PREDICTION OF CYBER TOXIC COMMENTS ON SOCIAL MEDIA USING MACHINE LEARNING

An Industrial Oriented Mini Project report submitted in partial fulfillment of the

Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

CHINTHAKINDHI SOWMYA	**20VD1A0541**
KHAMMAMPATI ANIL KUMAR	**20VD1A0506**
CHITTIPROLU KUMARASWAMY	**20VD1A0517**
BANDILA SANGEETHA	**20VD1A0557**

Under the Guidance of

N. VAMSHI KRISHNA
(Asst. professor of CSE)
Department of Computer Science and Engineering.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING MANTHANI

Pannur (Vil), Ramagiri (Mdl), Peddapally-505212, Telangana (India).

2023-2024

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING MANTHANI

Pannur (Vil), Ramagiri (Mdl), Peddapally-505212, Telangana (India).

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION BY THE CANDIDATE

We,

CHINTHAKINDHI SOWMYA	**20VD1A0541**
KHAMMAMPATI ANIL KUMAR	**20VD1A0506**
CHITTIPROLU KUMARASWAMY	**20VD1A0517**
BANDILA SANGEETHA	**20VD1A0557**

hereby declare that the industrial oriented mini project report entitled ***PREDICTION OF CYBER TOXIC COMMENTS ON SOCIAL MEDIA USING MACHINE LEARNING*** under the guidance of **N. VAMSHI KRISHNA,** Asst. professor, Department of Computer Science and Engineering, JNTUH University College of Engineering Manthani submitted in partial fulfilment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been reproduced or copied from any source. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

CHINTHAKINDHI SOWMYA	**20VD1A0541**
KHAMMAMPATI ANIL KUMAR	**20VD1A0506**
CHITTIPROLU KUMARASWAMY	**20VD1A0517**
BANDILA SANGEETHA	**20VD1A0557**

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

UNIVERSITY COLLEGE OF ENGINEERING MANTHANI

Pannur (Vil), Ramagiri (Mdl), Peddapally-505212, Telangana (India).

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the industrial oriented mini project report entitled ***PREDICTION OF CYBER TOXIC COMMENTS ON SOCIAL MEDIA USING MACHINE LEARNING*** is a bonafide work carried out by

CHINTHAKINDHI SOWMYA	**20VD1A0541**
KHAMMAMPATI ANIL KUMAR	**20VD1A0506**
CHITTIPROLU KUMARASWAMY	**20VD1A0517**
BANDILA SANGEETHA	**20VD1A0557**

in partial fulfillment of the requirements for the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** by the **Jawaharlal Nehru Technological University Hyderabad University College of Engineering, Manthani** during the academic year 2023-24.

The results of investigation enclosed in this report have been verified and found satisfactory. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

INTERNAL GUIDE
Mr. N Vamshi Krishna.

HEAD OF THE DEPARTMENT

EXTERNAL EXAMINER
DATE:

ACKNOWLEDGMENT

We feel honored and privileged to place our warm salutation to our **Principal Dr. CH. SRIDHAR REDDY, Professor of Mechanical Engineering, JNTUH University College of Engineering Manthani,** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We pay our deep sense of gratitude to **N. VAMSHI KRISHNA, Asst. Professor of Computer Science and Engineering, JNTUH University College of Engineering Manthani,** to encourage us to the highest peak and to provide us the opportunity to prepare the project. We are extremely grateful to his valuable suggestions and unflinching co- operation throughout project work.

We are immensely obliged to other faculty members of Department of CSE for their kind co-operation.

We wish to convey our thanks to one and all those who have extended their helping hands directly and indirectly in completion of our project.

Last, but not least, our parents are also an important inspiration for us. So, with due regards, we express our gratitude to them.

CHINTHAKINDHI SOWMYA	**20VD1A0541**
KHAMMAMPATI ANIL KUMAR	**20VD1A0506**
CHITTIPROLU KUMARASWAMY	**20VD1A0517**
BANDILA SANGEETHA	**20VD1A0557**

ABSTRACT

Toxic comments are disrespectful, abusive, or unreasonable online comments that usually make other users leave a discussion. The danger of online bullying and harassment affects the free flow of thoughts by restricting the dissenting opinions of people. Sites struggle to promote discussions effectively, leading many communities to limit or close down user comments altogether. This paper will systematically examine the extent of online harassment and classify the content into labels to examine the toxicity as correctly as possible. Here, we will use six machine learning algorithms and apply them to our data to solve the problem of text classification and to identify the best machine learning algorithm based on our evaluation metrics for toxic comments classification. We will aim at examining the toxicity with high accuracy to limit down its adverse effects which will be an incentive for organizations to take the necessary steps.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION

