**PREDICTION OF HATE SPEECH CLASSIFICATION BY USING SUPERVISED MACHINE LEARNING WITH NLP**

##### A PROJECT REPORT

###### ***Submitted by***

|  |  |
| --- | --- |
| **ASWINI T** | **[REGISTER NO:211419104031]** |
| **HEMALATHA P** | **[REGISTER NO:211419104100]** |
| **KAVYA S** | **[REGISTER NO:211419104135]** |

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**APRIL 2023**

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**BONAFIDE CERTIFICATE**

Certified that this project report **“Prediction of Hate Speech Classification by using Supervised Machine Learning with NLP”** is the bonafide work of “**ASWINI T (211419104031) , HEMALATHA P (211419104100) , KAVYA S (211419104135)”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.L.JABASHEELA,M.E.,Ph.D., Dr.KAVITHA SUBRAMANI,M.E,Ph.D.,**

**HEAD OF THE DEPARTMENT SUPERVISOR**

**PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE,

NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**DECLARATION**

We **ASWINI T (211419104031) , HEMALATHA P (211419104100) , KAVYA S (211419104135)** hereby declare that this project report titled “**Prediction of Hate Speech Classification by using Supervised Machine Learning with NLP**” , under the guidance of **Dr. Kavitha Subramani M.E,Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**ASWINI T**

**HEMALATHA P**

**KAVYA S**

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYA RAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank our guide **Dr.Kavitha Subramani M.E,Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

**ASWINI T**

**HEMALATHA P**

**KAVYA S**

**ABSTRACT**

Digital media has become a significant factor in many person’s day to day routine. Hate speech is a story that has been created in an intention to distract or misguide the readers. Due to increase in the online social network development in the past few years due to different purposes hate speech appear in large numbers and in the online world has a widespread. By these online hate speech online social networks users can get effected easily Hate speech have become a society problem, in some occasion spreading more and faster than the true information. A human being is unable to detect all these hate speech. So there is a need for machine learning model that can detect these hate speech automatically. Machine learning models are made to build using the algorithms so that it can classify whether a speech is hate speech, offensive speech and not hate and offensive speech. The dataset for this hate speech classification project has been taken from the Kaggle which includes the features such as count, hate speech, offensive, neither hate nor offensive, class and tweet. This project used three algorithms to detect hate speech such as Naive Bayes, Random Forest and Gradient Boosting. Different machine learning models have different strengths that make some better than others for certain tasks such as detecting hate speech. Some models are more accurate while others are more efficient. It is important to use different models and compare their performance in order to find the best one for hate speech detection. Gradient Boosting Algorithm yields the highest accuracy compared to the other algorithms. So the project deployment uses the Gradient Boosting Algorithm.

v

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE**  **NO** | **NAME OF THE TABLE** | **PAGE**  **NO** |
| 2.1 | Comparison table based on literature survey | 4 |
| A1.1 | Sample Dataset for hate speech classification | 39 |
|  |  |  |
|  |  |  |

vi

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE**  **NO** | **NAME OF THE FIGURE** | **PAGE**  **NO** |
| 4.1 | Use Case Diagram | 15 |
| 4.2 | Class Diagram | 16 |
| 4.3 | Activity Diagram | 17 |
| 4.4 | Sequence Diagram | 18 |
| 4.5 | Level 0 Data Flow diagram | 19 |
| 4.6 | Level 1 Data Flow diagram | 19 |
| 4.7 | Level 2 Data Flow diagram | 20 |
| 5.1 | System Architecture diagram | 21 |
| 5.2 | Work flow diagram | 22 |
| 7.1 | Screenshot of webpage for Hate Speech Classification | 29 |
| 7.2 | Screenshot of entering the text for classification | 29 |
| 7.3 | Screenshot of the prediction of Hate Speech | 30 |
| 7.4 | Screenshot of entering the text for classification | 30 |
| 7.5 | Screenshot of the prediction of Offensive language | 31 |
| 7.6 | Screenshot of entering the text for classification | 31 |
| 7.7 | Screenshot of the prediction of No hate and offensive | 32 |
| 7.8 | Screenshot of classification report of Random forest algorithm | 33 |
| 7.9 | Screenshot of classification report of Naïve Bayes Algorithm | 34 |
| 7.10 | Screenshot of classification report of Gradient Boosting algorithm | 34 |
| 7.11 | Screenshot of accuracy of Random Forest algorithm | 35 |
| 7.12 | Screenshot of accuracy of Naive bayes algorithm | 35 |
| 7.13 | Screenshot of accuracy of Gradient Boosting algorithm | 36 |
| 7.14 | Comparison of three models accuracy | 36 |
| 7.15 | Comparison of accuracies in hate speech classification | 37 |
| A3.1 | Screenshot of Data Preprocessing Coding | 65 |
| A3.2 | Screenshot of Data Visualization coding | 65 |
| A3.3 | Screenshot of Random forest algorithm coding | 66 |
| A3.4 | Screenshot of Naive Bayes algorithm coding | 66 |
| A3.5 | Screenshot of Gradient Boosting algorithm coding  vii | 67 |

**TABLE OF CONTENTS**

| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | **ABSTRACT** | v |
|  | **LIST OF TABLES** | vi |
|  | **LIST OF FIGURES** | vii |
| **1.** | **INTRODUCTION** | 1 |
|  | 1.1 Problem Definition | 1 |
| **2.** | **LITERATURE SURVEY** | 2 |
|  | 2.1 Literature Survey | 2 |
|  | 2.2 Comparison table based on literature survey | 4 |
| **3.** | **SYSTEM ANALYSIS** | 9 |
|  | 3.1 Existing System | 9 |
|  | 3.2 Proposed system | 10 |
|  | 3.3 Feasibility Study | 11 |
|  | 3.4 Project Requirements | 12 |
| **4.** | **SYSTEM DESIGN** | 15 |
|  | 4.1. UML Diagrams | 15 |
|  | 4.1.1. Use Case Diagram | 15 |
|  | 4.1.2. Class Diagram | 16 |
|  | 4.1.3. Activity Diagram | 17 |
|  | 4.1.4. Sequence Diagram | 18 |
|  | 4.2 Dataflow Diagram | 19 |
| **5.** | **SYSTEM ARCHITECTURE** | 21 |
|  | 5.1 System Architecture | 21 |
|  | 5.2 Workflow Diagram | 22 |
| **6.** | **SYSTEM IMPLEMENTATION** | 23 |
|  | 6.1 Module Description | 23 |
|  | 6.1.1 Data preprocessing | 23 |
|  | 6.1.2 Data Validation/Cleaning/Preparing Process | 25 |
|  | 6.1.3 Machine Learning Model Development | 25 |
|  | 6.1.3.1 Random Forest Algorithm | 26 |
|  | 6.1.3.2 Naïve Bayes Algorithm | 26 |
|  | 6.1.3.3 Gradient Boosting Algorithm | 27 |
|  | 6.1.4 Deployment using Flask | 28 |
| **7.** | **PERFORMANCE EVALUATION** | 29 |
|  | 7.1 Results and Discussion | 29 |
|  | 7.2 Comparative Analysis | 33 |
| **8.** | **CONCLUSION** | 38 |
|  | 8.1 Conclusion and Future Enhancements | 38 |
|  | **APPENDICES** | 39 |
|  | APPENDIX 1 Sample Dataset | 39 |
|  | APPENDIX 2 Sample Coding | 45 |
|  | APPENDIX 3 Sample Screens | 65 |
|  | **REFERENCES** | 68 |

**CHAPTER 1**

**INTRODUCTION**

**CHAPTER-1**

**INTRODUCTION**

**PROBLEM DEFINITION**

Hate speech is a poisonous discourse that can swiftly spread on social media or due to prejudices or disputes between different groups within and across countries. A hate crime refers to crimes committed against a person due to their actual or perceived affiliation with a specific group. The protected characteristics of Facebook define hate speech as an attack on an individual’s dignity, including their race, origin, or ethnicity. According to Twitter policies, tweets should not be used to threaten or harass others due to their ethnicity, gender, religion, or any other factor. In addition to age, caste, and handicap, YouTube also censors content that promotes violence or hatred toward certain persons or groups.Often, hate speech regarding online radicalization or criminal activities is studied. Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources.The performance and accuracy of a model depends on the size and quality of training dataset. Therefore, the training data must be carefully chosen to get more accuracy and good performance. This work demonstrates supervised machine learning technique along with NLP which is used to work with natural human language. Data visualization is done to get better insights about the data. Three algorithms are used for hate speech detection. The three models are compared based on accuracy and deployed.

**CHAPTER 2**

**LITERATURE SURVEY**

**CHAPTER-2**

**LITERATURE SURVEY**

**2.1 LITERATURE SURVEY**

Using deep learning algorithm [1] Created a multi-domain hate speech corpus (MHC) of English tweets that includes hate speech against religion, nationality, ethnicity, and gender in general and cover diverse domains, such as current affairs, politics, terrorism, technology, natural disasters, and human/drugs trafficking. Each instance in our dataset is manually annotated as hate or non-hate. We use the existing state-of-the-art models and present a stacked-ensemble-based hate speech classifier (SEHC) to identify hate speech from Twitter data. Our results indicate that the proposed method may serve as a strong baseline for future studies using this dataset [2] for classification such as Gated Recurrent Unit (GRU), a variety of Recurrent Neural Networks (RNNs), are used for hate speech detection The experimental results of hate speech detection, Word2Vec embedding and RNN-GRU achieved the best performance.[3] The first benchmark hate speech dataset covering multiple aspects of the issue. Each post in dataset is annotated from three different perspectives: the basic, commonly used 3-class classification, the target community, and the rationales. Models which utilize the human rationales, reduces bias towards target communities.[4]Deep natural language processing (NLP) model—combining convolutional and recurrent layers—for the automatic detection of hate speech in social media data and applied this model on the HASOC2019 corpus, and attained a macro F1 score of 0.63 in hate speech detection on the test set of HASOC[5]Describes three different Deep Neural Network (DNN) Architectures for detection of hate words in Twitter - Gated Recurrent Unit (GRU), useful in capturing sequence orders, Convolution Neural Network (CNN), good for feature extraction, and Universal Language Model Fine-tuning (ULMFiT) model, which is based on transfer learning technique AWD -LSTM model was pre trained using WikiText103 dataset. This method significantly outperformed the other Architectures.[6] Compared the performance of three feature engineering techniques and eight machine learning algorithms to evaluate their performance on a publicly available dataset having three distinct classes. The output of different comparisons will be used as state-of-art techniques to compare future researches for existing automated text classification techniques.

[7] Aimed to investigate several neural network models based on convolutional neural network (CNN) and recurrent neural network (RNN) to detect hate speech in Arabic tweets. CNN model gives the best performance with an F1-score of 0.79 and AUROC of 0.89.[8] Datasets are used to train the machine using different machine learning algorithms, based on classification and regression models. The datasets consist of twitter messages with two class labels “offensive” and “not offensive”. For Malayalam language got the F1 score of 0.77, Tamil language Model got the F1 score of 0.87.[9] Experiments are conducted by training four models with three different feature sets extracted from the dataset. The models are evaluated by computing the specified evaluation metrics and implemented an approach to detect hate speech in videos which deals with converting the video into text format before passing it as input to machine learning models.[10] Addresses the issue of augmenting text data in supervised Natural Language Processing problems, exemplified by deep online hate speech classification. Yields a significant increase in multi-class hate speech detection, outperforming the baseline in the largest online hate speech database by an absolute 5.7% increase in Macro-F1 score and 30% in hate speech class recall.[11] The proposed approach automatically detects hate speech patterns and most common unigrams and use these along with sentimental and semantic features to classify tweets into hateful, offensive and clean.[12] Used a crowdsourced hate speech lexicon to collect tweets containing hate speech keywords and crowd-sourcing to label a sample of these tweets into three categories: those containing hate speech, only offensive language, and those with neither and train a multi-class classifier to distinguish. Lexical methods are effective ways to identify potentially offensive terms.

