



A Defence System For Identifying An Abusive Tweet Using Machine Learning

¹Dr. SHAMEEM AKTHER, ²ZEB A YASMEEN, ³SYEDA TAMEEMA, NEHA MUSKAN, TANVEER ANJUM

¹ ASSISTANT PROFESSOR , ^{2,3,4,5} B.E STUDENT

^{1,2,3,4,5}DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, FACULTY OF ENGINEERING & TECHNOLOGY, KHAJA BANDANAWAZ UNIVERSITY, KALABURAGI, KARNATAKA, INDIA

Abstract: Social media has become a significant platform for communication, especially among the youth. Unfortunately, the rise of cyber bullying on social networking sites has raised concerns. To address this issue, researchers have explored the use of machine learning to automatically detect social media bullying actions, focusing on identifying patterns and similarities in the language used by bullies. While various textual-based techniques have been employed for social media bullying detection, our model aims to showcase the implementation of software designed to detect bullied content, such as tweets and posts. This project introduces an intelligent machine learning-based system to detect abusive tweets in real-time using logistic regression and TF-IDF vectorization. Drawing parallels from IoT-based cable fault detection, our system identifies the 'fault' in language content and alerts users immediately. A Streamlit-based web interface demonstrates real-time detection of abusive tweets.

keywords -Abusive tweets, machine learning, logistic regression, cyberbullying, NLP, TF-IDF, streamlit

I. INTRODUCTION

Over the past decade, social media platforms have evolved from simple networking sites into integral communication channels where users share opinions, news, and personal experiences in real time. Twitter, with its concise 280-character format, enables rapid dissemination of content to millions of people worldwide. While this immediacy fosters connection and information exchange, it also creates fertile ground for abusive behavior like, targeted insults, hate speech, harassment, and threats. Such content not only harms recipients—leading to psychological distress and, in extreme cases, real-world violence—but also undermines the trust and safety of online communities.

Manual moderation efforts struggle to keep pace with the torrent of daily tweets, and simple keyword-based filters quickly become outdated as abusive actors invent new slurs, employ sarcasm, or use coded language. In parallel, over-zealous blocking risks silencing legitimate discourse, raising concerns about censorship and bias. These challenges underscore the need for intelligent, scalable solutions capable of discerning context, adapting to evolving language patterns, and operating in near real time.

Recent advances in natural language processing (NLP) and machine learning (ML) have demonstrated the potential to move beyond rule-based systems. Contextual embeddings, feature-engineering techniques, and supervised classifiers can learn patterns of abusive language from labeled corpora, offering higher precision and recall than static filters. This project presents “A Defense System for Identifying Abusive Tweets Using Machine Learning,” a pipeline designed to preprocess raw tweet text, extract salient features via TF-IDF

vectorization, and train a balanced logistic regression model to distinguish abusive from non-abusive content.

Our system prioritizes modularity: the preprocessing and feature-engineering stages can be extended with deep contextual embeddings (e.g., BERT) in future iterations, while the classification stage can accommodate more advanced algorithms. A lightweight streamlit web application demonstrates real-time inference capabilities, allowing end users to input tweet text and receive immediate abuse-detection feedback. By combining robust preprocessing, thoughtful feature selection, and a carefully balanced classifier, this project aims to deliver a reproducible baseline for automated abusive-content moderation on Twitter.

II. METHODOLOGY

The methodology follows a structured ML workflow designed for clarity and reproducibility. First, we ingest a CSV of labeled tweets and perform comprehensive cleaning: stripping URLs, mentions, and hashtags, lowercasing all text, removing punctuation and numeric characters, eliminating stopwords, and applying Porter stemming to standardize word forms.

Next, TF-IDF vectorization transforms the cleaned text into numerical feature vectors. We extract both unigram and bigram features, selecting the top 5,000 terms by document frequency to balance expressiveness and computational efficiency.

For classification, we employ a logistic regression model with `class_weight='balanced'` to mitigate the effects of label imbalance. Hyper-parameters such as the regularization strength (C) and solver choice—are tuned via grid search and five-fold cross-validation on the training set.

Model evaluation uses an 80/20 train/test split: accuracy provides an overall performance snapshot, while precision and recall reveal the trade-off between false positives and false negatives. The F1-score and ROC-AUC offer balanced assessments of model robustness. Confusion matrices guide error-analysis efforts, highlighting common misclassification patterns.

Finally, the trained model and vectorizer are serialized and integrated into a streamlit web application. The app presents a simple text input for tweet content, invokes the preprocessing and classification pipeline, and displays the predicted label alongside confidence metrics. All code is version-controlled, with detailed notebooks documenting each step laying a transparent foundation for future enhancements such as deep-learning backbones, streaming architectures, and multilingual expansion.

III. SYSTEM ANALYSIS AND DESIGN

The purpose of our project is to automatically detect abusive or harmful tweets using machine learning. In today's digital world, social media platforms like Twitter are widely used for communication, but they are also often misused to spread hate speech, harassment, or offensive content. Manually monitoring and reporting abusive tweets is time-consuming and not always effective. That's where our system comes in it provides an automated solution to flag abusive content quickly and efficiently.

