International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Abusive Comment Detection and Prevention

G.Devi Srinivas[1], G.Pavani[2], J.B.S.S.Srinitha[3], K.V.Mainkanta Kumar[4], G.Phani Adithya[5], Mr. Siva Krishna[6]

[1,2,3,4,5]Bachelor Of Technology In Computer Science And Engineering, Vsm College Of Engineering
[6]M.Tech Associate Professor, Dept. Of CSE

ABSTRACT

Abusive language is any wording that includes abusive phrases, whether they are used in jokes or to provoke someone. On social media sites like Facebook, Instagram, Twitter, LinkedIn, and others, practically all users now use derogatory language. Finding an abusive word in the vast world of social media is one of the more challenging tasks because it cannot be solved by word matching alone. Social media's rapid growth and the increased directness of communication between people in different nations and mental states have led to an increase in the use of profanity amongst these individuals. The necessity to automatically identify such speech and edit any data that contains abusive language thus emerges. The goal of this project is to identify abusive words in reviews and categorize them as either positive or negative. The foundation of our method is the automatic collection of unigrams and patterns from the training set. In order to determine whether a review is abusive or not, we employ the Random Forest Decision Tree.

Keywords: Random Forest, data pre-processing, sentiment analysis, decision trees, and abusive words.

INTRODUCTION

1.1 INTRODUCTION

In social media, anyone is free to post their ideas and views while not declaring his/her identity. Detection of Hate Speech, Offensive language and utterance employed by social media users became one in every of the foremost challenges of this case. Social media users were being targeted by Hate Speech and Offensive language like abusive, hurtful, disparaging or unlawful user-generated content by some mischievous users. These on-line platforms offer academic degree open place to debate and treat totally different matters however abusive comments and on-line violence on people have turned this into a awfully necessary social issue.

As a result of the misuse of on-line interactions, an outsized range of people have fallen into depression, anxiety, and alternative psychological state issues. The worldwide use of social media is ever growing. The 2019 social network figures indicate that there are 3.5 billion consumers of social media globally, and that number is scarcely increasing. That's equal to 40 quarter of this country. With the growth of social media, users particularly teens are increasingly disbursing significant sums of it on various social networking platforms to communicate with others, exchange data, and follow mutual interests. In 2011, seventeenths of adolescents using commonplace social networking sites and almost one in four teenagers reach their top social network sites ten as more times regularly.

In fact, cyber-bullying happens through the social media indicate derogatory comments. This has been noticed that nineteen teenagers mention someone on social networking platforms having published or denoted negative or humiliating stuff about them. When teens become loads of relatively derogatory filled with skewed and dangerous material relative to adults, web monitoring of offensive content to secure teen health web is an essential activity for academic degree. Social network managers usually conduct a manual web analysis on-line contents to seek out and delete offensive material. The internet is a hurling spot too, full of interesting data and stuff to learn and speak. But for some, it would be a dark and frightening place to rule over their violence of influential and unknown trolls victimizing the numerous social networking sites. Last year, a study commissioned by the Bench Research Centre showed that 73% of adolescents witnessed someone abused in how on- line, with 40% witnessing harassment on the net directly. Celebrities and celebrities have removed their social media pages in recent years as they get criticism on their statements about violence. The regulation of social media has been headlining news this year. And one in every of the continual themes has been the potential for AI to be used as a filter or safety mechanism to help the varsity giants' police the content revealed on their platforms. this may be a step within the correct direction. and since the key platforms, Instagram enclosed, come underneath increasing pressure to manage you'll expect to check lots of constant. When you're facing abuse on-line, it'll feel overwhelming and you may not confirm how to report an issue like this. Here's a handy orient how to change abuse on the foremost social media platforms and therefore the thanks to report incidents and therefore the thanks to fully block such comments.

1.2 EXISTING SYSTEM

The existing systems for abusive comment detection mostly rely on conventional machine learning algorithms such as Support Vector Machines (SVM), Naive Bayes, and Decision Trees. These models are trained on labeled datasets using handcrafted features like word frequency, n-grams, TF-IDF, and sentiment scores. Although these methods are relatively easy to implement
and interpret, their effectiveness is limited due to the inability to understand the deeper context and semantics of the language used.Moreover, most of these systems are designed for detection only—they flag abusive content after it has been posted. This means harmful comments can still reach the public before moderation occurs, reducing the impact of such detection. Additionally, these systems often fail to detect indirect abuse, sarcasm, or newly coined abusive terms that are not part of their keyword-based filters. As a result, their accuracy and adaptability are limited, especially in dynamic online environments.Another limitation of existing systems is their lack of multilingual and code-mixed language support. Many abusive comments on social media platforms are written using a mix of languages, local dialects, or slang, which traditional systems struggle to handle. There is also minimal focus on real-time prevention mechanisms, such as warning users before they post abusive content or automatically filtering harmful language during submission.

LIMITATIONS:

Language Variability: These systems struggle to detect abuse in multilingual, code-mixed, or slang-heavy text, reducing their overall effectiveness.

Static Keyword-Based Filters: Many existing models rely on predefined keywords which can be easily bypassed with slight modifications or spelling tricks.

Lack of Real-Time Prevention: Current systems often focus only on post-submission detection rather than real-time prevention or user warnings before comment posting.

High False Positives/Negatives: Traditional methods may incorrectly label non-abusive content as harmful or fail to detect subtle abusive language, affecting accuracy.
Privacy and Ethical Issues: Monitoring user comments for abuse may raise privacy concerns, especially when analyzing personal data without clear consent.
Limited Adaptability: Existing systems are often not adaptive to the evolving nature of online language and require frequent updates to remain effective.

1.3 PROPOSED SYSTEM

The proposed system aims to enhance the detection and prevention of abusive comments using a combination of machine learning and deep learning techniques. Unlike traditional approaches, this system leverages advanced models like Support Vector Machines (SVM), Naive Bayes, and BERT (Bidirectional Encoder Representations from Transformers) to better understand the context, semantics, and tone of user comments. These models are trained on a custom dataset containing various forms of abusive language, including code-mixed and slang content, ensuring more accurate and robust classification.
To further improve efficiency, the system integrates natural language processing (NLP) techniques such as tokenization, stop word removal, and lemmatization. Feature extraction is done using both TF-IDF and contextual embeddings (from BERT), enabling the model to detect subtle and context-dependent abuse that keyword-based systems typically miss.In addition to detection, the proposed system includes a prevention mechanism. When a user attempts to post an abusive comment, the system performs real-time scanning and flags the content. If the comment is found abusive, the user is either warned or prevented from posting, depending on severity. This proactive step helps reduce the visibility of toxic content and promotes a safer online environment.Overall, the proposed system is designed to be scalable, adaptive, and accurate, making it suitable for implementation on social media platforms, forums, and comment sections of websites where user interaction is high. It not only detects harmful content effectively but also helps in discouraging users from spreading hate or offensive speech.

ADVANTAGES

Improved Accuracy: By using advanced models like BERT and SVM, the system captures the context and semantics of language more effectively than traditional keyword-based filters.
Real-Time Prevention: The system provides instant detection and alerts, preventing abusive comments from being posted in real time.
Adaptability: It can adapt to new slang, code-mixed language, and evolving abusive patterns through regular updates and retraining.
User Safety: Helps create a safer online environment by reducing the spread of toxic or harmful content.
Scalability: The system can be deployed on various platforms and scaled to handle large volumes of user-generated content.
Language Support: It can be extended to support multiple languages and dialects, increasing its reach and effectiveness.

2. LITERATURE SURVEY

Kirti Kumari and Jyothi Prakash in HASOC 2019 detected hate speech and abusive contentusing Deep Learning approach. The Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC) organizer team defines Hate Speech as describing negative attributes or deficiencies toward groups because of race, political opinion, sexual orientation, gender, social status, health condition or similar [10]. A large number of works [1,5,6,15] are reported by the researchers on Hate Speech detection

for English language only and very few works [2,8] have been reported for mixed languages such as English-Hindi, English-Bengali, and other languages.

In this paper, they have used the multi-lingual HASOC Corpus [10] and proposed a deep learning model based on Convolutional Neural Networks (CNN) to identify Hate Speech and Offensive content on multi-domain social media platforms collected from Facebook and Twitter. They have used three types of embedding of text and transliteration tools to normalize Devanagari to Roman script for code- mixed Hindi corpus. The rest of the paper is structured as follows. Section 2 presents associated works for the detection of Hate Speech and Offensive Language while Section 3 presents our suggested framework for identification of Hate Speech, Offensive language, and Profanity. Section 4presents the finding of the suggested scheme. Finally, in Section 5, they have concluded the paper and have discussed the future directions for these tasks. She described as social media and online platforms have grown in terms of impact and acceptance of users, various problems such as Hate Speech, Profanity and Offensive language on these platforms have increased drastically. Several systems have been proposed by researchers for automatic detection and classification of these problems. A lot of research works have been done for the English language but very few works have been done for the other languages and code-mixed languages. In this paper, we have focused on multi-lingual text, especially for Hindi and English languages and used a deep neural network model to detect Hate Speech, Offensive language, and Profanity. The Hindi-English code-mixed corpus was created by Aditya Bohra, Deepanshu Vijay and Vinay Singh using the tweets written online in the last five years.

Tweets have been discarded from Twitter then using Twitter Python API which uses twitter's sophisticated search option.

They've extracted the tweets by picking from politics other hashtags and keywords. Community marches, demonstrations, and so on, who have a strong reputation for hate speech. They accessed 1,12,718 tweets in json format from Twitter, which contains details such as timestamp, URL, email, person, re-tweets, comments, full name, I d and likes. A thorough review to delete all the disruptive messages was carried out. A noted corpus of Hindi-English code-mixed content, consisting of tweet ids and the accompanying annotations, is provided in this article.

They also introduce the controlled method used in code-mixed text to identify Hate Speech. The corpus is made up of 4575 code-mixed messages with annotated hate speech and regular expression. The vocabulary in the tweets isoften annotated with the term root words. Word n-grams, phrase n-grams, punctuations, terms ofnegation and lexicon of hatred are the characteristics included in our classification scheme. Excellent 71.7 percent precision is reached when all characteristics are integrated into the task vector using SVM as the classification scheme. The corpus may be annotated as part of potential research with part-of - speech tags at word level and can produce improved outcomes.

In addition, in particular, the annotations and studies outlined in this paper can also be performed for code-mixed texts comprising more than two languages from multilingual societies. In the case of Support Vector Machine and Random Forest Classifier respectively, Support Vector Machine does stronger than Random Forest Classifier and achieves 71.7 percent maximum accuracy when all attributes have been used. Character N-Grams was found to be the most effective in SVM while Word N-Grams was the most reliable in Random Forest Classifier. Warner and Hirschberg (2012) identified hate speech designed for different factors like ethnicity. Throughout their research, they described hate speech and then collected data from Yahoo and the American Jews Congress (AJC), that Yahoo received its newsgroup data, and

AJC gave URL identified as offensive websites. Throughout their first attempt, they categorized data at paragraph and accuracy higher than chance level only when using glottal or prosodic features. The best prediction performance was achieved with the new multichannel method, which used four features (prosodic, glottal, TEO, and spectral). In the case of the person-based approach with two sets of weights, the new multichannel method provided a high accuracy level of 73% and the sensitivity-to-specificity ratio of 79%/67% for predicting future depression. stage, but then used this data set for annotation by requiring annotators to manually transcribe the data collection. We concentrated on stereotyping and therefore agreed to allow stereotyping language model to label hate speech. They first created a classifier of an anti-Semitic expression. They found 9000 paragraphs which suited their normal speech and eliminated those subsections which were not provocative. Then seven more groups were chosen to annotate the results. They utilized two-fold cross validation classifier after that annotation for their gold corpus to find a polished collection of results.

Motivated by the research performed in Kwok and Wang, (2013) suggested a way of identifying black-hatred speech via Facebook. Thousands of tweets were organized to examine keywords or thoughts suggesting speeches of hatred. A test has been circulated to students of various races to determine the seriousness of the claims. A testing data collection of 24582

tweets was pre-processed to fix variance in pronunciation, delete terms from stoppage and eliminate URL etc. To identify tweets, NB classifier illustrated racial and non-racist messages, and they described influential characteristics from such tweets. The classifier reported 86 percent precision. To order to identify antagonistic posts on Twitter, Burnap et al. (2013) established a rule-based methodology and used relation terms as attributes. It often incorporated charges and forms of reference directed at an individual or individuals after a socially destructive incident as attributes, in an attempt to convey the meaning and usage of the word. Their findings indicated that traditional learning methods had changed. With the support of social network and text mining research, Ting et al. (2013) suggested framework for detecting hate groups over Twitter. We also extracted functionality like keywords which are sometimes found in classes. Sureka et al. (2012) suggested a method to uncover hate promotion posts, consumers and their secret groups on YouTube focused on the data analysis and social network analytics. Chen et al. (2013) developed a system for the detection of YouTube terrorist images. Author derived user-generated data from lexical, syntactic, and content-specific elements, and used different feature-based classification techniques to identify images. Agarwal and Sureka also suggested a based crawler (best first scan and shark hunt) method for extracting accounts of YouTube users who spread hatred and Different solutions to machine learning were made to tackle the toxic language issue. The bulk of methods deal with extraction of features from the document. Any research used lexical tools

such as dictionaries, and bag-of-words. Such features were found to struggle to grasp the meaning of the sentences. Approaches based on N-gram have also been used which display comparatively better performance. While lexical features work well in identifying offensive items, they struggle to discern offensive sentences that include the same terms but in specific orders without considering the syntactic structure of the entire sentence.