**2.2 COMPARISON TABLE BASED ON LITERATURE SURVEY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **YEAR & AUTHORS** | **TITLE** | **METHODOLOGY** | **MERITS AND DEMERITS** | **FUTURE SCOPE** |
| 2021,  Zewdie Mossie, Jenq-Haur Wang | Vulnerable community identification using hate speech detection  on social media | Distributed processing  framework, posts are automatically collected and pre-processed.Deep learning algorithms for  classification such as Gated Recurrent Unit (GRU), a variety of Recurrent Neural Networks (RNNs), are  used for hate speech detection | MERITS:The experimental results of hate speech detection, Word2Vec embedding and  RNN-GRU achieved the best performance with an AUC of 97.85% and an accuracy of 92.56%. DEMERITS:Not suitable for multi-lingual, multi-cultural,  and different social network platforms | In the future, other inherent problems in the RNN can be solved with a more powerful architecture like tree-LSTM that could learn meanings from characters and parts of words |
| 2021,Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, Animesh Mukherjee | HateXplain:  A Benchmark Dataset for Explainable Hate Speech Detection | This paper introduce HateXplain, the first benchmark hate speech dataset covering multiple aspects of the issue. Each post in dataset is annotated from three different perspectives: the basic, commonly used 3-class classification , the target community, and the rationales. | MERITS:Models which utilize the human rationales, reduces bias towards target communities DEMERITS:The focus is on English language and lack of  multilingual hate speech. | Future research on hate speech, should consider the impact of the  model performance on individual communities to have a clear understanding on the impact. |
| 2021,György Kovács Pedro Alonso Rajkumar Saini | Challenges of Hate Speech Detection in Social Media | Proposed a deep natural language processing (NLP) model—combining convolutional and recurrent layers—for the automatic detection of hate speech in social media data. We have applied our model on the HASOC2019 corpus, and attained a macro F1 score of 0.63 in hate speech detection on the test set of HASOC | MERITS:  The results showed that it was possible to significantly increase the classification score attained.  DEMERITS:  Low macro F1 score | For this, in the future we plan to extend our experiments with the use of more explainable models, as well as a more thorough examination of the explainability of our current models (transformer models for example have been successfully examined using the Captum tool) |
| 2020,Amrutha, B R and Bindu, K R | Detecting Hate Speech in Tweets Using Different Deep Neural Network Architectures | This paper describes three different Deep Neural Network (DNN) Architectures for detection of hate words in Twitter - Gated Recurrent Unit (GRU), useful in capturing sequence orders, Convolution Neural Network (CNN), good for feature extraction, and Universal Language Model Fine-tuning (ULMFiT) model, which is based on transfer learning technique | MERITS:AWD -LSTM model was pre-trained using WikiText103 dataset. This method significantly outperformed the other Architectures.  DEMERITS:  Complex architecture. | For future work,  Hate speech can be detected and classified using multiple Deep neural network architectures and also with high accuracy. |
| 2020,Sindhu Abro1 , Sarang Shaikh2 , Zafar Ali | Automatic Hate Speech Detection using Machine Learning: A Comparative Study | The aim of this paper is to compare the performance of three feature engineering techniques and eight machine learning algorithms to evaluate their performance on a publicly available dataset having three distinct classes | MERITS:  The output of different comparisons will be used as state-of-art techniques to compare future researches for existing automated text classification techniques.  DEMERITS:  It fails to predict the severity of the Hate speech | In the future, the objective is to improve the proposed ML model which can be used to predict the severity of the hate speech message as well. Moreover, to improve the proposed model’s classification performance two approaches will be used |
| 2020,Raghad Alshalan and Hend Al-Khalifa | A Deep Learning Approach for Automatic Hate Speech Detection in the Saudi Twittersphere | Aimed to investigate several neural network models based on convolutional neural network (CNN) and recurrent neural network (RNN) to detect hate speech in Arabic tweets | MERITS:  CNN model gives the best performance with an F1-score of 0.79 and AUROC of 0.89.  DEMERITS:  The dataset doesn’t support various writing styles, patterns, and topics. | Several ways to extend and improve this study in the future. The dataset can be extended to capture more writing styles, patterns, and topics. The dataset can also be annotated with multi-labels to enhance the results of the detection task beyond the binary classification |
| 2020,Varsha Pathak , Manish Joshi , Prasad A. Joshi , Monica Mundada , Tanmay Joshi | Using Machine Learning for Detection of Hate Speech and Offensive Code-Mixed Social Media text | These datasets are used to train the machine using different machine learning algorithms, based on classification and regression models. The datasets consist of twitter messages with two class labels “offensive” and “not offensive”. | MERITS:  Our model for Malayalam language got the F1 score of 0.77. For Tamil language Model we got the F1 score of 0.87.  DEMERITS:  Predicted only the specified languages. | To develop a system that could learn offensive terms from the text contents or even from speech irrespective of the language. We are interested in revealing hidden negative messages from the social media comments which may sound superficially positive. |
| 2020,Ching Seh Wu, Unnathi Bhandary | Detection of Hate Speech in Videos Using Machine Learning | Experiments are conducted by training four models with three different feature sets extracted from the dataset. The models are evaluated by computing the specified evaluation metrics. In this research, we implemented an approach to detect hate speech in videos which deals with converting the video into text format before passing it as input to machine learning models. | MERITS:  Four models were trained to compute the specified evaluation metrics.  DEMERITS:  Doesn’t support the larger size videos | For future work, it can also be extended to classifying more than three categories as well as increasing the size of the dataset for better classification. |
| 2019,Georgios Rizos,  Konstantin Hemker,  Bjorn Schuller | Augment to Prevent: Short-Text Data Augmentation in Deep Learning for Hate-Speech Classification | In this paper, we address the issue of augmenting text data in supervised Natural Language Processing problems, exemplified by deep online hate speech classification. | MERITS:  Yields a significant increase in multi-class hate speech detection, outperforming the baseline in the largest online hate speech database by an absolute 5.7% increase in Macro-F1 score and 30% in hate speech class recall.  DEMERITS:  Only for short-text data and not supports longer text inputs. | For future research, it would be interesting to see how the data augmentation techniques we have proposed perform on longer text inputs or to Natural Language Processing tasks other outside the domain of online hate speech detection. |
| 2018,  Hajime Watanabe, Mondher Bouazizi, Tomoaki Ohtsuki | Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection | The proposed approach automatically detects  hate speech patterns and most common unigrams and use  these along with sentimental and semantic features to classify tweets into hateful, offensive and clean. | MERITS:  Proposed  approach reaches an accuracy of 87.4% for the binary  classification and 78.4% for the ternary classification DEMERITS:Not suitable for richer dictionary words. | To build a richer dictionary  of hate speech patterns that can be used, along with a unigram dictionary, to detect hateful and offensive online texts. |
| 2017,Thomas Davidson,  Dana Warmsle,  Michael Macy,  Ingmar Weber | Automated Hate Speech Detection and the Problem of Offensive Language | Used a crowd-sourced hate speech lexicon to collect tweets containing hate speech keywords and crowd-sourcing to label a sample of these tweets into three categories: those containing hate speech, only offensive language, and those with neither and train a multi-class classifier to distinguish. | MERITS:  Lexical methods are effective ways to identify potentially offensive terms. DEMERITS:Lexical methods are inaccurate at identifying hate speech. | Future work should distinguish between different uses and look more closely at the social contexts and conversations in which hate speech occurs and aim to identify and correct social biases. |

**Table 2.1 Comparison table based on literature survey**

**CHAPTER 3**

**SYSTEM ANALYSIS**

**CHAPTER-3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

Sarcasm is commonly used in today’s social media platforms such as Twitter and Reddit. Sarcasm detection is necessary for analyzing people’s real sentiments as people usually use sarcasm to express a flipped emotion against the literal meaning. However, the current works neglect the fact that common sense knowledge is crucial for sarcasm recognition. In this paper, we propose a novel architecture in deep learning for sarcasm detection by integrating common sense knowledge. To be specific, we apply the pre-trained COMET model to generate relevant common sense knowledge. Besides, we compare two kinds of knowledge selection strategies to investigate how common sense knowledge influences performance. Finally, a knowledge-text integration module is designed to model both text and knowledge. The experimental results demonstrate our model’s effectiveness on three datasets, including two Twitter datasets and a Reddit dataset.

**Disadvantages:**

• The proposed model had a problem with classification as it is not that accurate.

• To a certain extent only it will predict correctly.

**3.2 PROPOSED SYSTEM**

The proposed model is to build a machine learning model that is capable of classifying whether the speech is hate speech, offensive speech and not hate and offensive speech. The hate speech are considered to be widespread and controlling them is very difficult as the world is developing toward digital everyone now has access to internet and they can post whatever they want. So there is a greater chance for the people to get misguided. The machine learning is generally build to tackle these type of complicated task like it takes more amount of time to analyze these type of data manually. The machine learning can be used to classify the speech is hate speech or not by using the previous data and make them to understand the pattern and improve the accuracy of the model by adjusting parameters and use that model as the classification model. Different algorithms can be compared and the best model can be used for classification purpose.

**Advantages:**

• The classification model build had the ability to classify the new whether it is speech is hate speech, offensive speech and not hate and offensive speech

• By classifying the new by using this process the chance of people getting misguided can be reduced.

**3.3 FEASIBILITY STUDY**

**Data Wrangling**

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

**Data collection**

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, Linear Support Vector Machine and Decision Tree Classifier are applied on the Training set and based on the test result accuracy, Test set prediction is done.The dataset for this hate speech classification project has been taken from the Kaggle.

**Preprocessing**

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

**3.4 PROJECT REQUIREMENTS**

**Functional requirements**

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

**Non-Functional Requirements**

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

**Environmental Requirements**

1. Software Requirements:

Operating System : Windows 10 or later

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

**SOFTWARE DESCRIPTION:**

**ANACONDA NAVIGATOR**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window.You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

**JUPYTER NOTEBOOK**

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.Project Jupyter is a project and community whose goal is to &quot;develop open-source software, open-standards, and services for interactive computing across dozens of programming languages&quot;. It was spun off from IPython in 2014 by Fernando Perez. Notebook documents are documents produced by the Jupyter Notebook App , which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

**FLASK**

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However,Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django,and was voted the most popular web framework in the Python Developers Survey 2018, 2019, 2020 and 2021

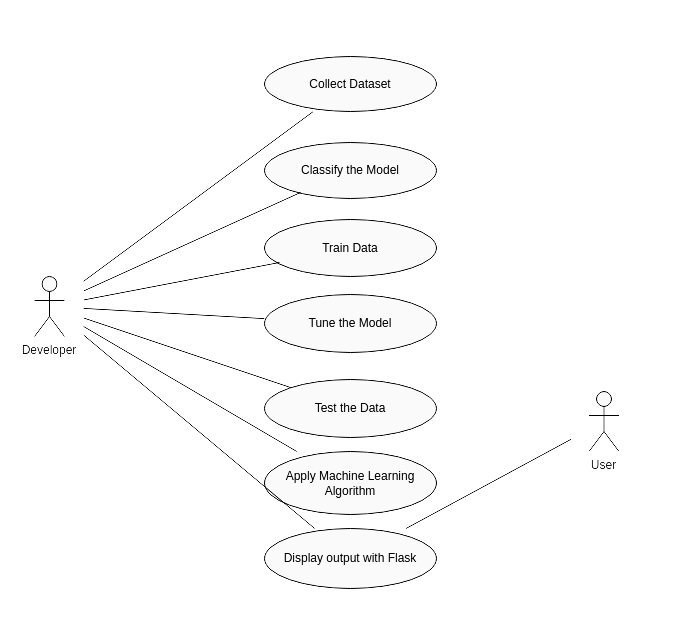
**CHAPTER 4**

**SYSTEM DESIGN**

**CHAPTER-4**

**SYSTEM DESIGN**

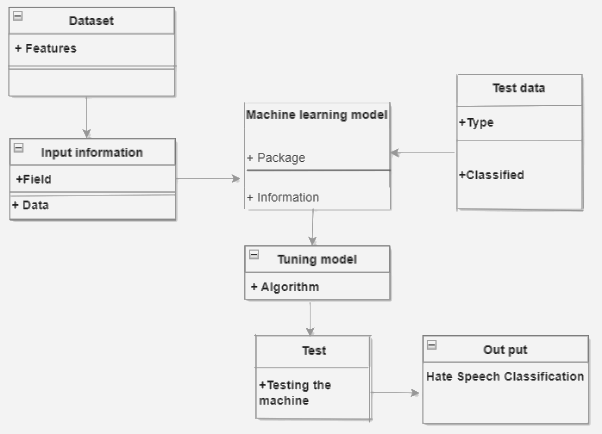
**4.1 UML DIAGRAMS**

**4.1.1 USE CASE DIAGRAM**

**Figure 4.1 Use case diagram**

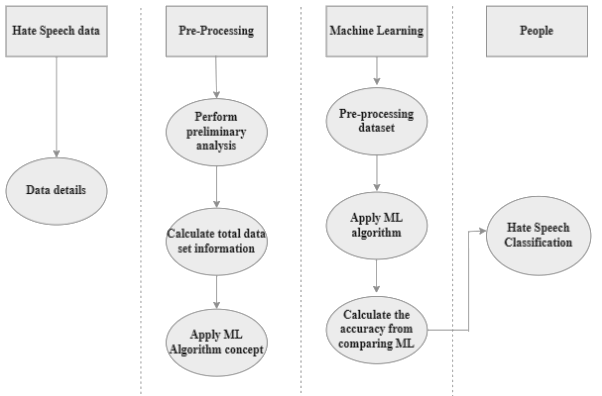
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that use cases are nothing but the system functionalities written in an organized manner.

