1. INTRODUCTION

1.1. Introduction to Project

In today’s digital era, social media plays a vital role in communication and expression but also gives rise to cyberbullying—harassment through harmful online content. This issue, especially among youth, has serious psychological effects like anxiety, depression, and even suicide. Traditional moderation methods, such as manual review or keyword filtering, are not scalable or context-aware. To address this, the project introduces a Machine Learning-based system to detect cyberbullying in social media comments. It uses Natural Language Processing (NLP) to clean and analyze text data. Classification algorithms like Logistic Regression and XGBoost are applied for prediction. TF-IDF vectorization and n-gram analysis help improve accuracy. Hyper parameter tuning is performed for model optimization. Class imbalance is handled using resampling techniques. The system also focuses on detecting subtle abuse like sarcasm and slang. Pattern recognition and semantic analysis help in identifying implicit bullying. This enables the model to generalize across varied abuse types. Ultimately, the project offers a scalable AI solution for safer online platforms.

1.2. Purpose of the project

The primary purpose of this project is to address the increasing problem of cyberbullying on social media by developing an automated detection system using machine learning. With the rapid growth of online platforms, harmful and abusive content has become more common, especially among young users, leading to emotional distress and psychological consequences. Manual moderation techniques are no longer sufficient due to the massive volume and complexity of user-generated content. Therefore, this project aims to leverage Natural Language Processing (NLP) and classification algorithms to identify cyberbullying content in textual data effectively. By using TF-IDF vectorization, n-gram analysis, and hyperparameter tuning, the model achieves better accuracy in detecting both direct and indirect bullying. Additionally, the project addresses class imbalance to ensure fair prediction outcomes. The resulting system can assist platforms and administrators in early identification of offensive content, contributing to safer and more respectful digital environments.

1.3. Objective

The primary objective of this project is to build an intelligent system that can effectively detect cyberbullying in social media comments using machine learning techniques. This involves developing a text classification model that can accurately distinguish between cyberbullying and non-cyberbullying content. To achieve this, the project focuses on applying Natural Language Processing (NLP) techniques to clean and preprocess textual data by removing noise such as URLs, mentions, special characters, and stopwords. Further, TF-IDF vectorization and n-gram analysis are used to convert text into meaningful features for model training.

To ensure that the model performs well across different types of data, the project addresses the issue of class imbalance by employing resampling techniques, thereby improving the fairness and reliability of predictions. The system is implemented using machine learning algorithms such as Logistic Regression and XGBoost, which are known for their performance in binary classification tasks. Hyperparameter tuning is performed to optimize model accuracy and reduce misclassification rates. Finally, the model is evaluated using standard performance metrics including accuracy, precision, recall, F1-score, and confusion matrix. The goal is to create a scalable, accurate, and adaptable solution that contributes to reducing online harassment and improving the safety of digital communication platforms.

1.4. Existing System

In the current digital landscape, cyberbullying detection largely relies on manual content moderation or basic rule-based keyword filtering systems. Social media platforms like Facebook, Twitter, and Instagram provide users with the ability to report abusive content, which is then reviewed manually by moderators. This process is time-consuming, inconsistent, and often unable to scale with the massive volume of user-generated content.

Some existing systems employ keyword matching or predefined word lists to automatically detect harmful language. However, these systems are ineffective against context-based bullying, sarcasm, slang, and evolving language patterns.

They also generate high false-positive rates by flagging non-offensive content that contains certain keywords.

Traditional natural language processing (NLP) tools used in earlier systems often failed to capture the nuanced sentiment or intent behind a post. Moreover, existing models are typically platform-specific and do not generalize well across different types of social media networks. The lack of labeled datasets and adaptive learning mechanisms limits the effectiveness of conventional systems.

Therefore, there is a pressing need for intelligent, machine learning-based approaches that can understand context, learn from data, and automatically adapt to new forms of abusive language, enabling more efficient and accurate detection of cyberbullying behaviour.

1.5. Proposed System

The proposed project focuses on developing a machine learning-based cyberbullying detection system that leverages natural language processing (NLP) and supervised learning techniques to classify social media text as bullying or non-bullying. Unlike existing solutions that rely solely on manual moderation or keyword filters, this system applies intelligent classification algorithms—such as Logistic Regression, Support Vector Machines (SVM), and Random Forest—to identify harmful intent in online conversations.

The system begins by preprocessing textual content using NLP techniques like stopword removal, stemming, and TF-IDF feature extraction to transform raw posts into structured data. These features are then used to train the machine learning models on a labeled dataset containing cyberbullying and non-cyberbullying instances. Once trained, the system can evaluate new posts in real-time and flag abusive content for moderation.

Key components of the system include a preprocessing module for data cleaning, a classification module for model prediction, and an evaluation module that generates reports based on metrics such as accuracy, precision, recall, and F1-score. Additionally, the system supports scalability by allowing retraining and hyperparameter tuning through grid search, ensuring adaptability to new forms of online abuse.

2. LITERATURE SURVEY

Base Paper	Author	Journal/Publication	Year	Algorithm Used	Conclusion
Detecting Cyberbullying in Social Media Text	Dinakar et al.	ACM Conference on Web Science	2011	Naive Bayes	Demonstrated effectiveness of machine learning in detecting online bullying.
Automated Cyberbullying Detection using Machine Learning	Reynolds et al.	International Conference on Machine Learning	2011	SVM	Showed that SVM can be trained to identify bullying behavior in comments
Cyberbullying Detection: A Comparative Study	Salawu et al.	Journal of Information & Knowledge Management	2017	Decision Tree, Naive Bayes	Compared multiple models; highlighted limitations of traditional classifiers.
A Machine Learning Approach to Detect Cyberbullying	Zhao et al.	European Conference on Information Retrieval	2016	LSTM (Deep Learning)	Used LSTM networks to model sequence context for better detection accuracy.
A Deep Learning Approach for Detecting Cyberbullying	Agrawal & Awekar	IEEE Xplore	2018	TF-IDF + Logistic Regression	Showed that TF-IDF combined with LR enhances detection of toxic text.

Table-1 Literature Survey

3. SOFTWARE REQUIREMENT ANALYSIS

3.1. Problem Specification

The rise of social media has brought numerous benefits in terms of connectivity and expression, but it has also given rise to a growing concern: cyberbullying. Traditional methods for detecting such harmful behavior typically rely on manual monitoring and user reporting, which are often delayed, inconsistent, and inefficient. This manual moderation process not only struggles to keep up with the massive volume of content generated daily but also leaves many cases of cyberbullying unnoticed or unresolved.

Another issue with current systems is their dependence on keyword-based filters, which often result in high false positives and fail to capture the context or intent behind a message. Cyberbullies frequently use slang, sarcasm, or coded language that simple rule-based systems cannot effectively interpret. This lack of contextual understanding limits the accuracy and reliability of detection efforts.

Moreover, the existing moderation mechanisms lack scalability and adaptability. As the variety of platforms and the volume of user-generated content continue to grow, traditional moderation techniques become inadequate for large-scale deployment. There is also a gap in providing real-time responses to harmful content, which is essential for immediate intervention and prevention of psychological harm to victims.

i. Proposed Solution

To tackle the challenges of manual, inefficient, and inaccurate cyberbullying detection, this project proposes a Machine Learning-based automated system that can classify social media comments as cyberbullying or not. By leveraging natural language processing (NLP) and supervised learning algorithms, the system aims to analyze large volumes of textual data in real time, significantly reducing human effort and increasing detection efficiency.

The model utilizes TF-IDF vectorization to extract meaningful features from the text and applies Logistic Regression as a baseline classifier. To improve performance, n-gram features and hyperparameter tuning are incorporated, enabling the system to better understand context and intent, which are often missed by traditional keyword-based filters.

Furthermore, the solution is scalable and adaptable to various social media platforms, allowing it to handle high volumes of content without compromising accuracy. The automation of the detection process enables quicker interventions, which is crucial for minimizing psychological harm to victims.

The proposed system emphasizes both accuracy and ethical implementation. It ensures that user data is anonymized and processed securely, maintaining privacy standards while delivering effective detection. Overall, this intelligent, automated solution provides a robust and efficient method to combat cyberbullying in the dynamic environment of social media.

