Machine Learning Based Cyberbullying Detection on Facebook Comment

by

Md. Ashraful Islam
ID: CSE2301028124

Mohammad Hasib Chowdhury
ID: CSE2301028050

Nadia Islam
ID: CSE2301028177

Md. Sabbir Hossen
ID: CSE2301028141

Supervised by
Md. Shamim Hossain

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)
September 2026

APPROVAL

The thesis titled **"Machine Learning Based Cyberbullying Detection on Facebook Comment"** submitted by Md. Ashraful Islam (CSE2301028124), Mohammad Hasib Chowdhury (CSE2301028050), Nadia Islam (CSE2301028177), and Md. Sabbir Hossen (CSE2301028141) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and content.

Board of Examiners

Md. Shamim Hossain
Lecturer
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Md. Shamim Hossain, Lecturer,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Md. Shamim Hossain)
Supervisor

Signature

Md. Ashraful Islam
ID: CSE2301028124

Mohammad Hasib Chowdhury
ID: CSE2301028050

Nadia Islam
ID: CSE2301028177

Md. Sabbir Hossen
ID: CSE2301028141

ABSTRACT

Cyberbullying has become an important social and technological problem as communication increasingly moves to social networking platforms. Harmful comments can be posted rapidly, distributed to large audiences, and remain visible for long periods. Manual moderation is difficult to scale, especially when comments contain informal expressions. This thesis therefore focuses on the design and development of a multilingual, near-real-time Facebook comment moderation system. The proposed system receives comments through a Facebook webhook or periodic polling, normalizes Bengali, Banglish, and English text without an English-only stemming pipeline, applies a binary machine-learning classifier, and supports policy-based actions such as displaying an alert, placing a comment in a human-review queue, or requesting deletion when authorization and moderation policy permit. Its aim is to reduce the time during which harmful comments remain visible while preserving human oversight for uncertain and high-impact decisions. The another research objective is to compare Logistic Regression and Linear SVC and select the more suitable detector for this application. The active dataset contains 22,975 labeled comments: 11,616 Cyberbullying and 11,359 Safe. Word one-to-two-gram and character three-to-five-gram TF-IDF features are combined in a FeatureUnion. On a stratified 80/20 split, the selected Logistic Regression model achieved 90.84% accuracy, 90.84% macro F1, and 90.98% Cyberbullying F1, with 2,051 true negatives, 221 false positives, 200 false negatives, and 2,123 true positives. It slightly exceeded the combined Linear SVC in accuracy, recall, F1, and macro F1 and provides native probabilities for an adjustable moderation threshold. And we get 84% Accuracy in Real world evaluation. Logistic Regression is consequently treated as one of the best-performing and most suitable algorithms in this experiment, not as universally superior.

This study demonstrates that a relatively lightweight TF-IDF and Logistic Regression pipeline can provide effective cyberbullying screening while remaining computationally practical for web deployment.

ACKNOWLEDGEMENT

First and foremost, we express our deepest gratitude to the Almighty for granting us the strength, patience, and perseverance to complete this project within the scheduled time.

We are profoundly indebted to our supervisor, **Md. Shamim Hossain, Lecturer,** Department of Computer Science and Engineering, Sonargaon University, for his thoughtful guidance, constructive feedback, and unwavering support throughout the development of Cyberbullying Detection And Moderation System. His insights at every stage from problem framing to architectural decisions have shaped this work in ways that go far beyond what these pages can capture.

We extend our sincere thanks to the Honorable Head of the Department, the faculty members, and the staff of the Department of Computer Science and Engineering at Sonargaon University. Their teaching across the four years of our undergraduate study laid the foundation for this project, and their continued encouragement made it possible to attempt something at this scale.

We also thank our peers, classmates, and the wider open-source community whose tools, libraries, and freely shared knowledge made the technical work tractable within an academic timeline.

Finally, our heartfelt gratitude goes to our parents and families for their patience, sacrifices, and endless love. None of what follows would exist without them.

LIST OF ABBREVIATIONS

API	Application Programming Interface
AUC	Area Under the Curve
CV	Cross-Validation
FN	False Negative
FP	False Positive
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LR	Logistic Regression
ML	Machine Learning
NLP	Natural Language Processing
PR-AUC	Precision-Recall Area Under the Curve
REST	Representational State Transfer
ROC-AUC	Receiver Operating Characteristic Area Under the Curve
SVC	Support Vector Classifier
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
UI	User Interface

TABLE OF CONTENTS

Title Page No.

DECLARATION iii

ABSTRACT iv

ACKNOWLEDGEMENT v

LIST OF ABBREVIATIONS vi

CHAPTER 1 1 – 9

INTRODUCTION

1.1 Introduction..... 1

1.2 Background and Motivation..... 1

1.3 Why the System Is Needed..... 2

1.4 Bangladesh and the Multilingual Challenge..... 3

1.5 Human Moderation Burden and Assistance..... 4

1.6 Problem Statement..... 4

1.7 Literature Review..... 5

1.7.1 Cyberbullying Detection Research..... 5

1.7.2 Traditional Machine-Learning Systems..... 5

1.7.3 TF-IDF and Linear Models..... 6

1.7.4 Social-Media-Specific Detection Systems..... 7

1.7.5 Research Gap..... 7

1.8 Objectives..... 8

1.9 Scope and Limitations..... 8

CHAPTER 2 10 – 15

METHODOLOGY

2.1 Introduction..... 10

2.2 System-Oriented Research Method..... 10

2.3 Model-development Methodology..... 11

Title Page No.

2.4 Dataset..... 11

2.5 Data Preprocessing..... 12

2.5.1 Multilingual and Code-Mixed Text..... 13

2.6 TF-IDF Feature Extraction..... 13

2.7 Classification and Hyperparameter Selection..... 14

2.8 Evaluation Metrics..... 15

CHAPTER 3 16 – 24

SYSTEM ARCHITECTURE AND DESIGN

3.1 Introduction..... 16

3.2 Overall Architecture..... 16

3.2.1 Near-Real-Time Moderation Loop..... 16

3.2.2 Detailed Data Flow 17

3.3 Component Design..... 19

3.4 Prediction Sequence..... 19

3.5 API Interaction Sequence..... 20

3.5.1 Manual Comment Prediction..... 21

3.5.2 Retrieval and Processing of Facebook Comments..... 21

3.5.3 Comment Deletion..... 22

3.5.4 Contribution to Near-Real-Time Moderation..... 22

3.6 Security, Privacy, and Failure Handling..... 22

3.7 Production Deployment Architecture..... 23

CHAPTER 4 25 – 28

IMPLEMENTATION

4.1 Development Environment..... 25

4.2 Data Loading and Validation..... 25

4.3 Model Training..... 26

4.4 Backend API..... 26

4.4.1 Comment Ingestion and Event Processing..... 26

Title Page No.

4.4.2 Prediction Response Contract..... 27

4.5 Frontend Interface..... 27

4.6 Polling and Near-Real-Time Operation..... 27

4.7 Storage, Configuration..... 28

CHAPTER 5 29 – 37

RESULTS AND DISCUSSION

5.1 Introduction..... 29

5.2 Experimental Setup..... 29

5.3 Logistic Regression Feature Comparison..... 30

5.4 Classifier Comparison..... 30

5.5 Selected Model Results..... 32

5.6 Cross-Validation and Generalization..... 34

5.6.1 Error Analysis and Moderation Cost..... 34

5.7 System-Level Validation..... 35

5.8 Real-World Evaluation..... 36

5.9 Discussion..... 36

CHAPTER 6 38-39

CONCLUSION AND FUTURE WORK

6.1 Conclusion..... 38

6.2 Future Work..... 39

REFERENCES..... 40 – 41

LIST OF TABLES

Table No.	Title	Page No.
Table 2.1	Dataset composition	12
Table 2.2	Selected hyperparameters	14
Table 2.3	Evaluation-metric definitions	15
Table 3.1	Main system components	19
Table 4.1	Software requirements	25
Table 4.2	Main application operations	27
Table 5.1	Logistic Regression feature comparison	30
Table 5.2	Selected Parameter	30
Table 5.3	Logistic Regression and Linear SVC comparison	31
Table 5.4	Exact performance of the selected combined Logistic Regression model	32
Table 5.5	Confusion-matrix outcomes and moderation implications	34
Table 5.6	System-level validation record	35
Table 5.7	Real-world evaluation	36

LIST OF FIGURES

Figure No.	Title	Page No.
Fig. 1.1	Multilingual text-processing design	3
Fig. 2.1	Model-development workflow	11
Fig. 2.2	Class distribution of the active dataset	12
Fig. 2.3	Word and character TF-IDF FeatureUnion	14
Fig. 3.1	Overall system architecture	16
Fig. 3.2	Near-real-time Facebook moderation loop	17
Fig. 3.3	Detailed Facebook moderation data flow	18
Fig. 3.4	API prediction sequence	19
Fig. 3.5	Interaction sequence among the frontend, FastAPI backend, and Facebook Graph API	20
Fig. 3.6	Proposed production deployment topology	23
Fig. 5.1	Macro-F1 comparison of evaluated models	31
Fig. 5.2	Exact selected-model performance metrics	33
Fig. 5.3	Confusion matrix of the selected model	33

CHAPTER 1

INTRODUCTION

1.1 Introduction

Online social platforms allow people to communicate rapidly and publicly, but the same accessibility can support insults, threats, harassment, and repeated harmful behavior. Cyberbullying is commonly associated with aggressive behavior conducted through digital communication technologies and may reach a victim beyond the physical boundaries of school or work [1]. Research has connected cyberbullying involvement with serious emotional and behavioral consequences, making prevention and timely intervention important [2], [3].

Manual moderation is necessary for difficult cases, yet it becomes slow and inconsistent when thousands of comments arrive. Automatic text classification can assist moderators by screening comments and prioritizing content that may require human attention. Such a tool should not be treated as a final judge. Its purpose is to provide a fast, repeatable first-stage decision that supports responsible moderation.

This project develops a web-based binary classifier. A submitted comment is normalized, represented through TF-IDF features, and classified as Cyberbullying or Safe. The system combines word and character evidence because abusive social-media language often includes phrases, misspellings, abbreviations, deliberate obfuscation, and repeated characters.