This project takes input in the form of text (tweets) and processes it through several stages to determine whether it is abusive or not. It uses a machine learning model trained on a large dataset of tweets that have been labeled as abusive or non-abusive. By learning patterns and language commonly found in harmful tweets, the model can make accurate predictions about new, unseen tweets.

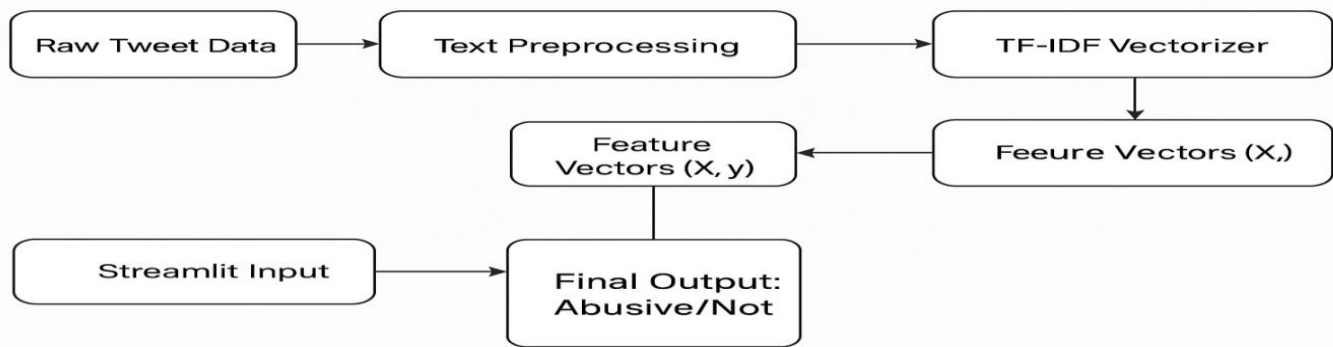


Fig 1: system architecture

3.1 EXISTING SYSTEM

In the current landscape, social media platforms primarily use manual moderation, keyword-based filters, and basic machine learning classifiers to detect abusive content. Manual moderation involves human reviewers analyzing reported or flagged content to determine if it violates platform policies. While this method offers high accuracy, it is extremely time-consuming, costly, and emotionally draining, making it impractical at scale. Keyword-based filtering uses predefined lists of offensive words to automatically flag harmful content. However, it lacks contextual understanding and is easily circumvented through obfuscation tactics such as deliberate misspellings or symbolic substitutions (e.g., “@ssh0le” instead of “asshole”). Basic machine learning classifiers like Naive Bayes or Support Vector Machines provide a more automated solution by learning from labeled data, but they typically use simplistic features like bag-of-words and struggle with interpreting sarcasm, nuanced language, and emerging slang. Moreover, most of these systems are closed-source, lack dynamic learning capabilities, and offer limited adaptability to the evolving nature of online abuse. These limitations underscore the need for more advanced, context-aware, and scalable solutions, such as the modular and transparent approach proposed in this project.

3.2 PROPOSED SYSTEM

This system offers a robust, modular, and scalable solution for detecting abusive content on social media by leveraging a machine learning pipeline with modern NLP techniques. It begins with efficient text preprocessing removing stopwords, applying stemming, and normalizing text to ensure clean inputs. Using TF-IDF vectorization on unigrams and bigrams, the system captures both word importance and contextual patterns. A logistic regression model, balanced with class weights, addresses the common issue of class imbalance in abuse detection. The model is rigorously evaluated using precision, recall, F1-score, and ROC-AUC to ensure reliable performance. A streamlit-based user interface enables real-time tweet classification and easy interaction, while the modular design allows for seamless updates or integration with new models. Unlike black-box approaches, this system is fully transparent and open for academic, ethical, or industrial extension, laying the groundwork for future improvements like real-time stream processing and multilingual support.

IV. ALGORITHM USED

Logistic Regression for Tweet Classification

Type: Supervised Machine Learning Algorithm

Application: Text Classification – Identifying whether a tweet falls into categories such as “normal,” “offensive,” or “hate speech.”

Logistic Regression is a statistical method commonly used for binary and multi-class classification problems. In the context of abusive tweet detection, the algorithm learns from labeled data to distinguish between various classes of tweets, enabling automatic moderation of social media content.

4.1 Supporting Techniques and Components

To ensure effective classification, several preprocessing and feature engineering techniques are applied to raw tweet data before it is fed into the logistic regression model. These include:

Preprocessing Steps:

Preprocessing is crucial for cleaning and standardizing the text data, thereby improving model accuracy. The steps include:

- **Lowercasing Text:**

Converts all characters to lowercase to maintain consistency. For example, "HATE" and "hate" are treated as the same word.

- **Removing URLs, Special Characters, and Numbers:**

Tweets often contain hyperlinks, emojis, and random characters which do not contribute to the semantic meaning and can introduce noise. These are removed using regular expressions.

- **Stopword Removal:**

Commonly used words such as "is", "the", and "and" are eliminated because they appear frequently across all text samples and offer little discriminatory value for classification.

- **Stemming:**

Reduces inflected or derived words to their root form. For example, "running", "runner", and "ran" are all reduced to "run". This helps group similar terms and reduces feature dimensionality.