The exponential development of computer science and technology provides us with one of the greatest innovations of the "Internet" of the 21st century, where one person can communicate to another worldwide with the help of a mere smartphone and internet. In the initial days of the internet, people used to communicate with each other through Email only and it was filled with spam emails. In those days, it was a big task to classify the emails as positive or negative i.e., spam or not spam. As time flows, communication, and flow of data over the internet got changed drastically, especially after the appearance of social media sites. With the advancement of social media, it becomes highly important to classify the content into positive and negative terms, to prevent any form of harm to society and to control antisocial behavior of people. In recent times there have many instances where authorities arrest people due to their harmful and toxic social media contents [1]. For example, one 28-year-old man was arrested in Bengal for posting an abusive comment against Mamata Banerjee on Facebook and one man from Indonesia was arrested for insulting the police of Indonesia on Facebook. Thus, there is an alarming situation and it is the need of the hour to detect such content before they got published because these negative contents are creating the internet an unsafe place and affecting people adversely. Suppose there is a comment on social media “Nonsense? Kiss off, geek. What I said is true”, it can be easily identified that the words like Nonsense and Kiss off are negative and thus this comment is toxic. But to mine the toxicity technically this comment needs to go through a particular procedure and then classification technique will be applied on it to verify the precision of the obtained result. Different machine learning algorithms will be used in the classification of toxic comments on the Data set of Kaggle.com. This paper includes six machine learning techniques i.e., logistic regression, random forest, SVM classifier, naive bayes, decision tree, and KNN classification to solve the problem of text classification. So, we will apply all the six machine learning algorithms on the given data set and calculate and compare their accuracy, log loss, and hamming loss.

1.2 PROBLEM STATEMENT

Develop a machine learning system to predict and identify instances of cyber-toxicity

in social media content. This system should analyze user-generated text data and classify it as either cyber-toxic or non-toxic, contributing to the creation of safer online environments and proactive moderation strategies. The aim is to deploy a real-time solution that assists in monitoring and mitigating the harmful impact of cyber-toxicity on social media platforms.

LITERATURE SURVEY

2. LITERATURE SURVEY

A huge amount of data is released daily through social media sites. This huge amount of data is affecting the quality of human life significantly, but unfortunately due to the presence of toxicity that is there on the internet, it is negatively affecting the lives of humans. Due to this negativity, there is a lack of healthy discussion on social media sites since toxic comments are restricting people to express themselves and to have dissenting opinions. So, it is the need of the hour to detect and restrict the antisocial behavior over the online discussion forums. Although, there were efforts in the past to increase the online safety by site moderation through crowd-sourcing schemes and comment denouncing, in most cases these techniques fail to detect the toxicity. So, we have to find a potential technique that can detect the online toxicity of user content effectively. As Computer works on binary data and in real-world, we have data in various other forms i.e., images or text. Therefore, we have to convert the data of the real world into binary form for proper processing through the computer. In this paper, we will use this converted data and apply Machine learning techniques to classify online comments. Text classification can be easily applied on given data set and set of labels by applying the data on a function, that will assign a value to each data value of data set. In this context, Wulczyn et al. research introduced a technique that incorporates crowdsourcing and machine learning to evaluate on-scale personal attacks. Recently, a project called perspective was introduced by Google and Jigsaw, to detect the online toxicity, threats, and offensive content with the help of machine learning algorithms. In another approach, Convolutional Neural Networks (CNN) was used in text classification over online content, without any knowledge of syntactic or semantic language. In the approach used by Y. Chen et al., introduced a combination of a parser and lexical feature to detect the toxic language in YouTube comments to protect adolescents. In the approach used by Sulked et al., Online comments are classified with the help of machine learning algorithms. So, lots of work has already been done to detect and classify online toxic comments. In our research paper, we will learn from the already published work and use machine learning algorithms to detect and classify online toxic comments with better accuracy.

2.MOTIVATION

The main motivation behind predicting cyber toxic comments on social media using machine learning is motivated by a commitment to user well-being, community building, brand reputation, and legal compliance. It aligns with the broader goal of fostering a positive, engaging, and inclusive digital environment for users around the world.

2.1 OBJECTIVE

To develop a machine learning model that can accurately identify and classify toxic comments in online content, with the goal of creating a safer and more inclusive online environment by automatically flagging and managing harmful or offensive comments.

