Gender Prediction and Hate Speech Detection with Textual Data using Machine Learning

A PROJECT REPORT

Submitted in the partial fulfilment of requirements to

RVR&JC COLLEGE OF ENGINEERING

For the award of the degree

B.Tech. in CSE

By

Juluru V Y H Lakshmi Narsitha (Y20CS069)

Katta Durga Bhavani (L21CS199)

Madhavath Kalyani Bai (Y20CS105)

May 2024

R.V.R. & J.C. COLLEGE OF ENGINEERING (Autonomous)

(Affiliated to Acharya Nagarjuna University)

Chandramoulipuram::Chowdavaram

GUNTUR – 522 019

R.V. R & J.C. COLLEGE OF ENGINEERING (Autonomous)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this Project Report titled "**Gender Prediction and Hate Speech Detection with Textual Data using Machine Learning**" is the study conducted **by Juluru V Y H Lakshmi Narsitha (Y20CS069), Katta Durga Bhavani (L21CS199), Madhavath Kalyani Bai (Y20CS105)** and submitted in partial fulfilment of the requirements for the award of the degree, B,Tech in Computer Science And Engineering, during the Academic Year **2023-2024.**

Dr.M.Sreelatha	**Dr. B. Vara Prasad Rao**	**Dr.M.Sreelatha**
Project Guide	Project In-charge	Prof. &Head, CSE

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without proper suggestions, guidance and environment. Combination of these three factors acts like backbone to our Project **"Gender Prediction and Hate Speech Detection with Textual Data using Machine Learning".**

We are very glad to express our special thanks to **Dr.M.Sreelatha,** Project Guide, who has inspired us to select this topic, and also for her valuable suggestions in completion of the project.

We are very thankful to **Dr. B. Vara Prasad Rao,** Project Incharge who extended his encouragement and assistance in ensuring the smooth progress of the project work.

We would like to express our profound gratitude to **Dr.M.Sreelatha**, Head of the Department of Computer Science and Engineering for her encouragement and support to carry out this Project successfully.

We are very much thankful to **Dr. Kolla Srinivas,** Principal, for having allowed us to conduct the study needed for the project.

Finally, we submit our heartfelt thanks to all the staff in the Department and to all our friends for their cooperation during the project work.

Juluru V Y H Lakshmi Narsitha (Y20CS069)

Katta Durga Bhavani (L21CS199)

Madhavath Kalyani Bai (Y20CS105)

ABSTRACT

This study investigates methods for predicting gender and detecting hateful language on Twitter, utilizing sophisticated computer tools to analyze user interactions. Gender prediction involves examining linguistic patterns within tweets and profiles to discern gender indicators. Additionally, the study explores the potential benefits of integrating diverse Twitter data, such as tweets and user profiles, to enhance gender prediction accuracy. Concurrently, it focuses on identifying hateful language in tweets, employing content analysis to detect mean or offensive words. Utilizing machine learning methodologies, the study evaluates whether tweets qualify as hate speech. Through rigorous testing of various computer methods like accuracy, the research seeks to identify effective approaches contributing to a deeper understanding of social media dynamics and strategies for fostering a more positive online environment.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	Abbreviation	Full form
1.	ML	Machine Learning
2.	NLP	Natural Language Processing
3.	SVM	Support Vector Machine
4.	LR	Logistic Regression
5.	RF	Random Forest
6.	NB	Naive Bayes
7.	XGB	eXtreme Gradient Boosting
8.	GPT	Generative Pre-trained Transformer
9.	Word2Vec	Word to Vector
10.	BoW	Bag of Words
11.	GloVe	Global Vectors

CHAPTER 1

INTRODUCTION

1.1 Background

In today's dynamic digital landscape, social media platforms like Twitter have become integral parts of our daily lives, serving as hubs for communication, expression, and interaction. Within this vibrant ecosystem, researchers are increasingly drawn to exploring two significant facets: gender prediction and hate speech detection. This study embarks on a journey to delve into these realms, leveraging Twitter data as a rich source of insights.

Gender prediction constitutes a focal point of inquiry, tapping into the nuances of language and communication patterns to discern the gender identity of users. By analyzing user-generated content, including tweets and profile descriptions, researchers aim to uncover textual cues and linguistic markers that may hint at an individual's gender identity. Through sophisticated text preprocessing techniques and machine learning algorithms, the study endeavors to predict gender accurately, shedding light on the complexities of identity representation in the online sphere.

Fig 1.1 Twitter-based Gender Classification

Concurrently, the study ventures into the challenging terrain of hate speech detection, a pressing concern in today's digital age. Hate speech, characterized by its derogatory, inflammatory, or discriminatory nature, poses significant risks to online communities, fostering toxicity and polarization. By focusing on Twitter data and employing robust classification methodologies, researchers seek to identify and classify instances of hate speech within the vast stream of user-generated content. This endeavor is crucial for mitigating the harmful impact of hate speech and fostering a safer and more inclusive online environment.

The research uses various techniques to analyze Twitter data. It aims to understand how gender is represented and how hate speech spreads on the platform. By using advanced tools, the study wants to uncover patterns in language and how people talk about identity online.

Furthermore, the study seeks to evaluate the different methodologies and algorithms employed in gender prediction and hate speech detection tasks. By assessing the performance and accuracy of diverse approaches, researchers can identify strengths, limitations, and areas for improvement within the field. This critical evaluation not only enhances the understanding of computational techniques but also informs the development of more robust and effective solutions for addressing gender-related issues and combating online hate speech.

This study embarks on a comprehensive exploration of gender prediction and hate speech detection within the dynamic landscape of social media. Through meticulous analysis of Twitter data and innovative computational methodologies, researchers aim to unravel the complexities of online identity representation and discourse, contributing valuable insights to the fields of gender studies, computational linguistics, and digital ethics.

1.2 Problem Definition

The primary objective is to explore two key aspects of Twitter: predicting gender and identifying hate speech. The aim is to achieve this by analyzing user-generated content, including tweets and profile descriptions.

To achieve this goal, the study will also look closely at individual tweets, aiming to enhance the accuracy of our predictions and the detection of hate speech. Through comprehensive analysis and validation using actual tweets, the study aims to provide a more holistic understanding of the effectiveness of different approaches in predicting gender and detecting hate speech on Twitter. By incorporating real-world examples, the research endeavors to offer practical insights and actionable recommendations for improving the accuracy.

Ultimately, the main goal is to gain deeper insights into these phenomena. By doing so, the objective is to have more positive and inclusive online environment on Twitter. It will make twitter as a safe and positive platform by closely examining these issues.

1.3 Significance of Work

The significance of this work extends beyond academic research to real-world implications for online communities. By focusing on gender prediction and hate speech detection on Twitter, we address critical issues that impact the well-being and safety of individuals engaging in online discourse. Gender prediction algorithms can offer insights into how users represent themselves online and how they are perceived by others. This understanding is crucial for promoting diversity and inclusivity in digital spaces, where individuals may face discrimination or marginalization based on their gender identity.

Moreover, hate speech detection plays a pivotal role in creating a more respectful and constructive online environment. Hate speech can perpetuate harmful stereotypes, incite violence, and undermine social cohesion. By developing effective methods to identify and mitigate hate speech that contributes to fostering healthier online interactions and reducing the spread of toxic content. This is particularly relevant given the increasing prevalence of hate speech on social media platforms and its detrimental impact on users' mental health and well-being.

Furthermore, our research has practical implications for platform moderation and content moderation policies. Social media companies rely on automated systems to detect and remove harmful content, including hate speech. By providing insights into the prevalence and nature of hate speech on Twitter, our work can inform the development and optimization of content moderation algorithms. This, in turn, can help platforms better enforce community guidelines and protect users from harassment, bullying, and other forms of online abuse.

Additionally, our study contributes to the growing body of research on computational social science and digital ethics. By leveraging computational techniques to analyze large-scale social media data gained valuable insights into online behavior and discourse dynamics. This interdisciplinary approach allows us to explore complex societal issues through the lens of technology, bridging the gap between social science research and computer science innovation.

Moreover, our work has implications for public policy and legislative efforts aimed at addressing online harms. Hate speech legislation varies widely across jurisdictions, and there is ongoing debate about the balance between freedom of expression and the prevention of hate speech. By providing empirical evidence of hate speech prevalence and characteristics on Twitter, our research can inform policymakers and lawmakers in their efforts to develop evidence-based policies and regulations to combat online hate speech effectively.

1.4 Objectives

1.Model Building for Gender Prediction:

To predict gender on Twitter, we gather labeled data including tweets and profiles. Then used to train different machine learning models such as Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, SVM, Bagging, VCH, VCS, XGB. The trained model's performance is assessed using metrics like accuracy on separate test data to ensure its effectiveness.

2.Model Building for Hate Speech Detection:

To detect hate speech on Twitter, we gather labeled data tweets. Then used to train machine learning models such as Decision Tree. The trained model's performance is assessed using metrics like accuracy on separate test data to ensure its effectiveness.

CHAPTER 2

LITERATURE REVIEW

2.1 Hate Speech Detection Methods

Ziqi Zhang, et.al [33] in their research, the authors leverage deep learning techniques, specifically a combination of Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs), to analyse and classify tweets as either containing hate speech or not. CNNs are well-suited for capturing local features in text data, while GRUs are effective in capturing temporal dependencies, making them suitable for sequential data like text.

The proposed model operates by first encoding the textual content of tweets using a convolutional layer to extract local features. The output of the convolutional layer is then fed into a GRU layer to capture temporal dependencies and context within the text. Finally, the model produces a classification output indicating whether a tweet contains hate speech or not.

By combining the strengths of CNNs and GRUs, the authors aim to create a more robust and effective model for hate speech detection on Twitter. They evaluate the performance of their approach using standard evaluation metrics such as accuracy, precision, recall, and F1-score, comparing it against other existing methods.

The results presented in the paper demonstrate the effectiveness of the proposed Convolution-GRU based deep neural network in accurately identifying hate speech on Twitter. The research contributes to the broader field of natural language processing and computational social science by addressing the pressing issue of hate speech detection in online social media platforms.

Thomas Davidson et.al [12] investigated hate speech detection by analyzing a large-scale dataset of tweets from Twitter. It confronts the complexities associated with distinguishing between offensive language, which may not necessarily constitute hate speech, and genuine instances of hate speech that target specific groups based on race, ethnicity, religion, gender, sexual orientation, or other characteristics.

To tackle this challenge, the authors propose a supervised learning approach that utilizes a combination of linguistic and semantic features to classify tweets as containing hate speech or not. This approach involves training machine learning models on labeled

data, where tweets are annotated as either hateful or non-hateful, to develop classifiers capable of automatically identifying hate speech.
One of the key contributions of the paper lies in its examination of the nuanced nature of hate speech and the difficulty in defining and detecting it accurately. The authors discuss the limitations of existing hate speech detection methods, particularly those that rely solely on lexical cues or keyword-based approaches, which may fail to capture the subtleties of hate speech and offensive language.
By presenting empirical findings and insights gained from their analysis of hate speech detection on Twitter, the authors shed light on the broader challenges and implications of addressing hate speech in online social networks. Their research underscores the importance of developing more sophisticated computational approaches and incorporating contextually relevant features to improve the accuracy and effectiveness of hate speech detection systems.

Nemanja Djuric et.al [14] addressed the challenge of identifying hate speech in online comments, which often contain complex language and subtle expressions of hostility. Traditional approaches to hate speech detection typically rely on handcrafted features or lexicons, which may struggle to capture the nuances of hate speech and context-dependent language.
To overcome these limitations, the authors propose a novel approach that utilizes comment embeddings—dense, low-dimensional vector representations of text learned from a large corpus of online comments. By training deep learning models to generate comment embeddings, the method captures semantic information and contextual nuances present in the comments, enabling more effective hate speech detection.
The paper describes the process of training comment embedding models using techniques such as word embeddings and neural networks. Once trained, these models are used to generate embeddings for individual comments, which are then fed into a hate speech classifier. The classifier, typically a machine learning model such as logistic regression or a neural network, learns to distinguish between hateful and non-hateful comments based on the embeddings.
The key contribution of the paper lies in its exploration of comment embeddings as a powerful representation for hate speech detection. By leveraging the semantic information encoded in comment embeddings, the proposed method achieves state-of-

the-art performance on various hate speech detection datasets, including Reddit and Twitter.
Overall, the research demonstrates the effectiveness of using comment embeddings for hate speech detection and highlights the potential of deep learning techniques in addressing the challenges of identifying hate speech in online comments. The findings contribute to advancing the field of natural language processing and computational linguistics, offering valuable insights into combating hate speech in online social platforms.

Shakir Khan et.al [20] addressed the challenge of accurately identifying hate speech in Twitter data, which is characterized by its brevity, informal language, and contextual ambiguity. Traditional methods for hate speech detection often struggle to capture the nuanced linguistic features and subtle contextual cues present in tweets.
To overcome these limitations, the authors propose a hierarchical attention-based approach that incorporates attention mechanisms to focus on relevant parts of the tweet and aggregate information at different levels of granularity. The model operates in two stages: at the word level and at the tweet level.
At the word level, the model uses a bidirectional Long Short-Term Memory (LSTM) network with attention mechanisms to capture the importance of individual words in the tweet. This enables the model to attend to informative words while ignoring irrelevant ones, enhancing the representation of the tweet's content.
At the tweet level, the model employs another attention mechanism to aggregate information from the word-level representations and generate a tweet-level representation. This allows the model to capture the overall sentiment and context of the tweet, which is essential for accurately detecting hate speech.
The key contribution of the paper lies in its hierarchical attention-based architecture, which effectively leverages both word-level and tweet-level information for hate speech detection. By attending to informative words and capturing the overall context of the tweet, the proposed approach achieves competitive performance on hate speech detection tasks in Twitter data.
Overall, the research demonstrates the effectiveness of the hierarchical attention approach in addressing the challenges of hate speech detection in Twitter. The findings contribute to advancing the field of natural language processing and computational

linguistics, offering a promising solution for identifying hate speech in online social platforms.

2.1.1 Limitations of existing hate speech detection techniques

Deep learning models used for hate speech detection, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can be computationally intensive and resource-demanding. Deploying these models in real-time systems or on platforms with large user bases poses scalability challenges, as they may require significant computational resources and memory overhead. Models trained on hate speech detection datasets may inherit biases and imbalances inherent to the data. When applied to different tasks, these biases can lead to skewed predictions or inaccurate classifications, particularly in scenarios where the distribution of classes differs from the hate speech detection task. Addressing data biases and imbalances may require careful preprocessing, augmentation, or collection of task-specific datasets to ensure model fairness and generalizability.

2.2 Gender Prediction Models

Alowibdi et al. [1] conducted an empirical evaluation where it analyzes a range of profile characteristics, such as user descriptions, profile images, tweet content, and metadata (e.g., number of followers, followings, and tweet count). These features are extracted from a dataset of Twitter users, and machine learning algorithms are employed to classify users into male and female categories based on their profile information.

The study evaluates the performance of different classification algorithms, such as logistic regression, support vector machines (SVM), and decision trees, in predicting gender based on the selected profile characteristics. It assesses the accuracy, precision, recall, and F1-score of each model to determine their effectiveness in gender classification on Twitter.

Additionally, the paper discusses the implications of the findings and provides insights into which profile characteristics contribute most significantly to gender prediction accuracy. The results of the empirical evaluation offer valuable insights for researchers and practitioners interested in gender classification and social media analysis.

Overall, this paper contributes to the understanding of gender prediction on Twitter by empirically evaluating the predictive power of various profile characteristics and shedding light on effective approaches for gender classification in online social networks.

Angeles et al. [3] employed a text-based approach, where features extracted from the text of users' tweets are used to train gender classification models. These features may include word frequencies, linguistic patterns, and other textual attributes that are indicative of gender-specific language use.
The research evaluates the performance of two popular machine learning algorithms, Naive Bayes and SVM, in classifying the gender of Twitter users based on their tweets. Both algorithms are trained on labelled datasets containing examples of users' tweets and their corresponding gender labels (e.g., male or female).
The study assesses the accuracy, precision, recall, and F1-score of the classification models to determine their effectiveness in predicting gender based on tweet content. It also explores the impact of different feature representations and preprocessing techniques on classification performance.
By comparing the performance of Naive Bayes and SVM algorithms, the paper provides insights into the strengths and weaknesses of each approach for gender classification on Twitter. The findings contribute to the understanding of text-based gender prediction techniques and offer practical implications for social media analysis and user profiling.
Overall, this paper advances the field of gender classification on Twitter by leveraging machine learning algorithms to analyse tweet content and predict users' gender based on their linguistic behaviour.

Bamman et al. [5] investigated how individuals express their gender identity through language usage in online social networks, specifically examining lexical features and patterns associated with different gender groups. The authors analyse a large dataset of Twitter posts to identify linguistic cues that may be indicative of gender identity.
One key aspect of the study is the examination of lexical variation, which refers to differences in language use between different gender groups. By analyzing the frequency of words and phrases used by male and female users on Twitter, the researchers aim to uncover patterns that reflect gender-specific linguistic behaviour.

The paper may employ various computational linguistic techniques to analyze the Twitter data, such as natural language processing (NLP) and machine learning algorithms. These techniques help identify gender-specific linguistic features and patterns, allowing the researchers to quantify the relationship between language use and gender identity.
Additionally, the study may investigate how factors such as age, location, and social network structure influence gendered language use on social media platforms. By considering these contextual factors, the authors aim to provide a more comprehensive understanding of the relationship between gender identity and language variation in online communication.
Overall, "Gender Identity and Lexical Variation in Social Media" contributes to the field of sociolinguistics and computational linguistics by shedding light on how gender identity is reflected in language use on platforms like Twitter. The findings of the study have implications for understanding gender dynamics in online communities and may inform research in areas such as identity representation, social interaction, and digital communication.