**4.1.2 CLASS DIAGRAM**

**Figure 4.2 Class diagram**

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

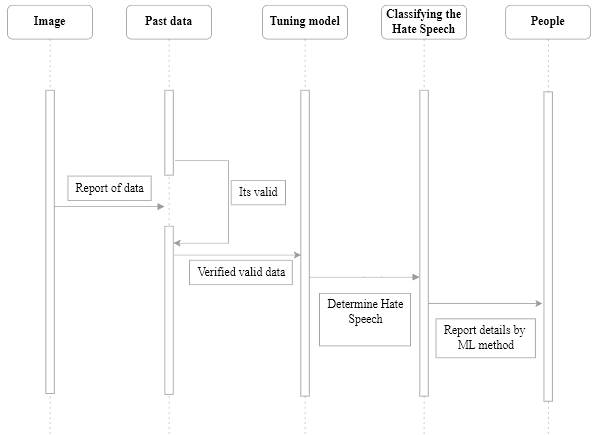
**4.1.3 ACTIVITY DIAGRAM**



**Figure 4.3 Activity diagram**

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

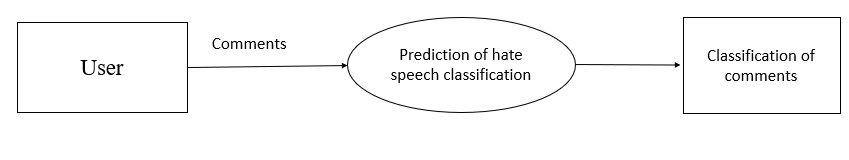
**4.1.4 SEQUENCE DIAGRAM**

**Figure 4.4 Sequence diagram**

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

**4.2 DATA FLOW DIAGRAM**

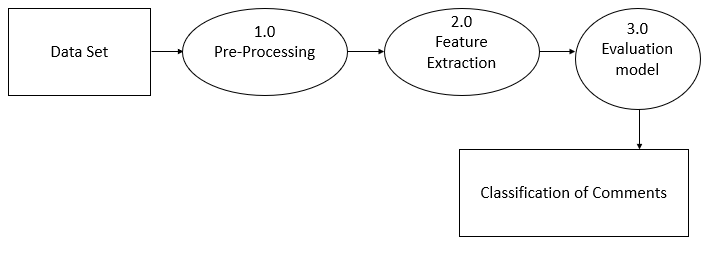
**4.2.1 LEVEL 0**



**Figure 4.5 Level 0 Data flow diagram**

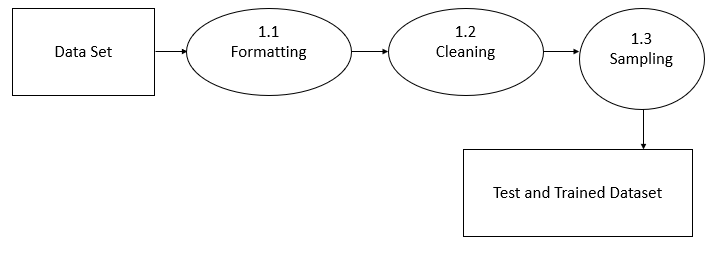
The Level 0 DFD diagram shows that the input will be given as text by the end user and the comments are classified .

**4.2.2 LEVEL 1**

**Figure 4.6 Level 1 Dataflow diagram**

The Level 1 DFD shows the way the system is divided into subsystems(processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole.It also identifies interna data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

**4.2.3. LEVEL 2**



**Figure 4.7 Level 2 Dataflow diagram**

The Level 2 DFD shows that the dataset will be divided into a test and trained dataset after being formatted and cleaned, which involves removing any incomplete values.

|  |  |
| --- | --- |
|  |  |

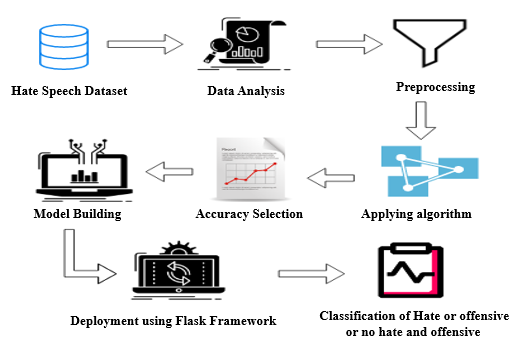
**CHAPTER 5**

**SYSTEM ARCHITECTURE**

**CHAPTER -5**

**SYSTEM ARCHITECTURE**

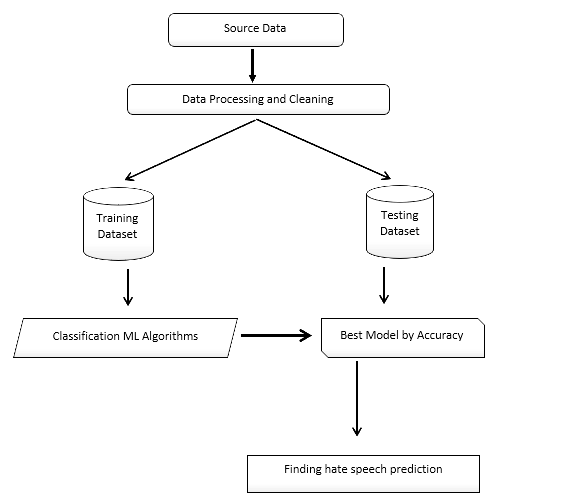
**5.1 SYSTEM ARCHITECTURE**



**Figure 5.1 System Architecture diagram**

First the Hate Speech dataset is taken from the source and cleaning and validation process is performed on the dataset which includes removal of redundancy, filling empty spaces in columns, converting necessary variable into factors or classes then data is divided into two parts, one is training dataset and another one is test dataset.Now the original sample is randomly partitioned into test and train dataset.

**5.2 WORK FLOW DIAGRAM**



**Figure 5.2 Work flow diagram**

A workflow diagram provides a graphic overview of the business process. Using standardized symbols and shapes, the workflow shows step by step how your work is completed from start to finish. It also shows who is responsible for work at what point in the process. Designing a workflow involves first conducting a thorough workflow analysis, which can expose potential weaknesses. A workflow analysis can help you define, standardize and identify critical areas of your process.

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**CHAPTER - 6**

**SYSTEM IMPLEMENTATION**

**6.1 MODULE DESCRIPTION**

**LIST OF MODULES**

* Data Pre-processing
* Data Validation/Cleaning/Preparing Process
* Applying Algorithm
* Deployment using Flask

**6.1.1 DATA PRE-PROCESSING**

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.The dataset for this hate speech classification project has been taken from the Kaggle which includes the features such as count, hate speech, offensive, neither hate nor offensive, class and tweet. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.A number of different data cleaning tasks using Python’s Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It’s important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it’s important to understand the sources of missing data. Here are some typical reasons why data is missing:

* User forgot to fill in a field.
* Data was lost while transferring manually from a legacy database.
* There was a programming error.
* Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multivariate analysis:

* 1. import libraries for access and functional purpose and read the given dataset
  2. General Properties of Analyzing the given dataset
  3. Display the given dataset in the form of data frame
  4. show columns
  5. shape of the data frame
  6. To describe the data frame
  7. Checking data type and information about dataset
  8. Checking for duplicate data
  9. Checking Missing values of data frame
  10. Checking unique values of data frame
  11. Checking count values of data frame
  12. Rename and drop the given data frame
  13. To specify the type of values
  14. To create extra columns

**6.1.2 DATA VALIDATION/CLEANING/PREPARING PROCESS**

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning models and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multivariate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

**6.1.3 MACHINE LEARNING MODEL DEVELOPMENT**

It is crucial to reliably compare the performance of various machine learning algorithms, and it will become clear that scikit-learn in Python can be used to build a test harness for this purpose. You can apply this test harness as a model for your own machine learning issues and include additional and various methods to contrast. There will be variations in the efficiency attributes of each model. You can assess each model's potential accuracy on unobserved data by using resampling techniques like cross validation. It must be able to select one or two of the best models from the group of models you have developed using these approximations. It is a good idea to visualize new datasets using a variety of methods in order to view the data from various angles. The choice of models follows the same logic. In order to select the one or two that will be used for finalization, you should examine the expected accuracy of your machine learning algorithms in a variety of ways. Using various visualization techniques to display the average accuracy, variance, and other characteristics of the range of model accuracies is one way to achieve this.

**6.1.3.1 RANDOM FOREST ALGORITHM**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be applied to ML issues involving both classification and regression. It is founded on the idea of ensemble learning, which is a method of combining various classifiers to address complex issues and enhance model performance. Random Forest, as the name implies, is a classifier that uses a number of decision trees on different subsets of the provided dataset and averages them to increase the dataset's predictive accuracy.

The formula for calculating the feature importance is:

f= (eq 6.1)

**6.1.3.2 NAIVE BAYES ALGORITHM**

The Naive Bayes algorithm is a simple technique that makes predictions using the probabilities of each attribute pertaining to each class. It is the supervised learning strategy you would come up with if you wished to probabilistically model a predictive modeling issue. By assuming that each attribute's chance of belonging to a particular class value is unrelated to all other attributes, naive bayes simplifies the calculation of probabilities. Although this is a strong assumption, it leads to a quick and efficient approach. A statistical classification method built on the Theorem is called naive Bayes. One of the easiest supervised learning methods is this one. The fast, effective, and dependable algorithm is the naive Bayes classifier. On big datasets, naive Bayes classifiers perform quickly and accurately.

Given a features vector X=(x1,x2,…,xn) and a class variable y, Bayes Theorem states that:

(y|X) = (eq 6.2)

Thus, by conditional independence, we have:

(y|X)= (eq 6.3)

**6.1.3.3 GRADIENT BOOSTING**

One of the well-liked learning ensemble modeling methods called "boosting" is used to create strong classifiers from a variety of weak classifiers. It begins by creating a primary model using training data sets that are readily accessible, and then it finds any errors in the base model. A secondary model is constructed after the error has been located, and a third model is then added to the procedure. In this manner, adding additional models is continued until we have a complete collection of training data from which the model can accurately predict.In the annals of machine learning, GBM is also used as an ensemble technique to transform weak learners into strong ones. In this topic, "GBM in Machine Learning," we'll talk about boosting algorithms, gradient machine learning algorithms, the history of GBM, how it functions, different GBM terminologies, etc. But first, familiarize yourself with the boosting idea and different boosting algorithms used in machine learning.

At each iteration t, the model is updated as:

(eq 6.4)

where)is a *weak* learner chosen from some family of functions.

**6.1.3.4 DEPLOYMENT USING FLASK**

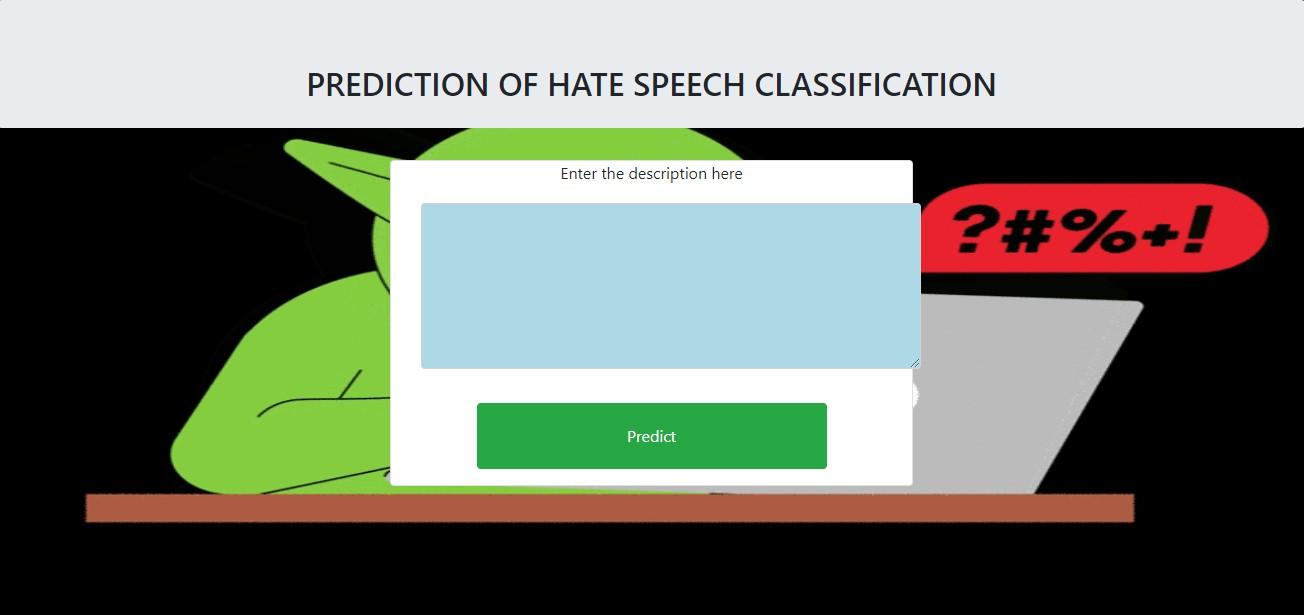
Flask is a micro web framework written in Python.It is classified as a micro-framework because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.However, flask supports extensions that can add application features as if they were implemented in flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.The most accurate algorithm is used to implement the model Gradient Boosting, one of three algorithms, is the strongest model. The Flask micro web framework is used to deploy this model.