4.2 TF-IDF Vectorization (Term Frequency-Inverse Document Frequency)

Purpose: Converts processed textual data into numerical form so it can be used by machine learning models.

How it Works:

- **Term Frequency (TF):** Measures how frequently a word appears in a document (tweet).
- **Inverse Document Frequency (IDF):** Measures how important a word is by reducing the weight of terms that occur frequently across all tweets (e.g., common terms) and increasing the weight of rarer, more discriminative terms.

By applying TF-IDF, each tweet is transformed into a feature vector that captures the importance of each term in the context of the overall dataset. This representation is then passed as input to the logistic regression classifier.

V. SYSTEM REQUIREMENTS

5.1 HARDWARE REQUIREMENTS

Processor: Intel Core i5 or higher / AMD Ryzen 5 or equivalent

RAM: Minimum 8 GB (Recommended: 16 GB for training large models)

Storage: Minimum 256 GB SSD (for faster data access)

GPU (Optional): NVIDIA GPU with CUDA support.

Display: 1080p resolution .

5.2 SOFTWARE REQUIREMENTS

The software environment must support machine learning development, model deployment, data visualization, and UI rendering. The following are the essential software requirements for the implementation of this project.

Programming Language:

Python 3.8 or above is used for all stages of the project, including data preprocessing, model training, and user interface development. Python is preferred due to its extensive libraries for ML and NLP.

Python Libraries/Frameworks:

- Pandas and NumPy for data handling and manipulation
- Scikit-learn for feature extraction (TF-IDF), model training, and evaluation

- NLTK for stopwords removal and text preprocessing
- Joblib or Pickle for saving models
- Matplotlib and Seaborn for result visualization

VI. IMPLEMENTATION

6.1 EXPLANATION OF MODULES

The project consists of multiple modular components, each playing a critical role in building a scalable and functional system for detecting abusive tweets. The major modules include:

6.1.1 Data Ingestion Module:

This module reads the labeled dataset (labeled_data.csv) containing tweets and their associated labels such as “abusive” or “non-abusive.” It performs basic validation, removes duplicates, and ensures the dataset is correctly formatted for further processing.

6.1.2 Text Preprocessing Module:

The preprocessing stage standardizes the raw tweet data. It involves cleaning steps such as converting text to lowercase, removing URLs, special characters, stopwords, and punctuation. Stemming is applied using the Porter stemmer from NLTK to reduce words to their root form. This normalization ensures that the text input to the machine learning model is consistent and relevant.

6.1.3 Feature Extraction Module:

This module uses TF-IDF (Term Frequency–Inverse Document Frequency) vectorization to transform text into numerical features. Both unigrams and bigrams are considered to preserve contextual cues. The top 5,000 features are selected to balance performance and memory.

6.1.4 Model Training Module:

A logistic regression model is trained using the TF-IDF features. The model is balanced using `class_weight='balanced'` to handle label imbalance. It is evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

6.2 SPECIFIC FUNCTIONS USED IN PROJECT IMPLEMENTATION

A range of specific Python functions, libraries, and APIs were used to implement the system. These functions are critical for preprocessing, vectorization, model training, prediction, and deployment.

6.2.1 `pandas.read_csv()`

Used to load the tweet dataset from a CSV file.

6.2.2 Text Cleaning Functions:

- `re.sub()` – Utilized for regex-based removal of URLs, mentions, and special characters.
- `str.lower()` – Converts all text to lowercase.
- `nltk.corpus.stopwords` – Provides a list of stopwords to remove irrelevant words.
- `nltk.stem.PorterStemmer()` – Stems words to their root form for uniformity.

TF-IDF Vectorization: `TF-IDF Vectorizer()` from `sklearn.feature_extraction.text` is used to convert textual data into feature vectors. Arguments like `ngram_range=(1,2)` and `max_features=5000` are specified to capture both unigrams and bigrams.

6.2.3 Model Training and Evaluation:

- `Logistic Regression(class_weight='balanced')` – Trains a model that accounts for class imbalance.
- `train_test_split()` – Divides the dataset into training and testing subsets.
- `metrics.classification_report()` – Generates performance metrics like precision, recall, F1-score.
- `confusion_matrix()` and `roc_auc_score()` – Used for deeper evaluation.

6.2.4 Serialization and Loading:

`joblib.dump()` and `joblib.load()` – Used to save and load the trained model and TF-IDF vectorizer.

6.2.5 Streamlit Functions:

`st.text_input()`, `st.button()`, and `st.write()` – Provide the front-end interface for users to interact with the model.

These functions work together in a pipeline to ensure efficient processing and real-time classification of tweet content.