Alowibdi et al. [2] focused upon the findings of the previous work by Alowibdi et al. (2013a), which focused on evaluating profile characteristics for gender classification on Twitter. In this study, the authors seek to overcome language barriers and develop models that can accurately predict gender regardless of the language of the tweets.
To achieve language independence, the paper explores the use of features that are not tied to specific languages, such as user metadata and behavioural patterns. Additionally, the study investigates the effectiveness of machine learning algorithms that can generalize well across languages, such as support vector machines (SVM) and decision trees.
The research evaluates the performance of these language-independent approaches on multilingual Twitter datasets, considering users who tweet in languages other than English. The classification models are trained and tested using a diverse range of linguistic data to assess their ability to predict gender accurately across different languages.
By focusing on language-independent features and algorithms, the paper aims to overcome the limitations of language-specific approaches and develop more robust gender classification models for social media platforms like Twitter. The findings

contribute to the advancement of gender prediction techniques in multilingual online environments and have implications for cross-cultural social media analysis.

Barber et al. [6] examined whether gender differences influence investment decisions and performance in the stock market. It explores the hypothesis that men tend to exhibit higher levels of overconfidence compared to women, which may lead to differences in investment strategies and outcomes.
The title of the paper, "Boys will be boys," reflects the notion that traditional gender norms and societal expectations may contribute to behavioral differences between men and women in the context of investment decision-making. The phrase suggests that certain behavioral tendencies associated with masculinity, such as overconfidence or risk-taking, may manifest in investment behavior.
The study may employ various research methods, including empirical analysis of investment data, surveys, and experiments, to investigate the research questions. It may analyze historical trading data to examine differences in investment performance between male and female investors and explore whether these differences can be attributed to overconfidence or other behavioral biases.
Additionally, the paper may delve into psychological theories of overconfidence and decision-making to provide theoretical insights into the underlying mechanisms driving gender differences in investment behavior. It may also discuss the implications of the findings for financial markets, investor education, and policy interventions aimed at promoting financial literacy and reducing gender disparities in investment outcomes.
Overall, "Boys will be boys: Gender, overconfidence, and common stock investment"

Burger et al. [9] contributed to the literature on behavioural finance by highlighting the role of gender and overconfidence in shaping investment behaviour and outcomes. The findings of the study have implications for understanding investor behaviour, risk management, and financial decision-making in both individual and institutional contexts.
explore whether linguistic cues and patterns within tweets could reliably indicate the gender of the user. This investigation was particularly relevant given the burgeoning popularity of Twitter as a platform for social interaction and communication. By employing computational techniques and analyzing a large dataset of tweets, Burger

and his team aimed to uncover underlying trends and linguistic markers that could potentially distinguish between male and female users on the platform.
The study employed a combination of linguistic features, such as word choice, syntax, and writing style, to train machine learning algorithms to predict the gender of Twitter users. Through rigorous analysis and experimentation, Burger et al. uncovered several patterns that exhibited significant predictive power. These patterns included differences in vocabulary usage, emoticon usage, and syntactic structures between male and female users. By leveraging these insights, the researchers were able to develop models capable of accurately classifying gender based solely on the textual content of tweets.
The findings of Burger et al. shed light on the intricate relationship between language use and gender expression in online social platforms like Twitter. Their study not only demonstrated the potential of computational methods in deciphering sociolinguistic phenomena but also underscored the importance of considering diverse linguistic behaviors in digital contexts. Moreover, the research opened avenues for further exploration into the intersection of gender, language, and online communication, with implications for fields ranging from computational linguistics to gender studies and beyond.

Liu, W., et al. [21] focused specifically on the utility of first names as predictive features. The researchers aimed to investigate whether the simple inclusion of users' first names could enhance the accuracy of gender classification algorithms. This approach was novel in its simplicity yet potentially powerful, as first names often carry strong cultural associations and are readily available in users' profiles.
Through meticulous analysis of a vast dataset of Twitter users and their associated first names, Liu et al. explored the extent to which certain names exhibited gender-specific tendencies. By leveraging computational techniques and machine learning algorithms, they assessed the predictive strength of first names in accurately discerning the gender of Twitter users. The study encompassed diverse linguistic and cultural contexts, recognizing that naming conventions vary widely across different regions and demographics.
The findings of Liu and his team revealed intriguing patterns in the relationship between first names and gender on Twitter. Certain names emerged as strong indicators of gender, while others exhibited more ambiguity or variability. By incorporating first names as features, the researchers were able to enhance the performance of gender

classification models, underscoring the importance of considering even seemingly straightforward attributes in computational analyses of social media data. This study not only contributed to the evolving landscape of gender inference in digital contexts but also highlighted the nuanced interplay between cultural markers, linguistic cues, and algorithmic predictions in online platforms like Twitter.

Vashisth, P., et al. [28] aimed to leverage the wealth of textual information available on Twitter to discern gender based on linguistic cues and patterns. Building upon prior work in computational linguistics and gender inference, the study sought to refine existing methodologies and enhance the accuracy of gender classification algorithms. Vashisth et al. employed sophisticated natural language processing techniques and machine learning algorithms to analyze a large corpus of Twitter text data. Through meticulous preprocessing and feature engineering, the researchers extracted informative linguistic features that could effectively discriminate between male and female users. These features encompassed a wide array of textual attributes, including vocabulary usage, syntactic structures, sentiment expressions, and discourse patterns. The findings of the study demonstrated the viability of gender classification using Twitter text data, with robust performance achieved by the developed models. By elucidating the intricate relationship between language use and gender identity in online communication, Vashisth and his team contributed valuable insights to both computational linguistics and gender studies. Furthermore, their research underscored the importance of context-aware approaches in gender inference, recognizing the diverse linguistic behaviours and cultural nuances inherent in social media discourse. Overall, the study advanced our understanding of gender classification in digital environments while paving the way for future investigations into the intersection of language, identity, and technology.

2.2.1 Limitations of existing Gender Prediction techniques

Gender prediction models developed using social media data may not generalize well across different demographics, cultures, or languages. Similarly, findings related to investment behavior may not be generalizable to all market conditions or investor profiles. Many studies rely on social media data for gender prediction, which may suffer from biases and lack representativeness. Social media users may not be representative of the general population, leading to skewed results. Similarly, investment behaviour studies may be based on historical trading data, which may not capture the full diversity of investors. The effectiveness of gender prediction models and investment behaviour studies heavily depends on the choice and engineering of features. Selecting relevant features from social media profiles or investment data and encoding them appropriately can be challenging and may impact the performance of the models.

To overcome this, we used combination of tweets and user description for better accuracy and we trained various machine learning models and then selected the best one which yields high accuracy for testing the real-world tweets.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Requirement Specification

Requirements analysis, also called requirements engineering, is the process of determininguser expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

1. Less response Time
2. High Data Utility
3. Preserve security of database
4. Better Performance
5. Less computational intensity

3.1.1 Functional Requirements

1.Data Collection and Enrichment:

Collect Twitter data for analysis, including tweets and user profiles. Later extend the dataset with additional information

2.Gender Prediction Method:

Develop a method for predicting the gender of Twitter users. Analyse words used in tweets and profiles to infer gender. Investigate the effectiveness of combining multiple sources of information (tweets and profiles) for gender prediction.

3.Hate Speech Detection Method:

Develop a method for detecting hatef4ul language in tweets. Analyze tweet content to identify offensive or hateful words.

Implement simple methods for assessing tweet language and determining hate speech.

4.Experimentation and Evaluation:

Conduct experiments to evaluate the effectiveness of different computer methods for gender prediction and hate speech detection. Test various techniques and algorithms for predicting gender and detecting hate speech. Measure the accuracy and performance of each method through evaluation metrics.

3.1.2 Non-Functional Requirements

1. Performance:

The system should process data efficiently, considering the large volume of Twitter data. Response time for gender identification and hate speech detection should be minimal to provide a smooth user experience.

2. Accuracy:

The gender identification and hate speech detection methods should achieve high accuracy in classifying users' genders and detecting hate speech and abusive content. The system should strive to minimize false positives and false negatives in gender classification and hate speech detection.

Requirements Model

Requirement analysis, also called requirements engineering, is the process of determininguser expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirement analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. **User Requirements**

1. Execution should be fast
2. More accurate
3. User-friendly

Software Requirements

1. Operating System: Windows
2. Language: Python3

Hardware Requirements

- Processor - Pentium IV or higher
- Speed – 2.4GHz
- RAM - 256 MB (min)
- Hard Disk - 512 MB (min)

3.2 UML Diagrams for the project work

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it’s one of the most popular business process modeling techniques. It is based on diagrammatic representations of software components.

The various UML diagrams are:

1. Use Case diagram
2. Activity diagram
3. Sequence diagram
4. Collaboration diagram
5. State chart diagram
6. Class diagram
7. Component diagram
8. Deployment diagram

3.2.1 System View Diagrams

Use Case Diagram

Here there are mainly three actors such as developer, tester and end user. The use cases for the end user are collecting and sending data to developer and analysing the final results. The use cases for the developer are the data collection, perform preprocessing, building NLP models, building ML models, perform model training. Later the developed project is tested by tester. The use cases for the tester are testing and result analysis. The tester tests the project against all types of test cases where the end user interacts in analysing final results and performance of every proposed model with the help of their accuracy scores. We linked the data collection and result analysis part between end user, developer and tester as the input and output data is interlinked to all the actors involved in this project.

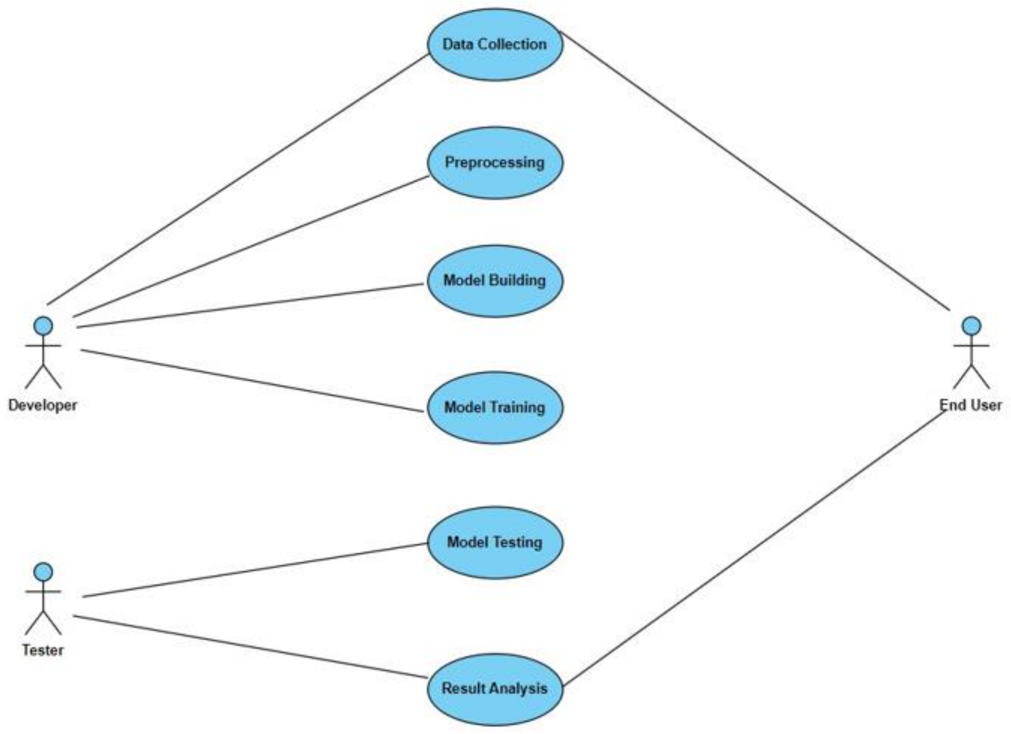


Fig 3.1 Use Case Diagram

Activity Diagram

This diagram shows the different activities involved in gender classification from data collection to result analysis. At first the control flow starts from collecting and loading dataset. If the data is cleaned and ready to process through the proposed models then it can be directly sent to the NLP model as input. Otherwise, we perform preprocessing and send the data to further procedure. Now we build a NLP model with the pre-processed data as input to perform text analysis through that model. And we build proposed ML algorithms. Next the processed output text is given as input to each ML model and perform training. Later we perform testing the project against multiple test cases. Then we analyse results and performance of all the proposed models in identifying the gender of a twitter user.

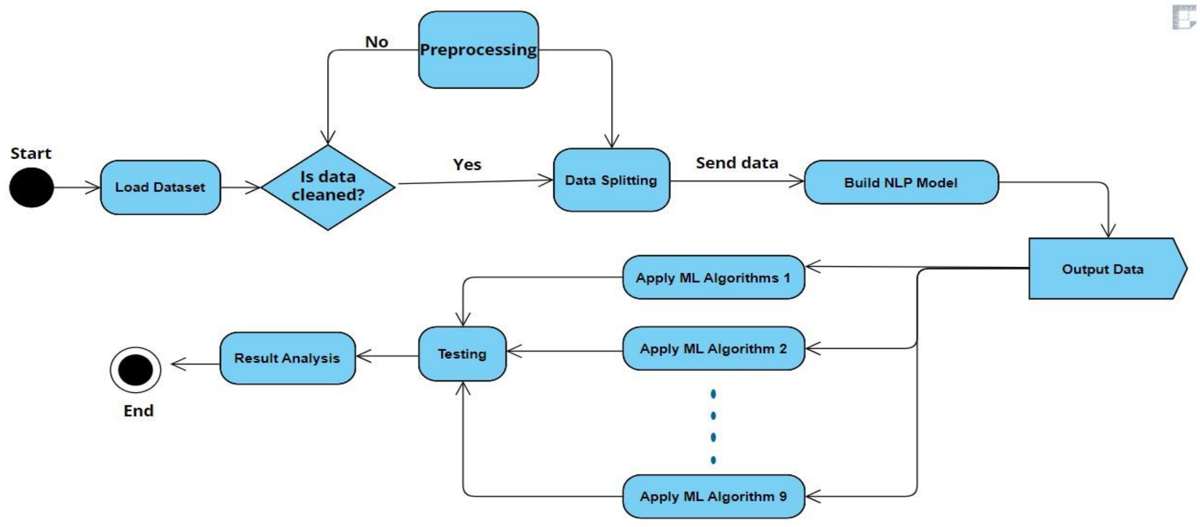


Fig 3.2 Activity Diagram

Sequence Diagram

This diagram represents the lifetime of every module involved in this project. At first this project starts from loading dataset. The lifetime of the collected dataset is from starting and exists up to preprocessing phase. Besides we added additional data to the existing data in order to make it more accurate. The lifetime of extended data is in between original data collection and preprocessing phase. After we perform preprocessing on the extended dataset. It produces the pre-processed data that exists up to NLP model building. We pass this processed data to the NLP model. The lifetime of NLP model exists between preprocessing and building ML models. Later it gives processed text to the ML model which exists in between NLP model to testing. The lifetime of the testing phase starts from the completion of the ML model training and result analysis which is the final phase in determining the gender of the twitter user.

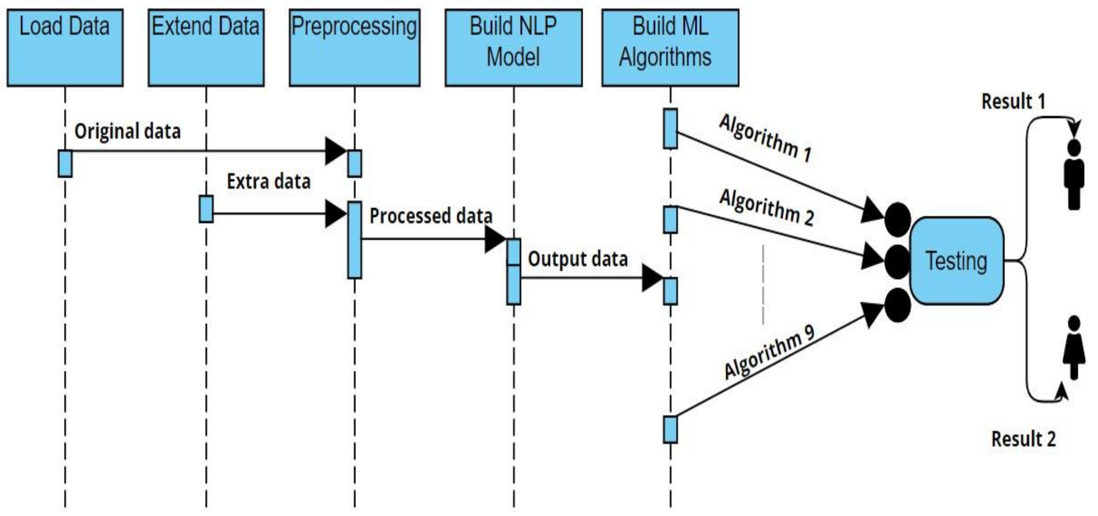


Fig 3.3 Sequence Diagram

Collaboration Diagram

This collaboration diagram shows the view of interactions among the onjects, modules and their relations. In this diagram the user interactes with the system that are represented using rectangle symbol. The connection between user and system are represented by drawing a linking line between them. The interactions or dataflow between user and system are shown with the help of arrow marks with appropriate directions. The order of steps or operations performed in predicting the gender of a twitter user are mentioned in a rectangular symbol. The user starts from dataset collection and contributes to perform final result analysis. Besides the collaboration between user and system, the intermediate components such as NLP model components, ML model components and testing components are also collaborates among them for data transferring to their neighbourhood modules after processing text through them.

