University of North Texas

Course: Natural Language Processing

Hate Speech Detection Using Twitter Tweets.

December 2, 2021

Sai Anjali Potula -SaiAnjaliPotula@my.unt.edu

Seemaparvez Shaik -seemaparvezshaik@my.unt.edu

Vishnu Manoj Deepala-vishnumanojdeepala@my.unt.edu

Introduction

As internet access increases and more people join social media platforms, people tend to post online as they wish and label it as free. speech. This is one of the main problems of social media, Overall user experience. Hate speech (HS) is a type of offensive public speech that is based on a specific individual or group. Characteristics such as race, religion, ethnic origin, country of origin, gender, disability, sexual orientation, and gender identity. This phenomenon manifests itself verbally or physically (language, text, gestures, etc.) The origin of racism and ethnocentrism. Due to the social cost of hate speech, some countries are considering this Illegal activity, especially if violence or hatred is encouraged. Facebook defines hate speech as a direct attack on For people, in the case of Twitter, hateful behavior includes words that dehumanize others based on religion or caste. Increasing hate speech on the Internet is a major cultural issue. Threat because it has already led to minority crimes. See, for example, there is to deal with this issue Detect growing interest in detecting malicious expressions And adjust this phenomenon. Many models claim to have the latest performance on some datasets but are not generalized. Models can categorize comments related to specific IDs that are frequently attacked (gay, black, Muslim, etc.). Classified as toxic without comment with toxic intent. Big Faced with a particular trigger vocabulary leads to biased predictions that can distinguish a particular group It has already been the target of such abuse. Another problem with that The current method is a lack of explanation for the decisions made. A model for detecting constantly growing malicious expressions It gets more and more complicated and difficult for them. This requires a change in perspective from a performance-based model to an interpretable model. In our work approach the explainability of the model by learning the goals Together with the reasons for classification and human decision, and also for their common improvement.

Background

Nowadays, abusive or angry comments are extremely normal in social network media. Hence, in earlier years, a portion of the scientists have applied a regulated ML-based text order way to deal with characterized disdain discourse content. Various specialists have utilized a distinctive assortment of component portrayal strategies specifically, word reference-based, Bag-of-words based, N-grams-based, TFIDF-based what's more Deep-Learning-based. Peter Burnap et al. utilized a word reference-based way to deal with distinguish digital disdain on Twitter. In this examination, they utilized an N-gram include designing a strategy to produce the numeric vectors from the predefined word reference of contemptuous words. The creators took care of the produced numeric vector to ML classifier to be specific, SVM, and got a limit of 67% F-score.ML-based classifier to group disdain discourse in web gatherings and online journals. The creators utilized a word reference-based way to deal with producing an expert component vector. The highlights depended on opinion articulations utilizing semantic and subjectivity highlights with a direction to loathe discourse. A short time later, the creators took care of the expert including vector to a standard-based classifier. In the test settings, the creators assessed their classifier by utilizing an accuracy execution metric and acquired 73% accuracy

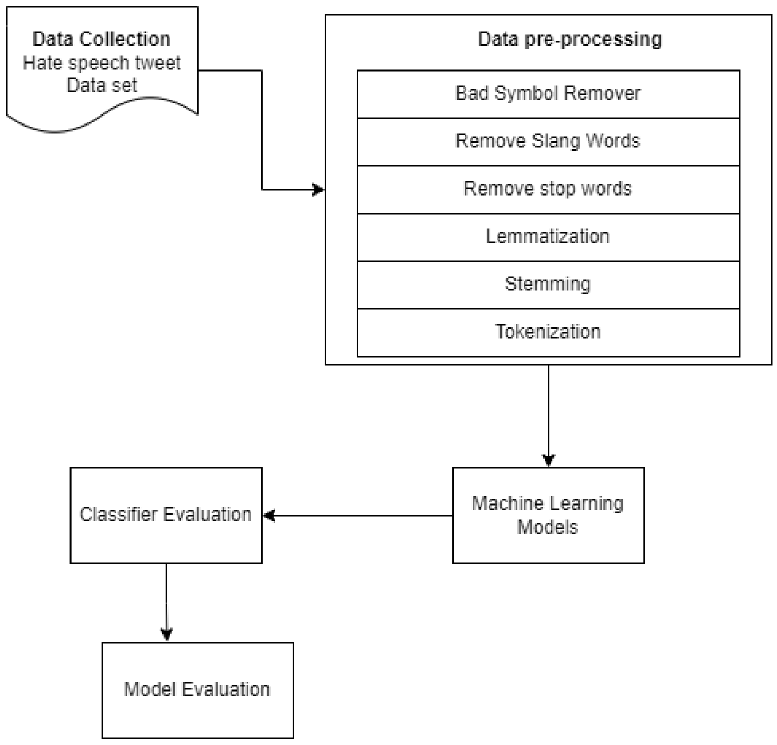
Reference:https://thesai.org/Downloads/Volume11No8/Paper_61-Automatic_Hate_Speech_Detection.pdf

Model:

This section explains the proposed system which we have employed to classify tweets into three different classes namely, “hate speech, offensive but not hate speech, and neither hate speech nor offensive speech”. Fig. 1 shows the complete research methodology. As shown in this figure, the research methodology is contained six key steps namely, data collection, data preprocessing, feature engineering, data splitting, classification model construction, and classification

model evaluation. Each of the steps is discussed in detail in the subsequent sections

Architecture Diagram:



Data Collection:

In this project dataset hate speech tweets is taken which is available publicly.

In labelled.csv the tweets are labelled into three different classes namely "hate speech, offensive speech, and neither hate speech nor offensive speech whereas the test1.csv dataset is labelled into two distinct classes i.e "hate speech and offensive speech".

Data Pre-processing:

Data pre-processing transforms raw data into well-formed datasets so that data mining analysis can be applied.

Pre-processing includes both data validation and data assignment.

1. Removing Bad Symbol: Symbols need to be removed because while tokenizing symbols will not be considered.

2. Removing Slang words: Slang words need to be removed because we don't know while classifying, we don't know that data to be hate speech or not.

3. Removing stop words: While performing tokenization, text summarization, text classification, or other equivalent operations, you need to remove the stopword. Since there are no stopwords, you can understand the context of the text material provided. They need to be removed to reduce their weight.

4.Lemmatization: It is a semantic term. that refers to the process of collecting words having the same root, or lemma, but various ending or meaning derivatives, so that they may be studied as a single item. To bring out the dictionary form of the term, ending suffixes and prefixes are removed.

