CYBER BULLING DETECTION USING MACHINE LEARNING

A PROJECT REPORT
PHASE II

Submitted by

MOHAMMED RISHAN P	20104120
NIVED PK	20104128
NARAYANAN K	20104806

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
in

COMPUTER SCIENCE AND ENGINEERING

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHNOLOGY
Approved by AICTE, New Delhi, Accredited with 'A++' Grade by NAAC
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Valley Campus, Pollachi Highway, Coimbatore – 641 032

MAY 2024

Hindusthan College of Engineering And Technology
Approved by AICTE, New Delhi, Accredited with 'A++' Grade by NAAC
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Valley Campus, Pollachi Highway, Coimbatore – 641 032

BONAFIDE CERTIFICATE

Certified that this project report **"CYBER BULLING DETECTION USING MACHINE LEARNING "** is the bonafide work of **"MOHAMMED RISHAN P (20104120), NIVED PK (20104128), and NARAYANAN K (20104806)"** who carried out the project work under my supervision.

SIGNATURE
Mrs.N.PRIYA. M.E.,
SUPERVISOR
Assistant Professor
Computer Science and Engineering,
Hindusthan College of Engineering and Technology, Coimbatore-32

SIGNATURE
Dr. S. SHANKAR, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Computer Science and Engineering,
Hindusthan College of Engineering and Technology,Coimbatore-32

Submitted for the Autonomous Institution Project Phase II Viva-Voce conducted on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We, hereby jointly declare that the project work entitled "**CYBER BULLING DETECTION USING MACHINE LEARNING**", submitted to the Autonomous Institution Project Phase II Viva voice – April 2024 in partial fulfilment for the award of the degree of "**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING**", is the report of the original project work done by us under the guidance of **Mrs.N.PRIYA M.E,** Assistant professor, Department of Computer science and Engineering, Hindusthan College of Engineering and Technology, Coimbatore

	NAME	SIGNATURE
1.	**MOHAMMED RISHAN P**	
2.	**NIVED PK**	
3.	**NARAYANAN K**	

I certify that the declaration made by the above candidates are true

Project Guide,

Ms.N.PRIYA,M.E,

Assistant Professor

Department of CSE,

Hindusthan College of Engineering and Technology, Coimbatore-32

ACKNOWLEDGEMENT

We take this opportunity to express our wholehearted thanks and our profound respect to all those who guided and inspired us in the completion of this project work.

We extend our sincere thanks to the founder and chairman of Hindusthan Educational and Charitable Trust **Shri. T.S.R. KHANNAIYANN** and the Managing Trustee, **Smt. SARASWATHI KHANNAIYANN** and Executive Trustee & Secretary **Mrs. PRIYA SATISH PRABHU** for providing essential infrastructure.

We would like to extend our sincere gratitude to our chief executive officer, **Dr. K. KARUNAKARAN PhD.,** whose visionary leadership and unwavering support have been the driving force behind our project's success.

We would like to reveal our profound thanks to our respected Principal, **Dr. J. JAYA M.Tech, Ph.D.,** who happens to be the striving force in all endeavors.

We would like to express our gratitude to our Head of the Department **Dr. S. SHANKAR M.E, Ph.D.,** for bringing out the project successfully and for strengthening the ray of hope.

We would like to express our sincere thanks and deep sense of gratitude to our Class Advisor and Project Coordinator / Supervisor **Ms. N.PRIYA, M.E.,** Associate Professor, Department of Computer Science and Engineering, for his valuable guidance, suggestions and constant encouragement which paved way forth successful completion of the project work

We are intended to all teaching and non-teaching staff members of the Department of Computer science engineering for their kind cooperation and encouragement.

We express our immense pleasure and thankfulness to all our department faculty members, technical staffs and friends who helped us for the successful completion of this project. We express our earnest gratitude to **Mr. ABUBACKER PALATHINGAL** and **Ms. ASIYA** Parents of MOHAMMED RISHAN P, **Mr. RAJAN PK** and **Ms. BEENA V** parents of NIVED PK, **Mr. PARAMESWRAN K** and **Ms. SREEJA KA** parents of NARAYANAN K, and our family members who encouraged and strengthened us in perilous path, encountered during our task.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	I
	LIST OF FIGURES	II
	LIST OF ABBREVATIONS	III
1	**INTRODUCTION**	1
2	**OBJECTIVES**	2
3	**IDEATION PHASE**	3
	3.1 Literature Survey	3
	3.2 Problem Statement	5
4	**PROJECT DESIGN PHASE 1**	6
	4.1 Existing System	6
	4.1.1 Disadvantages	7
	4.2 Proposed Solution	7
	4.2.1 Advantages	8
	4.3 Feasibility Study	8
	4.3.1 Economical Feasibility	8
	4.3.2 Technical Feasibility	8
	4.3.3 Social Feasibility	9
	4.3.4 Operational Feasibility	9
	4.4 System Architecture	9
	4.4.1 System Technical Architecture	11
5	**PROJECT DESIGN PHASE 2**	12
	5.1 System Specifications	12
	5.1.1 Hardware Requirements	12
	5.1.2 Software Requirements	12
	5.2 Data Flow Diagrams	13
	5.2.1 Data Flow Diagram	14
	5.3 Software Descriptions	14

5.3.1 Python 15
5.3.2 Tkinter 18
5.3.3 Matplotlib 19
5.3.4 Numpy 20
5.3.5 Pandas 21
5.3.6 Scikit-learn 21
5.4 Module Description 22
5.4.1 Load Data Module 22
5.4.2 Data pre-processing Module 23
5.4.3 Feature Selection Module 23
5.4.4 Training Module 23
5.4.5 Testing Module 23
5.4.6 Evaluation and Performance Module 23
6 PROJECT DEVELOPMENT PHASE 24
6.1 Project Development – 1 24
6.1.1 Use case Diagrams 24
6.1.2 Sequence Diagrams 25
6.1.3 Flow Diagrams 26
6.2 Project Development – 2 27
6.2.1 Sample screen of system 29

7	**TESTING**	31
	7.1 System Testing	31
	7.2 Black Box and White Box Testing	33
	7.3 Unit Testing	33
	7.4 Integration Testing	33
	7.5 Verification and Validation	34
8	**CONCLUSION**	35
9	**FUTURE SCOPE**	36
10	**APPENDEX**	37
	REFERENCE	69
	JOURNAL PUBLICATION	71

ABSTRACT

Cyberbullying has emerged as a pervasive and concerning issue on social media platforms, impacting the mental health and well-being of individuals worldwide. To address this problem, this study proposes a cyberbullying detection system using the Support Vector Machine (SVM) algorithm. Leveraging the power of machine learning, the system aims to automatically identify and flag instances of cyberbullying in social media content. The development of the detection system begins with the collection and labelling of a comprehensive dataset containing examples of cyberbullying and non-cyberbullying posts or comments. After pre-processing the text data by removing irrelevant information, converting text to lowercase, and tokenizing it, meaningful features are extracted using the bag-of-words or TF-IDF techniques. These transformed feature vectors serve as inputs for training the SVM classifier, which seeks to find the optimal hyper plane for effectively distinguishing cyberbullying from non-cyberbullying content. The performance of the SVM model is evaluated using a separate testing dataset, with metrics such as accuracy, precision, recall, F1-score, and ROC-AUC analysed to assess its effectiveness in identifying cyberbullying instances. Model fine-tuning is conducted through experimentation with various SVM hyper parameters and cross-validation techniques to optimize performance

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
4.4.1	System Technical Architecture	11
5.2.1	Data Flow Diagram Level	14
6.1.1	Use Case Diagram	24
6.1.2	Sequence Diagram	25
6.1.3	Flow Diagram	26

LIST OF ABBREVATIONS

S.NO	ABBREVATIONS	EXPANSIONS
1	ML	Machine Learning
2	SVM	Support vector machine
3	NLP	Natural language Processing
4	BERT	Bidirectional encoder Representations from Transformers
5	EFA	Exploratory Factor Analysis
6	CFA	Confirmatory Factor Analysis
7	NB	Naïve Bayes
8	LR	Logistic Regression
9	LGBM	Light Gradient Boosting Machine
10	SGD	Stochastic Gradient Descent
11	RF	Random Forest
12	ADB	Ada Boost
13	ID	Interpersonal Deviance
14	COR	Conservation of Resource

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Cyberbullying, a pervasive issue in today's digital age, refers to the use of electronic communication to bully, harass, or intimidate others. It can take various forms, including mean comments, spreading rumors, sharing embarrassing images, and more, all conducted through online platforms such as social media, messaging apps, forums, and emails. With the anonymity and ubiquity of the internet, cyberbullying has become a significantconcern, particularly among adolescents and young adults, leading to psychologicaldistress, social isolation, and even tragic consequences like self-harm or suicide. Traditional methods of combating cyberbullying often rely on manual monitoring and intervention, which can be labor-intensive, time-consuming, and sometimes ineffective due to the sheer volume of online content. Machine learning (ML) offers a promising solution by automating the detection of cyberbullying instances, enabling faster response times and more comprehensive coverage of online spaces.

Machine learning algorithms analyze vast amounts of textual, visual, and behavioral data to learn patterns indicative of cyberbullying behavior. These patterns may include specific keywords, linguistic cues, sentiment analysis, image content analysis, and user interaction dynamics. By training on labeled datasets containing examples of cyberbullying and non-cyberbullying instances, ML models can identify subtle differences between benign and harmful online interactions.

CHAPTER 2
OBJECTIVES

The system should effectively distinguish cyberbullying from non-cyberbullying posts or comments, minimizing false positives and false negatives. The system should be capable of detecting cyberbullying in real-time, enabling prompt intervention and prevention of harm. The system should be able to detect cyberbullying across different languages, catering to a wider user base and addressing global cyberbullying concerns. The system should consider the context and intent of the text, going beyond keyword matching to capture nuanced cyberbullying instances. Data Collection: Gather a diverse dataset of text based interactions such as social media posts, comments, messages, etc., that encompass various forms of cyberbullying. Feature Engineering: Extract relevant features from the text data. These features may include word frequencies, sentiment analysis, linguistic patterns, and contextual information. Labeling: Annotate the dataset with labels indicating whether each instance contains cyberbullying behavior. This labeling can be done manually or through automated techniques, depending on the availability of resources. Deployment: Integrate the trained model into a real-world application or platform where it can automatically detect cyberbullying behavior in new, unseen text data.