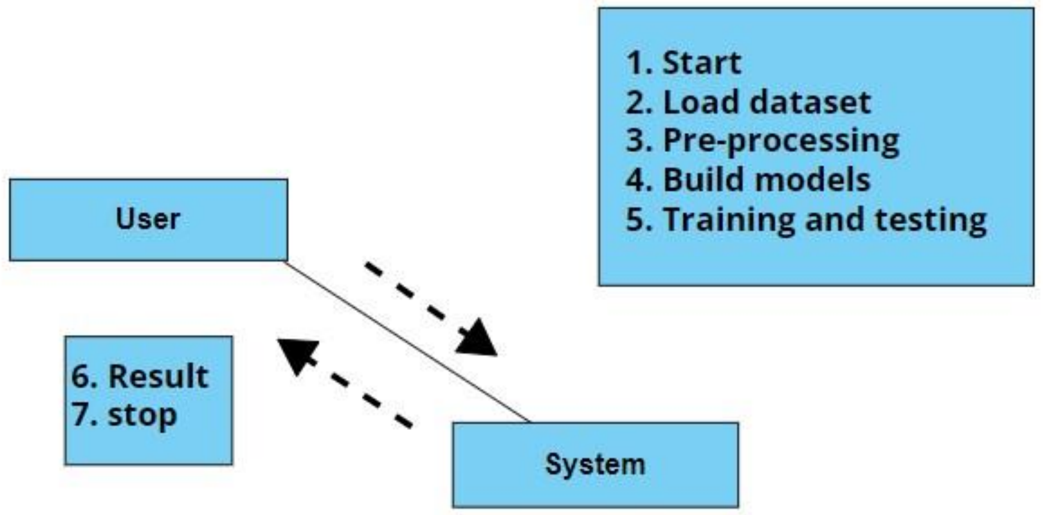


Fig3.4 Collaboration Diagram

Class Diagram

This diagram shows the interactions and connections between different classes used in implementing the project. It starts from Data_collection class which contains the methods like load_dataset() and extend_data(). It is used in loading and extending the dataset. Next the extended dataset is passesd to preprocessing class which contains the methods such as Remove_stopwords(), Remove_symbols(), Remove_urls(), Remove_duplicates() and Feature_extraction(). This class is responsible for cleaning and filtering the raw dataset. Next this preprocessed dataset is given to NLP_model class which contains the methods like Model_building(), Model_training() and Get_accuracy(). This class is responsible for text processing through NLP model. This process is applied for every NLP model. Later the output text is given to the ML_models class which contains the methods like Naïve_bayes(), Logistic_regression(), Decision_tree() etc. Later the Testing class involves which contains the methods like Testcase1(), Testcase2() ect. This class is used to test the project against all the test cases. Later results we be analyzed to predict the gender of a twitter user.

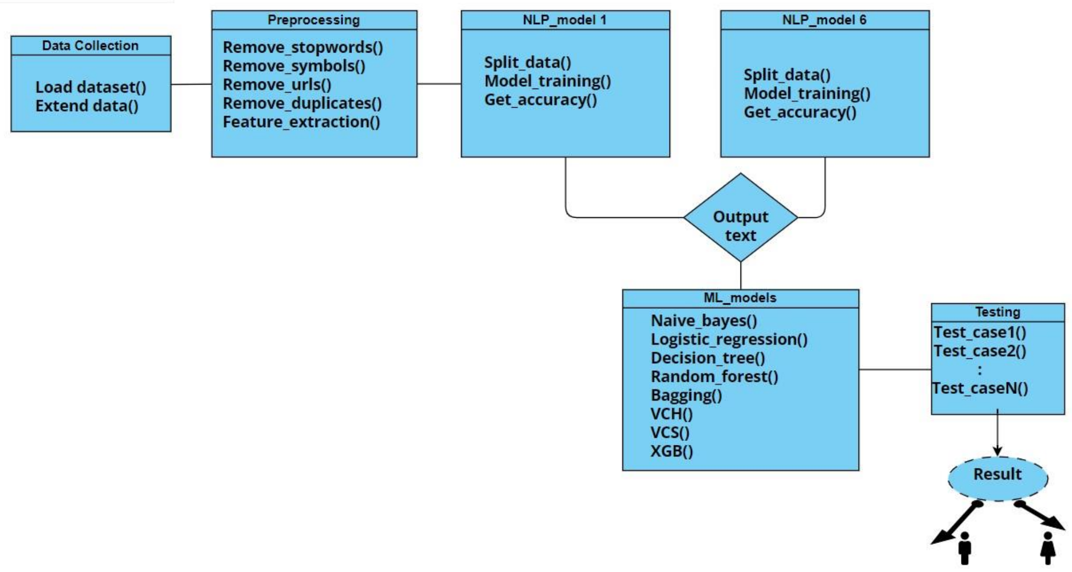


Fig 3.5 Class Diagram

State Chart Diagram

This diagram represents the different stages of gender classification process which starts from data collection phase and continues up to result analysis phase. At first, we load the datasets and check whether the data is cleaned. If the data is not cleaned then it undergoes through the preprocessing phase and pass this data to further phase. If the data is already cleaned i.e ready to perform text classification, then it is directly passed to the further phase. Now the filtered data is passed to NLP model. It generates the textual data that has processed by NLP model. The same data is forwarded to ML model phase which builds all proposed ML models and performs training. The data processed by each NLP model is further continued to process under all the prposed ML models. Later the model is tested under different test cases and the results are predicted. This is the end of the project. Later the performance of every proposed model is analyzed.

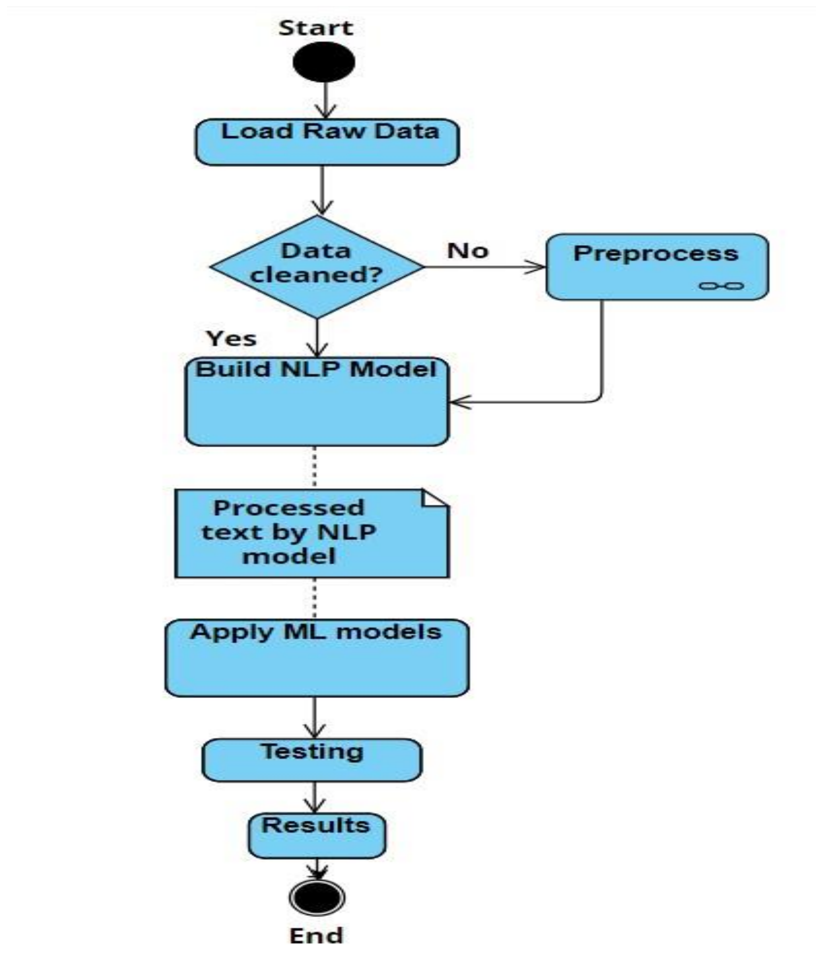


Fig 3.6 State Chart Diagram

3.2.2 Detailed View Diagrams

Component Diagram

This diagram represents the relationships between every component involved in predicting the gender of a twitter user. It Contains the different components such as loading data component, preprocessing component, NLP_model component, ML_model component and testing component. The data collection component is connected to the preprocessing component as the preprocessing should be performed after collecting the raw data. the processed text is connected to the NLP_model component and preprocessed component as the preprocessed(filtered) data is passed to NLP_model component and the NLP_model performs the text analysis and generates the processed text. The processed text is connected to each and every ML_model component where the intermediate resultant data produced by NLP_model component is further applied to ML_model components. Later each ML_model model component is connected to Testing component where each model is tested under all test cases. Later results are analyzed and performs gender prediction for twitter account user.

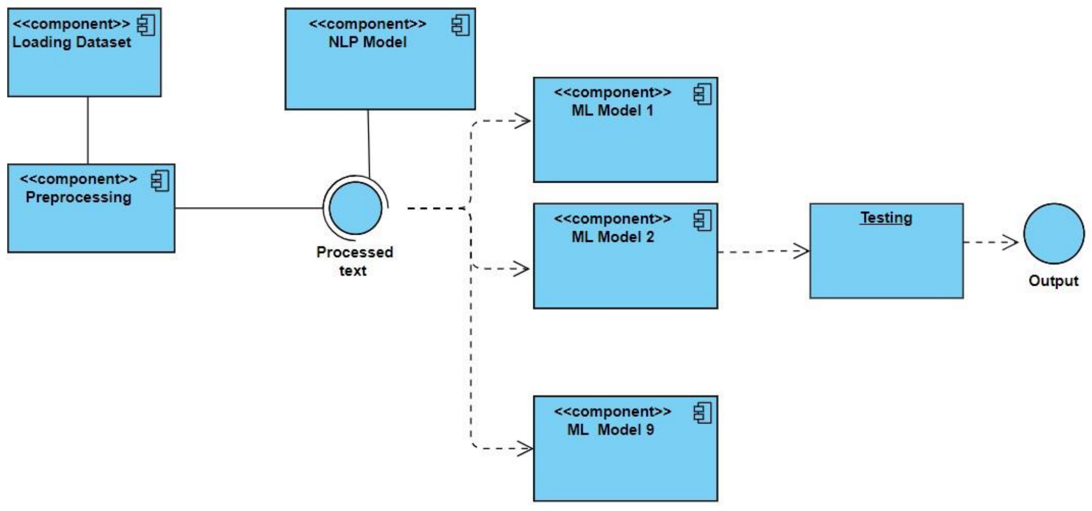


Fig 3.7 Component Diagram

Deployment Diagram

The deployement diagram shows the distribution of hardware modules, services, artifact nodes and their execution within the system architecture. This diagram workflow starts from data collection node. This node contains pre-processing node as sub node as the preprocessing is also a part of it. Later the NLP_model component is an artifact node that performs text analysis. It is conneceted to data collection node as it occurs after pre-processing. Later this NLP_model node is connected to all the proposed ML_model nodes. Each NLP_model node is interacted with every ML_model nodes to compare the efficiency of each NLP_model with respect to every ML_model. All the ML_model nodes are then interacted with testing node to test the results and accuracy scores. Then the testing node is connected to the results node to predict the final result and performance analysis.

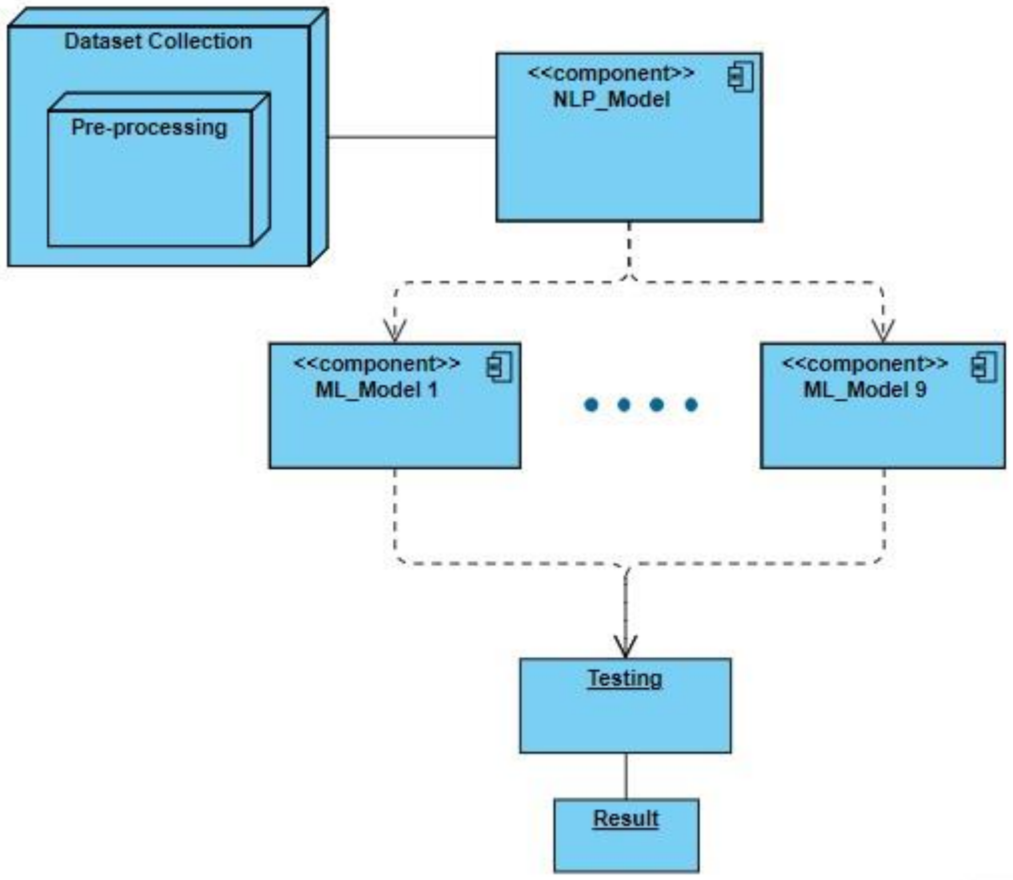


Fig 3.8 Deployment Diagram

CHAPTER 4
SYSTEM DESIGN

4.1 Architecture of the proposed system

4.1.1 Naïve Bayes

The architecture of Naive Bayes is straightforward. It's based on Bayes' theorem, which calculates the probability of something happening given other known probabilities. In the case of Naive Bayes classification, it calculates the probability of a data point belonging to a certain class based on the features it has.

Here's how it works:

Probability Model: Naive Bayes assumes that all features are independent of each other. So, it calculates the probability of each class label given the features using Bayes' theorem. This means it calculates how likely it is for a data point to belong to a certain class based on the features it has.

Parameter Estimation: To make these calculations, Naive Bayes needs to estimate two types of probabilities from the training data:

Class Prior Probability: How often each class appears in the dataset.

Class-Conditional Feature Probability: How likely each feature is to appear given a specific class. This can be calculated differently depending on the type of features.

Decision Rule: When it's time to classify new data, Naive Bayes uses the probabilities it calculated to make a decision. It picks the class label with the highest probability as the predicted class for the new data point.

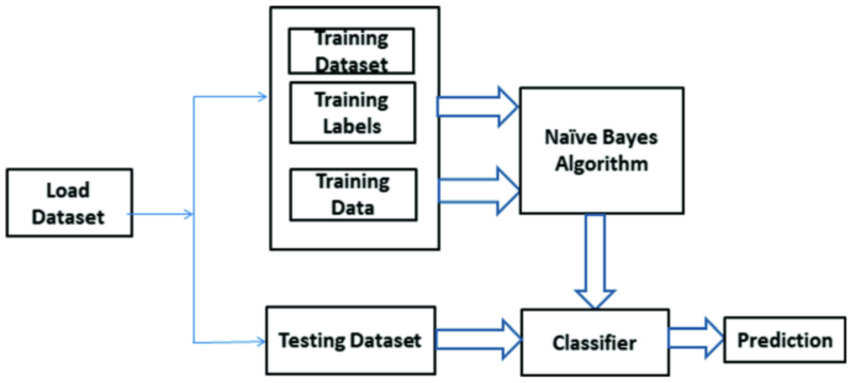


Fig 4.1 Overview of Naïve Bayes Model

4.1.2 Logistic Regression

Logistic Regression is a popular statistical method used for binary classification tasks, where the goal is to predict the probability that an instance belongs to one of two classes. Despite its name, it's actually a linear model for classification rather than regression.

Here's how it works:

Model Representation: Logistic Regression models the relationship between the independent variables (features) and the binary dependent variable (class label) using the logistic function. The logistic function (also called the sigmoid function) transforms any input into a value between 0 and 1, representing the probability of the instance belonging to the positive class.

Parameter Estimation: The model parameters, including coefficients for each feature, are estimated during the training phase using optimization techniques such as gradient descent. The objective is to minimize a cost function that quantifies the difference between the predicted probabilities and the actual class labels in the training data.

Decision Boundary: Once trained, Logistic Regression defines a decision boundary that separates the instances of different classes in the feature space. This boundary is typically linear in the input space for binary classification but can be more complex in higher dimensions or with more classes.

