

Hate Speech Detection

A PROJECT REPORT

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Under the guidance of

MR. MAHENDRA DATTA

BY

UDAY SAHA

AJIT BARMAN

ROUNAK DEY

SUMIT RAJ



SILIGURI INSTITUTE OF TECHNOLOGY

SILIG, WEST-BENGAL, INDIA

In association with



1. TITLE OF THE PROJECT :-
HATE SPEECH DETECTION:

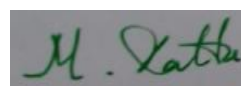
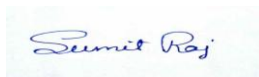
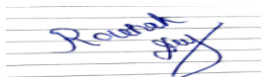
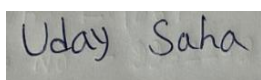
1. PROJECT MEMBERS :-

UDAY SAHA,
AJIT BARMAN,
ROUNAK DEY,
SUMIT RAJ

2. NAME OF THE GUIDE :- MAHENDRA DATTA.

3. PROJECT VERSION CONTROL: -

Version	Primary Author	Description Of Version	Date Completed
Final	UDAY SAHA, AJIT BARMAN, ROUNAK DEY, SUMIT RAJ	Project Report	<u>27-08-2021</u>



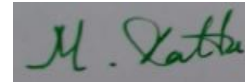
Signature of the Student

Signature of the Supervisor

Date: 27-08-2021

Date: 27-08-2021

For Office Use Only



MAHENDRA DATTA

Project proposal / Evaluator

APPROVED



**NOT
APPROVED**

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled “**HATE SPEECH DETECTION**” in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT LTD, SALT LAKE, KOLKATA, WEST BENGAL**, is an authentic work carried out under the guidance of **MR. MAHENDRA DATTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 27-08-2021

Name of the Student:

UDAY SAHA,
AJIT BARMAN,
ROUNAK DEY,
SUMIT RAJ

Signature of the student:

Uday Saha

Ajit Barman

Rounak Dey

Sumit Raj



CERTIFICATE

This is to certify that this proposal of minor project entitled “**HATE SPEECH DETECTION ANALYSIS**” is a record of bonafide work, carried out by **UDAY SAHA, AJIT BARMAN, ROUNAK DEY and SUMIT RAJ** under my guidance at **ARDENT COMPUTECH PVT LTD.** In my opinion, the report in its present form is in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT COMPUTECH PRIVATE LIMITED.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

M. Latha

MR. MAHENDRA DATTA

Project Engineer

ARDENT COMPUTECH PVT. LTD.

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to *Mr. Mahendra Datta*, Project Engineer at ARDENT COMPUTECH PRIVATE LIMITED, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent computech pvt.ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS:-

1) ABSTRACT

2) LIST OF COMMON MACHINE LEARNING ALGORITHM

3) PROBLEM STATEMENT

4) INTRODUCTION

5) DETAILS OF THE PROJECT

a) Data Collection

b) About The Data

6) SYSTEM REQUIREMENTS

a) Software Requirements

b) Hardware Requirements

7) MODULES USED IN THE PROJECT(with explanation)

8) CODES WITH OUTPUTS

9) CONCLUSION

10) UPLOADED PROJECT AND DOCUMENT IN OPENSOURCE AND GOOGLE COLAB LINK

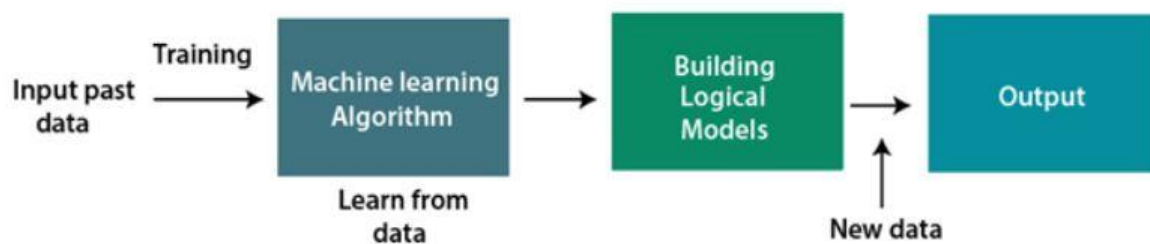
11) BIBLIOGRAPHY

ABSTRACT

Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.



Classification of Machine Learning

1. Supervised Learning.
2. Unsupervised Learning.
3. Reinforcement Learning.

1. Supervised Learning:

Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well labeled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples (data) so that the supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labeled data.

❖ Types:-

Regression:

Logistic Regression

Classification:

Naive-Bayes Classifiers

K-NN (k nearest neighbors)

Decision Trees

Support Vector Machine

❖ Advantages:-

- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.
- Supervised machine learning helps to solve various types of real-world computation problems.

❖ Disadvantages:-

- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.

2. Unsupervised Learning :

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.

- **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

❖ Types of Unsupervised Learning:-

Clustering:

Hierarchical clustering

K-means clustering

Principal Component Analysis

Singular Value Decomposition

Independent Component Analysis

Association:

Apriori

FP-Growth

❖ Advantages:-

There are some reasons why we sometimes choose unsupervised learning in place of supervised learning. Here are some of the advantages:

Labeling of data demands a lot of manual work and expenses. Unsupervised learning solves the problem by learning the data and classifying it without any labels.

The labels can be added after the data has been classified which is much easier.

It is very helpful in finding patterns in data, which are not possible to find using normal methods.

Dimensionality reduction can be easily accomplished using unsupervised learning.

❖ **Disadvantages:-**

Now, let's have a look at some cons of unsupervised learning algorithm:

The result might be less accurate as we do not have any input data to train from.

The model is learning from raw data without any prior knowledge.

It is also a time-consuming process. The learning phase of the algorithm might take a lot of time, as it analyses and calculates all possibilities.

3. Reinforcement learning :

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

❖ **Types of Reinforcement:**

There are two types of Reinforcement:

- **Positive :-**

Positive Reinforcement is defined as when an event, occurs due to a particular behavior, increases the strength and the frequency of the behavior. In other words, it has a positive effect on behavior.

Advantages of reinforcement learning are:

Maximizes Performance.

Sustain Change for a long period of time.

Disadvantages of reinforcement learning:

Too much Reinforcement can lead to overload of states which can diminish the results.

- **Negative:** –

Negative Reinforcement is defined as strengthening of a behavior because a negative condition is stopped or avoided.

Advantages of reinforcement learning:

Increases Behavior.

Provide defiance to minimum standard of performance.

Disadvantages of reinforcement learning:

It only provides enough to meet up the minimum behavior.

LIST OF COMMON MACHINE LEARNING ALGORITHMS:

Here is the list of commonly used machine learning algorithms. These algorithms can be applied to almost any data problem:

1. Linear Regression
2. Logistic Regression
3. Decision Tree
4. SVM
5. Naive Bayes
6. KNN (K-Nearest Neighbors)
7. K-Means
8. Random Forest
9. Dimensionality Reduction Algorithms.

There are also Gradient Boosting Algorithms:

1.GBM

2.XGBoost

3.Light GBM

4.CatBoost

1. Linear Regression

It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation $Y = a * X + b$.