CHAPTER – 3

IDEATION PHASE

3.1 LITERATURE SURVEY

Title 1: CYBERBULLYING AMONG TURKISH HIGH SCHOOL STUDENTS

Authors: BARIS CAGIRKAN

Has proposed in this paper Cyberbullying, a new form of the traditional bullying that has been transferred to the electronic environments (social media, online gaming environments, blogs, etc.), from the physical context to the virtual context, refers mainly to aggression that is deliberately carried out by adolescents. This study aims to measure the level of cyberbullying in Turkish high school students living in Eastern Turkey and identify the demographic and socio-economic factors which lead to being bully and being cyberbullied. The study population consists of 470 students aged from 15–19 years. Exploratory factor analysis (EFA) and confirmatory factor analysis (CFA) were implemented to identify the factor structure of the scale and it was observed that the Turkish version of the cyberbullying scale (CBS) is best represented by a one-factor structure. The comparisons across demographic and socio-economic variables were implemented using independent samples t test, one-way ANOVA, and Tukey HSD.

Title 2: ONLINE TIME, EXPERIENCE OF CYBER BULLYING AND PRACTICES TO COPE WITH IT AMONG HIGH SCHOOL STUDENTS IN HANOI

Authors: Pham Thi Lan Ch

Has proposed in this paperThis study is conducted to learn about experiences and practices to cope with cyberbullying among high school students in Hanoi and to explore the association between the average time of Internet used per day among high school students in Hanoi, Vietnam, and the risk of being cyberbullied. A total of 215 students

aged 13–18years completed an online survey using respondent-driven sampling method. The experience of being cyberbullied was examined using the modified Patchin and Hinduja's scale. The prevalence of experiencing at least one type of cyberbullying was 45.1%. The most common type of cyberbullying was being called by names/made fun of. The average daily time spent on Internet showed dose-response association with the risk of being cyberbulling.

Title3: A COMPARATIVE ANALYSIS OF MACHINE LEARNING TECHNIQUES FOR CYBERBULLYING DETECTION ON TWITTER

Authors: Amgad Muneer

Has proposed in this system, The advent of social media, particularly Twitter, raises many issues due to a misunderstanding regarding the concept of freedom of speech. One of these issues is cyberbullying, which is a critical global issue that affects both individual victims and societies. Many attempts have been introduced in the literature to intervene in, prevent, or mitigate cyberbullying; however, because these attempts rely on the victims' interactions, they are practical.

Title 4: PSYCHOLOGICAL, PHYSICAL, AND ACADEMIC CORRELATES OF CYBERBULLYING AND TRADITIONAL BULLYING

Author: Robin M. Kowalski

Has proposed in this systemBullying has long been present in schools, although awareness of the harms that bullying may cause is fairly recent . Bullying is commonly defined as acts of aggression that are repeated over time and that involve a power imbalance between the perpetrator and his or her targets. More recently, a new mode of bullying has emerged, known as cyberbullying

Title 5: COMPARISON AND CONTRAS.T OF PIAGET AND VYGOTSKY'S THEORIES

Author: Yu-Chia Huang

Has proposed in this paper Jean Piaget and Lev Vygotsky are the two most influential developmental psychologists. Their contributions to the field of developmental psychology, though different, are still similarly remarkable and unique. In spite of such resemblances, there exists a crucial, and generally unnoticed, the difference between Piaget's and Vygotsky's theories, and that this difference underlies the way each author addresses the concept of cognitive development.

3.2 PROBLEM STATEMENT

The problem at hand is to develop an effective and reliable cyberbullying detection system for social media platforms using Machine Learning (ML) techniques. With the rapid growth of social media usage, cyberbullying has become a pressing concern, causing emotional distress, social isolation, and even potential harm to victims. Therefore, an ML-based solution is sought to automatically identify instances of cyberbullying, allowing for timely interventions and fostering a safer online environment. The primary objective is to design a model capable of accurately classifying social media posts and comments as cyberbullying or non-cyberbullying, while addressing challenges such as data bias, false positives, false negatives, and the evolution of cyberbullying tactics and language trends. Furthermore, ensuring the system complies with privacy regulations and user acceptance is crucial for its successful deployment and positive impact on the well-being of social media users.

CHAPTER – 4
PROJECT DESIGN PHASE 1

4.1 EXISTING SYSTEM

Information and Communication Technologies fueled social networking and facilitated communication. However, cyberbullying on the platform had detrimental ramifications. The user-dependent mechanisms like reporting, blocking, and removing bullying posts online is manual and ineffective. Bag of-words text representation without metadata limited cyberbullying post text classification. This research developed an automatic system for cyberbullying detection with two approaches: Conventional Machine Learning and Transfer Learning. This research adopted AMICA data encompassing significant amount of cyberbullying context and structured annotation process. Textual, sentiment and emotional, static and contextual word embedding, psycholinguistics, term lists, and toxicity features were used in the conventional Machine Learning approach. This study was the first to use toxicity features to detect cyberbullying. This study is also the first to use the latest psycholinguistics features from the Linguistic Inquiry and Word (LIWC) 2022 tool, as well as Empath's lexicon, to detect cyberbullying. The contextual embedding of gilbert, TN Bert, and Distil Bert have alike performance, however Distil Bert embedding were elected for higher F-measure. Textual features, Distil Bert embedding, and toxicity features that struck new benchmark were the top three unique features when fed individually. The model's performance was boosted to F-measure of 64.8% after feeding with a combination of textual, sentiment, DistilBert embedding, psycholinguistics, and toxicity features to the Logistic Regression model that outperforms Linear SVC with faster training time and efficient handling of high-dimensionality features. Transfer Learning approach was by fine-tuning optimized version Pre-trained Language Models namely, Distil Bert, Distil Roberta, and Electra-small which were found to have speedier

training computation than their base form. The fine-tuned DistilBert resulted with the highest F-measure of 72.42%, surpassing CML.

4.1.1 DISADVANTAGES

ML models are only as good as the data they are trained on. If the training data contains biases or is not representative of the diverse ways cyberbullying can manifest, the model might perform poorly on real-world scenarios or unintentionally reinforce existing biases.

4.2 PROPOSED SOLUTION

The proposed system aims to develop an efficient and accurate cyberbullying detection solution for social media platforms. Leveraging the power of machine learning, the system will employ the Support Vector Machine (SVM) algorithm to automatically identify instances of cyberbullying in social media content. The process will begin with the collection and labeling of a diverse dataset containing either cyberbullying and non-cyberbullying posts or comments. Preprocessing techniques, including text cleaning, lowercasing, and tokenization, will be applied to transform the raw text data into a suitable format for feature extraction. The bag-of-words or TF-IDF techniques will then be employed to extract meaningful features from the preprocessed text data. These features will serve as inputs for training the SVM classifier, which will learn to distinguish between cyberbullying a The system's performance will be rigorously evaluated using various metrics, and fine-tuning will be performed to optimize its efficiency. Once trained and evaluated, the SVM-based cyberbullying detection system will be deployed to operate in on social media platforms, providing timely alerts and support to users facing potential cyberbullying incidents. By ensuring continuous monitoring and updating, nd non-cyberbullying content by finding an optimal hyperplane in the feature space.

4.2.1 ADVANTAGES

Leveraging the power of SVM and feature extraction techniques, the system achieves high accuracy in distinguishing between cyberbullying and non-cyberbullying content, reducing false positives and negatives.The proposed system's machine learning-based approach can easily scale to handle large volumes of social media data, making it suitable for deployment on various social media platforms with diverse user bases.

4.3 FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:Technical Feasibility Operation Feasibility.

4.3.1 ECONOMIC FEASIBILITY

The project's economic feasibility is evident in its potential to generate long-term cost savings and return on investment (ROI). By automating labor-intensive tasks such as model adaptation and prioritization, the project reduces the need for manual intervention, thereby minimizing labor costs. Furthermore, the system's proactive approach to handling evolving data patterns and threats can lead to significant savings by mitigating the risks associated with misclassifications and system downtime. Additionally, the project's scalability and compatibility with existing infrastructure ensure efficient resource utilization and cost-effectiveness over time.

4.3.2 TECHNICAL FEASIBILITY

From a technical perspective, the project demonstrates strong feasibility due to its reliance on established machine learning techniques and frameworks. Leveraging well-documented algorithms such as XGBoost for model prioritization ensures robustness and scalability. The

system's ability to integrate seamlessly with popular ML frameworks enhances its technical viability, enabling easy adoption and deployment across diverse environments. Moreover, the project's real-time monitoring and visualization capabilities leverage cutting-edge technologies to provide actionable insights and recommendations, further underscoring its technical feasibility.

4.3.3 SOCIAL FEASIBILITY

Socially, the project aligns with the growing demand for reliable and trustworthy AI solutions. By enhancing machine learning robustness and adaptability, the project contributes to building trust and confidence in AI technologies among stakeholders and end-users. Moreover, the project's focus on detecting and responding to evolving data poisoning attacks addresses concerns related to data integrity and security, thereby fostering a safer and more transparent digital ecosystem. By promoting responsible AI practices and accountability, the project enhances its social acceptance and relevance in today's data-driven society.

4.3.4 OPERATIONAL FEASIBILITY

Operationally, the project offers tangible benefits by streamlining machine learning model management and decision-making processes. Its user-friendly interfaces and intuitive dashboards empower stakeholders to monitor and manage models effectively, reducing operational complexity and enhancing efficiency. Additionally, the project's real-time insights and recommendations enable organizations to make data-driven decisions promptly, improving overall operational agility and responsiveness. Furthermore, the project's modular architecture and scalability ensure seamless integration into existing workflows, minimizing disruptions and maximizing operational feasibility.

4.4 SYSTEM ARCHITECTURE

Detecting cyberbullying using machine learning involves several steps and components. Here's a high-level overview of a system architecture for cyberbullying

detection: Data Collection: Gather data from various sources where cyberbullying occurs, such as social media platforms, forums, emails, etc. This data may include text, images, videos, or other multimedia content. Data Preprocessing: Text Data: Tokenization, removing stop words, stemming or lemmatization, handling emojis and special characters, etc. Image/Video Data: Feature extraction, resizing, normalization, etc. Feature Extraction: Text Data: Convert text into numerical feature vectors using techniques like TF-IDF, word embeddings (Word2Vec, GloVe), or deep learning-based methods like BERT. Image/Video Data: Extract features using techniques like CNNs (Convolutional Neural Networks) or pre-trained models like ResNet, VGG, etc. Model Training: Choose a suitable machine learning algorithm or deep learning architecture such as: Supervised Learning: Support Vector Machines (SVM), Random Forest, Gradient Boosting, etc. Deep Learning: Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Transformer models (like BERT), etc. Train the model on labeled data, where examples are labeled as either cyberbullying or non-cyberbullying. Model Evaluation: Evaluate the trained model using appropriate metrics such as accuracy, precision, recall, F1-score, ROC-AUC, etc. Perform cross-validation to ensure the model's generalization capability. Deployment: Integrate the trained model into a system or platform where cyberbullying detection is required. Develop APIs or services to interact with the model for real-time or batch processing. Post-Processing: Apply post-processing techniques to refine the model's predictions, such as thresholding, filtering, or ensemble methods. Monitoring and Updates: Continuously monitor the model's performance in production. Update the model periodically with new data and retrain it to adapt to evolving patterns of cyberbullying behavior. Feedback Loop: Incorporate feedback mechanisms to improve the model based on user input or manual review of flagged content. Privacy and Ethics: Ensure compliance with privacy regulations and ethical considerations, especially when dealing with sensitive user-generated content. Implement measures to protect user privacy and prevent misuse of the cyberbullying detection system.