1.2 Background and Motivation

The concept of cyberbullying is broader than the presence of an offensive word. Repetition, intent, power imbalance, audience, relationship, and context can affect whether an interaction constitutes bullying. A systematic review by Rosa et al. found that many automatic studies simplify the concept into isolated-message classification and emphasized the need for clearer definitions and realistic evaluation [4]. The present work therefore uses the operational label available in its dataset: whether an individual comment is labeled cyberbullying or non-cyberbullying. It does not claim to infer every social or psychological element of a bullying episode.

The motivation is practical. A lightweight classical model can be trained and executed on ordinary computer hardware, integrated into a web application, and inspected more easily than a very large neural model. Previous work has shown that TF-IDF representations with supervised machine-learning classifiers remain competitive for abusive-language and aggression detection [6], [9].

1.3 Why the System Is Needed

The need for this system begins with scale and exposure. A UNICEF U-Report poll involving more than 170,000 respondents aged 13 to 24 across 30 countries, including Bangladesh, found that one in three respondents had experienced online bullying and one in five reported skipping school because of cyberbullying and violence. Almost three-quarters identified social networks, explicitly including Facebook, as the most common place for online bullying [17]. These findings do not estimate the prevalence among every Facebook user, but they demonstrate that online bullying is widespread among young people and closely connected to the social platforms where public comments are exchanged.

A harmful Facebook comment can be copied, shared, replied to, and viewed by a large audience before a page administrator notices it. Unlike face-to-face abuse, the content may remain accessible continuously and can follow the target into private spaces through a connected device. The resulting burden is not limited to the person who receives the comment. Other users see the language, page communities become less welcoming, and administrators must decide rapidly whether the material violates policy. A detection service positioned close to the comment stream can shorten the interval between publication and review.

The health motivation is also significant. Molero et al. synthesized 13 studies with 7,348 adolescent participants and reported statistically significant associations between cybervictimization and anxiety ($r = .31$) and depression ($r = .28$) [19]. The authors described these effects as moderate to large and emphasized prevention and intervention. These results do not mean that every exposure produces a clinical condition or that automatic deletion is a medical treatment. They do show why delayed or inconsistent handling of online abuse is not merely an administrative inconvenience.

The proposed system helps through early screening. Every received comment can be processed with the same pipeline, suspicious comments can be highlighted immediately,

and moderators can focus attention on the highest-risk material. When a page policy authorizes it, an administrator may delete a clearly harmful comment; otherwise, the system can notify or queue the item for human judgment. This design supports faster response while recognizing that context, humor, quotation, political criticism, and reclaimed language can make fully automatic enforcement unsafe.

1.4 Bangladesh and the Multilingual Challenge

Bangladesh presents a particularly important moderation context because online conversation does not occur in a single standardized language. A Facebook discussion may contain Bengali script, English vocabulary, Banglish transliteration, abbreviations, dialectal spelling, emoji, and several of these forms within one sentence. Banglish has no single accepted spelling: the same Bengali expression may be written in multiple Latin-script forms. Users may also insert spaces, punctuation, digits, or repeated letters to intensify an insult or avoid simple keyword filters.

Research on Bengali abusive-language detection describes Bengali as a low-resource setting in which labeled data, mature preprocessing tools, and standardized benchmarks are more limited than in English. Aurpa, Sadik, and Ahmed studied abusive Bangla comments collected from Facebook with transformer-based models [20], while Hoque and Seddiqui evaluated machine-learning approaches for low-resource Bengali cyberbullying text [21]. Kumar, Lahiri, and Ojha compared aggressive and offensive language identification across Hindi, Bangla, and English, illustrating that language choice materially affects modeling decisions [22].

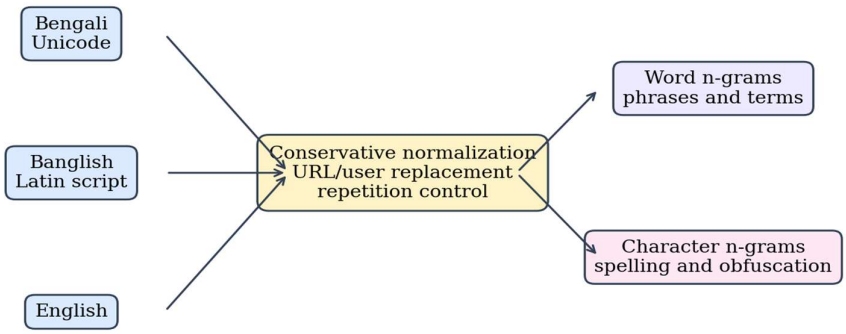


Fig. 1.1 Multilingual text-processing design

This project adopts a deliberately script-tolerant representation. **Fig. 1.1** illustrate, how Bengali characters are retained rather than transliterated or discarded, Latin-script Banglish is preserved, and English words remain available to the same vectorizers. Word n-grams capture recurring terms and short phrases, while character n-grams capture fragments that may remain stable despite spelling variation and deliberate obfuscation. The architecture is therefore multilingual in design.

1.5 Human Moderation Burden and Assistance

Manual moderation provides contextual judgment but does not scale linearly with comment volume. A moderator must read the message, infer tone, inspect surrounding discussion when necessary, compare the content with page policy, choose an action, and sometimes document the reason. Fatigue, time pressure, language variation, and disagreement between reviewers can reduce consistency. During a high-traffic event, even a capable moderation team may develop a queue in which harmful comments remain visible for an unacceptable period.

Automation changes the allocation of attention. Instead of asking a moderator to inspect all comments in arrival order, the classifier can assign a risk score and divide the stream into operational groups. Low-risk items can remain visible while being logged, medium-risk items can be queued for review, and high-risk items can trigger an urgent notification or an authorized action. The threshold is therefore not only a statistical value; it is part of a moderation policy that expresses how the page balances user safety, workload, and the cost of suppressing legitimate speech.

Human oversight remains essential. A false positive may remove criticism, satire, or a quoted statement that does not target another user. A false negative may leave harassment visible and expose a victim to continuing harm. The proposed design records the prediction and gives administrators a review interface so that the model supports rather than replaces accountable decision making. Moderator corrections can later become audited feedback for retraining, subject to privacy and annotation controls.

1.6 Problem Statement

The central problem addressed in this thesis is the automated identification of cyberbullying in social-media comments and the integration of that classifier into a practical moderation workflow. Manual moderation alone does not scale efficiently when the number of

comments is large. An automated system can assist by screening incoming text and highlighting comments that are more likely to require attention.

Three technical problems must be solved. First, raw social-media text must be transformed into a representation that preserves useful lexical information while reducing irrelevant variation. Second, the classifier must generalize beyond the examples on which it was trained. Third, the resulting model must be integrated into an application that can receive individual comments and monitor eligible Facebook Page content without exposing platform credentials.

A further research problem concerns model selection. A model that produces an excellent training score may still perform less effectively on unseen data.

1.7 Literature Review

1.7.1 Cyberbullying Detection Research

Automatic cyberbullying detection has used lexicons, manually engineered features, conventional classifiers, neural networks, and transformer models. Salawu, He, and Lumsden reviewed automated approaches and identified text representation, platform differences, data imbalance, and evaluation consistency as recurring challenges [5]. Rosa et al. likewise observed that dataset construction and the operational definition of cyberbullying can materially influence conclusions [4]. These findings support a transparent pipeline in which data composition, splitting, metrics, and limitations are documented.

1.7.2 Traditional Machine-Learning Systems

Traditional supervised machine-learning systems learn patterns from previously labelled social-media comments. A typical system preprocesses the text, transforms it into numerical features, and trains a classifier such as Logistic Regression, Support Vector Machine, Naïve Bayes, Random Forest, or Decision Tree. Dinakar et al. investigated textual cyberbullying detection using approximately 4,500 YouTube comments. They divided the detection problem into sensitive topics and evaluated binary and multiclass classifiers. Their findings showed that topic-sensitive binary classifiers could be effective for detecting particular forms of cyberbullying [23]. This study was important because it demonstrated that cyberbullying detection could be approached as a supervised text-classification problem. However, separate topic-sensitive models can be more difficult to

maintain than a single binary classifier, particularly when a moderation system must process comments quickly.

TF-IDF is widely used in traditional systems to represent the importance of terms. Word-level TF-IDF captures individual words and short phrases, while character-level TF-IDF captures fragments inside words. Character features are valuable for social-media text because users commonly employ misspellings, repeated characters, informal spellings, abbreviations, and deliberately modified abusive words. Linear classifiers are particularly suitable for these sparse, high-dimensional representations because they can be trained and executed efficiently.

Salawu, He, and Lumsden reviewed automated cyberbullying detection approaches and found that machine-learning and natural-language-processing methods were commonly used to identify characteristics associated with cyberbullying exchanges [5]. Their review also demonstrates that existing approaches differ substantially in feature engineering, classification methods, datasets, and evaluation procedures. Consequently, an algorithm that performs well on one dataset cannot automatically be assumed to be the best algorithm for every platform or language.

1.7.3 TF-IDF and Linear Models

TF-IDF weights a term according to its frequency in one document and its relative rarity across the full corpus. The approach originates from established term-weighting research in information retrieval [10]. Its sparse representation is appropriate for linear classifiers and enables direct use of word and character n-grams. Risch and Krestel used TF-IDF-normalized word and character n-grams with Logistic Regression for aggression identification and explained that character n-grams help recover evidence from obfuscated forms that word features may miss [7].

Rahman, Talukder, and Mithila compared several machine-learning algorithms with TF-IDF for cyberbullying detection, demonstrating that conventional supervised methods provide meaningful performance on labeled text [6]. Gaydhani et al. compared Logistic Regression, Naive Bayes, and Support Vector Machines for hate and offensive-language detection; after tuning, Logistic Regression was the strongest of their evaluated configurations [8]. These studies do not prove universal superiority, but they establish Logistic Regression as one of the effective classical algorithms for related detection tasks.

1.7.4 Social-Media-Specific Detection Systems

Cyberbullying detection has been studied on several platforms, including YouTube, Twitter, Instagram, discussion forums, and other social-networking services. Platform characteristics influence how cyberbullying should be detected. A Twitter post may be short and hashtag-oriented, while an Instagram incident may include an image followed by a sequence of comments. A Facebook comment may contain Bengali script, English words, Banglish transliteration, emojis, abbreviations, reply context, and deliberately obfuscated terms.

Hosseinmardi et al. developed an approach for analysing cyberbullying incidents on Instagram using images and their associated comments [24]. Their work used human-labelled social-media sessions and examined relationships between different features and cyberbullying. This research illustrates that cyberbullying can be a property of a broader interaction rather than a single isolated sentence. Images, comment history, user relationships, and repeated behavior may all provide useful evidence.