**CHAPTER 7**

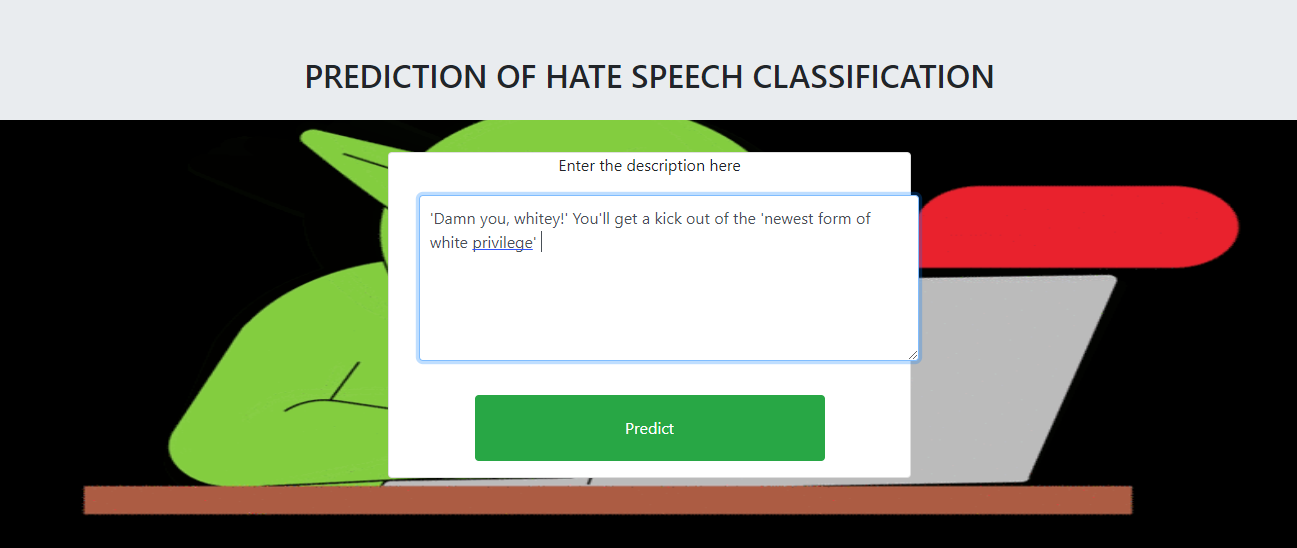
**PERFORMANCE ANALYSIS**

**CHAPTER-7**

**PERFORMANCE EVALUATION**

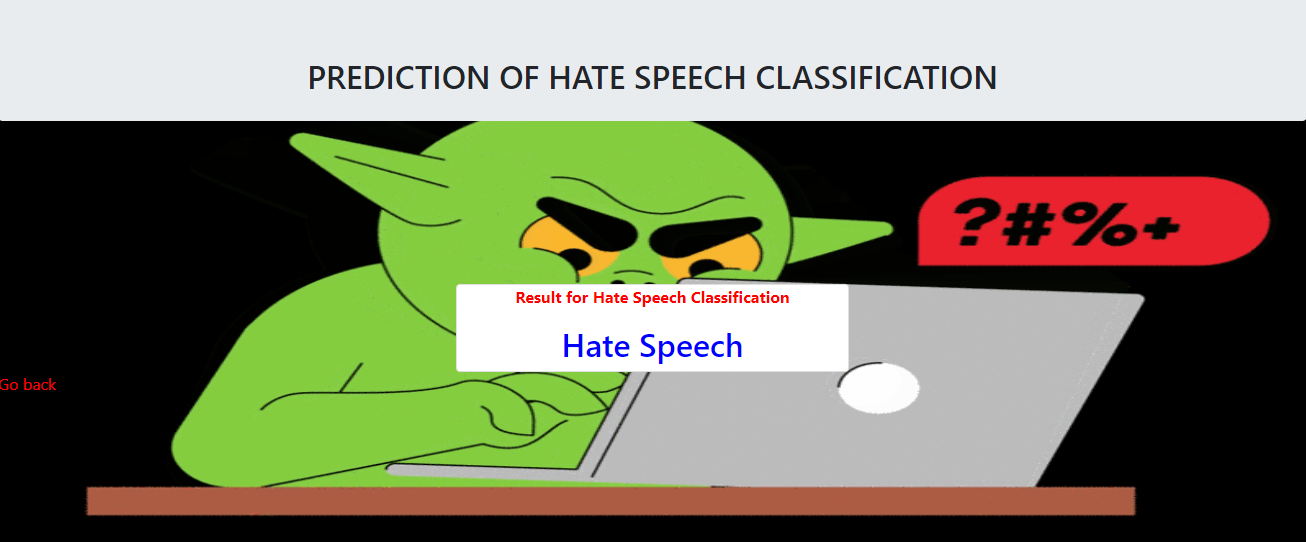
**7.1. RESULTS AND DISCUSSION**

**Figure 7.1 Screenshot of webpage for Hate Speech Classification**

 This screenshot shows the main page where the text is given as input and the comments are classified.

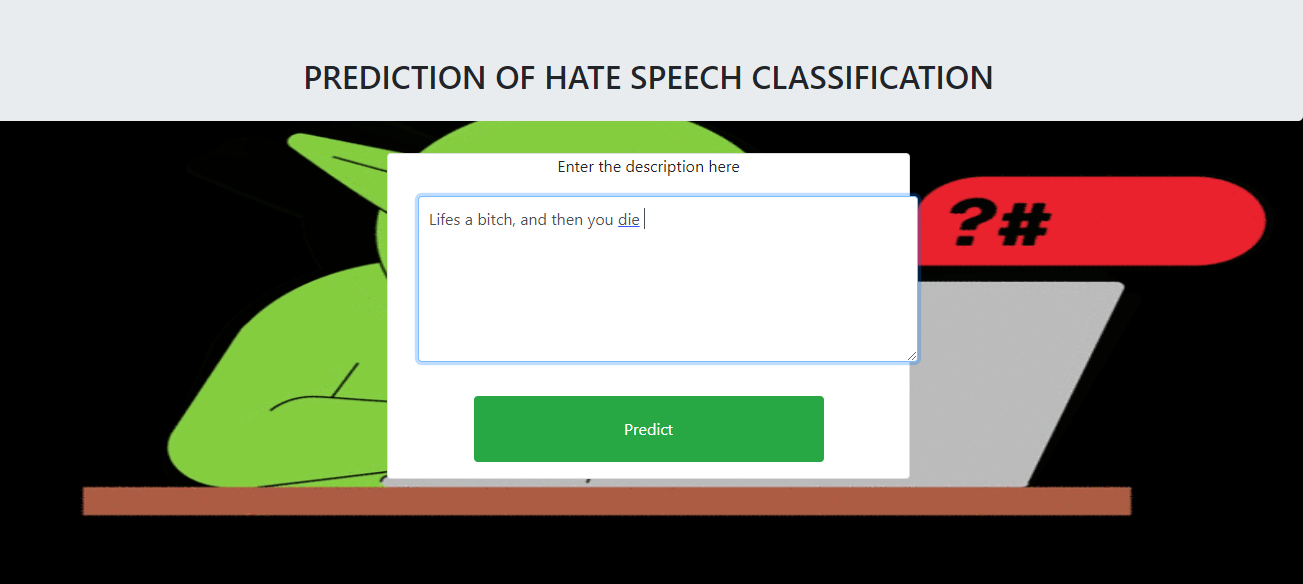
**Figure 7.2 Screenshot of entering the text for classification**

The above screenshot shows the way of entering text into the textbox by typing the comments.After entering the text into the textbox the “Predict” button should be clicked to display the type of the comment.



**Figure 7.3 Screenshot of the prediction of Hate Speech**

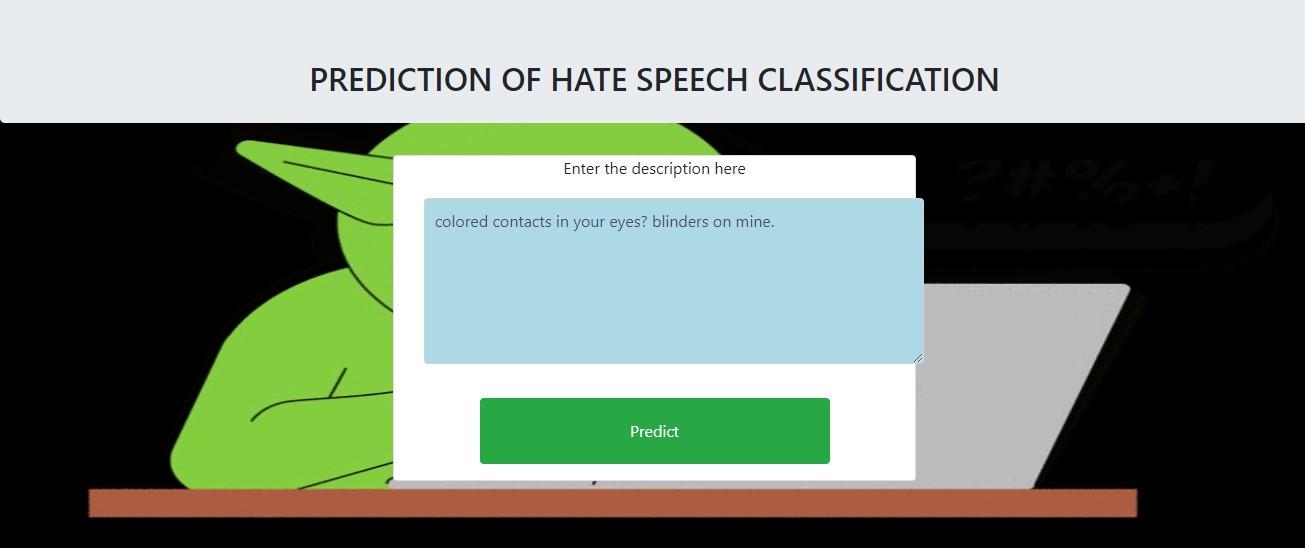
This figure shows how the comments will be classified and displayed in the webpage when the “Predict” button is clicked after entering the text.In this figure, it shows the type of comment is Hate Speech.

 **Figure 7.4 Screenshot of entering the text for classification**

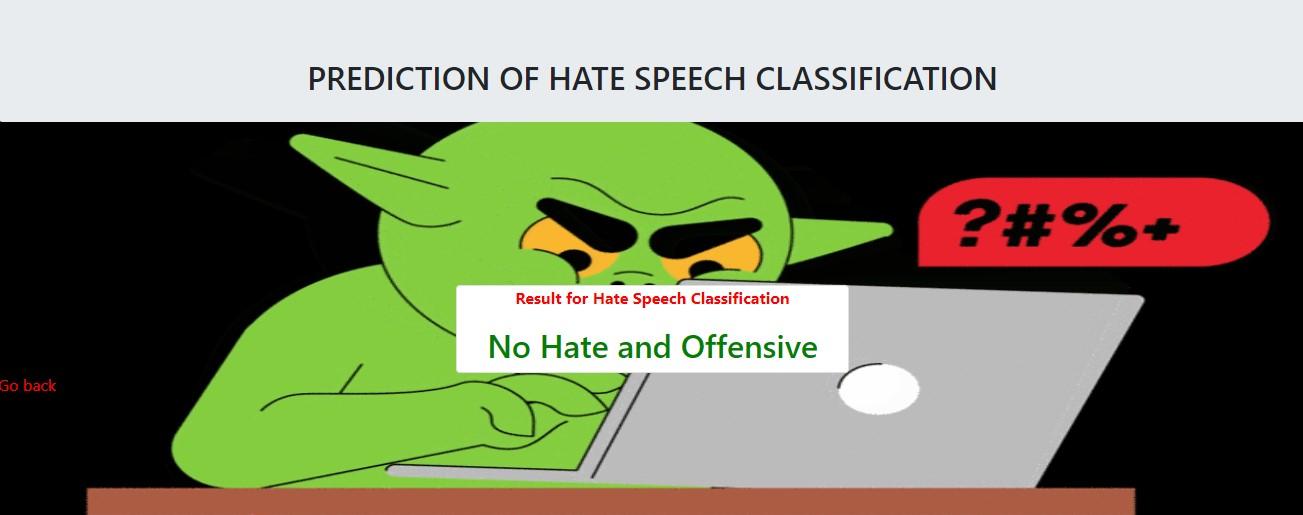
The above screenshot shows the way of entering text into the textbox by typing the comments.After entering the text into the textbox the “Predict” button should be clicked to display the type of the comment.