4.4.1 PROPOSED SYSTEM TECHNICAL ARCHITECTURE

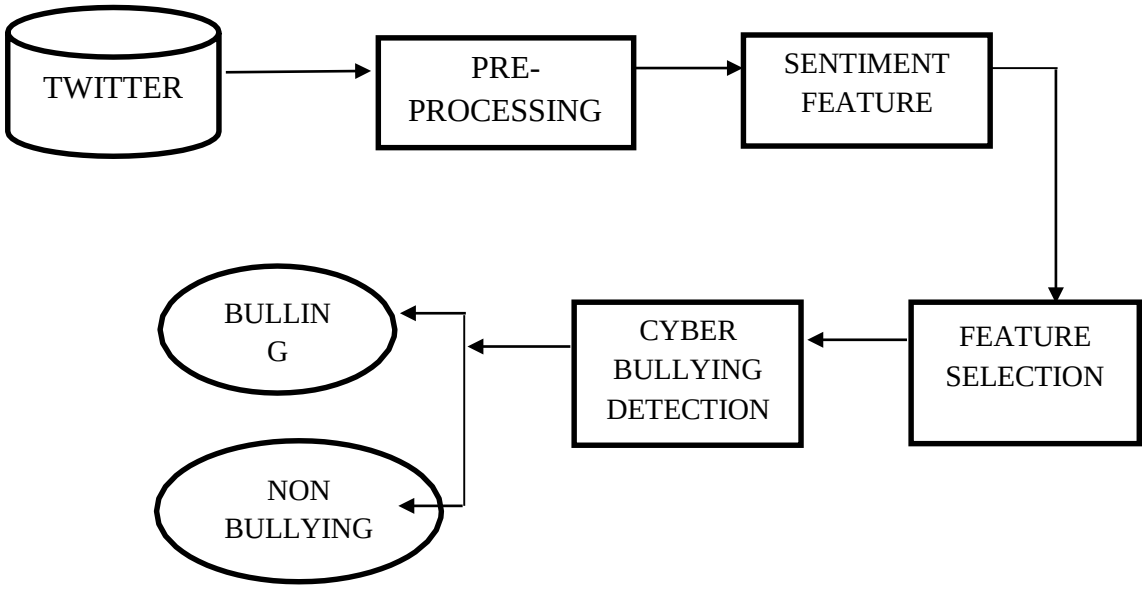


Fig 4.4.1

CHAPTER-5

PROJECT DESIGN PHASE 2

5.1 SYSTEM SPECIFICATION

5.1.1 HARDWARE REQUIREMENTS

Processor Type	- AMD RYZEN 7
Speed	- 4.40GHZ
RAM	-16 GB RAM
Hard disk	- 1 TB
Keyboard	- 101/102 Standard Keys
Mouse	- Optical Mouse

5.1.2 SOFTWARE REQUIREMENTS

0perating System	- Windows 10
Python Modules	- Tkinter, Matplotlib, Numpy
Python Version	- Python 3.x
Other Tools/Libraries	- Pandas, Sub process, Shutil
Script	- Shell Script
Front end	- Html, Css, Javascript

5.2 DATA FLOW DIAGRAMS

Creating a data flow diagram (DFD) for a cyberbullying detection system using machine learning involves illustrating the flow of data within the system, including inputs, processes, and outputs. Here's a basic outline of what such a diagram might look like: External Entities These are the individuals interacting with the system, such as social media platform users. Data Sources: This could include sources of data such as social media posts, comments, messages, etc. Processes Data Collection: Data from external sources is collected and fed into the system. This might involve web scraping or accessing APIs provided by social media platforms. Preprocessing: Raw data is cleaned and prepared for analysis. This may involve tasks like text normalization, removing stop words, and tokenization. Feature Extraction: Relevant features are extracted from the preprocessed data. These features could include sentiment analysis, frequency of certain words, linguistic patterns, etc. Model Training: Machine learning models are trained using the extracted features. This could involve algorithms like Naive Bayes, Support Vector Machines, or deep learning models like Recurrent Neural Networks (RNNs) or Transformers Model Evaluation: The trained model is evaluated using validation data to assess its performance and make any necessary adjustments. Detection: The trained model is used to detect instances of cyberbullying in new data.. Response Once cyberbullying is detected, appropriate actions may be taken, such as alerting moderators, notifying users, or implementing content restrictions. Data Stores a Training Data This is the historical data used to train the machine learning model. Model Parameters of the trained model are stored for later use in detection.

Detected Instances: Data about detected instances of cyberbullying may be stored for analysis or reporting purposes. Data Flow Input Data Flow Data flows from external sources to the data collection process, then through preprocessing, feature extraction, and into the model training process.

5.2.1 Data Flow Diagram

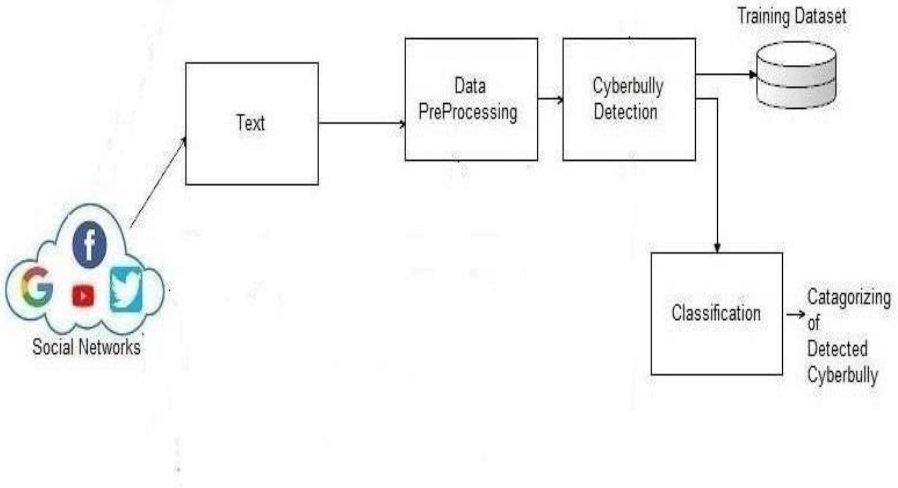


Fig 5.2.1

5.3 SOFTWARE DESCRIPTION

The Resume Builder Application is a web-based tool designed to assist users in creating, customizing, and generating professional resumes. It provides a user-friendly interface for inputting personal and professional details and offers a range of visually appealing templates to choose from. This application empowers users to build personalized resumes with ease while ensuring data security and customization options.

5.3.1 Python

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc. In this article we will see what characteristics describe the python programming language

Features of Python

In this section we will see what the features of Python programming language are:

1. Free and Open Source

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

2. Easy to code

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

4. Object-Oriented Language

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in Python. PyQt5 is the most popular option for creating graphical apps with Python.

6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. Large Community Support

Python has gained popularity over the years. Our questions are constantly answered by the enormous Stack Overflow community. These websites have already provided answers to many questions about Python, so Python users can consult them as needed.

8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

9. Python is a Portable language

Python language is also a portable language. For example, if we have Python code for Windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

10. Python is an integrated language

Python is also an integrated language because we can easily integrate Python with other languages like C, C++, etc.

11. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. Like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **byte code**.

12. Large Standard Library

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

14. Frontend and backend development

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python it's extensively used for this work cause of its frameworks like Django and Flask.

15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

5.3.2 Tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python tkinter is the fastest and easiest way to create GUI applications. Creating a GUI using tkinter is an easy task.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of tools and widgets to build desktop applications with buttons, menus, text boxes, labels, and more.

Features of Tkinter

1. **Widgets:** Tkinter offers various built-in widgets such as buttons, labels, text boxes, frames, canvas, and radio buttons, check buttons, etc. These widgets serve as the building blocks for constructing GUIs.
2. **Geometry Managers:** Tkinter includes three main geometry managers - **pack**, **grid**, and **place**. These managers help in arranging and positioning widgets within the application window.
3. **Event-Driven Programming:** Tkinter uses an event-driven programming model where actions like button clicks, mouse movements, keyboard inputs, etc., generate events. Developers can bind functions to these events to execute specific actions in response.
4. **Customization:** Developers can customize the appearance of widgets using various

configuration options like colors, fonts, sizes, and styles.

5. **Cross-platform:** Tkinter is cross-platform and comes pre-installed with Python, allowing developers to create applications that run on different operating systems without requiring additional installations.

5.3.3 Matplotlib

Matplotlib is a powerful plotting library in Python that enables the creation of various types of high-quality and customizable visualizations. It provides a wide range of functionalities to visualize data and generate publication-quality figures. It stands as a versatile Python library designed to craft static, interactive, and animated visualizations. Its widespread use encompasses data analysis, scientific computing, and intricate visual representation of complex datasets.

Offering an expansive suite of plotting options, Matplotlib empowers users to create an extensive spectrum of visuals, spanning line plots, scatter plots, bar charts, histograms, heat maps, contour plots, and more. Its hallmark lies in the extensive customization capabilities, granting fine-tuning control over colors, line styles, markers, labels, legends, axis scales, grid styles, and plot layout. Furthermore, Matplotlib excels in providing output in various formats like PNG, PDF, SVG, and EPS, making it conducive for publication, reports, presentations, or web integration. Its seamless integration with other Python libraries such as NumPy, Pandas, SciPy, Jupyter Notebooks, and GUI toolkits like Tkinter fortifies its utility. Additionally, Matplotlib's compatibility across multiple operating systems and environments ensures smooth functionality without necessitating significant code alterations.

Features of Matplotlib

1. Simple and Intuitive Interface: Provides an easy-to-use API for creating plots, making it accessible for both beginners and experienced users.
2. Versatile Plotting: Supports a wide variety of plot types and styles, catering to diverse

visualization requirements.

3. Publication-Quality Output: Generates high-resolution, publication-ready figures suitable for scientific papers, presentations, and reports.
4. Extensibility: Offers flexibility to extend functionality through customization, incorporating third-party plugins, and creating complex visualizations.
5. Matplotlib Gallery: Provides an extensive gallery of examples and code snippets showcasing its capabilities for different plot types and customizations, aiding users in learning and implementing various plot styles efficiently**.**

5.3.4 Numpy

NumPy stands for Numerical Python and is a fundamental library for numerical computing in Python. It provides support for multidimensional arrays and matrices along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is extensively used in scientific computing, data analysis, machine learning, and more.