The suffixes "s," "'s," and "s'" are removed from the words "cats," "cat's," and "cats'" to reveal the base word "cat." Lemmatization is crucial in the subject of artificial intelligence (AI) known as "natural language processing (NLP)" or "natural language understanding," as it is used to teach machines to speak and converse.

5.Stemming: Stemming is the process by which a word is reduced to its root, which is added to the root, called a suffix and prefix, or lemma. Stemming is important in Natural Language Understanding (NLU) and Natural Language Processing (NLP). ... It can open up new research opportunities when a new word is found

6.Tokenization: Tokenization is the process of splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph.

Machine Learning Models:

1. **TF-IDF (term frequency-inverse document frequency)** is mathematical measure of how relevant a word is to a document in a document collection. It is done by multiplying two metrics: the number of occurrence of a word in the document and the inverse

document frequency across set of documents.

TF-IDF score for the word t and document d is calculated as

$$tf\,idf\,(t, d, D) = tf\,(t, d)\,.\,idf\,(t, D)$$

Where:

$$tf\,(t, d) = \log\,(1 + freq\,(t, d))$$

$$idf\,(t, D) = \log\left(\frac{N}{count\,(d \in D : t \in d)}\right)$$

2. Decision Tree Classifier: Decision tree is a supervised machine learning algorithm which is used for classification and regression. It is a flowchart structure where each internal node represents the test and the branch represents the result of the test. Entropy is used to determine how splitting of data is chosen by the decision tree. Information gain is used to determine the order of attributes in a node. Main node is referred to as the parent node and sub-nodes are referred to as the child node. Low Entropy gives high information gain i.e entropy is inversely proportional to information gain. Gini impurity tells us about the probability of misclassifying an observation.

3. Random forest:

Random forest is a combination of decision tree with reasonable depth which takes decision tree as a base learners which has row sampling with replacement and column/feature sampling. Random forest is an ensemble machine learning algorithm known as Bagging.

4.Gradient Boosting

It is an implementation of the extension decision tree which is highly efficient, portable and flexible. It implements machine learning algorithm under gradient boosting framework. It solves data science problems in a quick and accurate way by using parallel tree boosting

Classifier Evaluation:

True Positives (TP) - These are correctly predicted positive values where both actual class and predicted values are yes.

True Negatives (TN) - These are the correctly predicted negative values where both actual class and predicted values are no.

False Positives (FP) – In FP the actual class is no and predicted class is yes.

False Negatives (FN) – In FN actual class is yes but predicted class in no.

Accuracy: Accuracy is the intuitive performance measure and it is defined as ratio of correctly predicted observation to the total observations.

Accuracy = TP+TN/TP+FP+FN+TN

Precision: Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations.

Precision = TP/TP+FP

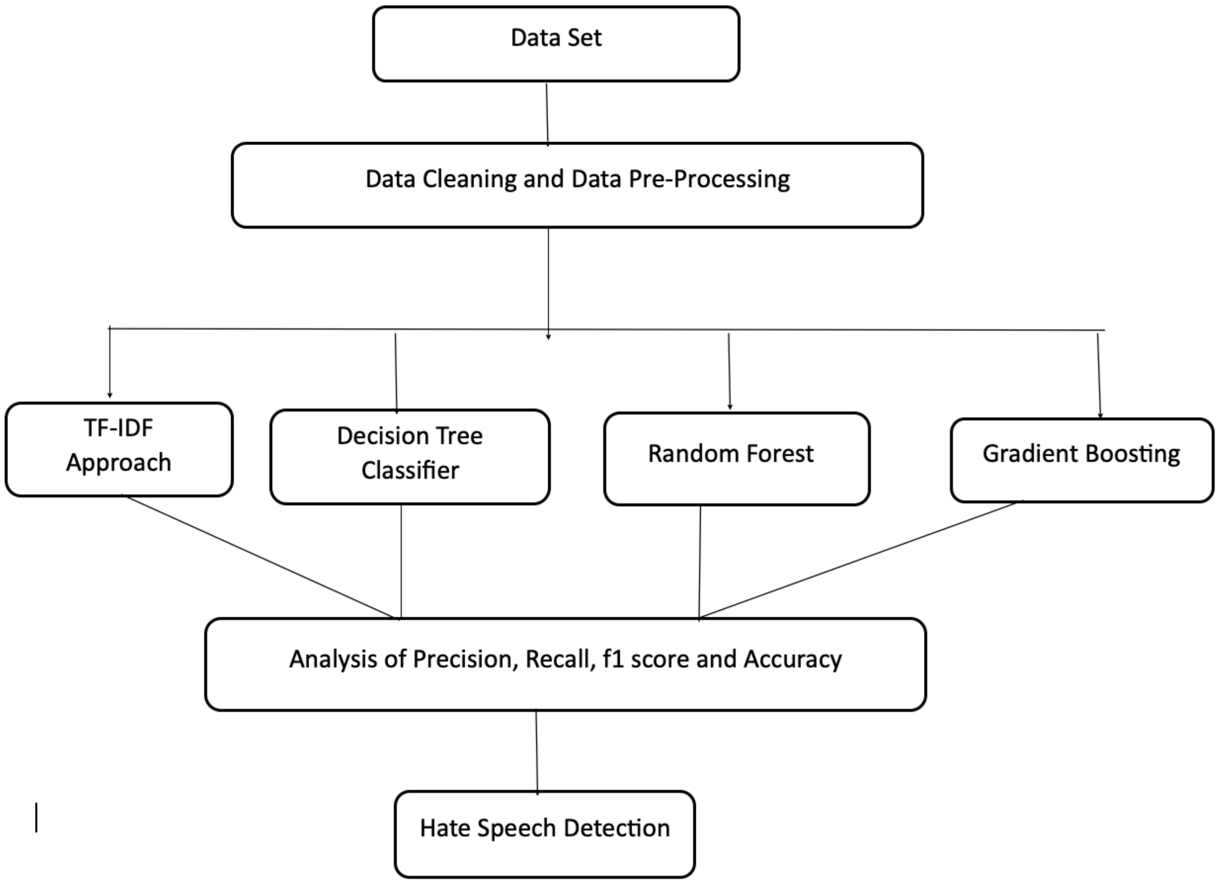
Recall (Sensitivity) - Recall is defined as the ratio of correctly predicted positive observations to the all observations in actual class