Prediction: To classify new instances, Logistic Regression calculates the probability that each instance belongs to the positive class based on its features and compares it to a threshold (usually 0.5). If the probability is above the threshold, the instance is classified as belonging to the positive class; otherwise, it's classified as belonging to the negative class.

4.1.3 Decision Tree

Decision trees are a fundamental tool in machine learning used for both classification and regression tasks. They operate by recursively partitioning the feature space into smaller regions, ultimately forming a tree-like structure. Here's how they work:

Model Representation: At the root of the decision tree, the entire dataset is considered. The tree then splits the dataset into smaller subsets based on the value of a chosen feature. This process continues recursively for each subset, creating a tree structure where each internal node represents a decision based on a feature, and each leaf node represents the predicted class or value.

Splitting Criteria: The decision tree algorithm determines the feature and split point at each node to maximize the homogeneity of the resulting subsets. Common splitting criteria include Gini impurity and entropy, which measure the impurity or disorder of a set of samples. The goal is to minimize impurity and maximize the separation of classes in the resulting subsets.

Stopping Criteria: The tree-building process continues until a stopping criterion is met, which may include reaching a maximum tree depth, having a minimum number of samples at each node, or achieving perfect purity in the subsets.

Prediction: Once the decision tree is constructed, it can be used to make predictions for new instances by traversing the tree from the root to a leaf node based on the features of the instance. The predicted class or value associated with the leaf node reached by the instance determines the final prediction.

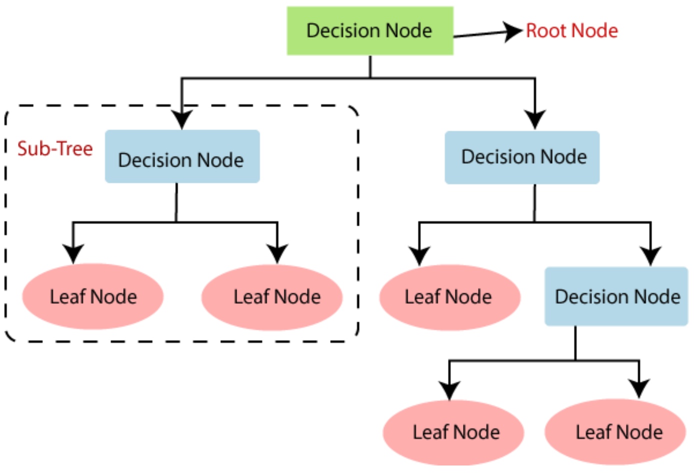


Fig 4.2 Overview of Decision Tree Model

4.1.4 Random Forest

Random Forest is a powerful ensemble learning method commonly used for classification and regression tasks. It operates by building multiple decision trees during training and combining their predictions to make more accurate and robust predictions. Here's how it works:

Bootstrap Sampling: Random Forest starts by randomly selecting subsets of the training data with replacement, a process known as bootstrap sampling. This creates multiple "bootstrapped" datasets of the same size as the original dataset but with some instances repeated and others omitted.

Decision Tree Construction: For each bootstrapped dataset, a decision tree is constructed using a random subset of features at each split point. This randomness helps to decorrelate the trees and reduce overfitting. The trees are typically grown deep, without pruning, to capture complex patterns in the data.

Voting or Averaging: Once all the decision trees are constructed, predictions are made for new instances by either taking a majority vote (for classification) or averaging the predictions (for regression) of all the individual trees. This ensemble approach helps to improve predictive accuracy and generalization by reducing variance and bias.

Feature Importance: Random Forest also provides a measure of feature importance, indicating the contribution of each feature to the predictive performance of the model. This information can be valuable for feature selection and understanding the underlying patterns in the data.

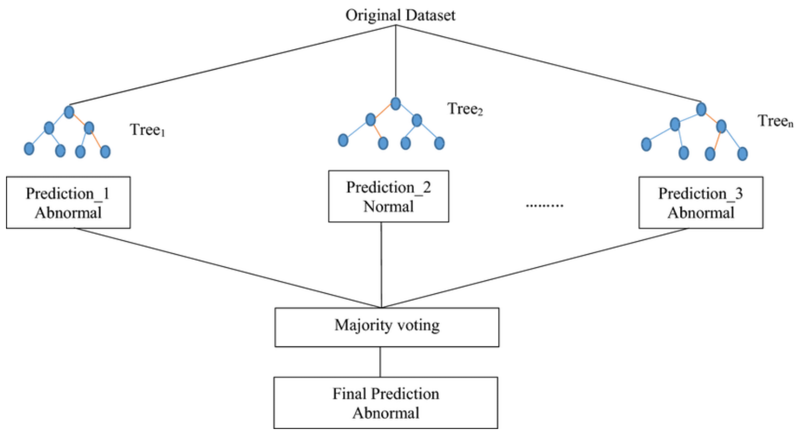


Fig 4.3 Overview of Random Forest Model

4.1.5 SVM

Support Vector Machines (SVM) are powerful supervised learning models used for classification and regression tasks. Here's an overview of how SVM works:

Linear Separation: The fundamental idea behind SVM is to find the hyperplane that best separates the instances of different classes in the feature space. For binary classification, this hyperplane is a line (in 2D) or a plane (in 3D) that maximizes the margin, i.e., the distance between the hyperplane and the nearest instances (called support vectors) of each class.

Kernel Trick: SVM can efficiently handle nonlinear separation by using the kernel trick. Instead of directly mapping the input features to a higher-dimensional space, where separation might be easier, SVM computes the dot product between instances in the transformed space without explicitly calculating the transformation. This allows SVM to implicitly operate in high-dimensional feature spaces without the computational cost of explicitly transforming the data.

Margin Maximization: SVM aims to maximize the margin around the decision boundary, as instances lying closer to the decision boundary are more likely to be misclassified. By maximizing the margin, SVM seeks to improve generalization performance and robustness to noise in the data.

Regularization: SVM also incorporates a regularization parameter (C) that controls the trade-off between maximizing the margin and minimizing the classification error on the training data. A smaller value of C leads to a wider margin but may result in more misclassifications, while a larger value of C prioritizes correct classification but may lead to overfitting.

Nonlinear Decision Boundaries: SVM can learn nonlinear decision boundaries by using different kernel functions such as polynomial, radial basis function (RBF), and sigmoid kernels. These kernels allow SVM to capture complex patterns and relationships in the data, making it suitable for a wide range of classification tasks.

4.1.6 Bagging

Bagging, short for Bootstrap Aggregating, is an ensemble learning technique used to improve the stability and accuracy of machine learning models, particularly decision trees. Here's how it works:

Bootstrap Sampling: Bagging starts by creating multiple subsets of the original training data through bootstrap sampling. In bootstrap sampling, random samples are drawn with replacement from the training dataset, resulting in multiple subsets of the same size as the original dataset. Each subset may contain duplicate instances and may also miss some original instances.

Base Model Training: Once the bootstrap samples are created, a base learning algorithm (e.g., decision tree) is trained independently on each subset of the training data. This results in multiple base models, each trained on a slightly different subset of the data.

Aggregate Predictions: During the prediction phase, the predictions of all base models are combined or aggregated to make a final prediction. For classification tasks, this aggregation can be done by majority voting, where the class that receives the most votes from the base models is selected as the final prediction. For regression tasks, the predictions of all base models may be averaged to obtain the final prediction.
Reduced Variance: The key idea behind bagging is that by training multiple models on different subsets of the data and aggregating their predictions, the variance of the model's predictions can be reduced. This helps to improve the model's generalization performance and reduces the risk of overfitting to the training data.
Diverse Models: Bagging encourages diversity among the base models by training them on different subsets of the data. This diversity helps to capture different aspects of the underlying data distribution and improves the robustness of the ensemble model.

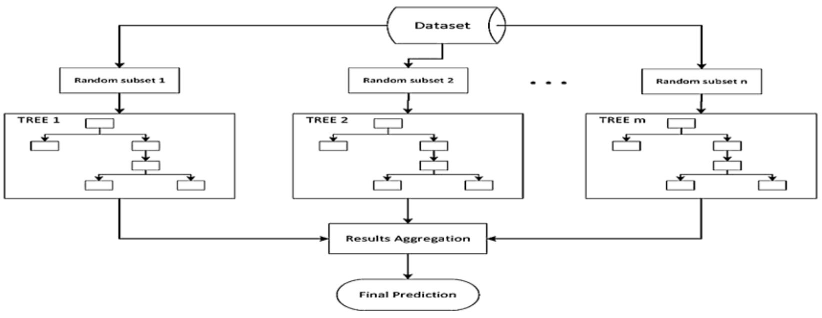


Fig 4.4 Overview of Bagging Model

4.1.7 VCH

Voting Classifier with Hard Voting (VCH) is an ensemble learning method that combines the predictions of multiple individual classifiers to make a final decision. Here's how VCH works:
Classifier Selection: VCH starts by selecting a set of base classifiers, each trained independently on the same training dataset. These base classifiers can be of different types (e.g., logistic regression, support vector machines, decision trees) or variations of the same type with different hyperparameters.
Voting: During the prediction phase, each base classifier independently predicts the class label for a given input instance. In the case of VCH with hard voting, the final prediction is determined by a majority vote among the predictions of all base classifiers. The class label that receives the most votes is selected as the final prediction.

Equal Weighting: In VCH with hard voting, all base classifiers are given equal weight, regardless of their individual performance or confidence in their predictions. This means that each base classifier contributes equally to the final decision, and the majority vote determines the final outcome.

Simple Aggregation: VCH with hard voting is relatively simple to implement and requires minimal computational overhead. It does not require any additional training beyond the individual base classifiers, making it easy to deploy and scale to large datasets.

Robustness: VCH with hard voting can improve the overall robustness and generalization performance of the ensemble model by combining the predictions of multiple classifiers. It can mitigate the risk of overfitting to the training data and reduce the impact of individual classifier errors.

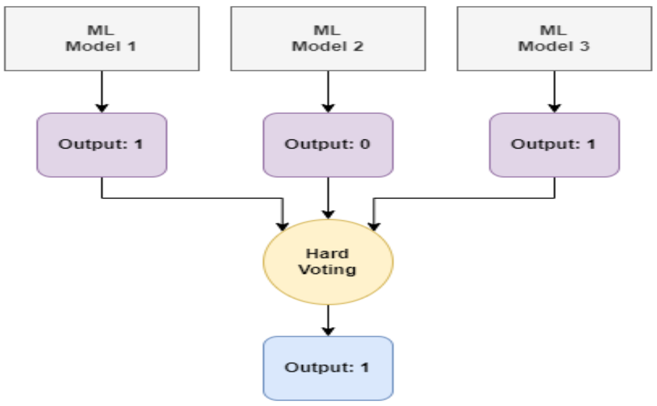


Fig 4.5 Overview of VCH Model

4.1.8 VCS

The Voting Classifier with Soft Voting (VCS) architecture involves the following components and steps:

Base Classifiers: The VCS ensemble starts with a collection of base classifiers, which can be any classification algorithms such as decision trees, support vector machines (SVM), logistic regression, or others. Each base classifier is trained independently on the same training dataset.

Soft Voting Mechanism: Unlike hard voting, where the final prediction is determined by a majority vote among the base classifiers, VCS with soft voting calculates the weighted average of the predicted probabilities for each class label across all classifiers.

This means that instead of simply counting votes, VCS considers the confidence level or certainty of each classifier's prediction.
Weighted Average Calculation: In VCS with soft voting, each base classifier's contribution to the final prediction is weighted based on its estimated performance or confidence level. More accurate classifiers or those with higher confidence in their predictions are assigned higher weights, while less reliable classifiers receive lower weights.
Final Prediction: The final prediction is determined based on the class label with the highest average probability across all classifiers. By taking into account the predicted probabilities from each classifier and their respective weights, VCS generates a more refined and probabilistic prediction compared to hard voting.
Flexibility and Adaptability: VCS offers flexibility in handling classifiers with different strengths and weaknesses. It can effectively combine predictions from diverse classifiers and adapt to various types of data distributions and classification tasks.
Robustness and Generalization: Similar to other ensemble methods, VCS with soft voting improves the overall robustness and generalization performance of the model by leveraging the collective knowledge of multiple classifiers. It helps mitigate the impact of individual classifier errors and reduces overfitting by considering the consensus among multiple classifiers.

4.1.9 XGB

XGBoost, short for eXtreme Gradient Boosting, is an efficient and scalable implementation of gradient boosting machines, a popular machine learning technique known for its accuracy and speed. Here's how XGBoost works:
Gradient Boosting Framework: XGBoost is based on the gradient boosting framework, which sequentially trains a series of weak learners (usually decision trees) to correct the errors made by the previous models. It combines the predictions of these weak learners to make a final prediction that typically exhibits superior performance compared to any individual learner.
Objective Function: XGBoost optimizes a differentiable objective function that measures the difference between the predicted and actual values of the target variable. The objective function consists of two parts: a loss function, which quantifies the

difference between the predicted and actual values, and a regularization term, which penalizes complex models to prevent overfitting.

Boosting Trees: XGBoost uses gradient boosting to train decision trees sequentially, with each tree attempting to correct the errors made by its predecessors. Unlike traditional gradient boosting methods, which grow trees greedily using a fixed structure, XGBoost employs a more sophisticated algorithm that allows for parallel tree construction and optimization of both the tree structure and leaf weights.

Regularization Techniques: XGBoost incorporates several regularization techniques to prevent overfitting and improve generalization performance. These techniques include shrinkage (or learning rate), which controls the contribution of each tree to the final prediction, and tree pruning, which removes nodes from the trees that contribute little to the overall improvement in the objective function.

Parallelization and Optimization: XGBoost is designed for efficiency and scalability, with support for parallel processing and distributed computing. It leverages hardware optimization techniques such as cache awareness and out-of-core computing to handle large datasets that do not fit into memory.

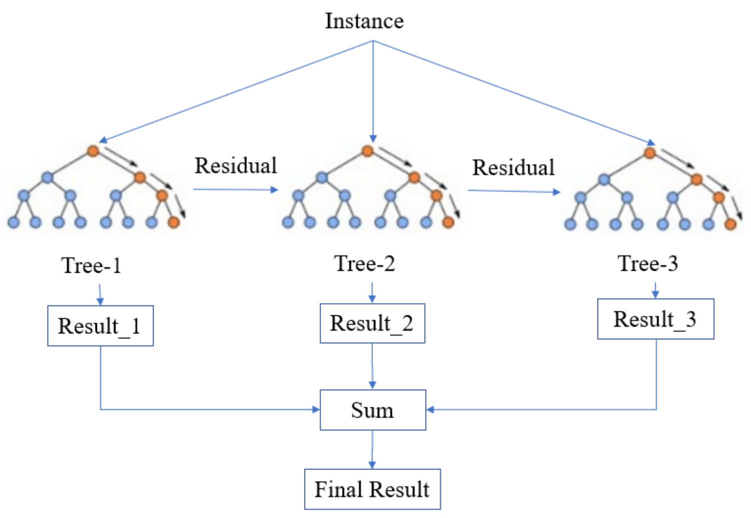


Fig 4.6 Overview of XGB Model

4.2 Workflow of the proposed System

The gender classification process begins with an input dataset containing columns such as tweets, user descriptions, user IDs, names, and creation dates. This raw data undergoes thorough preprocessing steps to ensure its suitability for analysis. These steps include the removal of hashtags, URLs, and punctuation, as well as converting text to lowercase and tokenizing it into individual words or tokens. By standardizing the text data in this manner, we ensure consistency and prepare it for further analysis.

Following preprocessing, the preprocessed data is subjected to various natural language processing (NLP) techniques for feature extraction. These techniques include advanced models like BERT and GPT-2, which capture semantic meanings and contextual information, as well as traditional methods like GloVe, Word2Vec (W2V), bag-of-words (BoW), and term frequency-inverse document frequency (TF-IDF). Each technique offers unique insights into the text data, allowing us to represent it in different ways and extract meaningful features.

Once the features are extracted, the preprocessed and feature-engineered data is fed into machine learning algorithms for training and prediction. A range of classification algorithms is employed, including Naive Bayes (NB), Logistic Regression (LR), Decision Trees (DT), Support Vector Machines (SVM), bagging, voting classifiers with heuristics (VCH), voting classifiers with scores (VCS), and Extreme Gradient Boosting (XGB). These algorithms analyze the extracted features and learn patterns to predict the gender labels of users based on their textual data.

To evaluate the performance of each model, accuracy metrics are computed by comparing the predicted gender labels to the true labels in the dataset. These accuracy scores serve as indicators of how well each model performs in classifying gender based on textual data. By plotting these scores on a graph, we can visually compare the performance of different NLP techniques and machine learning algorithms. This comparison helps us identify the most effective combinations of techniques and algorithms for gender classification tasks.

In addition to quantitative evaluation, qualitative analysis may also be conducted by providing real-world tweets as examples. These tweets demonstrate how the trained models classify gender based on actual textual data. By showcasing the models' predictions in real-world contexts, we gain insights into their effectiveness and potential areas for improvement. This holistic approach to evaluation ensures a comprehensive understanding of the models' performance and their applicability in practical scenarios.