2.2 HISTORY OF MACHINE LEARNING

Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively.

2.3 APPLICATION

2.3.1 Influencer and Celebrity Protection:

Protect influencers, celebrities, and public figures from online harassment by predicting and mitigating toxic comments directed at them, preserving their mental well-being and public image.

2.3.2 Social Platform Compliance:

Ensure compliance with community standards, policies, and legal regulations by predicting and addressing cyber toxic comments, reducing the risk of legal consequences and fostering responsible platform usage.

2.3.3 Preventing Cyberbullying:

Predict and intervene in instances of cyberbullying, helping to create a supportive online environment and protect users.

2.4.4 Enhancing User Experience:

Improve the overall user experience on social platforms by filtering out toxic comments, fostering a more positive and engaging community environment.

2.4.5 Promoting Free Speech and Debate:

Facilitate healthy discussions by minimizing the impact of toxic comments, encouraging diverse opinions, and ensuring a safe space for constructive debates.

2.4.6 Algorithmic Moderation:

Implementing machine learning algorithms to automate comment moderation processes, reducing the burden on human moderators and ensuring a more efficient content review system.

2.4.7 Safeguarding Mental Health:

Contribute to mental health awareness by curbing online harassment, thereby reducing stress, anxiety, and emotional distress caused by exposure to toxic content.

2.4.8 Collaboration with Law Enforcement:

Support law enforcement agencies by providing predictive models to identify potential threats or illegal activities in online comments, aiding in proactive measures against cybercrime.

SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

A System Analysis is a method for identifying and solving problems that looks at each component in the overall system for the purpose of achieving specific task.

3.1 EXISTING SYSTEM

The existing system for classifying toxic comments using machine learning might involve techniques like natural language processing (NPL) and supervised learning algorithms. This system would likely require labelled data, where comments are categorized as toxic and non-toxic.

3.2 PROPOSED SOLUTION

The proposed system could potentially improve upon the existing one by incorporating more advanced NLP techniques, utilizing larger and more diverse data sets, and exploring newer machine-learning algorithms. It might also consider techniques like transfer learning using pre-trained language models, which have shown promising results in various NLP tasks.

3.3 METHODOLOGY OF PROPOSED SYSTEM:

3.3.1 Data Collection: Gather a diverse dataset of social media comments that includes examples of both toxic and non-toxic comments. Ensure the dataset is representative of the platform's user base and includes various topics and contexts.

3.3.2 Data Preprocessing: Clean and preprocess the data, including tasks like, Removing irrelevant information (e.g., timestamps, usernames). Handling missing data. Tokenization, stemming, and lemmatization for text data. Balancing the dataset if there's a class imbalance.

3.3.3 Model Selection: Choose appropriate machine learning models. Common choices include: Logistic regression, random forest, SVM classifier, naive Bayes, decision tree, and KNN classification.

3.3.4 Model Evaluation and Optimization:

Performance Metrics: Employ metrics such as accuracy, precision, recall, F1-score, ROC-AUC to evaluate model performance.

Hyperparameter Tuning: Optimize model performance through techniques like grid search, random search, or Bayesian optimization to fine-tune model parameters.

3.3.5 Validation and Generalization: Validate models using techniques like k-fold cross-validation to ensure robustness and generalizability. Compare models based on performance metrics to select the best-performing model(s).

3.3.6 Testing and Deployment: Evaluate the selected model(s) on a separate test dataset to assess real-world performance and validate against unseen data. Deploy the most effective model(s) for toxic comment classification on the targeted platform, ensuring seamless integration and real-time prediction capabilities.

3.3.7 Monitoring and Iteration: Continuously monitor model performance on live data and incorporate feedback to improve accuracy and address any emerging issues. Implement periodic updates or retraining cycles to adapt to evolving trends, language usage, and user behaviour.

SYSTEM REQUIREMENT ANALYSIS

4. SOFTWARE REQUIREMENT ANALYSIS

Software Requirements Specification plays an important role in creating quality software solutions. Specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation. Requirements may be specifiedin a variety of ways. However, there are some guidelines worth following:

- Representation format and content should be relevant to the problem.
- Information contained within the specification should be nested
- Diagrams and other notational forms should be restricted in number.