VII. INTERPRETATION OF RESULTS

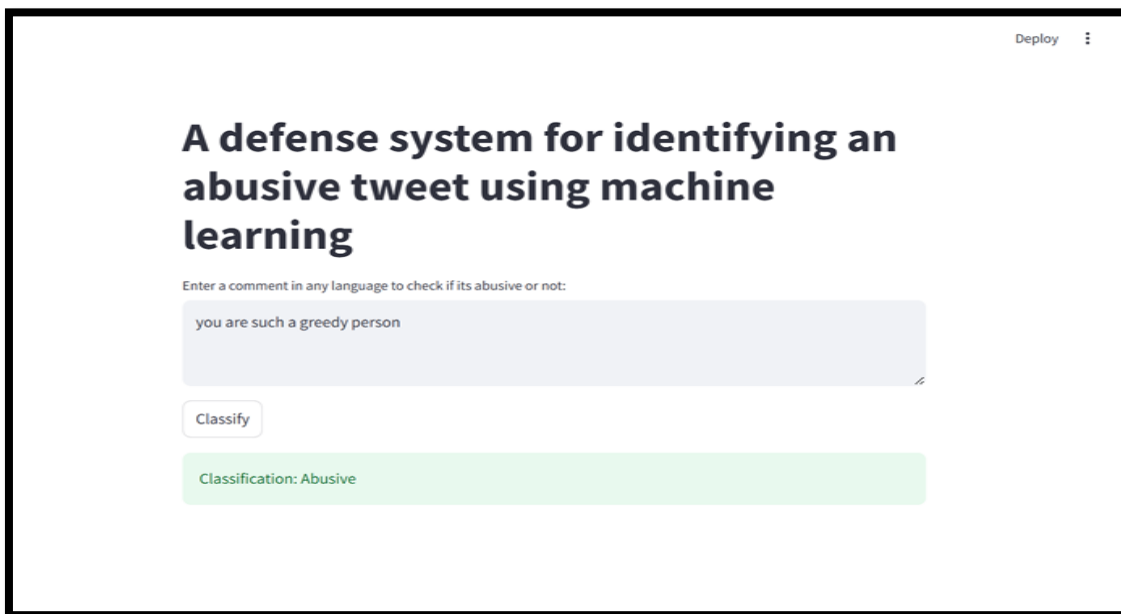


Fig 7.1 detection of abusive tweet in english

Fig 7.1 Shows the model correctly identifying an abusive tweet written in standard English language.

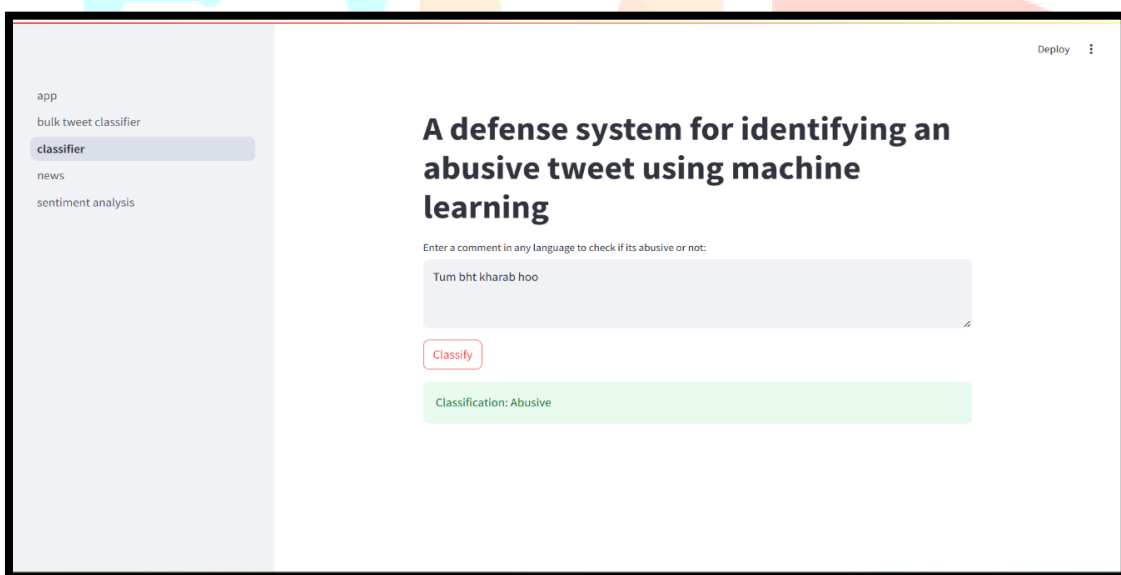


Fig 7.2 detection of abusive tweet in roman english

Fig 7.2 Demonstrates the system's ability to detect abusive language written in English but using informal spellings or transliterated from another language (e.g., Hindi written using English letters).