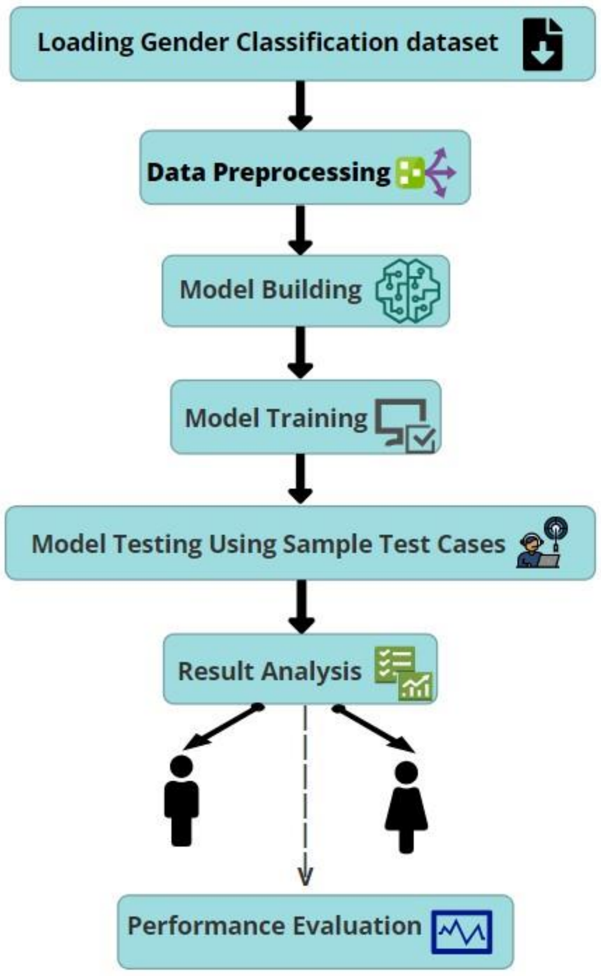


Fig 4.7 Workflow for Gender Prediction

For hate speech detection, the process begins with loading the dataset, which typically contains columns such as tweets, labels indicating whether the tweet contains hate speech or not, and possibly other metadata such as user IDs or timestamps. This raw data is then pre-processed to prepare it for analysis. Preprocessing steps may involve removing irrelevant characters, punctuation, special symbols, and stopwords, as well as converting text to lowercase and tokenizing it into individual words or tokens. These steps help standardize the text data and make it suitable for further analysis.

Following preprocessing, the pre-processed data is transformed into features using the bag-of-words (BoW) representation. BoW represents text data as a sparse matrix of word frequencies or presence indicators, capturing the occurrence of each word in the corpus. This feature extraction technique helps convert textual data into a numerical format that machine learning algorithms can process.

With the BoW representation in place, the preprocessed and feature-engineered data is ready to be used for training machine learning models. One common choice for hate speech detection is the Decision Tree (DT) algorithm, which is a simple yet effective classification model. Decision trees learn from the data by recursively partitioning the feature space into regions that correspond to different class labels, making decisions based on feature thresholds at each node.

Once the Decision Tree model is trained on the BoW-transformed data, its performance is evaluated using accuracy metrics. Accuracy measures how well the model predicts whether a tweet contains hate speech or not compared to the true labels in the dataset. By testing the model's predictions against real-world tweets, we can assess its ability to generalize to unseen data and identify any potential areas for improvement.

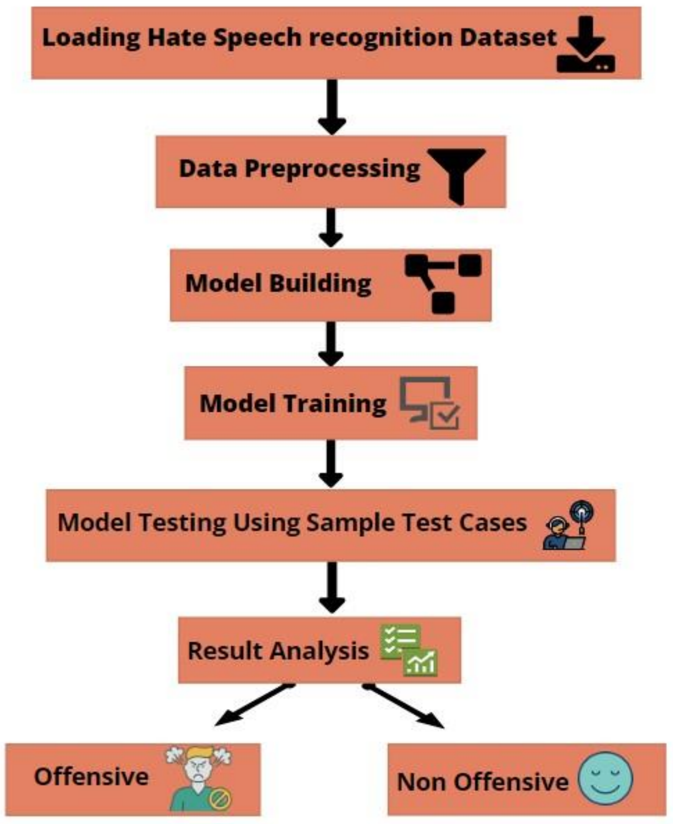


Fig 4.8 Workflow for Hate Speech Detection

4.3 Module Description

Two modules are implemented in this model

1. Gender Prediction Module
2. Hate Speech Detection Module

In the gender prediction module, the dataset with columns like tweets, user descriptions, IDs, names, and creation dates is loaded and preprocessed by removing hashtags, URLs, converting text to lowercase, and tokenizing it. Various Natural Language Processing (NLP) models including BERT, GPT-2, GloVe, Word2Vec, Bag of Words, and TF-IDF are applied to prepare the data. Multiple machine learning algorithms like Naive Bayes, Logistic Regression, Decision Trees, Support Vector Machines, Bagging, VCH, VCS, and XGBoost are trained on the prepared data. The accuracy of each model is assessed and compared using accuracy plots, with real-world tweets serving as a test dataset to evaluate model performance.

For the hate speech detection module, we begin by loading the dataset, which typically comprises columns such as tweets, user descriptions, IDs, names, and creation dates. Preprocessing steps involve removing hashtags, URLs, converting text to lowercase, and tokenization. Following this, we employ the Bag-of-Words (BoW) technique to represent the textual data. Machine learning algorithms such as Decision Trees are then trained on the BoW representation. The accuracy of the model is evaluated using appropriate metrics, and real-world tweets are used to test the model's performance.

CHAPTER 5
IMPLEMENTATION

5.1 Algorithms

5.1.1 NLP Algorithms

5.1.1.1 Bag of Words (BoW)

This is a simple yet effective technique used in natural language processing (NLP) to represent text data. It converts text documents into numerical vectors by counting the frequency of words in each document. Each unique word in the vocabulary is assigned a unique index, and the presence or absence of each word is recorded in the vector. While BoW is straightforward and easy to implement, it lacks the ability to capture the semantic meaning and context of words.

5.1.1.2TF-IDF (Term Frequency-Inverse Document Frequency)

This is another popular method for text representation. It enhances BoW by weighting the frequency of each word by its importance in the document relative to its frequency across all documents. Words that occur frequently in a specific document but rarely in others are given higher weights, helping to emphasize their importance in distinguishing the document's content. TF-IDF addresses some of the limitations of BoW by considering the rarity of words across the entire corpus.

5.1.1.3 BERT (Bidirectional Encoder Representations from Transformers)

This is a state-of-the-art transformer-based language model developed by Google. Unlike BoW and TF-IDF, BERT is a contextualized word embedding model, meaning it captures the meaning of words based on their context within a sentence or document. BERT is pre-trained on large amounts of text data and fine-tuned for specific NLP tasks, making it highly effective for tasks like sentiment analysis, text classification, and question answering. Its bidirectional nature allows it to understand the meaning of words in relation to both preceding and following words, leading to more accurate representations.

5.1.1.4 GloVe (Global Vectors for Word Representation)

This is a word embedding technique that aims to capture the semantic relationships between words in a corpus. It represents words as dense vectors in a continuous vector space, where the distance and direction between vectors encode semantic similarity and relatedness. GloVe utilizes global word-word co-occurrence statistics to learn word embeddings, allowing it to capture complex linguistic patterns and relationships. It has been widely used in various NLP tasks, including text classification, machine translation, and sentiment analysis.

5.1.1.5 GPT-2 (Generative Pre-trained Transformer 2)

This is a transformer-based language model developed by OpenAI. It is trained on a diverse range of text data and is capable of generating human-like text based on a given prompt. GPT-2 employs a transformer architecture with attention mechanisms, allowing it to capture long-range dependencies and contextual information in text data. While GPT-2 is primarily known for its generative capabilities, it can also be fine-tuned for specific NLP tasks like text classification and language modelling.

5.1.1.6 Word2Vec (W2V)

This is a word embedding technique developed by Google that represents words as dense vectors in a continuous vector space. It learns word embeddings by predicting the context of words within a fixed window size in a large corpus of text. Word2Vec captures semantic similarity between words by placing similar words close together in the vector space.

5.1.2 Machine Learning Algorithms

5.1.2.1 Naïve Bayes

In this algorithm, each feature's probability distribution is modelled independently, which simplifies the computation while still providing effective classification results. During training, MNB estimates the probabilities of each class and the likelihood of each feature given the class, enabling it to make predictions by calculating the posterior probability of each class given the observed features.

During the training phase, MNB learns from the labelled training data by estimating the prior probabilities of each class and the likelihood of each feature given the class. The prior probabilities are calculated based on the frequency of each class in the training data, while the likelihood estimates are derived from the occurrence of each feature (word) within each class. This information is then used to build a probabilistic model that can predict the most likely class for new input data during the testing phase.

5.1.2.2 Logistic Regression

By utilizing the logistic function, the model learns to estimate the probability that a given individual belongs to a particular gender class. The LogisticRegression() from the scikit-learn library is employed to instantiate the logistic regression model. During the training phase, the model learns from labelled data containing examples of individuals and their corresponding genders. It optimizes its parameters to best fit the training data, adjusting the coefficients of the input features to minimize the error between predicted and actual gender labels.

After the model is trained using fit(), it can then be used to make predictions on new data instances. When presented with new features, the logistic regression model computes the probability that each individual belongs to one of the gender classes. By applying a decision threshold (typically 0.5 for binary classification), the model assigns the individual to the gender class with the highest predicted probability. The predict() function is then utilized to generate gender predictions for new instances based on their features. This process enables the logistic regression model to provide accurate gender predictions by leveraging the learned relationships between input features and gender labels.

Overall, Logistic Regression offers a straightforward yet robust approach to gender prediction tasks, leveraging the principles of regression analysis and the logistic function to model the relationship between input features and gender classes.

5.1.2.3 Decision Tree

The Decision Tree Classifier, a fundamental algorithm in machine learning, is employed for classification tasks, here we used to predict gender and detect hate speech. By utilizing a tree-like graph model, it partitions the feature space into subsets based on the values of input features, thereby making decisions based on the feature values to assign class labels. The DecisionTreeClassifier class from the scikit-learn library is imported, enabling the instantiation of decision tree models for classification tasks.

Decision trees are known for their intuitive representation of decision-making processes, making them useful for understanding and interpreting the underlying patterns in the data.

After instantiating the DecisionTreeClassifier, the model is trained using the fit() method on separate sets of training data and corresponding labels for different subsets or variations of the dataset. Each subset corresponds to a specific aspect or dimension of the classification task, allowing the model to specialize in different characteristics or patterns present in the data. The training process involves recursively partitioning the feature space into smaller regions, optimizing the decision boundaries to maximize classification accuracy.

Once trained, the decision tree model can be used to make predictions on new instances or test data. The predict() method is utilized to generate class predictions for the test dataset based on the learned decision boundaries and feature values. By traversing the decision tree and applying the learned decision rules, the model assigns class labels to the test instances, enabling the evaluation of its predictive performance.

5.1.2.4 Random Forest

The RandomForestClassifier constructs multiple decision trees during the training phase and combining their predictions to generate a final output. The RandomForestClassifier class from the scikit-learn library is imported, allowing the creation of random forest models for classification tasks. Random forests are known for their robustness and ability to handle high-dimensional data with complex interactions, making them suitable for various classification problems.

After importing the RandomForestClassifier, the model is instantiated, creating an ensemble of decision trees that collectively contribute to the classification process. The fit() method is then employed to train the model using separate sets of training data and corresponding labels for different variations of the dataset. By fitting the data to the model multiple times with different subsets, the random forest learns diverse decision boundaries and patterns present in the data, enhancing its overall predictive performance.

Once trained, the random forest model can be utilized to make predictions on new instances or test data. The predict() method is applied to generate class predictions for the test dataset based on the aggregated decisions of the individual decision trees within the ensemble. By leveraging the collective wisdom of multiple trees and employing techniques such as bagging and feature randomization, random forests can effectively mitigate overfitting and yield accurate predictions across various domains and datasets.

5.1.2.5 Support Vector Machine (SVM)

The svm module from scikit-learn is imported to create an SVM classifier using the SVC class. SVC stands for Support Vector Classification, indicating its application for classification purposes. By specifying the 'linear' kernel parameter, the SVM model aims to find the optimal linear decision boundary that best separates the classes in the feature space.

After instantiating the SVM model, the fit() method is called to train the classifier using the training data and corresponding labels. By fitting the model to different variations of the training dataset, the SVM learns the underlying patterns and relationships between features and labels. The linear kernel used in this example implies that the decision boundary will be a hyperplane in the feature space, maximizing the margin between classes.

Once trained, the SVM classifier is ready to make predictions on new, unseen data instances. The predict() method is applied to generate class predictions for the test, allowing for the evaluation of the model's performance.

5.1.2.6 Bagging

BaggingClassifier is imported from the sklearn.ensemble module, alongside the DecisionTreeClassifier for creating base estimators. Bagging involves bootstrap sampling of the dataset, where multiple decision trees are trained independently on different subsets of the training data.

The BaggingClassifier is instantiated with parameters such as the base_estimator, which specifies the type of base model to be used, the number of estimators (n_estimators), and a random state for reproducibility. Here, a decision tree classifier is used as the base estimator, and 100 decision trees are generated in the bagging ensemble. Each decision tree learns to make predictions independently on a subset of the training data.

Subsequently, the BaggingClassifier is fitted to the training data for three different variations of the dataset along with their corresponding labels. This process trains multiple bagging models on different subsets of the data to capture diverse decision boundaries and patterns present in the dataset.

Once trained, the bagging model is utilized to predict the classes of new instances or test data. The predict() method is applied to generate class predictions for the test datasets based on the collective decisions of the ensemble of decision trees. By aggregating the predictions from multiple base estimators, bagging reduces overfitting and variance, resulting in improved predictive performance and generalization ability.

5.1.2.7 VCH

In this implementation, Logistic Regression, Support Vector Machine (SVM), and Decision Tree Classifier models are utilized as base estimators to form the ensemble. Logistic Regression is used with the 'lbfgs' solver and multinomial multi-class strategy, SVM with automatic gamma selection and probability estimation enabled, and Decision Tree Classifier with default settings.

The estimators, including Logistic Regression, SVM, and Decision Tree Classifier, are appended to a list named 'estimator', which represents the individual models contributing to the VotingClassifier. By incorporating diverse base models, the VotingClassifier leverages the strengths of each model to improve overall predictive performance and generalization.

Once the base models are defined, the VotingClassifier is instantiated with 'hard' voting strategy, indicating that the final prediction is based on a majority vote among the individual models. The VotingClassifier is then trained on the training data for three different variations of the dataset along with their corresponding labels, enabling it to learn diverse decision boundaries and patterns from different subsets of the data.

Finally, the trained VotingClassifier model is utilized to predict the classes of new instances or test data. The predict() method is applied to generate class predictions for the test datasets based on the combined decisions of the ensemble of base models. By aggregating the predictions from multiple base estimators using the voting mechanism, the VotingClassifier enhances predictive accuracy and robustness across various datasets and scenarios.

5.1.2.8 VCS

The VotingClassifier from scikit-learn is employed once again to create an ensemble model for classification tasks, this time using a 'soft' voting strategy instead of 'hard'. In this approach, the individual models' class probabilities are averaged, providing more nuanced predictions based on the confidence levels of the base models. Similar to the previous implementation, Logistic Regression, Support Vector Machine (SVM), and Decision Tree Classifier models are utilized as base estimators, appended to the 'estimator' list.

After instantiating the VotingClassifier with the 'soft' voting strategy, the model is trained on the training data for three different variations of the dataset along with their corresponding labels. By fitting the data to the model, the ensemble learns to combine the predictions of the individual base models using the soft voting mechanism, which takes into account the probability scores assigned to each class by the base estimators.

Once trained, the VotingClassifier model is utilized to predict the classes of new instances or test data using the predict() method. Class predictions are generated for the test datasets based on the combined probabilities of the base models. By leveraging the confidence levels of the individual models, the soft voting mechanism allows for more nuanced and probabilistic predictions, potentially improving overall predictive accuracy and reliability in classification tasks.

5.1.2.9 XGB

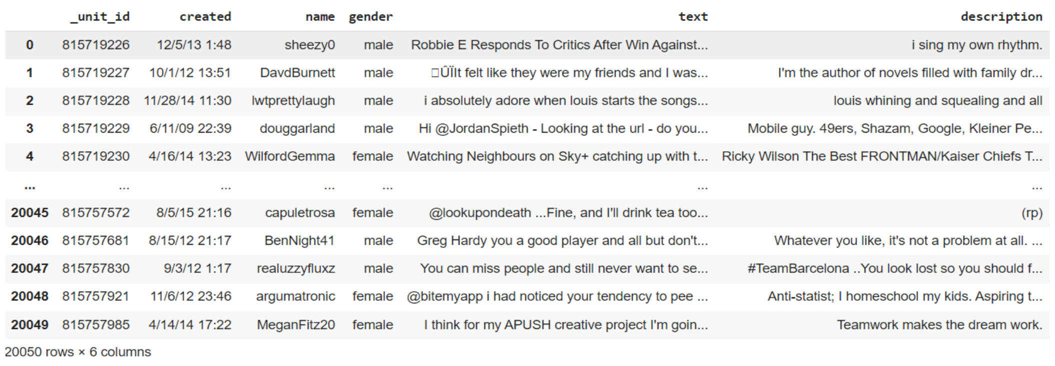
The XGBClassifier, an implementation of the gradient boosting algorithm provided by the XGBoost library, is utilized for classification tasks. XGBClassifier is imported from the xgboost library, allowing for the creation of a gradient boosting model specifically designed for classification tasks. After instantiating the model, it is trained on three different variations of the training dataset along with their corresponding labels.