The natural language process parser, introduced by Stanford Natural Language Processing Group, has been used in the same analysis to catch the grammatical dependencies inside a phrase. Linguistic tools such as parts-of - speech were often used in hate speech identification issues, these methods consist of identifying the phrase type, for example, personal pronoun (PRP), Verb non-3rd person singular present form (VBP), Adjectives (JJ), Determiners (DT), Verb base types (VB).

Many experiments have been carried out on sentiment-based approaches to identify the offensive language that has been released in recent years. One illustration is the work utilizing emotion analysis to spot abuse in messages, and using subject models from the Latent Dirichlet Allocation (LDA) to classify important topics in such texts. Research for the Identification of Violence were also performed on Cloud 2.0. Recent times, dispersed representations of terms, often called word embedding, have been suggested for a specific reason. Current usage of deep learning strategies is made using paragraph2vec methodology in text recognition and sentiment analysis. Convolutionary Neural Network (CNN)enabled classification, which applies to the development of a text classification CNN, where they worked with a Twitter hate-speech content classification method built on a deep-learning, CNN Model. Our methodology is different from the aforementioned analysis as much of the research conducted on hate speech is on a particular subject or area based on the phenomenon that is emerging / developing in a defined time. At the time they attempted to concentrate various topics on specific research such as culture, sex, ethnicity etc. We have provided an approach in this research to basic queries.
Secondly, our methodology varies from above as we use real-time stream tweets as well as details regarding user profiles.

3. SYSTEM REQUIREMENTS ANALYSIS AND SPECIFICATION

Data Collection and Integration: Collecting user-generated content from social media platforms, comment sections, forums, and custom datasets. This data is preprocessed and labeled to train the detection models effectively

.**Text Processing and Analysis:** Applying Natural Language Processing (NLP) techniques such as tokenization, stop word removal, lemmatization, and vectorization (TF-IDF, word embeddings) to analyze the content and extract meaningful features.

Machine Learning Model Integration: Using classification algorithms like SVM, Naive Bayes, and BERT to detect abusive or harmful content. These models are trained to understand the context, tone, and intent behind comments.

Real-Time Detection and Prevention: The system scans user input in real-time and flags or blocks comments if they are found abusive. It can also provide warning messages to the user before allowing submission.

Privacy and Ethical Considerations: Ensuring user data privacy by anonymizing input data and adhering to ethical guidelines in content moderation and AI usage.

User Interface and Accessibility: Developing a simple and intuitive interface for users and moderators to submit, review, or flag comments, ensuring accessibility across devices and platforms.

Evaluation and Feedback Mechanisms: Implementing systems to monitor detection accuracy and gather user or moderator feedback to continuously improve model performance.

3.2 HARDWARE REQUIREMENTS

MINIMUM (Required for Execution)		**MY SYSTEM (Development)**
System	i3 Processor 5th Gen	i5 Processor 11th Gen
Hard Disk	30 Gb	500 Gb
Ram	4Gb	8Gb

3.3 SOFTWARE REQUIREMENTS

Operating System	**Windows 11**
Development Software	Python 3.10
Programming Language	Python
Domain	Artificial Intelligence &Machine Learning
Integrated Development Environment (IDE)	Visual Studio Code
Front End Technologies	HTML5, CSS3, Java Script
Back End Technologies or Framework	Django
Database Language	SQL
Database (RDBMS)	MySQL
Database Software	XAMPP Server
Web Server or Deployment Server	Django Application Development Server
Design/Modelling	Rational Rose

3.4 FUNCTIONAL REQUIREMENTS

Abusive Comment Detection System:Implement an advanced comment detection system capable of accurately identifying abusive, toxic, or offensive content in real time using natural language processing and machine learning models.

Database Integration:Integrate with datasets containing labeled abusive and non-abusive comments, along with social media APIs or comment platforms, to train, test, and update the detection models.

Machine Learning Algorithms:Develop and train machine learning models such as Support Vector Machines (SVM), Naive Bayes, and BERT to analyze comment text and detect patterns of abusive language. Utilize tokenization, sentiment analysis, and contextual embedding techniques to enhance accuracy.

Scalability:Ensure the system is scalable to process a large volume of user-generated content across multiple platforms efficiently, especially during peak activity or trending discussions.

User Interface:Design an intuitive dashboard for moderators and administrators to monitor flagged comments, adjust detection thresholds, and visualize real-time analytics on abusive comment trends.

Feedback Mechanism:Incorporate a feedback loop that allows users and moderators to flag false positives/negatives, enabling the system to learn and adapt over time for improved accuracy..

3.5 NON-FUNCTIONAL REQUIREMENTS

PERFORMANCE

The system should provide fast and accurate classification of comments, even when processing large volumes of user-generated content in real time.

SCALABILITY

It should be able to handle an increasing number of users, comment submissions, and content moderation requests without compromising system performance.

RELIABILITY

The system should consistently detect abusive content with high accuracy and low false positive/negative rates, operating without failures or significant downtime.

USABILITY

The interface should be intuitive and user-friendly for both content moderators and end users, enabling efficient interaction and review of flagged comments.

MAINTAINABILITY

The system must be easy to maintain and upgrade, with clear documentation of machine learning models, APIs, and deployment procedures.

ACCESSIBILITY

The system should be accessible to users with disabilities, complying with relevant accessibility standards and providing alternative modes of interaction.

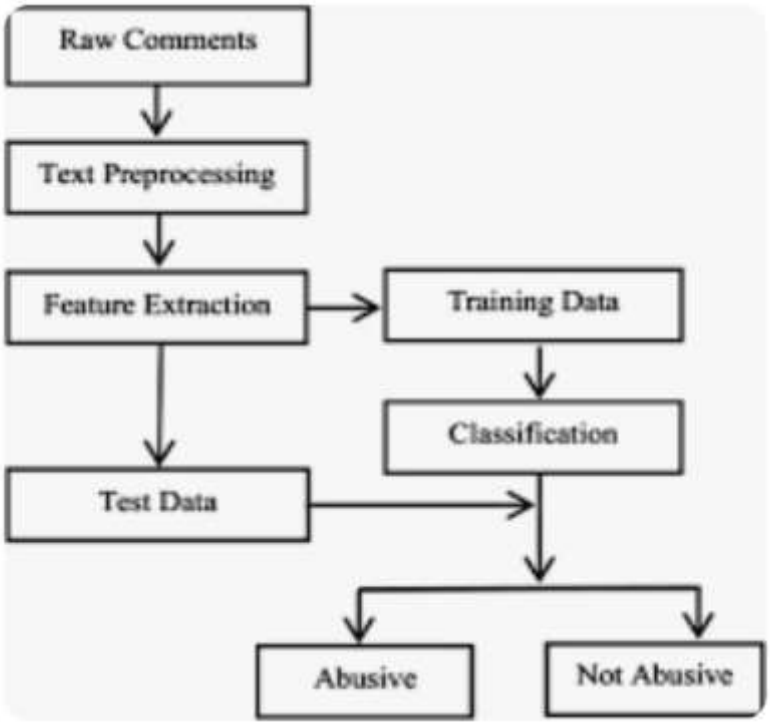
SECURITY

Robust data privacy and security measures should be implemented to protect user data, prevent misuse, and ensure compliance with content moderation policies

SYSTEM DESIGN

1 SYSTEM ARCHITECTURE

Machine learning models are employed to detect and classify abusive comments. These include algorithms such as Support Vector Machines (SVM), Naive Bayes, and BERT, which analyze the textual content of user-generated comments to detect toxic, offensive, or harmful language. The user interface provides a simple and intuitive platform where users can submit comments or texts, and moderators can view flagged content and take appropriate action. The system incorporates a feedback mechanism for continuous model improvement based on user and moderator responses. The architecture supports login functionality for administrators to manage reports and refine detection rules based on evolving abuse patterns



4.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

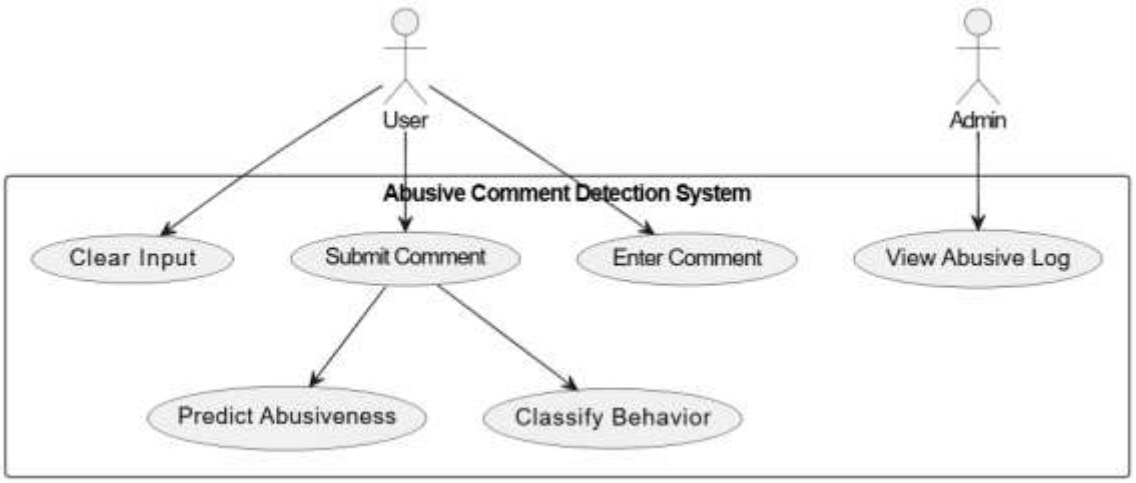
The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