 **Figure 7.5 Screenshot of the prediction of Offensive language**

This figure shows how the comments will be classified and displayed in the webpage when the “Predict” button is clicked after entering the text.In this figure, it shows the type of comment is Offensive Language.

 **Figure 7.6 Screenshot of entering the text for classification**

The above screenshot shows the way of entering text into the textbox by typing the comments.After entering the text into the textbox the “Predict” button should be clicked to display the type of the comment.

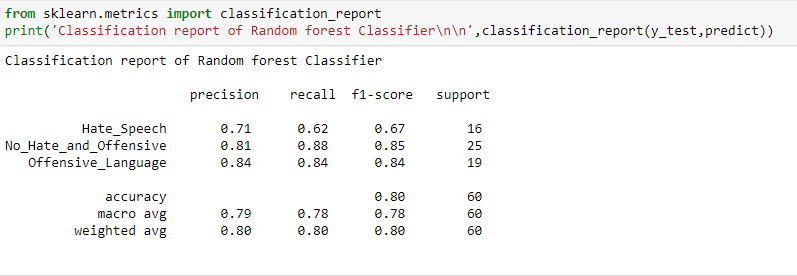
 **Figure 7.7 Screenshot of the prediction of No hate and offensive**

This figure shows how the comments will be classified and displayed in the webpage when the “Predict” button is clicked after entering the text.In this figure, it shows the type of comment is No Hate and Offensive.

**7.2 COMPARATIVE ANALYSIS**

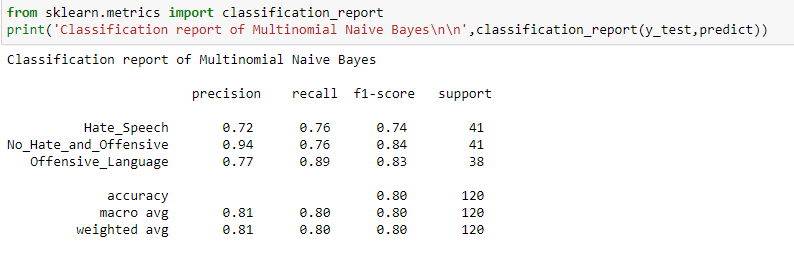
**CLASSIFICATION REPORT**

It is one of the performance evaluation metrics of a classification-based machine learning model. It displays the model’s precision, recall, F1 score and support. It provides a better understanding of the overall performance of our trained model. To understand the classification report of a machine learning model, we need to know all of the metrics displayed in the report.Precision is defined as the ratio of true positives to the sum of true and false positives.Recall is defined as the ratio of true positives to the sum of true positives and false negatives.The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.Support is the number of actual occurrences of the class in the dataset. It doesn’t vary between models, it just diagnoses the performance evaluation process.

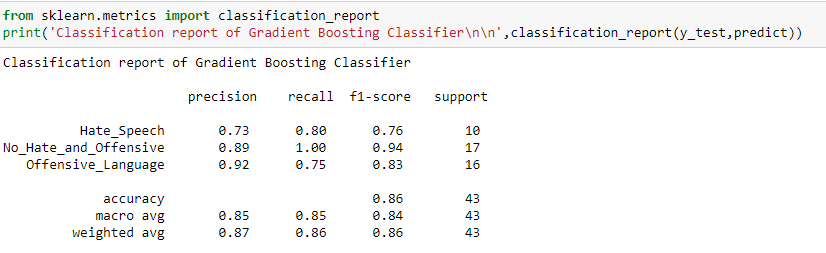


**Figure 7.8 Screenshot of classification report of Random forest algorithm**

The above figure shows the precision, recall, f1-score and support value for Hate\_Speech, No\_Hate\_and\_Offensive and Offensive Language obtained by training the model using Random Forest algorithm.

 **Figure 7.9 Screenshot of classification report of Naïve Bayes Algorithm**

The above figure shows the precision, recall, f1-score and support value for Hate\_Speech, No\_Hate\_and\_Offensive and Offensive Language obtained by training the model using Naive Bayes Algorithm.

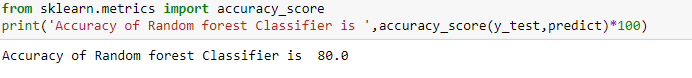
****

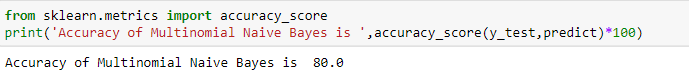
**Figure 7.10 Screenshot of classification report of Gradient Boosting algorithm**

The above figure shows the precision, recall, f1-score and support value for Hate\_Speech, No\_Hate\_and\_Offensive and Offensive Language obtained by training the model using Gradient Boosting Algorithm.

**ACCURACY OF ALGORITHMS**

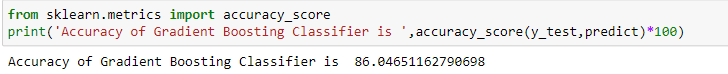
Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.Gradient Boosting Algorithm yields the highest accuracy compared to the other algorithms. So the project deployment uses the Gradient Boosting Algorithm.

**Figure 7.11 Screenshot of accuracy of Random Forest algorithm**

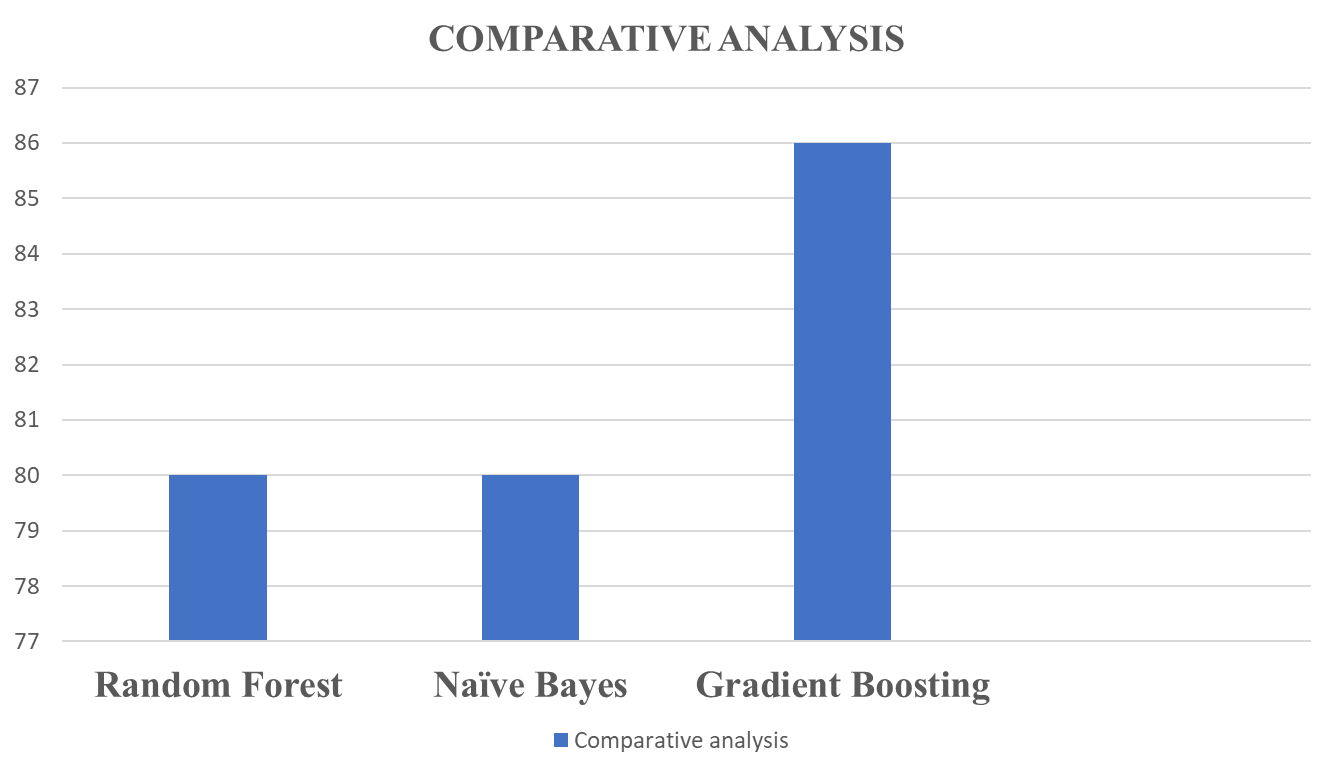
The above figure shows the accuracy obtained by training the model using Random Forest algorithm. The accuracy obtained is, that is Accuracy of Random forest Classifier is 80.0%.

**Figure 7.12 Screenshot of accuracy of Naive bayes algorithm**

The above figure shows the accuracy obtained by training the model using Naive Bayes algorithm. The accuracy obtained is, that is, Accuracy of Multinomial Naive Bayes is 80.0%.

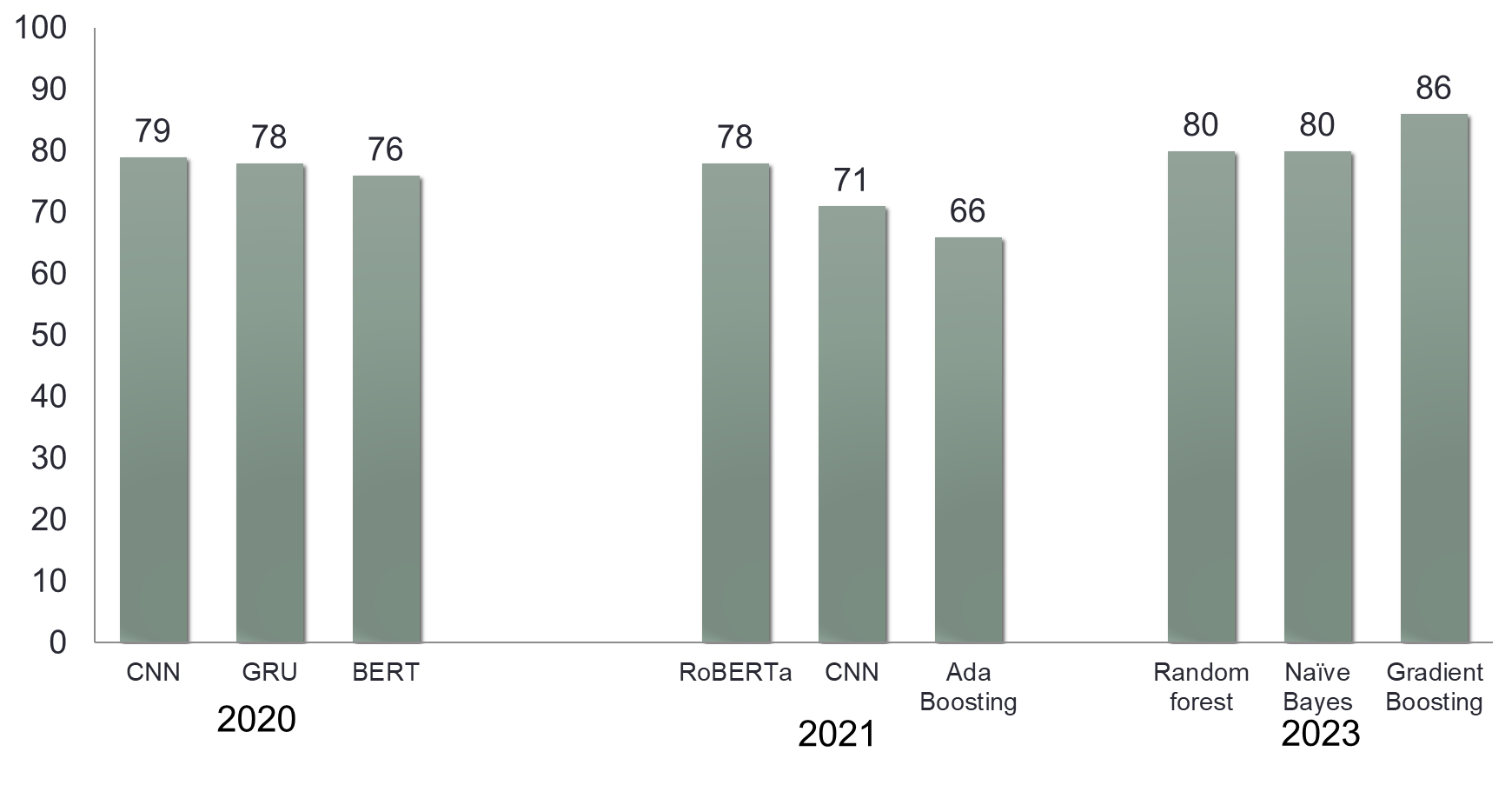
**Figure 7.13 Screenshot of accuracy of Gradient Boosting algorithm**

The above figure shows the accuracy obtained by training the model using Gradient Boosting algorithm. The accuracy obtained is, that is, Accuracy of Gradient Boosting Classifier is 86.04%.