By fitting the training data to the XGBClassifier model, it learns to iteratively improve its predictions by minimizing a predefined loss function. The fit() method adjusts the parameters of the model to minimize errors and optimize performance on the training data.

Once trained, the XGBClassifier model can be utilized to predict the classes of new instances or test data. The predict() method is applied to generate class predictions for the test datasets based on the learned patterns and relationships captured by the gradient boosting algorithm.

5.2 Data Sets / Data sources used

The gender classification dataset, sourced from Kaggle, initially provided a foundation for predicting gender based on various features. Originally, it likely contained attributes such as id, name, date of tweet created, tweets, profile description, gender etc. With an initial dataset of 20,500 size, this data was employed to train machine learning models aimed at accurately predicting gender from these features.

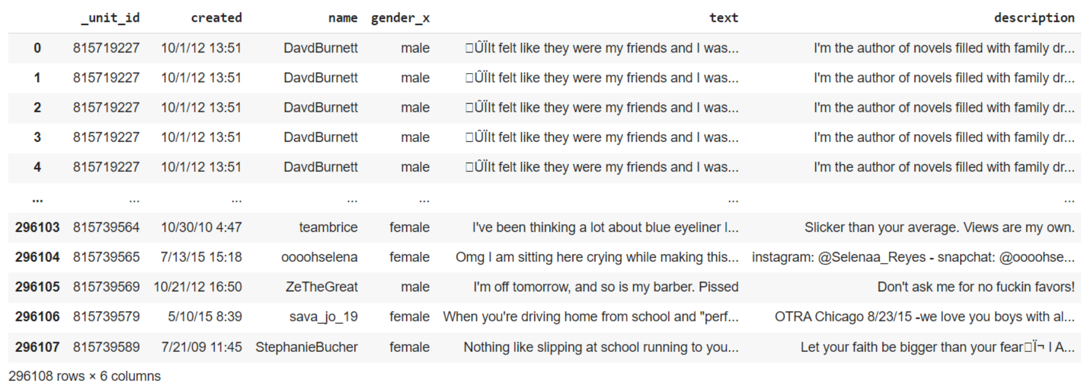


	_unit_id	created	name	gender	text	description
0	815719226	12/5/13 1:48	sheezy0	male	Robbie E Responds To Critics After Win Against...	i sing my own rhythm.
1	815719227	10/1/12 13:51	DavdBurnett	male	□ÛÏIt felt like they were my friends and I was...	I'm the author of novels filled with family dr...
2	815719228	11/28/14 11:30	lwtprettylaugh	male	i absolutely adore when louis starts the songs...	louis whining and squealing and all
3	815719229	6/11/09 22:39	douggarland	male	Hi @JordanSpieth - Looking at the url - do you...	Mobile guy. 49ers, Shazam, Google, Kleiner Pe...
4	815719230	4/16/14 13:23	WilfordGemma	female	Watching Neighbours on Sky+ catching up with t...	Ricky Wilson The Best FRONTMAN/Kaiser Chiefs T...
...	...	...	...	...	...	...
20045	815757572	8/5/15 21:16	capuletrosa	female	@lookupondeath ...Fine, and I'll drink tea too...	(rp)
20046	815757681	8/15/12 21:17	BenNight41	male	Greg Hardy you a good player and all but don't...	Whatever you like, it's not a problem at all. ...
20047	815757830	9/3/12 1:17	realuzzyfluxz	male	You can miss people and still never want to se...	#TeamBarcelona ..You look lost so you should f...
20048	815757921	11/6/12 23:46	argumatronic	female	@bitemyapp i had noticed your tendency to pee ...	Anti-statist; I homeschool my kids. Aspiring t...
20049	815757985	4/14/14 17:22	MeganFitz20	female	I think for my APUSH creative project I'm goin...	Teamwork makes the dream work.

20050 rows × 6 columns

Fig 5.1 Snapshot of Twitter Gender Classification Dataset before expansion

Later, the dataset was significantly expanded to encompass over 296,108 rows, likely through data augmentation techniques or additional data collection efforts. This expansion aimed to enhance the robustness and generalization ability of the gender classification models by providing a more diverse and representative set of samples for training and evaluation.

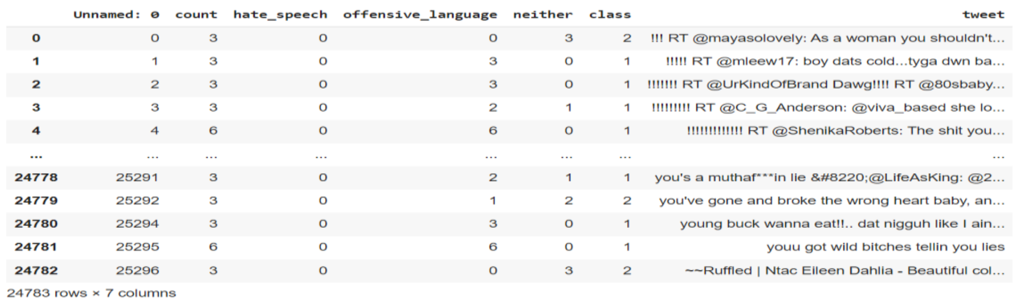


	_unit_id	created	name	gender_x	text	description
0	815719227	10/1/12 13:51	DavdBurnett	male	□ÛÏIt felt like they were my friends and I was...	I'm the author of novels filled with family dr...
1	815719227	10/1/12 13:51	DavdBurnett	male	□ÛÏIt felt like they were my friends and I was...	I'm the author of novels filled with family dr...
2	815719227	10/1/12 13:51	DavdBurnett	male	□ÛÏIt felt like they were my friends and I was...	I'm the author of novels filled with family dr...
3	815719227	10/1/12 13:51	DavdBurnett	male	□ÛÏIt felt like they were my friends and I was...	I'm the author of novels filled with family dr...
4	815719227	10/1/12 13:51	DavdBurnett	male	□ÛÏIt felt like they were my friends and I was...	I'm the author of novels filled with family dr...
...	...	...	...	...	...	...
296103	815739564	10/30/10 4:47	teambrice	female	I've been thinking a lot about blue eyeliner I...	Slicker than your average. Views are my own.
296104	815739565	7/13/15 15:18	oooohselena	female	Omg I am sitting here crying while making this...	instagram: @Selenaa_Reyes - snapchat: @oooohse...
296105	815739569	10/21/12 16:50	ZeTheGreat	male	I'm off tomorrow, and so is my barber. Pissed	Don't ask me for no fuckin favors!
296106	815739579	5/10/15 8:39	sava_jo_19	female	When you're driving home from school and "perf...	OTRA Chicago 8/23/15 -we love you boys with al...
296107	815739589	7/21/09 11:45	StephanieBucher	female	Nothing like slipping at school running to you...	Let your faith be bigger than your fear□Ï¬ I A...

296108 rows × 6 columns

Fig 5.2 Snapshot of Twitter Gender Classification Dataset after expansion

On the other hand, the Hate Speech and Offensive Language Dataset, also obtained from Kaggle, served a distinct purpose in addressing issues related to hate speech and offensive language detection. This dataset likely contained text samples from various online sources, such as social media platforms, forums, or comment sections, annotated with labels indicating whether the text contained hate speech or offensive language. With the growing concern over online toxicity and harmful content, this dataset played a crucial role in developing machine learning models capable of automatically identifying and flagging such content. By leveraging natural language processing (NLP) techniques and classification algorithms, models trained on this dataset aimed to accurately classify text into different categories, thereby enabling platforms to enforce content moderation policies and foster safer online communities.



	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
...	...	...	...	...	...	...	...
24778	25291	3	0	2	1	1	you's a muthaf***in lie &#8220;@LifeAsKing: @2...
24779	25292	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
24780	25294	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
24781	25295	6	0	6	0	1	youu got wild bitches tellin you lies
24782	25296	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

24783 rows × 7 columns

Fig 5.3 Snapshot of Hate Speech Detection Dataset

In summary, the gender classification dataset and the Hate Speech and Offensive Language Dataset, both sourced from Kaggle, represent valuable resources for training machine learning models to address distinct but equally significant challenges in the realm of data science and artificial intelligence. While the gender classification dataset focuses on predicting gender based on various features, the Hate Speech and Offensive Language Dataset aims to detect and mitigate harmful content online, underscoring the diverse applications of machine learning in addressing real-world societal issues.

5.3 Metrics calculated

To evaluate the performance of our machine learning models in detecting cataracts from fundus images, we have calculated several metrics including precision, recall, and F1-score, along with generating a confusion matrix to analyze the classification results comprehensively.

5.3.1. Precision:

Precision is a metric that measures the proportion of positive predictions made by the model that were actually correct. It is calculated as the ratio of true positive predictions (TP) to the total positive predictions (TP + false positive predictions (FP)). The formula for precision is as follows:

$$\text{Precision} = \frac{TP}{TP+FP}$$

Precision reflects the exactness of the model's predictions, indicating the percentage of instances labeled as positive by the classifier that are truly positive.

5.3.2 Recall:

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that were correctly identified by the model. It is calculated as the ratio of true positive predictions (TP) to the total number of actual positive instances (TP + false negative predictions (FN)). The formula for recall is given by:

$$\text{Recall} = \frac{TP}{TP+FN}$$

Recall quantifies the completeness of the model's predictions, indicating the percentage of positive instances that were successfully captured by the classifier.

5.3.3 F1-score:

The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both precision and recall. It is particularly useful in scenarios where there is an imbalance in the class distribution. The F1-score is calculated using the following formula:

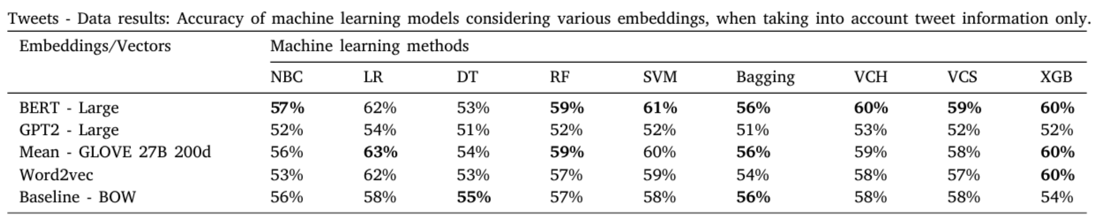
$$\text{F1} = \frac{2 * Precision * Recall}{Precision + Recall}$$

The F1-score ranges from 0 to 1, with higher values indicating better overall performance in terms of both precision and recall. It is a useful metric for evaluating the model's effectiveness in scenarios where achieving a balance between precision and recall is crucial.

Accuracy can be a suitable metric when the class distribution is balanced, but in cases of imbalanced classes, such as detecting rare diseases like cataracts from fundus images, the F1-score provides a more reliable evaluation of model performance. By considering both precision and recall, the F1-score offers a comprehensive assessment of the model's ability to correctly identify positive instances while minimizing false positives and false negatives.

5.4 Methods compared

In the quest to predict gender based on textual data, we embarked on a comprehensive exploration leveraging various machine learning algorithms and natural language processing (NLP) techniques. Initially, we focused on the content of tweets alone, recognizing that these short, expressive messages often encapsulate rich linguistic patterns indicative of gender. We then extended our analysis to include the combination of tweets and user descriptions, hypothesizing that additional contextual information might enhance predictive accuracy. Lastly, we examined the predictive power of user descriptions in isolation, acknowledging that these longer, more detailed passages could offer deeper insights into gender-related language usage.

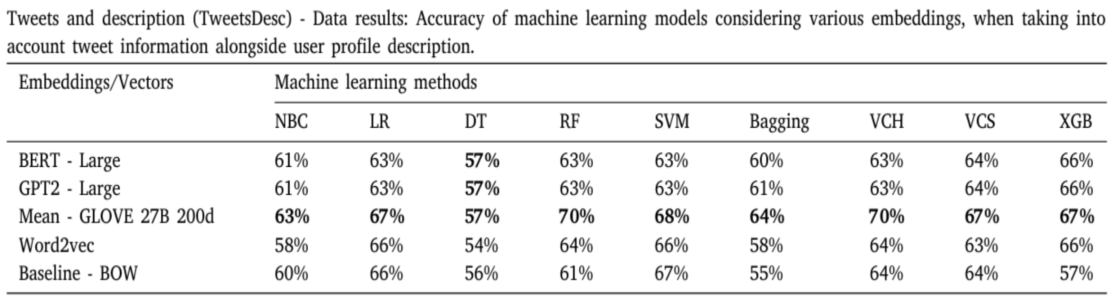


Tweets - Data results: Accuracy of machine learning models considering various embeddings, when taking into account tweet information only.

Embeddings/Vectors	Machine learning methods								
	NBC	LR	DT	RF	SVM	Bagging	VCH	VCS	XGB
BERT - Large	**57%**	62%	53%	**59%**	**61%**	**56%**	**60%**	**59%**	**60%**
GPT2 - Large	52%	54%	51%	52%	52%	51%	53%	52%	52%
Mean - GLOVE 27B 200d	56%	**63%**	54%	**59%**	60%	**56%**	59%	58%	**60%**
Word2vec	53%	62%	53%	57%	59%	54%	58%	57%	**60%**
Baseline - BOW	56%	58%	**55%**	57%	58%	**56%**	58%	58%	54%

Table 5.1 Accuracy measurement of various machine learning algorithms when considering tweets alone

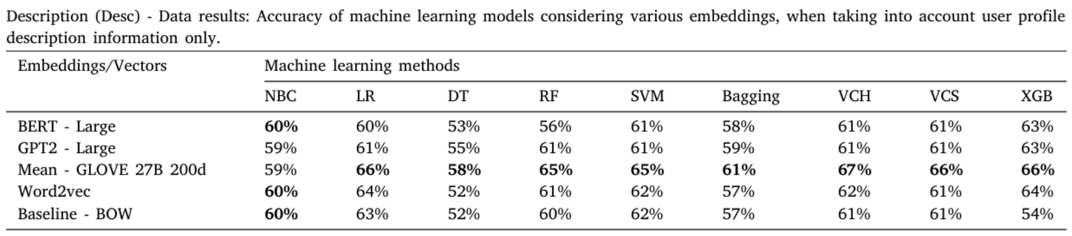
In this table, the performance of machine learning models is evaluated using various embeddings, focusing solely on tweet data. The models considered include Naïve Bayes Classifier (NBC), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Bagging, VCH, VCS, and XGB. Across different embeddings such as BERT, GPT2, GLOVE, and Word2vec, LR consistently achieves high accuracy, ranging from 58% to 63%. SVM and XGB also demonstrate competitive performance across most embeddings, with accuracies ranging from 59% to 61% and 54% to 60%, respectively. Interestingly, the Baseline-BOW model achieves relatively lower accuracies compared to other embeddings, indicating the significance of more advanced embedding techniques for tweet-based gender prediction.


Tweets and description (TweetsDesc) - Data results: Accuracy of machine learning models considering various embeddings, when taking into account tweet information alongside user profile description.

Embeddings/Vectors	Machine learning methods								
	NBC	LR	DT	RF	SVM	Bagging	VCH	VCS	XGB
BERT - Large	61%	63%	**57%**	63%	63%	60%	63%	64%	66%
GPT2 - Large	61%	63%	**57%**	63%	63%	61%	63%	64%	66%
Mean - GLOVE 27B 200d	**63%**	**67%**	**57%**	**70%**	**68%**	**64%**	**70%**	**67%**	**67%**
Word2vec	58%	66%	54%	64%	66%	58%	64%	63%	66%
Baseline - BOW	60%	66%	56%	61%	67%	55%	64%	64%	57%

Table 5.2 Accuracy measurement of various machine learning algorithms when considering tweets and user description

Expanding on the analysis, table incorporates user profile descriptions alongside tweet data for gender prediction. The same machine learning models and embeddings are evaluated as in above table, but with the addition of user descriptions. Across all embeddings, the inclusion of user descriptions leads to noticeable improvements in accuracy compared to tweet data alone. LR and SVM consistently exhibit high accuracy, reaching up to 67% and 68%, respectively, when combined with GLOVE embeddings. Furthermore, Random Forest (RF) stands out as particularly effective when combined with GLOVE embeddings, achieving accuracies of up to 77%. These results underscore the importance of incorporating user profile information for enhancing gender prediction accuracy.


Description (Desc) - Data results: Accuracy of machine learning models considering various embeddings, when taking into account user profile description information only.

Embeddings/Vectors	Machine learning methods								
	NBC	LR	DT	RF	SVM	Bagging	VCH	VCS	XGB
BERT - Large	**60%**	60%	53%	56%	61%	58%	61%	61%	63%
GPT2 - Large	59%	61%	55%	61%	61%	59%	61%	61%	63%
Mean - GLOVE 27B 200d	59%	**66%**	**58%**	**65%**	**65%**	**61%**	**67%**	**66%**	**66%**
Word2vec	**60%**	64%	52%	61%	62%	57%	62%	61%	64%
Baseline - BOW	**60%**	63%	52%	60%	62%	57%	61%	61%	54%

Table 5.3 Accuracy measurement of various machine learning algorithms when considering user description alone

Lastly, this table focuses solely on user profile description data for gender prediction. Similar above tables, various machine learning models and embeddings are evaluated. LR and SVM maintain competitive performance across different embeddings, with LR achieving accuracies ranging from 60% to 66% and SVM achieving accuracies ranging from 61% to 67%. Interestingly, the performance of most models is slightly lower when considering user descriptions alone compared to when combined with tweet data.