Features of Numpy

1. **Arrays:** NumPy's primary feature is its **ndarray**, an N-dimensional array object that provides a fast and efficient way of storing and manipulating large arrays of numerical data.
2. **Mathematical Functions:** NumPy offers a wide range of mathematical functions that operate element-wise on arrays, such as trigonometric, statistical, linear algebra, Fourier transforms, etc.
3. **Broadcasting:** NumPy provides broadcasting, allowing operations between arrays of different shapes, which simplifies computation and eliminates the need for explicit looping.
4. **Integration with C/C++ and Fortran:** NumPy arrays are memory-efficient and can interface with code written in C, C++, or Fortran, which enhances performance.
5. **Random Number Capabilities:** It includes a robust random number generation library to generate random data or samples from various probability distributions.

5.3.5 Pandas

Pandas is an open-source library built on top of NumPy, designed for data manipulation, cleaning, analysis, and preparation. It offers data structures and operations for manipulating numerical tables and time series data, making it a powerful tool for data science and analysis tasks.

Features of Pandas:

1. **Data Frame:** Pandas introduces the Data Frame, a two-dimensional labeled data structure similar to a table or spreadsheet in Excel, making it easy to handle and analyze structured data.
2. **Data Alignment and Handling Missing Data:** Pandas provides tools to handle missing data efficiently, either by filling, dropping, or interpolating values. Additionally, it aligns data automatically based on labels.
3. **Data Operations and Manipulation:** Pandas enables powerful operations on data, including merging and joining datasets, reshaping, pivoting, slicing, indexing, and grouping data for aggregation and transformation.
4. **Time Series Data:** Pandas has robust support for working with time series data, providing tools for date range generation, frequency conversion, and time shifting.
5. **Input/output Tools:** It offers functionalities to read data from various file formats such as CSV, Excel, SQL databases, JSON, and more. Likewise, it provides methods to save data in these formats.

5.3.6 Scikit-Learn

Scikit-learn is a powerful machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It's built on top of other popular libraries such as NumPy, SciPy, and Matplotlib.

Features of Scikit-learn:

1. **Simple and Consistent API:** Scikit-learn offers a unified and consistent interface for various machine learning algorithms, making it easy to use and switch between different models.
2. **Supervised and Unsupervised Learning Algorithms:** It includes a wide range of algorithms for both supervised learning (classification, regression) and unsupervised learning (clustering, dimensionality reduction).
3. **Preprocessing and Feature Extraction:** Scikit-learn provides tools for data preprocessing, feature scaling, normalization, encoding categorical variables, and handling missing values.
4. **Model Evaluation and Selection:** It offers methods for model evaluation through various metrics like accuracy, precision, recall, F1-score, ROC curves, etc. It also includes utilities for model selection via cross-validation and hyper parameter tuning.
5. **Integration with Other Libraries:** Scikit-learn integrates well with other data science libraries like Pandas, NumPy, and Matplotlib, enabling seamless data manipulation, analysis, and visualization.
6. **Extensibility:** While providing a rich set of functionalities out-of-the-box, Scikit-learn allows easy integration of custom machine learning algorithms and extends its capabilities through third-party contributions

5.4 MODULE DESCRIPTION

5.4.1 Load Data Module:

This module is responsible for loading the labeled dataset containing social media posts or comments for training and testing the cyberbullying detection system. It reads the dataset from a file or database, extracting the text data and corresponding labels (cyberbullying or non-cyberbullying)

5.4.2 Data Pre-processing Module:

This module is designed to pre-process the raw text data to make it suitable for feature extraction and SVM classification. Clean the text data by removing special characters, URLs, and other irrelevant information. Convert the text to lowercase to ensure case insensitivity. Tokenize the text into individual words or tokens. Apply stemming or lemmatization to reduce words to their root form (optional).

5.4.3 Feature Selection Module:

This module performs feature extraction from the pre-processed text data, converting it into numerical feature vectors that the SVM can process. Utilize techniques like bag-of-words or TF-IDF to represent the text data as numerical vectors. Create feature matrices containing the transformed data, ready for training the SVM model.

5.4.4 Training Module:

This module is responsible for training the SVM classifier on the pre-processed and feature-selected data. Split the dataset into training and testing sets. Use the training set to train the SVM classifier with appropriate hyper parameters and kernelsettings.

5.4.5 Testing Module:

This module assesses the performance of the trained SVM classifier on unseen data. Use the testing set to evaluate the SVM classifier's performance in detecting cyberbullying instances. Calculate accuracy, precision, recall, F1-score, and ROC-AUC to evaluate the classifier's effectiveness.

5.4.6 Evaluation and Performance Module:

This module analyses the results obtained from the testing module to evaluate performance