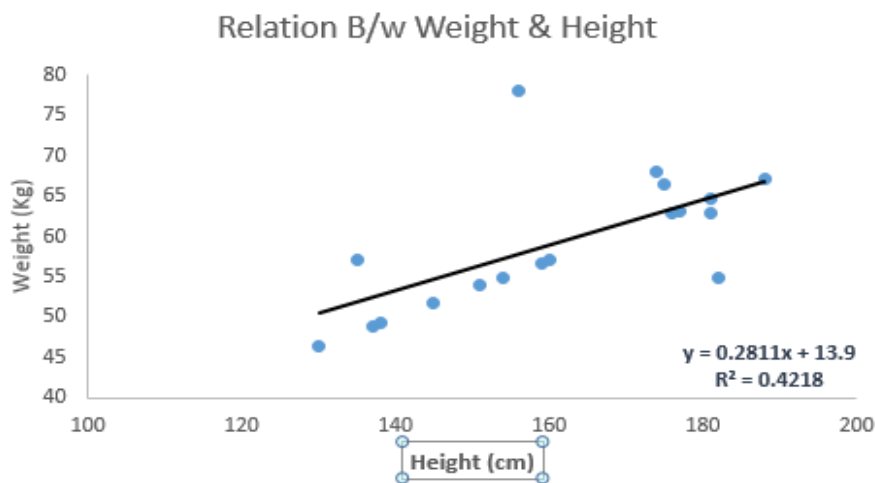
The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He / she would likely look (visually analyze) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above.

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

These coefficients a and b are derived based on minimizing the sum of squared difference of distance between data points and regression line.

Look at the below example. Here we have identified the best fit line having linear equation $y = 0.2811x + 13.9$. Now using this equation, we can find the weight, knowing the height of a person.



Linear Regression is mainly of two types: Simple Linear Regression and Multiple Linear Regression. Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression(as the name suggests) is characterized by multiple (more than 1) independent variables. While finding the best fit line, you can fit a polynomial or curvilinear regression. And these are known as polynomial or curvilinear regression.

2. Logistic Regression

It is a classification not a regression algorithm. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variables. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).

odds= $p / (1-p)$ = probability of event occurrence / probability of not event occurrences

$$\ln(\text{odds}) = \ln(p/(1-p))$$

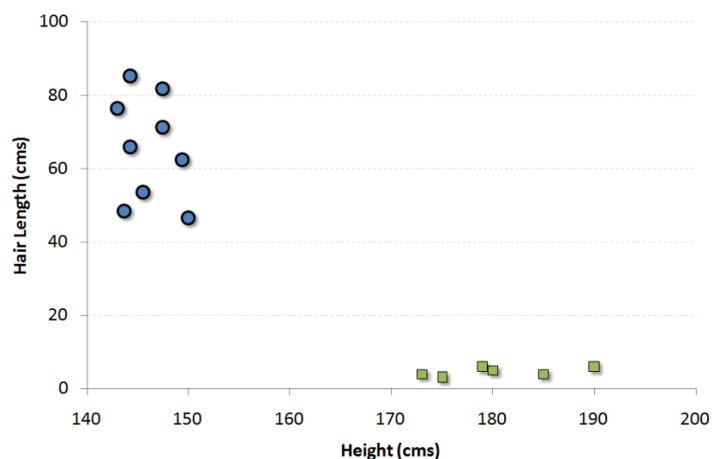
$$\text{logit}(p) = \ln(p/(1-p)) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_kX_k$$

Above, p is the probability of presence of the characteristic of interest. It chooses parameters that maximize the likelihood of observing the sample values rather than that minimize the sum of squared errors (like in ordinary regression).

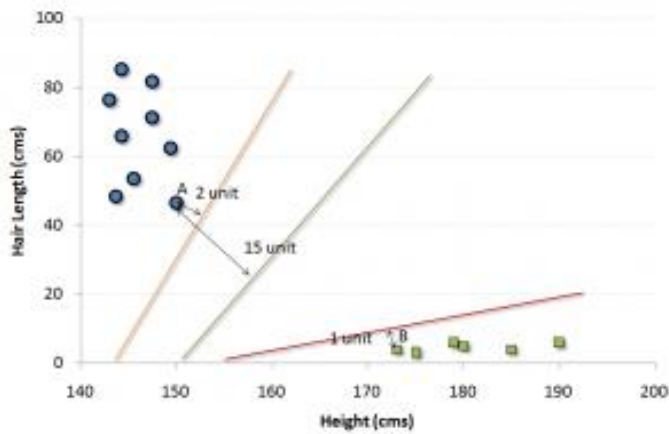
3. SVM (Support Vector Machine)

It is a classification method. In this algorithm, we plot each data item as a point in n -dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two-dimensional space where each point has two co-ordinates (these co-ordinates are known as **Support Vectors**)



Now, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.

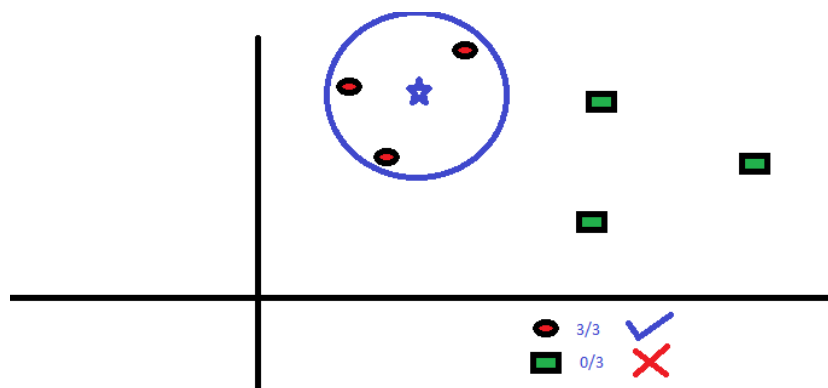


In the example shown above, the line which splits the data into two differently classified groups is the *black* line, since the two closest points are the farthest apart from the line. This line is our classifier. Then, depending on where the testing data lands on either side of the line, that's what class we can classify the new data as.

4. kNN (k- Nearest Neighbors)

It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.

These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modelling.



KNN can easily be mapped to our real lives. If you want to learn about a person, of whom you have no information, you might like to find out about his close

friends and the circles he moves in and gain access to his/her information!

Things to consider before selecting kNN:

- KNN is computationally expensive
- Variables should be normalized else higher range variables can bias it
- Works on pre-processing stage more before going for kNN like an outlier, noise removal

5. Random Forest

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is planted & grown as follows:

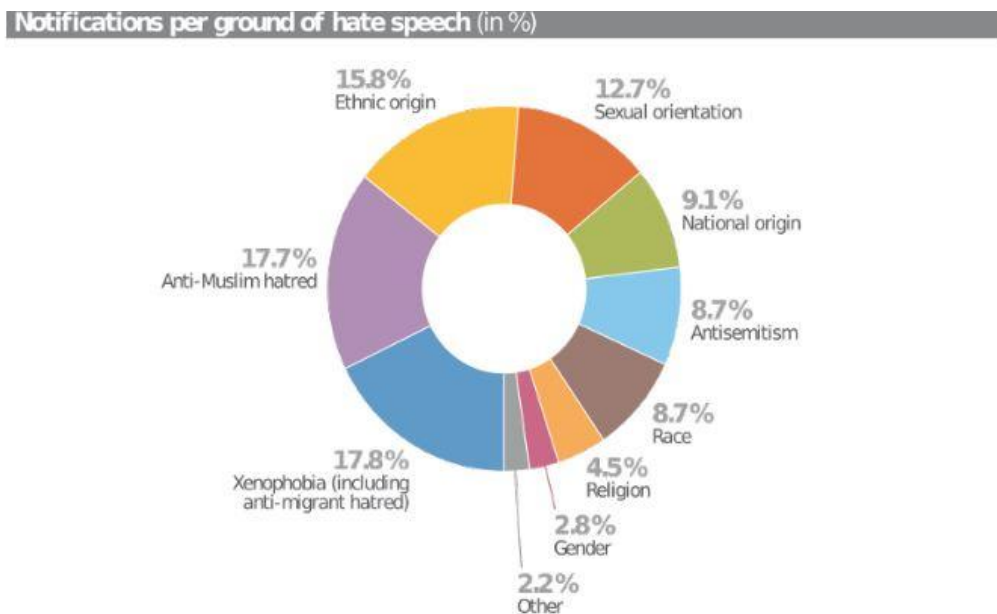
1. If the number of cases in the training set is N , then sample of N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.
2. If there are M input variables, a number $m \ll M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

PROBLEM STATEMENT

HATE SPEECH DETECTION

INTRODUCTION

What is Hate Speech?