4.2.1 USE-CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Fig 4.2.1.1



4.2.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

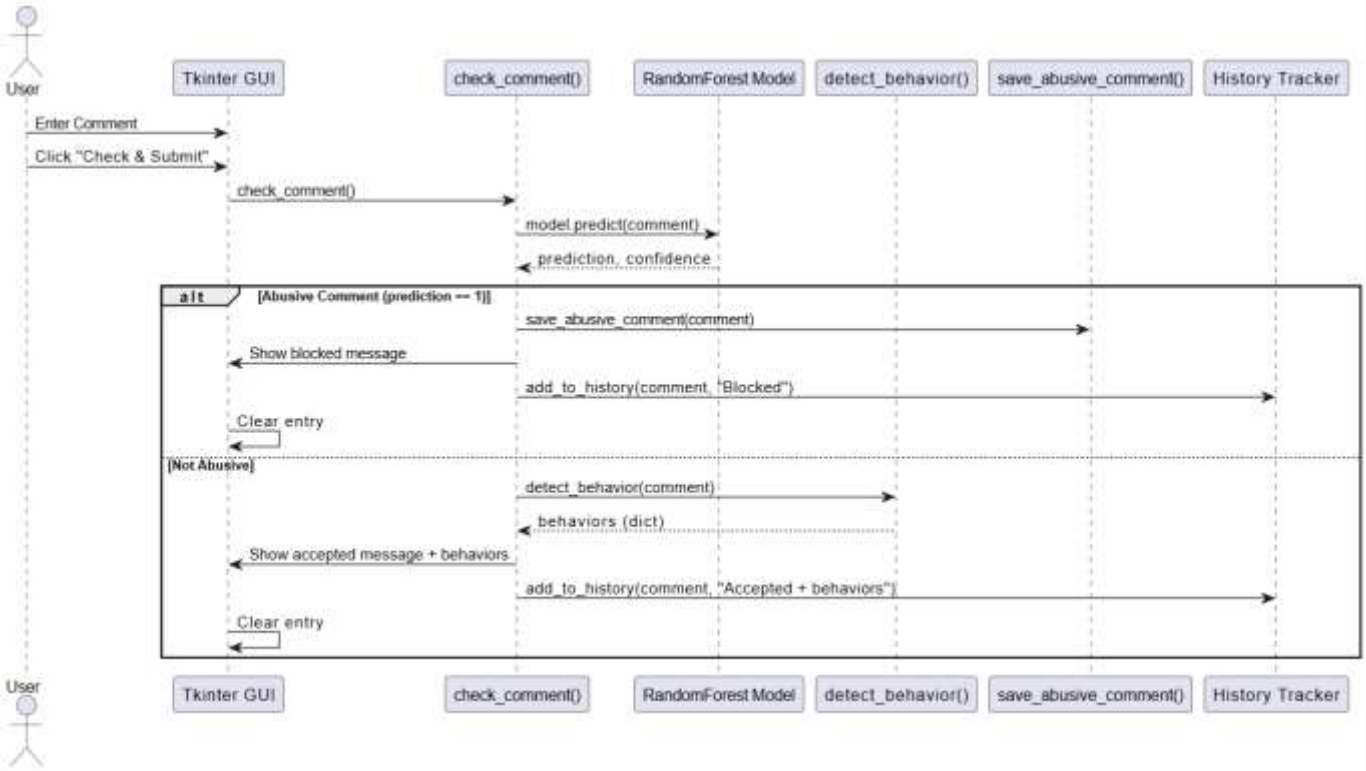


Fig 4.2.3.2

4.2.4 ACTIVITY DIAGRAM :

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

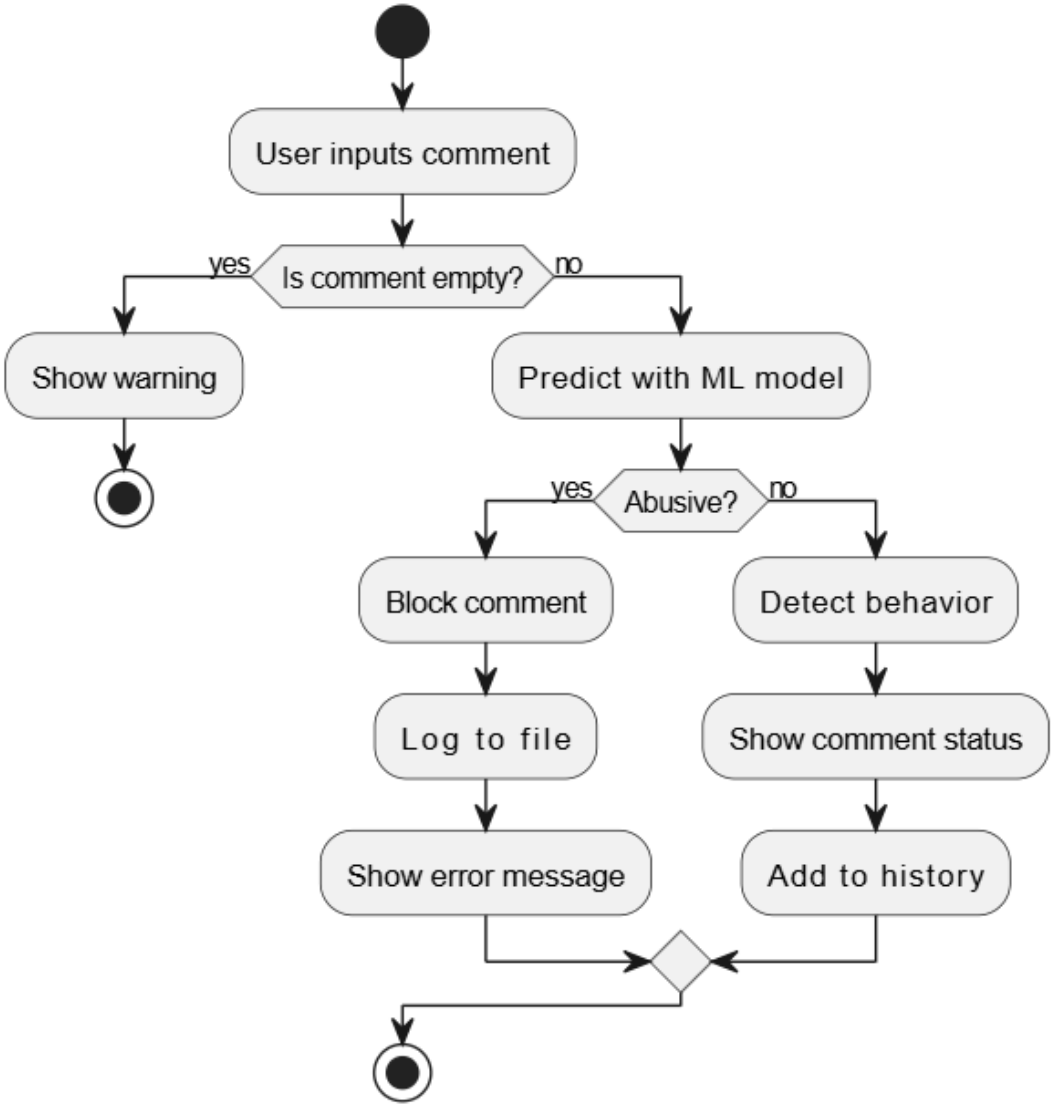


Fig 4.2.4

4.2.5 DEPLOYMENT DIAGRAM :

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.The deployment diagram maps the software architecture created in design to the physical system architecture that executes it. In distributed systems, it models the distribution of the software across the physical nodes.

The software systems are manifested using various artifacts, and then they are mapped to the execution environment that is going to execute the software such as nodes. Many nodes are involved in the deployment diagram; hence, the relation between them is represented using communication paths.

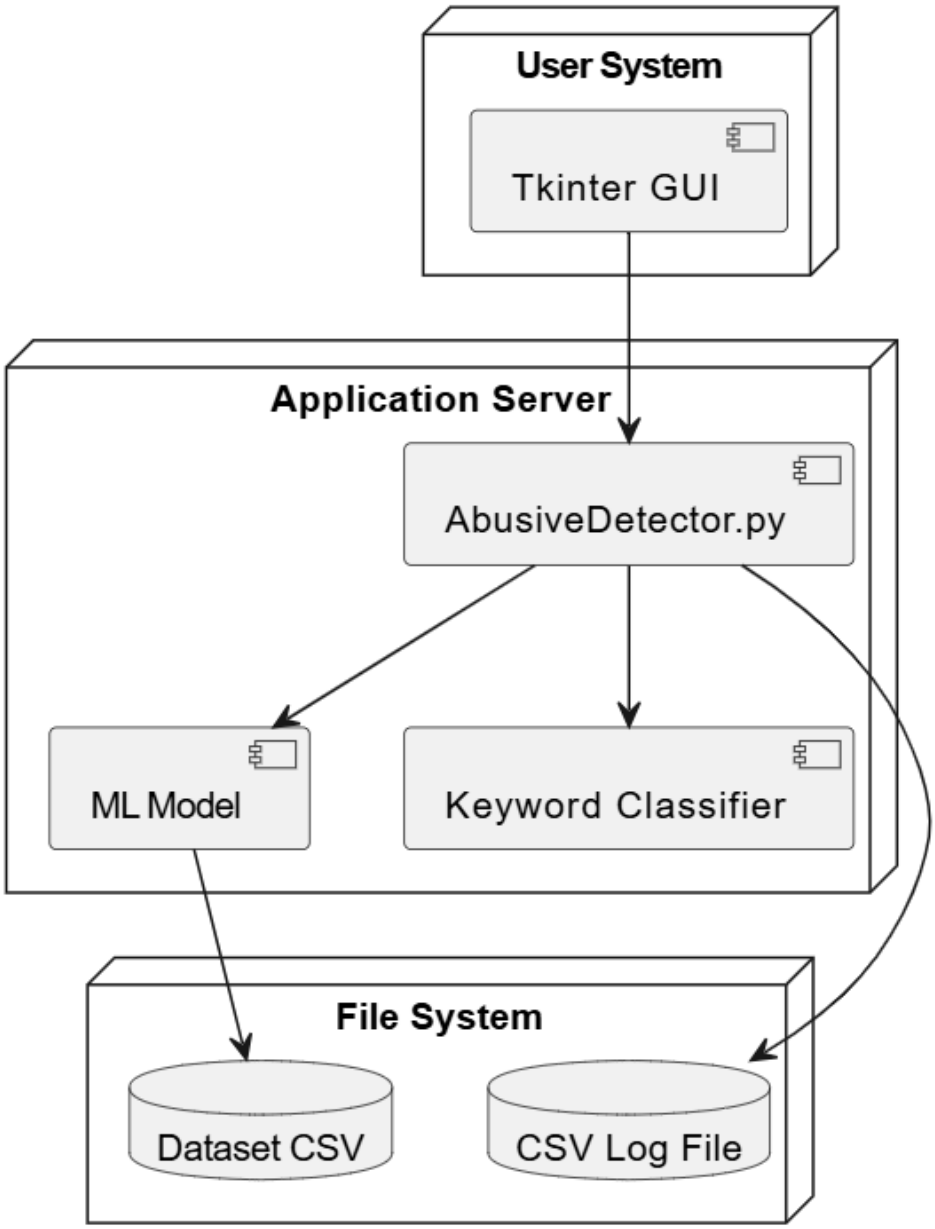


Fig 4.2.5

5. IMPLEMENTATION

5.1 PROJECT MODULES

Reading Data

Pre-Processing

Special Characters Removal

Tokenization

Stop Words Removal

Hate/Not Classification

Classification

Decision Tree Algorithm

Random Forest Algorithm
Confusion Matrix
Classification Rate/ Accuracy
Precision
Recall

5.2 MODULE DESCRIPTION:

The" ABUSIVE COMMENT DETECTION AND PREVENTION" system consists of several modules that work together to achieve its objectives.

Here are key modules:

READING DATA:

The data set in the form of .tsv (Tab separated values) is translated to the format of .csv (Comma separated values). In order to perform some csv file operations, it is important to convert it to a data frame. To do this, a built-method in python called 'read csv' is available. In a very library named pandas, this method has to be imported into our IDE.

This process is begin by examining the structure of TSV file. Typically, a TSV file consists of row and columns, with each row representing a data entry and columns separated by tab characters. Python provides several methods to convert tsv files to csv format here we used the pandas library its offers powerful data manipulation capabilities.

PRE-PROCESSING:

Pre-processing refers to the transformations that were applied before feeding on our data. In order to obtain better outcomes from the implemented model in machine learning projects, the data format must be very accurate. Every specified machine learning models requires information information in a very specific format, For example, the Random Forest algorithm does'nt accept null values, so null values must be handled from the first raw data set to execute random forest algorithms. Another thing is that data set should be structures in such a simple way that quite a similar machine learning and deep learning algorithm is implemented in one swt of data, and the best is choosen from them.

Steps Involved In Pre-Processing:

Special Characters Removal
Tokenization
Stop Word Removal
Hate/not Classification

SPECIAL CHARACTERS REMOVAL:

Removal of special characters is a critical preprocessing step aimed at enhancing the accuracy of machine learning models by reducing noise and focusing on relevant tetual features.

Role of special characters removal in abusive Content:

- **Obfuscation of Abusive Words:** Users often manipulate words by inserting special characters like for example, @ction for Action, "$peed for Speed" to evade detection systems.
- **Noise Introduction:** Special characters, including punctuation marks and symbols can introduce noise,making it challenging for models to accurately identify abusive language.

Removal of Unnecessary Special Characters:

- Eliminate non-alphanumeric characters that do not contribute to the semantic meaning of the text.

- Utilize regular expressions to identify and remove unwanted characters, streaming the text for analysis.

TOKENIZATION:

Tokenization is a fundamental preprocessing step that involves breaking down into smaller units, such as words or subwords, to facilitate analysis by machine learning models. This process transforms unstructured text into structured data that models can interpret and learn from effectively.

Role of tokenization in Abusive comment Detecti0on:

- Text Standarization: Tokenization helps standardize text by coverting it into a consisting format, making it easier to apply various preprocessing techniques and ensuring that the input data is uniform for model training.
- Facilitating Advanced Models: Modern natural language processing(NLP) models,require tokenized input to function correctly. These models rely on tokenized data to capture contextual nuances and semantic relationships within the text.

  Approaches to Tokenization in Abusive Comment Detection:
- Word-Level Tokenization: Splitting texts into individual words. While straight forward, this method may not effectively handle out-of-vocabulary words or capture contextual meanings.
- Subword Tokenization: Breaking down words into smaller units(subwords) to manage rare or unknown words better. Techniques like Byte pair encoding or Wordpiece are commonly used in this approach.
- Character-Level Tokenization: Dividing texts into characters. This method can capture morphological nuances but may lose word-level semantic information.

STOP-WORD REMOVAL:

Stop word removal is a preprocessing technique aimed at eliminating common words that do not contribute significant meaning to the text. These words are known as stop words, including articles(‘a’, ‘an’, ‘the’) prepositions(‘for’, ‘on’, ‘in’) and conjunctions(‘and’, ‘but’, ‘or’). While essential for constructing sentences, stop words often add noise in text analysis, especially in tasks like abusive content detection.

Benefits of Stop Word Removal:

- Enhanced Model Performance: By removing stop words, thefeature space is reduced, leading to improved efficiency and accuracy in machine learning models.
- Reduced Data Dimensionally: Eliminating these words decreases the dimensionality of the data, making processing faster and less resource-intensive.

HATE/NOT CLASSIFICATION:

This process classifying comments as ‘hate’ or ‘not hate’ involves developing a system that can accurately distinguish between hate speech and non-hate content. This classification is essential for moderating online platforms and preventing spread of harmful content.

Understanding Hate Speech Vs Offensive Language:

- Hate Speech: Language that expresses hatred, contempt, or prejudice against a particular group based on attributes like race, religion, ethnicity.
- Offensive Language: Words or expressions that may be rude, vulgar, or disrespectful but do not necessarily target specific groups or incite violence.

  Approaches to Classification:
- Machine Learning Models: Employ supervised learning using labelled datasets to train models capable of distinguishing between hate speech and non-hate content.

Features may include n-grams, syntactic patterns, and semantic embeddings.
Challenges include accurately capturing and handling nuances in language.