CHAPTER - 6

PROJECT DEVELOPMENT PHASE

6.1 PROJECT DEVELOPMENT - 1

6.1.1 USE CASE DIAGRAM

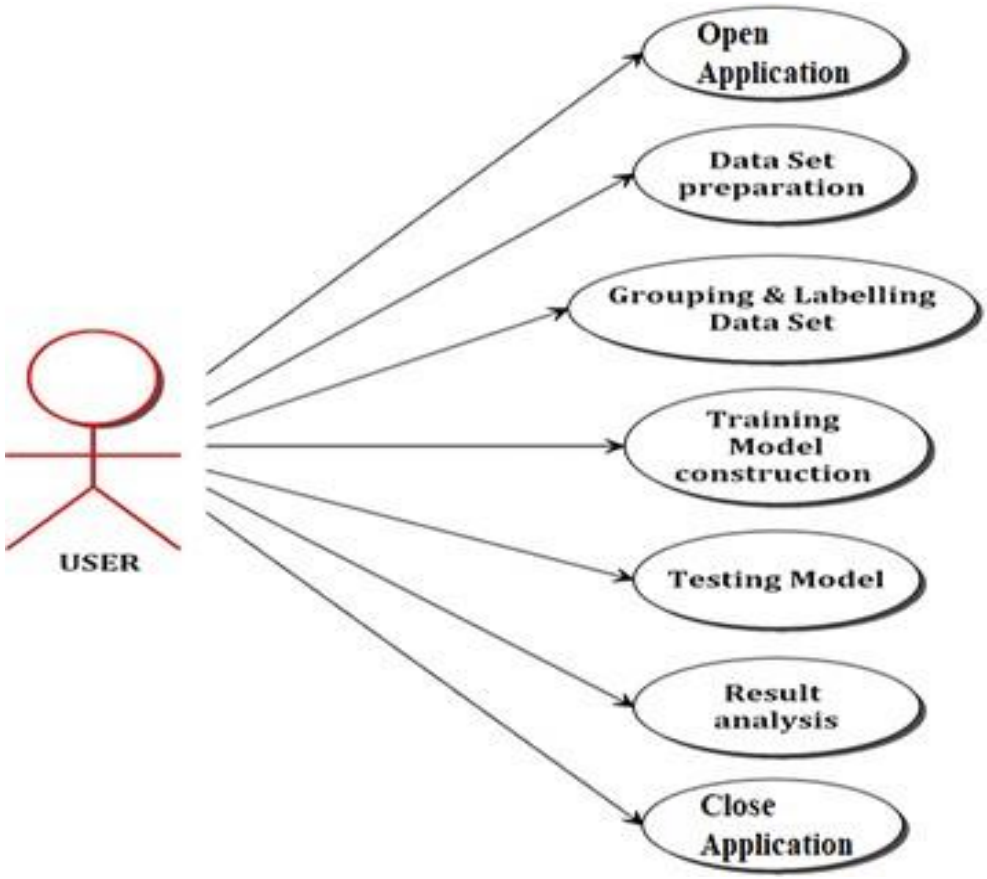


Fig 6.1.1

6.1.2 SEQUENCE DIAGRAM

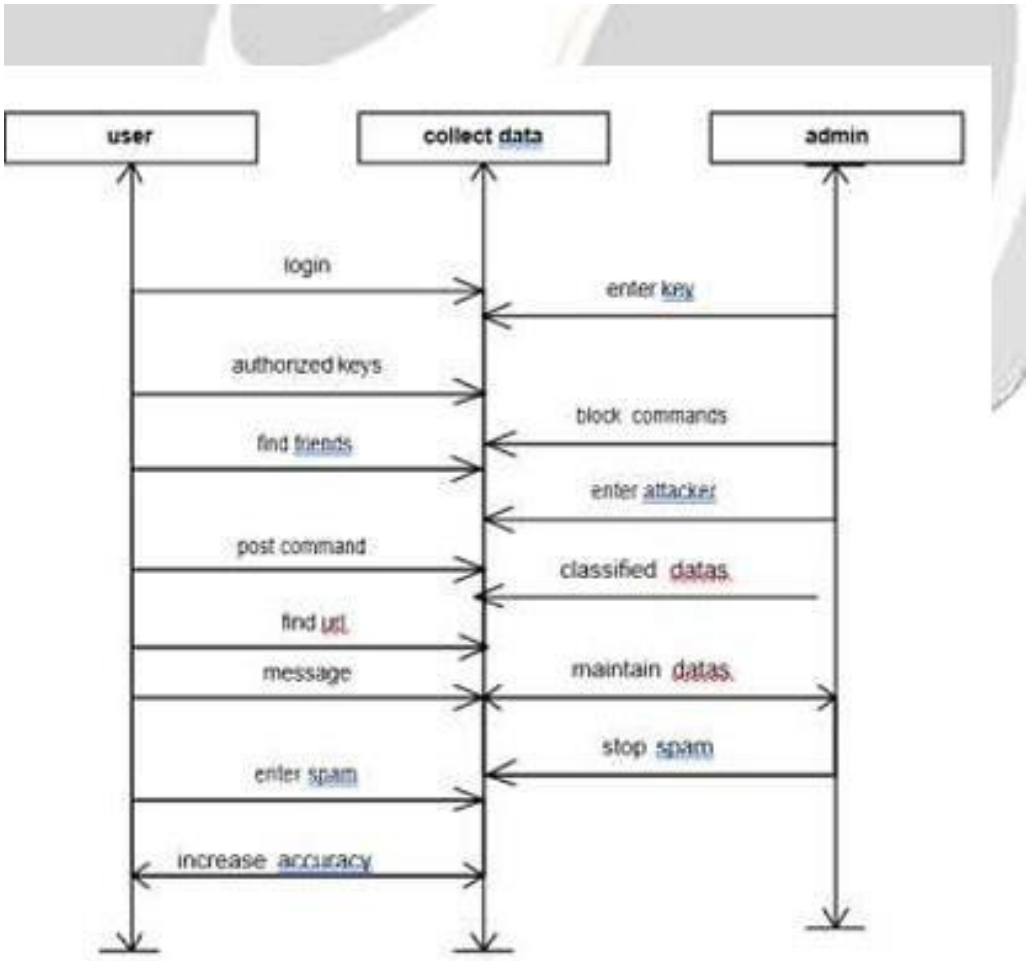


Fig 6.1.2

6.1.3 FLOW DIAGRAM

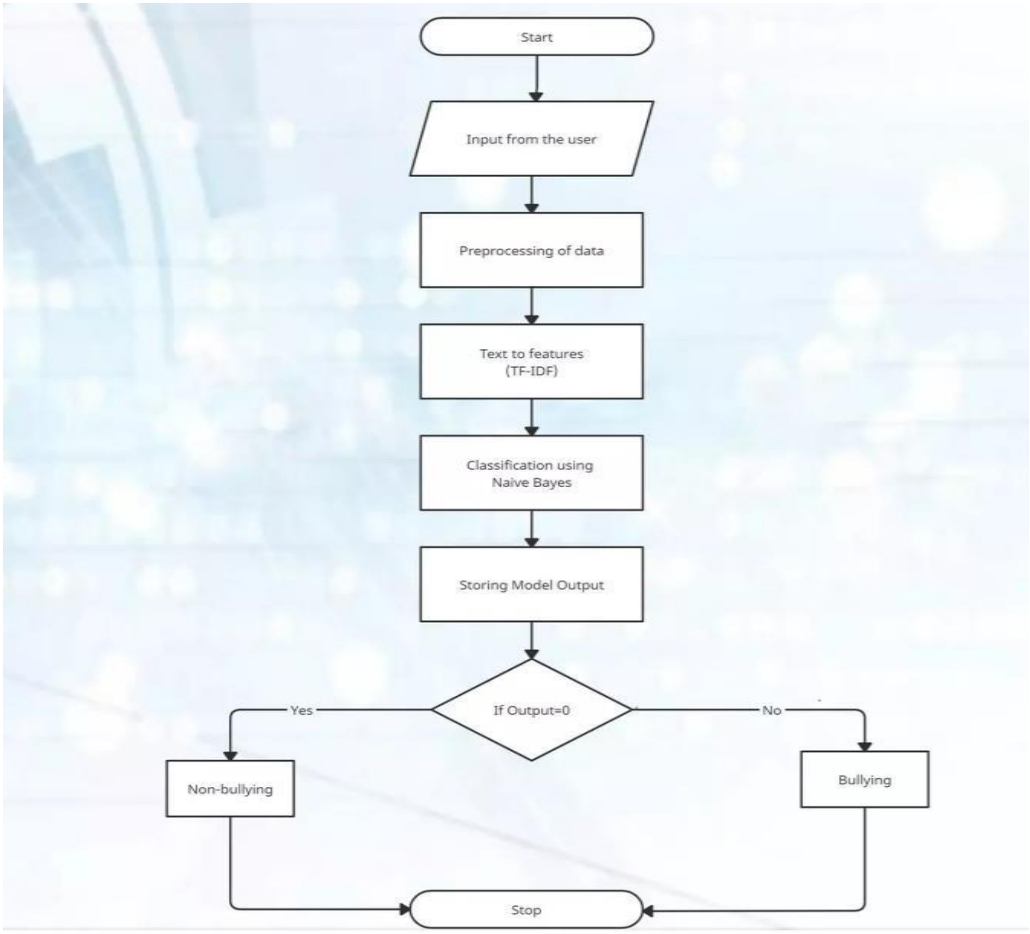
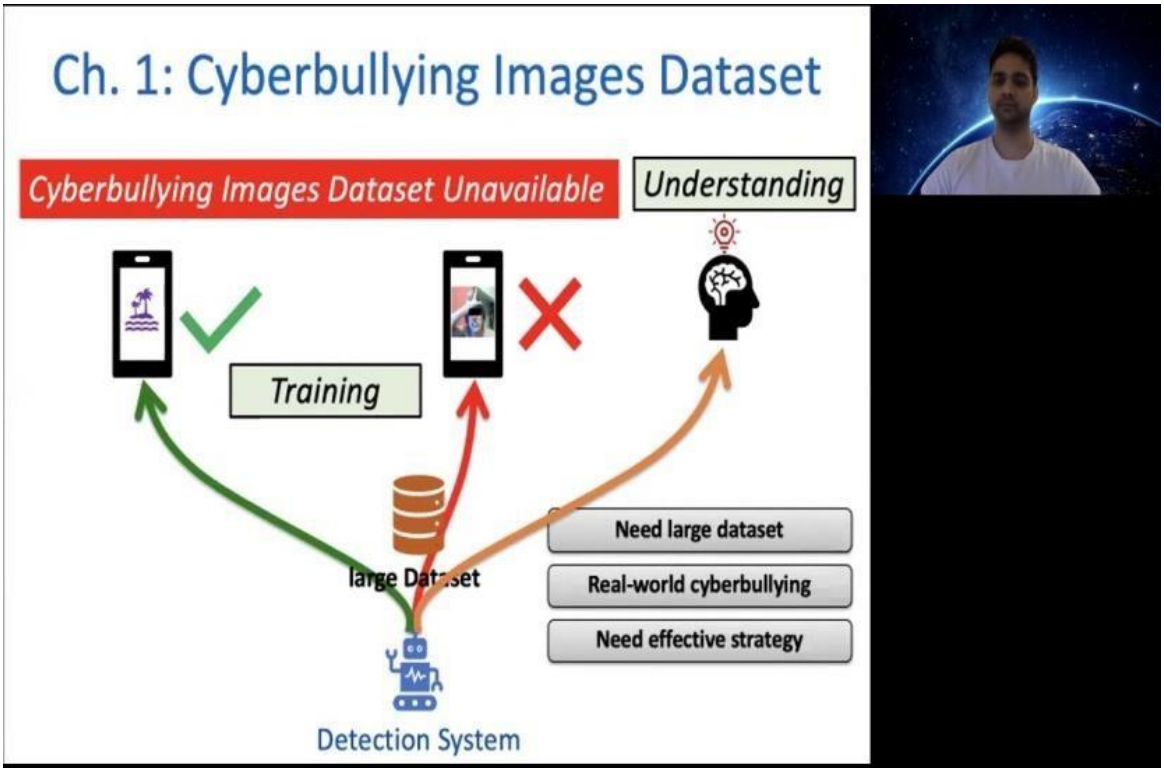