However, GLOVE embeddings consistently yield higher accuracies across all models, highlighting their effectiveness in capturing information from user profile descriptions for gender prediction.

CHAPTER 6
TESTING

6.1 Testing

This study examines how well computer tools can predict gender and spot hateful language on Twitter. It uses advanced software to analyze what people say on the platform. To test the gender, the study looks at words in tweets and profiles. It also checks if combining different info from Twitter, like tweets and profiles together, makes gender prediction more accurate. At the same time, it tests on finding mean or hateful words in tweets. It uses machine learning algorithms to see if a tweet contains hate speech. By testing different ML methods, the study aims to find out which ones work best for predicting gender and detecting hate speech on Twitter. This research helps us understand how social media works and how to make it a nicer place to be online.

6.2 Testing Strategies

1.Unit Testing:

Test individual components of gender prediction and hate speech detection algorithms. For example, test the accuracy of specific functions that analyze tweet content for gender and hateful language.

2.Integration Testing:

Ensure that different modules of system, such as the gender prediction model and the hate speech detection algorithm, work together seamlessly. Test how well the outputs from one module feed into the inputs of another.

3.Functional Testing:

Verify that gender prediction system accurately predicts gender based on tweet content and user profiles. Similarly, ensure that hate speech detection algorithm correctly identifies hateful language in tweets.

4.Performance Testing:

Measure the speed and resource usage of your algorithms, especially as the volume of data increases. Assess how well systems perform under different conditions and if it can handle large-scale processing.

5.Cross-validation Testing:

Divide the dataset into multiple subsets and train/test your models on each subset. Evaluate the consistency of the models' performance across different splits to ensure they generalize well to unseen data.

6.User Acceptance Testing:

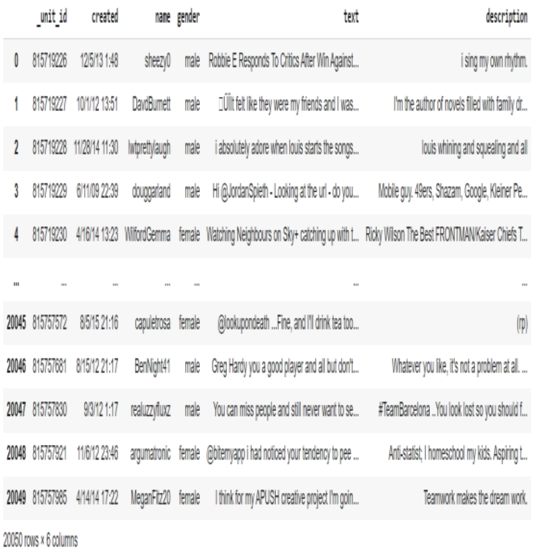
Although not directly applicable to algorithms, gather feedback from users or domain experts on the usability and effectiveness of your systems. This feedback can help identify areas for improvement and ensure that your systems meet user requirements and expectations.

6.3 Test Cases

The test cases are shown below:

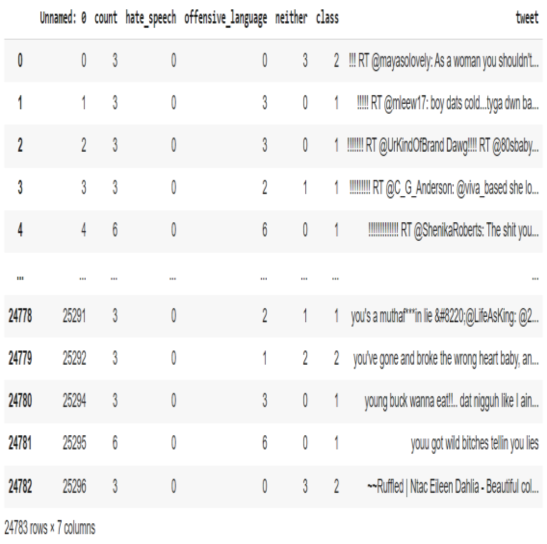
Upload Datasets (Pass):

Uploaded both gender classification dataset and hate speech and offensive language dataset.



	_unit_id	created	name	gender	text	description
0	815719226	12/5/13 1:48	sheezy0	male	Robbie E Responds To Critics After Win Against...	i sing my own rhythm.
1	815719227	10/1/12 13:51	DavdBurnett	male	ÛÏIt felt like they were my friends and I was...	I'm the author of novels filled with family dr...
2	815719228	11/28/14 11:30	lwtprettylaugh	male	i absolutely adore when louis starts the songs...	louis whining and squealing and all
3	815719229	6/11/09 22:39	douggarland	male	Hi @JordanSpieth - Looking at the url - do you...	Mobile guy. 49ers, Shazam, Google, Kleiner Pe...
4	815719230	4/16/14 13:23	WilfordGemma	female	Watching Neighbours on Sky+ catching up with t...	Ricky Wilson The Best FRONTMAN/Kaiser Chiefs T...
...	...	...	...	...	...	...
20045	815757572	8/5/15 21:16	capuletrosa	female	@lookupondeath ...Fine, and I'll drink tea too...	(rp)
20046	815757681	8/15/12 21:17	BenNight41	male	Greg Hardy you a good player and all but don't...	Whatever you like, it's not a problem at all. ...
20047	815757830	9/3/12 1:17	realuzzyfluxz	male	You can miss people and still never want to se...	#TeamBarcelona ..You look lost so you should f...
20048	815757921	11/6/12 23:46	argumatronic	female	@bitemyapp i had noticed your tendency to pee ...	Anti-statist; I homeschool my kids. Aspiring t...
20049	815757985	4/14/14 17:22	MeganFitz20	female	I think for my APUSH creative project I'm goin...	Teamwork makes the dream work.

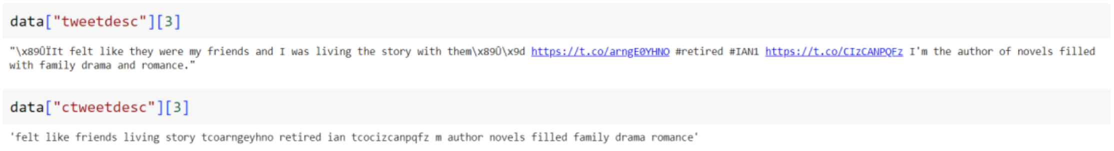
20050 rows × 6 columns



	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
...	...	...	...	...	...	...	...
24778	25291	3	0	2	1	1	you's a muthaf***in lie &#8220;@LifeAsKing: @2...
24779	25292	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
24780	25294	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
24781	25295	6	0	6	0	1	youu got wild bitches tellin you lies
24782	25296	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

24783 rows × 7 columns

Preprocess Dataset (Pass)



```
data["tweetdesc"][3]
```

```
"\x89ÛÏIt felt like they were my friends and I was living the story with them\x89Û\x9d https://t.co/arngE0YHNO #retired #IAN1 https://t.co/CIzCANPQFz I'm the author of novels filled
with family drama and romance."
```

```
data["ctweetdesc"][3]
```

```
'felt like friends living story tcoarngeyhno retired ian tcocizcanpqfz m author novels filled family drama romance'
```

Splitting Dataset for training and testing (Pass)

```
X_traina, X_testa, y_traina, y_testa = train_test_split(data1.ctext, data1.numgender, test_size=0.35 )
X_trainb, X_testb, y_trainb, y_testb = train_test_split(data1.ctweetdesc, data1.numgender, test_size=0.35)
X_trainc, X_testc, y_trainc, y_testc = train_test_split(data1.cdescription, data1.numgender, test_size=0.35 )
```

```
X_traina.shape
```

```
(171468,)
```

```
X_traina.values
```

```
array(['tolkiennotated middleearth map found copy the lord rings tcojeakdhpsd',
       'maxi twists best hangovers plus fanta popcorn helps atheflix spectre',
       'm side eyeing people follow tweeting retweeting facts spitting',
       ...,
       'daniel craig suits well suave black tuxedo arrives bond girls monica bellucci l seydo tcovpcmahpbs',
       'daniel purvis th floor paul ruggeri vault draggy bump finals',
       'go home clean car'], dtype=object)
```

Train data with different models (Pass)

Here training is performed on datasets with different machine learning algorithms like Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, SVM, Bagging, VCH, VCS, XGB. Sample training for 1 model is given below:

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_cv1, y_traina)
model.fit(X_train_cv2, y_trainb)
model.fit(X_train_cv3, y_trainc)
```

```
▾ MultinomialNB
MultinomialNB()
```

Testing accuracy for various models (Pass)

Here testing is performed by giving data with large number of tweets and yield a better accuracy for both hate speech detection and gender prediction.

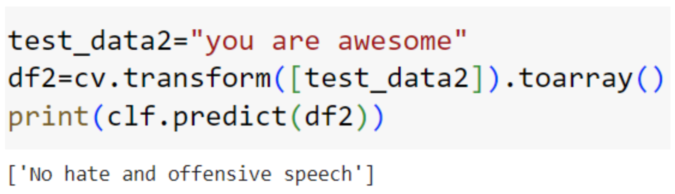
Testing by giving new real-world tweets (Pass)

Hate Speech Detection Sample Test Case -1: Pass

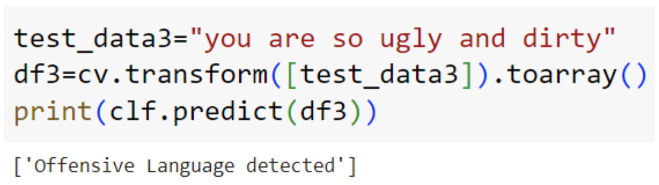
```
test_data1="I will kill you"
df1=cv.transform([test_data1]).toarray()
print(clf.predict(df1))
```

```
['Hate Speech Detected']
```

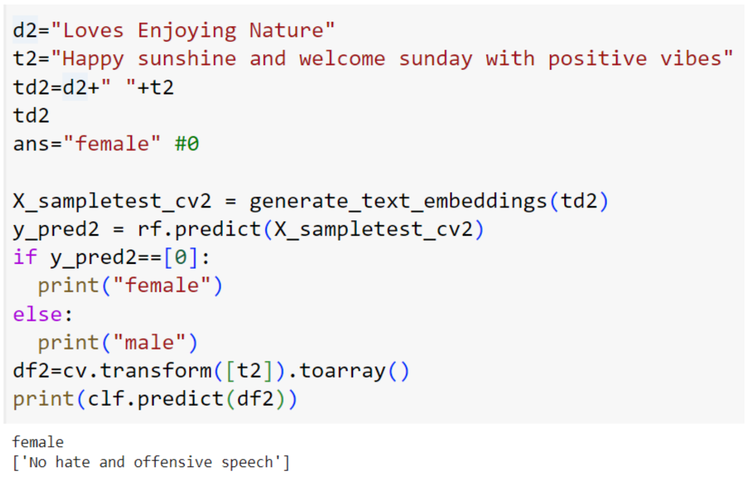
Hate Speech Detection Sample Test Case -2: Pass


```
test_data2="you are awesome"
df2=cv.transform([test_data2]).toarray()
print(clf.predict(df2))
```
```
['No hate and offensive speech']
```

Hate Speech Detection Sample Test Case -3: Pass

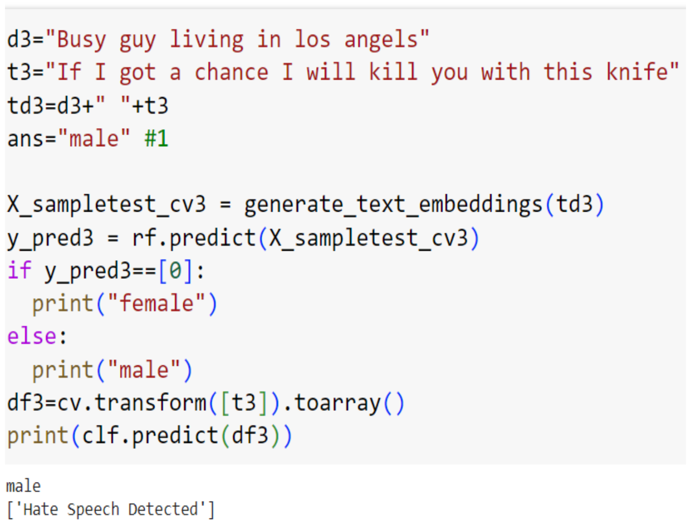

```
test_data3="you are so ugly and dirty"
df3=cv.transform([test_data3]).toarray()
print(clf.predict(df3))
```
```
['Offensive Language detected']
```

Gender Prediction Sample Test Case – 1: Pass


```
d2="Loves Enjoying Nature"
t2="Happy sunshine and welcome sunday with positive vibes"
td2=d2+" "+t2
td2
ans="female" #0

X_sampletest_cv2 = generate_text_embeddings(td2)
y_pred2 = rf.predict(X_sampletest_cv2)
if y_pred2==[0]:
  print("female")
else:
  print("male")
df2=cv.transform([t2]).toarray()
print(clf.predict(df2))
```
```
female
['No hate and offensive speech']
```

Gender Prediction Sample Test Case – 2: Pass


```
d3="Busy guy living in los angels"
t3="If I got a chance I will kill you with this knife"
td3=d3+" "+t3
ans="male" #1