Although the expression has become very common, there is no set definition of hate speech. In fact, the quest for a shared definition clashes with juridical, political-philosophical, and cultural debates over the boundaries of freedom of expression. How can we define – and, therefore, contrast – hate speech without limiting a fundamental freedom? Such a dilemma predates the Internet, but has been strongly revived by the digital revolution.

The different definitions of hate speech share a common ground in the documents produced by international institutions after the Second World War.

According to a recommendation by the Council of Europe in 1997, hate speech includes “all forms of expression which spread, incite, promote or justify racial hatred, xenophobia, anti-Semitism or other forms of hatred based on intolerance, including intolerance expressed by aggressive nationalism and ethnocentrism, discrimination and hostility towards minorities, migrants and

people of immigrant origin”.

Although the expression “hate speech” gained ground during the 1990s, the observation of the phenomenon and the commitment to contrast it are not new (previously, the expression “ incitement to hatred ” was preferred). For many decades, the focus was on racial hatred, antisemitism, and historical revisionism. With the new millennium, awareness on the topic has come to include religious minorities (especially Muslims, increasingly targeted by threats and discrimination) and, more recently, women, LGBT persons, the disabled, and the elderly.

In synthesis, no matter the form (written or oral, verbal or non verbal, explicit or implicit) and juridical status (possible “hate crimes”), hate speech includes any expression of violence and discrimination against other persons or groups.

As hate speech targets people on the basis of their personal characteristics and/or conditions, contrasting actions need to be appropriate for the current social, economic, political, and technological context.

Efforts to contrast hate speech today face the dilemmas and contradictions of the digital age. In a recent report, the Council of Europe addressed hate speech within the wider issue of information disorder, a global tampering of contents where hate speech and so-called fake news cross paths: in this view, disinformation stems from the encounter between mis-information (spreading news that are false, but harmless) and mal-information (spreading authentic news, with the intent to harm).

DETAILS OF THE PROJECT

1. Data Collection.

- There was 1 data set
I) HateSpeechData.csv
required for this project was taken from “*kaggle.com*”.
- Kaggle allows users to find and publish datasets to solve challenges or web-based data science projects.

2. About the Data.

- The data-set consists of 24784 Rows and 7 Columns for HateSpeechData.csv
- The description of each Columns is as follows:-

Content

- I) Title: Consist hate speech
 - II) Text: Consist of the hate speech details
 - III) Subject: Which type of social site it is (eg. facebook , twitter , etc.)
 - IV) Date: Status updating date
-

FOR ONLINE USE:

1.SOFTWARE REQUIREMENTS

- **OS: Windows or linux**
- **Web: chrome**
- **Website google Colab (<https://colab.research.google.com/>) or using Jupyter Notebook**
- **language python**

FOR OFFLINE USE:

1.HARDWARE REQUIREMENTS

- **RAM:4GB and higher**
- **processor :intel i3 and above**
- **hard disk:500GB: minimum**

2.SOFTWARE REQUIREMENTS

- **os: windows or linux**
- **python IDE:python 2.7.x and above**
- **jupyter notebook**
- **setup tools and pin to be installed for 3.6 and above**
- **language python**

Modules used in the Project:

- **NumPy** :- It is a general purpose array processing package.it provides a high performance multidimensional array object, and tools for working with these arrays .its also an efficient multidimensional container of generic data.
- **Pandas** :- It is the most popular python library that is used for data analysis.it provides highly optimized performance with back-end source code is purely written in **C** or **python** .subdivided into two parts it is :- **series and dataframe**.
- **Matplotlib** :- It is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environment across platforms .it tries to make easy things easy and hard things possible. you can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code.
- **Seaborn** :- It is a python data visualization library based on matplotlib.it provides a high level interface for drawing attractive and informative statistical graphics.it is closely integrated with pandas datastructures.
- **Tokenization** :- Tokenization is the process of turning sensitive data into nonsensitive data called "tokens" that can be used in a database or internal system without bringing it into scope. Tokenization can be used to secure sensitive data by replacing the original data with an unrelated value of the same length and format. The tokens are then sent to an organization's internal systems for use, and the original data is stored in a secure token vault.
- **Pad sequence:-Padding** As we know all the neural networks needs to have the inputs that should be in similar shape and size. When we pre-process the texts and use the texts as an inputs for our Model. Note that not all the sequences have the same length, as we can say naturally some of the sequences are long in lengths and some are short. Where we know that we need to have the inputs with the same size, now here padding comes into picture. The inputs should be in same size at that time padding is necessary.
- **Word Cloud:-** Word Cloud is a data visualization technique used for representing text data in which the size of each word

indicates its frequency or importance. ... The dataset used for generating word cloud is collected from UCI Machine Learning Repository.

- **Sequence Model:-** Sequence models are the machine learning models that input or output sequences of data. Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm used in sequence models.

ACTUAL CODES WITH OUTPUT

Importing necessary libraries

```
import numpy as np
import pandas as panda
import matplotlib.pyplot as plt
import seaborn
import nltk
import string
import warnings
from wordcloud import WordCloud
```

Explanation:-

We import the numpy, pandas, matplotlib, seaborn, Word Cloud(provides high-level interface for attractive and informative statistical graphics).

Importing models:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import accuracy_score
from sklearn.svm import LinearSVC
```

```

from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

```

Explanation:-

We import the CountVectorizer , RandomForestClassifier, LinearSVC from sklearn model and also tensorflow.keras.

Importing the hate speech dataset

```

dataset = panda.read_csv("HateSpeechData.csv")
dataset

```

Explanation:-

The csv file(dataset) is read using the “panda.read_csv(“filename or link)” command.

Understanding the Hate Speech Data

```
dataset.head()
```

Explanation:-

head(), gives us a quick look at our data set.

Output :-

Out[2]:

	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...
...

Adding text-length as a field in the dataset

```
dataset['text length'] = dataset['tweet'].apply(len)
print(dataset.head())
```

Explanation:-

- Print the Columns of the data set.

Output :-

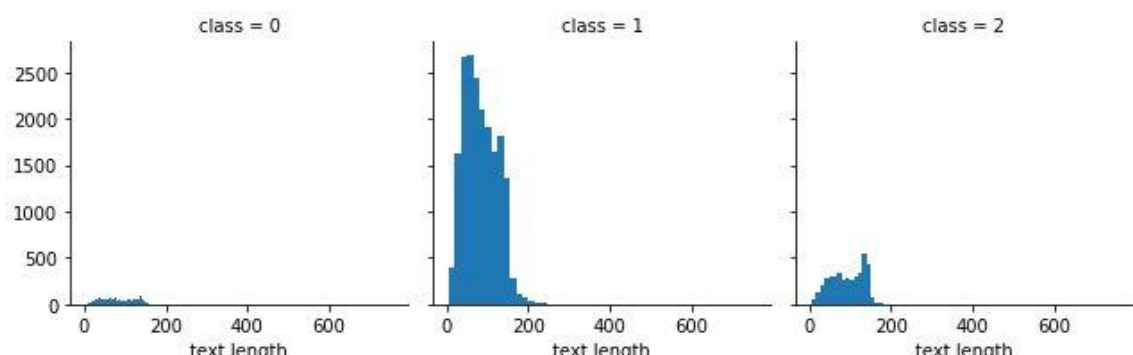
```
Unnamed: 0  count  hate_speech  offensive_language  neither  class \
```

#Basic visualization of data using histograms