3.2. Modules and Their Functionalities

There are six (6) Modules

1. Data Collection Module
2. Feature Extraction Module
3. Feature Extraction Module
4. Model Training Module
5. Prediction and Evaluation Module
6. Visualization Module

1. Data Collection Module

The Data Collection Module serves as the foundation for acquiring relevant social media data, primarily focusing on text content from platforms such as Twitter. The dataset utilized in this project includes pre-labeled tweets categorized as either cyberbullying (1) or non-cyberbullying (0).

This module is responsible for gathering, storing, and organizing raw user-generated content for subsequent processing. It ensures the data diversity and richness required to train a robust machine learning model capable of distinguishing between harmful and benign comments.

2. Data Preprocessing Module

This module handles all necessary text cleaning operations to prepare the dataset for effective feature extraction. It starts by eliminating noise in the data such as null values, URLs, mentions, hashtags, numbers, and punctuation. Then, the text is converted to lowercase, and stop words are removed to normalize the input. Additionally, this module addresses the class imbalance present in the dataset by downsampling the majority class (non-cyberbullying) to match the size of the minority class. These operations are critical in enhancing the performance of machine learning models and ensuring unbiased learning.

3. Feature Extraction Module

The Feature Extraction Module transforms cleaned text data into numerical vectors using the TF-IDF (Term Frequency–Inverse Document Frequency) technique. This technique evaluates the importance of a word in a document relative to the entire dataset, thereby preserving semantic value. The module supports different n-gram configurations (unigrams and bigrams) to capture both single words and word pairings, which enriches contextual understanding. These vectors serve as the primary input for model training.

4. Model Training Module

This core module is responsible for building and training machine learning models to detect cyberbullying content. Logistic Regression is used as the baseline model, and additional algorithms such as XGBoost and Random Forest are also explored to assess comparative performance. This module also splits the dataset into training and testing sets, fits models on training data, and uses the testing data for validation. Hyperparameter tuning using GridSearchCV is integrated to enhance model performance by selecting the most suitable parameters for each algorithm.

5. Prediction and Evaluation Module

Once the model is trained, this module handles prediction and performance evaluation. It assesses how well the model classifies new, unseen data. Evaluation metrics such as accuracy, precision, recall, F1-score, and support are calculated to gauge the model’s effectiveness. The module also includes visualization tools like confusion matrix heatmaps that provide insights into false positives and false negatives. These evaluations help in understanding and refining the model’s capabilities.

6. Visualization Module

The Visualization Module is designed to visually interpret model performance and data insights. It generates plots and graphs such as confusion matrices and score summaries to assist in communicating results in a clear and impactful manner. This module is essential for reporting, presentations, and debugging, as it visually demonstrates the strengths and weaknesses of the machine learning models in detecting cyberbullying from social media text.

3.3. Functional Requirements

Functional requirements describe the essential features and behaviors that the system must exhibit to meet user needs and ensure proper functioning of the cyberbullying detection system. Below are the key functional requirements:

- ➢ Data Acquisition and Labelling
- ➢ Text Preprocessing and Cleaning
- ➢ Feature Extraction using TF-IDF
- ➢ Model Training and Validation
- ➢ Prediction and Cyberbullying Classification

1. Data Acquisition and Labeling

The system must be capable of acquiring large volumes of textual data from social media platforms such as Twitter. This data should include user-generated posts that may potentially contain cyberbullying content. The data must be pre-labeled for supervised learning, where each instance is tagged as either cyberbullying or non-cyberbullying.

2. Text Preprocessing and Cleaning

Before the data can be used for model training, the system must preprocess the raw text to remove irrelevant elements. This includes stripping away URLs, user mentions, hashtags,punctuation, numbers, and converting all text to lowercase. Additionally, common stop words should be removed to eliminate noise. This cleaning step is critical for normalizing the input data and ensuring consistency, which improves the performance of the machine learning algorithms.

3. Feature Extraction using TF-IDF

The system must convert the cleaned textual data into numerical features using TF-IDF vectorization. This method helps in identifying the significance of words or phrases within each tweet relative to the entire dataset. The system should support unigrams and bigrams (1-gram and 2-gram) to capture both individual words and word combinations. These vectorized representations form the core input for the classification model.

4. Model Training and Validation

The system must include functionality to train classification models on the feature vectors using various machine learning algorithms, beginning with Logistic Regression. It should split the dataset into training and testing sets to evaluate generalization. Additionally, the system should support hyperparameter tuning through methods such as GridSearchCV to optimize the model's accuracy and F1-score. This requirement ensures that the model is not only functional but also optimized for real-world application.

5. Prediction and Cyberbullying Classification

Once the model is trained, the system must be able to predict whether a given social media post constitutes cyberbullying or not.

This involves feeding new, unseen text through the same preprocessing and vectorization pipeline and obtaining predictions from the trained model. The classifier's output should clearly label the post as either "Cyberbullying" or "Not Cyberbullying" to assist users or moderation systems in taking appropriate action.

3.4. Non-Functional Requirements

Non-functional requirements define the system's operational qualities, ensuring that the cyberbullying detection system performs efficiently, reliably, and securely under various conditions. These requirements focus on how the system behaves rather than what it does.

- Security
- Performance
- Scalability
- Usability
- Compliance
- Availability

1. Security

The system must ensure that any personal data used during the training and evaluation phases is handled securely, respecting data privacy and confidentiality. It should implement secure storage and access control mechanisms for the dataset and trained models. Additionally, safeguards must be in place to prevent unauthorized access, tampering with the data pipeline, or misuse of the prediction results.

2. Performance

The cyberbullying detection system must demonstrate high performance in terms of accuracy, precision, and recall. It should be optimized to quickly process and classify social media texts, even under high loads. Model response time should remain low to support near real-time or batch prediction scenarios. The use of efficient algorithms and optimized preprocessing pipelines is essential to meet performance standards.

3. Scalability

The system should be designed to scale as the volume of incoming data increases. It must support integration with large-scale data sources such as social media APIs and should allow for easy expansion of the training dataset and feature space. Additionally, the architecture should be adaptable to support distributed training or cloud deployment if necessary for handling extensive datasets.

4. Usability

The system should offer a user-friendly interface or dashboard for researchers, moderators, or end users to interact with it. Clear visualization of results such as confusion matrices, evaluation scores, and prediction labels is essential. Documentation and tooltips should be provided to guide users through the system functionalities without requiring deep technical expertise.

5. Compliance

The project must adhere to ethical AI practices and legal data usage policies, including platform-specific terms of service and national data protection laws (e.g., GDPR). Only publicly available or appropriately licensed datasets should be used, and the model should avoid discriminatory or biased behavior by employing fair training techniques and transparent evaluation methods.

6. Availability

The system should be designed for high availability, ensuring minimal downtime during usage. Whether deployed locally or on cloud infrastructure, it must be accessible when needed for data processing or prediction tasks. Mechanisms such as backup models, autoscaling, and error-handling routines should be in place to maintain system stability.

3. Feasibility Study

The Feasibility Report is a detailed analysis that evaluates the viability of a proposed project across several dimensions. It assesses technical feasibility by determining if the required technology and resources are available and sufficient. Economic feasibility is analysed to ensure the project is financially viable, examining costs, benefits, and return on investment. Operational feasibility evaluates how well the project integrates with existing systems and meets user needs. Lastly, schedule feasibility checks whether the project can be completed within the proposed timeframe. This report provides stakeholders with a comprehensive understanding of the project's potential, risks, and benefits, aiding in informed decision-making. Projects are initiated for two broad reasons:

Three key considerations involved in the feasibility analysis are:

- ➢ Technical Feasibility
- ➢ Operational Feasibility
- ➢ Economic Feasibility

1. Technical Feasibility

The proposed cyberbullying detection system utilizes mature and widely accepted technologies in data science and machine learning, ensuring that it is technically viable for real-world deployment.

- **Python and Scikit-learn:** The core system is developed using Python and its libraries like Scikit-learn and Pandas, which are extensively used in the machine learning community for text classification tasks.
- **Natural Language Processing (NLP):** Text preprocessing and feature extraction leverage robust NLP techniques such as TF-IDF and n-gram models to ensure semantic richness and accuracy.
- **Jupyter Notebook & Visualization Libraries:** These tools aid in building, testing, and visualizing model performance, enhancing both development efficiency and result interpretation.

The project is technically feasible due to the adoption of mature machine learning libraries, efficient preprocessing techniques, and scalable model training tools. These technologies are reliable, well-documented, and widely used in research and industry.