Van Hee et al. created English and Dutch social-media corpora and investigated the automatic detection of cyberbullying messages associated with bullies, victims, and bystanders [25]. Their work demonstrated the feasibility of binary cyberbullying classification while also showing the importance of fine-grained annotation. Identifying an offensive sentence is not always equivalent to identifying cyberbullying because bullying may involve a target, harmful intent, repetition, and an imbalance of power.

These studies provide important foundations, but many research systems remain experimental classifiers rather than complete moderation applications. They frequently report classification scores without measuring comment-acquisition delay, dashboard-display time, notification delivery, moderator interaction, deletion authorization, system failures, or end-to-end latency. Therefore, high classification accuracy alone does not establish that a model is suitable for real-world, near-real-time moderation.

1.7.5 Research Gap

Many published systems concentrate on benchmark scores without offering a complete moderation application. Other systems use only word features, which may be fragile when users intentionally alter spelling. This project addresses a practical gap by combining word and character TF-IDF, comparing Logistic Regression with Linear SVC, exposing an adjustable threshold through a web API, and connecting the classifier to an optional

Facebook-comment workflow. The work remains an assistive prototype and does not replace contextual human review.

1.8 Objectives

- Develop a binary classifier that labels comments as Cyberbullying or Safe.
- Build a consistent preprocessing process for training and prediction.
- Combine word-level and character-level TF-IDF representations.
- Compare Logistic Regression and Linear SVC using a held-out test set and five-fold cross-validation to find the best algorithm for our model.
- Deploy the selected classifier through FastAPI and a browser-based interface.
- Support optional collection, display, and moderation of Facebook comments.
- Document limitations and provide a structure for later real-world validation.

1.9 Scope and Limitations

Scope: The scope of the research is text-based cyberbullying detection. The classifier operates on comments represented as text and does not attempt to infer harmful behaviour from images, video, audio, user identity, or social-network relationships. The primary prediction task is binary: Non-Cyberbullying versus Cyberbullying.

The machine-learning scope includes preprocessing, word TF-IDF, character TF-IDF, FeatureUnion, Logistic Regression, Linear SVC, baseline evaluation, five-fold cross-validation, GridSearchCV, and held-out test evaluation. The application scope includes a FastAPI backend, a simple browser interface, saved machine-learning artifacts, and Facebook Page comment monitoring through permitted API access.

The project is intended as an academic prototype and moderation-assistance system. It is not designed to replace human judgement in high-impact moderation decisions. A prediction should be interpreted as an automated screening signal rather than an unquestionable statement about a user's intent.

Limitations:

• The model is dependent on the quality, coverage, and annotation consistency of the dataset.

• Text-only classification cannot fully understand sarcasm, conversation history, images, memes, or implicit threats.

• Social-media vocabulary changes over time, creating a risk of distribution shift.

• Bangla and Banglish variation can reduce the effectiveness of purely lexical representations.
• Facebook access is subject to permissions, platform policies, API limits, and network conditions.
• Periodic polling introduces delay, so the monitoring workflow is described as near-real-time rather than instantaneous real-time.
• Training performance is substantially higher than held-out performance, indicating a generalization gap that should be considered when interpreting results.
• The system is a prototype and should not automatically punish users solely on the basis of a model score.

CHAPTER 2

METHODOLOGY

2.1 Introduction

This chapter describes the methodology used to develop and evaluate the cyberbullying detection system. The research follows a structured experimental pipeline: dataset preparation, preprocessing, TF-IDF feature extraction, classifier training, hyperparameter tuning, held-out testing, comparative analysis, and application integration. The methodology is designed to minimize evaluation bias. The final test set is isolated before tuning, and GridSearchCV is performed only on the training portion. This makes it possible to compare models using the same unseen data and to determine whether optimization produces a meaningful improvement.

2.2 System-Oriented Research Method

Because one of the objective is a working moderation system. So, the methodology combines controlled machine-learning experimentation with software prototype development. The experimental stream determines which representation and classifier should supply the deployed prediction. The engineering stream connects the selected artifact to an API, interface, Facebook ingestion process, event store, and moderation actions. Keeping these streams distinct prevents a successful interface demonstration from being mistaken for evidence of model accuracy, and prevents a high test score from being mistaken for evidence that the complete operational workflow is usable.

Requirements were derived from the moderation scenario. The system must accept a comment, produce a binary decision and risk score, return a response without retraining, expose the result to a reviewer, and handle new Facebook events continuously. It must also fail safely: missing model files, invalid requests, unavailable platform services, or absent authorization must not silently produce a deletion. These requirements guide both the software architecture and the evaluation plan.

The work uses reproducibility controls including a fixed random state, stratified splitting, common preprocessing, explicit hyperparameters, and saved experiment outputs. Every candidate model is evaluated on the same held-out examples.

2.3 Model-development Methodology

The study follows an experimental system-development design. Labeled examples are cleaned and divided into training and testing subsets. Candidate text representations and classifiers are tuned on the training portion through five-fold cross-validation. The held-out test portion is then used for final comparison.

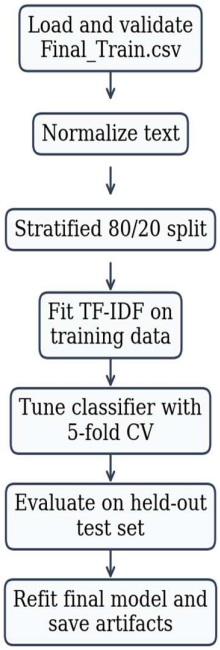


Fig. 2.1 Model-development workflow

The selected pipeline is retrained on all available labeled rows for deployment after evaluation has been recorded. This distinction prevents the deployed retraining step from contaminating the previously reported test results.

2.4 Dataset

According to **Table 2.1** and **Fig. 2.2**, The active dataset is containing the columns text and cyberbullying. The label 0 represents Safe or non-cyberbullying content, while 1 represents Cyberbullying. The dataset has 22,975 rows and the class distribution is close to balanced, which reduces the risk that a classifier can obtain a superficially high accuracy by simply favoring one class. A stratified split with random_state=42 assigns 18,380 rows to training and 4,595 to testing.

Table 2.1 Dataset composition

Class	Label	Count	Percentage
Safe	0	11,359	49.44%
Cyberbullying	1	11,616	50.56%
Total	—	22,975	100.00%

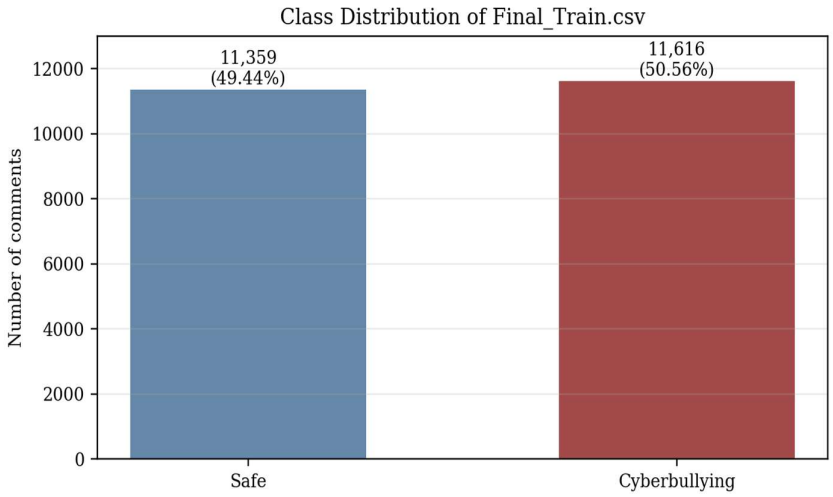


Fig. 2.2 Class distribution of the active dataset

2.5 Data Preprocessing

Preprocessing is applied before TF-IDF extraction. The purpose is to reduce superficial variation while retaining words and character patterns that may be useful for cyberbullying detection. The project preprocessing module normalizes URLs, user mentions, repeated characters, and whitespace.

URLs are replaced with a common URL token, and user mentions are replaced with a common user token. Long repeated characters are reduced so that exaggerated forms do not create an unlimited number of unique features. Multiple whitespace characters are normalized. These operations are intentionally lightweight because aggressive linguistic normalization can remove useful signals in informal abusive language. Character n-grams can benefit from preserving local surface form. The approach also reduces the risk of unintentionally damaging Bangla or Banglish text through an English-only linguistic pipeline.

2.5.1 Multilingual and Code-Mixed Text

Bengali Unicode and Banglish require different surface handling even when they express the same meaning. A Bengali word is represented by characters from the Bengali script, whereas Banglish is a user-created Latin transliteration whose spelling depends on personal habit and phonetic interpretation. English may appear as a complete sentence, a borrowed word, or part of a code-mixed phrase. A rigid language detector followed by separate pipelines could fail on short comments because there may be too little evidence to select one language reliably.

The common feature space avoids making language identification a prerequisite. Character n-grams can learn recurring fragments in either script, and word n-grams can learn frequent whole terms and local combinations. This is particularly helpful for variants produced by repeated letters, omitted vowels, phonetic spelling, or inserted punctuation. Nevertheless, a feature can only be learned if sufficiently represented in labeled training data. Script tolerance is therefore an architectural capability, not proof of equal recall for everylanguage community.

2.6 TF-IDF Feature Extraction

TF-IDF is a numerical statistic used to reflect how important a word (term) is to a document within a collection of documents (corpus). It is widely used in text mining and information retrieval, particularly for feature extraction in machine learning-based text classification tasks such as cyberbullying detection. For term t in document d, TF-IDF is represented conceptually as $TF\text{-}IDF(t,d) = TF(t,d) \times \log(N/DF(t))$, where N is the number of documents and DF(t) is the number containing the term. Implementations use smoothing and normalization details, but the central idea is to reward terms that are frequent in a particular comment and less common across the corpus [10].

The word vectorizer uses ngram_range=(1,2), max_features=30000, min_df=2, and sublinear term frequency. Unigrams capture individual terms, while bigrams preserve short local phrases. The character vectorizer uses analyzer=char_wb, ngram_range=(3,5), max_features=30000, min_df=1, and sublinear term frequency. Character features are helpful for spelling variation, word fragments, and obfuscation [7]. **Fig. 2.3** illustrate how FeatureUnion concatenates the two sparse matrices.

