ready to train. In this project, we have used three classification algorithms to train over our dataset. The algorithms are as follows.

1. **Logistic Regression**
2. **Naive Bayes Classifier**
3. **Kth Nearest Neighbor(KNN)**

RESULT

ACCURACY MATRICES:

This section elaborates on the performance of the three models we've used for text classification. From Table 1.1 it is evident that Logistic Regression has the best performance among the different models applied with the Naive Bayes Classifier showing the worst performance. The accuracy metrics we've taken are precision score, recall score, and f1 score. The precision score measures how accurately the model predicts the true positives. The recall score measures the model's ability to find the true positives among all the positive correct outputs. F1 score is a harmonic mean of the precision score and recall score. The results for the three algorithms are stored in Tables 1, 2, and 3.

From Tables 1, 2, and 3. It is clearly seen that Logistic Regression performed best among all three algorithms while Naive Bayes Classifier was the worst. For example, the precision scores for correctly predicting the true positives for "Hate Speech" for Logistic Regression, Naive Bayes Classifier, and Kth Nearest Neighbor are respectively 1, 0.98, and 0.99. The relatively bad performance of NVC was due to the algorithm's conditional independence of its features. The algorithm starts to show weakness as the number of features increases.

Logistic Regression	Precision	recall	f1_score
0	0.96	1.0	0.98
1	1.0	0.91	0.95
2	0.95	1.0	0.97

Table 1- Logistic Regression

Naive Bayes Classifier	Precision	recall	f1_score
0	0.89	1.0	0.94
1	0.98	0.77	0.86
2	0.89	0.99	0.94

Table 2-Naive Bayes Classifier

Kth Nearest Neighbor(KNN)	Precision	recall	f1_score
0	0.93	1.0	0.97
1	0.99	0.81	0.89
2	0.88	0.99	0.93

Table 3- Kth Nearest Neighbor(KNN)

CONFUSION MATRIX:

Below are the confusion matrices for each of our models. The confusion matrix shows the summary of the performance of a classification algorithm. The values in the primary diagonal represent the number of times models correctly predicted a label. The other values represent incorrect predictions.