Recall = TP/TP+FN

F1 score: F1 Score is the average of Precision and Recall.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Work Flow Diagram:

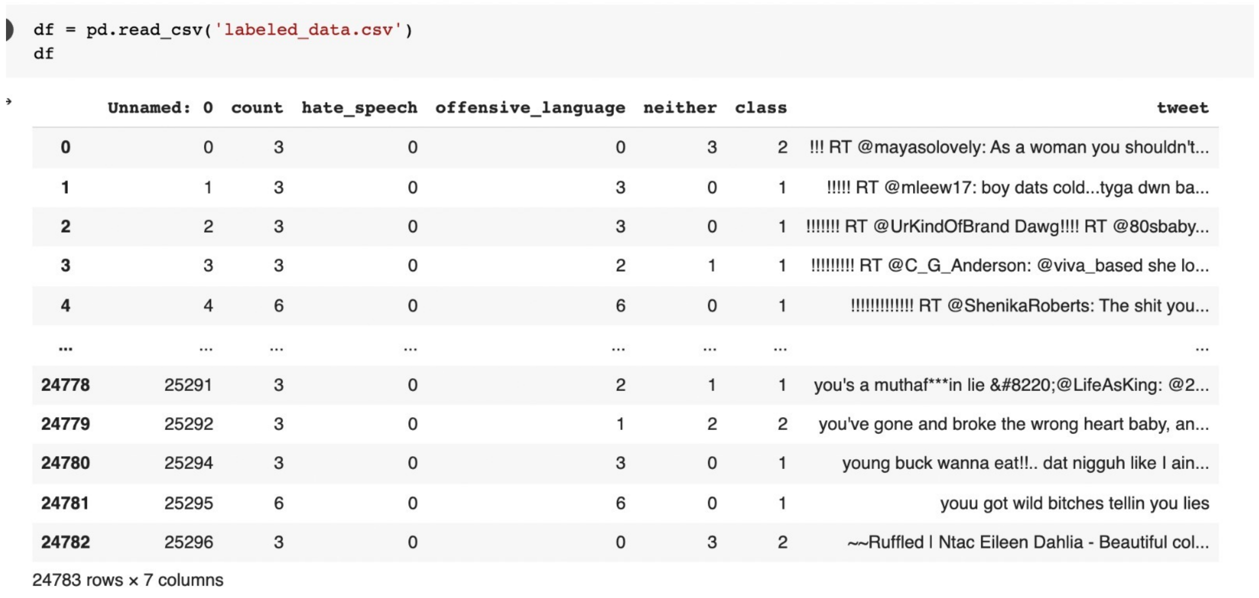


Explanation:

This project provides a solution for detecting Hate speech on social media platforms such as Twitter, Facebook, etc. This consists of two data sets i.e labeled.csv and test1.csv. Data set test1.csv includes 31962 rows and 3 columns and the labeled.csv data set includes 24783 rows and 7 columns. These data sets are sent to data cleaning and data pre-processing in which bad symbols, slang words, stop words are removed, and also stemming, lemmatization, tokenization is done. After this process, various techniques like the TF-IDF approach and algorithms like Decision tree classifier, Random Forest, and Gradient Boosting are applied. Since the data sets are labeled, there is no need of splitting the data into training sets and test sets. The data proficient efficiency-driven with the model derived 88.93% in a random forest, 88.04% Decision Tree, 88.45% in Gradient Boosting for the data set 1, and now for dataset-2 accuracy was 96.15% Random Forest, 94.35% for the decision tree, 94.82%for Gradient Boosting. Among all the algorithms gradient boosting has the highest accuracy. Random forest is not chosen as the highest accuracy algorithm because while splitting subset of features are chosen randomly and hence accuracy may vary.

Dataset:

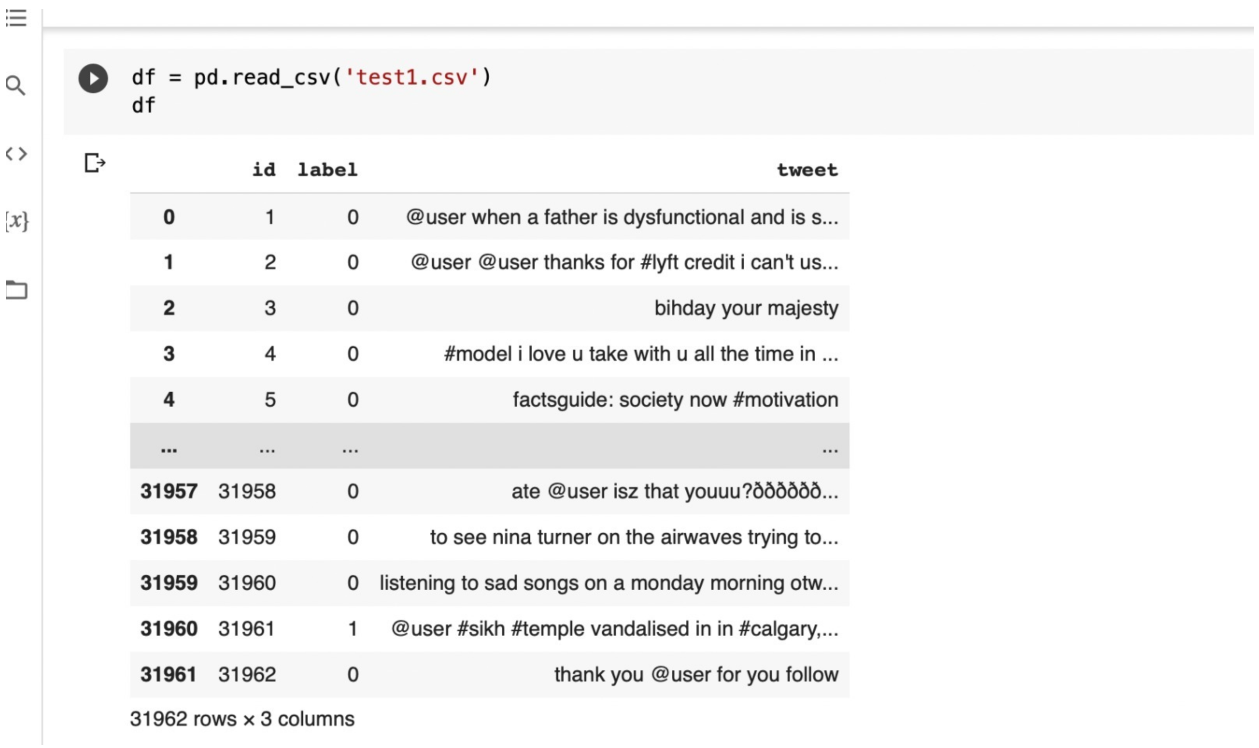
Description: This Project consists of two data sets namely labeled_data. csv and test1.csv. labeled_data.csv consists of 24783 rows and 7 columns. Data set test1.csv consists of 31962 rows and 3 columns. In labeled.csv the tweets are labelled into three different classes namely "hate speech, offensive speech, and neither hate speech nor offensive speech whereas the test1.csv dataset is labelled into two distinct classes i.e "hate speech and offensive speech".



```
df = pd.read_csv('labeled_data.csv')
df
```