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions

4.2 NON-FUNCTIONAL REQUIREMENTS:

- **Usability**

  Usability is the case of use and learns ability of a human-made object. The object of use canbe a software application, website, book, tool, machine, process, or anything a human interacts with.A usability study may be conducted as a primary job function by a usability analyst or as a secondaryjob function by designers, technical writers, marketing personnel, and others.

- **Reliability**

  The probability that a component part, equipment, or system will satisfactorily perform its intended function under given circumstances, such as environmental conditions, limitations as to operating time, and frequently and thoroughness of maintenance for a specified period of time.

- **Supportability**

  To which the design characteristics of a stand by or support system meet the operational requirements of an organization.

- **Implementation**

  Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy.

- **Interface**

  An interface refers to a point of interaction between components and is applicable at the levelof both hardware and software. This allows a component whether a piece of hardware such as a graphics card or a piece of software such as an internet browser to function independently while using interfaces to communicate with other components via an input/output system and an associatedprotocol.

- **Legal**

  It is established by or founded upon law or official or accepted rules of or relating to jurisprudence: "legal loop hole". Having legal efficacy or force", "a sound title to the property" Relating to or characteristic of the profession of law, "the legal profession". Allowed by official rules; "a legal pass receiver".

4.3 HARDWARE REQUIREMENTS

- System: Pentium IV 2.4 GHz or higher.
- Hard Disk: Minimum 40 GB storage capacity.
- Floppy Drive: 1.44 MB.
- Monitor: 15-inch VGA colour display.
- Mouse: Logitech or compatible.
- RAM: Minimum 512 MB.

4.4 SOFTWARE REQUIREMENTS:

4.4.1 Python 3.7 or latest versions: Python serves as the primary programming language for implementing the machine learning models, data preprocessing, and system development.

4.4.2 Machine Learning Libraries:

Scikit-learn, TensorFlow, Keras: Scikit-learn provides tools for data mining, TensorFlow is used for building neural networks, and Keras operates as a high-level API on top of TensorFlow.

4.4.3 Text Processing Libraries: NLTK (Natural Language Toolkit), SpaCy, Gensim: NLTK is a platform for working with human language data, SpaCy handles natural language processing tasks, and Gensim is used for topic modeling and similarity retrieval.

4.4.4 Data Handling and Visualization Tools: Pandas, Matplotlib, Seaborn: Pandas allows data manipulation, Matplotlib creates visualizations, and Seaborn provides a statistical data visualization interface.

4.4.5 Integrated Development Environment (IDE): Anaconda, Jupyter Notebook, Spyder: Anaconda is a Python distribution that includes Jupyter Notebook and Spyder, offering tools for data science and machine learning development.

4.4.6 Other Necessary Software: Git, Docker: Git is a version control system, and Docker enables containerization for deployment purposes. Web Development Tools might also be needed for web deployment if integrating the system into a web-based platform.

SYSTEM DESIGN

5. SYSTEM DESIGN

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data the goes through that system. It is meant to satisfy specific needs and requirements through the engineering of a coherent and well-running system.

CONCEPT OF UML

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML stands for Unified Modelling Language. UML is different from the other common programming languages such as C++, Java, COBOL, etc., UML is a pictorial language used to make software blueprints. There are a number of goals for developing UML but the most important is to define some general-purpose modelling language, which are modelers can use and it also needs to be made simple to understand and use.

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 objects-oriented software and the

software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

5.1 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.