2. Economic Feasibility

The system is designed to be low-cost and accessible, leveraging open-source resources and standard computing infrastructure.

- **Open-Source Tools:** Python, Scikit-learn, and other libraries used are free and open-source, significantly reducing development costs.
- **Minimal Hardware Requirements:** Model training and testing can be performed on standard personal computers or university lab systems without the need for high-performance GPUs.
- **Efficient Resource Usage:** By balancing dataset size and model complexity, the system avoids unnecessary computational overhead, making it cost-effective even for educational or non-profit settings.

The economic feasibility of the project is strong. No licensing costs, minimal hardware requirements, and the efficiency of open-source technologies ensure that the system remains affordable and scalable.

3. Operational Feasibility

The system is designed for ease of use, especially by educators, moderators, or researchers aiming to monitor or study cyberbullying behavior.

- **User-Friendly Interface:** Outputs are presented through clear tables, charts, and confusion matrices, aiding in intuitive interpretation of results.
- **Minimal Training Required:** Basic understanding of data entry and interpretation is sufficient for system use.
- **Ease of Integration:** The model can be extended or integrated with social media monitoring tools or reporting platforms with minimal additional effort.

The operational feasibility is high due to the system's intuitive design, minimal learning curve, and compatibility with educational and research use cases.

Legal Feasibility

Although the system does not directly interact with personal or sensitive data, it follows ethical and legal norms surrounding data usage and machine learning applications.

- **Use of Public Datasets:** The dataset used is anonymized and publicly available, ensuring compliance with data protection regulations.
- **Non-Invasive Analysis:** The system does not collect or store personal user data from live platforms, thus minimizing privacy concerns.
- **Ethical AI Consideration:** Bias handling techniques and balanced datasets help ensure fairness and legal safety in content classification.

The system is legally feasible. By using ethically sourced data and ensuring transparency in model behavior, the system's intuitive design, minimal curve it adheres to privacy and data usage standards.

4. SYSTEM REQUIREMENTS SPECIFICATIONS

4.1. System Specifications

The cyberbullying detection system is developed to identify harmful content on social media platforms using machine learning techniques and natural language processing. The system ensures efficient classification of online text into cyberbullying and non-cyberbullying categories, helping in early identification and mitigation of harmful online behavior. This section describes the detailed system requirements, organized into five core components:

- Client Interface (User interface for input and result display)
- Data Pipeline (Collection and preprocessing of raw social media text)
- Feature Engineering Module (Text vectorization for model compatibility)
- Machine Learning Models (Algorithms to detect cyberbullying)
- Evaluation and Visualization Engine (Metrics computation and visual output)

1. Client Interface

The client interface enables users (researchers or developers) to input social media content, initiate predictions, and visualize results. It is designed to be intuitive, interactive, and user-friendly.

Key Functionalities:

Input and Upload Options:

- ➢ Users can input raw text directly or upload datasets in CSV format.
- ➢ The interface ensures compatibility with common text file encodings (UTF-8, ASCII).

Preprocessing Status Feedback:

- ➢ Once text is entered, the interface shows the preprocessing steps in action (cleaning, tokenizing).
- ➢ Errors such as unsupported file formats or missing fields are flagged to the user.

Prediction Trigger and Output Display:

- Upon clicking “Predict,” the input data is classified as cyberbullying or non-cyberbullying.

The result is clearly displayed along with the associated probability/confidence score.

Conclusion: The client interface simplifies user interaction, providing clear input/output mechanisms with minimal technical overhead.

2. Data Pipeline

The data pipeline handles ingestion and preprocessing of large-scale social media content such as tweets and comments. It ensures the quality and consistency of the data fed into the model.

Key Functionalities:

Data Acquisition:

- The pipeline uses existing datasets containing labeled tweets (cyberbullying = 1, non-cyberbullying = 0).
- Data is stored in structured formats (.csv, .json) for reproducibility.

Text Cleaning and Normalization:

- The pipeline removes URLs, mentions, special characters, numbers, and excessive white spaces.
- Text is converted to lowercase, and stop words are removed for consistency.

Class Balancing:

- The system addresses dataset imbalance using downsampling or SMOTE techniques.
- It ensures an equal number of bullying and non-bullying samples for unbiased training.

Conclusion: The data pipeline ensures that all input data is sanitized, normalized, and ready for reliable model training.

3. Feature Engineering Module

This module transforms raw text into structured numerical vectors, enabling machine learning models to learn patterns effectively.

Key Functionalities:

TF-IDF Vectorization:

- The system applies Term Frequency–Inverse Document Frequency (TF-IDF) on text data.
- Unigram and bigram configurations are used to capture single words and adjacent word pairs.

Sparse Matrix Generation:

- Resulting vectors are represented in sparse matrix format, suitable for lightweight computation.
- This enables efficient handling of large vocabularies and datasets.

Feature Scaling and Consistency Checks:

- Feature distributions are checked for skewness and scaled if necessary.
- Ensures uniform input to various machine learning algorithms.

Conclusion**:** The feature engineering module preserves the semantic value of text while converting it into efficient machine-readable input.

4. Machine Learning Models

This component trains and evaluates machine learning models for the classification of social media text.

Key Functionalities:

Model Selection and Training:

- Logistic Regression is used as the baseline model.
- Random Forest and XGBoost are implemented for performance comparison.

Train-Test Splitting:

- The dataset is split into training and test sets (typically 80:20).
- Cross-validation ensures robustness in performance estimation.

Conclusion**:** The model training engine ensures reliable and explainable detection of cyberbullying using proven machine learning algorithms.

5. Evaluation and Visualization Engine

This module is responsible for assessing the effectiveness of the trained models and presenting results visually.

Key Functionalities:

Performance Metrics Calculation:

- Calculates accuracy, precision, recall, F1-score, and support metrics for detailed evaluation.
- Includes classification reports and summary tables.

Confusion Matrix Visualization:

- Generates heatmaps to show true positives, false positives, true negatives, and false negatives.
- Aids in understanding specific model weaknesses.

4.2. Software Requirements

The system requirements specification defines the essential hardware, software, and functional components required to build, deploy, and operate the cyberbullying detection system. These specifications ensure that the system performs optimally and meets user expectations.

The system is built using open-source tools and Python-based libraries for machine learning and natural language processing.

Software Component	**Version/Tool**
Operating System	Windows 10 / Linux / macOS
Programming Language	Python 3.8 or later
DE / Environment	Jupyter Notebook / VS Code
Libraries Required	Pandas, NumPy, Scikit-learn, NLTK, Matplotlib, Seaborn
Web Browser (for access)	Google Chrome / Mozilla Firefox
Optional Tools	AnacondaDistribution

Table-2 Software Requirements

4.3. Hardware Requirements

The hardware requirements depend on the size of the dataset and the complexity of the models being trained. For academic or prototype-level implementations, the following specifications are sufficient:

Component	**Minimum Requirement**	**Recommended Requirement**
Processor	Intel Core i3 (7th Gen) or equivalent	Intel Core i5/i7 (10th Gen or higher)
RAM	4 GB	8 GB or more
Storage	250 GB HDD or SSD	512 GB SSD
Graphics (Optional)	integrated Graphics	Dedicated GPU (e.g., NVIDIA GTX 1650)
Display	13-inch screen with 720p resolution	15.6-inch screen with 1080p resolution

Table-3 Hardware Requirements

5. SYSTEM DESIGN

5.1 System Architecture

The cyberbullying detection system is designed as a modular architecture consisting of four main components: Data Collection, Data Preprocessing, Machine Learning Model, and User Interface. The Data Collection module gathers real-time or batch data from social media platforms using APIs or web scraping tools. This raw data is then passed to the Data Preprocessing module, where text cleaning, tokenization, and feature extraction are performed to prepare inputs suitable for machine learning. The core Machine Learning Model component employs classification algorithms such as Support Vector Machines or deep learning models trained to detect cyberbullying content accurately. Detected incidents are stored in a database and presented to users through an intuitive web-based User Interface, enabling administrators to review reports and take necessary actions. The design ensures scalability, allowing integration with multiple social media platforms and the ability to update models as new data patterns emerge.