	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...
...	...	...	...	...	...	...	...
24778	25291	3	0	2	1	1	you's a muthaf***in lie &#8220;@LifeAsKing: @2...
24779	25292	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
24780	25294	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
24781	25295	6	0	6	0	1	youu got wild bitches tellin you lies
24782	25296	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

24783 rows × 7 columns



```
df = pd.read_csv('test1.csv')
df
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
...	...	...	...
31957	31958	0	ate @user isz that youuu?ðððððð...
31958	31959	0	to see nina turner on the airwaves trying to...
31959	31960	0	listening to sad songs on a monday morning otw...
31960	31961	1	@user #sikh #temple vandalised in in #calgary,...
31961	31962	0	thank you @user for you follow

31962 rows × 3 columns

Analysis of Data:

Data pre-processing:

Before feeding the tweets into the classification models, we performed numerous pre-processing processes on the datasets. When dealing with data that is noisy and colloquial, such as Twitter, pre-processing is critical. Given the uncertainty and wide range of terminology used on Twitter, this is especially critical for tweets. We ran our data through numerous preparation procedures before putting it into our Experimental Models.
Among the steps are:

- Data cleansing
- Stop Words Removal
- Lemmatization
- Stemming

- Tokenization
- Distinguishing Hate Words from Common Words.

Data Cleaning:: Data cleaning is a vital phase in which punctuation, extra whitespace, and slang terms are eliminated. A dataset is not usually created for your specific use-case, but rather for various purposes in general. And that is why it is important to clean the data at first before feeding it to the model.

1. **Removing " @" from the dataset:**
   Special characters should be eliminated since when we tokenize the text, punctuation will not be given additional weightage.

2. **Removing Numbers from the dataset:**
   Numbers should be eliminated as we don't need them for classifying the data to be hate speech or not.

3. **Removing Greek Characters:**
   Greeks words should be eliminated since We don't need them to determine whether or not the data is hate speech.

4. Removing' hmm' and it's variants

We don't need them in our dataset to determine if the data is hate speech or not.

Stop Words Removal:
If you are performing tokenization, text summarization, text classification, or any other equivalent operation, stopwords should be deleted. You can grasp the context of the textual material supplied to you since there are no stop-words. They must be removed to lower their weight.

Removing Slang Words:
Words like
'luv':'love','wud':'would','lyk':'like','wateva':'whatever','ttyl':'talk to you later',

'kul':'cool','fyn':'fine','omg':'oh my god!','fam':'family','bruh':'brother',
'Cud':'could','fud':'food
Are to be changed.

Lemmatization:
Lemmatization is a Text Normalization (or Word Normalization) approach used in Natural Language Processing. Lemmatization assists us in obtaining the root forms (often referred to as synonyms in search context) of inflected (derived) words.

Stemming:
Stemming is the process of reducing word inflection to its basic forms, such as mapping a set of words to the same stem even if the stem itself is not a legitimate word in the Language.

Distinguishing Hate Words from Common Words

With the datasets, we classify nasty and regular tweets, as well as their neighborhood, concerning complicated content. Our findings are counterintuitive. To begin, hostile users utilize phrases associated with hatred, rage, shame, terrorism, violence, and misery. We've utilized a word cloud to show the same thing.

Implementation

ML Models/Pseudocode

The Machine Learning models used are

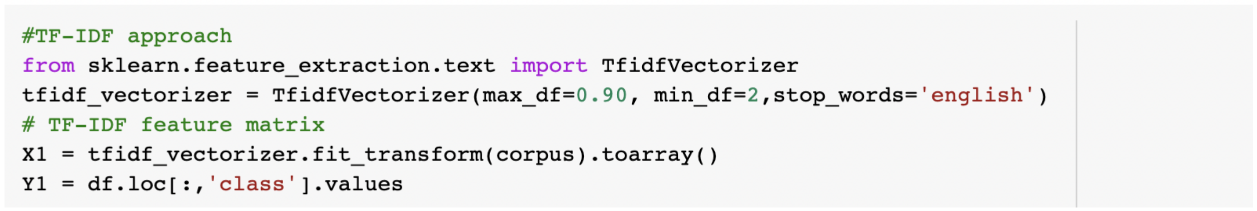
TF-IDF vectorization Approach:

Tf is an abbreviation for term frequency, which is the number of times a word appears in each text. In the last chapter,

we used CountVectorizer to do this.

Idf is an abbreviation for inverse document frequency, which is an inverse count of the number of documents in which a

term appears. Idf assesses the importance of a word in the context of the entire corpus.