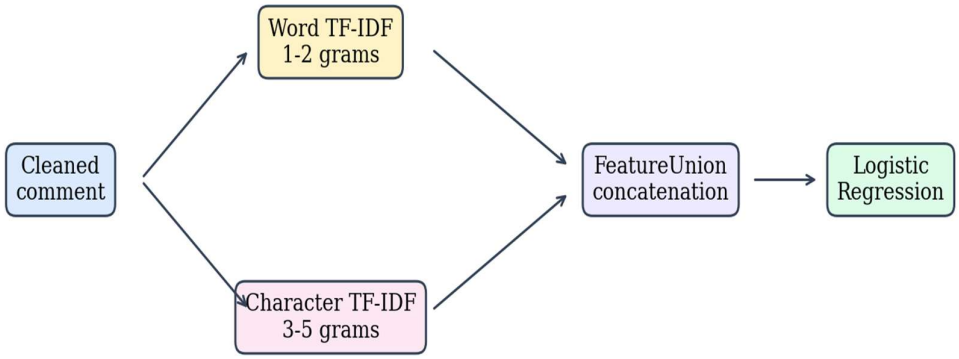


Fig. 2.3 Word and character TF-IDF FeatureUnion

2.7 Classification and Hyperparameter Selection

Logistic Regression models the probability of the positive class through a logistic transformation of a linear decision function. It is computationally efficient for sparse text data, supports regularization, and provides predict_proba output for threshold-based decisions. **Table 2.2** Demonstrate the selected hyper parameter. The selected configuration uses C=10, max_iter=500, and class_weight=balanced. Linear SVC was evaluated as a strong margin-based comparator because support-vector methods have a long history in high-dimensional text categorization [13].

Table 2.2 Selected hyperparameters

Component	Selected configuration
Word TF-IDF	1–2 grams; 30,000 maximum features; min_df=2
Character TF-IDF	char_wb 3–5 grams; 30,000 maximum features; min_df=1
Logistic Regression	C=10; max_iter=500; balanced class weight
Validation	Five-fold cross-validation; macro F1 scoring
Hold-out split	80% training and 20% testing; stratified; random state 42

GridSearchCV was used for the separate word and character searches. After the strongest settings were identified, their combined FeatureUnion configuration was evaluated.

Because the combined run contains one assembled candidate, it should be described as validation of the selected combination rather than a broad grid search over all joint possibilities

.

2.8 Evaluation Metrics

Accuracy measures the overall proportion of correct predictions. Precision measures how many comments predicted as cyberbullying are actually labeled cyberbullying. Recall measures how many labeled cyberbullying comments are detected. F1 is the harmonic mean of precision and recall. Macro F1 computes the mean class-level F1 without allowing the larger class to dominate. ROC-AUC and PR-AUC summarize ranking behavior across thresholds. **Table 2.3** summaries Evaluation-metric definitions. Evaluation literature recommends choosing metrics that match the classification objective and class distribution rather than relying on accuracy alone [14].

Table 2.3 Evaluation-metric definitions

Metric	Definition	Moderation meaning
Accuracy	(TP + TN) / all examples	Overall correctness
Precision	TP / (TP + FP)	Reliability of a bullying alert
Recall	TP / (TP + FN)	Ability to catch harmful comments
F1-score	2PR / (P + R)	Balance of precision and recall
Macro F1	Mean F1 across classes	Equal importance to both labels

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Introduction

The system separates interface, application logic, text-processing code, model artifacts, and optional external-platform integration. This separation makes the classifier easier to retrain without redesigning the interface and permits the prediction logic to be tested independently from Facebook connectivity.

3.2 Overall Architecture

Fig. 3.1 shows that a comment originates from the local web interface or a Facebook event. FastAPI validates the request and passes the text to predict_text(). The preprocessing function normalizes the input, the saved FeatureUnion converts it into a sparse feature vector, and Logistic Regression calculates a cyberbullying probability. The default threshold is 0.50. A result at or above the threshold is returned as Cyberbullying; otherwise, the verdict is Safe. A small rule layer may add a positive flag for selected expressions or nonsense-like input.

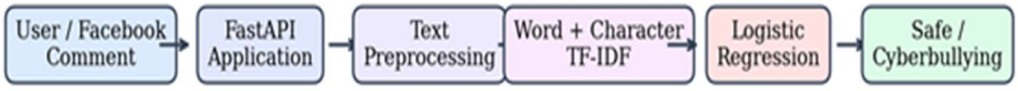


Fig. 3.1 Overall system architecture

3.2.1 Near-Real-Time Moderation Loop

Fig. 3.2 illuminate Near-real-time Facebook moderation loop. The loop starts when a user publishes a Facebook page comment. In webhook mode, the platform sends an event to the configured endpoint; in polling mode, the application periodically requests recent comments. The backend extracts the text and identifiers, obtains a model score, applies the configured policy, stores the event, and exposes it to the dashboard. A safe or low-risk comment remains available. A likely cyberbullying comment can be highlighted for immediate review, sent to a notification channel when one is configured, or deleted through an authorized request after the responsible policy condition is met.

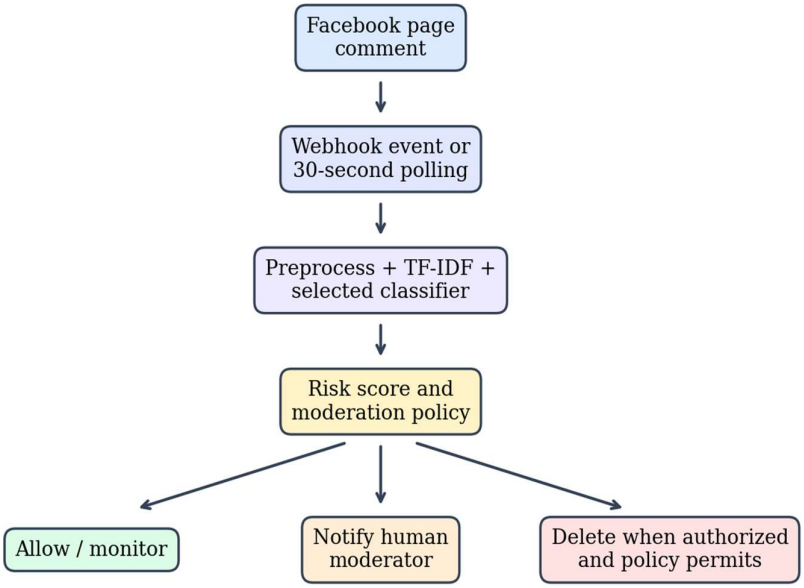


Fig. 3.2 Near-real-time Facebook moderation loop

The diagram separates detection from enforcement because these actions have different risks. Detection estimates whether the text resembles labeled cyberbullying examples. Enforcement decides what to do with that estimate. Keeping a policy layer between them permits conservative deployments to notify only, while a page with carefully validated rules may permit automatic deletion above a higher threshold. This separation also makes it possible to change policy without retraining the classifier.

3.2.2 Detailed Data Flow

Fig. 3.3 shows that the acquisition layer receives platform data but is not allowed to make an enforcement decision. It validates the event shape, extracts a stable identifier and text, and checks whether the event has already been handled. The inference layer then creates model and rule diagnostics. A moderation record joins these diagnostics with the event identifier, time, source, processing state, and any later human action. The dashboard reads moderation records rather than controlling the classifier directly.

This data flow makes duplicate delivery safer. Webhook systems may retry when acknowledgements are delayed, and a polling cycle may rediscover a recent comment. If the identifier already has a completed record, the system should update delivery metadata

without repeating deletion or notification. Idempotent handling is particularly important for destructive actions because a retry must not be interpreted as a second independent decision.

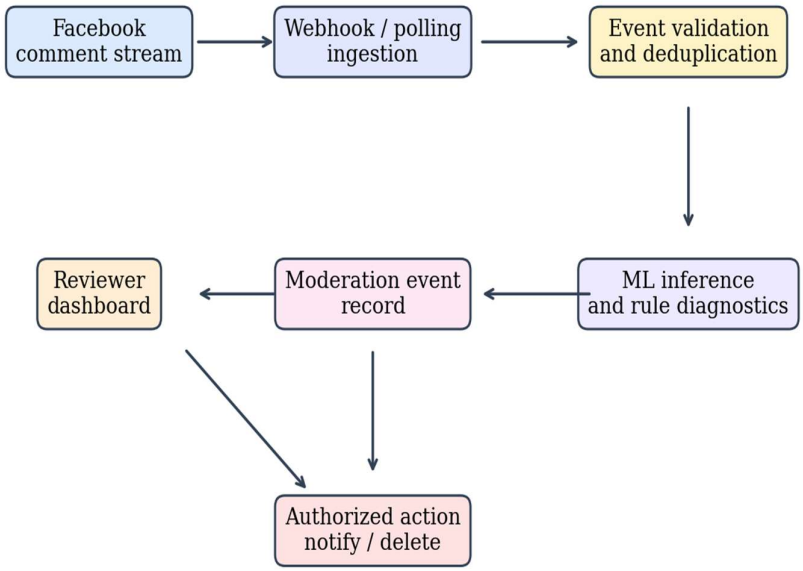


Fig. 3.3 Detailed Facebook moderation data flow

The design also supports later analysis. By keeping the raw model score, threshold, rule flags, final verdict, moderator decision, and action status separate, researchers can measure where errors occur. An inaccurate model score, an overly aggressive threshold, a faulty rule, and an unsuccessful Facebook request are different failure categories and require different corrections.

3.3 Component Design

Table 3.1 Demonstrate Main system components and its responsivities.