```
import seaborn as sns
import matplotlib.pyplot as plt
graph = sns.FacetGrid(data=dataset, col='class')
graph.map(plt.hist, 'text length', bins=50)
```

Output :-

```
Out[4]: <seaborn.axisgrid.FacetGrid at 0x22ba2d05f70>
```



Explanation:-

To calculate hate speech

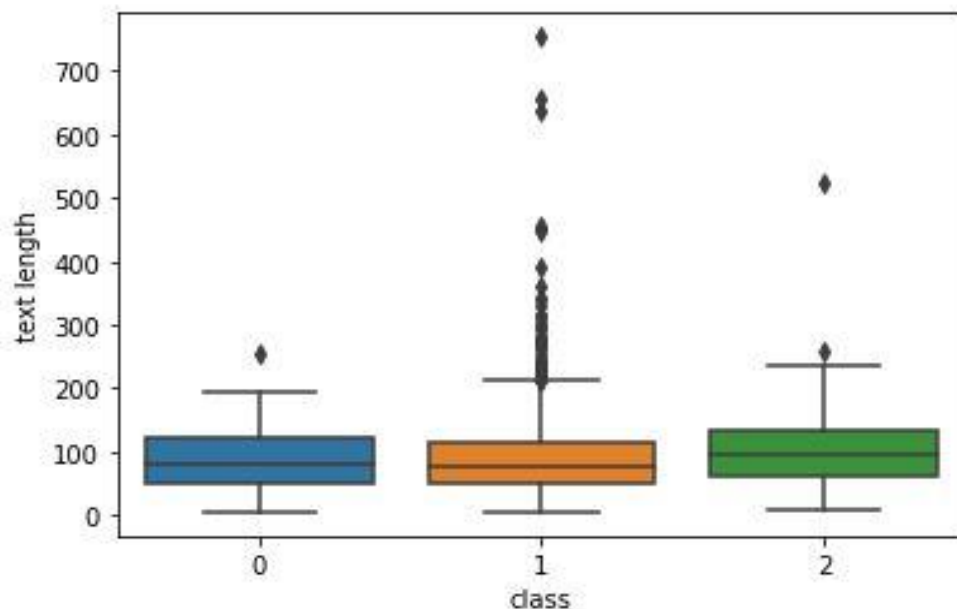
- a. Distribution of text-length almost seem to be similar across all three classes
- b. Number of tweets seem to be skewed a lot higher towards the class-1

Box-plot visvualization

```
sns.boxplot(x='class', y='text length', data=dataset)
```

Output :-

```
Out[5]: <AxesSubplot:xlabel='class', ylabel='text length'>
```



Explanation :-

Visualise The Hate speech From the box-plot, looks like the class-1 tweets have much longer text. There are also outliers present so text-length won't be a useful feature to consider.

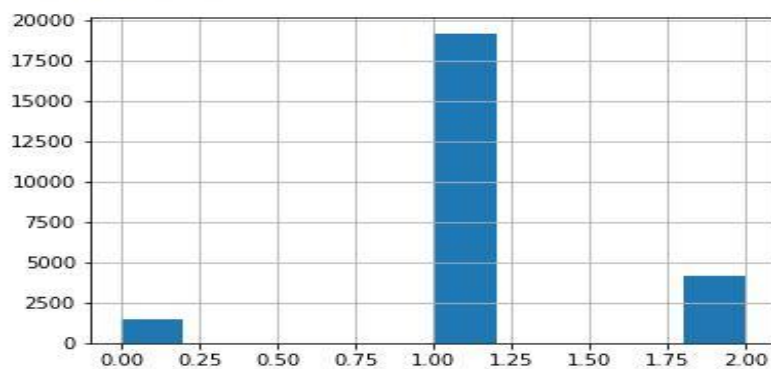
Visualise the data set in a bar graph figure.

Check The word in tweets

```
dataset['class'].hist()
```

Output :-

Out[6]: <AxesSubplot:>



Explanation:-

The above histogram shows that most of the tweets are considered to be offensive words by the CF coders.

Use WordCloud to:

```
from wordcloud import WordCloud
# imshow-Display data as an image
# interpolation - https://matplotlib.org/3.2.1/gallery/images\_contours\_and\_fields/interpolation\_methods.html
all_words = ' '.join([text for text in dataset['processed_tweets'] ])
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(all_words)
#random=0.30
plt.figure(figsize=(10, 7))|
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

Output :-



Explanation:-

Use wordcloud to show the text use in Fake data as a wordcloud banner. The word which occurs frequently are represented by larger font size.

Importing the hate speech dataset

```
dataset = panda.read_csv("HateSpeechData.csv")
```

dataset

Explanation:-

The csv file(dataset) is read using the “panda.read_csv(‘filename or link)’” command.

Understanding the Hate Speech Data

```
dataset.head()
```

Explanation:-

head(), gives us a quick look at our data set.

Use Wordcloud to :

```
# visualizing which of the word is most commonly used for hatred speech
hatred_words = ' '.join([text for text in dataset['processed_tweets'][dataset['class'] == 0]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(hatred_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

Output :-



Explanation:-

Use word-cloud to show the text use in Hate speech data as a word-cloud banner. The word which occurs frequently are represented by larger font size and in hate speech there is available in the status

Running various model Using TFIDF without additional features

```
X = tfidf
y = dataset['class'].astype(int)
X_train_tfidf, X_test_tfidf, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)
model = LogisticRegression().fit(X_train_tfidf,y_train)
y_preds = model.predict(X_test_tfidf)
report = classification_report( y_test, y_preds )
print(report)
acc=accuracy_score(y_test,y_preds)
print("Logistic Regression, Accuracy Score:" , acc)
```

Explanation:-

If you don't specify the random_state in the code,
then every time you run(execute) your code a new random value is generated
and the train and test datasets would have different values each time.

Output:-

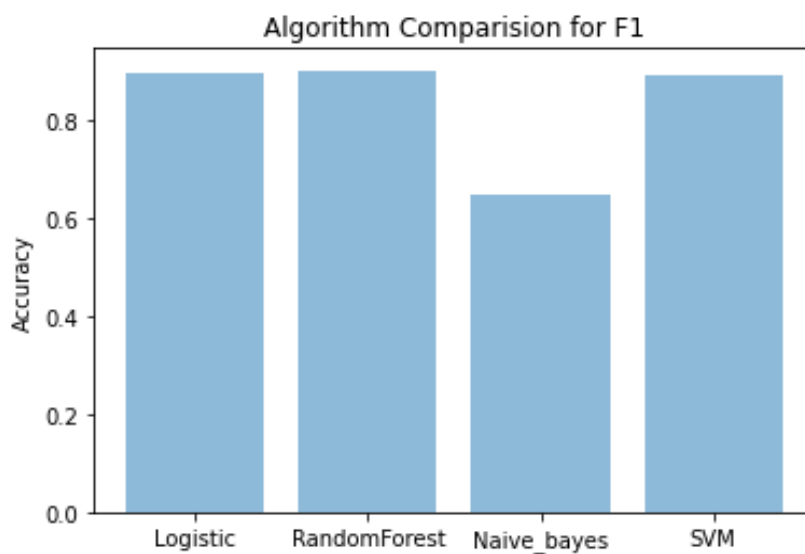
	precision	recall	f1-score	support
0	0.46	0.15	0.23	290
1	0.93	0.96	0.94	3832
2	0.83	0.91	0.87	835
accuracy			0.90	4957
macro avg	0.74	0.67	0.68	4957
weighted avg	0.89	0.90	0.89	4957

Random Forest, Accuracy Score: 0.9023602985676821

Sentiment Analysis, using polarity scores as features(F1)