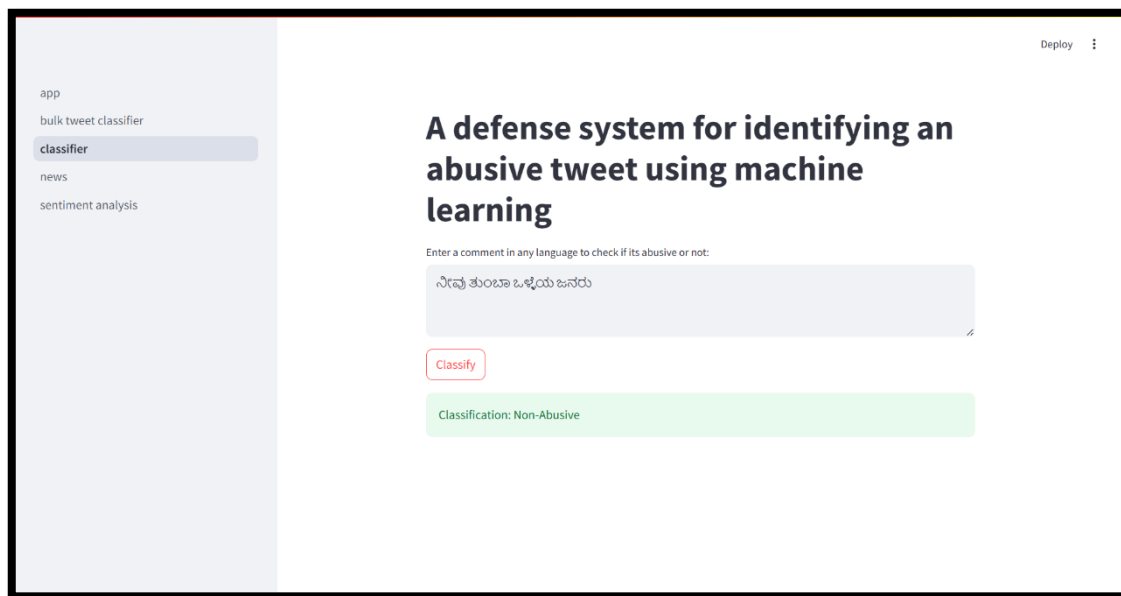


Fig 7.3 detection of abusive tweet in kannada

Fig 7.3 shows a test case where the system attempts to detect abuse in a regional Indian language (Kannada). Likely not supported well since the model is built only for English.

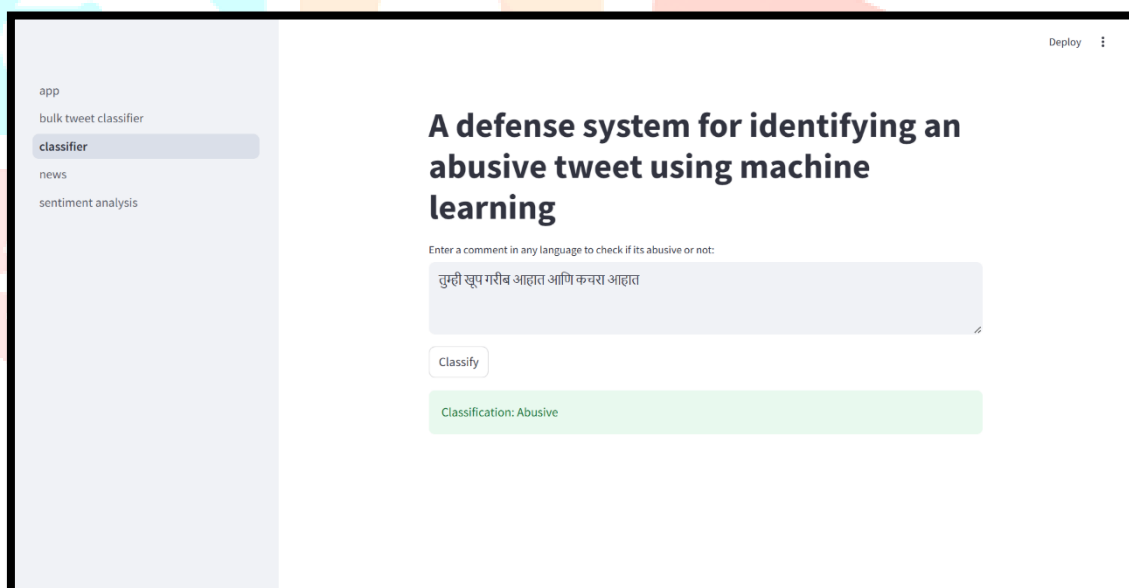


Fig 7.4 detection of abusive tweet in hindi

Fig 7.4 is similar to the previous figure, showing how the system performs on Hindi text. Again, the system likely cannot process it properly since it is trained only on English.