```
#TF-IDF approach
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df=2,stop_words='english')
# TF-IDF feature matrix
X1 = tfidf_vectorizer.fit_transform(corpus).toarray()
Y1 = df.loc[:,'class'].values
```

Random Forest using Pipelines

Random Forest is a versatile and simple ensemble learning approach. Because of its simplicity and ability to do both

classification and regression tasks, it is one of the most often used algorithms. It is a meta estimator that fits many

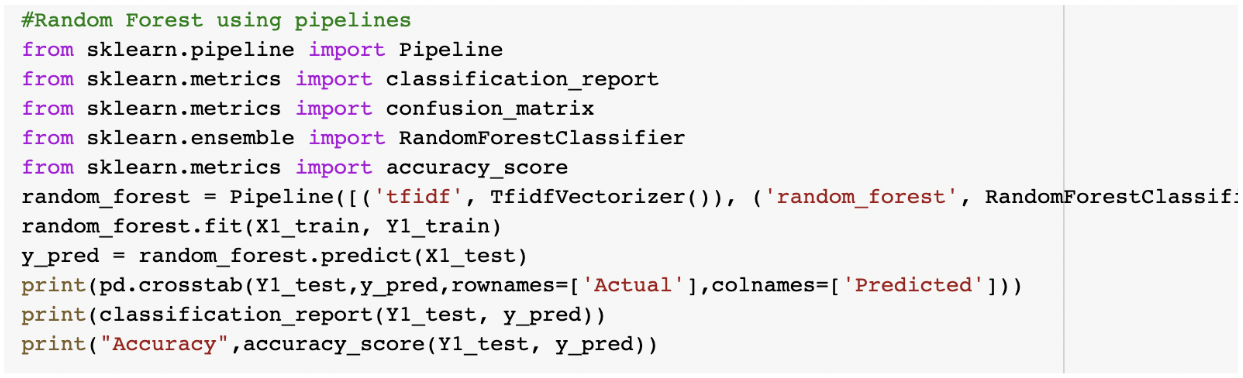
decision tree classifiers to different sub-samples of the dataset. It combats large variance by increasing the

model's unpredictability while developing the trees.

When splitting a node, it looks for the best feature from a random group of characteristics rather than the most

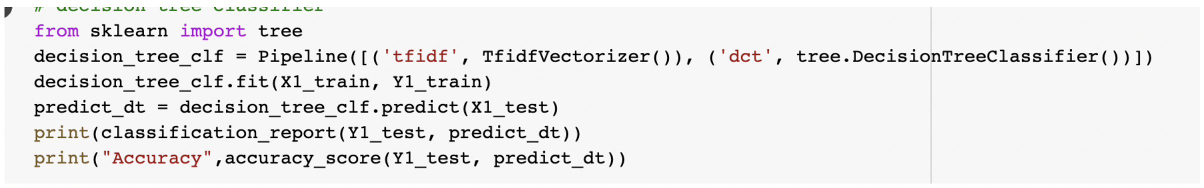
essential feature. As a result, there is a wide range of variability, which leads to a better model in general.

For dataset :labeled_data.csv



```
#Random Forest using pipelines
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
random_forest = Pipeline([('tfidf', TfidfVectorizer()), ('random_forest', RandomForestClassif
random_forest.fit(X1_train, Y1_train)
y_pred = random_forest.predict(X1_test)
print(pd.crosstab(Y1_test,y_pred,rownames=['Actual'],colnames=['Predicted']))
print(classification_report(Y1_test, y_pred))
print("Accuracy",accuracy_score(Y1_test, y_pred))
```

For dataset test1.csv



```
# decision tree classifier
from sklearn import tree
decision_tree_clf = Pipeline([('tfidf', TfidfVectorizer()), ('dct', tree.DecisionTreeClassifier())])
decision_tree_clf.fit(X1_train, Y1_train)
predict_dt = decision_tree_clf.predict(X1_test)
print(classification_report(Y1_test, predict_dt))
print("Accuracy",accuracy_score(Y1_test, predict_dt))
```

Decision Tree Classifier

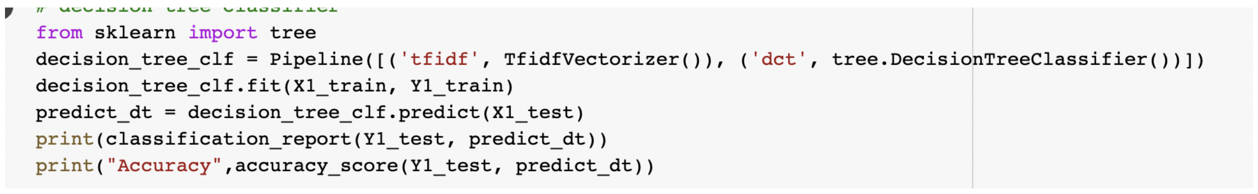
The decision tree classifiers used a tree structure to arrange a set of test questions and conditions.

The root and internal nodes of the decision tree include attribute test criteria that separate records with distinct qualities.

A class label is assigned to each terminal node. No, or yes.

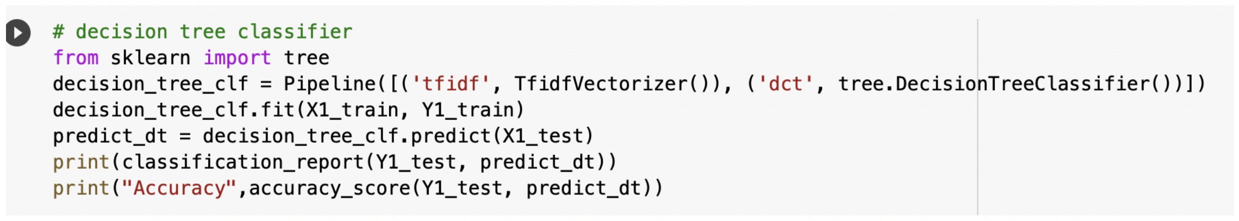
decision tree classifier

For dataset:labeled_data.csv



```python
from sklearn import tree
decision_tree_clf = Pipeline([('tfidf', TfidfVectorizer()), ('dct', tree.DecisionTreeClassifier())])
decision_tree_clf.fit(X1_train, Y1_train)
predict_dt = decision_tree_clf.predict(X1_test)
print(classification_report(Y1_test, predict_dt))
print("Accuracy",accuracy_score(Y1_test, predict_dt))
```

For dataset:Test1.csv



```python
# decision tree classifier
from sklearn import tree
decision_tree_clf = Pipeline([('tfidf', TfidfVectorizer()), ('dct', tree.DecisionTreeClassifier())])
decision_tree_clf.fit(X1_train, Y1_train)
predict_dt = decision_tree_clf.predict(X1_test)
print(classification_report(Y1_test, predict_dt))
print("Accuracy",accuracy_score(Y1_test, predict_dt))
```

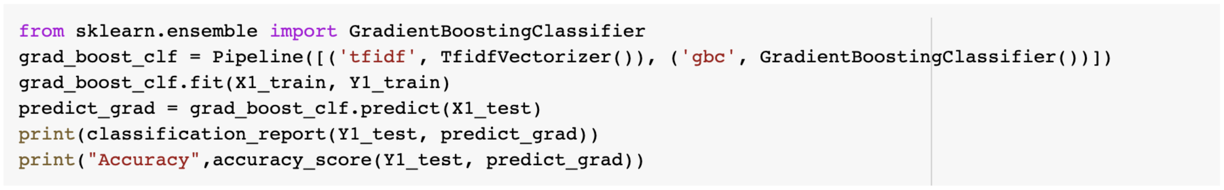
Gradient boosting Classifier.

The gradient boosting technique is one of the most powerful algorithms in the field of machine learning

. As we all know, errors in machine learning algorithms are broadly categorized into two types:

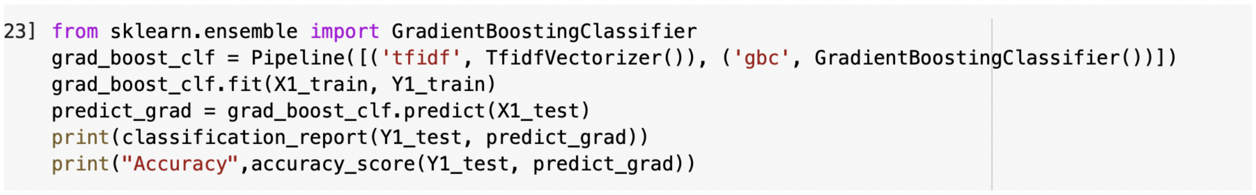
bias errors and variance errors. As one of the boosting strategies, gradient boosting is used to reduce the model's bias error

For dataset: labeled_data.csv