Table 3.1 Main system components

Component	Responsibility
api/static/	Browser interface, user input, filters, result display, and polling
api/main.py	FastAPI routes, validation, static serving, and Facebook operations
src/preprocess.py	Shared normalization used by training and prediction
src/train_model.py	Dataset validation, vectorization, model training, evaluation, and saving
src/predict.py	Cached artifact loading, probability calculation, rules, and verdict
models/	Serialized vectorizer and classifier
data/	Active training data and stored Facebook event records

3.4 Prediction Sequence

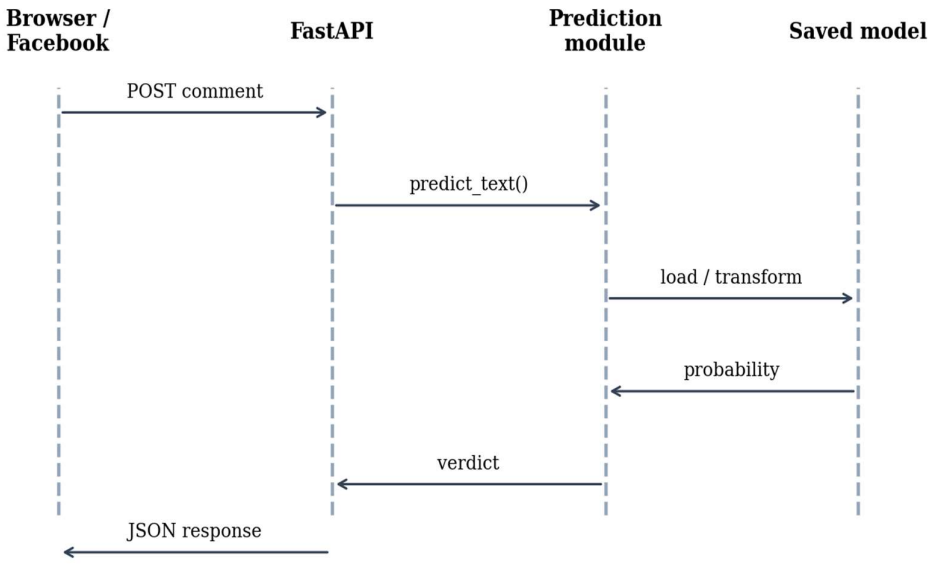


Fig. 3.4 API prediction sequence

Fig. 3.4 Clarify the browser sends JSON containing text and a threshold to POST /predict. Pydantic validates the request structure. The backend calls the prediction module, which loads cached artifacts only once, cleans the comment, transforms it, and obtains the positive-class probability. The API returns the original text, cleaned text, probability values, flags, label, and verdict. Caching reduces repeated disk access during a running session.

A Facebook event follows the same core inference sequence after event-specific validation. This shared path matters because separate implementations could create inconsistent results for identical text. The response is fast at the application level because vectorization and a linear decision are lightweight operations, but end-to-end speed also includes network delivery, event parsing, storage, dashboard refresh, and any platform action.

3.5 API Interaction Sequence

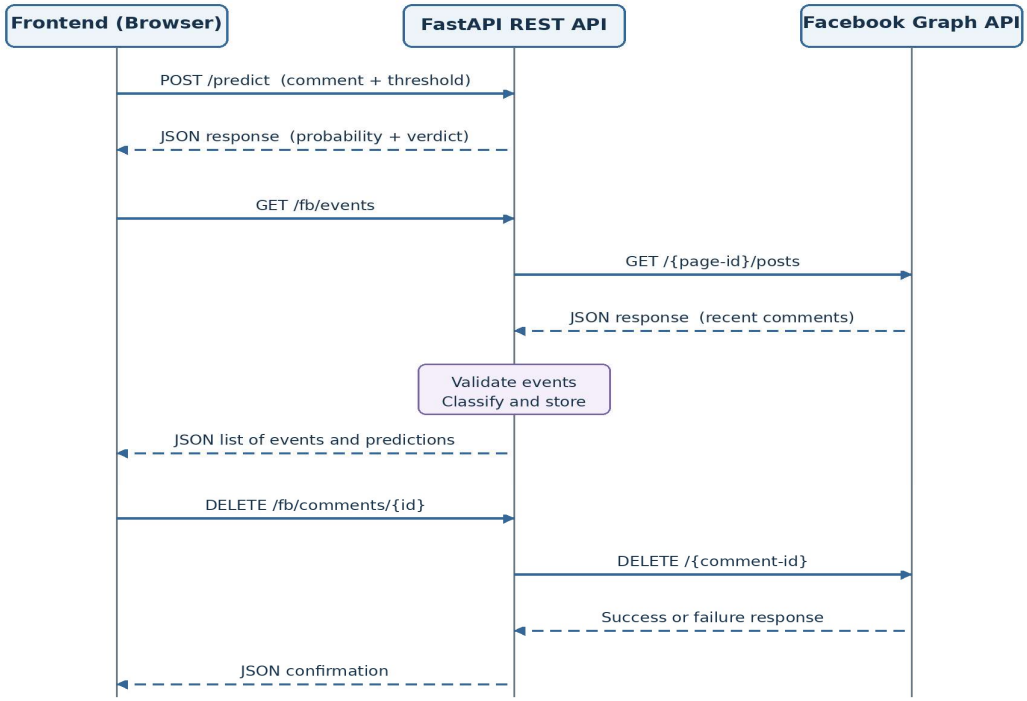


Fig. 3.5 Interaction sequence among the frontend, FastAPI backend, and Facebook Graph API

Fig. 3.5 illustrates the sequence of interactions among the frontend application, the FastAPI REST API, and the Facebook Graph API. The FastAPI backend acts as the central communication and processing layer. Therefore, the frontend does not directly access the trained machine-learning model, database, Facebook access token, or Facebook Graph API. This separation improves system security, maintainability, and control over moderation operations.

3.5.1 Manual Comment Prediction

For manual prediction, the frontend sends a POST /predict request to the FastAPI backend. The JSON request contains the comment text and, where applicable, the selected decision threshold. FastAPI first validates the request to ensure that the supplied text is present and correctly formatted.

The validated comment is then passed through the same preprocessing and feature-extraction pipeline used during model training. Word-level and character-level TF-IDF features are generated through the FeatureUnion, after which the trained Logistic Regression classifier estimates the probability that the comment contains cyberbullying. The rule layer is subsequently applied to produce the final system decision. The backend returns a JSON response containing the prediction probability and binary verdict, such as Cyberbullying or Non-Cyberbullying.

3.5.2 Retrieval and Processing of Facebook Comments

The frontend requests stored Facebook events and their prediction results through the GET /fb/events endpoint. When Facebook polling is enabled, the backend communicates with the Facebook Graph API to retrieve recent page posts and comments. The received JSON objects are validated before being accepted by the system.

The backend checks event identifiers to prevent the same Facebook comment from being processed more than once. Each newly received comment is classified using the common machine-learning and rule-based pipeline. The comment information, prediction probability, final verdict, processing time, and moderation status are then stored in the database. Finally, the backend returns a JSON list of processed events and predictions to the frontend for presentation on the moderator dashboard.

3.5.3 Comment Deletion

When a moderator decides to remove a harmful comment, the frontend sends a DELETE /fb/comments/{id} request to the FastAPI backend. The backend verifies the request and checks whether the necessary Facebook authorization is available. It then forwards the deletion request to the Facebook Graph API using the corresponding Facebook comment identifier.

The Graph API returns either a success or failure response. FastAPI records the result of the moderation action and sends a JSON confirmation to the frontend. A failed deletion request must not be presented as successful; instead, the interface should display an appropriate error message so that the moderator can take further action.

The prediction and deletion operations are intentionally separated. A classification result identifies a potentially harmful comment, but deletion remains an authorized moderation action. This design reduces the possibility of legitimate comments being removed automatically because of an incorrect prediction.

3.5.4 Contribution to Near-Real-Time Moderation

The interaction sequence supports near-real-time moderation by automatically collecting, classifying, storing, and displaying newly received Facebook comments with limited delay. Page administrators can examine flagged comments and initiate moderation actions from a single interface. However, the system is described as near-real-time rather than strictly real-time because its response time depends on network latency, Facebook API availability, polling frequency, database operations, and model-processing time.

3.6 Security, Privacy, and Failure Handling

The system stores Facebook credentials in environment variables instead of directly including them in the source code. The FastAPI backend controls communication between the browser, trained model, and Facebook Graph API, preventing sensitive access tokens from being exposed to the frontend. The system also checks Facebook webhook verification tokens and requires a configured page token before submitting deletion requests.

For privacy, only information necessary for comment detection and moderation should be collected. The prototype stores recent Facebook events in local JSON files, including comment identifiers, text, prediction results, and limited author information. These files

should be protected from unauthorized access. Any comments used for future model development should be anonymized by removing names, profile information, and Facebook identifiers.

The system handles several common failures explicitly. Missing model files produce an HTTP 503 error instead of classifying the comment as Safe. Facebook requests use a 15-second timeout, and polling continues after temporary network or API failures. Duplicate event identifiers are updated rather than stored repeatedly. Failed deletion requests are reported to the moderator and are not presented as successful actions.

These protections are suitable for a controlled academic prototype. However, production deployment would require moderator authentication, role-based authorization, webhook-signature validation, encryption, structured audit logs, data-retention controls, and more detailed system-health monitoring.

3.7 Production Deployment Architecture

Fig. 3.6 shows Proposed production deployment topology The current project can run as one FastAPI process for demonstration, but a production arrangement should separate public ingress, application execution, model inference, persistent records, and moderator access. A reverse proxy terminates HTTPS, restricts routes, and applies request limits. The FastAPI service verifies webhook requests and exposes authenticated moderation endpoints. Model workers load versioned artifacts and perform prediction. A transactional database stores events and audit information, while the moderator interface uses authenticated API calls.

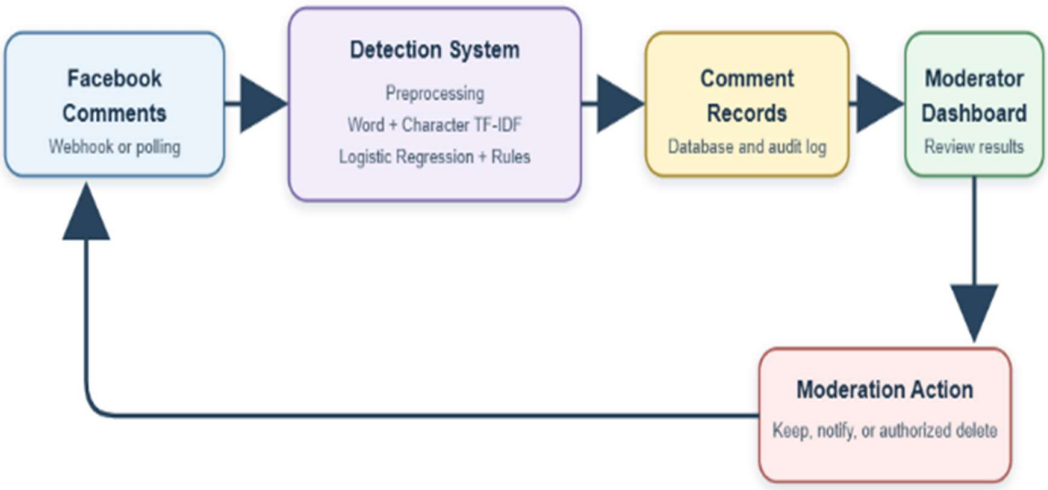


Fig. 3.6 Proposed production deployment topology

A queue between ingestion and inference is recommended when comment volume is bursty. The webhook can acknowledge valid delivery after safely recording the event, while workers process items independently. Queue depth then becomes a measurable indicator of delay. Dead-letter handling preserves events that repeatedly fail, and idempotency keys prevent one comment from triggering multiple actions after retries.