 **Figure 7.14 Comparison of three model accuracy**

The above graph shows the comparison between three algorithms of proposed work. From the above, it can be said that “Gradient Boosting Algorithm” yields highest accuracy compared to other two.Now the project deployment uses the Gradient Boosting Algorithm which yields the highest accuracy.

**Comparison of accuracies in hate speech classification**



**Figure 7.15 Comparison of accuracies in hate speech classification**

Based on the above performance analysis, it is inferred that Gradient Boosting Algorithm yields highest accuracy. In 2020 research papers, the algorithms like CNN, GRU, BERT yielded 79%, 78%,76% accuracy respectively. In 2021 research papers, the algorithms like RoBERTa, CNN, Ada Boosting yielded 78%, 71%,66% accuracy respectively. From this, it is concluded that the proposed work has yielded higher accuracies compared to it’s related work.

**CHAPTER 8**

**CONCLUSION**

**CHAPTER-8**

**CONCLUSION**

**8.1. CONCLUSION AND FUTURE ENHANCEMENTS**

As the world moves more and more toward digital technology, everyone now has access to the internet and can publish whatever they want, making the control of hate speech very difficult. Therefore, there is a higher likelihood that individuals will be misled. Machine learning is typically designed to handle these types of complex tasks because it requires more time to manually analyze these types of data. By using previous data, making them comprehend patterns, and improving the accuracy of the model by changing parameters, machine learning can be used to classify speech as hate speech or not. The model is then used as the classification model. The optimal model can be used for classification by comparing various algorithms. The Gradient Boosting algorithm is determined to provide the best accuracy of the three algorithms. As a result, the project's implementation used the Gradient boosting algorithm. The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out. application can help to find the Prediction of hate speech.

**APPENDICES**

**APPENDIX 1**

**SAMPLE DATASET**

**count**: (Integer) number of users who coded each tweet (min is 3, sometimes more users coded a tweet when judgments were determined to be unreliable

**hate\_speech\_annotation**: (Integer) number of users who judged the tweet to be hate speech,

**offensive\_language\_annotation**: (Integer) number of users who judged the tweet to be offensive

**neither\_annotation**: (Integer) number of users who judged the tweet to be neither offensive nor non-offensive

**label**:(Class Label) class label for majority of CF users (0: 'hate-speech', 1: 'offensive-language' or 2: 'neither'),

**tweet**: (string)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Row ID | Count | Hate  speech | Offensive  language | Neither | Class | Tweet |
| 0 | 3 | 0 | 0 | 3 | 2 | As a woman you shouldn't complain about cleaning up your house |
| 1 | 3 | 0 | 3 | 0 | 1 | boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place |
| 2 | 3 | 0 | 3 | 0 | 1 | You ever fuck a bitch and she start to cry? You be confused as shit |
| 3 | 3 | 0 | 2 | 1 | 1 | @viva\_basedshe look like a tranny |
| 4 | 6 | 0 | 6 | 0 | 1 | The shit you hear about me might be true |
| 5 | 3 | 1 | 2 | 0 | 1 | The shit just blows me.claim you so faithful and down for somebody |
| 6 | 3 | 0 | 3 | 0 | 1 | I can not just sit up and HATE on another bitch .. I got too much shit going on!" |
| 7 | 3 | 0 | 3 | 0 | 1 | cause I'm tired of you big bitches coming for us skinny girls!!&#8221; |
| 8 | 3 | 0 | 3 | 0 | 1 | " &amp; you might not get ya bitch back &amp; thats that " |
| 9 | 3 | 1 | 2 | 0 | 1 | " @rhythmixx\_ :hobbies include: fighting Mariam"bitch |
| 10 | 3 | 0 | 3 | 0 | 1 | " Keeks is a bitch she curves everyone " lol I walked into a conversation like this. Smh |
| 11 | 3 | 0 | 3 | 0 | 1 | " Murda Gang bitch its Gang Land " |
| 12 | 3 | 0 | 2 | 1 | 1 | " So hoes that smoke are losers ? " yea ... go on |
| 13 | 3 | 0 | 3 | 0 | 1 | " bad bitches is the only thing that i like " |
| 14 | 3 | 1 | 2 | 0 | 1 | " bitch get up off me " |
| 15 | 3 | 0 | 3 | 0 | 1 | " bitch nigga miss me with it " |
| 16 | 3 | 0 | 3 | 0 | 1 | " bitch plz whatever " |
| 17 | 3 | 1 | 2 | 0 | 1 | " bitch who do you love " |
| 18 | 3 | 0 | 3 | 0 | 1 | " bitches get cut off everyday B " |
| 19 | 3 | 0 | 3 | 0 | 1 | " black bottle &amp; a bad bitch " |
| 20 | 3 | 0 | 3 | 0 | 1 | " broke bitch cant tell me nothing " |
| 21 | 3 | 0 | 3 | 0 | 1 | " cancel that bitch like Nino " |
| 22 | 3 | 0 | 3 | 0 | 1 | " cant you see these hoes wont change " |
| 23 | 3 | 0 | 3 | 0 | 1 | " fuck no that bitch dont even suck dick " the Kermit videos bout to fuck IG up |
| 24 | 3 | 0 | 3 | 0 | 1 | " got ya bitch tip toeing on my hardwood floors " |
| 25 | 3 | 0 | 2 | 1 | 1 | " her pussy lips like Heaven doors " &#128524; |
| 26 | 3 | 0 | 3 | 0 | 1 | " hoe what its hitting for " |
| 27 | 3 | 0 | 3 | 0 | 1 | " i met that pussy on Ocean Dr . i gave that pussy a pill " &#128524; |
| 28 | 3 | 0 | 3 | 0 | 1 | " i need a trippy bitch who fuck on Hennessy " |
| 29 | 3 | 0 | 3 | 0 | 1 | " i spend my money how i want bitch its my business " |
| 30 | 3 | 0 | 3 | 0 | 1 | " i txt my old bitch my new bitch pussy wetter " |
| 31 | 3 | 0 | 3 | 0 | 1 | " i'd say im back to the old me but my old bitches would get excited " &#128524; |
| 32 | 3 | 0 | 3 | 0 | 1 | " if you aint bout that Murder Game pussy nigga shut up " |
| 33 | 3 | 0 | 3 | 0 | 1 | " if you're toes ain't done you pussy stinks " |
| 34 | 3 | 0 | 3 | 0 | 1 | " im done with bitter bitches its a wrap for that . if you a angry bird theres a app for that " |
| 35 | 3 | 0 | 3 | 0 | 1 | " is that ya bitch " |
| 36 | 3 | 0 | 3 | 0 | 1 | " it aint nothing to cut a bitch off " |
| 37 | 3 | 0 | 3 | 0 | 1 | " jus meet son now he ya mane ass bitches " #Shots |
| 38 | 3 | 0 | 2 | 1 | 1 | " lames crying over hoes thats tears of a clown " |
| 39 | 3 | 0 | 3 | 0 | 1 | " like Snoop said in 94 we dont love these hoes " |
| 40 | 3 | 0 | 1 | 2 | 2 | " momma said no pussy cats inside my doghouse " |
| 41 | 3 | 0 | 3 | 0 | 1 | " most hated but the hoes favorite " #2MW #SevenOne # http://t.co/BMdSVMc3rC |
| 42 | 3 | 0 | 3 | 0 | 1 | " nice girls bad, make me get naughty. Bad yello hoe, real nice body. Down south chick, like em real thick" |
| 43 | 3 | 0 | 3 | 0 | 1 | " pimps up pimps up hoes down " Future voice |
| 44 | 3 | 0 | 3 | 0 | 1 | " post a picture of that pussy get 200 likes " |
| 45 | 3 | 0 | 3 | 0 | 1 | " pussy is a powerful drug " &#128517; |
| 46 | 3 | 0 | 3 | 0 | 1 | " quick piece of pussy call it a drive by " |
| 47 | 3 | 0 | 3 | 0 | 1 | " running round here like some brand new pussy thats bout to get fucked " |
| 48 | 3 | 0 | 3 | 0 | 1 | " these bitches even worst they'll send them guys for you " |
| 49 | 3 | 1 | 2 | 0 | 1 | " these hoes like niggas that spend money not talk bout it " |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 50 | 3 | 1 | 2 | 0 | 1 | " we dont trust these niggas all these bitches " |
| 51 | 3 | 0 | 3 | 0 | 1 | " yall niggas b cuffing hoes cause yall aint never have bitches |
| 52 | 3 | 0 | 3 | 0 | 1 | "All da bitches I cut off pregnant or bound to be" |
| 53 | 3 | 0 | 3 | 0 | 1 | " young Pill Chamberlain these bitches love my music " |
| 54 | 3 | 0 | 2 | 1 | 1 | Trump probably cleaner than most of these twitter hoes |
| 55 | 3 | 0 | 3 | 0 | 1 | Gay niggas couldn't wait to act like bitches tonight |
| 56 | 3 | 0 | 3 | 0 | 1 | “And ima steal his cat n replace it with a pitbull" |
| 57 | 3 | 0 | 3 | 0 | 1 | "&amp; he's gone always be a hoe" |
| 58 | 3 | 0 | 3 | 0 | 1 | "...Son of a bitch took my Tic Tacs." |
| 59 | 3 | 0 | 3 | 0 | 1 | "All I wanna do is get money and fuck model bitches!" |
| 60 | 3 | 0 | 2 | 1 | 1 | "I probably wouldnt mind school as much as teachers” |
| 61 | 3 | 0 | 3 | 0 | 1 | "I'm a fucking fag they said" |
| 62 | 3 | 0 | 3 | 0 | 1 | "Females think dating a pussy is cute now? " |
| 63 | 3 | 0 | 0 | 3 | 2 | " SimplyAddictedToGuys woof woof hot scally lad” |
| 64 | 3 | 0 | 3 | 0 | 1 | "Have ya ever asked your bitch for other bitches " Yes |
| 65 | 3 | 0 | 3 | 0 | 1 | " How u gone bring ur side bitch to a game” |
| 66 | 3 | 0 | 1 | 2 | 2 | " woof woof and hot soles” |
| 67 | 3 | 0 | 1 | 2 | 2 | " Lemmie eat a Oreo &amp; do these dishes." |
| 68 | 3 | 0 | 3 | 0 | 1 | " bitch fuck u maybe you'll get better.“ |
| 69 | 3 | 1 | 2 | 0 | 1 | " Fuck Red Malone man bitch ass niggah" |
| 70 | 3 | 0 | 0 | 3 | 2 | "Why the eggplant emoji doe?" |
| 71 | 3 | 0 | 3 | 0 | 1 | "Hennessey Venom that's one sexy bitch” |
| 72 | 3 | 0 | 3 | 0 | 1 | "Him seh me pussy wetter then a shower curtain “ |
| 73 | 3 | 0 | 3 | 0 | 1 | need to see that pic I just posted the most racist shit" |
| 74 | 3 | 1 | 2 | 0 | 1 | Yo fuck skateboarding, all y'all some wood pushing man |
| 75 | 3 | 0 | 1 | 2 | 2 | baby monkey bathtime http://t.co/7KPWA |
| 76 | 3 | 0 | 3 | 0 | 1 | "@BaylaaGottaBody; I ain't shit ." Damn Skippy lol |
| 77 | 3 | 1 | 2 | 0 | 1 | "@BeEasyJrizzy: u ever kill a ant on the sidewalk |
| 78 | 3 | 0 | 3 | 0 | 1 | "@BeenBasedB: 0 rings 0 mvps 0 bitches lol |
| 79 | 3 | 1 | 2 | 0 | 1 | "@BeenFLYnSolo: ppl talk bad about the ghetto/hood!" |
| 80 | 9 | 0 | 7 | 2 | 1 | The facts on tattoos.he's a pussy with not tattooing them |
| 81 | 3 | 0 | 3 | 0 | 1 | " Bitches be wanting to act like niggas so bad . |
| 82 | 3 | 0 | 3 | 0 | 1 | "@BitchJones92: Get worshiping bitch! " woof woof |
| 83 | 3 | 1 | 2 | 0 | 1 | "@ Wow. RT I'm lost. Are those buttcheek piercings? |
| 84 | 3 | 0 | 3 | 0 | 1 | " Ok, sis." She'd rather be a broke bitch? it works for her |
| 85 | 3 | 2 | 1 | 0 | 0 | "@Blackman38Tide: queer" gaywad |
| 87 | 3 | 0 | 3 | 0 | 1 | "faggot read my tweets after dat k" it wasn't even funny lol |
| 88 | 3 | 0 | 3 | 0 | 1 | " This bitch was so ungrateful " fr ..... LULWHORE |
| 89 | 3 | 0 | 3 | 0 | 1 | "@CASHandBOOBIES: I been kidnapped yo bitch" |
| 90 | 3 | 3 | 0 | 0 | 0 | " hes a beaner smh you can tell hes a mexican |
| 91 | 3 | 1 | 2 | 0 | 1 | "@CCobey: happy birthday nigs" Thanks you |
| 92 | 6 | 1 | 5 | 0 | 1 | " when ur teacher tells u that u have homework " |
| 93 | 3 | 1 | 2 | 0 | 1 | "@CaelanG15: " that nigga was eating that hoe lol |
| 94 | 3 | 0 | 3 | 0 | 1 | " What would y'all lil ugly bald headed bitches do?" |
| 95 | 3 | 1 | 2 | 0 | 1 | " Leafs better win this damn game so I can go riot and shit |
| 96 | 3 | 0 | 3 | 0 | 1 | "Going back to school sucks more dick than the hoes." |
| 97 | 3 | 0 | 3 | 0 | 1 | On my way to fuck yo bitch " me as a 9 year old |
| 98 | 3 | 0 | 3 | 0 | 1 | "@CeleyNichole: how come you never bring me food" |
| 99 | 3 | 0 | 3 | 0 | 1 | " If Richnow doesn't show up with hella tinder hoes " |
| 100 | 3 | 1 | 2 | 0 | 1 | How bout them Cowboys!!!!" Shutup pussy |