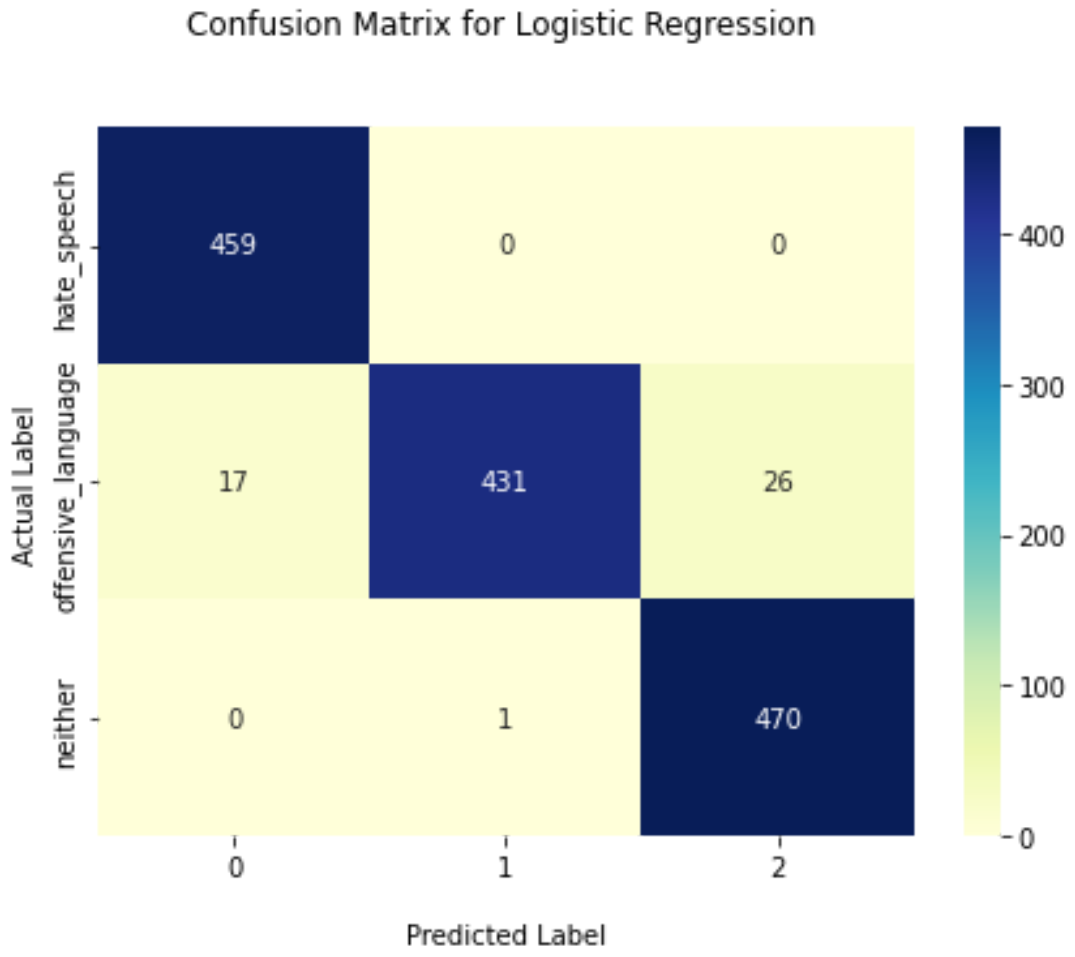

Figure 2: Confusion matrix for Logistic Regression

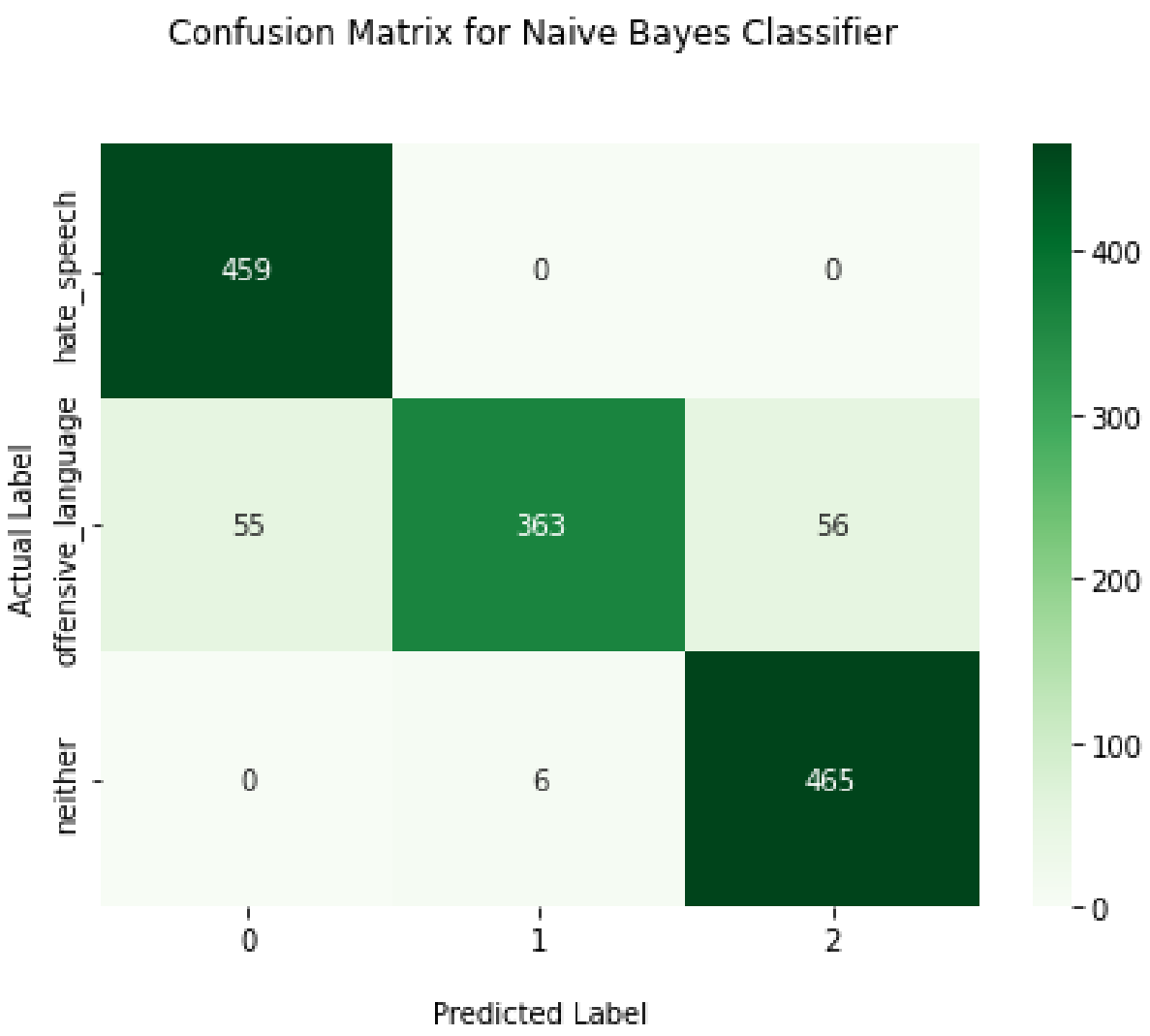

Figure 3: Confusion matrix for Naive Bayes Classifier

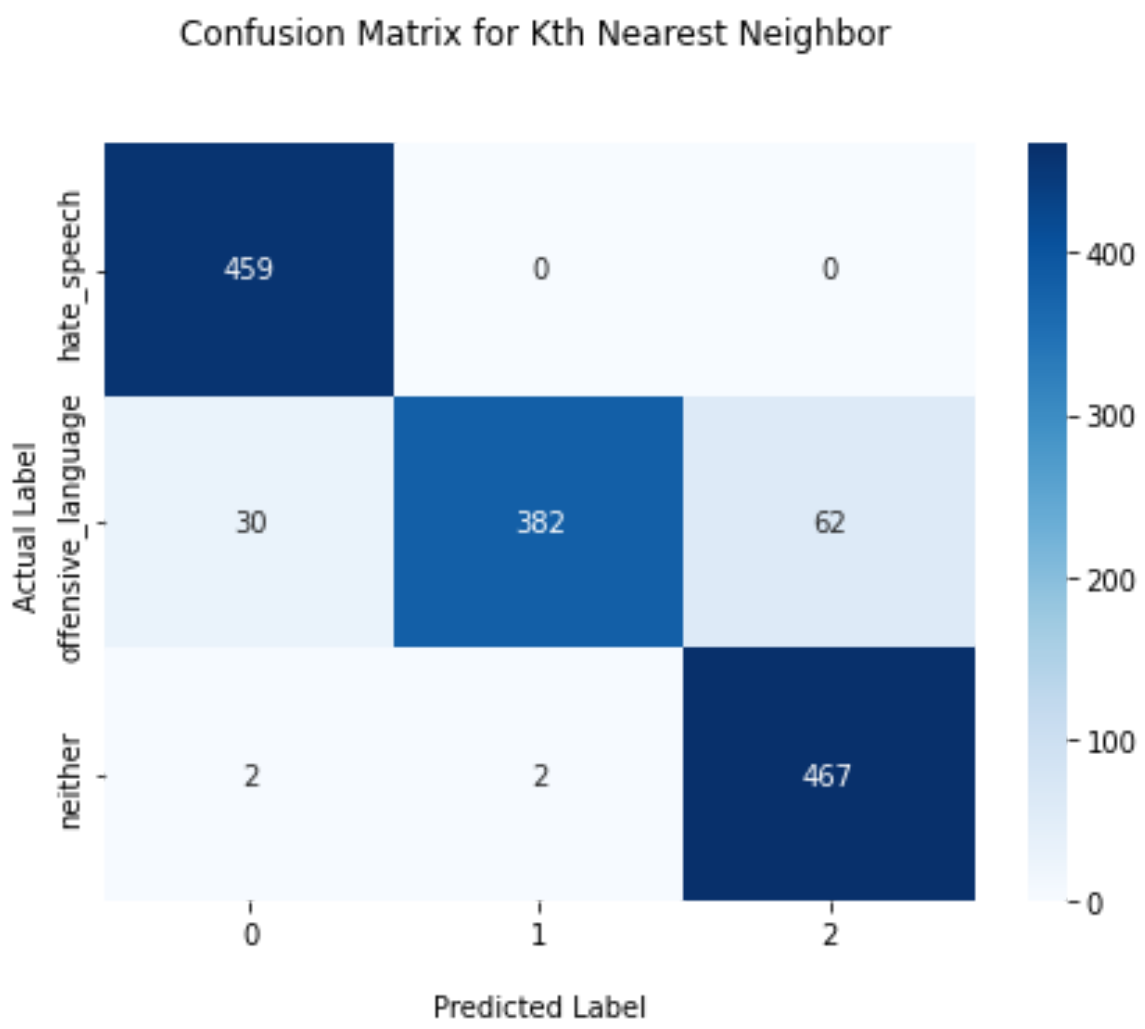

Figure 4: Confusion matrix for Kth Nearest Neighbor(KNN)

Conclusion:

Throughout this project we automated text classification techniques to classify hate speech and offensive tweets . We have used three classification algorithms to train over our dataset - Logistic Regression, Naive Bayes Classifier, and Kth Nearest Neighbor(KNN). Among them, Logistic Regression has the best accuracy, and naive Bayes Classifier has the worst performance results.

References:

1. *Deepansi*, "Text preprocessing NLP: Text preprocessing in NLP with python codes," *Analytics Vidhya*, 19-Jul-2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/. [Accessed: 31-Aug-2022].
2. *Pynative.com*. [Online]. Available: https://pynative.com/python-regex-split/. [Accessed: 31-Aug-2022].
3. "How to remove any URL within a string in Python," *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/11331982/how-to-remove-any-url-within-a-string-in-python. [Accessed: 31-Aug-2022].
4. V. Aruchamy, "How to plot confusion matrix in python and why you need to?," *Stack Vidhya*. Available at: https://www.stackvidhya.com/plot-confusion-matrix-in-python-and-why/#:~:text=Plot%20Confusion%20Matrix%20for%20Binary%20Classes%20With%20Labels&text=You%20need%20to%20create%20a,matrix%20with%20the%20labels%20annotation. [Access: 29-sept-2021]