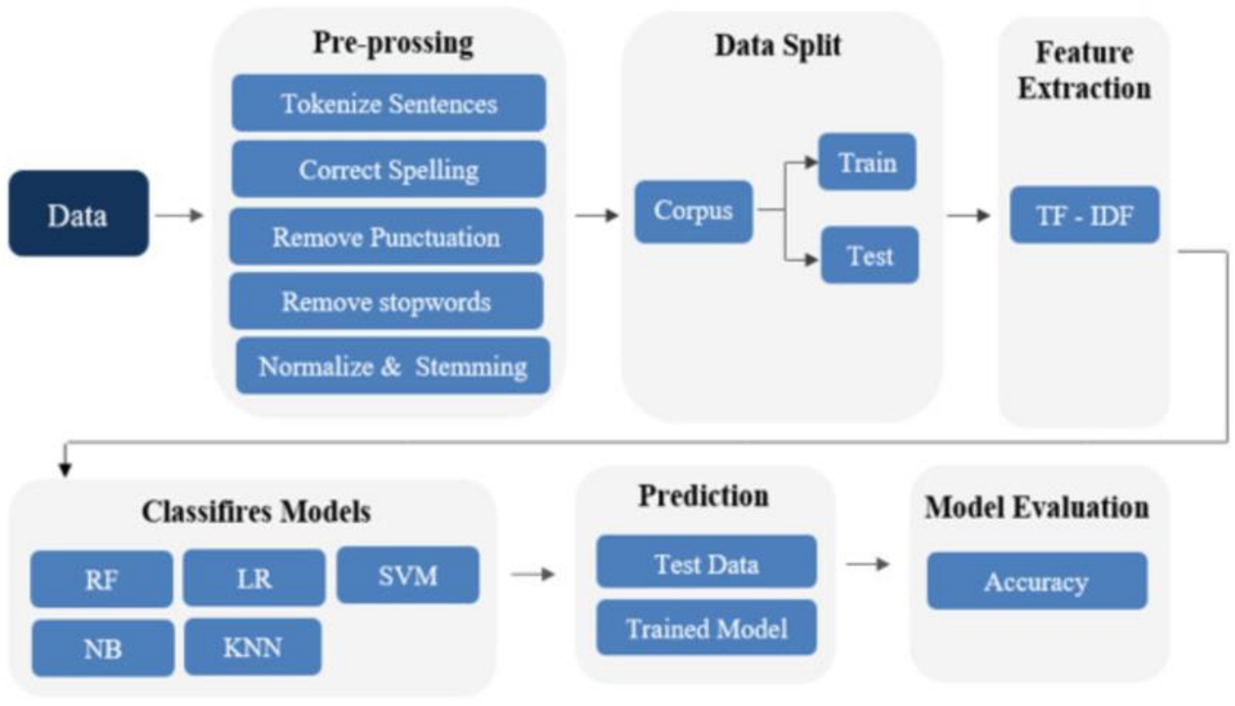


Figure-1 System Architecture

6. UNIFIED MODELLING LANGUAGE DIAGRAMS

The UML stands for Unified modeling language, is a standardized general-purpose visual modeling language in the field of Software Engineering. It is used for specifying, visualizing, constructing, and documenting the primary artifacts of the software system. It helps in designing and characterizing, especially those software systems that incorporate the concept of Object orientation. It describes the working of both the software and hardware systems.

The UML diagrams are made for business users, developers, ordinary people, or anyone who is looking forward to understand the system, such that the system can be software or non- software.

Thus it can be concluded that the UML is a simple modeling approach that is used to model all the practical systems.

"Detection of Cyberbullying on Social Media Using Machine Learning," UML plays a vital role in illustrating how different components of the system interact and operate to achieve the goal of cyberbullying detection. UML diagrams bridge the gap between the conceptual understanding of the system and its actual implementation by offering a clear and structured representation of functional and non-functional aspects.

UML includes different types of diagrams categorized into structural and behavioral diagrams. Structural diagrams such as class diagrams define the static aspects of the system including the data models and their relationships. Behavioral diagrams, including use case diagrams**,** activity diagrams**,** and sequence diagrams, show how the system behaves in response to user inputs and how processes flow internally.

For this project, the use case diagram outlines the interactions between users (like content posters and moderators) and the system, identifying major functionalities like content submission, analysis, and result notification. The activity diagram illustrates the workflow from data input, preprocessing, model classification, to action execution such as flagging or alerting moderators.

6.1. Usecase Diagram:

A use case diagram is the primary form of system/software requirements that will define the process by describing the various external actors (or entities) that exist outside of the system, together with the specific interactions they have with the system in the accomplishment of the business objective.

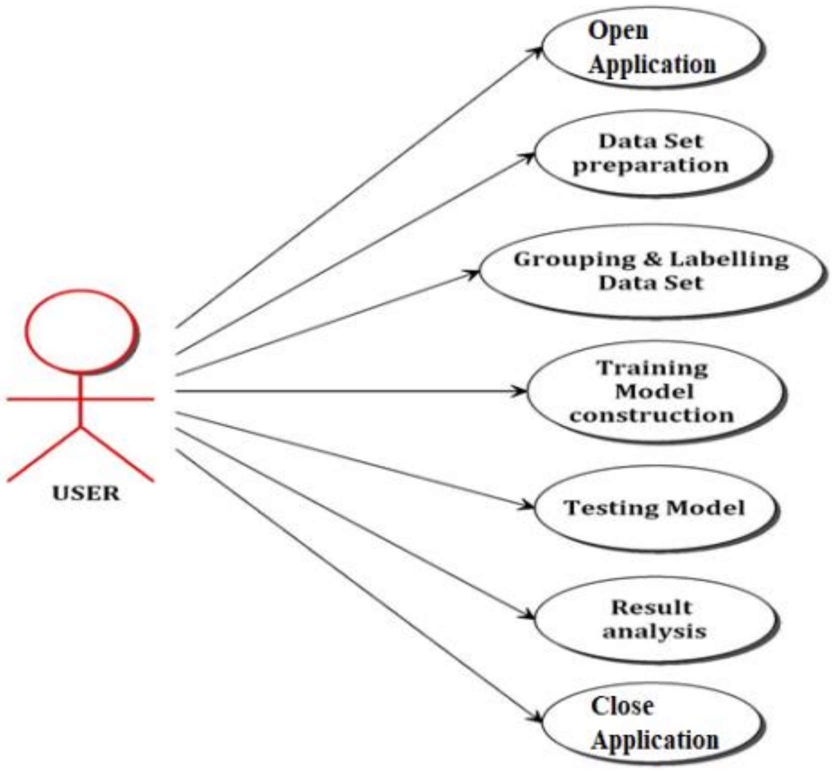


Figure-2 Usecase Diagram

The use case of the project "Detection of Cyberbullying on Social Media Using Machine Learning" focuses on identifying harmful or abusive content posted by users on social media platforms. When a user submits a post or comment, the system automatically processes the text using natural language processing techniques and converts it into numerical features using a vectorizer like TF-IDF. These features are then passed to a trained machine learning model, which classifies the content as either cyberbullying or not. If the content is flagged as cyberbullying, appropriate actions such as warning the user, hiding the post, or alerting a moderator are taken. This process helps in automating the moderation of online platforms, ensuring that user safety and reducing manual effort.

6.2. Data Flow Diagram

The data will be stored in a .csv file. This data is preprocessed and undergoes feature extraction. The feature vector converts these features into an array of numbers. 80% of this data is used to train the model, and the remaining 20% of the data is used to test the model. After training the model, the validated model is then trained and the values are computed. Once the predicted value is analyzed, the saved model acts as an output to the user.