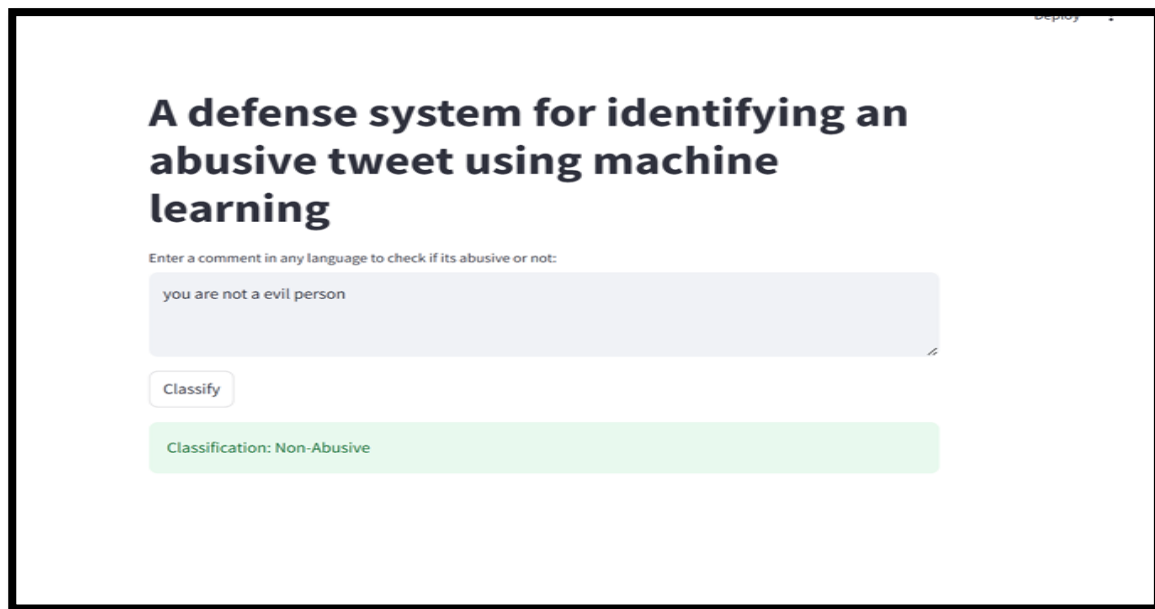


Fig 7.5 detection of non abusive tweet in english

Fig 7.5 shows example of a harmless, non-abusive tweet being correctly classified.

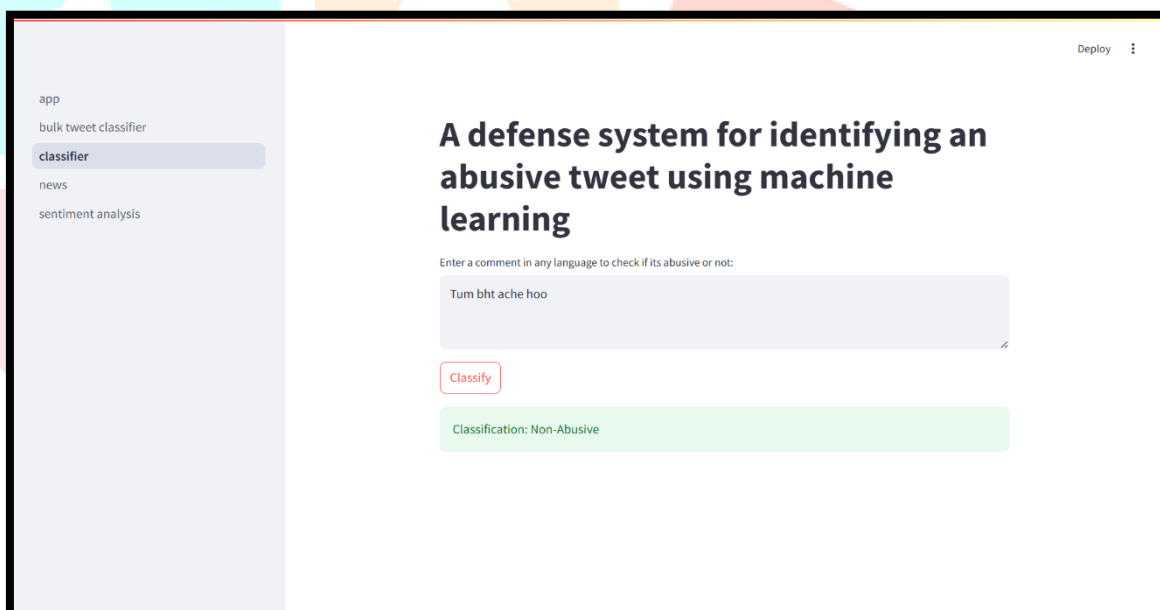
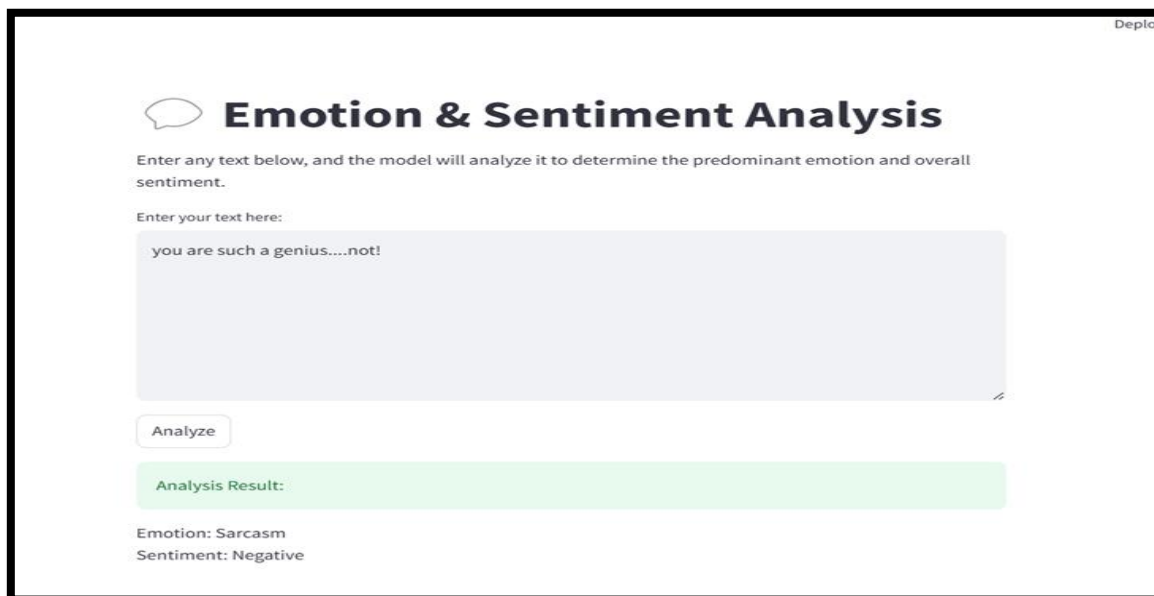