CHAPTER 4

IMPLEMENTATION

4.1 Development Environment

Table 4.1 simplify Software requirements. The implementation uses Python, pandas, NumPy, scikit-learn, joblib, FastAPI, Uvicorn, requests, python-dotenv, and a static HTML/CSS/JavaScript interface. Scikit-learn supplies interoperable tools for feature extraction, model selection, linear classification, and evaluation [15]. FastAPI supplies request validation and asynchronous web capabilities suitable for a compact machine-learning service [16].

Table 4.1 Software requirements

Software	Role
Python 3.12	Programming language and runtime
pandas / NumPy	Dataset manipulation and numerical operations
scikit-learn	TF-IDF, FeatureUnion, splitting, tuning, metrics, and classifiers
joblib	Serialization of trained model artifacts
FastAPI / Uvicorn	HTTP application and development server
HTML / CSS / JavaScript	Browser-based moderation interface
requests / python-dotenv	Facebook HTTP calls and environment configuration

4.2 Data Loading and Validation

The training module resolves paths relative to the project root and loads data/Final_Train.csv. It verifies that text and cyberbullying exist, converts labels to integers, casts comments to strings, and applies clean_text. The loader also recognizes spreadsheet-format files and can expand a malformed single-column CSV representation. Explicit validation provides clear error messages instead of allowing silent column mismatch.

4.3 Model Training

The dataset is split using stratification so both partitions preserve class proportions. The vectorizer is fitted only on training text for evaluation, then transforms the held-out test text. Logistic Regression is fitted on the training vectors, and classification_report summarizes test performance. After evaluation, a new FeatureUnion and classifier are fitted on all labeled rows, then saved as models/tfidf.joblib and models/clf.joblib for deployment. This two-stage process is important: the test score must come from a model that never fitted the test examples, while the production artifact may use all labeled examples after the experimental result has been fixed. Any later model change requires a new evaluation rather than reusing the old score.

4.4 Backend API

The backend exposes the main prediction route along with status, Facebook webhook, event retrieval, and deletion routes. A TextIn schema defines the request fields, while a Prediction schema documents the response. StaticFiles and FileResponse serve the browser interface from the same application, simplifying local demonstration. Facebook events are retained in bounded JSON files for the prototype.

4.4.1 Comment Ingestion and Event Processing

Webhook verification establishes that the configured endpoint belongs to the application. After verification, page-feed events can be received as HTTP requests. The event handler extracts the fields required for moderation, rejects or ignores unrelated structures, and obtains a prediction for comment text. Polling provides a second acquisition mode for demonstrations or environments where webhook delivery is not available. The polling task requests recent comments at the configured interval and passes unseen identifiers through the same processing path.

Using one processing function for webhook and polling inputs is important because acquisition mode should not change the model decision. Event identifiers are used to reduce repeated processing, while stored records contain the text, platform metadata needed by the interface, prediction output, and action state. Storage is bounded so the prototype does not grow indefinitely.

4.4.2 Prediction Response Contract

Table 4.2 shows that the Main application operations. The response contract is designed for both human display and machine integration. It includes normalized text, class probabilities or scores, the configured threshold, rule indicators, and the final binary verdict. Returning the threshold makes an audit record interpretable: the same probability can produce different actions under different policies. Returning rule indicators also exposes whether the statistical classifier or a handcrafted condition caused the alert.

Table 4.2 Main application operations

Interface operation	Input	Output or effect
Predict comment	Text and optional threshold	Cyberbullying or Safe verdict
Receive Facebook event	Facebook webhook event payload	Stored event with prediction
List Facebook events	Filter and pagination options	Recent stored events and their prediction results
Delete comment	Authorized comment identifier	Platform deletion request and recorded status
Health/status check	No input required	Returns a basic {"status": "ok"} response

4.5 Frontend Interface

The frontend lets a user type a comment, choose a threshold, send a prediction request, and view the returned verdict. A separate Facebook section periodically obtains stored events, supports result filtering and pagination, displays author or page metadata when available, and offers deletion when credentials permit. Clear binary labels are used consistently throughout the interface and API response.

4.6 Polling and Near-Real-Time Operation

The monitoring workflow uses webhook and periodic polling. Polling means that the application makes a request at a configured interval to check for new comments or events.

This is different from an event-driven push architecture in which a platform sends a notification when an event occurs.

Polling is easy to understand and implement, but it introduces a delay between a comment being created and the application checking for it. The shorter the polling interval, the lower the expected detection delay, but frequent requests can increase API usage and may be constrained by rate limits. While webhook provide near instant delivery. But network latency, server processing and prediction pipeline introduce small but measurable delays.

For this reason, the thesis uses the term near-real-time rather than real-time.

4.7 Storage Configuration

The system stores Facebook event information in JSON files such as data/fb_events.json and raw event payloads in a separate file where configured. Model artifacts are stored under the models directory. This lightweight storage approach is suitable for an academic prototype but would need to be replaced with a database or durable event store for a production-scale system.

Configuration is handled using environment variables loaded through python-dotenv. Sensitive values such as Facebook access tokens should never be placed in frontend JavaScript or committed to Git. Logs should also avoid printing full access tokens or unnecessary personal information.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

This chapter presents the documented experimental results. The evaluation begins with independent word-level and character-level TF-IDF experiments using Logistic Regression, followed by a combined FeatureUnion model. Linear SVC is then compared with the Logistic Regression results. The analysis emphasizes held-out test performance, cross-validation behavior, confusion matrices, and the gap between training and test performance.

The results show that character-level features outperform word-level features in the individual Logistic Regression experiments. The combined Word + Character representation performs best among the reported Logistic Regression configurations. Linear SVC also performs strongly, especially with character features, but the combined Logistic Regression model achieves the strongest final test F1-score among the documented models.

5.2 Experimental Setup

All reported experiments use the same 22,975-row binary dataset, an 80/20 stratified split, and five-fold cross-validation on the training portion. Macro F1 is the primary selection measure because it gives equal weight to Safe and Cyberbullying. Word and character feature settings were tuned separately to control computational cost, after which the selected settings were combined.

The experiment is essential to find the best model for system. The service needs one reproducibly selected model. Logistic Regression and Linear SVC receive comparable word, character, and combined representations. The comparison uses point estimates on one held-out test partition, so small differences are interpreted cautiously. Selection additionally considers probability availability, recall of the harmful class, and compatibility with threshold-based moderation.

5.3 Logistic Regression Feature Comparison

Table 5.1 Logistic Regression feature comparison

Feature representation	Test accuracy	Cyberbullying F1	Macro F1	ROC-AUC	PR-AUC
Word TF-IDF	89.40%	89.63%	89.40%	95.69%	95.49%
Character TF-IDF	90.36%	90.51%	90.36%	96.68%	96.49%
Word + character TF-IDF	90.8379%	90.9792%	90.8356%	96.8848%	96.8265 %

Table 5.1 demonstrate Character TF-IDF improved macro F1 by 0.96 percentage points over word TF-IDF. Combining both representations produced a further 0.48-point improvement and the strongest Logistic Regression result. This supports the design assumption that word sequences and subword surface patterns provide complementary evidence. Risch and Krestel similarly observed that character n-grams can capture obfuscated aggressive forms that word-based features may not represent effectively [7].

5.4 Classifier Comparison

Table 5.2 shows The Selected parameter for Both Logistic Regression and Linear SVC and **Table 5.3** compared using five-fold cross-validation and GridSearchCV on the same dataset.

Table 5.2 Selected parameter

Model	Word Features	Character Features	C Value
Logistic Regression	n-gram: (1,2)Max Features: 30,000Min DF: 2Sublinear TF: True	Analyzer: char_wbn-gram: (3,5)Max Features: 30,000Min DF: 1Sublinear TF: True	10
Linear SVC	n-gram: (1,2)Max Features: 30,000Min DF: 1Sublinear TF: True	Analyzer: char_wbn-gram: (3,5)Max Features: 30,000Min DF: 1Sublinear TF: True	1

Table 5.3 Logistic Regression and Linear SVC comparison

Feature set	Classifier	Accuracy	Cyberbullying precision	Cyberbullying recall	Cyberbullying F1	Macro F1
Word	Logistic Regression	89.40%	88.67%	90.62%	89.63%	89.40%
Character	Logistic Regression	90.36%	90.10%	90.92%	90.51%	90.36%
Combined	Logistic Regression	90.84%	90.57%	91.39%	90.98%	90.84%
Word	Linear SVC	89.16%	88.42%	90.40%	89.40%	89.16%
Character	Linear SVC	90.45%	90.26%	90.92%	90.59%	90.44%
Combined	Linear SVC	90.66%	91.00%	90.49%	90.74%	90.66%

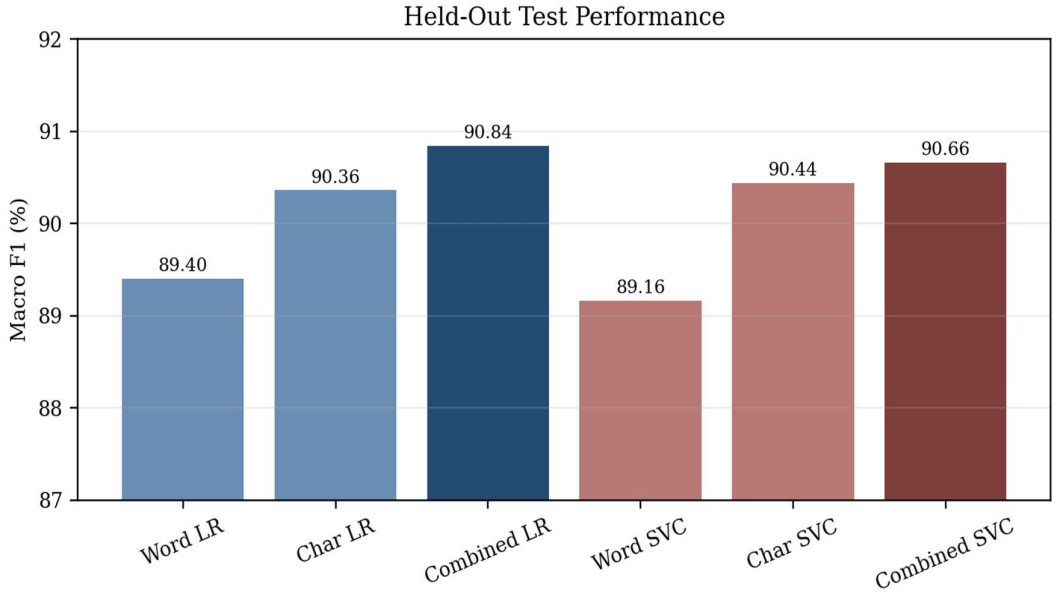


Fig. 5.1 Macro-F1 comparison of evaluated models

Fig. 5.1 relate two combined classifiers are close. Linear SVC gives slightly higher cyberbullying precision, whereas Logistic Regression gives higher recall, F1, accuracy, and

macro F1. Logistic Regression misses 200 labeled cyberbullying comments compared with 221 for Linear SVC. In a moderation-support setting, reducing false negatives is valuable because fewer harmful comments are passed as Safe. Logistic Regression also provides native probabilities for the adjustable threshold. It is therefore selected as the final classifier and described as one of the effective classical algorithms for detection, not as universally superior.