CASE STUDY: TWITTER DATA ANALYSIS

A study analyzed tweets containing hate speech keywords, categorizing them into hate speech, offensive language, and neither. Racist and homophobic tweets were more likely to be classified as hate speech, While sexist tweets were generally labelled as offensive. Tweets lacking explicit hate keywords posed classification challenges, underscoring the need for sophisticated models that consider context and semantics.

CLASSIFICATION:

Classification is a matter of defining the group or groups (sub-populations) a new observation belongs to, on the basis of a training set of data with observations (or instances) whose membership in the category is understood. Classifier efficiency is highly dependent on the data properties to be categorized. No single classifier works best on any given question. Various empirical tests are performed to check the performance of the classifier and to search the information characteristics which determine the performance of the classifier. For a given question, however, determining an appropriate classifier is often more an art than a science. Precision and recall indicators are common metrics used to determine an organization's quality. There are a number of classification techniques available, such as Decision Tree, Support Vector Machine (SVM), logical regression, Naive Bayes, we used the Tree classifier option to encourage better results**.**

DECISION TREE ALGORITHM:

Decision Tree Classifier could also be a basic classification method which is widely utilized. Decision trees all were seen as one of the its most commonly applied algorithms in machine learning, mostly for classification but more often in regression cases. Decision Tree Classifier asks a series of carefully constructed issues regarding qualities of test records. A add-up questions are asked upon having a comment before a decision can be made about name of the album group. Selecting spatial splits significantly impacts a tree's accuracy. The Classification and Regression trees eligibility requirements were special.

Decision trees use several algorithms to generate a push for a node to be divided into two or maybe more sub-nodes. The formation of forum-nodes improves the uniformity of the sub nodes that arise. In other letters, we're willing to say that with relevance the aim function improves the potency of the node. The selection tree divides the partitions into all obtainable variables in order to pick the split ending of most stratified sub-nodes. Furthermore, the form of target variables is provided by the algorithm choice. Decision Tree algorithms consist mainly of data for training, raw data, a function for heuristic validation and an operate for the stop criteria. These systems use the recursion of the divide and conquer system to elicit downwards data from node of inspiration.

In the end, the decision trees are the "gold-standard" for 14 partition-based models. The decision trees it was first applied to the modeling of languages. One node was that the position to begin is preceded by binary queries which are asked as a way of arbitrarily partitioning the space of history. Due to the partitioning of the space, "leaves" are created and the training data is used to measure the probability of P (w) for the next component. As the traversal continues the questioning becomes more insightful with the use of theoretical metrics of information.

This module is responsible for training machine learning and deep learning models for depression detection. It uses the depression dataset from Twitter tweets to train the models..

RANDOM-FOREST ALGORITHM:

Random Forest is a supervised machine learning algorithm that operates as an ensemble method, combining multiple decision trees to enhance predictive accuracy and control overfitting. In the context of abusive comment detection and prevention, Random Forest is employed to classify comments as abusive or non-abusive by analyzing various features extracted from the text.

Working of Random Forest in Abusive Comment Detection:

- Data Sampling: Random Forest begins by generating multiple subsets of the training data through bootstrapping, where each subset is created by randomly sampling the original dataset with replacement.
- Decision Tree Construction: For each data subset, a decision tree is built. During the construction, at each node, a random subset of features is considered for splitting, introducing diversity among the trees.
- Prediction Aggregation: When predicting the class of a new comment, each decision tree provides a vote. The final prediction is determined by the majority vote (for classification tasks), leading to a more robust and accurate model.

Application in Abusive Comment Detection:

In the realm of abusive comment detection, Random Forest classifiers are trained on datasets containing labeled comments. Features such as word frequency, presence of offensive terms, sentiment scores, and syntactic patterns are extracted from the comments. The Random Forest model learns to distinguish between abusive and non-abusive comments based on these features. For instance, studies have demonstrated that Random Forest classifiers can effectively identify cyberbullying language by analyzing features like text length, use of profanities, and emoticon usage.

CONFUSION-MATRIX:

A confusion matrix could be a technique for summarizing the performance of a classification algorithm. the quantity of correct and incorrect predictions is summarized with count values and weakened by each class. this can be the key to the confusion matrix.

Calculating a confusion matrix can provide you with a much better idea of what your classification model is getting right and what kinds of errors it's making.The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives your insight not only into the errors being made by your classifier but more importantly the categories of errors that are being made. it's this breakdown that overcomes the limitation of using classification accuracy alone.

CLASSIFICATION RATE/ACCURACY:

The accuracy of a machine learning classification algorithm is a method to live how often the algorithm classifies a datum correctly. Accuracy is that the number of correctly predicted data points out of all the info points. More formally, it's defined because the number of true positives and true negatives divided by the 17 metrics that use various ratios of true/false positives/negatives. Together, these metrics provide an in depth observe how the algorithm is classifying data points. Consider a classification algorithm that decides whether an email is spam or not.

The algorithm is trained, and that we want to determine how well it performs on a collection of ten emails it's never seen before. Of the ten emails, six aren't spam and 4 are spam. The algorithm classifies three of the messages as spam, of which two are literally spam, and one isn't spam. within the table, truth positives (the emails that are correctly identified as spam) are coloured in green, truth negatives (the emails that are

correctly identified as not spam) are coloured in blue, the false positives (the not spam emails that are incorrectly classified as spam) are coloured in red, and therefore the false negatives (the spam emails that are incorrectly identified as not spam) are coloured in orange. There are two true positives, five true negatives, two false negatives, and one false positive. Using the formula for accuracy, we get:
This algorithm has 70% accuracy classifying emails as spam or not.

PRECISION:

Precision improves when there is a high price for false positives. And let's say the matter is about carcinoma detection. When we have a very low-precision model, so several patients are advised they need melanoma, which will have certain exacerbations. There's plenty of rigorous checking and tension involved. When false positives are too heavy, after being barraged with false reports, persons who track the outcome will learn to recognize these. To encourage accuracy, we divide the total amount of correctly identified good examples by total number of similar cases expected. Large precision implies a strong example (a small number of FP) classified as positive.

RECALL:

Remember helps when false negatives are priced high. A false negative has catastrophic effects. False execute and we all die. When false negatives are prevalent, the thing you want to stop gets you hit. If you plan to disregard the vibration of a twig broken in an extremely dark wood, a false negative is and you get eaten by a cub. (A false positive is to stay sleepless all night in your tent in an excessively nervous state, keeping track of every noise in the woods, only to learn the morning after that those noises were made by a chipmunk. Not fun.) If you had a model that mistakenly makes nuclear weapons, you'd like to throw this out.

$$Recall = \frac{truepositives}{truepositives + falsenegatives}$$

Formula for calculating Recall

High recall, low precision: This implies that most of the positive examples are correctly recognized (low FN) but there are plenty of false positives.

Low recall, high precision: This shows that we miss plenty of positive examples (high FN) but those we predict as positive are indeed positive (low FP).

F -MEASURE: F1 is an aggregate indicator of the accuracy of a model that combines precision and recall, in this interesting way of adding and multiplying merely combines two items together just to create a separate bowl. That is, a good F1 score means you've got low false positives and low false negatives, so you recognise real threats correctly and you shouldn't be bothered by false alarms. Great consideration is given to an F1 score when it becomes 1 while the template is a complete failure and when 0. Since we have two measurements (Precision and Recall) it helps to own a measurement which portrays either. We quantify an F-measure that uses in situ mean of absolute value, since it more punishes the extreme values.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Perks of using a contradictory matrix. It reveals how frustrating every classification model was when making predictions. Confusion matrix not just to provides your intuition with your classifier's faults, it also provides you any kind of mistakes that are made. This summary allows you to solve the drawback of using precision classification only. Each column of the matrix of confusion reflects the cases of that predicted class. Every row of the confusion matrix reflects the specific class of instances. It gives knowledge not only of the mistakes made by a classifier but also of the bad decisions made.

5.3 INTRODUCTION TO TECHNOLOGIES:

PYTHON

Python is a **high-level, interpreted**, **interactive** and **object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

1.1 PYTHON FEATURES

Python's features include:

Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax.
This allows the student to pick up the language quickly.

Easy-to-read: Python code is more clearly defined and visible to the eyes.

A broad standard library: Python's bulk of the library is very portable and crossplatform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

NATURAL LANGUAGE PROCESSING:

Natural language processing (NLP) can be a computational area in which computers interpret, comprehend and infer significance from human language in a rather knowledgeable and useful manner. Through utilizing NLP, developers can coordinate and arrange information to conduct tasks such as automated summing, encoding, identification of named persons, abstraction of associations, analysis of emotions, identification of expression, and standardization of subjects. NLP is used to interpret language, making it easier for computers to learn how people talk. This connection between person and machine enables real-

world applications such as automated text summarization, emotion interpretation, subject extraction, identification of named persons, labeling of sections of expression, abstraction of associations, halting, and more. NLP is also used for text mining, MT, and the addressing of automated queries.

OPEN-SOURCE LIBRARIES:

Such libraries have real-world implementations for the algorithmic foundations of NLP. Algorithms includes a free endpoint API for some of such algorithms, without building up or running servers and storage at all.

- Apache open NLP: A machine learning toolkit of tokenizers, fragmentation of words, part-of-speech labelling, specified extracting of items, chunking, sorting, coreference resolution, and far more.
- Natural Language Toolkit(NLTK): A Python framework with plugin for word processing, sorting, tokenization, halting, labeling, filtering, and more.
- Stanford NLP: A series of NLP tools to include part-speech marking, the defined object recognizer, coreference negotiation framework, sentiment analysis and more
- Mallet: A Java package that has Latent Dirichlet Allocation, sorting of records, clustering, simulation of subjects, extraction of information, and more.
- To accomplish the language processing and us the Sentimental Analysis we'd like an open-source library named language NATURAL LANGUAGE TOOL KIT(NLTK)

REASONS FOR CHOOSING NLTK OVER OTHER OPEN SOURCE LIBRARIES:

- **NLTK Vs MALLET:** MALLET is java-based package for machine learning projects that are mainly deployed in industrial purposes. MALLET is more geared toward professional purpose and also highly complex to use whereas the NLTK is more user friendly and also it contains more documentation compared to MALLET**.**
- **NLTK Vs STANFORD NLP:** STANFORD NLP is additionally a python- based library in similar with NLTK. The speciality of STANFORD NLP is to a part of speech tagger. But the range of libraries provided by the NLTK lags in STANFORD NLP. STANFORD NLP has only one window such it can't be accessed by the third-party site. Once if this classifier able to operate it should be able to interact with numerous sites. With an amateur knowledge we've got decided that to change with NLTK in situ of STANFORD NLP.
- **NLTK Vs APACHE open NLP:** during this instance also, the most problem is primarily the programing language used. APACHE open NLP uses the java as programing language and it had been released sooner than NLTK such a large amount of of the standard users feel their comfortable levels in OPEN NLP than NLTK. But as NLTK is primarily focuses than text analysis kind of like OPEN NLP but the benefit of accessing the built-in libraries and also the inbuilt corpus is add-on feature for language Tool kit.

TOOLS USED:

NATURAL LANGUAGE TOOLKIT:

NLTK may be a pioneering tool for developing Python programs with human language data to work out. It offers quick-to-use frameworks for more than 50 organizational and lexical tools such as WordNet, along with a set of text processing libraries for grouping, tokenization, halting, labeling, filtering, and semance reasoning, wrappers for NLP repositories of institutional intensity, and a vibrant debate board. The language toolkit, or more generally NLTK, may be a collection of libraries and systems built within the Python programming language for functional and mathematical language processing (NLP) for English. Steven Bird and Edward Loper had created it inside the University of Pennsylvania's Department of

Computer and Information Technology. NLTK provides experimental details and interactive proofs. It is in the center of a book which explains the fundamental concepts behind the toolkit-supported language processing tasks, and a cookbook. NLTK is meant to help NLP or similarly associated study and training, including analytical linguistics, academic science, computation, knowledge processing, and machine learning. NLTK was widely used as a teaching method, as a private analysis device, and as a forum for the machining and development of test structures. NLTK provides advanced features for grouping, tokenization, trailing, labeling, sorting, and semanthetic thinking.

Modules used in building model:

Corpus (): Corpus can well be a series of written documents. You've got several corpora in NLTK, including Gutenberg Corpus, then on Internet and Chat Script.

Submodule stopwords():

A stop word might well be a widely encountered term (such as "the," "a," "an," "in") that has been configured to avoid a groundwork machine, both when indexing search inputs and when recovering them regardless of the effects of a foundations question.

Tokenize (): Tokenisation is the method by which vast volumes of text are separated into smaller pieces called tokens. To break a sentence into words we use the word_tokenize() process. In machine learning systems, the performance of word tokenisation may also be translated to Data Frame for improved text comprehension. As an entry for more text cleaning moves such as punctuation elimination, numerical character elimination or trailing, it is received visiting. Machine learning models need to practice numerical data and render a forecast. Word tokenization to numerical translation is a very necessary aspect of the text (string).