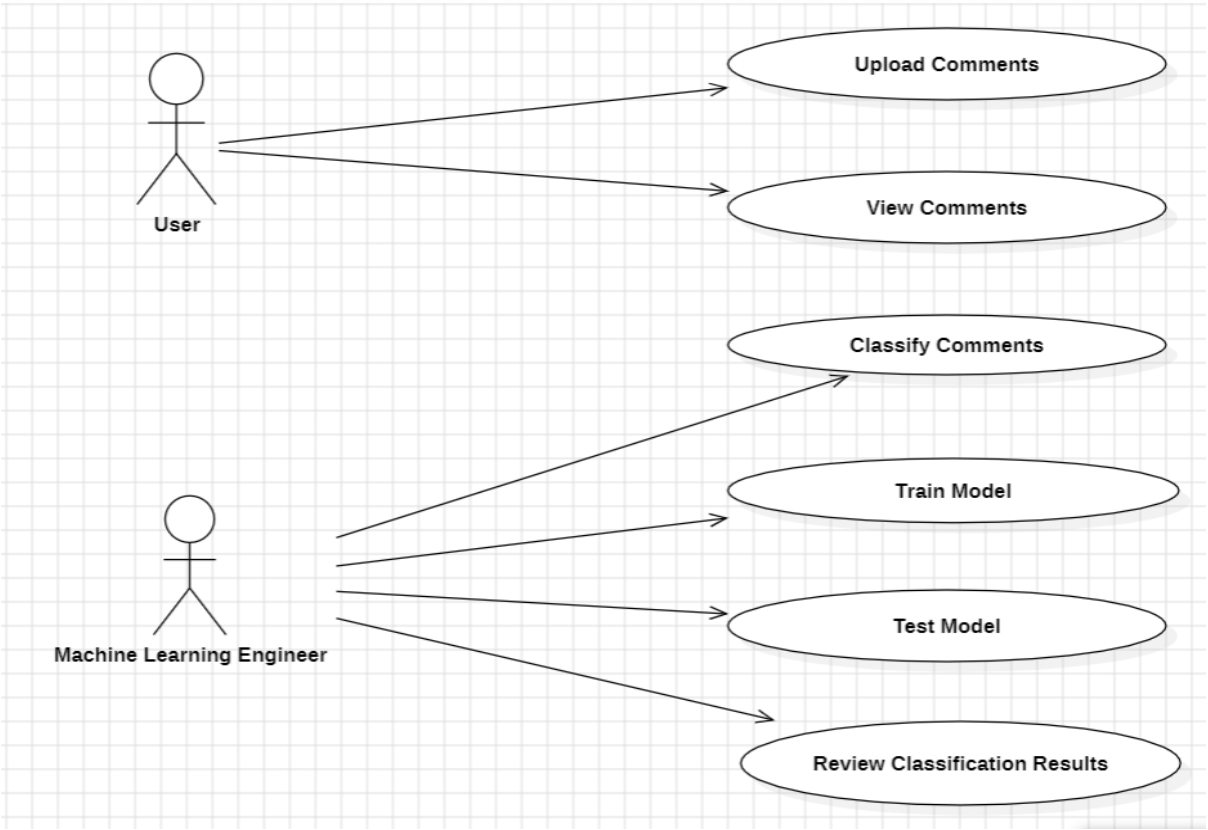

Fig 5.1.1 Use Case Diagram

5.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

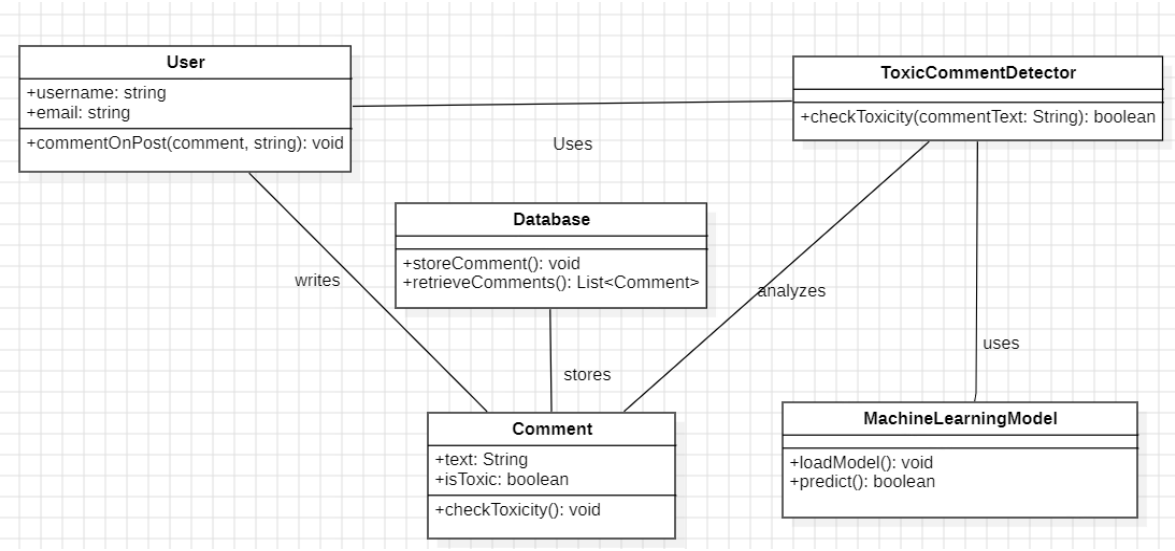

Fig 5.2.1 Class Diagram

5.3 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.