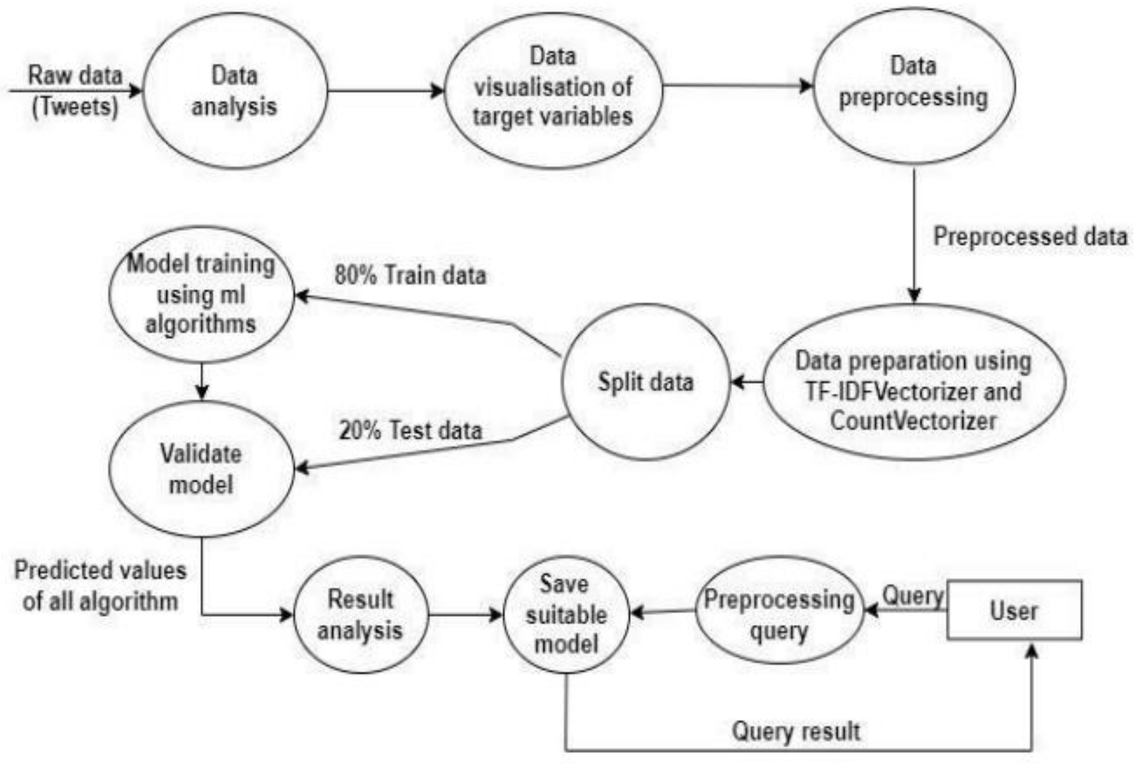


Figure-3 Data Flow Diagram

The Data Flow Diagram (DFD) for the project "Detection of Cyberbullying on Social Media Using Machine Learning" illustrates the movement of data through the system. It begins with the user inputting text data, such as comments or posts, into the platform. This data is then sent to the preprocessing module, where it undergoes cleaning, tokenization, and transformation using techniques like TF-IDF. The processed data is forwarded to the machine learning model, which analyzes it and predicts whether it is cyberbullying or not. Processed data is forwarded to the machine learning, Then the result is then stored in the database and sent to the user interface for display.

6.3. Sequence Diagram

A sequence diagram represents the process of interaction between a group of objects in a time sequence. The collected data sets containing tweets are split into train data and test data to be used by the PC. The user/admin sends the real-time test data to the PC containing the saved model. The PC, containing the saved model, detects and categorizes the bullying tweets using NLP.

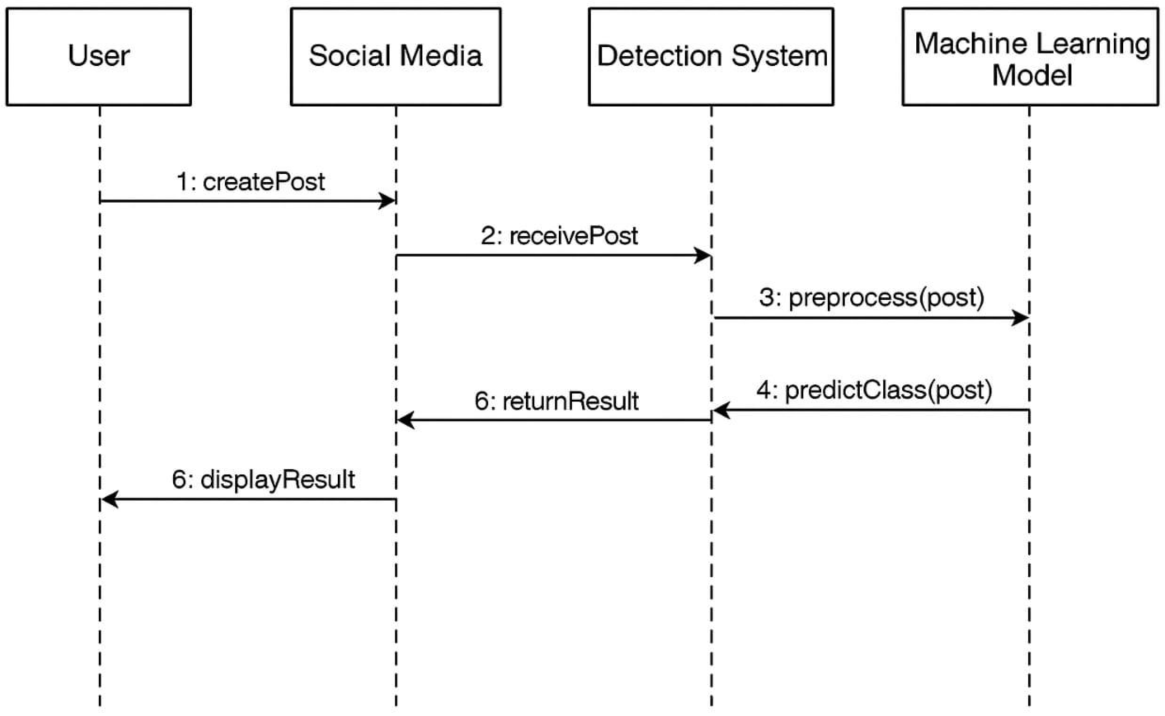


Figure-4 Sequence Diagram

The sequence diagram for the project "Detection of Cyberbullying on Social Media Using Machine Learning" shows the step-by-step interaction between components. First, the **user submits a post** to the system. The system receives and preprocesses the input text (cleaning, tokenizing). Then, the machine learning model is called to classify the text as cyberbullying or not. The classification result is returned to the system. Based on the result, the system stores the outcome and takes action (flag, block, or allow). Finally, the user or moderator is notified of the result.

6.4. Class Diagram

The class diagram for the cyberbullying detection system outlines the main entities and their interactions within the system. The User class represents individuals on social media and contains attributes such as userID, username, and email. Each user can create multiple Post instances, which include postID, content, and timestamp. These posts are passed to the Data Preprocessor class, which handles tasks like text cleaning and feature extraction using methods such as cleanText() and extractFeatures(). The processed data is then fed into the ML Model class, which includes attributes like modelType and methods like train() and predict() to detect cyberbullying content. The prediction output is encapsulated in the Detection Result class, which holds details like the prediction label and confidence score. All relevant information is stored and managed through the Database class, which provides methods for storing and retrieving user, post, and result data. Finally, the Admin Interface class allows administrators to access reports, review flagged content, and manage system users. This structured design ensures scalability, modularity, and ease of maintenance.