```
from sklearn.ensemble import GradientBoostingClassifier
grad_boost_clf = Pipeline([('tfidf', TfidfVectorizer()), ('gbc', GradientBoostingClassifier())])
grad_boost_clf.fit(X1_train, Y1_train)
predict_grad = grad_boost_clf.predict(X1_test)
print(classification_report(Y1_test, predict_grad))
print("Accuracy",accuracy_score(Y1_test, predict_grad))
```

For dataset:test1.csv



```
23] from sklearn.ensemble import GradientBoostingClassifier
    grad_boost_clf = Pipeline([('tfidf', TfidfVectorizer()), ('gbc', GradientBoostingClassifier())])
    grad_boost_clf.fit(X1_train, Y1_train)
    predict_grad = grad_boost_clf.predict(X1_test)
    print(classification_report(Y1_test, predict_grad))
    print("Accuracy",accuracy_score(Y1_test, predict_grad))
```

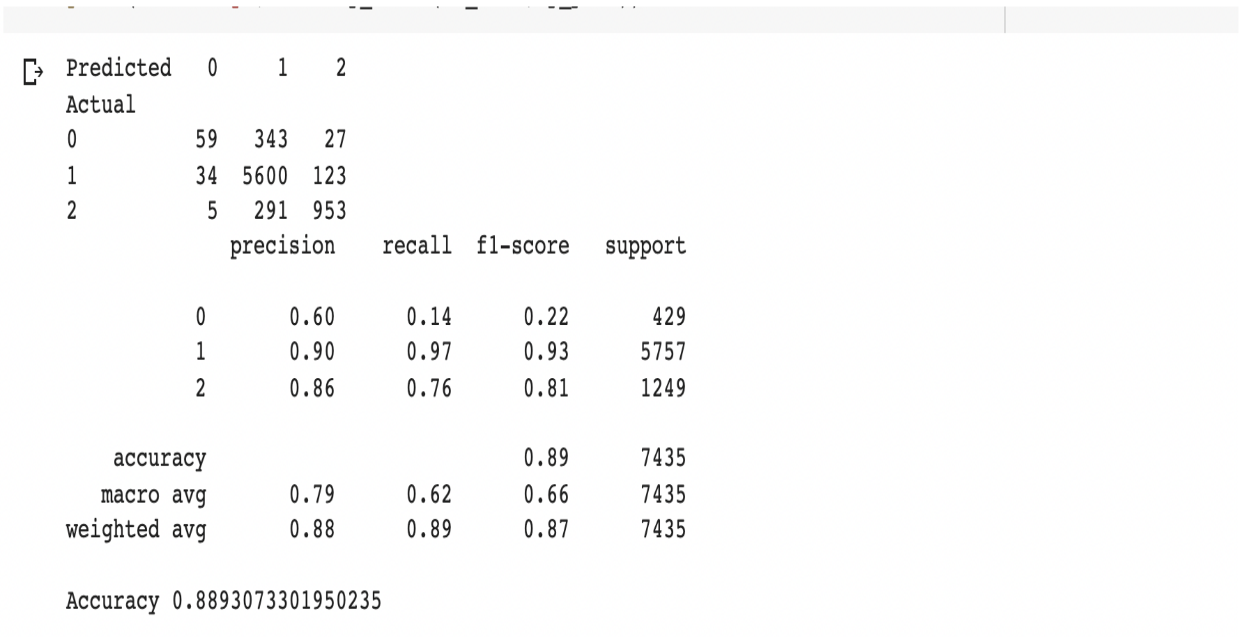
Explanation:

We've used two different datasets for Hate speech detection from Kaggle. The data set test1.csv has 31962 rows, three columns, and is labeled. There are 24783 rows and 7 columns in this CSV data collection. These data sets are subjected to data cleaning and pre-processing, which includes the removal of bad symbols, slang terms, and stop words, as well as stemming, lemmatization, and tokenization. Following this, algorithms such as Decision Tree Classifier, Random Forest, and Gradient Boosting are used, as well as strategies such as the TF-IDF approach. There is no need to divide the data into training and test sets because the data sets are labeled. For dataset 1, the data proficient efficiency-driven with the model derived 88.93 percent in a random forest, 88.04 percent Decision Tree, 88.45 percent in Gradient Boosting, and now for dataset 2, the accuracy was 96.15 percent Random forest, 94.35 percent Decision Tree, and 94.82 percent Gradient Boosting.

Results:

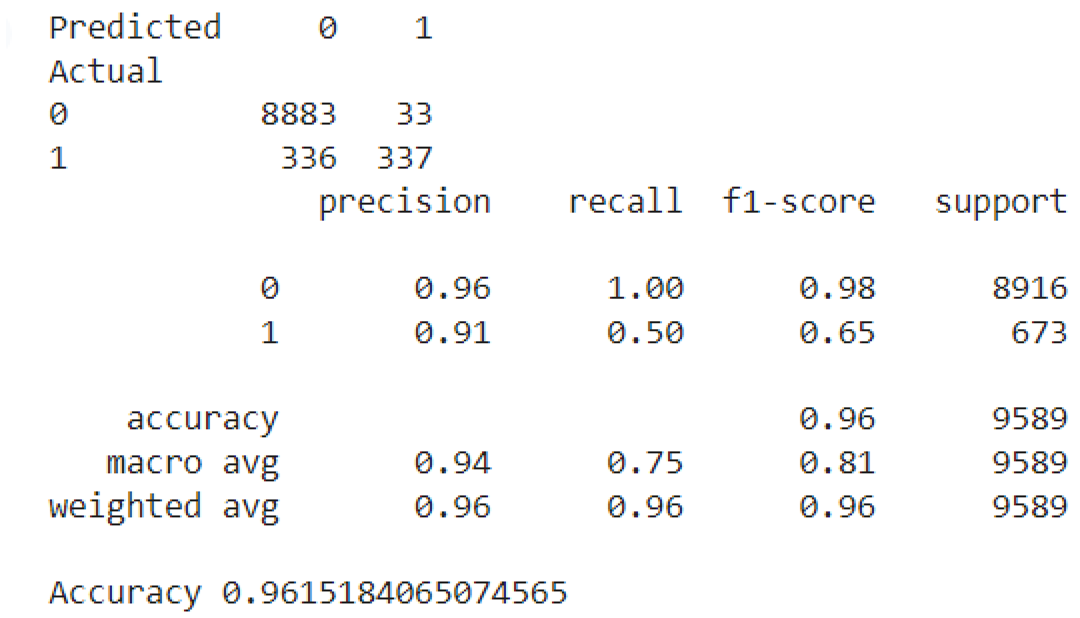
Random Forest

For dataset: labeled_data.csv