**Table A1.1 Sample Dataset for hate speech classification**

**APPENDIX 2**

**SAMPLE CODING**

**MODULE-1**

**DATA PREPROCESSING**

**#import library packages**

import pandas as p

import numpy as n

import warnings

warnings.filterwarnings("ignore")

**#Load given dataset**

data = p.read\_csv('data.csv')

data.head()

data.shape

df=data.dropna()

df.head()

df.shape

df.columns

df.describe()

df.head()

**# checking datatype and information about dataset**

df.info()

**# checking for duplicate data**

df.duplicated()

**# find sum of duplicated data**

sum(df.duplicated())

df.isnull().sum()

df.columns

df['count'].unique()

df['hate\_speech'].unique()

df['offensive\_language'].unique()

df['neither'].unique()

df['class'].value\_counts()

from sklearn.preprocessing import LabelEncoder

var=['Unnamed: 0', 'count', 'hate\_speech', 'offensive\_language', 'neither',

'class', 'tweet']

le=LabelEncoder()

for i in var:

df[i]=le.fit\_transform(df[i].astype(str))

df.head()

**MODULE 2**

**DATA VISUALIZATION**

#import library packages

import pandas as p

import numpy as n

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

#Load given dataset

data = p.read\_csv('data.csv')

data.head()

df.columns

**# plotting graph for distribution**

import matplotlib.pyplot as plt

import seaborn as sns

sns.countplot(x='class',data=df)

df.loc[:,'class'].value\_counts()

plt.title('HATE SPEECH')

df['class'].unique()

import nltk

nltk.download('stopwords')

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

import re

import string

**# remove whitespaces**

df['tweet']=df['tweet'].str.strip()

**# lowercase the text**

df['tweet'] = df['tweet'].str.lower()

**#remove punctuation**

punc = string.punctuation

table = str.maketrans('','',punc)

df['tweet']=df['tweet'].apply(lambda x: x.translate(table))

**# tokenizing each message**

df['word\_tokens']=df.apply(lambda x: x['tweet'].split(' '),axis=1)

**# removing stopwords**

df['cleaned\_text'] = df.apply(lambda x: [word for word in x['word\_tokens'] if word not in stopwords.words('english')],axis=1)

**# stemming**

ps = PorterStemmer()

df['stemmed']= df.apply(lambda x: [ps.stem(word) for word in x['cleaned\_text']],axis=1)

**# remove single letter words**

df['final\_text'] = df.apply(lambda x: ' '.join([word for word in x['stemmed'] if len(word)>1]),axis=1)

**# Now we'll create a vocabulary for the training set with word count**

from collections import defaultdict

vocab=defaultdict(int)

for text in df['final\_text'].values:

for elem in text.split(' '):

vocab[elem]+=1

print(vocab)

**# divide the set in training and test**

from sklearn.model\_selection import train\_test\_split

X,X\_test,y,y\_test=train\_test\_split(df.loc[:,'cleaned\_text':],df['class'],test\_size=0.2)

**#pip install wordcloud**

from wordcloud import WordCloud

hate\_speech\_text=''.join(X.loc[y==0,'final\_text'].values)

hate\_wordcloud=WordCloud(background\_color='white',max\_words=1000,width=800,height=800).generate(hate\_speech\_text)

Offensive\_text=''.join(X.loc[y==1,'final\_text'].values)

Offensive\_wordcloud=WordCloud(background\_color='black',max\_words=1000,width=800,height=800).generate(Offensive\_text)

No\_Hate\_Offensive\_text=''.join(X.loc[y==2,'final\_text'].values)

No\_hate\_wordcloud=WordCloud(background\_color='pink',max\_words=1000,width=800,height=800).generate(No\_Hate\_Offensive\_text)

plt.figure(figsize=[30,50])

plt.subplot(1,3,1)

plt.imshow(hate\_wordcloud,interpolation='bilinear')

plt.title('')

plt.axis('off')

plt.subplot(1,3,2)

plt.imshow(Offensive\_wordcloud,interpolation='bilinear')

plt.axis('off')

plt.title('')

plt.subplot(1,3,3)

plt.imshow(No\_hate\_wordcloud,interpolation='bilinear')

plt.axis('off')

plt.title('')

**MODULE 3**

**RANDOM FOREST ALGORITHM**

**#import library packages**

import pandas as p

import numpy as n

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

**#Load given dataset**

data = p.read\_csv('data.csv')

data.head()

df=data.dropna()

df

del df['Unnamed: 0']

del df['count']

del df['hate\_speech']

del df['offensive\_language']

del df['neither']

df.head()

df.tail()

df.shape

df['class']=df['class'].map({0:'Hate\_Speech',1:'Offensive\_Language',2:'No\_Hate\_and\_Offensive'})

print(df.head())

**# preprocessing, split test and dataset, split response variable**

X=df.drop(labels='class',axis=1)

**# Response variable**

y=df.loc[:,'class']

import imblearn

from imblearn.under\_sampling import RandomUnderSampler

from collections import Counter

ros=RandomUnderSampler(random\_state=42)

x\_ros,y\_ros=ros.fit\_resample(X,y)

print('OUR DATASET COUNT :',Counter(y))

print('OVER SAMPLING DATA COUNT :',Counter(y\_ros))

x\_ros

y\_ros

import re

import string

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

**# remove whitespaces**

x\_ros['tweet']=x\_ros['tweet'].str.strip()

**# lowercase the text**

df['tweet'] = df['tweet'].str.lower()

**#remove punctuation**

punc = string.punctuation

table = str.maketrans('','',punc)

df['tweet']=df['tweet'].apply(lambda x: x.translate(table))

**# tokenizing each message**

df['word\_tokens']=df.apply(lambda x: x['tweet'].split(' '),axis=1)

**# removing stopwords**

df['cleaned\_text'] = df.apply(lambda x: [word for word in x['word\_tokens'] if word not in stopwords.words('english')],axis=1)

**# stemming**

ps = PorterStemmer()

df['stemmed']= df.apply(lambda x: [ps.stem(word) for word in x['cleaned\_text']],axis=1)

**# remove single letter words**

df['final\_text'] = df.apply(lambda x: ' '.join([word for word in x['stemmed'] if len(word)>1]),axis=1)

x\_ros[0:10]

y\_ros[0:10]

X=n.array(x\_ros['tweet'])

y=n.array(y\_ros)

X[0:10]

y[0:10]

from sklearn.feature\_extraction.text import CountVectorizer

cv=CountVectorizer()

cv\_X=cv.fit\_transform(X) **# fit the data**

from sklearn.ensemble import RandomForestClassifier

RFC=RandomForestClassifier(n\_estimators=2000)

RFC.fit(X\_train,y\_train)

predict=RFC.predict(X\_test)

from sklearn.metrics import accuracy\_score

print('Accuracy of Random forest Classifier is ',accuracy\_score(y\_test,predict)\*100)

from sklearn.metrics import confusion\_matrix

print('confusion matrix of Random forest Classifier\n\n',confusion\_matrix(y\_test,predict))

from sklearn.metrics import classification\_report

print('Classification report of Random forest Classifier\n\n',classification\_report(y\_test,predict))

**Saving Model**

import joblib

joblib.dump(RFC, 'FC.pkl')

joblib.dump(cv\_X, 'cv.pkl')

**MODULE-4**

**NAIVE BAYES ALGORITHM**

**#import library packages**

import pandas as p

import numpy as n

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

**#Load given dataset**

data = p.read\_csv('data.csv')

data.head()

df=data.dropna()

df

del df['Unnamed: 0']

del df['count']

del df['hate\_speech']

del df['offensive\_language']

del df['neither']

df.head()

df.tail()

df.shape

df['class']=df['class'].map({0:'Hate\_Speech',1:'Offensive\_Language',2:'No\_Hate\_and\_Offensive'})

print(df.head())

**# preprocessing, split test and dataset, split response variable**

X=df.drop(labels='class',axis=1)

**# Response variable**

y=df.loc[:,'class']

import imblearn

from imblearn.under\_sampling import RandomUnderSampler

from collections import Counter

ros=RandomUnderSampler(random\_state=42)

x\_ros,y\_ros=ros.fit\_resample(X,y)

print('OUR DATASET COUNT :',Counter(y))

print('OVER SAMPLING DATA COUNT :',Counter(y\_ros))

x\_ros

y\_ros

import re

import string

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

**# remove whitespaces**

x\_ros['tweet']=x\_ros['tweet'].str.strip()

**# lowercase the text**

df['tweet'] = df['tweet'].str.lower()

**#remove punctuation**

punc = string.punctuation

table = str.maketrans('','',punc)

df['tweet']=df['tweet'].apply(lambda x: x.translate(table))

**# tokenizing each message**

df['word\_tokens']=df.apply(lambda x: x['tweet'].split(' '),axis=1)

**# removing stopwords**

df['cleaned\_text'] = df.apply(lambda x: [word for word in x['word\_tokens'] if word not in stopwords.words('english')],axis=1)

**# stemming**

ps = PorterStemmer()

df['stemmed']= df.apply(lambda x: [ps.stem(word) for word in x['cleaned\_text']],axis=1)

**# remove single letter words**

df['final\_text'] = df.apply(lambda x: ' '.join([word for word in x['stemmed'] if len(word)>1]),axis=1)

x\_ros[0:10]

y\_ros[0:10]

X=n.array(x\_ros['tweet'])

y=n.array(y\_ros)

X[0:10]

y[0:10]

from sklearn.feature\_extraction.text import CountVectorizer

cv=CountVectorizer()

cv\_X=cv.fit\_transform(X) **# fit the data**

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(cv\_X,y,test\_size=120, random\_state=42)

from sklearn.naive\_bayes import MultinomialNB

mnb=MultinomialNB(alpha = 8.0 ,fit\_prior=True)

mnb.fit(X\_train, y\_train)

predict=mnb.predict(X\_test)

from sklearn.metrics import accuracy\_score

print('Accuracy of Multinomial Naive Bayes is ',accuracy\_score(y\_test,predict)\*100)

from sklearn.metrics import confusion\_matrix

print('confusion matrix of Multinomial Naive Bayes\n\n',confusion\_matrix(y\_test,predict))

from sklearn.metrics import classification\_report

print('Classification report of Multinomial Naive Bayes\n\n',classification\_report(y\_test,predict))

**MODULE-5**

**GRADIENT BOOSTING**

**#import library packages**

import pandas as p

import numpy as n

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

**#Load given dataset**

data = p.read\_csv('data.csv')

data.head()

df=data.dropna()

df

del df['Unnamed: 0']

del df['count']

del df['hate\_speech']

del df['offensive\_language']

del df['neither']

df.head()

df.tail()

df.shape

df['class']=df['class'].map({0:'Hate\_Speech',1:'Offensive\_Language',2:'No\_Hate\_and\_Offensive'})

print(df.head())