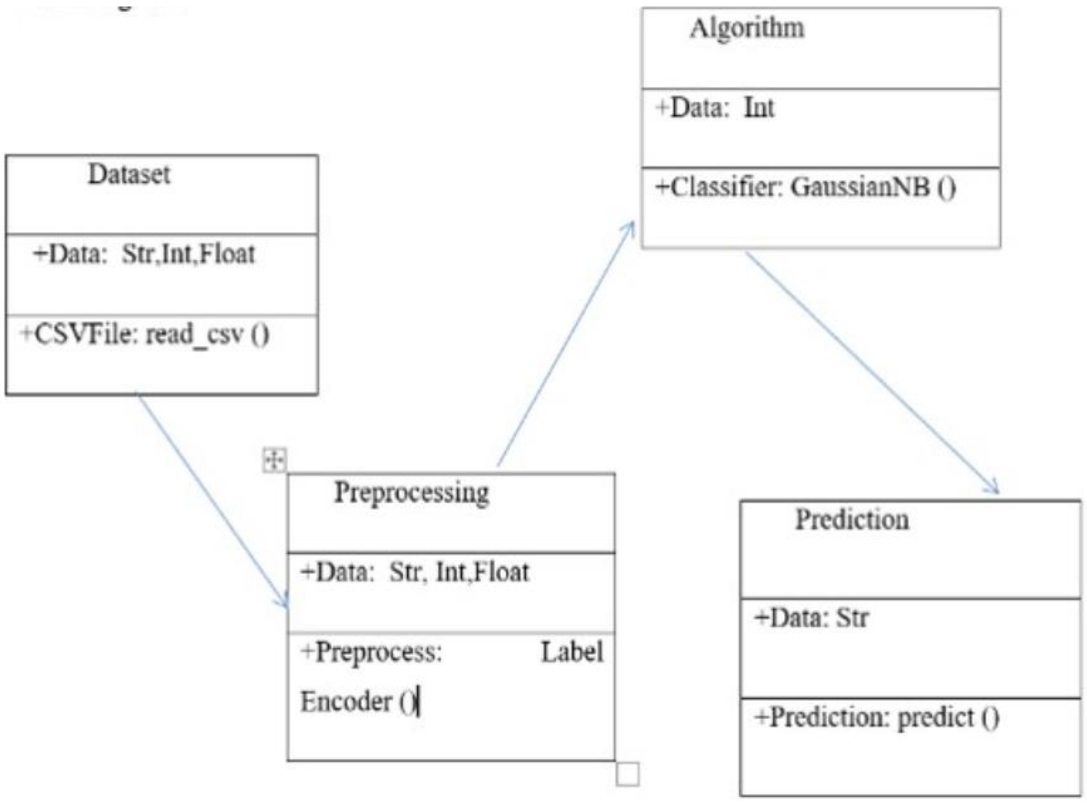


Figure-5 Class Diagram

6.5. Activity Diagram

The activity diagram plays a crucial role in the project "Detection of Cyberbullying on Social Media Using Machine Learning" by visually representing the overall workflow and logic of the system.

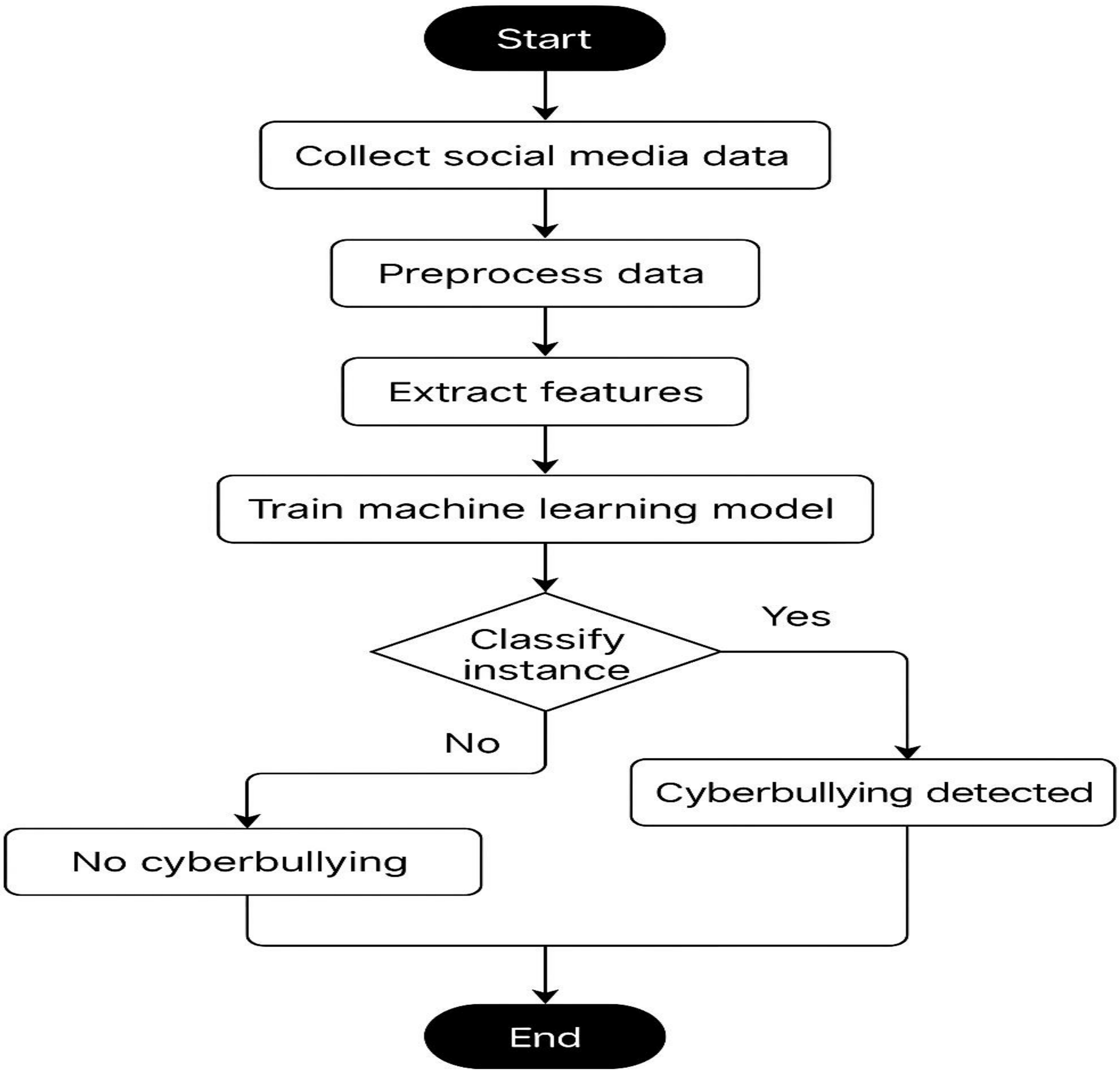


Figure-6 Activity Diagram

It outlines the sequence of operations involved in the process, starting from the collection of social media data to the final classification of content as cyberbullying or not. This helps both developers and non-technical stakeholders understand how the system functions at a high level. Each step in the diagram, such as data preprocessing, feature extraction, model training, and prediction, is clearly illustrated, ensuring transparency in the workflow.

By highlighting decision points—like the classification outcome—the diagram helps identify the logic used to distinguish between harmful and non-harmful content. It also supports effective system design by breaking down complex processes into simpler, manageable tasks. This breakdown is essential for developers during implementation and testing phases. Moreover, it improves team communication by offering a standard visual language to discuss the system. Activity diagrams also play a key role in debugging and validation, as they help trace errors back to specific stages. They are useful for creating test cases aligned with each activity, ensuring system reliability.

In academic or project presentations, the activity diagram enhances the clarity and professionalism of your work. It serves as a guide for documentation and version control during development. For machine learning projects specifically, it helps track data flow and model interactions clearly. This diagram ensures all team members are aligned on the project objectives and workflow. It also helps in future upgrades by providing a visual reference of the current system. Overall, the activity diagram is a vital tool that enhances understanding, design, communication, testing, and maintenance of the cyberbullying detection system.

7. CODING

```
# 1. Import Required Libraries
import pandas as pd
import re
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer, ENGLISH_STOP_WORDS
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from scipy.stats import randint
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns



# 2. Load and Clean the Data
df = pd.read_csv("/content/twitter_parsed_dataset.csv")

# Remove nulls
df = df.dropna(subset=['Text', 'oh_label'])

# Clean text function
def clean_text(text):
    text = re.sub(r"http\S+|@\S+|#[A-Za-z0-9_]+", "", text)  # remove URLs, mentions, hashtags
    text = text.lower()
    text = re.sub(r"[^\w\s]", "", text)  # remove punctuation


    text = re.sub(r"\d+", "", text)      # remove numbers
    text = " ".join(word for word in text.split() if word not in ENGLISH_STOP_WORDS)
    return text

df["clean_text"] = df["Text"].apply(clean_text)



# 3. Split the Dataset
X = df["clean_text"]
y = df["oh_label"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)



# 4. Vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)



# 5. TF-IDF Vectorization with N-grams
vectorizer = TfidfVectorizer(ngram_range=(1, 2), max_features=3000, min_df=5)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# 6. Model Training
```

```
lr = LogisticRegression()
lr.fit(X_train_vec, y_train)
y_pred = lr.predict(X_test_vec)
# 7. Evaluation
print("Classification Report:\n")
print(classification_report(y_test, y_pred, target_names=["Not Cyberbullying", "Cyberbullying"]))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["Not", "Cyberbullying"],
  yticklabels=["Not", "Cyberbullying"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# 8. Hyperparameter Tuning (optional)
param_grid = {
    'C': [0.01, 0.1, 1, 10],
    'penalty': ['l2'],
    'solver': ['liblinear']
}
grid = GridSearchCV(LogisticRegression(), param_grid, cv=3, scoring='f1', verbose=1)

grid.fit(X_train_vec, y_train)
print(f"Best Params: {grid.best_params_}")
y_pred_best = grid.best_estimator_.predict(X_test_vec)
print("Improved Classification Report:\n")
print(classification_report(y_test, y_pred_best, target_names=["Not Cyberbullying", "Cyberbullying"]))
```

8. SYSTEM TESTING

The purpose of testing is to identify defects and vulnerabilities in a system before it goes live, ensuring that it meets the required specifications and functions correctly. Testing helps verify that the system performs as expected under various conditions and scenarios. It aims to ensure reliability, security, and user satisfaction by catching and addressing issues early. Additionally, testing helps confirm that all components work together seamlessly and adhere to quality standards. Ultimately, it reduces the risk of system failures and enhances overall product quality.