5.5 Selected Model Results

Table 5.4 and **Fig 5.2** shows the combined Logistic Regression model correctly classified 4,174 of 4,595 held-out examples. Using the researcher-supplied output, accuracy is 0.908379, positive-class precision is 0.905717, recall is 0.913904, F1-score is 0.909792, and macro F1 is 0.908356. ROC-AUC is 0.968848 and PR-AUC is 0.968265. For Safe comments, the earlier classification report gives precision of approximately 91.12%, recall of 90.27%, and F1 of 90.69%. The close class-level performance is consistent with the macro F1 of 90.8356%.

Table 5.4 Exact performance of the selected combined Logistic Regression model

Measure	Exact value	Percentage
Accuracy	0.908379	90.8379%
Precision	0.905717	90.5717%
Recall	0.913904	91.3904%
F1-score	0.909792	90.9792%
Macro F1	0.908356	90.8356%
ROC-AUC	0.968848	96.8848%
PR-AUC	0.968265	96.8265%
True Negative	2051	—
False Positive	221	—
False Negative	200	—
True Positive	2123	—

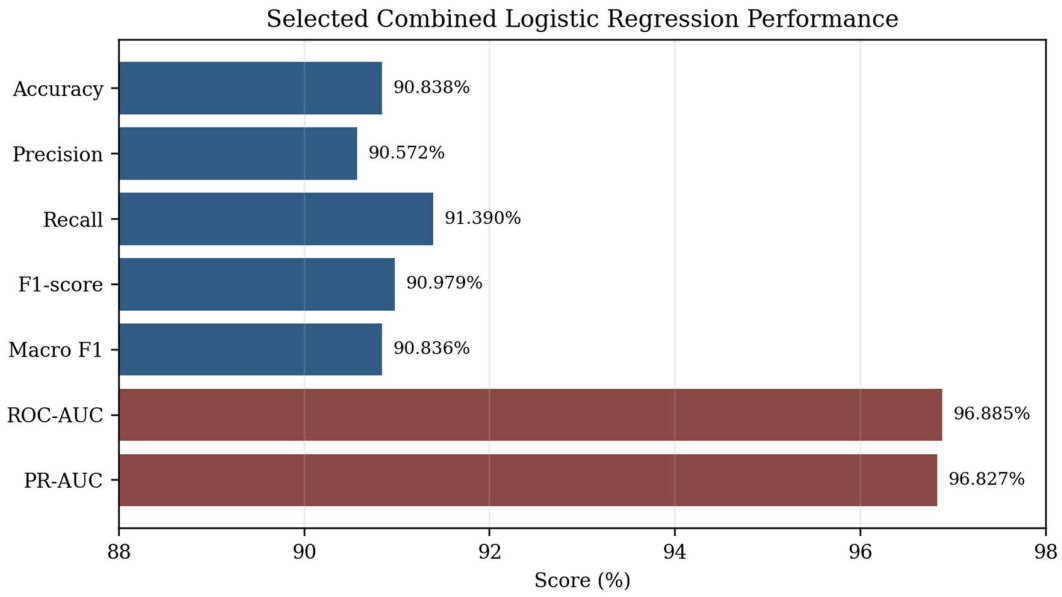


Fig. 5.2 Exact selected-model performance metrics

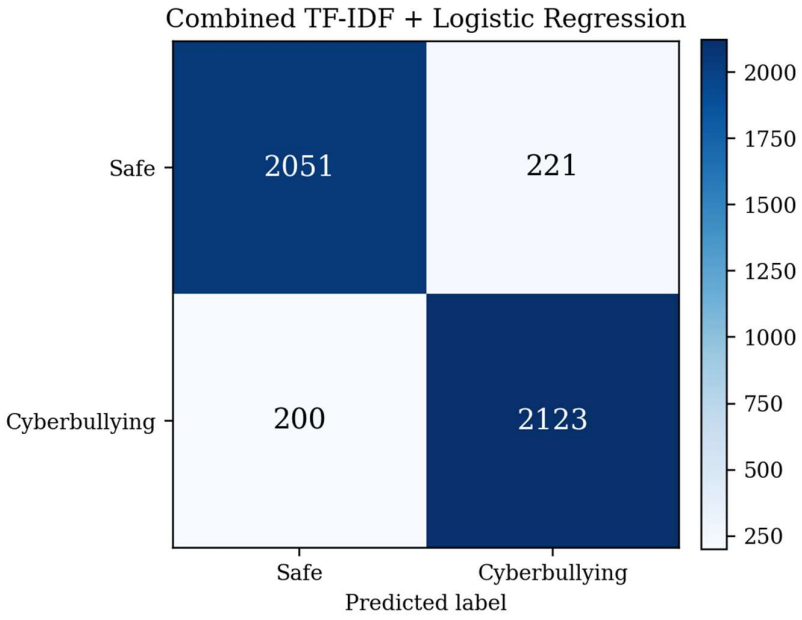


Fig. 5.3 Confusion matrix of the selected model

Fig. 5.3 spell out The confusion matrix contains 2,051 true negatives, 221 false positives, 200 false negatives, and 2,123 true positives. False positives may cause unnecessary moderation review or removal, while false negatives may leave harmful content visible. Threshold adjustment can move this trade-off, but threshold selection should be based on validation data and the operational cost of each error.

5.6 Cross-Validation and Generalization

The combined Logistic Regression configuration obtained 90.12% validation macro F1 in five-fold cross-validation. Training macro F1 was 99.57%, producing a 9.45-point cross-validation gap. Its training F1 on the selected split was 99.43%, compared with 90.98% test F1 for the cyberbullying class, an 8.45-point gap. These gaps indicate moderate overfitting despite strong held-out performance.

5.6.1 Error Analysis and Moderation Cost

Table 5.5 demonstrate The confusion matrix records 221 false positives and 200 false negatives. A false positive is a Safe comment predicted as Cyberbullying. In the proposed system, this error may create unnecessary review, delay a legitimate comment, or cause wrongful deletion if an unsafe automatic policy is used. A false negative is a cyberbullying comment predicted as Safe. It may remain visible, receive further engagement, and continue affecting the target until a user report or random review identifies it.

The observed false-negative count is lower than the false-positive count, and positive-class recall is higher than precision. This operating point is reasonable for an assistive moderation queue because missing harm is costly, but it does not automatically justify deletion. The safest interpretation is that high-scoring results deserve faster attention. Whether a particular score should cause an automatic action depends on independent validation, page policy, language subgroup performance, and the reversibility of the action.

Table 5.5 Confusion-matrix outcomes and moderation implications

Observed outcome	Count	Operational consequence
True Negative	2,051	Safe comment passes without review burden
False Positive	221	Legitimate content may be flagged
False Negative	200	Harmful content may remain visible
True Positive	2,123	Harmful content is detected

5.7 System-Level Validation

System-level validation was conducted to determine whether the main components of the proposed system operate together as expected. Unlike model evaluation, which focuses on predictive performance, system-level validation examines the complete workflow, including manual prediction, Facebook event processing, duplicate prevention, error handling, multilingual input, and moderation actions. **Table 5.6** summarizes the tested scenarios and their outcomes.

The manual prediction test confirmed that a valid comment produces a probability score and a binary Safe or Cyberbullying verdict. The Facebook event test also confirmed that a valid event can be received, classified, stored, and displayed with its prediction result.

The duplicate-event test showed that an event with an existing identifier does not create an additional record. Instead, the stored record is updated. This reduces repeated processing and duplicate entries on the moderation dashboard.

Table 5.6 System-level validation record

Test scenario	Expected behavior	Status
Valid manual comment	Return score and binary verdict	Pass
Valid Facebook event	Store event and display prediction	Pass
Repeated event identifier	Do not create duplicate moderation action	Pass
Missing model artifact	Return explicit service error; never default to Safe	Pass
Missing deletion authorization	Reject action without changing Facebook content	Pass
Bengali/Banglish /English input	Preserve script and pass through common feature pipeline	Pass

Failure handling was tested by making the required model artifact unavailable. In this situation, the API returned an explicit service error rather than automatically classifying the comment as Safe. Similarly, when the Facebook page token required for deletion was missing, the system rejected the deletion request without changing Facebook content.

The multilingual input test confirmed that Bengali, Banglish, and English comments are preserved and processed through the same preprocessing, word- and character-level TF-IDF, Logistic Regression, and rule-based pipeline. Overall, the results demonstrate that the major components of the prototype work together correctly under the tested conditions.

5.8 Real-World Evaluation

Table 5.7 describe the real-world evaluation was conducted on 100 comments, including 49 safe comments and 51 cyberbullying comments. The system achieved an overall accuracy of 84%, correctly detecting 43 safe comments and 41 cyberbullying comments.

Table 5.7 Real-world evaluation

Measure	Result
Number of real-world comments	100
Safe comments	49
Cyberbullying comments	51
Accuracy	84%
Detected safe comments	43
Detected cyberbullying comments	41

5.9 Discussion

The experiments support three conclusions. First, character-level information is important for the active dataset. Second, combining word and character evidence produces the strongest overall result. Third, Logistic Regression and Linear SVC are statistically close in observed point estimates, but Logistic Regression better fits this application because of its recall, macro F1, and native probability output. These results align with prior studies showing that TF-IDF with tuned linear models remains a strong baseline for harmful-text detection [6], [9].