```
objects = ('Logistic', 'RandomForest', 'Naive_bayes', 'SVM')
y_pos = np.arange(len(objects))
performance = [acc, acc1, acc2, acc3]
plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy')
plt.title('Algorithm Comparision for F1')
plt.show()
```

Output:-



Explanation:-

Sentiment Analysis, using polarity scores as features
Check the accuracy of the algorithms

Running various model Using TFIDF and additional features

```
X = panda.DataFrame(modelling_features)
y = dataset['class'].astype(int)
X_train_bow, X_test_bow, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

model = LogisticRegression().fit(X_train_bow,y_train)
y_preds = model.predict(X_test_bow)
report = classification_report( y_test, y_preds )
print(report)
acc=accuracy_score(y_test,y_preds)
print("Logistic Regression,Accuracy Score:" , acc)
```

OutPut:-

	precision	recall	f1-score	support
0	0.60	0.19	0.29	290
1	0.92	0.96	0.94	3832
2	0.85	0.84	0.85	835
accuracy			0.90	4957
macro avg	0.79	0.67	0.69	4957
weighted avg	0.89	0.90	0.89	4957

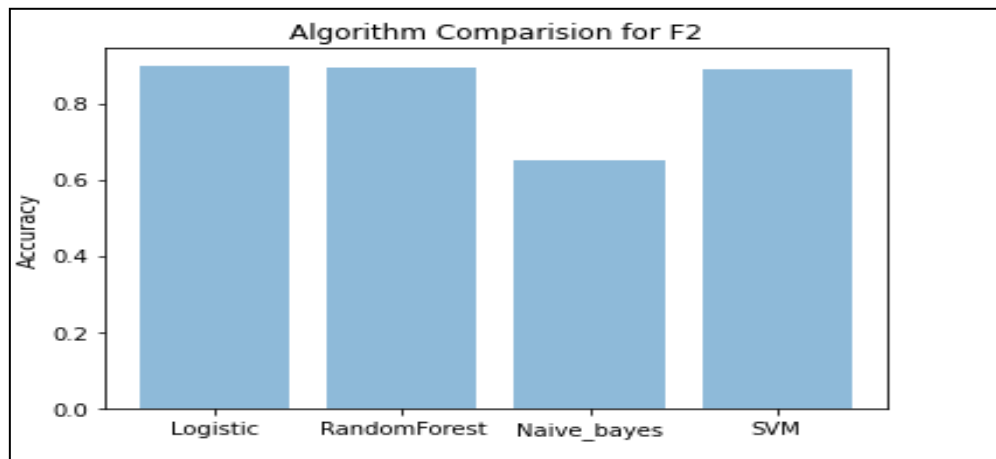
Logistic Regression,Accuracy Score: 0.898930804922332

Sentiment Analysis, using polarity scores as features(F2)

```
In [24]: objects = ('Logistic', 'RandomForest', 'Naive_bayes', 'SVM')
y_pos = np.arange(len(objects))
performance = [acc,acc1,acc2,acc3]
plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy')
plt.title('Algorithm Comparision for F2')
plt.show()
```

Explanation:- Sentiment Analysis, using polarity scores as features , Check the accuracy of the algorithms,F2 Algorithm.

OutPut:-



Running the models Using TFIDF with additional features from sentiment analysis and doc2vec

```
In [27]: X = panda.DataFrame(modelling_features)
y = dataset['class'].astype(int)
X_train_bow, X_test_bow, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

model = LogisticRegression().fit(X_train_bow,y_train)
y_preds = model.predict(X_test_bow)
report = classification_report( y_test, y_preds )
print(report)
acc=accuracy_score(y_test,y_preds)
print("Logistic Regression, Accuracy Score:" , acc)
```

Output :-

	precision	recall	f1-score	support
0	0.58	0.17	0.26	290
1	0.92	0.96	0.94	3832
2	0.84	0.84	0.84	835
accuracy			0.90	4957
macro avg	0.78	0.66	0.68	4957
weighted avg	0.88	0.90	0.88	4957

Logistic Regression, Accuracy Score: 0.8971151906394997

Running the models Using TFIDF with sentiment scores,doc2vec and enhanced features

```
X = panda.DataFrame(modelling_features_enhanced)
y = dataset['class'].astype(int)
X_train_features, X_test_features, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)

model = LogisticRegression().fit(X_train_features,y_train)
y_preds = model.predict(X_test_features)
report = classification_report( y_test, y_preds )
print(report)
acc=accuracy_score(y_test,y_preds)
print("Logistic Regression, Accuracy Score:" , acc)
```

Output :-

	precision	recall	f1-score	support
0	0.33	0.00	0.01	279
1	0.83	0.96	0.89	3852
2	0.67	0.37	0.48	826
accuracy			0.81	4957
macro avg	0.61	0.45	0.46	4957
weighted avg	0.77	0.81	0.77	4957

Logistic Regression, Accuracy Score: 0.8117813193463789

Explanation:- # Running the model Using TFIDF with enhanced features.

Add the #Confusion Matrix for TFIDF with additional features

```

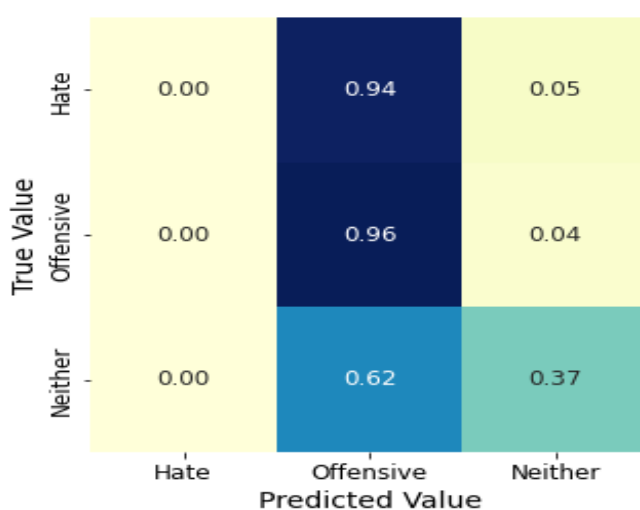
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,y_preds)
matrix_proportions = np.zeros((3,3))
for i in range(0,3):
    matrix_proportions[i,:] = confusion_matrix[i,:]/float(confusion_matrix[i,:].sum())
names=['Hate','Offensive','Neither']
confusion_df = panda.DataFrame(matrix_proportions, index=names,columns=names)
plt.figure(figsize=(5,5))
seaborn.heatmap(confusion_df,annot=True,annot_kws={"size": 12},cmap='YlGnBu',cbar=False, square=True,fmt='.2f')
plt.ylabel(r'True Value',fontsize=14)
plt.xlabel(r'Predicted Value',fontsize=14)
plt.tick_params(labelsize=12)

```

Explanation:- From the confusion matrix its clear that the model misclassifies 78% of the hate data as offensive data. This explains the reduction in class-0

bar on the histogram for the predicted class and increase of bar for class-1 (offensive).

OutPut:-



Print the # Histogram presenting the count of different classes- Actual