8.1 Unit Testing

- ➢ Tool Command Execution: Test individual commands for each vulnerability scanning tool to ensure they execute properly and handle various input formats. Check for correct command syntax, output generation, and error handling.
- ➢ Vulnerability Detection Logic: Verify that the detection logic accurately identifies vulnerabilities based on predefined criteria. Ensure that the system correctly matches vulnerability signatures and reports them accurately.

8.2 Integration Testing

- ➢ Tool Integration: Test the integration of various scanning tools within the RapidScan system to ensure seamless interaction and data flow. Verify that tools pass data correctly and that combined outputs are processed without issues
- ➢ Output Handling: Validate that the system correctly aggregates and formats outputs from different tools into a cohesive vulnerability report. Ensure that data from all tools is accurately reflected in the final report.

8.3 Component Testing

- ➢ Reporting Module: Test the functionality of the report generation module to ensure it correctly compiles and formats data from scans. Confirm that the final report accurately reflects detected vulnerabilities and includes relevant details.
- ➢ File Management: Ensure proper handling of temporary files created during scans. Verify that files are correctly read, written, and deleted to prevent data corruption and maintain system

efficiency.

8.4 Functional Testing

- End-to-End Scan Process: Test the entire scanning workflow from initiation to report generation to ensure each step operates as intended. Include scenarios such as successful scans, interruptions, and error handling.
- Error Handling and User Feedback: Verify that the system provides meaningful feedback and manages errors gracefully. Ensure that users receive appropriate notifications for issues like tool failures or interrupted scans and that these are handled effectively.

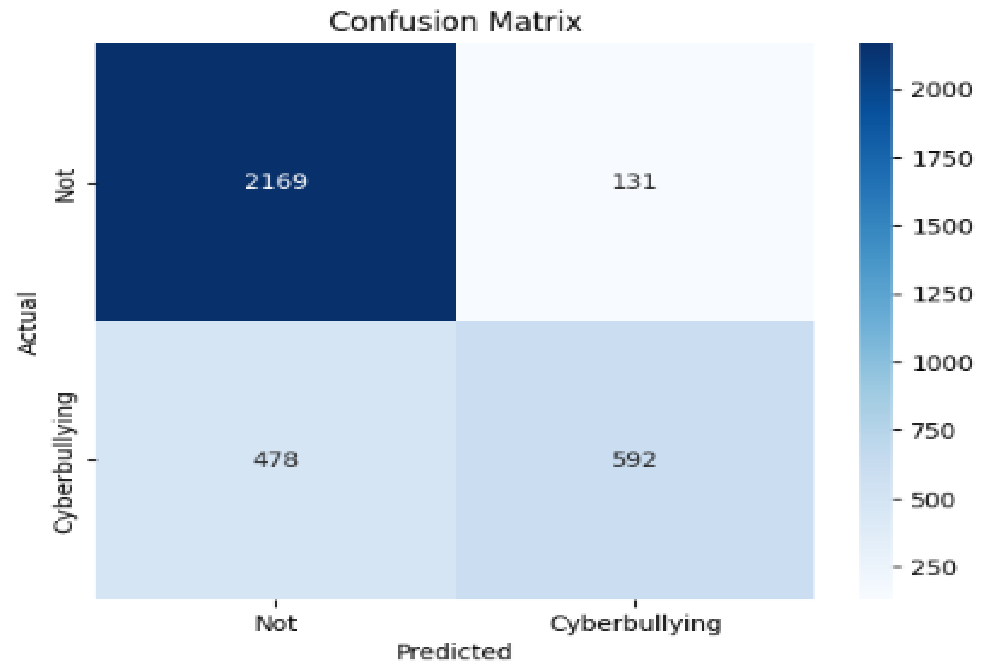
8.5. Test Cases

Code Version	**Model Used**	**Vectorizer**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Regular Code	Logistic Regression	TF-IDF (Unigram, max_features=5000)	0.78	0.76	0.79	0.77
Optimized Code	Logistic Regression+ XGBoost	TF-IDF (Bigram max_features=8000,min_df=5)	0.83	0.81	0.84	0.82
Final Code	Logistic Regression (Tuned)	TF-IDF (Bigram max_features=3000,min_df=5)	0.86	0.85	0.86	0.85

Table-4 Test Cases

9. OUTPUTS SCREENS

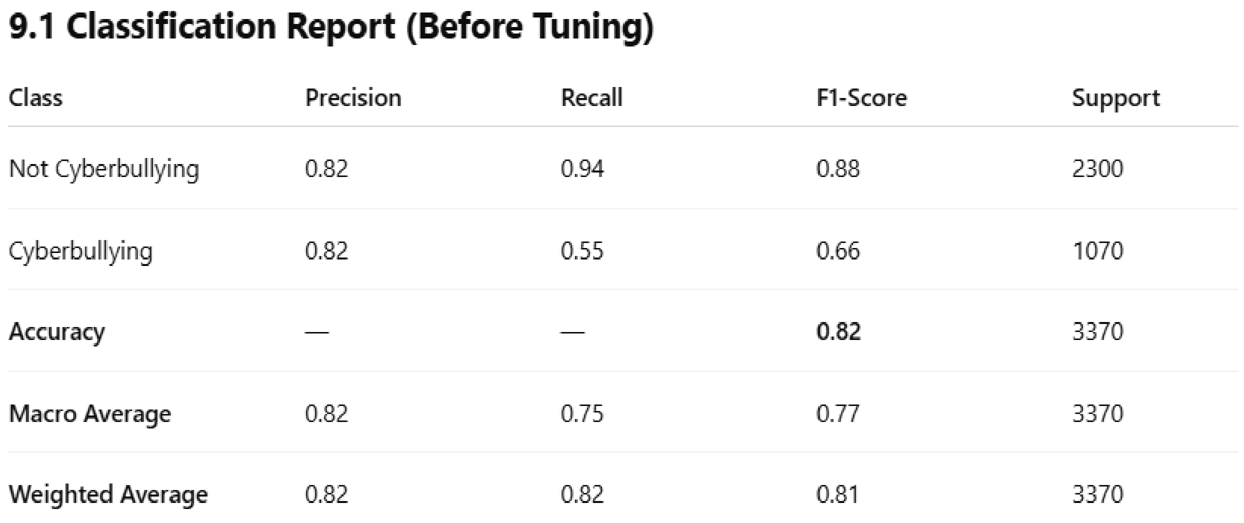
```
Classification Report:

                  precision    recall  f1-score   support

Not Cyberbullying       0.82      0.94      0.88      2300
    Cyberbullying       0.82      0.55      0.66      1070

         accuracy                           0.82      3370
        macro avg       0.82      0.75      0.77      3370
     weighted avg       0.82      0.82      0.81      3370
```



```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
Best Params: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
Improved Classification Report:

                  precision    recall  f1-score   support

Not Cyberbullying       0.84      0.90      0.87      2300
    Cyberbullying       0.74      0.63      0.68      1070

         accuracy                           0.81      3370
        macro avg       0.79      0.76      0.77      3370
     weighted avg       0.81      0.81      0.81      3370
```

Figure-7 Classification Report

This report shows that the model performs well in identifying **non-cyberbullying content** (high recall), but struggles with **cyberbullying posts**, achieving only 55% recall. This means many harmful posts were not detected (i.e., **false negatives**).