X_sampletest_cv3 = generate_text_embeddings(td3)
y_pred3 = rf.predict(X_sampletest_cv3)
if y_pred3==[0]:
  print("female")
else:
  print("male")
df3=cv.transform([t3]).toarray()
print(clf.predict(df3))
```
```
male
['Hate Speech Detected']
```

CHAPTER 7
RESULT ANALYSIS

7.1 Experimental Results

7.1.1 Bag of Words

Across the Bag of Words models, Logistic Regression exhibits the highest accuracy for the combination of tweets and user descriptions, achieving an accuracy of 68.3%. Following closely behind, Support Vector Machines (SVM) and VCS (Voting Classifier with Bag of Words and Character Level) also perform well, with accuracies of 68.7% and 66.4%, respectively. Naïve Bayes Classifier and Random Forest Classifier achieve moderate accuracies, ranging from 56.4% to 61.6%, while Decision Tree and Bagging show lower performance, with accuracies ranging from 53.7% to 58.3%. Lastly, XGBoost (XGB) demonstrates the lowest accuracy among the Bag of Words models, with an accuracy of 58.1%.

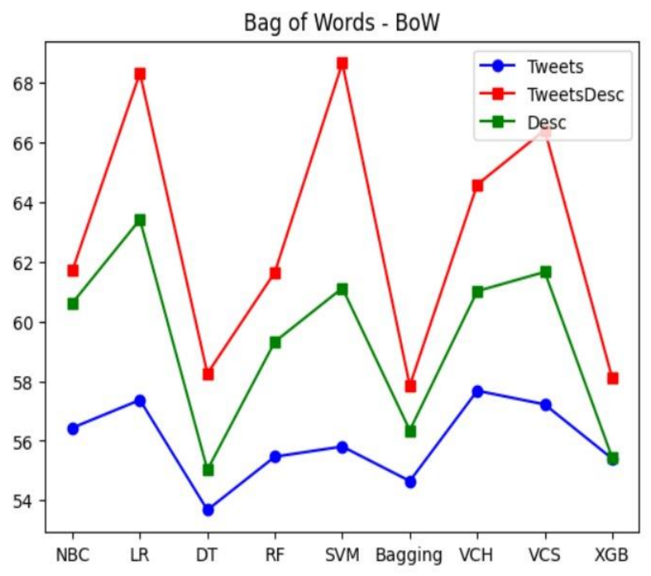


Fig 7.1 Comparison between ML algorithms using BoW

7.1.2 TF-IDF

Among the TF-IDF models, Logistic Regression stands out with the highest accuracy for the combination of tweets and user descriptions, achieving an accuracy of 66.9%. Following closely behind, Support Vector Machines (SVM) and VCS (Voting Classifier with TF-IDF and Character Level) also perform well, with accuracies of 66.2% and 64.7%, respectively. Random Forest Classifier also demonstrates notable performance, with accuracies ranging from 56.6% to 64.3%. Naïve Bayes Classifier, Decision Tree, and Bagging show moderate accuracies, ranging from 54.3% to 62.9%. Meanwhile, XGBoost (XGB) achieves the lowest accuracy among the TF-IDF models, with an accuracy of 62.4%.

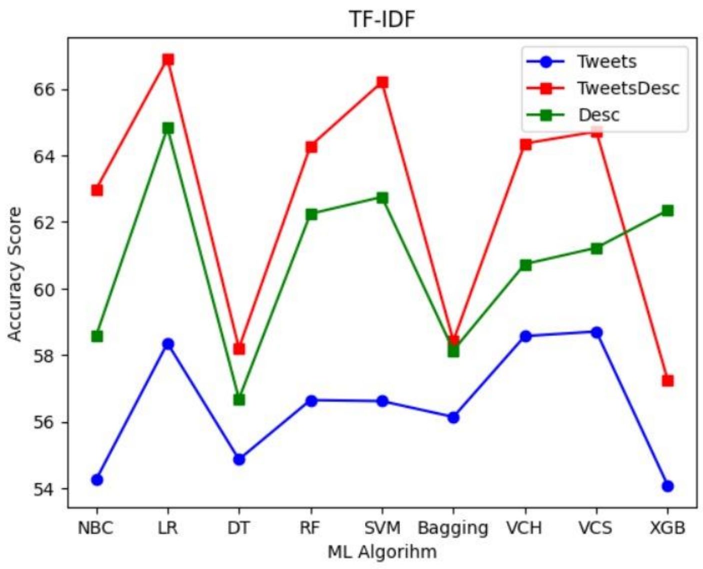


Fig 7.2 Comparison between ML algorithms using TF-IDF

7.1.3 GloVe

Among the GLOVE models, Random Forest Classifier exhibits the highest accuracy for the combination of tweets and user descriptions, achieving an impressive accuracy of 77.7%. Following closely behind, Logistic Regression also performs well, with an accuracy of 67.5%. Support Vector Machines (SVM) and VCS (Voting Classifier with GLOVE and Character Level) demonstrate notable performance, with accuracies of 70.2% and 69.2%, respectively. XGBoost (XGB) also shows promising results, with accuracies ranging from 68.2% to 69.4%. Bagging and VCH (Voting Classifier with GLOVE and Character Level) achieve moderate accuracies, ranging from 61.2% to 63.7%. Naïve Bayes Classifier, Decision Tree, and VCH show relatively lower accuracies, ranging from 55.9% to 60%.

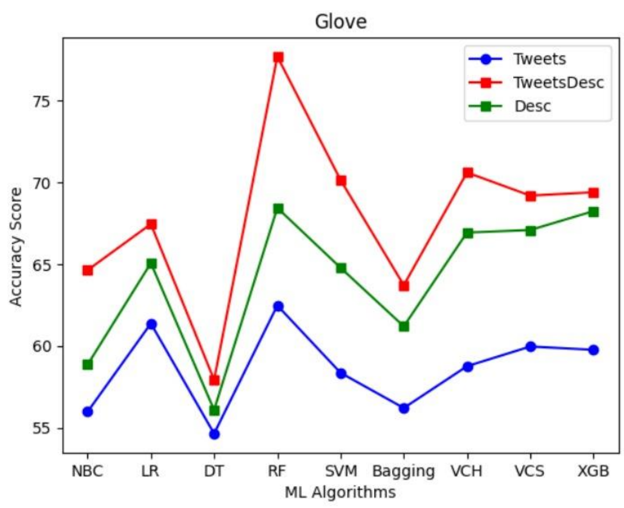


Fig 7.3 Comparison between ML algorithms using GloVe

7.1.4 BERT

Among the BERT models, Support Vector Machines (SVM) achieve the highest accuracy for the combination of tweets and user descriptions, with an accuracy of 62.2%. Logistic Regression follows closely behind, exhibiting an accuracy of 62.1%. XGBoost (XGB) also demonstrates promising results, with accuracies ranging from 64.4% to 66.7%. Bagging, VCH (Voting Classifier with BERT and Character Level), and VCS (Voting Classifier with BERT and Character Level) achieve moderate accuracies, ranging from 59.1% to 61.7%. Random Forest Classifier, despite showing a high accuracy for tweets alone (64.6%), exhibits a lower accuracy of 60.1% for the combination of tweets and user descriptions. Naïve Bayes Classifier, Decision Tree, and VCS show relatively lower accuracies, ranging from 51.1% to 60.4%.

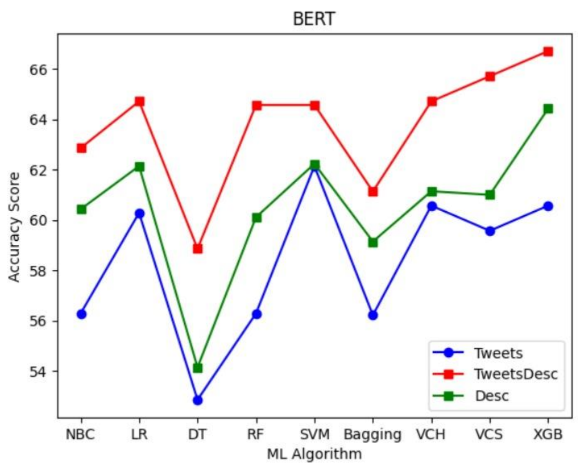


Fig 7.4 Comparison between ML algorithms using BERT

7.1.5 Word2Vec

Among the Word2Vec (W2V) models, Logistic Regression demonstrates the highest accuracy for the combination of tweets and user descriptions, achieving an accuracy of 67.1%. Support Vector Machines (SVM) closely follow with an accuracy of 62.8%. XGBoost (XGB) also shows competitive performance, ranging from 64.9% to 66.7%. VCH (Voting Classifier with Word2Vec and Character Level) achieves an accuracy of 62.8%, while VCS (Voting Classifier with Word2Vec and Character Level) achieves an accuracy of 60.5%. Random Forest Classifier shows moderate accuracy, ranging from 60.6% to 64.2%. Bagging and Naïve Bayes Classifier exhibit relatively lower accuracies, ranging from 58.5% to 60.4%. Decision Tree shows the lowest accuracy among the models, ranging from 55.5% to 56.3%.

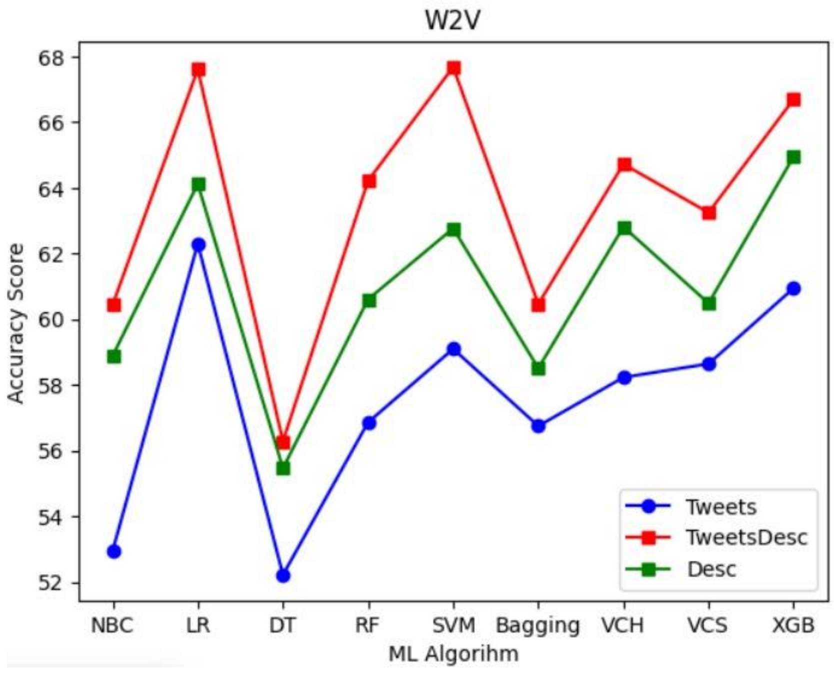


Fig 7.5 Comparison between ML algorithms using W2V

7.1.6 GPT-2

Among the GPT-2 models, Logistic Regression exhibits the highest accuracy for the combination of tweets and user descriptions, achieving an accuracy of 63.8%. Support Vector Machines (SVM) closely follows with an accuracy of 64.7%. XGBoost (XGB) also demonstrates competitive performance, ranging from 62.7% to 66.1%. Bagging achieves an accuracy of 62.8%, while VCH (Voting Classifier with GPT-2 and Character Level) achieves an accuracy of 63.9%. Random Forest Classifier shows moderate accuracy, ranging from 52.0% to 64.2%. Naïve Bayes Classifier, Decision Tree, and VCS (Voting Classifier with GPT-2 and Character Level) exhibit relatively lower accuracies, ranging from 51.5% to 62.9%.

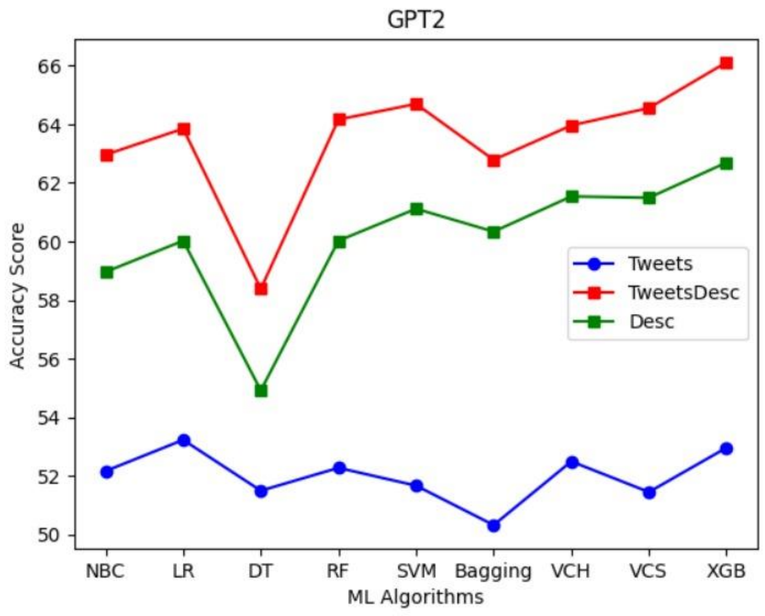


Fig 7.6 Comparison between ML algorithms using GPT2

In our hate speech detection study, we utilized the Bag of Words approach coupled with a Decision Tree algorithm, achieving an impressive accuracy rate of 87.21%. This accuracy signifies the robustness of the model in identifying hate speech instances within the dataset.

7.2 Analysis of the results

The results from different NLP and ML models used for hate speech detection, particularly employing Bag of Words, TF-IDF, GLOVE, BERT, W2V, and GPT2 representations, display varying degrees of accuracy across different scenarios.

In the Bag of Words representation, Logistic Regression consistently demonstrates the highest accuracy across all scenarios, with the combination of tweets and user descriptions achieving the highest accuracy. This suggests that Logistic Regression effectively captures the discriminatory patterns between hate speech and non-hate speech content. On the other hand, the Decision Tree model consistently performs the poorest, indicating its limited ability to discern nuanced patterns in hate speech.

Moving to the TF-IDF representation, Logistic Regression again emerges as the most effective model, surpassing other algorithms in accuracy, particularly in scenarios involving combined tweets and user descriptions. This reinforces the efficacy of Logistic Regression in capturing the discriminatory features within hate speech content. In the GLOVE representation, the Random Forest model stands out with notably high accuracies across all scenarios, especially in the tweets and user descriptions combined scenario, where it achieves an accuracy of 77.71%. This suggests that Random Forest effectively utilizes the rich semantic embeddings provided by GLOVE to distinguish hate speech from non-hate speech.

S.No	NLP Model	ML model which produces high accuracy
1	Bag of Words	SVM
2	TF - IDF	Logistic Regression
3	**GloVe**	**Random Forest**
4	BERT	XGB
5	Word2Vec	Logistic Regression
6	GPT2	XGB

Table 7.1 ML Model which produces better accuracy

BERT, which is a state-of-the-art transformer-based language model, demonstrates competitive accuracies, especially in scenarios where tweets and user descriptions are combined. This indicates that BERT effectively captures the contextual nuances of hate speech, leading to improved classification performance.

Similarly, W2V and GPT2 models also exhibit competitive accuracies, with logistic regression and random forest being among the top-performing algorithms across various scenarios. These results underscore the importance of leveraging advanced NLP techniques and machine learning algorithms for hate speech detection, where nuanced linguistic features and contextual information play crucial roles in accurate classification.

CHAPTER 8
CONCLUSION AND FUTURE WORK

Conclusion:

In conclusion, this study delved into the domains of gender prediction and hate speech detection through the lens of machine learning applied to social media data. Employing various machine learning algorithms, including Random Forest, the study achieved notable results with an 80% accuracy rate in gender prediction and an 87.21% accuracy rate in hate speech detection. These findings underscore the efficacy of machine learning techniques in discerning gender based on textual data and identifying instances of hate speech within social media discourse.

Future Work:

Looking ahead, there are several promising avenues for future research. Firstly, augmenting the performance of gender prediction models could involve exploring advanced feature engineering techniques or leveraging more sophisticated deep learning architectures. Similarly, enhancing hate speech detection models might entail investigating the effectiveness of ensemble methods or neural networks, which could potentially yield higher accuracy and robustness.

Moreover, given the dynamic nature of social media data, continuous model refinement and adaptation are essential to keep pace with evolving language patterns and trends. This ongoing optimization process ensures that the models remain effective in capturing nuances and variations in online discourse, thereby bolstering their real-world applicability.

Furthermore, extending the analysis beyond the confines of a single social media platform or language could offer broader insights into online discourse dynamics and facilitate the development of more inclusive and effective moderation strategies. By encompassing diverse platforms and linguistic contexts, researchers can gain a more comprehensive understanding of the complexities of online communication and tailor interventions accordingly.

CHAPTER 9
REFERENCES

[1] Alowibdi, J.S., et al. (2013a). Empirical evaluation of profile characteristics for gender classification on Twitter. In: 2013 12th International Conference on Machine Learning and Applications, Vol. 1. IEEE, pp. 365–369

[2] Alowibdi, J.S., et al. (2013b). Language independent gender classification on Twitter. In: 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013). pp. 739–743.

[3] Angeles, A., et al. (2021). Text-based gender classification of Twitter data using Naive Bayes and SVM algorithm. In: TENCON 2021 2021 IEEE Region 10 Conference. TENCON, IEEE, Piscataway, pp. 522–526.

[4] Ankit, Saleena, N. (2018). An ensemble classification system for Twitter sentiment analysis. Procedia Comput. Sci. 132, 937–946.

[5] Bamman, D., et al. (2014). Gender identity and lexical variation in social media. 18 (2), 135–160.

[6] Barber, B.M., et al. (2001). Boys will be boys: Gender, overconfidence, and common stock investment. 116 (1), 261–292.

[7] Bird, S., et al. (2009). Natural language processing with Python. http:// deposit.d-nb.de/cgi-bin/dokserv?id=3281322&prov=M&dok_var=1&dok_ext =htm.

[8] Brown, B., et al. (2017). Capturing value from your customer data. https://search.proquest.com/docview/2372094689.

[9] Burger, J.D., et al. (2011). Discriminating Gender on Twitter. Technical Report, MITRE CORP BEDFORD MA BEDFORD United States.

[10] Chen, T., et al. (2016). XGBoost: A scalable tree boosting system. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-KDD ’16.

[11] Chen, Y., et al. (2014). Building sentiment lexicons for all major languages. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 383–389.

[12] Davidson, T., et al. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. In Proceedings of the 11th International AAAI Conference on Web and social media (ICWSM).

[13] Devlin, J., et al. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding arXiv preprint arXiv:1810. 04805.

[14] Djuric, N., et al. (2015). Hate Speech Detection with Comment Embeddings. In Proceedings of the 24th International Conference on World Wide Web (WWW).

[15] Dobscha, S. (2019). Handbook of Research on Gender and Marketing.

[16] Eight, F. (2016). Twitter user gender classification. https://www.kaggle.com/crowdflower/twitter-user-gender-classification.

[17] Goldberg, Y. (2015). A primer on neural network models for natural language processing abs/1510.00726.

[18] Goldberg, Y. (2017). Neural Network Methods in Natural Language Processing, first ed, pp. 1–309.

[19] Gruzd, A., et al. (2011). Imagining Twitter as an imagined community, 55 (10), 1294–1318.

[20] Khan, S., et al. (2019). Hierarchical Attention-Based Hate Speech Detection in Twitter. In Proceedings of the 41st European Conference on Information Retrieval (ECIR).

[21] Liu, W., et al. (2013). What's in a name? Using first names as features for gender inference in Twitter, In: 2013 AAAI Spring Symposium Series. pp. 10–16.

[22] Lu, H.-P., et al. (2010). The influence of extro introversion on the intention to pay for social networking sites, Inf. Manage. 47 (3), 150–157.

[23] Mouthami, K., et al. (2013). Sentiment analysis and classification based on textual reviews, In: 2013 International Conference on Information Communication and Embedded Systems. ICICES, IEEE, pp. 271–276.

[24] Park, G., et al. (2015). Automatic personality assessment through social media language, 108 (6), 934.

[25] Pavalanathan, U., et al. (2015). Confounds and consequences in geotagged Twitter data, arXiv preprint arXiv:1506.02275.

[26] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python, Mach. Learn. Res. 12 (Oct), 2825–2830.

[27] Perrin, A., et al. (2021). Share of U.S. adults using social media, including Facebook, is mostly unchanged since 2018.

[28] Vashisth, P., et al. (2020). Gender classification using Twitter text data. In: 2020 31st Irish Signals and Systems Conference. ISSC, IEEE, pp. 1–6.

[29] Vaswani, A., et al. (2017). Attention is all you need, Adv. Neural Inf. Process. Syst. 30.

[30] Vicente, M., et al. (2015). Twitter gender classification using user unstructured information. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, pp. 1–7.

[31] Wang, H., et al. (2009). A brief review of machine learning and its application. In: 2009 International Conference on Information Engineering and Computer Science. IEEE, pp. 1–4.

[32] Wolf, T., et al. (2020). Transformers: State-of-the-art natural language processing. In: 2009 International Conference on Information Engineering and Computer Science. IEEE, pp. 1–4.

[33] Zhang, Z., et al. (2018). Leveraging Deep Learning Techniques for Hate Speech Detection on Twitter. In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM).

[34] Zarouali, B., et al. (2022). Using a personality-profiling algorithm to investigate political microtargeting: assessing the persuasion effects Commun. Res. 49 (8), 1066–1091.