**# preprocessing, split test and dataset, split response variable**

X=df.drop(labels='class',axis=1)

**# Response variable**

y=df.loc[:,'class']

import imblearn

from imblearn.under\_sampling import RandomUnderSampler

from collections import Counter

ros=RandomUnderSampler(random\_state=42)

x\_ros,y\_ros=ros.fit\_resample(X,y)

print('OUR DATASET COUNT :',Counter(y))

print('OVER SAMPLING DATA COUNT :',Counter(y\_ros))

x\_ros

y\_ros

import re

import string

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

**# remove whitespaces**

x\_ros['tweet']=x\_ros['tweet'].str.strip()

**# lowercase the text**

df['tweet'] = df['tweet'].str.lower()

**#remove punctuation**

punc = string.punctuation

table = str.maketrans('','',punc)

df['tweet']=df['tweet'].apply(lambda x: x.translate(table))

**# tokenizing each message**

df['word\_tokens']=df.apply(lambda x: x['tweet'].split(' '),axis=1)

**# removing stopwords**

df['cleaned\_text'] = df.apply(lambda x: [word for word in x['word\_tokens'] if word not in stopwords.words('english')],axis=1)

**# stemming**

ps = PorterStemmer()

df['stemmed']= df.apply(lambda x: [ps.stem(word) for word in x['cleaned\_text']],axis=1)

**# remove single letter words**

df['final\_text'] = df.apply(lambda x: ' '.join([word for word in x['stemmed'] if len(word)>1]),axis=1)

x\_ros[0:10]

y\_ros[0:10]

X=n.array(x\_ros['tweet'])

y=n.array(y\_ros)

X[0:10]

y[0:10]

from sklearn.feature\_extraction.text import CountVectorizer

cv=CountVectorizer()

cv\_X=cv.fit\_transform(X) **# fit the data**

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(cv\_X,y,test\_size=0.01,random\_state=6)

from sklearn.ensemble import GradientBoostingClassifier

gbc=GradientBoostingClassifier()

gbc.fit(X\_train,y\_train)

predict=gbc.predict(X\_test)

from sklearn.metrics import accuracy\_score

print('Accuracy of Gradient Boosting Classifier is ',accuracy\_score(y\_test,predict)\*100)

from sklearn.metrics import confusion\_matrix

print('confusion matrix of Gradient Boosting Classifier\n\n',confusion\_matrix(y\_test,predict))

from sklearn.metrics import classification\_report

print('Classification report of Gradient Boosting Classifier\n\n',classification\_report(y\_test,predict))

**Saving Model**

import joblib

joblib.dump(gbc, 'gbc.pkl')

joblib.dump(cv\_X, 'cv.pkl')

**MODULE-6**

**DEPLOYMENT**

* **home.html**

<!DOCTYPE html>

<html>

<head>

<title>Home</title>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/bootstrap.min.css') }}">

<style>

.back{background-image: url("{{ url\_for('static', filename='image/hate1.gif') }}");

background-repeat: no-repeat;

background-attachment: fixed;

background-size: 100% 100%;

}

.white{

color:white;

}

.space{

margin:10px 30px;

padding:10px 10px;

background: lightblue;

width:500px

}

.gap{

padding:10px 20px;

}

</style>

</head>

<body class="back">

<header class="jumbotron" style="height:100px;">

<div class="container">

<center>

<h2>PREDICTION OF HATE SPEECH CLASSIFICATION</h2>

</center>

</div>

</header>

<center>

<div class="card ml-container" style="width:40%" >

<form class="form-group" action="{{ url\_for('predict')}}" method="POST">

<label class="black" for="">Enter the description here</label>

<!-- <input type="text" name="comment"/> -->

<textarea name="message" class="space form-control" rows="6" cols="50"></textarea>

<br/>

<input type="submit" class="btn btn-success btn-block" style="width:350px;padding:20px" value="Predict">

</form></div></center>

</body>

</html>

* **result.html**

<!DOCTYPE html>

<html>

<head>

<title></title>

<link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='css/bootstrap.min.css') }}">

<style>

.back{

background-image: url("{{ url\_for('static', filename='image/hate1.gif') }}");

background-repeat: no-repeat;

background-attachment: fixed;

background-size: 100% 100%;

}

center{

padding-top:10%;

}

a{

color:red;

}

</style>

</head>

<body class="back">

<header class="jumbotron" style="height: 100px;">

<div class="container">

<h2 style="text-align:center">PREDICTION OF HATE SPEECH CLASSIFICATION</h2>

</div></header>

<center>

<div class="card" style="width:30%">

<p style="color:red;font-size:20;text-align: center;"><b>Result for Hate Speech Classification</b></p>

<div class="results">

{% if prediction[0] == 'Hate\_Speech' %}

<h2 style="color:blue;">Hate Speech</h2>

{% elif prediction[0] == 'Offensive\_Language' %}

<h2 style="color:red;">Offensive Language</h2>

{% elif prediction[0] == 'No\_Hate\_and\_Offensive' %}

<h2 style="color:green;">No Hate and Offensive</h2>

{% endif %}

</div>

</center>

</div>

<a href="{{ url\_for('home')}}">Go back</a>

</body></html>

**FLASK CODE**

from flask import Flask,render\_template,url\_for,request

import pandas as pd

import joblib

**# load the model from disk**

clf = joblib.load("gbc.pkl")

cv = joblib.load("cv.pkl")

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('home.html')

@app.route('/predict',methods=['POST'])

def predict():

if request.method == 'POST':

message = request.form['message']

data = [message]

vect = cv.transform(data).toarray()

my\_prediction = clf.predict(vect)

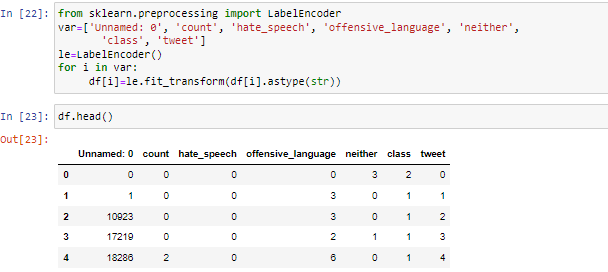
return render\_template('result.html',prediction = my\_prediction)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=False)

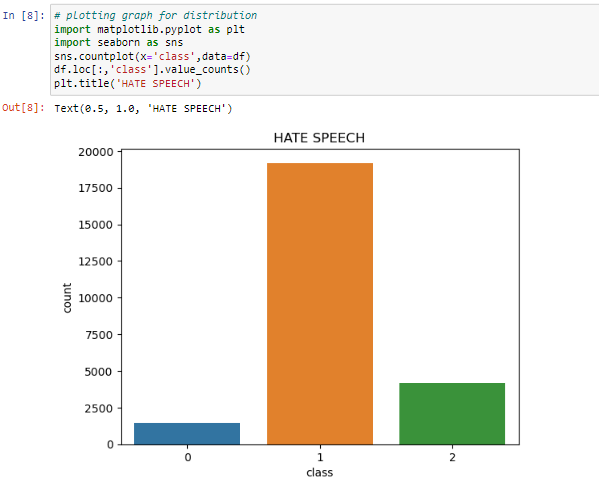
**APPENDIX 3**

**SAMPLE SCREENS**



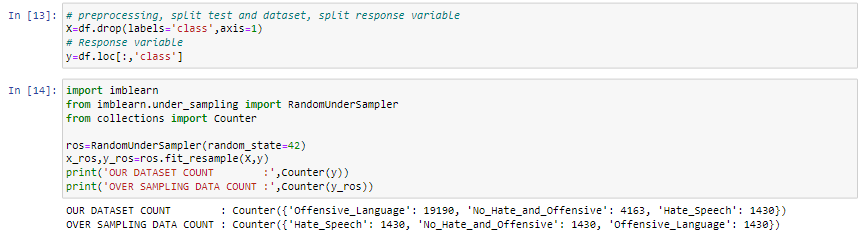
**Figure A3.1 Screenshot of Data Preprocessing Coding**

The above figure shows a part of coding in Data Pre-processing technique.In this data pre-processing process the raw dataset will be converted into clean dataset and making the dataset suitable for a machine learning model takes place.



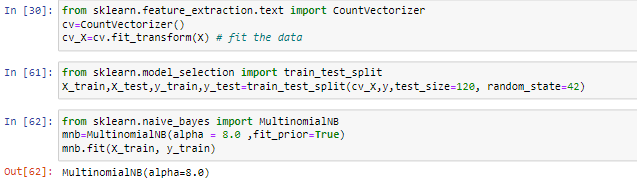
**Figure A3.2 Screenshot of Data Visualization coding**

This figure shows a part of Data Visualization coding which represents the data in the form of bar graph .Here, in the X-axis it represents the classes that is, 0, 1, 2 and in the Y-axis it represents the count values.



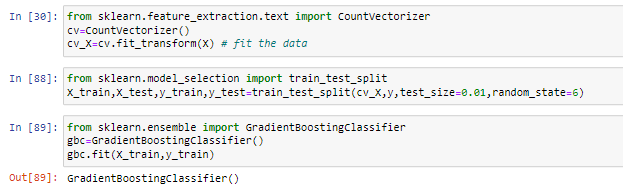
**Figure A3.3 Screenshot of Random forest algorithm coding**

This figure shows a part of Random forest algorithm coding.This algorithm combines the output of multiple decision trees to reach out a single output.Thus, instead of relying on one decision tree, the random forest takes the precision from each tree and based on the majority vote, it will predict the output.



**Figure A3.4 Screenshot of Naive Bayes algorithm coding**

This figure shows a part of Naive Bayes algorithm coding.This algorithm will assumes that each of the occurrence of a certain feature is independent of the occurrence of other feature and will make the predictions.



**Figure A3.5 Screenshot of Gradient Boosting algorithm coding**

This figure shows a part of Gradient Boosting algorithm coding.This algorithm starts with building a primary model from available training datasets then it identifies the errors present in the base model,this process of introducing more models is continued until the correct prediction is obtained.

**REFERENCES**

[1] Soumitra Ghosh , Asif Ekbal , Pushpak Bhattacharyya, Tista Saha, Alka Kumar, and Shikha Srivastava,”SEHC: A Benchmark Setup to Identify Online Hate Speech in English” in IEEE Transactions on Computational Social Systems, 2022

[2] Zewdie Mossie, Jenq-Haur Wang,”Vulnerable community identification using hate speech detection on social media” in Information Processing & Management 57 (3), 102087, 2020

[3] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, Animesh Mukherjee,” HateXplain “A Benchmark Dataset for Explainable Hate Speech Detection” in Proceedings of the AAAI Conference on Artificial Intelligence 35 (17), 14867-14875, 2021

[4]Kovács Pedro Alonso Rajkumar Saini,” Challenges of Hate Speech Detection in Social Media” in SN Computer Science 2, 1-15, 2021

[5] Amrutha, B R and Bindu, K R, “Detecting Hate Speech in Tweets Using Different Deep Neural Network Architectures in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 923-926, 2019

[6] Sindhu Abro1 , Sarang Shaikh2 , Zafar Ali ,” Automatic Hate Speech Detection using Machine Learning: A Comparative Study”in International Journal of Advanced Computer Science and Applications 11 (8), 2020

[7] Raghad Alshalan and Hend Al-Khalifa,” A Deep Learning Approach for Automatic Hate Speech Detection in the Saudi Twittersphere”in Applied Sciences 10 (23), 8614, 2020

[8] Varsha Pathak , Manish Joshi , Prasad A. Joshi , Monica Mundada , Tanmay Joshi,” Using Machine Learning for Detection of Hate Speech and Offensive Code-Mixed Social Media text “ in arXiv preprint arXiv:2102.09866, 2021

[9] Ching Seh Wu, Unnathi Bhandary,” Detection of Hate Speech in Videos Using Machine Learning” in 2020 International Conference on Computational Science and Computational Intelligence (CSCI), 585-590, 2020

[10] Georgios Rizos, Konstantin Hemker, Bjorn Schuller,” Augment to Prevent: Short-Text Data Augmentation in Deep Learning for Hate-Speech Classification” in Proceedings of the 28th ACM international conference on information and knowledge management, 991-1000, 2019

[11] Hajime Watanabe, Mondher Bouazizi, Tomoaki Ohtsuki ,” Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection”in IEEE access 6, 13825-13835, 2018

[12] Thomas Davidson,Dana Warmsle,Michael Macy, Ingmar Weber,” Automated Hate Speech Detection and the Problem of Offensive Language” in Proceedings of the international AAAI conference on web and social media 11 (1), 512-515, 2017