```
Predicted   0     1     2
Actual
0          59   343    27
1          34  5600   123
2           5   291   953
              precision    recall  f1-score   support

           0       0.60      0.14      0.22       429
           1       0.90      0.97      0.93      5757
           2       0.86      0.76      0.81      1249

    accuracy                           0.89      7435
   macro avg       0.79      0.62      0.66      7435
weighted avg       0.88      0.89      0.87      7435

Accuracy 0.8893073301950235
```

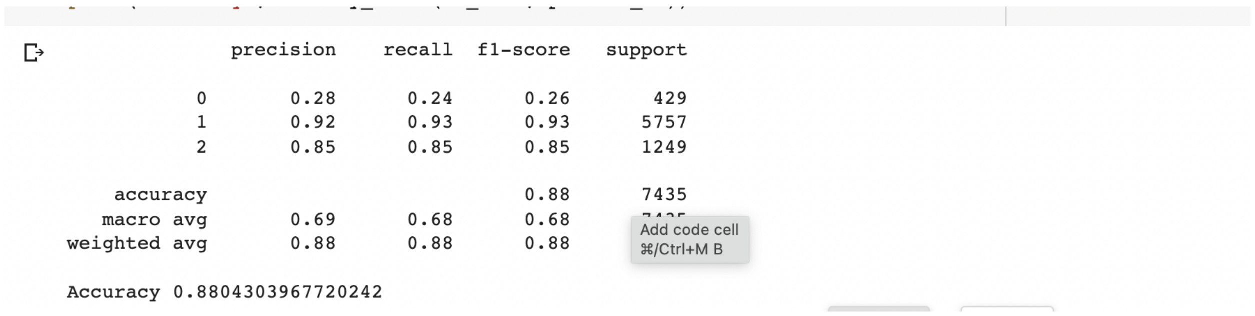
For dataset: test1.csv



```
Predicted     0    1
Actual
0          8883   33
1           336  337
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      8916
           1       0.91      0.50      0.65       673

    accuracy                           0.96      9589
   macro avg       0.94      0.75      0.81      9589
weighted avg       0.96      0.96      0.96      9589

Accuracy 0.9615184065074565
```

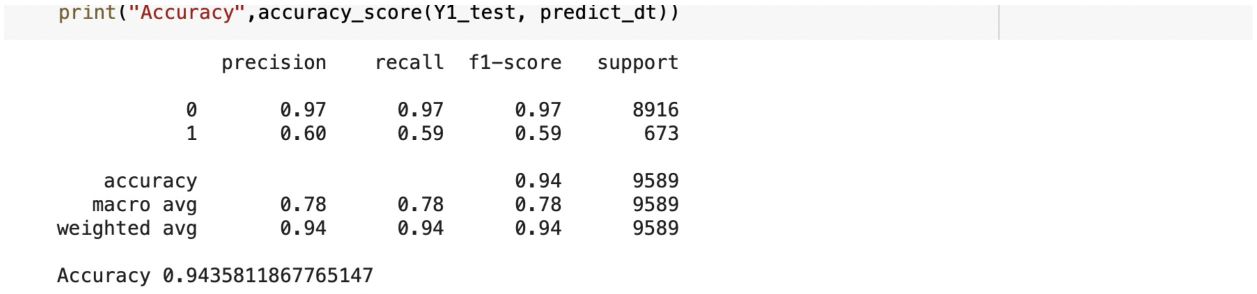
Decision Tree Classifier

For dataset: labeled_data.csv



```
              precision    recall  f1-score   support

           0       0.28      0.24      0.26       429
           1       0.92      0.93      0.93      5757
           2       0.85      0.85      0.85      1249

    accuracy                           0.88      7435
   macro avg       0.69      0.68      0.68      7435
weighted avg       0.88      0.88      0.88

Accuracy 0.8804303967720242
```

For dataset: test1.csv

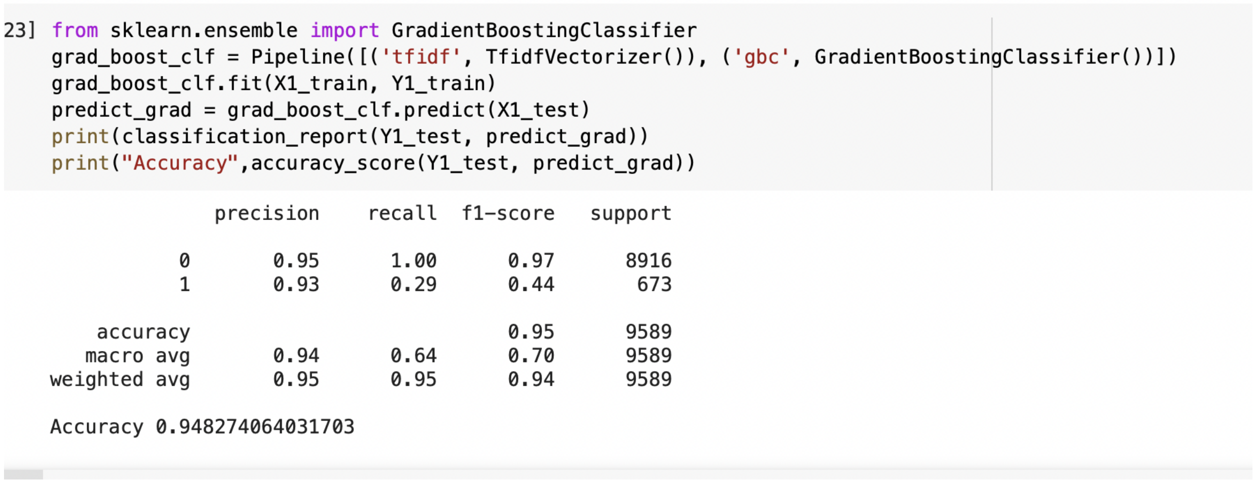