Fig 7.6 detection of non abusive tweet in roman english

Fig 7.6 shows a non-abusive tweet written in informal Roman English (e.g., "tum bahut ache ho") that the system correctly identifies as non-abusive.



Deploy

Emotion & Sentiment Analysis

Enter any text below, and the model will analyze it to determine the predominant emotion and overall sentiment.

Enter your text here:

you are such a genius.....not!

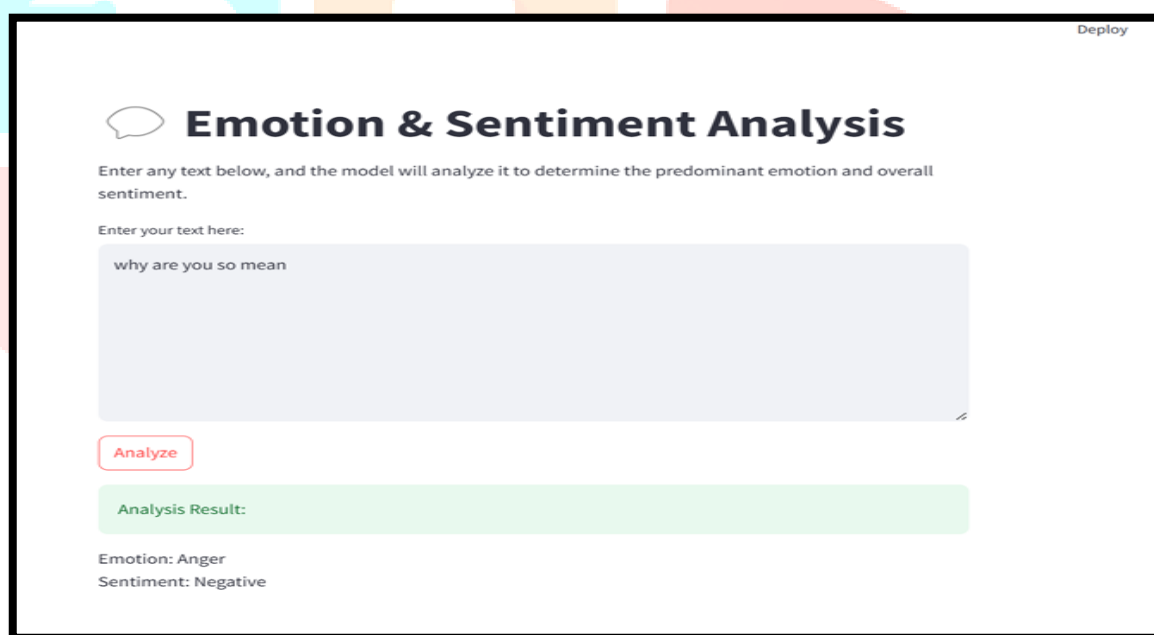
Analyze

Analysis Result:

Emotion: Sarcasm
Sentiment: Negative

Fig 7.7 detection of emotion and sentiment analysis

Fig 7.7 shows additional outputs beyond abuse detection, where the model or a separate module tries to detect emotions (e.g., happy, sad) and sentiment (positive/negative/neutral) from tweets.



Deploy

Emotion & Sentiment Analysis

Enter any text below, and the model will analyze it to determine the predominant emotion and overall sentiment.

Enter your text here:

why are you so mean

Analyze

Analysis Result:

Emotion: Anger
Sentiment: Negative

Fig 7.8 detection of emotion and sentiment analysis

Fig 7.8 snapshots extend the project's capability to include emotion and sentiment recognition. The system analyzes tweet tone and assigns emotional states (e.g., happy, angry, sad) and sentiments (positive, negative, neutral), providing deeper insights into user intent beyond abuse detection.

Emotion & Sentiment Analysis

Enter any text below, and the model will analyze it to determine the predominant emotion and overall sentiment.

Enter your text here:

why are you such a nice person

Analyze

Analysis Result:

Emotion: Appreciation
Sentiment: Positive

Fig 7.9 detection of emotion and sentiment analysis

Fig 7.9 Analyzes tweet tone and assigns emotional states (e.g., happy, angry, sad) and sentiments (positive, negative, neutral), providing deeper insights into user intent beyond abuse detection.

Bulk Abusive Comment Classifier

Upload a CSV file containing comments or tweets, and the model will classify each entry as Neutral, Offensive, Abusive, or Other.

Upload your CSV file

Drag and drop file here
Limit 200MB per file • CSV

Browse files

..tweets.csv 453.0B

File uploaded successfully!