Sklearn: Scikit-learn (also named sklearn) might well be a sample library for the Python language training computer program. It supports numerous grouping, regression, and clustering algorithms which include supporting vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is intended to modularize with the NumPy and SciPy Python mathematical and science libraries**.**

BRIEF DESCRIPTION OF VARIOUS MODULES OF THE SYSTEM:

Text collection (dataset elicitation)

- Text Pre-processing
- Text Transformation
- Feature Selection
- Pattern Discovery
- Interpretation and Evaluation

Actually, this project deals with the analysis of comment and prevention the hate, offensive and profane words. According to the HASOC data set collected the we need to treat the text in a proper manner in order to obtain optimal results. Actually, here the data will be provided by the user but here as there will very less and secure access to the social media sites, we have taken predefined dataset provided at FIRE-2019 i.e., the Indo- European Languages. So, we have done analysis on two languages but the tests cases and the other illustrational activities are represented in only English language. Despite the Project covers English and Hindi but the stopwords sub-module is not predefined for Hindi language. So, the main challenge here is that we can remove the stopwords and also the special characters ain Hindi are also not predefined. So, the illustrations and further proceedings are carried out only in English Language dataset.

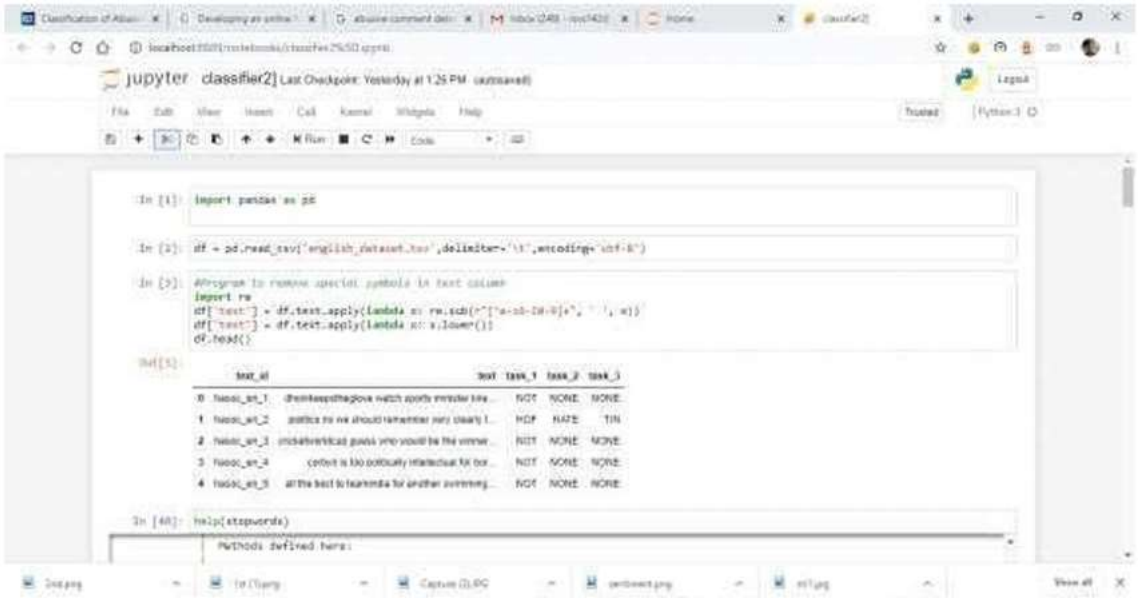
TEXT COLLECTION:

Here the text is collected using the twitter dataset and the we have an self sufficient number of comments or tweets to highlight the need to analyse and also detect the bad words in the given twitter dataset. The Twitter Data set is taken from the recent events and real-time tweets at HASOC workshop at FIRE-2019. The Text which is to to be pre-processed is collected and they are made into text file. File can be of any format i.e., the csv or txt file. But the to our convenience and the depending on the system requirements we are considering a text file i.e., in the format of txt file.

A Raw Data set of text from Twitter

TEXT PRE-PROCESSING:

In the pre-processing of the text for the raw data we have many challenges of clearing the special characters and also making it into a set of words so that they can be used for further analysis and also pattern recognition can be achieved optimally, so in this order of events we have some peculiar tasks but they prove to be important in the further proceedings of building the model. Here in comments, we tend to use the hastags(#) and also underscores (_) and many other special characters to represent the importance of the post and also to highlight the events and posting times. Also, we use different cases of letters tom maintain grammatical balance. This makes a bit difficult to analyse. As python is case sensitive so in order to make it convenient, we have to convert all the characters into single case either into lowercase. As universally used and more evident case is lower case, we have used and converted into the lower- case letters.



TEXT TRANSFORMATION:

As the first step is carries out there is a need to make it more organised for the further analysis. So, the need to make the words into groups has arouse. But in the further analysis we need to calculate the polarity scores for the group of words. It became difficult to calculate the polarity scores and also carry out sentiment analysis for numerous sets of words. So, it is hardly needed to remove the unwanted words that don't affect the meaning of the sentence. So, these words are predefined listed by the NLP toolkit NLTK and named as stopwords. This stopwords sub module is present in corpus module Also, after removing the stopwords we have to make some pattern son that it can be effective recognition of the text. In this model we are transforming the processed text into lists. Lists can easily and optimally handle in python.

In this step of process basic calculations and the other simple methodologies are used to achieve the optimality in the system. Firstly, by using the vader Sentiment

Analysis module we try to analyse the sentimental intensity and we try to calculate the polarity scores of the corresponding lists. If the score is more than predicted score (0.05) then it will be returned with 1 or else with 0. Also, some simple statistics and calculations have been calculated to count the number of words according to the category like hate, offensive and profane.

AI & ML :

In this domain, the primary objective is to develop and utilize AI and machine learning models to identify signs and symptoms of depression in individuals. This involves the analysis of various data sources, including text data from social media or chat conversations, and visual data from facial expressions. The goal is to create automated systems that can accurately detect depression, potentially aiding in early intervention and support for individuals experiencing depressive symptoms.

6.SAMPLE CODE

#ABUSIVE COMMENT DETECTION AND PREVENTION

#importing libraries

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline
import tkinter as tk
from tkinter import messagebox, scrolledtext
import datetime

# Load dataset
df = pd.read_csv("C:\\Users\\LENOVO\\Downloads\\final_comments_200plus.csv")

# Split data
X_train, X_test, y_train, y_test = train_test_split(df['comment'], df['label'], test_size=0.2, random_state=42)

# Train model with Random Forest
model = make_pipeline(TfidfVectorizer(), RandomForestClassifier(n_estimators=100, random_state=42))
model.fit(X_train, y_train)

# Save abusive comments to a file
def save_abusive_comment(comment):
with open("abusive_comments_log.csv", "a", encoding="utf-8") as file:
time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
file.write(f"{comment}","{time}"\n')

# Behavior detection function
def detect_behavior(comment):
comment = comment.lower()

threat_keywords = [ "kill", "murder", "hurt", "harm", "attack", "beat", "burn", "destroy", "slaughter", "wipe out",
"end you", "stab", "punch", "bomb", "explode", "shoot", "break your", "cut you", "eliminate",
"you're dead", "terminate", "you will suffer", "you'll pay", "watch your back", "massacre",
"i will find you", "i'll come for you", "you're finished", "i swear i'll", "i'll get you",
"take you down", "i'm coming for you", "i'm gonna beat you", "threaten", "hunt you",
"snap your neck", "crush you", "blow you up", "drop a bomb", "set you on fire", "acid attack",
"ruin your life", "end your life", "i'll destroy you", "brutalize", "ambush", "lynch",
"execute", "you're over", "exterminate", "obliterate", "hang you", "trap you","kill", "i'll kill you", "i will kill you", "hurt", "i'll hurt you", "destroy", "i'll destroy you",
"attack", "i will attack", "harm", "i'll harm", "beat you", "shoot", "i'll shoot", "burn", "bomb",
"stab", "punch", "explode", "assault", "slap", "murder", "eliminate", "wipe you out", "i'll end you",
"take you down", "ruin you", "end you", "threaten", "you're dead", "watch your back",
"break your bones", "you won't survive", "suffer", "you will regret", "put you in pain",
```

"i'm coming for you", "you are next", "final warning", "your days are numbered", "i swear i'll",
"you'll pay", "can't protect you", "i'm gonna get you", "i will hunt you down"] # keep your long threat list here
suggestion_keywords = ["you should", "i suggest", "i recommend", "how about", "what if", "maybe try", "i propose",
"you could", "have you considered", "it would help if", "consider doing", "i think you should",
"perhaps you can", "try to", "it might be better", "my advice is", "think about", "i encourage you to",
"i believe you should", "one way is", "you ought to", "we could", "you may want to",
"you need to", "a better way is", "would it help", "take a look at", "a good idea is to",
"why not try", "i'd say", "another option is","you should", "i suggest", "i recommend", "you could try", "how about", "maybe try",
"consider doing", "have you thought about", "it would help if", "you ought to", "why not",
"let's try", "i think it's better to", "would be good if", "can you try", "a better way could be",
"it might be helpful", "i advise", "you can try", "don't you think", "better if", "just a suggestion",
"perhaps you should", "my advice is", "in my opinion", "i propose", "would suggest", "make sure to",
"how would it be if", "how about trying", "i feel like you should", "i'd say", "i'd advise"] # keep your long suggestion list here
encouragement_keywords = ["great job", "well done", "keep it up", "nice work", "excellent", "awesome", "you got this",
"proud of you", "fantastic", "brilliant", "amazing", "you're doing great", "you're awesome",
"keep going", "never give up", "don't stop", "believe in yourself", "go for it",
"that's the spirit", "strong work", "keep pushing", "you inspire me", "don't give up",
"way to go", "you nailed it", "keep shining", "love your energy", "well played",
"you're the best", "excellent effort", "you did it", "keep trying", "never stop improving",
"you've improved a lot", "this is amazing", "you've got talent", "great effort",
"you're so talented", "awesome stuff", "just wow", "salute to you", "this made my day",
"bless you", "i admire this", "you're strong","great job", "keep it up", "you're amazing", "well done", "proud of you", "you got this",
"keep going", "you can do it", "don't give up", "excellent work", "brilliant effort",
"you're strong", "never give up", "amazing job", "you're doing great", "stay strong",
"you're improving", "nice progress", "fantastic work", "incredible", "awesome effort",
"i believe in you", "way to go", "keep trying", "you're capable", "keep pushing",
"good luck", "don't stop", "be confident", "you're the best", "you've got potential",
"you are inspiring", "you're on the right path", "so proud of you", "keep shining"] # keep your long encouragement list here
insult_keywords = ["stupid", "idiot", "dumb", "moron", "loser", "worthless", "trash", "shut up", "jerk",
"clown", "fat", "lazy", "retard", "dumbass", "you suck", "garbage", "nonsense", "disgrace",
"pathetic", "get lost", "worst", "ugly", "freak", "brainless", "noob", "failure", "hopeless",
"i hate you", "you're annoying", "fool", "useless", "go to hell", "shame on you", "filthy",
"repulsive", "creep", "weirdo", "your face", "gross", "idiotic", "freaking idiot",
"pain in the neck", "waste of space", "disgusting", "annoying person", "mental",
"psychopath", "lowlife", "a joke", "childish", "trash talker", "pest", "ugliest",

"shut your mouth", "go away", "who even are you", "you're cringe","stupid", "idiot", "dumb", "you're dumb", "fool", "loser", "ugly", "shut up", "moron", "worthless",
"trash", "pathetic", "retard", "disgrace", "you suck", "worst ever", "no one likes you", "annoying",
"fat", "lazy", "disgusting", "waste of space", "what a joke", "you're horrible", "embarrassment",
"clown", "failure", "nonsense", "noob", "jerk", "you're nothing", "you make me sick", "freak",
"you don't matter", "go to hell", "ugliest", "most useless", "you're a pest", "why do you exist",
"can't stand you", "you're dumb af", "so cringe", "just shut up", "who even asked you"] # keep your long insult list here

behaviors = {
"Threat": any(phrase in comment for phrase in threat_keywords),
"Suggestion": any(phrase in comment for phrase in suggestion_keywords),
"Encouragement": any(phrase in comment for phrase in encouragement_keywords),
"Insult": any(phrase in comment for phrase in insult_keywords)
}

return behaviors

 # Function to check and prevent abusive comment
def check_comment():
user_input = entry.get()
if user_input.strip() == "":
messagebox.showwarning("Input Error", "Please enter a comment.")
return

prediction = model.predict([user_input])[0]
confidence = model.predict_proba([user_input])[0].max()

if prediction == 1:
result_label.config(text=f"⊘ Blocked: Abusive Detected\nConfidence: {confidence:.2f}", fg="red")
behavior_label.config(text="")
messagebox.showerror("Blocked", "Abusive language is not allowed. Please revise your comment.")
save_abusive_comment(user_input)
add_to_history(user_input, "Blocked")
entry.delete(0, tk.END)
else:
behaviors = detect_behavior(user_input)
behavior_text = "\n".join([f"{key}: {value}" for key, value in behaviors.items()])
result_label.config(text=f"✓ Accepted: Not Abusive\nConfidence: {confidence:.2f}", fg="green")
behavior_label.config(text=f"Type of Comment:\n{behavior_text}", fg="white")
messagebox.showinfo("Success", "Your comment is safe and accepted.")
add_to_history(user_input, f"Accepted\n{behavior_text}")
entry.delete(0, tk.END)