```

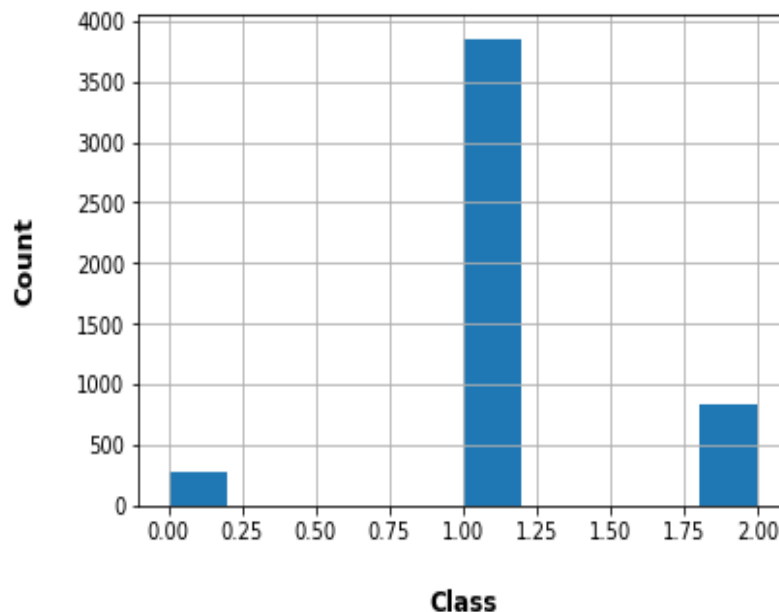
ax=y_test.hist()
ax.set_xlabel("Class", labelpad=20, weight='bold', size=12)

```

```
ax.set_ylabel("Count", labelpad=20, weight='bold', size=12)
```

Output:-

```
Out[38]: Text(0, 0.5, 'Count')
```



Explanation:-

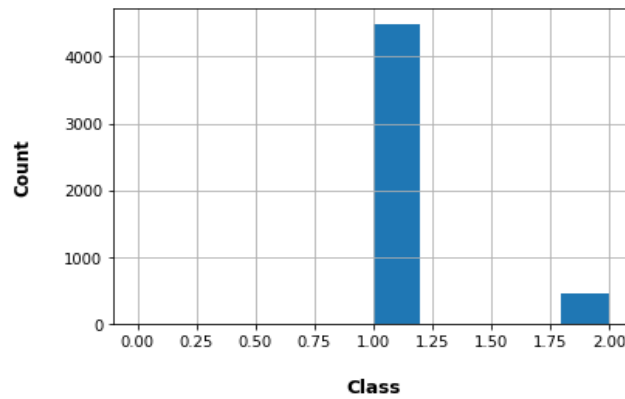
Histogram presenting the count of different classes.

Histogram presenting the count of different classes- Predicted

```
ax=panda.Series(y_preds).hist()  
ax.set_xlabel("Class", labelpad=20, weight='bold', size=12)  
ax.set_ylabel("Count", labelpad=20, weight='bold', size=12)
```

Output:-

Out[39]: Text(0, 0.5, 'Count')

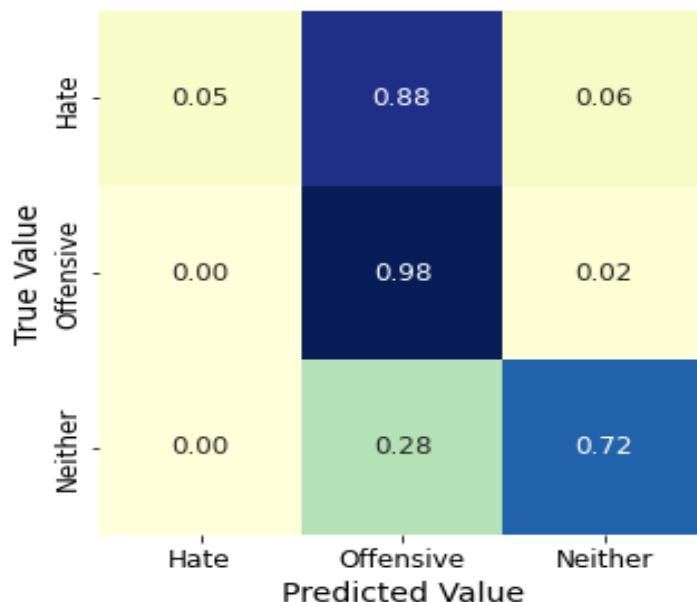


#Confusion Matrix for TFIDF with additional features

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,y_preds)
matrix_proportions = np.zeros((3,3))
for i in range(0,3):
    matrix_proportions[i,:] = confusion_matrix[i,:]/float(confusion_matrix[i,:].sum())
names=['Hate','Offensive','Neither']
confusion_df = panda.DataFrame(matrix_proportions, index=names,columns=names)
plt.figure(figsize=(5,5))
seaborn.heatmap(confusion_df,annot=True,annot_kws={"size": 12},cmap='YlGnBu',cbar=False, square=True,fmt='.2f')
plt.ylabel(r'True Value',fontsize=14)
plt.xlabel(r'Predicted Value',fontsize=14)
plt.tick_params(labelsize=12)
```

Explanation:- To avoid some typy of confusion with hate speech.

OutPut:-

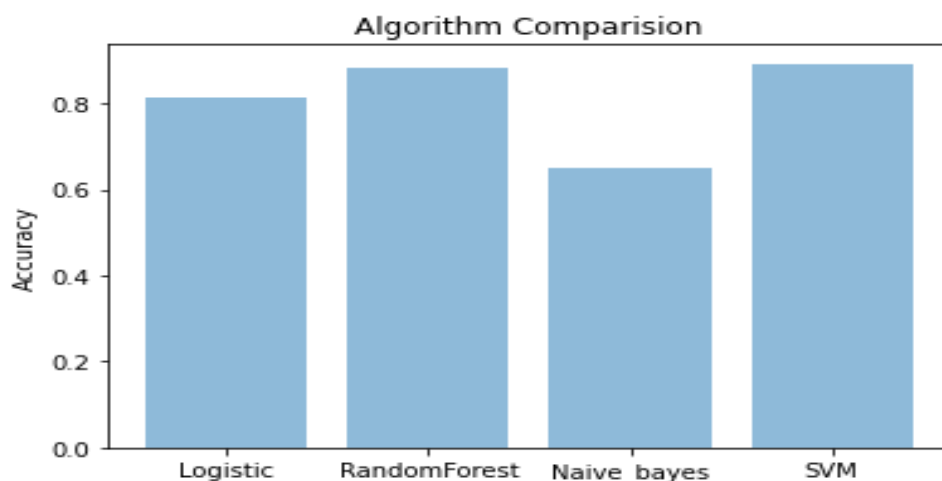


Algorithm Comparision

```
In [44]: objects = ('Logistic', 'RandomForest', 'Naive_bayes', 'SVM')
y_pos = np.arange(len(objects))
performance = [acc, acc1, acc2, acc3]
plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy')
plt.title('Algorithm Comparision')
plt.show()
```

Explanation:- For Comparision Algorithms

OutPut:-



Combining the database of Hate speech:

```
hate = hate[['text', 'class' ]]
```

Explanation:- Combine test and class column in hate speech dataset.

Combining different features:

```
tfidf_a = tfidf.toarray()
modelling_features_one =
np.concatenate([tfidf_a, doc2vec_df, fFeatures], axis=1)
modelling_features_one.shape
```

Explanation:- Combining others features put it into data .