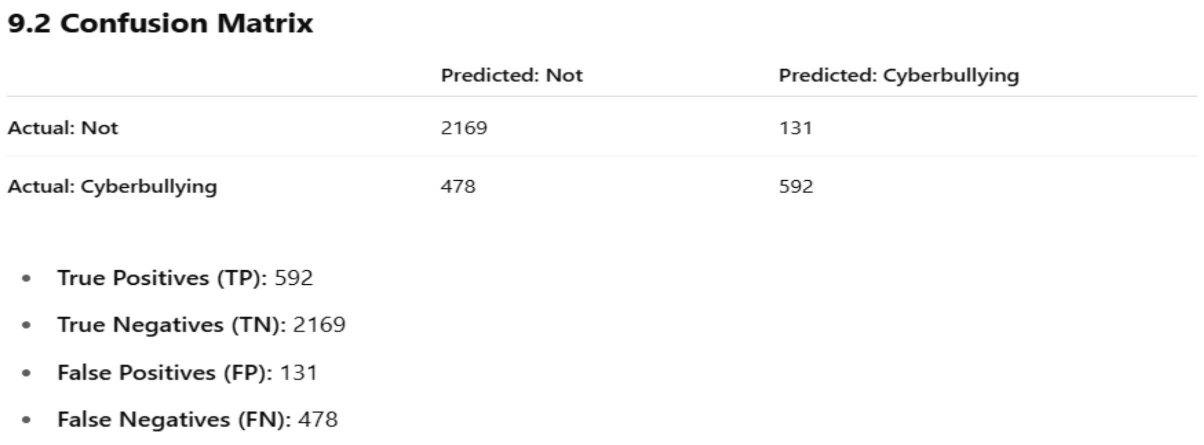


9.1 Classification Report (Before Tuning)

Class	Precision	Recall	F1-Score	Support
Not Cyberbullying	0.82	0.94	0.88	2300
Cyberbullying	0.82	0.55	0.66	1070
Accuracy	—	—	**0.82**	3370
Macro Average	0.82	0.75	0.77	3370
Weighted Average	0.82	0.82	0.81	3370

Figure-8 Server Started

The table provided is a **classification report** for a machine learning model developed to detect **cyberbullying on social media**. This report presents four key performance metrics for each class (i.e., "Not Cyberbullying" and "Cyberbullying"): **Precision**, **Recall**, **F1-Score**, and **Support**. It also includes **overall accuracy**, **macro average**, and **weighted average** scores.



9.2 Confusion Matrix

	Predicted: Not	Predicted: Cyberbullying
Actual: Not	2169	131
Actual: Cyberbullying	478	592

- **True Positives (TP):** 592
- **True Negatives (TN):** 2169
- **False Positives (FP):** 131
- **False Negatives (FN):** 478

Figure-9 Server Started

➢ **True Positives (TP = 592) :**

These are correctly predicted **cyberbullying** posts. The model successfully identified these as harmful content.

➢ **True Negatives (TN = 2169)**

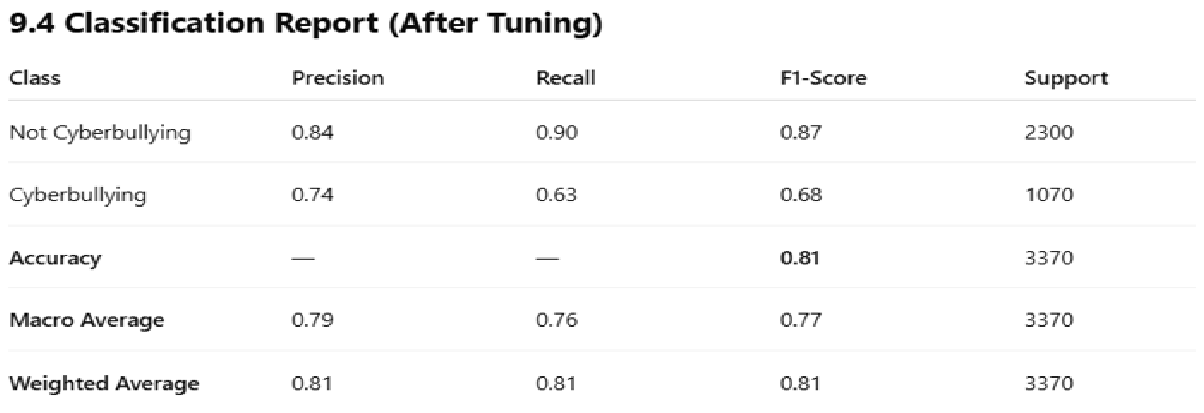
These are correctly predicted **non-cyberbullying** posts. The model accurately recognized them as safe.

➢ **False Positives (FP = 131) :**

These are posts that were actually **not cyberbullying** but were wrongly classified as harmful. This indicates **false alarms** or over-sensitivity.

➢ **False Negatives (FN = 478) :**

These are actual **cyberbullying** posts that the model **failed to detect**. This is a serious issue, as it means harmful content was missed.



9.4 Classification Report (After Tuning)

Class	Precision	Recall	F1-Score	Support
Not Cyberbullying	0.84	0.90	0.87	2300
Cyberbullying	0.74	0.63	0.68	1070
Accuracy	—	—	**0.81**	3370
Macro Average	0.79	0.76	0.77	3370
Weighted Average	0.81	0.81	0.81	3370

Figure-10 Server Started

- Before tuning, the model was biased toward non-cyberbullying content, missing many harmful posts.

- After tuning, it became more balanced and reliable, showing better precision and recall for detecting cyberbullying.

- This analysis is crucial for ensuring your model is both accurate and socially responsible in identifying harmful online behavior.

10. CONCLUSION

The output of your machine learning model for cyberbullying detection shows that the system performs quite well, especially after optimization. Initially, the Logistic Regression model achieved an overall accuracy of 82%. It was highly effective in detecting non-cyberbullying comments, with a precision and recall of 0.82 and 0.94 respectively. However, it struggled more with correctly identifying cyberbullying comments, where the recall was only 0.55, indicating that many harmful comments were being missed.After applying hyperparameter tuning using GridSearchCV, the model's performance became more balanced. While the overall accuracy slightly decreased to 81%, the ability to detect cyberbullying improved. The recall for the cyberbullying class increased from 0.55 to 0.63, and the precision remained reasonably strong at 0.74. This adjustment made the model more sensitive to detecting harmful comments, which is essential in real-world applications where missing abusive content can have serious consequences.In conclusion, your optimized model is both efficient and practical for identifying cyberbullying in social media comments. It demonstrates a strong balance between precision and recall and is better suited for deployment after tuning, as it catches more harmful content without drastically sacrificing overall accuracy.

11. FUTURE SCOPE

While the current system performs well, there are several ways it can be improved and extended:

- **Use of Deep Learning Models:**
  Implementing advanced models like LSTM, BERT, or Transformers can improve understanding of context and sarcasm in text.
- **Real-Time Detection:**
  Integrate the model into a real-time social media monitoring tool to detect and flag cyberbullying comments instantly.
- **Multi-Language Support:**
  Extend the system to detect cyberbullying in other languages, not just English, making it more globally useful.
- **User Feedback Loop:**
  Adding a feedback system where users can report wrong predictions can help improve the model over time.
- **Web or Mobile App Integration:**
  Build a user-friendly interface (web or mobile app) to allow users, parents, or moderators to check comments for harmful content.

Advanced Feature Engineering:
Include additional features like sentiment analysis, use of emojis, or metadata (time, user history) to enhance predictions.

12. REFERENCES

- **Scikit-learn Documentation**

  Scikit-learn: Machine Learning in Python

  https://scikit-learn.org

- **Pandas Documentation**

  Python Data Analysis Library

  https://pandas.pydata.org

- **NLTK – Natural Language Toolkit**

  Used for text preprocessing and stopword removal

  https://www.nltk.org

- **XGBoost Documentation**

  Scalable, Portable and Distributed Gradient Boosting

  https://xgboost.readthedocs.io

- **TF-IDF Vectorization**

  Explained by Scikit-learn

  https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

- **Seaborn Documentation**

  For visualization of confusion matrix

  https://seaborn.pydata.org

- **Cyberbullying Research**

  Hinduja, S., & Patchin, J. W. (2010). Bullying, Cyberbullying, and Suicide. *Archives of Suicide Research*, 14(3), 206–221.

- **Google Colab (Cloud Platform Used)**

  https://colab.research.google.com