```
# Clear fields
def clear_all():
entry.delete(0, tk.END)
result_label.config(text="")
behavior_label.config(text="")
history_box.delete(1.0, tk.END)

# Update comment history
def add_to_history(comment, status):
history_box.insert(tk.END, f"{status}: {comment}\n")
history_box.yview(tk.END)

# GUI Setup
root = tk.Tk()
root.title("Abusive Comment Detection and Prevention System")
root.configure(bg="#1e1e2f")
root.state("zoomed")

# Title
title_label = tk.Label(root, text="ABUSIVE COMMENT DETECTION & PREVENTION", font=("Arial Black", 28), fg="#00ffff", bg="#1e1e2f")
title_label.pack(pady=40, anchor="center")

# Input frame
main_frame = tk.Frame(root, bg="#1e1e2f")
main_frame.pack(fill="both", expand=True, padx=40)

left_frame = tk.Frame(main_frame, bg="#1e1e2f")
left_frame.pack(side="left", fill="both", expand=True)

right_frame = tk.Frame(main_frame, bg="#1e1e2f")
right_frame.pack(side="right", fill="y", padx=40)

# Input Label
tk.Label(left_frame, text="Enter a comment:", font=("Arial", 16), bg="#1e1e2f", fg="white").pack(anchor="w")

# Input Entry
entry = tk.Entry(left_frame, width=100, font=("Arial", 14), bg="#ffffff", fg="black", relief="solid", bd=2)
entry.pack(pady=10, anchor="w")

# Buttons
```

```
btn_frame = tk.Frame(left_frame, bg="#1e1e2f")
btn_frame.pack(pady=10, anchor="w")

tk.Button(btn_frame, text="Check & Submit Comment", command=check_comment,
bg="#007acc", fg="white", font=("Arial", 14), relief="raised", padx=10).grid(row=0, column=0, padx=5)

tk.Button(btn_frame, text="Clear", command=clear_all,
bg="#ff5555", fg="white", font=("Arial", 14), relief="raised", padx=10).grid(row=0, column=1, padx=5)

# Result Label
result_label = tk.Label(left_frame, text="", font=("Arial", 18, "bold"), bg="#1e1e2f")
result_label.pack(pady=10)

# History Section
tk.Label(left_frame, text="📝 Comment History", font=("Arial", 16, "bold"), fg="#00ffff",
bg="#1e1e2f").pack(anchor="w")
history_box = scrolledtext.ScrolledText(left_frame, width=100, height=10, font=("Arial", 12),
bg="#252536", fg="white")
history_box.pack(pady=10)

# Right-side Behavior Label
behavior_label = tk.Label(right_frame, text="", font=("Arial", 16), bg="#1e1e2f", fg="white",
justify="left", anchor="n")
behavior_label.pack(pady=20, anchor="n")