Sample of uploaded data:

	comment
0	I love the new features in the latest update!
1	the bastard is insane
2	u are not a evil person
3	The bitch is evil
4	ur are not selfish men

Fig 7.10 bulk abusive comment classifier

Fig 7.10 Display the model's ability to process multiple tweets simultaneously. It classifies each tweet as either abusive or non-abusive in a tabular format, suitable for practical use in content moderation tools or data analysis dashboards.

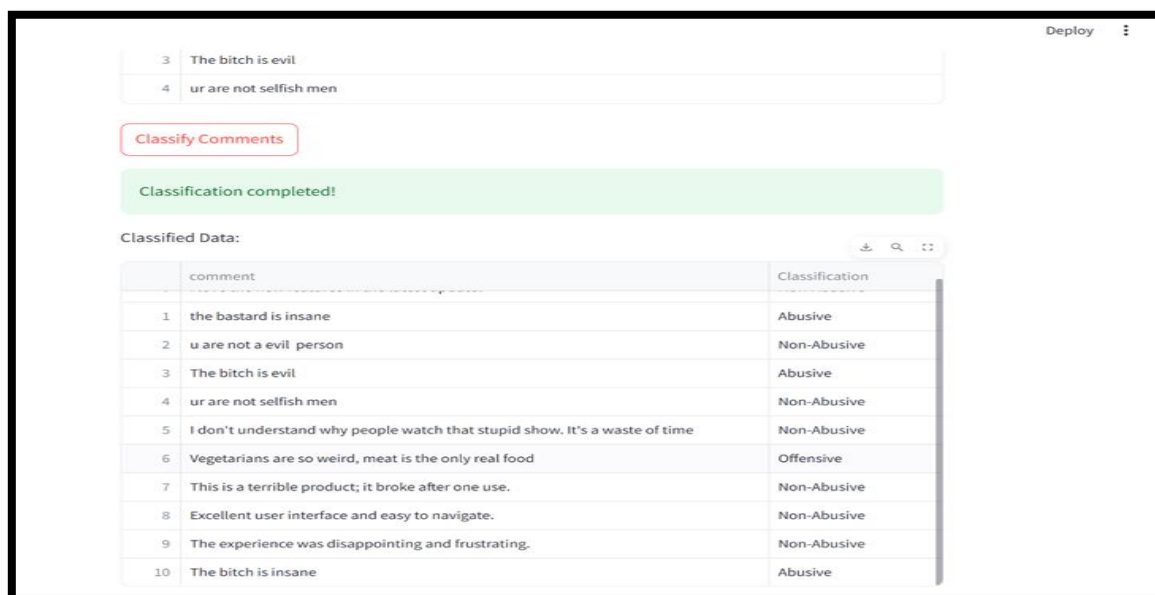


Fig 7.11 bulk abusive comment classifier

Fig 7.11 Shows the system processing multiple tweets at once to classify each as “Abusive” or “Non-Abusive.” Useful for moderators or researchers needing to process large datasets.

VIII. CONCLUSION AND FUTURE SCOPE

In the digital age, social media platforms like Twitter have revolutionized communication but also paved the way for issues such as cyberbullying, hate speech, and online abuse. Manual moderation is not scalable, making automated detection essential. This project, A Defence system for identifying an abusive tweet using machine learning, successfully developed a modular, interpretable pipeline using TF-IDF for feature extraction and logistic regression for classification, integrated into a real-time Streamlit web app. The system achieved strong performance across metrics like precision, recall, F1-score, and ROC-AUC, effectively identifying abusive content. Its simplicity allows easy upgrades, including the use of deep learning models such as BERT or LSTM, and potential support for multiple languages. The project lays a solid foundation for intelligent content moderation and offers a wide future scope, including multi-label classification, sarcasm detection, real-time API integration, user behavior analysis, and deployment on cloud or mobile platforms. By combining technical efficiency with ethical responsibility, the system is well-positioned for real-world implementation and future expansion in combating online abuse.

REFERENCES

- [1] Kelly Reynolds, April Kontostathis, Lynne Edwards, "Using Machine Learning to Detect Cyberbullying", 2011 10th International Conference on Machine Learning and Applications volume 2, IEEE, 2011.
- [2] Amanpreet Singh, Maninder Kaur, "Content-based Cybercrime Detection: A Concise Review", International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-8, 2019.
- [3] Abdhullah-Al-Mamun, Shahin Akhter, "Social media bullying detection using machine learning on Bangla text", 10th International Conference on Electrical and Computer Engineering, IEEE Xplore, 2018.
- [4] Davidson, T., Warmesley, D., Macy, M., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. Proceedings of the International AAAI Conference on Web and Social Media.
<https://ojs.aaai.org/index.php/ICWSM/article/view/14955>

[5] Waseem, Z., & Hovy, D. (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. Proceedings of the NAACL Student Research Workshop.

<https://aclanthology.org/N16-2013.pdf>

[6] Scikit-learn Documentation. Machine Learning in Python.

<https://scikit-learn.org/stable/documentation.html>

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

<https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>

[8] Streamlit Documentation. The fastest way to build data apps in Python.

<https://docs.streamlit.io>

[9] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media.

[10] NLTK: Natural Language Toolkit. Text Processing Libraries in Python.

<https://www.nltk.org>