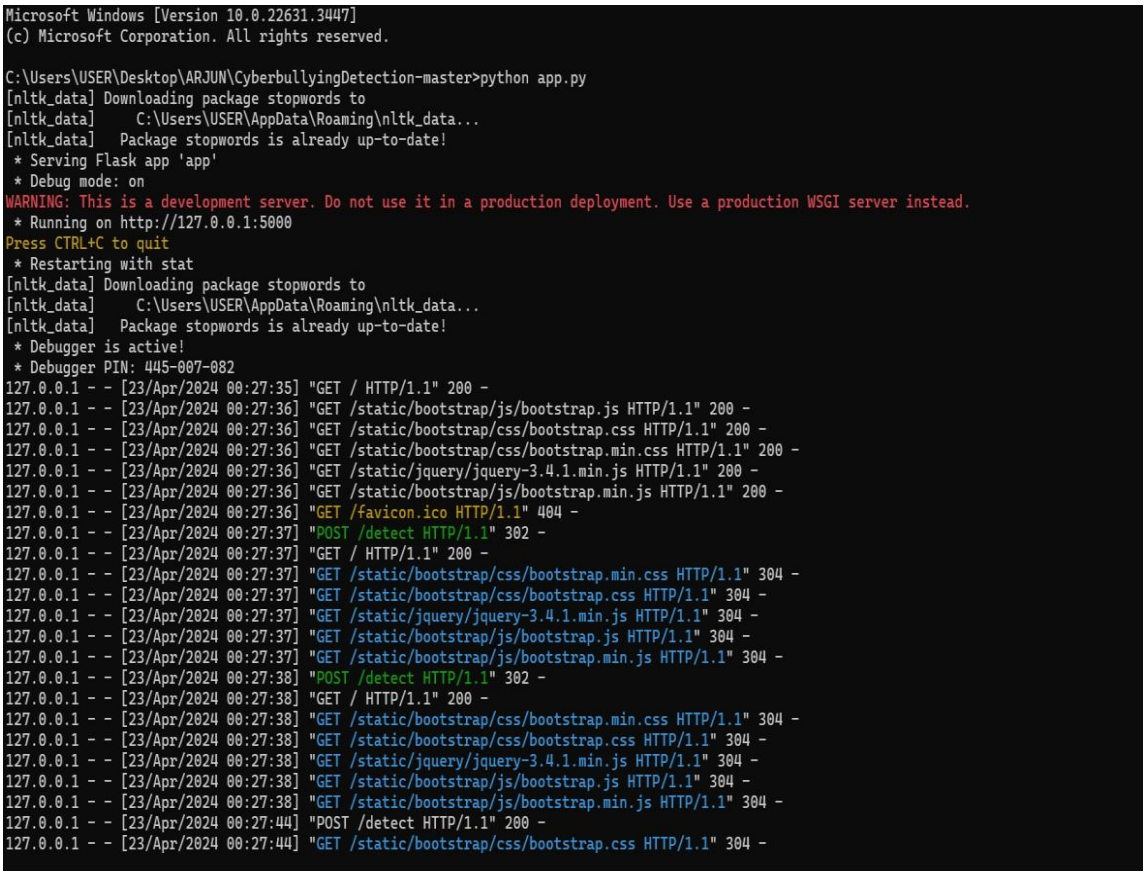
Fig 6.1.3

6.2 PROJECT DEVELOPMENT – 2

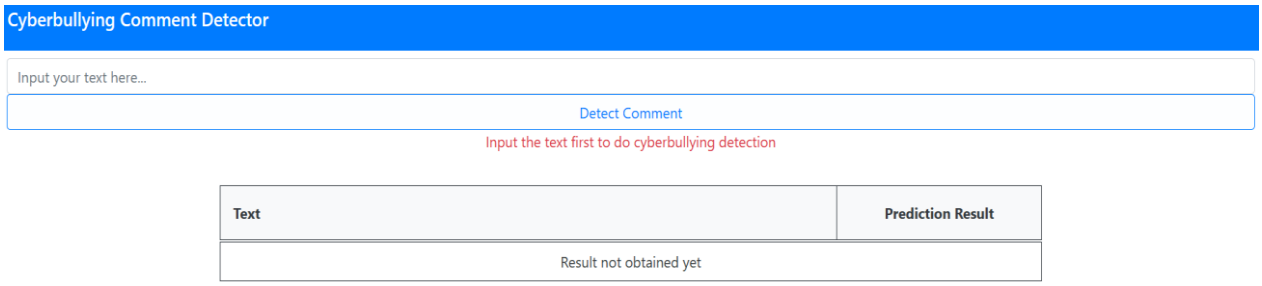
6.2.1 SAMPLE SCREENS OF SYSTEM

Cyber Bulling Detection

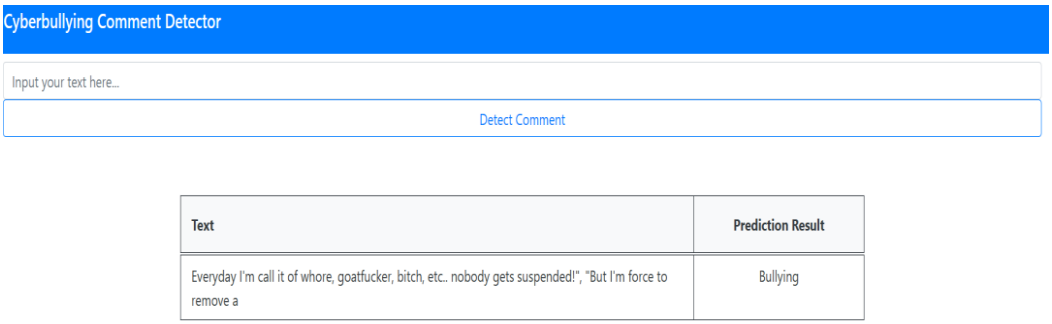




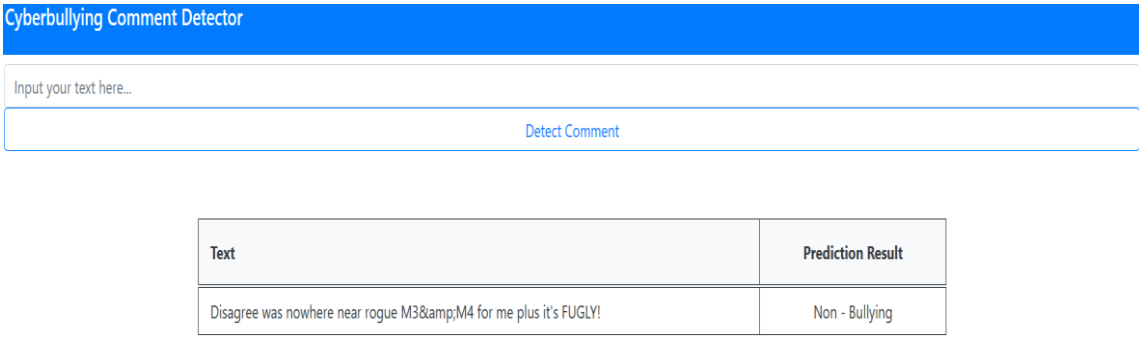
launch server



Cyberbullying Command Detector



Bulling



Non - Bulling

CHAPTER - 7

TESTING

7.1 SYSTEM TESTING

System Testing (ST) is a black box testing technique performed to evaluate the complete system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing. The purpose of integration testing is to detect any inconsistencies between the units that are integrated together (called assemblages). System testing seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. [Citation needed] The actual result is the behavior produced or observed when a component or system is tested.

System testing is performed on the entire system in the context of either functional requirement specifications (FRS) or system requirement specification (SRS), or both. System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software or hardware requirements specification(s).

7.2 BLACK BOX AND WHITE BOX TESTING

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that the software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Some prefer saying Software testing as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application under Test (AUT).

BENEFITS OF SOFTWARE TESTING

Here are the benefits of using software testing:

• **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

• **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

• **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

• **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

7.3 UNIT TESTING

Unit testing is carried out for testing modules constructed from the system design. Each part is compiled using inputs for specific modules. Every module is assembled into a Kroger unit during the unit testing process. Testing has been performed on each phase of project design and coding. The testing of module interface is carried out to ensure the proper flow of information into and court of the program unit while testing. The temporarily generated output data is ensured that maintains its integrity throughout the algorithm's execution by examining the local data structure, finally, all error-handling paths are also tested.

7.4 INTEGRATION TESTING

We usually perform system testing to find errors resulting from unanticipated interaction between the subsystem and system components, Software must be tested to detect and rectify all possible errors once the source code is generated before delivering it to the customers. For finding emirs, seeks of test cases must be developed which binately uncover all the possibly existing errors. Different Software techniques can be used for this process. These techniques provide system guidance for designing tests that exercise the internal logic of the software components and exercise the input and output domains of a program to uncover errors in program function, behavior and Performance we test the software using two methods:

White Box testing: Internal program logic is exercised using this test case design techniques.

Black Box testing: Software requirements are exercised using this test case design techniques. Both techniques help in finding the maximum number of error with minimal effort and time.

7.5 VERIFICATION AND VALIDATION

The testing process is part of a broader subject referring to verification and validation. We have to acknowledge the system specifications and try to meet the customer's requirements and for this sole purpose, we have to verify and validate the product to make sure everything is in place.

Verification and validation are two different things. (One & performed to ensure that the software correctly implements a specific functionality and other & done to ensure if the customer requirements are properly met or not by the end product.

Verification of the project was carried out to ensure that the project met all the requirements and specifications of our project. We made sure that our project is up to the standard as we planned at the beginning of our project development.

CHAPTER - 8

CONCLUSION

In conclusion, the proposed cyberbullying detection system, utilizing the Support Vector Machine (SVM) algorithm, offers a robust and efficient solution to address the growing concern of cyberbullying on social media platforms. By leveraging machine learning techniques, the system can automatically identify instances of cyberbullying in social media content, providing timely alerts and support to users facing potential cyberbullying incidents. The implementation of the system involves several essential modules, including data loading, data pre-processing, feature selection, SVM training, testing, evaluation, and performance analysis. Additionally, the optional monitoring module ensures continuous monitoring of social media activity for proactive cyberbullying detection and intervention. The system's advantages lie in its ability to deliver high accuracy in distinguishing between cyberbullying and non-cyberbullying content, enabling users to take prompt action to create a safer online environment. Furthermore, the system's scalability and continuous improvement mechanisms allow it to adapt to evolving cyberbullying patterns and maintain time.

CHAPTER - 9

FUTURE SCOPE

Future work for the cyberbullying detection system could focus on enhancing its performance and effectiveness by exploring more advanced machine learning techniques. One promising avenue is the integration of deep learning models, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), which can capture complex patterns and semantic relationships in text data. Additionally, incorporating sentiment analysis and contextual information could improve the system's ability to understand the intent behind social media content. Another crucial aspect for future work is increasing the system's adaptability to diverse languages and cultural nuances, enabling it to effectively detect cyberbullying across different regions and communities.