# Run the App
root.mainloop()
```

7.TESTING

7.1 TESTING METHODOLOGIES

INTRODUCTION

Testing is the process of executing a program to find errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software.

Types of Software Testing: Different Testing Types with Details

We, as testers, are aware of the various types of Software Testing like Functional Testing, Non-Functional Testing, Automation Testing, Agile Testing, and their sub-types, etc.

Each type of testing has its own features, advantages, and disadvantages as well. However, in this tutorial, we have covered mostly each and every type of software testing which we usually use in our day-to-day testing life.

Different Types of Software Testing

FUNCTIONAL TESTING METHODS

The following are the Functional Testing Methodologies:

Unit Testing.

Integration Testing. ➢ System Testing.
Acceptance Testing

1)UNIT TESTING

Unit testing is a type of software testing which is done on an individual unit or component to test its corrections. Typically, Unit testing is done by the developer at the application development phase. Each unit in unit testing can be viewed as a method, function, procedure, or

44

object. Developers often use test automation tools such as NUnit, Xunit, JUnit for the test execution.

Unit testing is important because we can find more defects at the unit test level. **For example,** there is a simple calculator application. The developer can write the unit test to check if the user can enter two numbers and get the correct sum for addition functionality.

a) White Box Testing

White box testing is a test technique in which the internal structure or code of an application is visible and accessible to the tester. In this technique, it is easy to find loopholes in the design of an application or fault in business logic. Statement coverage and decision coverage/branch coverage are examples of white box test techniques.

b) Gorilla Testing

Gorilla testing is a test technique in which the tester and/or developer test the module of the application thoroughly in all aspects. Gorilla testing is done to check how robust your application is.**For example,** the tester is testing the pet insurance company's website, which provides the service of buying an insurance policy, tag for the pet, Lifetime membership. The tester

can focus on any one module, let's say, the insurance policy module, and test it thoroughly with positive and negative test scenarios.

2)Integration Testing

Integration testing is a type of software testing where two or more modules of an application are logically grouped together and tested as a whole. The focus of this type of testing is to find the defect on interface, communication, and data flow among modules. Top-down or Bottom-up approach is used while integrating modules into the whole system.

This type of testing is done on integrating modules of a system or between systems. **For example,** a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems. Integration testing should be done while integrating of airline website and payment processing system.

a) Gray box testing

As the name suggests, gray box testing is a combination of white-box testing and blackbox testing. Testers have partial knowledge of the internal structure or code of an application.

3)System Testing

System testing is types of testing where tester evaluates the whole system against the specified requirements.

a) End to End Testing

It involves testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.**For example,** a tester is testing a pet insurance website. End to End testing involves testing of buying an insurance policy, LPM, tag, adding another pet, updating credit card

information on users' accounts, updating user address information, receiving order confirmation emails and policy documents.

b) Black Box Testing

Blackbox testing is a software testing technique in which testing is performed without knowing the internal structure, design, or code of a system under test. Testers should focus only on the input and output of test objects. Detailed information about the advantages, disadvantages, and types of Black Box testing can be found here.

c) Smoke Testing

Smoke testing is performed to verify that basic and critical functionality of the system under test is working fine at a very high level. Whenever a new build is provided by the development team, then the Software Testing team validates the build and ensures that no major issue exists. The testing team will ensure that the build is stable, and a detailed level of testing will be carried out further. **For example,** tester is testing pet insurance website. Buying an insurance policy, adding another pet, providing quotes are all basic and critical functionality of the application. Smoke testing for this website verifies that all these functionalities are working fine before doing any in-depth testing.

d) Sanity Testing

Sanity testing is performed on a system to verify that newly added functionality or bug fixes are working fine. Sanity testing is done on stable build. It is a subset of the regression test.

For example, a tester is testing a pet insurance website. There is a change in the discount for buying a policy for second pet. Then sanity testing is only performed on buying insurance policy module.

e) Happy path Testing

The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on valid and positive inputs through which the application generates the expected output.

f) Monkey Testing

Monkey Testing is carried out by a tester, assuming that if the monkey uses the application, then how random input and values will be entered by the Monkey without any knowledge or understanding of the application. The objective of Monkey Testing is to check if an application orsystem gets crashed by providing random input values/data. Monkey Testing is performed randomly, no test cases are scripted, and it is not necessary to be aware of the full functionality of the system.

4)Acceptance Testing

Acceptance testing is a type of testing where client/business/customer test the software with real time business scenarios. The client accepts the software only when all the features and functionalities work as expected. This is the last phase of testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

a) Alpha Testing

Alpha testing is a type of acceptance testing performed by the team in an organization to find as many defects as possible before releasing software to customers. **For example,** the pet insurance website is under UAT. UAT team will run real-time scenarios like buying an insurance website is under UAT. UAT team will run real-time scenarios like buying an insurance policy, buying annual membership, changing the address, ownership transfer of the pet in a same way the user uses the real website. The team can use test credit card information to process payment-related scenarios **B) Beta Testing**

Beta Testing is a type of software testing which is carried out by the clients/customers. It is performed in the **Real Environment** before releasing the product to the market for the actual end-users. Beta Testing is carried out to ensure that there are no major failures in the software or product, and it satisfies the business requirements from an end-user perspective. Beta Testing is successful when the customer accepts the software. Usually, this testing is typically done by the end-users. This is the final testing done before releasing the application for commercial purposes. Usually, the Beta version of the software or product released is limited to a certain number of users in a specific area. So, the end-user uses the software and shares the feedback with the company.

The company then takes necessary action before releasing the software worldwide.

c) Operational acceptance testing (OAT)

Operational acceptance testing of the system is performed by operations or system administration staff in the production environment. The purpose of operational acceptance testing is to make sure that the system administrators can keep the system working properly for the users in a real-time environment.

The focus of the OAT is on the following points:

- Testing of backup and restore.
- Installing, uninstalling, upgrading software.
- The recovery process in case of natural disaster.
- User management.
- Maintenance of the software.

NON-FUNCTIONAL TESTING METHODS

The following are the Non-Functional Testing Methodologies:

Security Testing.

Performance Testing.

Usability Testing.

Compatibility Testing.

1)Security Testing

It is a type of testing performed by a special team. Any hacking method can penetrate the system. Security Testing is done to check how the software, application, or website is secure from internal and/or external threats. This testing includes how much software is secure from malicious programs, viruses and how secure & strong the authorization and authentication processes. It also checks how software behaves for any hacker's attack & malicious programs and how software is maintained for data security after such a hacker attack.

a) Penetration Testing

Penetration Testing or Pen testing is the type of security testing performed as an authorized cyberattack on the system to find out the weak points of the system in terms of security Pen testing is performed by outside contractors, generally known as ethical hackers. That is why it is also known as ethical hacking. Contractors perform different operations like SQL injection,

URL manipulation, Privilege Elevation, session expiry, and provide reports to the organization.

2) Performance Testing

Performance testing is testing of an application's stability and response time by applying load. The word stability means the ability of the application to withstand in the presence of load. Response time is how quickly an application is available to users. Performance testing is done with the help of tools. Loader.IO, JMeter, LoadRunner, etc. are good tools available in the market **a) Load testing**

Load testing is testing of an application's stability and response time by applying load, which is equal to or less than the designed number of users for an application. **For example,** your application handles 100 users at a time with a response time of 3 seconds, then load testing can be done by applying a load of the maximum of 100 or less than 100 users. The goal is to verify that the application is responding within 3 seconds for all the users.49