Install some package:

```
!pip install spacy==2.2.3
!python -m spacy download en_core_web_sm
!pip install beautifulsoup4==4.9.1
!pip install textblob==0.15.3
!pip install git+https://github.com/laxmimerit/preprocess_kgptalkie.
git --upgrade --force-reinstall
```

Output :-

```
Collecting spacy==2.2.3
  Downloading spacy-2.2.3-cp37m-manylinux1_x86_64.whl (10.4 MB)
    |████████████████████| 10.4 MB 5.3 MB/s
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (1.1.3)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (1.19.5)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (3.0.5)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (2.23.0)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (1.0.5)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (2.0.5)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (57.4.0)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (0.8.2)
Requirement already satisfied: srsly<1.1.0,>=0.1.0 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (1.0.5)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (0.4.1)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /usr/local/lib/python3.7/dist-packages (from spacy==2.2.3) (1.0.0)
Collecting thinc<7.4.0,>=7.3.0
  Downloading thinc-7.3.1-cp37m-manylinux1_x86_64.whl (2.2 MB)
    |████████████████████| 2.2 MB 22.3 MB/s
Requirement already satisfied: importlib-metadata>=0.20 in /usr/local/lib/python3.7/dist-packages (from catalogue<1.1.0,>=0.0.7->spacy==2.2.3) (4.6.4)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=0.20->catalogue<1.1.0,>=0.0.7->spacy==2.2.3) (3.5.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=0.20->catalogue<1.1.0,>=0.0.7->spacy==2.2.3) (3.7.4.3)
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy==2.2.3) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy==2.2.3) (2021.5.30)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy==2.2.3) (2.10)
Requirement already satisfied: chardet4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.13.0->spacy==2.2.3) (3.0.4)
Requirement already satisfied: tqdm<5.0.0,>=4.10.0 in /usr/local/lib/python3.7/dist-packages (from thinc<7.4.0,>=7.3.0->spacy==2.2.3) (4.62.0)
Installing collected packages: thinc, spacy
  Attempting uninstall: thinc
    Found existing installation: thinc 7.4.0
    Uninstalling thinc-7.4.0:
      Successfully uninstalled thinc-7.4.0
```

Explanation:- Install some python package.

Import preprocess_kgptalkie

```
import preprocess_kgptalkie as ps
```

Explanation:-

```
import preprocess_kgptalkie
```

Removing the special Character:

```
data['text'] = data['text'].apply(lambda x: ps.remove_special_chars(x))
```

Explanation:-

Remove all the special character from data set.

Understanding the data Data (Preprocessing of the tweets)

```
data.head()
```

```
dataset['processed_tweets'] = processed_tweets
```

```
print(dataset[["tweet","processed_tweets"]].head(10))
```

Explanation:-head(), gives us a quick look at our data set.

Output :-

```

                                tweet  \
0  !!! RT @mayasolovely: As a woman you shouldn't...
1  !!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2  !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3  !!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4  !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
5  !!!!!!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...
6  !!!!!!"@_BrighterDays: I can not just sit up ...
7  !!!!!&#8220;@selfiequeenbri: cause I'm tired of...
8  " & you might not get ya bitch back & ...
9  " @rhythmixx_ :hobbies include: fighting Maria...

                                processed_tweets
0  woman complain clean hous amp man alway take t...
1  boy dat cold tyga dwn bad cuffin dat hoe st place
2      dawg ever fuck bitch start cri confus shit
3      look like tranni
4  shit hear might true might faker bitch told ya
5  shit blow claim faith somebodi still fuck hoe
6      sit hate anoth bitch got much shit go
7      caus tire big bitch come us skinni girl
8      amp might get ya bitch back amp that
9      hobbi includ fight mariam bitch
```

Import gensim

```
import gensim
```

Explanation:-

Importing gensim library.

Set the data class value

```
y = data['class'].values
```

Explanation:-

Set the subfield class value of data in y or targeting input as y.

Split the text

```
X = [d.split() for d in data['text'].tolist()]
```

Explanation:-

All the text data are converted into sequential of list and by using list comprehension method ,all the text into list of list and store in x list type variable.

Print the type

```
type(X[0])
```

Output:-

List

Explanation:-

Print the type of the variable x

Print the element in x

```
print(X[0])
```

Output:-

```
['as', 'us', 'budget', 'fight', 'looms', 'republicans', 'flip', 'their', 'fiscal']
```

Explanation:-

Print the element of the variable x, and get list of list.

Convert word to vector

```
In [25]: # create doc2vec vector columns
# Initialize and train the model
from gensim.test.utils import common_texts
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
#The input for a Doc2Vec model should be a list of TaggedDocument(['List', 'of', 'word'], [TAG_001]).
#A good practice is using the indexes of sentences as the tags.
documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(dataset["processed_tweets"].apply(lambda x: x.split(" ")))

# train a Doc2Vec model with our text data
# window- The maximum distance between the current and predicted word within a sentence.
# mincount-Ignores all words with total frequency lower than this.
# workers -Use these many worker threads to train the model
# Training Model - distributed bag of words (PV-DBOW) is employed.
model = Doc2Vec(documents, vector_size=5, window=2, min_count=1, workers=4)

#infer_vector - Infer a vector for given post-bulk training document.
# Syntax- infer_vector(doc_words, alpha=None, min_alpha=None, epochs=None, steps=None)
# doc_words-A document for which the vector representation will be inferred.

# transform each document into a vector data
doc2vec_df = dataset["processed_tweets"].apply(lambda x: model.infer_vector(x.split(" "))).apply(panda.Series)
doc2vec_df.columns = ["doc2vec_vector_" + str(x) for x in doc2vec_df.columns]
doc2vec_df
```

conctaenation of tf-idf scores, sentiment scores and doc2vec columns

```
modelling_features = np.concatenate([tfidf_a,final_features,doc2vec_df],axis=1)
```

```
modelling_features.shape
```

Output:-

(24783, 6453)

Explanation:-

No. of words in this vocabulary

Print the vector data set of w2v_model

```
w2v_model.wv.vocab
```

Output:-

```
↳ {'as': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc2ed0>,
    'us': <gensim.models.keyedvectors.Vocab at 0x7f3490a55410>,
    'budget': <gensim.models.keyedvectors.Vocab at 0x7f3490a55a90>,
    'fight': <gensim.models.keyedvectors.Vocab at 0x7f3490a55b90>,
    'looms': <gensim.models.keyedvectors.Vocab at 0x7f3490a556d0>,
    'republicans': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5e50>,
    'flip': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5f90>,
    'their': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5d50>,
    'fiscal': <gensim.models.keyedvectors.Vocab at 0x7f3490a55ed0>,
    'script': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc59d0>,
    'the': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5cd0>,
    'head': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5950>,
    'of': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5610>,
    'a': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5b10>,
    'conservative': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5690>,
    'republican': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5750>,
    'faction': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5f50>,
    'in': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5090>,
    'congress': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5650>,
    'who': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5b50>,
    'voted': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5350>,
    'this': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5410>,
    'month': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5a10>,
    'for': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5150>,
    'huge': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5990>,
    'expansion': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5890>,
    'national': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5a50>,
    'debt': <gensim.models.keyedvectors.Vocab at 0x7f33fcdc5850>,
    'to': <gensim.models.kevedvectors.Vocab at 0x7f33fcdc5910>.
```

Explanation:-

Print the vector data set of w2v_model, 231911 are those unique words for which this vector are created.