CHAPTER 10

APPENDIX

SOURCE CODE:

```
#Import libraries
import pandas as pd
import numpy as np
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
from nltk.stem import WordNetLemmatizer
```

```
# In[data set import]:
#import dataset
import pandas as pd
df1 = pd.read_csv("cleanprojectdataset.csv")

# In[define data head]:
print(df1)

# In[Create lists for tweets and label]:



Tweet = []
Labels = []

for row in df1["Tweet"]:
    #tokenize words
    words = word_tokenize(row)
    #remove punctuations
    clean_words = [word.lower() for word in words if word not in
set(string.punctuation)]
    #remove stop words
    english_stops = set(stopwords.words('english'))
    characters_to_remove = ["""",'``',"rt","https","’","“","”","\u200b","--
","n't","'s","...","//t.c" ]
    clean_words = [word for word in clean_words if word not in english_stops]
    clean_words = [word for word in clean_words if word not in
set(characters_to_remove)]
```

```
    #Lematise words
    wordnet_lemmatizer = WordNetLemmatizer()
    lemma_list = [wordnet_lemmatizer.lemmatize(word) for word in clean_words]
    Tweet.append(lemma_list)

    for row in df1["Text Label"]:
        Labels.append(row)



# In[5]:

#Combine lists
combined = zip(Tweet, Labels)



# In[6]:

#Create bag of words
def bag_of_words(words):
    return dict([(word, True) for word in words])



# In[7]:

#Create new list for modeling
Final_Data = []
for r, v in combined:
```

```
    bag_of_words(r)
    Final_Data.append((bag_of_words(r),v))


# In[8]:


import random
random.shuffle(Final_Data)
print(len(Final_Data))


# In[9]:

#Split the data into training and test
train_set, test_set = Final_Data[0:746], Final_Data[746:]

#Naive Bayes for Unigramsm check accuracy
import nltk
import collections
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
from nltk import metrics

refsets = collections. defaultdict(set)
testsets = collections.defaultdict(set)

classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
for i, (feats, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)

print("Naive Bayes Performance with Unigrams ")
print("Accuracy:",nltk.classify.accuracy(classifier, test_set))



# In[10]:

#Naive Bayes for Unigrams, Recall Measure
nb_classifier = nltk.NaiveBayesClassifier.train(train_set)

nbrefset = collections.defaultdict(set)
nbtestset = collections.defaultdict(set)

for i, (feats, label) in enumerate(test_set):
    nbrefset[label].add(i)
    observed = nb_classifier.classify(feats)
    nbtestset[observed].add(i)
print("UnigramNB Recall")
print('Bullying recall:', recall(nbtestset['Bullying'], nbrefset['Bullying']))
print("")
```

```
# In[11]:

#Find most informative features
classifier.show_most_informative_features(n=10)



# In[12]:

#Decision Tree for Unigrams
from nltk.classify import DecisionTreeClassifier

dt_classifier = DecisionTreeClassifier.train(train_set,
                                             binary=True,
                                             entropy_cutoff=0.8,
                                             depth_cutoff=5,
                                             support_cutoff=30)
refset = collections.defaultdict(set)
testset = collections.defaultdict(set)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = dt_classifier.classify(feats)
    testset[observed].add(i)
print("UnigramDT Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))
print("")
```

```
# In[13]:

#Logisitic Regression for Unigrams
from nltk.classify import MaxentClassifier

logit_classifier = MaxentClassifier.train(train_set, algorithm='gis', trace=0, max_iter=10, min_lldelta=0.5)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = logit_classifier.classify(feats)
    testset[observed].add(i)
print("UnigramsLogit Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))
print("")



# In[14]:

#Support Vector Machine for Unigrams
from nltk.classify import SklearnClassifier
from sklearn.svm import SVC
SVM_classifier = SklearnClassifier(SVC(), sparse=False).train(train_set)

for i, (feats, label) in enumerate(test_set):
```

```
    refset[label].add(i)
    observed = SVM_classifier.classify(feats)
    testset[observed].add(i)

print("UnigramSVM Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))


# In[15]:

#Same thing with Bigrams
from nltk import bigrams, trigrams
from nltk.collocations import BigramCollocationFinder
from nltk.metrics import BigramAssocMeasures


# In[16]:


combined = zip(Tweet,Labels)


# In[17]:

#Bag of Words of Bigrams
def bag_of_bigrams_words(words, score_fn=BigramAssocMeasures.chi_sq,
n=200):
```

```
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(score_fn, n)
    return bag_of_words(bigrams)


# In[18]:


Final_Data2 =[]

for z, e in combined:
    bag_of_bigrams_words(z)
    Final_Data2.append((bag_of_bigrams_words(z),e))


# In[19]:


import random
random.shuffle(Final_Data2)
print(len(Final_Data2))

train_set, test_set = Final_Data2[0:747], Final_Data2[747:]

import  nltk
import collections
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
```

```
from nltk import metrics

#Naive Bayes for Bigrams

refsets = collections. defaultdict(set)
testsets = collections.defaultdict(set)

classifier = nltk.NaiveBayesClassifier.train(train_set)


for i, (feats, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)


print("Naive Bayes Performance with Bigrams ")
print("Accuracy:",nltk.classify.accuracy(classifier, test_set))


# In[20]:



# In[28]:
```

```
print('bullying precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('bullying recall:', recall(refsets['Bullying'], testsets['Bullying']))


# In[29]:


classifier.show_most_informative_features(n=10)


# In[30]:

#Decision Tree for Trigrams
from nltk.classify import DecisionTreeClassifier

dt_classifier = DecisionTreeClassifier.train(train_set,
                                             binary=True,
                                             entropy_cutoff=0.8,
                                             depth_cutoff=5,
                                             support_cutoff=30)
refset = collections.defaultdict(set)
testset = collections.defaultdict(set)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = dt_classifier.classify(feats)
```

```
    testset[observed].add(i)
print("TrigramDT Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))
print("")


# In[48]:

#Logistic Regression for Trigrams
from nltk.classify import MaxentClassifier

logit_classifier = MaxentClassifier.train(train_set, algorithm='gis', trace=0,
max_iter=10, min_lldelta=0.5)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = logit_classifier.classify(feats)
    testset[observed].add(i)
print("TrigramsLogit Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))
print("")


# In[31]:

#Support Vector Machine for Trigrams
from nltk.classify import SklearnClassifier
```

```
from sklearn.svm import SVC
SVM_classifier = SklearnClassifier(SVC(), sparse=False).train(train_set)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = SVM_classifier.classify(feats)
    testset[observed].add(i)

print("Trigrams Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))


# In[32]:


combined = zip(Tweet,Labels)


# In[33]:

#Combine both unigrams, bigrams, and trigrams

# Import Bigram metrics - we will use these to identify the top 200 bigrams
def bigrams_words(words, score_fn=BigramAssocMeasures.chi_sq,
n=200):
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(score_fn, n)
```

```
    return bigrams

from nltk.collocations import TrigramCollocationFinder

# Import Bigram metrics - we will use these to identify the top 200 trigrams
from nltk.metrics import TrigramAssocMeasures

def trigrams_words(words, score_fn=TrigramAssocMeasures.chi_sq,
n=200):
    trigram_finder = TrigramCollocationFinder.from_words(words)
    trigrams = trigram_finder.nbest(score_fn, n)
    return trigrams

#bag of ngrams
def bag_of_Ngrams_words(words):
    bigramBag = bigrams_words(words)

    #The following two for loops convert tuple into string
    for b in range(0,len(bigramBag)):
        bigramBag[b]=' '.join(bigramBag[b])

    trigramBag = trigrams_words(words)
    for t in range(0,len(trigramBag)):
        trigramBag[t]=' '.join(trigramBag[t])

    return bag_of_words(trigramBag + bigramBag + words)
```

```
# In[34]:


Final_Data4 =[]

for z, e in combined:
    bag_of_Ngrams_words(z)
    Final_Data4.append((bag_of_Ngrams_words(z),e))


# In[35]:


import random
random.shuffle(Final_Data4)
print(len(Final_Data4))

train_set, test_set = Final_Data4[0:747], Final_Data4[747:]

import  nltk
import collections
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
from nltk import metrics
#Naive Bayes for Ngrams

refsets = collections. defaultdict(set)
```

```
testsets = collections.defaultdict(set)

classifier = nltk.NaiveBayesClassifier.train(train_set)


for i, (feats, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)


print("Naive Bayes Performance with Ngrams ")
print("Accuracy:",nltk.classify.accuracy(classifier, test_set))


# In[59]:


classifier.show_most_informative_features(n=10)


# In[36]:


print('bullying precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('bullying recall:', recall(refsets['Bullying'], testsets['Bullying']))
```

```
# In[37]:

#Decision Tree for Ngrams
from nltk.classify import DecisionTreeClassifier

dt_classifier = DecisionTreeClassifier.train(train_set,
                                             binary=True,
                                             entropy_cutoff=0.8,
                                             depth_cutoff=5,
                                             support_cutoff=30)
refset = collections.defaultdict(set)
testset = collections.defaultdict(set)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = dt_classifier.classify(feats)
    testset[observed].add(i)
print("NgramDT Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))
print("")


# In[38]:

#Logistic Regression for Ngrams
from nltk.classify import MaxentClassifier
```

```
logit_classifier = MaxentClassifier.train(train_set, algorithm='gis', trace=0, max_iter=10, min_lldelta=0.5)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = logit_classifier.classify(feats)
    testset[observed].add(i)
print("NgramsLogit Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))
print("")


# In[39]:

#Support Vector Machine for Ngrams
from nltk.classify import SklearnClassifier
from sklearn.svm import SVC
SVM_classifier = SklearnClassifier(SVC(), sparse=False).train(train_set)

for i, (feats, label) in enumerate(test_set):
    refset[label].add(i)
    observed = SVM_classifier.classify(feats)
    testset[observed].add(i)

print("Trigrams Recall")
print('Bullying recall:', recall(testset['Bullying'], refset['Bullying']))
```

```
# In[41]:


train_set, test_set = Final_Data[0:747], Final_Data[747:]

import nltk
import collections
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
nb_classifier = nltk.NaiveBayesClassifier.train(train_set)
nb_classifier.show_most_informative_features(10)

from nltk.classify.util import accuracy
print(accuracy(nb_classifier, test_set))

refsets = collections.defaultdict(set)
testsets = collections.defaultdict(set)

for i, (Final_Data, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = nb_classifier.classify(Final_Data)
    testsets[observed].add(i)

print('bullying precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('bullying recall:', recall(refsets['Bullying'], testsets['Bullying']))
print('bullying F-measure:', f_measure(refsets['Bullying'], testsets['Bullying']))
```

```
print('not-bullying precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('not-bullying recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('not-bullying F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))
```

In[32]:

```
import collections
from nltk import metrics
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
from nltk.classify import DecisionTreeClassifier
from nltk.classify.util import accuracy
dt_classifier = DecisionTreeClassifier.train(train_set,
                                             binary=True,
                                             entropy_cutoff=0.8,
                                             depth_cutoff=5,
                                             support_cutoff=30)
from nltk.classify.util import accuracy
print(accuracy(dt_classifier, test_set))

refsets = collections.defaultdict(set)
testsets = collections.defaultdict(set)

for i, (Final_Data, label) in enumerate(test_set):
```

```
    refsets[label].add(i)
    observed = dt_classifier.classify(Final_Data)
    testsets[observed].add(i)

print('bullying precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('bullying recall:', recall(refsets['Bullying'], testsets['Bullying']))
print('bullying F-measure:', f_measure(refsets['Bullying'], testsets['Bullying']))
print('non-bullying precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('non-bullying recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('non-bullying F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))


# In[33]:


#Create Logistic Regression model to compare
from nltk.classify import MaxentClassifier
import collections
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)

logit_classifier = MaxentClassifier.train(train_set, algorithm='gis', trace=0, max_iter=10, min_lldelta=0.5)

for i, (Final_Data, label) in enumerate(test_set):
```

```
    refsets[label].add(i)
    observed = logit_classifier.classify(Final_Data)
    testsets[observed].add(i)

print('pos precision:', precision(refsets['Bullying'], testsets['Non-Bullying']))
print('pos recall:', recall(refsets['Bullying'], testsets['Non-Bullying']))
print('pos F-measure:', f_measure(refsets['Bullying'], testsets['Non-Bullying']))
print('neg precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))


# In[34]:


# SVM model

from nltk.classify import SklearnClassifier
from sklearn.svm import SVC

SVM_classifier = SklearnClassifier(SVC(), sparse=False).train(train_set)

for i, (Final_Data, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = SVM_classifier.classify(Final_Data)
    testsets[observed].add(i)
```

```
print('pos precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('pos recall:', recall(refsets['Bullying'], testsets['Bullying']))
print('pos F-measure:', f_measure(refsets['Bullying'], testsets['Bullying']))
print('neg precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))
```

```
# In[42]:
```

```
zl = zip(Tweet,Labels)

#define a bag_of_words function to return word, True.

def bag_of_words(words):
    return dict([(word, True) for word in words])

def bag_of_words_not_in_set(words, badwords):
    return bag_of_words(set(words) - set(badwords))

# Define another function that will return words that are in words, but not in
badwords

from nltk.corpus import stopwords
```

```
#define a bag_of_non_stopwords function to return word, True.

def bag_of_non_stopwords(words, stopfile='english'):
    badwords = stopwords.words(stopfile)
    return bag_of_words_not_in_set(words, badwords)

from nltk.collocations import BigramCollocationFinder

# Import Bigram metrics - we will use these to identify the top 200 bigrams
from nltk.metrics import BigramAssocMeasures

def bag_of_bigrams_words(words, score_fn=BigramAssocMeasures.chi_sq,
n=200):
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(score_fn, n)
    return bag_of_words(bigrams)

    bigrams = bag_of_bigrams_words(words)

#Creating our unigram featureset dictionary for modeling

Final_Data = []

for k, v in zl:
    bag_of_bigrams_words(k)
    Final_Data.append((bag_of_bigrams_words(k),v))
```

```
import random
random.shuffle(Final_Data)

#splits the data around 70% of 500 *350 reviews* for both testing and training

train_set, test_set = Final_Data[0:778], Final_Data[778:]

#Now we will calculate accuracy, precision, recall, and f-measure using Naives Bayes classifier

import  nltk
import collections
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
nb_classifier = nltk.NaiveBayesClassifier.train(train_set)
nb_classifier.show_most_informative_features(10)

from nltk.classify.util import accuracy
print(accuracy(nb_classifier, test_set))

refsets = collections.defaultdict(set)
testsets = collections.defaultdict(set)

for i, (Final_Data, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = nb_classifier.classify(Final_Data)
    testsets[observed].add(i)
```

```
print('bullying precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('bullying recall:', recall(refsets['Bullying'], testsets['Bullying']))
print('bullying F-measure:', f_measure(refsets['Bullying'], testsets['Bullying']))
print('not-bullying precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('not-bullying recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('not-bullying F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))
```