b) Stress Testing

Stress testing is testing an application's stability and response time by applying load, which is more than the designed number of users for an application. **For example,** your application handles 1000 users at a time with a response time of 4 seconds, then stress testing can be done by applying a load of more than 1000 users. Test the application with 1100,1200,1300 users and notice the response time. The goal is to verify the stability of an application under stress.

c) Scalability Testing

Scalability testing is testing an application's stability and response time by applying load, which is more than the designed number of users for an application. **For example,** your application handles 1000 users at a time with a response time of 2 seconds, then scalability testing can be done by applying a load of more than 1000 users and gradually increasing the number of users to find out where exactly my application is crashing.

Let's say my application is giving response time as follows:

- 1000 users -2 sec
- 1400 users -2 sec
- 4000 users -3 sec
- 5000 users -45 sec
- 5150 users- crash – This is the point that needs to identify in scalability testing

d) Volume testing (flood testing)

Volume testing is testing an application's stability and response time by transferring a large volume of data to the database. Basically, it tests the capacity of the database to handle the data.

e) Endurance Testing (Soak Testing)

Endurance testing is testing an application's stability and response time by applying load continuously for a longer period to verify that the application is working fine. **For example,** car companies soak testing to verify that users can drive cars continuously for hours without any problem.

3) Usability Testing

Usability testing is testing an application from the user's perspective to check the look and feel and user-friendliness. **For example,** there is a mobile app for stock trading, and a tester is performing usability testing. Testers can check the scenario like if the mobile app is easy to operate with one hand or not, scroll bar should be vertical, background colour of the app should be black and price of and stock is displayed in red or green colour. The main idea of usability testing of this kind of app is that as soon as the user opens the app, the user should get a glance at the market.

a) Exploratory testing

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Testers use the knowledge of the business domain to test the application. Test charters are used to guide the exploratory testing.

b) Cross browser testing

Cross browser testing is testing an application on different browsers, operating systems, mobile devices to see look and feel and performance. Why do we need cross-browser testing? The answer is different users use different operating systems, different browsers, and different mobile devices. The goal of the company is to get a good user experience regardless of those devices Browser stack provides all the versions of all the browsers and all mobile devices to test the application

c) Accessibility Testing

The aim of Accessibility Testing is to determine whether the software or application is accessible for disiabled people or not.

Here, disability means deafness, colour blindness, mentally disabled, blind, old age, and other disabled groups. Various checks are performed, such as font size for visually disabled, colour and contrast for colour blindness, etc.

4) Compatibility testing

This is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment

7.2TEST CASES GENERATION

S.NO	TEST CASE ID	TEST CASE NAME	TEST CASE DESCRIPTION	STEP	EXECUTED RESULTS	ACTUAL RESULTS	TEST CASE STATUS
1	Register new user	Registering New User before Logging in.	To Login and register you need to create/register.	Click on register and enter user credentials	Enter the Registration form	View the form	pass
2	Admin login	Validate login	Verifying login page	Enter admin user name and password	Validate login	Validate login	Pass
3	Manage Users	Accept user or reject user	Either accept or reject the registered users	Accept user or reject user	Accept user or reject user	Accepted or rejected	Pass
4	Login user	Validate login page	To verify login name on page	Enter the Email Id and password.	Validate user	Validate user	pass
5	View your profile	View details	Verify profile	Verify the profile	Display profile	Display Profile page	Pass
6	Depression Detection	Detecting Depression	Identifying facial features	Click on Camera icon	Activates camera to capture live video	Depression Detected	Pass

7	Feedback	Feedback submitting by user along with rating	Sharing user experience	Enter rating	Feedback submitting	Pop up display with feedback submitted	Pass
8	logout	logout	Logging out form	Click on	Successful Log out	Log out	Pass

8. OUTPUT SCREENSHOTS

JUPYTER NOTEBOOK

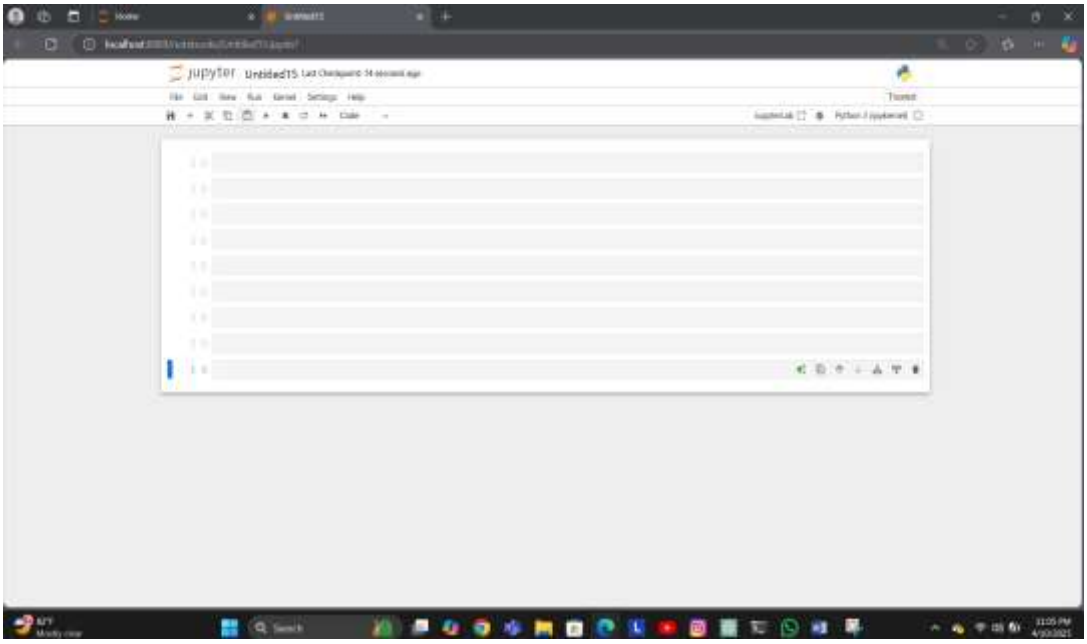


Fig 8.1

CODE DEPLOYMENT

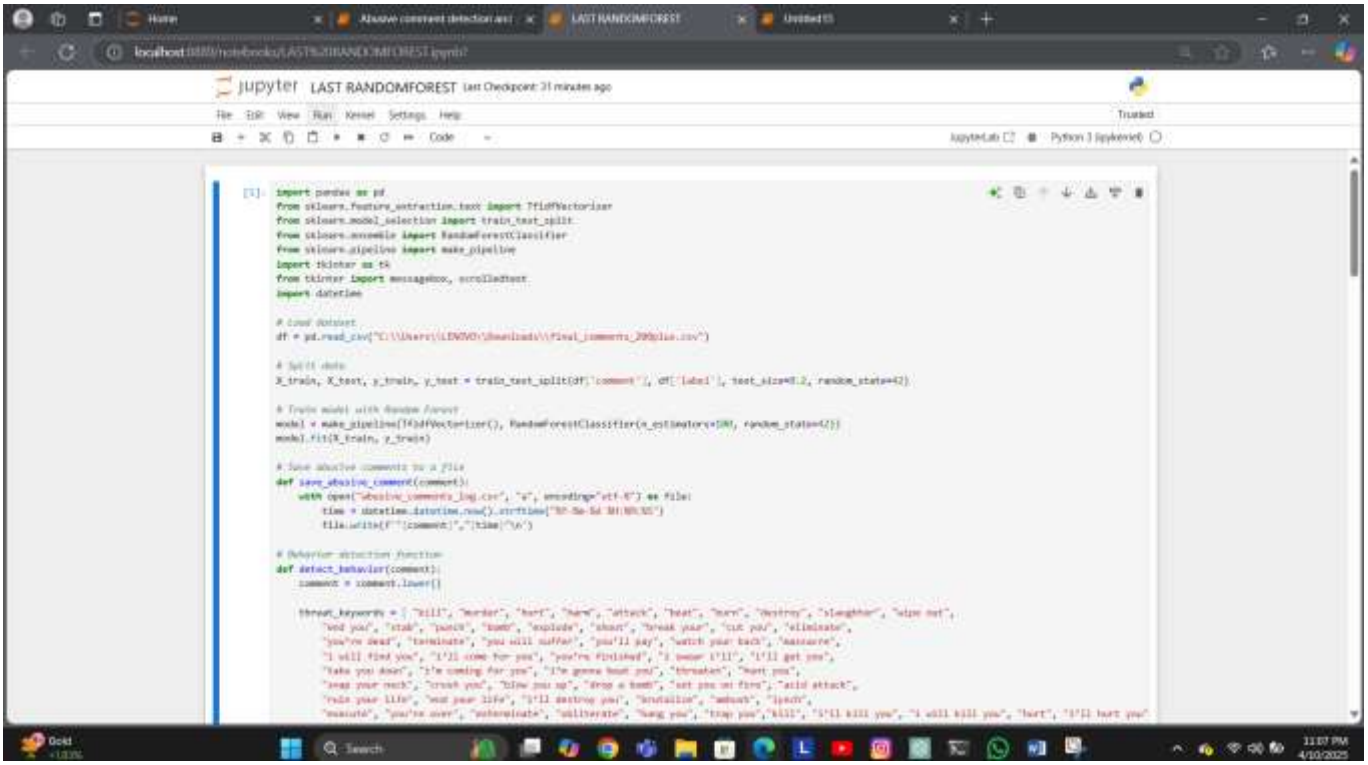


Fig 8.2

AFTER CODE EXECUTION

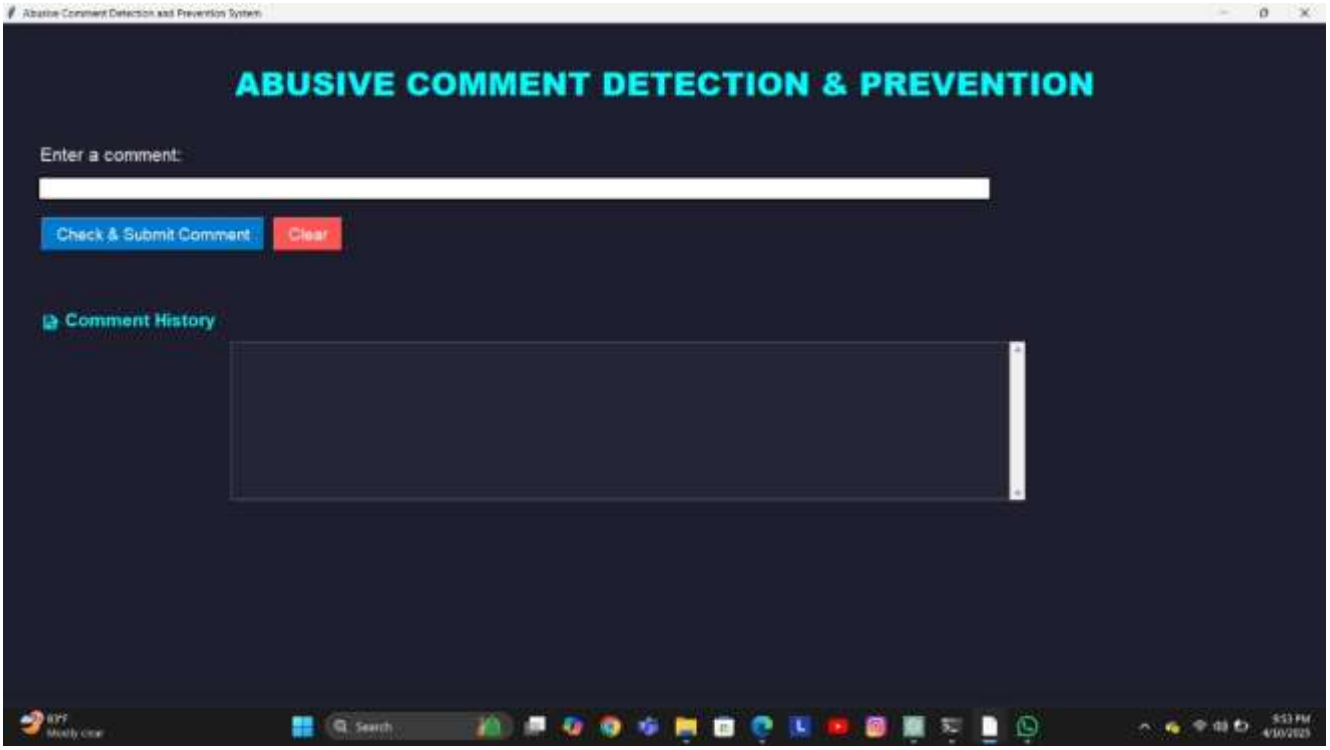


Fig 8.3

Alert to Enter input

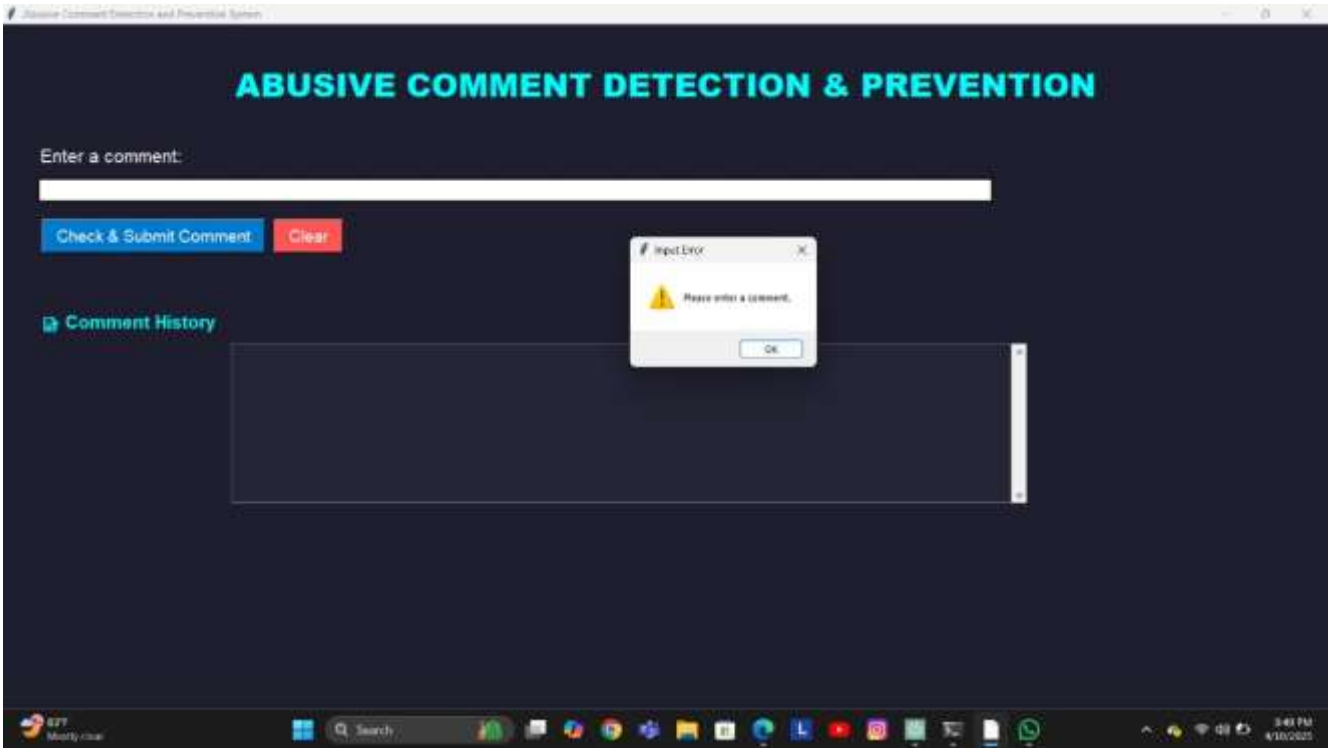


Fig 8.4

Abusive comment detection and prevention

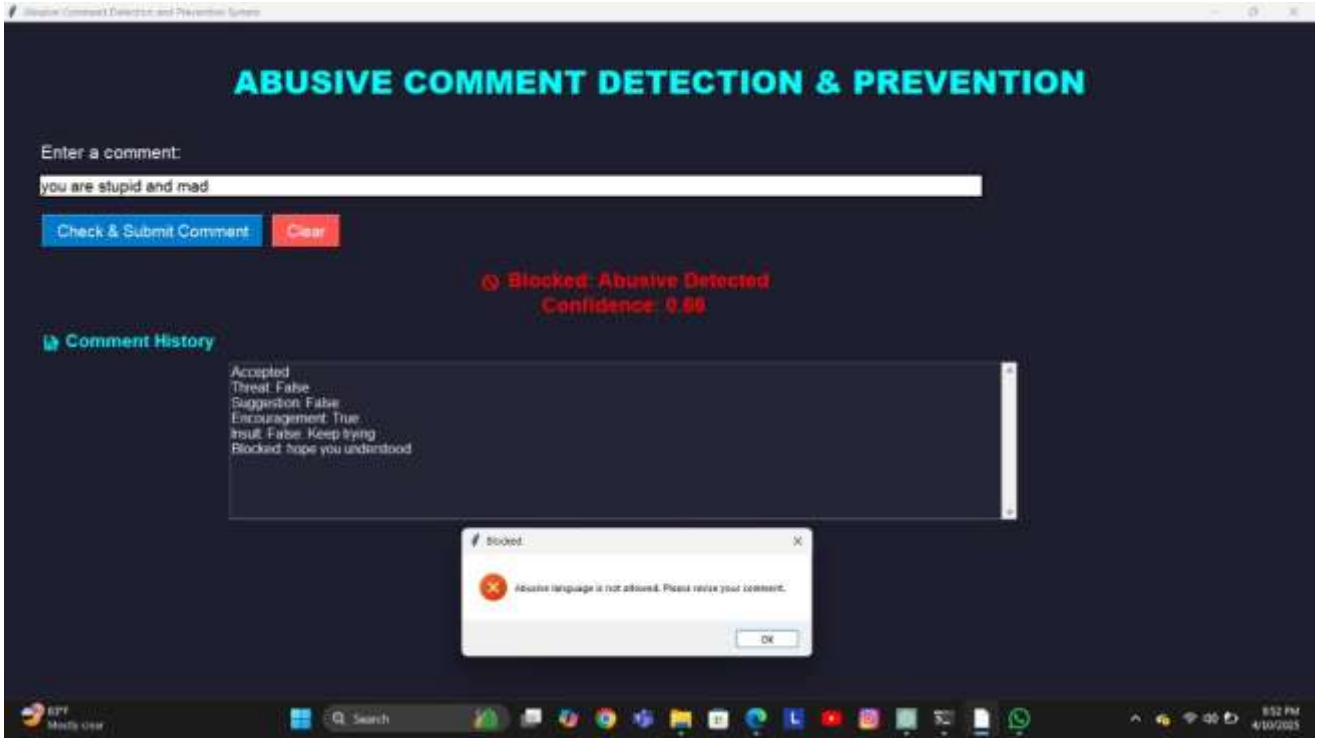


Fig:8.5

Comment Classification

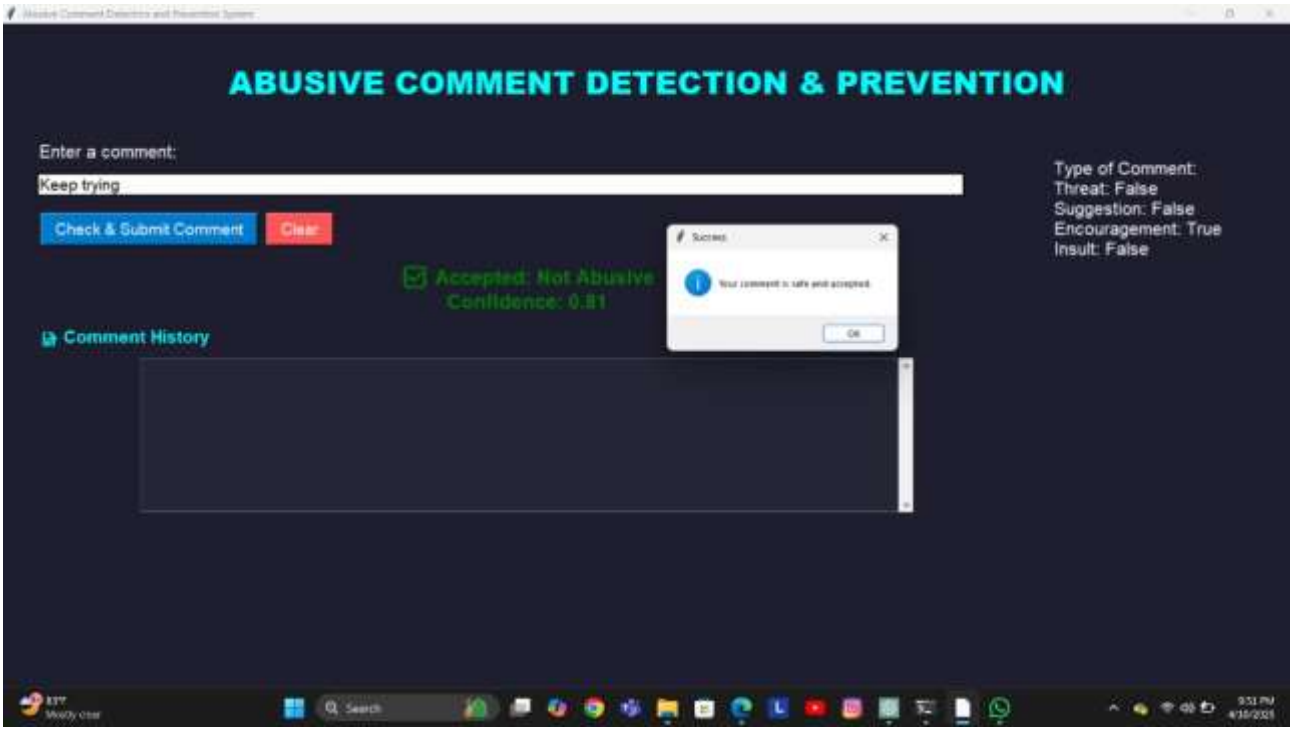


Fig 8.6

Dataset View

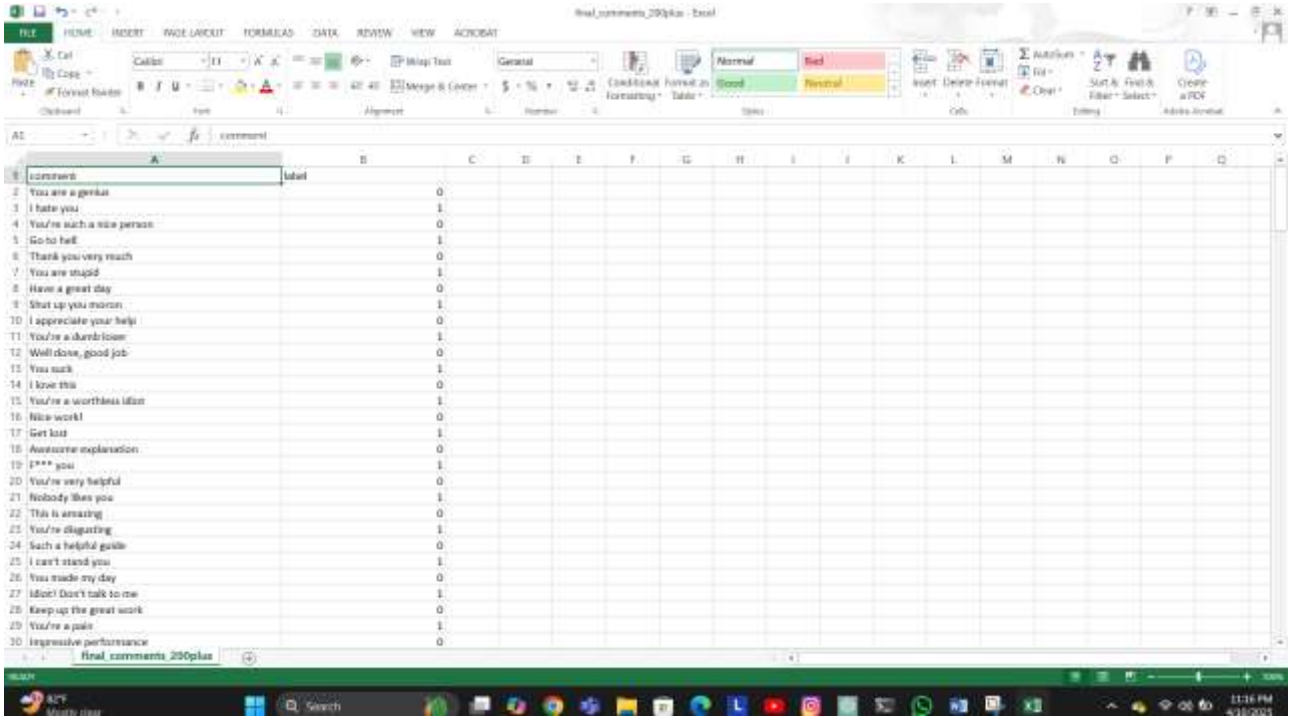


Fig 8.18

9.CONCLUSION AND FUTURE RECOMMENDATIONS

9.1 CONCLUSION

In conclusion, the project "ABUSIVE COMMENT DETECTION AND PREVENTION" has significantly advanced the development of automated systems capable of identifying and mitigating harmful content across various online platforms. By leveraging machine learning and deep learning techniques, such as Support Vector Machines (SVM), Naive Bayes, Convolutional Neural Networks (CNNs), and Long Short-Term Memory networks (LSTMs), researchers have achieved notable success in classifying abusive comments. For instance, a study employing combined with LSTM achieved an accuracy of 76.89%, highlighting the effectiveness of contextual embeddings in understanding nuanced language patterns.

However, the detection of abusive content remains a complex challenge, influenced by factors such as linguistic diversity, cultural context, and the evolving nature of online discourse. The scarcity of large, annotated datasets in low-resource languages poses additional hurdles, as exemplified by the limited availability of datasets for languages like Tamil. To address these challenges, collaborative efforts are essential to curate comprehensive datasets and develop models that generalize well across different languages and contexts.

Moreover, the dynamic and context-dependent nature of abusive language necessitates continuous model updates and refinements. Incorporating user feedback and contextual information, such as comment threads and user history, can enhance detection accuracy. Ethical considerations also play a crucial role, ensuring that detection systems respect user privacy and freedom of expression while effectively filtering harmful content. Ongoing research and development in this field are vital to create safer and more inclusive online environments, fostering positive user interactions and mitigating the impact of abusive language.

9.2 FUTURE RECOMMONDATIONS :

Advancements in abusive comment detection and prevention require a comprehensive approach that addresses current challenges and leverages emerging opportunities. Future efforts should focus on enhancing contextual understanding by developing models capable of interpreting sarcasm, irony, and cultural nuances, thereby improving detection accuracy.

Reducing biases in detection systems is crucial; implementing strategies such as debiased word embeddings and diverse data augmentation can promote fairness and equity. Addressing ethical and human rights concerns necessitates adopting frameworks that balance moderation needs with the protection of individual freedoms, ensuring that detection technologies do not inadvertently suppress underrepresented voices.

Developing explainable models will enhance transparency, allowing users to understand and trust the decisions made by these systems. Improving multilingual and cross-cultural detection involves collecting diverse datasets and creating models that comprehend various linguistic and cultural contexts, ensuring global applicability.

Balancing precision and recall is vital to minimize false positives and negatives, ensuring that detection systems are both effective and user-friendly. Finally, ensuring real-time processing efficiency is essential for deployment in dynamic online environments, requiring optimization of algorithms to reduce computational load and latency. By integrating these recommendations, future systems can achieve more accurate, fair, and ethical detection of abusive content, fostering safer and more inclusive online communities.

REFERENCES

1. Hajime Watanabe, Mondher Bouazizi , and Tomoaki Ohtsuki "Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection".February 15, 2018.
2. Muhammad Okky Ibrohim, Indra Budi, A Dataset and Preliminaries Study for "Abusive Language Detection in Indonesian Social Media",Procedia Computer Science 135 (2018) 222–229.
3. Jezabel Molina-Gil, José A. Concepción Sánchez and Pino Caballero- Gil,"Harassment Detection Using Machine Learning and Fuzzy Logic Techniques" 20 November 2019.
4. Navoneel Chakrabarty, "A Machine Learning Approach to Comment Toxicity Classification",2020.
5. Rehanulla Khan,Nasir Ahmad,"Random Forests and Decision Trees",September,2012.
6. Prakhyat Rai, Shamantha Rai, Sweekriti Shetty, "Sentiment Analysis Using Machine Learning Classifiers: Evaluation of Performance",pp.IEEE 2019.
7. J.I.Sheeba, S. Pradeep Devaneyan, Revathy Cadiravane, "Identification and Classification of Cyberbully Incidents using Bystander Intervention Model" International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277- 3878, Volume-8 Issue- 2S4, July 2019.
8. Dinesh Kannan,Rajkumar Murukeshan ,"Online Abuse Detection",2018.
9. Aishwarya Ganesan,"Offensive language detection using AI technique ".April 2018.