Print the vector data set with word usa

```
doc2vec_df = dataset["processed_tweets"]
```

Output:-

Out[25]:

	doc2vec_vector_0	doc2vec_vector_1	doc2vec_vector_2	doc2vec_vector_3	doc2vec_vector_4
0	-0.082253	0.046886	0.011808	-0.154324	0.081295
1	0.155935	0.059153	0.158496	-0.036872	-0.009260
2	-0.056845	0.064258	-0.032898	0.030926	-0.114712
3	0.089205	0.007175	-0.114093	-0.067137	0.089104
4	-0.055238	0.019616	0.184693	0.022708	-0.247036
...
24778	0.275679	0.236415	0.192969	-0.297502	-0.046564
24779	-0.037886	0.131372	0.186789	-0.089816	-0.072251
24780	-0.061102	0.264164	0.181874	0.101063	-0.259724
24781	0.020009	-0.025722	0.073097	-0.112793	-0.144977
24782	0.450743	-0.155761	0.376136	-0.357863	0.251160

24783 rows × 5 columns

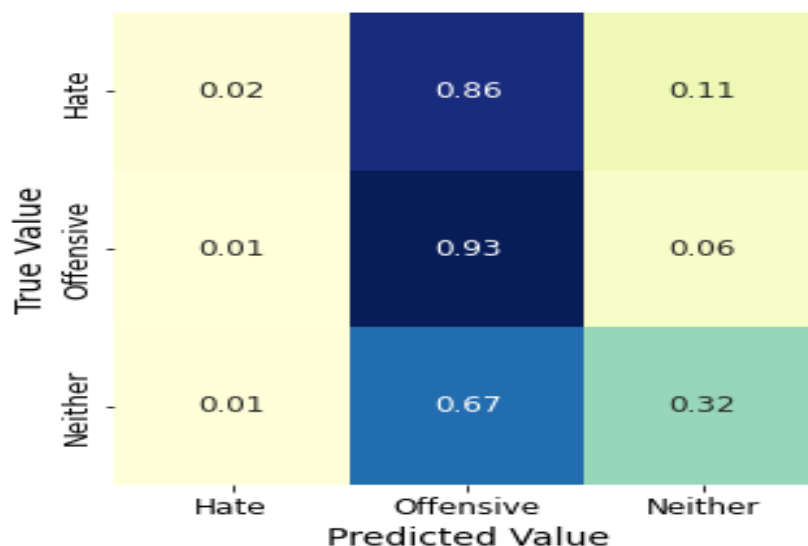
Explanation:-

Print the vector data set which contains the hate word, this are vector which have converted by social media.

#Confusion Matrix for TFIDF with additional features

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,y_preds)
matrix_proportions = np.zeros((3,3))
for i in range(0,3):
    matrix_proportions[i,:] = confusion_matrix[i,:]/float(confusion_matrix[i,:].sum())
names=['Hate','Offensive','Neither']
confusion_df = panda.DataFrame(matrix_proportions, index=names,columns=names)
plt.figure(figsize=(5,5))
seaborn.heatmap(confusion_df,annot=True,annot_kws={"size": 12},cmap='YlGnBu',cbar=False, square=True,fmt='.2f')
plt.ylabel(r'True Value',fontsize=14)
plt.xlabel(r'Predicted Value',fontsize=14)
plt.tick_params(labelsize=12)
```

Output:-



Explanation:-

Print the hate speech,neither,offensive predicted data set which is most similar with the word social media,etc.

Split the word with tokenizer

```
tokeniser = Tokenizer()  
tokeniser.fit_on_texts(X)
```

Explanation:-

Split the words based on regular expression.

Set the text to sequences to

```
X= tokeniser.texts_to_sequences(X)
```

Explanation:-

Set the text to sequences to x, x is converted into set of sequence of numbers.

Add a numpy array contains the length of x

```
nos = np.array([len(x) for x in X])  
len(nos[nos>1000])
```

Output:-

1584

Explanation:-

Add a numpy array contains the length of length of x in nos and put the count the of len of nos where nos is greater than 1000, which means 1584 news having more than 1000 words.

Set the preferred length to x

```
maxlen = 1000
```

```
X = pad_sequences(X, maxlen=maxlen)
```

Explanation:-

Set a preferred length which longer than x observed sequences to x, every sequence at the length of 1000,i.e when the sequence is greater than 1000 was truncated and if less than 1000 then padded.

Store the vector of model in an array

```
def get_weight_matrix(model):  
    weight_matrix = np.zeros((vocab_size,DIM))  
  
    for word, i in vocab.items():
```

```
weight_matrix[i] = model.wv[word]
```

```
return weight_matrix
```

Explanation:-Store the vector of model in an array weight_matrix, this vectors are fed as initial input vectors in ML model and then model retrain to get max accuracy.

Store the matrix

```
embedding_vectors = get_weight_matrix(w2v_model)
```

Explanation:-

Store the get_weight_matrix(w2v_model) in embedding_vectors.

Train the x and y of the given data set

```
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

Explanation:-

Train the x and y from the given data set by just simply spiting them

Find the model prediction


```
y_preds = rf.predict(X_test_features)
acc1=accuracy_score(y_test,y_preds)
report = classification_report( y_test, y_preds )
```

Explanation:-

Predict the model from x_test as a int type and store in y_predict

Print the accuracy score

```
accuracy_score(y_test , y_pred)
```

Output:-

```
0.992338530066815
```

Explanation:-

Printing the accuracy score of the test model.and we get almost 99% accuracy.

Print the Classification Result

```
report = classification_report( y_test, y_preds )
print(report)
acc=accuracy_score(y_test,y_preds)
```

```
print("Logistic Regression, Accuracy Score:" , acc)
```

Output:-

	precision	recall	f1-score	support
0	0.58	0.17	0.26	290
1	0.92	0.96	0.94	3832
2	0.84	0.84	0.84	835
accuracy			0.90	4957
macro avg	0.78	0.66	0.68	4957
weighted avg	0.88	0.90	0.88	4957

Explanation:-

Printing the Classification report of the trained model

Example:

Machine Learning App Using Flask

Hate Speech / Tweet Detector

Enter Your Tweet Here

@user #white #supremacists want everyone to see the new #birds #birds #birds #birds #birds and here #birds #birds #birds why

Predict

Hate Tweet

© 2019 GitHub, Inc.

Machine Learning App Using Flask

Hate Speech / Tweet Detector

Enter Your Tweet Here

#studiolife #aislife #requires #passion #dedication #willpower to find #newmaterials

Predict

Not a Hate Tweet

© 2019 GitHub, Inc.

Status: Success

CONCLUSION

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. In this research, we discussed the problem of classifying fake news articles using machine learning models and ensemble techniques. The data we used in our work is collected from the World Wide Web and contains news articles from various domains to cover most of the news rather than specifically classifying political news. The primary aim of the research is to identify patterns in text that differentiate fake articles from true news. We extracted different textual features from the articles using an LIWC tool and used the feature set as an input to the models. The learning models were trained and parameter-tuned to obtain optimal accuracy. Some models have achieved comparatively higher accuracy than others. We used multiple performance metrics to compare the results for each algorithm. The ensemble learners have shown an overall better score on all performance metrics as compared to the individual learners

UPLODED PROJECT AND DOCUMENT IN OPENSORSE

- Opensource Code link:

https://github.com/Ajit0011/Project-HATE-SPEECH_ML_SIT

- Documentation Link:

- https://github.com/Ajit0011/Project_Report_ML

- <https://drive.google.com/drive/folders/1L6IC5737byXgtopomgv2EKj7DcMmcxkO?usp=sharing>

BIBLIOGRAPHY

- https://www.tensorflow.org/api_docs/python/tf/all_symbols
 - www.kaggle.com
 - <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
 - <https://scikit-learn.org/>
 - <https://www.visualcapitalist.com/fake-news-problem-one-chart/>
-

THANK YOU