In[44]:

```
def bigrams_words(words, score_fn=BigramAssocMeasures.chi_sq,
n=200):
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(score_fn, n)
    return bigrams

from nltk.collocations import TrigramCollocationFinder

# Import Bigram metrics - we will use these to identify the top 200 bigrams
from nltk.metrics import TrigramAssocMeasures

def trigrams_words(words, score_fn=TrigramAssocMeasures.chi_sq,
n=200):
```

```
    trigram_finder = TrigramCollocationFinder.from_words(words)
    trigrams = trigram_finder.nbest(score_fn, n)
    return trigrams


def bag_of_Ngrams_words(words):
    bigramBag = bigrams_words(words)

    #The following two for loops convert tuple into string
    for b in range(0,len(bigramBag)):
        bigramBag[b]=' '.join(bigramBag[b])

    trigramBag = trigrams_words(words)
    for t in range(0,len(trigramBag)):
        trigramBag[t]=' '.join(trigramBag[t])

    return bag_of_words(trigramBag + bigramBag + words)


# In[47]:


zl = zip(Tweet,Labels)

Final_Data = []

for k, v in zl:
```

```
    bag_of_words(k)
    Final_Data.append((bag_of_words(k),v))

import random
random.shuffle(Final_Data)

#splits the data around 70% of 500 *350 reviews* for both testing and training

train_set, test_set = Final_Data[0:778], Final_Data[778:]

#Now we will calculate accuracy, precision, recall, and f-measure using Naives Bayes classifier

import  nltk
import collections
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
nb_classifier = nltk.NaiveBayesClassifier.train(train_set)
nb_classifier.show_most_informative_features(10)

from nltk.classify.util import accuracy
print(accuracy(nb_classifier, test_set))

refsets = collections.defaultdict(set)
testsets = collections.defaultdict(set)

for i, (Final_Data, label) in enumerate(test_set):
    refsets[label].add(i)
```

```
    observed = nb_classifier.classify(Final_Data)
    testsets[observed].add(i)

print('bullying precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('bullying recall:', recall(refsets['Bullying'], testsets['Bullying']))
print('bullying F-measure:', f_measure(refsets['Bullying'], testsets['Bullying']))
print('not-bullying precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('not-bullying recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('not-bullying F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))


# In[48]:


import collections
from nltk import metrics
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)
from nltk.classify import DecisionTreeClassifier
from nltk.classify.util import accuracy
dt_classifier = DecisionTreeClassifier.train(train_set,
                                             binary=True,
                                             entropy_cutoff=0.8,
                                             depth_cutoff=5,
                                             support_cutoff=30)
from nltk.classify.util import accuracy
```

```
print(accuracy(dt_classifier, test_set))

refsets = collections.defaultdict(set)
testsets = collections.defaultdict(set)

for i, (Final_Data, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = dt_classifier.classify(Final_Data)
    testsets[observed].add(i)

print('bullying precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('bullying recall:', recall(refsets['Bullying'], testsets['Bullying']))
print('bullying F-measure:', f_measure(refsets['Bullying'], testsets['Bullying']))
print('non-bullying precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('non-bullying recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('non-bullying F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))


# In[49]:


#Create Logistic Regression model to compare
from nltk.classify import MaxentClassifier
import collections
```

```
from nltk.metrics.scores import (accuracy, precision, recall, f_measure)

logit_classifier = MaxentClassifier.train(train_set, algorithm='gis', trace=0, max_iter=10, min_lldelta=0.5)

for i, (Final_Data, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = logit_classifier.classify(Final_Data)
    testsets[observed].add(i)

print('pos precision:', precision(refsets['Bullying'], testsets['Non-Bullying']))
print('pos recall:', recall(refsets['Bullying'], testsets['Non-Bullying']))
print('pos F-measure:', f_measure(refsets['Bullying'], testsets['Non-Bullying']))
print('neg precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))


# In[50]:


# SVM model

from nltk.classify import SklearnClassifier
from sklearn.svm import SVC
```

```
SVM_classifier = SklearnClassifier(SVC(), sparse=False).train(train_set)

for i, (Final_Data, label) in enumerate(test_set):
    refsets[label].add(i)
    observed = SVM_classifier.classify(Final_Data)
    testsets[observed].add(i)

print('pos precision:', precision(refsets['Bullying'], testsets['Bullying']))
print('pos recall:', recall(refsets['Bullying'], testsets['Bullying']))
print('pos F-measure:', f_measure(refsets['Bullying'], testsets['Bullying']))
print('neg precision:', precision(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg recall:', recall(refsets['Non-Bullying'], testsets['Non-Bullying']))
print('neg F-measure:', f_measure(refsets['Non-Bullying'], testsets['Non-Bullying']))
```

REFERENCES

[1] B. Cagirkan and G. Bilek, ‘‘Cyberbullying among Turkish high school students,’’ Scandin. J. Psychol., vol. 62, no. 4, pp. 608–616, Aug. 2021, doi: 10.1111/sjop.12720

[2] P. T. L. Chi, V. T. H. Lan, N. H. Ngan, and N. T. Linh, ‘‘Online time, experience of cyber bullying and practices to cope with it among high school students in Hanoi,’’ Health Psychol. Open, vol. 7, no. 1, Jan. 2020, Art. no. 205510292093574, doi: 10.1177/2055102920935747

[3] A. López-Martínez, J. A. García-Díaz, R. Valencia-García, and A. Ruiz-Martínez, ‘‘CyberDect. A novel approach for cyberbullying detection on Twitter,’’ in Proc. Int. Conf. Technol. Innov., Guayaquil, Ecuador: Springer, 2019, pp. 109–121, doi: 10.1007/978-3-030-34989- 9_9.

[4] R. M. Kowalski and S. P. Limber, ‘‘Psychological, physical, and academic correlates of cyberbullying and traditional bullying,’’ J. Adolescent Health, vol. 53, no. 1, pp. S13–S20, Jul. 2020, doi: 10.1016/j.jadohealth.2012.09.018

[5] Y.-C. Huang, ‘‘Comparison and contrast of piaget and Vygotsky’s theories,’’ in Proc. Adv. Social Sci., Educ. Humanities Res., 2021, pp. 28–32, doi: 10.2991/assehr.k.210519.007

[6] A. Anwar, D. M. H. Kee, and A. Ahmed, ‘‘Workplace cyberbullying and interpersonal deviance: Understanding the mediating effect of silence and emotional exhaustion,’’ Cyberpsychol., Behav., Social Netw., vol. 23, no. 5, pp. 290–296, May 2020, doi: 10.1089/cyber.2019.0407.

[7] D. M. H. Kee, M. A. L. Al-Anesi, and S. A. L. Al-Anesi, ‘‘Cyberbullying on social media under the influence of COVID-19,’’ Global Bus. Organizational Excellence, vol. 41, no. 6, pp. 11–22, Sep. 2022, doi: 10.1002/joe.22175

[8] I. Kwan, K. Dickson, M. Richardson, W. MacDowall, H. Burchett, C. Stansfield, G. Brunton, K. Sutcliffe, and J. Thomas, ‘‘Cyberbullying and children and young people’s

mental health: A systematic map of systematic reviews,'' Cyberpsychol., Behav., Social Netw., vol. 23, no. 2, pp. 72–82, Feb. 2020, doi: 10.1089/cyber.2019.0370.

JOURNAL PUBLICATION

Vol-2 Issue-1 2016 IJARIIE-ISSN(O)-2395-4396

TITLE: CYBER BULLING DETECTION USING MACHINE LEARNING

Sathya S, Mohammed Rishan P, Nived PK, Narayanan K

[1] Assistant Professor, Department of Computer Science & Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India
[2] Student, Department of Computer Science & Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India
[3] Student, Department of Computer Science & Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India
[4] Student, Department of Computer Science & Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India
[5] Student, Department of Computer Science & Engineering, Hindusthan College of Engineering and Technology, Tamil Nadu, India

ABSTRACT

Cyberbullying has emerged as a pervasive and concerning issue on social media platforms, impacting the mental health and well-being of individuals worldwide. To address this problem, this study proposes a cyberbullying detection system using the Support Vector Machine (SVM) algorithm. Leveraging the power of machine learning, the system aims toautomatically identify and flag instances of cyberbullying in social media content. The development of the detection system begins with the collection and labelling of a comprehensive dataset containing examples of cyberbullying and non-cyberbullying postsor comments. After pre-processing the text data by removing irrelevant information, converting text to lowercase, and tokenizing it, meaningful features are extracted using thebag-of-words or TF-IDF techniques. These transformed feature vectors serve as inputs fortraining the SVM classifier, which seeks to find the optimal hyper plane for effectively distinguishing cyberbullying from non-cyberbullying content. The performance of the SVM model is evaluated using a separate testing dataset, with metrics such as accuracy, precision, recall, F1-score, and ROC-AUC analysed to assess its effectiveness in identifying cyberbullying instances. Model fine-tuning is conducted throughexperimentation with various SVM hyper parameters and cross-validation techniques to optimize performance

INTERNATIONAL JOURNAL OF ADVANCE
RESEARCH AND INNOVATIVE IDEAS IN EDUCATION

★★★

CERTIFICATE

of

PUBLICATION

The Board of International Journal of Advance Research and Innovative Ideas in Education
is hereby Awarding this Certificate to

NARAYANAN K

In Recognition of the Publication of the Paper Entitled

CYBER BULLING DETECTION USING MACHINE LEARNING

Published in E-Journal

Volume-10 Issue-3 2024

Peer Review Journal
Paper Id : 23881
ISSN(O) : 2395-4396

www.ijariie.com

Editor In Chief

INTERNATIONAL JOURNAL OF ADVANCE
RESEARCH AND INNOVATIVE IDEAS IN EDUCATION

★★★

CERTIFICATE

of

PUBLICATION

The Board of International Journal of Advance Research and Innovative Ideas in Education
is hereby Awarding this Certificate to

MOHAMMED RISHAN P

In Recognition of the Publication of the Paper Entitled

CYBER BULLING DETECTION USING MACHINE LEARNING

Published in E-Journal

Volume-10 Issue-3 2024

Peer Review Journal
Paper Id : 23881
ISSN(O) : 2395-4396

www.ijariie.com

Editor In Chief

INTERNATIONAL JOURNAL OF ADVANCE RESEARCH AND INNOVATIVE IDEAS IN EDUCATION

★ ★ ★

CERTIFICATE

of

PUBLICATION

The Board of International Journal of Advance Research and Innovative Ideas in Education
is hereby Awarding this Certificate to

NIVED PK

In Recognition of the Publication of the Paper Entitled

CYBER BULLING DETECTION USING MACHINE LEARNING

Published in E-Journal

Volume-10 Issue-3 2024

Peer Review Journal
Paper Id : 23881
ISSN(O) : 2395-4396

www.ijariie.com

Editor In Chief