The model still cannot fully recognize sarcasm, implicit hostility, quoted abuse, reclaimed terms, or bullying that emerges only across a conversation. Comment-level labels also

collapse the difference between an isolated insult and repeated targeted behavior. Furthermore, language changes over time and across communities. Continued monitoring, careful human review, error analysis, and periodic retraining are necessary for responsible use.

The central research outcome is the integration of detection into a moderation loop. A 90.84% macro F1 score is useful only when the software can receive a comment, expose the decision promptly, and support a proportionate action. The architecture connects these functions, but operational latency and live multilingual performance remain unmeasured. For that reason, the system is accurately described as a near-real-time capable prototype rather than a fully validated real-time production service.

The system can help potential victims by reducing exposure time and by giving page administrators a consistent mechanism for noticing harmful language. It can also reduce moderator workload by ranking the review queue. It cannot eliminate the psychological consequences documented in prior research [19], determine the intent of every author, or replace support from families, schools, clinicians, platforms, and legal authorities. Its contribution is a technical layer within a wider safeguarding response.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This thesis presented a complete binary cyberbullying-detection and Facebook-comment moderation prototype. The system processes a text comment, performs consistent normalization, combines word and character TF-IDF features, and uses Logistic Regression to return either Cyberbullying or Safe. The model is exposed through FastAPI and a static browser interface, while optional Facebook integration supports event collection, filtering, display, and authorized deletion.

The experimental results show that a classical linear text-classification pipeline can achieve strong performance on the project dataset. The dataset contains 22,975 comments and is close to balanced between Non-Cyberbullying and Cyberbullying. The best documented combined TF-IDF Logistic Regression configuration uses 30,000 word features and 30,000 character features with word n-grams of (1,2), character n-grams of (3,5), and C=10.

On the held-out test set, the selected combined Logistic Regression model achieved 90.84% accuracy, 90.84% macro F1, and 90.98% F1 for Cyberbullying. It slightly exceeded the combined Linear SVC result and produced fewer false negatives. The findings justify Logistic Regression as one of the effective and suitable classical algorithms for this detection setting. The work also demonstrates why complementary character features matter for noisy online language.

The application component demonstrates how the model can be integrated into a practical system. A browser can send a comment to the FastAPI service, and eligible Facebook comments can be retrieved through the monitoring component and passed through the same prediction pipeline. Because the Facebook workflow uses periodic polling, the system is correctly described as near-real-time rather than instantaneous real-time monitoring.

Overall, the project demonstrates that a lightweight NLP pipeline can serve as a practical first-stage moderation assistant. At the same time, the train-test gap, language variation, annotation limitations, and platform constraints show that the system should be viewed as an academic prototype and foundation for further research.

6.2 Future Work

Larger and More Diverse Datasets:
Future research should expand the dataset with more Bengali, English, and Banglish comments collected under appropriate ethical and platform-compliance procedures. More diverse sources would help reduce overfitting to the vocabulary and writing style of the current dataset.

Bangla and Banglish Normalization:
A dedicated normalization layer could handle transliteration, spelling variants, Unicode normalization, common abbreviations, and code-switching. This may improve performance on comments that differ substantially from the training vocabulary.

Transformer-Based Models:
Future experiments can compare the classical pipeline with XLM-RoBERTa, Bangla-specific transformer models, and code-mixed language models. Recent Bengali cyberbullying studies show that contextual transformer representations are promising in low-resource settings [26]. A hybrid system could use the classical model for fast screening and a transformer for uncertain cases.

Threshold Calibration and Human Review:
Instead of using a single default threshold, the system could calibrate probabilities on a validation set and define different review levels. High-confidence harmful comments could be prioritized, while uncertain comments could be routed to a human moderator. This would make the system more suitable for practical moderation.

Explainability and Auditability:
Future versions could display the probability score, important contributing terms, and moderation history to authorized reviewers. Such information would help moderators understand why a comment was flagged and would support auditing of false positives and false negatives.

Production-Scale Storage and Security:
A relational or event-oriented database could replace JSON storage for larger deployments. Role-based access, encryption, audit logs, secure secret management, monitoring, and automated backup would be required before production use.

REFERENCES

[1] P. K. Smith, J. Mahdavi, M. Carvalho, S. Fisher, S. Russell, and N. Tippett, "Cyberbullying: Its nature and impact in secondary school pupils," Journal of Child Psychology and Psychiatry, vol. 49, no. 4, pp. 376–385, 2008, doi: 10.1111/j.1469-7610.2007.01846.x.

[2] R. M. Kowalski, G. W. Giumetti, A. N. Schroeder, and M. R. Lattanner, "Bullying in the digital age: A critical review and meta-analysis of cyberbullying research among youth," Psychological Bulletin, vol. 140, no. 4, pp. 1073–1137, 2014, doi: 10.1037/a0035618.

[3] S. Hinduja and J. W. Patchin, "Bullying, cyberbullying, and suicide," Archives of Suicide Research, vol. 14, no. 3, pp. 206–221, 2010, doi: 10.1080/13811118.2010.494133.

[4] H. Rosa et al., "Automatic cyberbullying detection: A systematic review," Computers in Human Behavior, vol. 93, pp. 333–345, 2019, doi: 10.1016/j.chb.2018.12.021.

[5] S. Salawu, Y. He, and J. Lumsden, "Approaches to automated detection of cyberbullying: A survey," IEEE Transactions on Affective Computing, vol. 11, no. 1, pp. 3–24, 2020, doi: 10.1109/TAFFC.2017.2721347.

[6] S. Rahman, K. H. Talukder, and S. K. Mithila, "An empirical study to detect cyberbullying with TF-IDF and machine learning algorithms," in Proc. 2021 International Conference on Electronics, Communications and Information Technology (ICECIT), 2021, doi: 10.1109/ICECIT54077.2021.9641251.

[7] J. Risch and R. Krestel, "Aggression identification using deep learning and data augmentation," in Proc. First Workshop on Trolling, Aggression and Cyberbullying, 2018, pp. 150–158.

[8] A. Gaydhani, V. Doma, S. Kendre, and L. Bhagwat, "Detecting hate speech and offensive language on Twitter using machine learning: An n-gram and TFIDF based approach," arXiv:1809.08651, 2018.

[9] K. S. Alam, S. Bhowmik, and P. R. K. Prosun, "Cyberbullying detection: An ensemble based machine learning approach," in Proc. 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021, pp. 710–715, doi: 10.1109/ICICV50876.2021.9388499.

[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988, doi: 10.1016/0306-4573(88)90021-0.

[11] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in Proc. 11th International AAAI Conference on Web and Social Media, 2017, pp. 512–515.

[12] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter," in Proc. NAACL Student Research Workshop, 2016, pp. 88–93, doi: 10.18653/v1/N16-2013.

[13] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in Machine Learning: ECML-98, Berlin, Germany: Springer, 1998, pp. 137–142, doi: 10.1007/BFb0026683.

[14] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," Information Processing & Management, vol. 45, no. 4, pp. 427–437, 2009, doi: 10.1016/j.ipm.2009.03.002.

[15] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[16] S. Ramírez, "FastAPI documentation," FastAPI. [Online]. Available: https://fastapi.tiangolo.com/. [Accessed: Sep. 5, 2026].
[17] UNICEF, "More than a third of young people in 30 countries report being a victim of online bullying," Sep. 4, 2019. [Online]. Available: https://www.unicef.org/eca/press-releases/unicef-poll-more-third-young-people-30-countries-report-being-victim-online-bullying. [Accessed: Sep. 5, 2026].
[18] UNICEF Bangladesh, "Safer Internet Day: UNICEF calls for concerted action to prevent bullying and harassment for the 32 per cent of children online in Bangladesh," Feb. 5, 2019. [Online]. Available: https://www.unicef.org/bangladesh/en/press-releases/unicef-calls-concerted-action-prevent-bullying-and-harassment-32-cent-children. [Accessed: Sep. 5, 2026].
[19] M. M. Molero, A. Martos, A. B. Barragán, M. C. Pérez-Fuentes, and J. J. Gázquez, "Anxiety and depression from cybervictimization in adolescents: A meta-analysis and meta-regression study," The European Journal of Psychology Applied to Legal Context, vol. 14, no. 1, pp. 42–50, 2022, doi: 10.5093/ejpalc2022a5.
[20] T. T. Aurpa, R. Sadik, and M. S. Ahmed, "Abusive Bangla comments detection on Facebook using transformer-based deep learning models," Social Network Analysis and Mining, vol. 12, art. no. 24, 2022, doi: 10.1007/s13278-021-00852-x.
[21] M. N. Hoque and M. H. Seddiqui, "Detecting cyberbullying text using the approaches with machine learning models for the low-resource Bengali language," IAES International Journal of Artificial Intelligence, vol. 13, no. 1, pp. 358–367, 2024, doi: 10.11591/ijai.v13.i1.pp358-367.
[22] R. Kumar, B. Lahiri, and A. K. Ojha, "Aggressive and offensive language identification in Hindi, Bangla, and English: A comparative study," SN Computer Science, vol. 2, pp. 1–20, 2021, doi: 10.1007/s42979-020-00414-6.
[23] Dinakar, K., Reichart, R., & Lieberman, H. (2021). Modeling the Detection of Textual Cyberbullying. Proceedings of the International AAAI Conference on Weband Social Media, 5(3), 11–17. https://doi.org/10.1609/icwsm.v5i3.14209
[24] Hosseinmardi, H., Mattson, S.A., Ibn Rafiq, R., Han, R., Lv, Q., Mishra, S. (2015). Analyzing Labeled Cyberbullying Incidents on the Instagram Social Network. In: Liu, TY., Scollon, C., Zhu, W. (eds) Social Informatics. SocInfo 2015. Lecture Notes in Computer Science(), vol 9471. Springer, Cham. https://doi.org/10.1007/978-3-319-27433-1_4
[25] C. Van Hee et al., "Automatic detection of cyberbullying in social media text," PLOS ONE, vol. 13, no. 10, Art. no. e0203794, 2018, doi: 10.1371/journal.pone.0203794.
[26] A. C. Roy, T. Mahmud, and T. Abrar, "A multi-class cyberbullying classification on image and text in code-mixed Bangla-English social media content," Natural Language Processing Journal, vol. 13, Art. no. 100191, 2025, doi: 10.1016/j.nlp.2025.100191.