```
print("Accuracy",accuracy_score(Y1_test, predict_dt))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      8916
           1       0.60      0.59      0.59       673

    accuracy                           0.94      9589
   macro avg       0.78      0.78      0.78      9589
weighted avg       0.94      0.94      0.94      9589

Accuracy 0.9435811867765147
```

Gradient Boosting Classifier

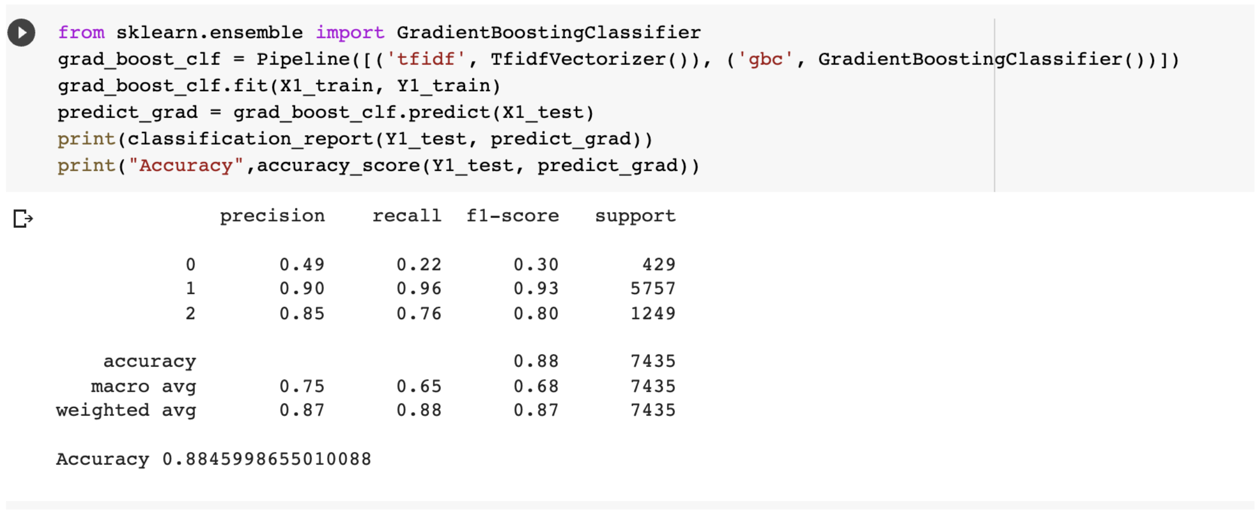
For dataset: labeled_data.csv



```
23] from sklearn.ensemble import GradientBoostingClassifier
    grad_boost_clf = Pipeline([('tfidf', TfidfVectorizer()), ('gbc', GradientBoostingClassifier())])
    grad_boost_clf.fit(X1_train, Y1_train)
    predict_grad = grad_boost_clf.predict(X1_test)
    print(classification_report(Y1_test, predict_grad))
    print("Accuracy",accuracy_score(Y1_test, predict_grad))
```

```
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      8916
           1       0.93      0.29      0.44       673

    accuracy                           0.95      9589
   macro avg       0.94      0.64      0.70      9589
weighted avg       0.95      0.95      0.94      9589

Accuracy 0.948274064031703
```

For dataset: test1.csv



```python
from sklearn.ensemble import GradientBoostingClassifier
grad_boost_clf = Pipeline([('tfidf', TfidfVectorizer()), ('gbc', GradientBoostingClassifier())])
grad_boost_clf.fit(X1_train, Y1_train)
predict_grad = grad_boost_clf.predict(X1_test)
print(classification_report(Y1_test, predict_grad))
print("Accuracy",accuracy_score(Y1_test, predict_grad))
```

```
              precision    recall  f1-score   support

           0       0.49      0.22      0.30       429
           1       0.90      0.96      0.93      5757
           2       0.85      0.76      0.80      1249

    accuracy                           0.88      7435
   macro avg       0.75      0.65      0.68      7435
weighted avg       0.87      0.88      0.87      7435

Accuracy 0.8845998655010088
```

Project Management

Implementation status report: Work Completed

Description:

We used this proposed system to categorize tweets into three categories: "hate speech," "offensive but not hate speech," and "neither hate speech nor offensive speech." The entire research approach is depicted in Figure 1. Data collecting, data preprocessing, feature engineering, data splitting, classification model development, and classification model evaluation are the six major processes in the research technique, as depicted in the architecture diagram. We've used all the steps on two different datasets,determined their accuracy and also found out that the accuracy changes with the number of tweets in the dataset i.e the dataset with more data has higher accuracy than that of with less data.

Responsibility :

Coding: Sai Anjali Potula, Seemaparvez Shaik, Vishnu Manoj Deepala

Documentation: Sai Anjali Potula, Seemaparvez Shaik

Presentation: Sai Anjali Potula, Seemaparvez Shaik.

References:

https://thesai.org/Downloads/Volume11No8/Paper_61-Automatic_Hate_Speech_Detection.pdf.

https://docs.trifacta.com/display/DP/Remove+Data

https://www.datacamp.com/community/tutorials/stemming-lemmatization-python

https://towardsdatascience.com/cleaning-preprocessing-text-data-by-building-nlp-pipeline-853148add68a

https://www.kaggle.com/usharengaraju/dynamically-generated-hate-speech-dataset

https://www.kaggle.com/dv1453/twitter-sentiment-analysis-analytics-vidya

https://openclassrooms.com/en/courses/6532301-introduction-to-natural-language-processing/7067116-apply-the-tf-idf-vectorization-approach

https://medium.com/@benfenison/gridsearching-a-random-forest-classifier-fc225609699c

Github Link:
https://github.com/Seemaparvez17/NLP-Project
Yotube Link:
https://youtu.be/HmqM61MQ4N4