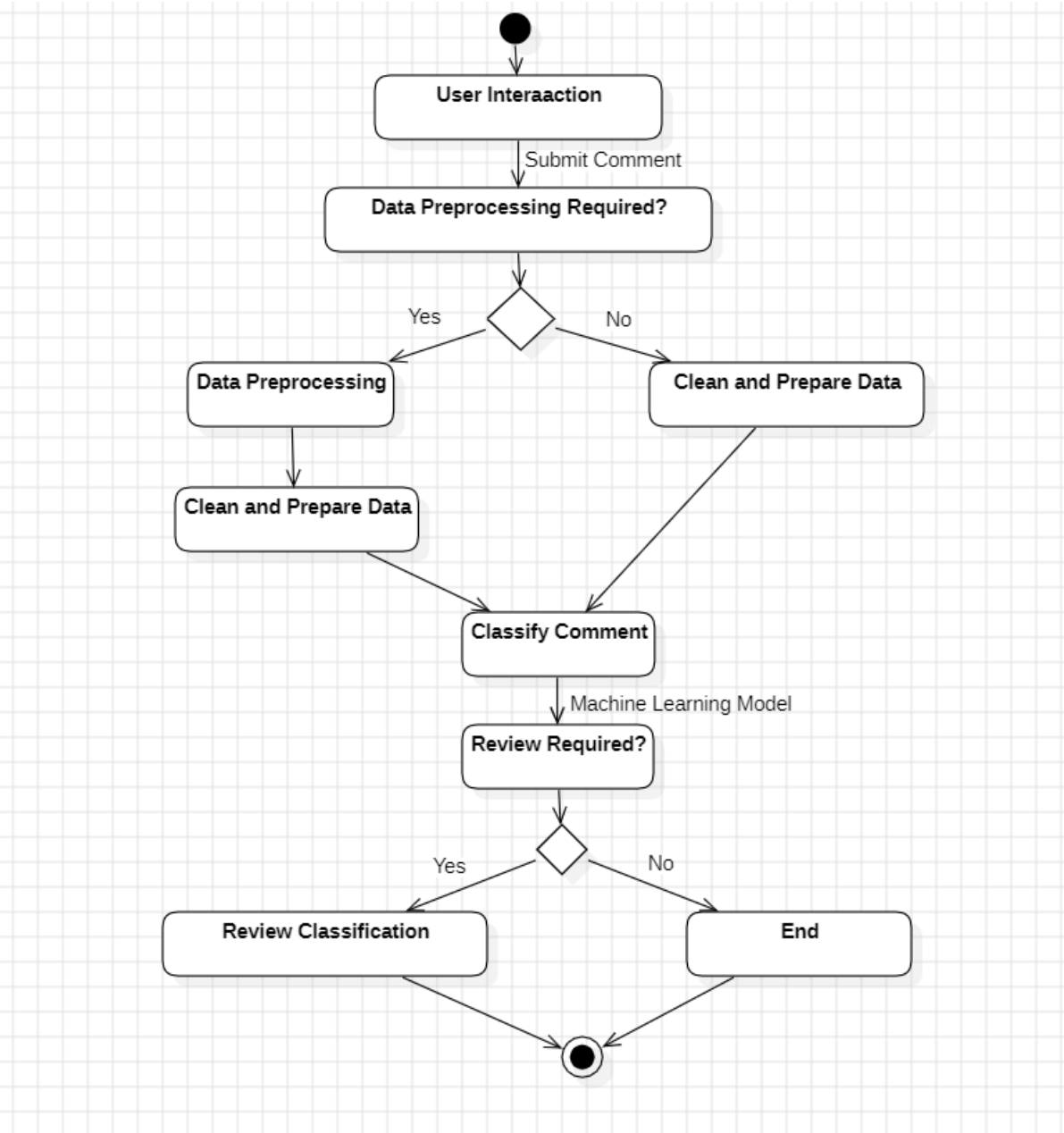

Fig 5.3.1 Activity Diagram

5.4 MODEL ARCHITECTURE

Model architecture refers to the specific design and layout of a machine learning model. It outlines the structure, layers, connections, and flow of information within the model. In the context of toxic comment classification: input layer, hidden layers, output layer, activation functions, loss function, optimization algorithm, evaluation metrics.

The architecture's complexity and the number of layers/neurons depend on various factors, including the complexity of the task, dataset size, and available computational resources. An effective architecture balances complexity with performance, aiming to capture intricate patterns in text data while avoiding overfitting or excessive computational load. Adjustments and experimentation with architectures help in achieving optimal performance for toxic comment classification.

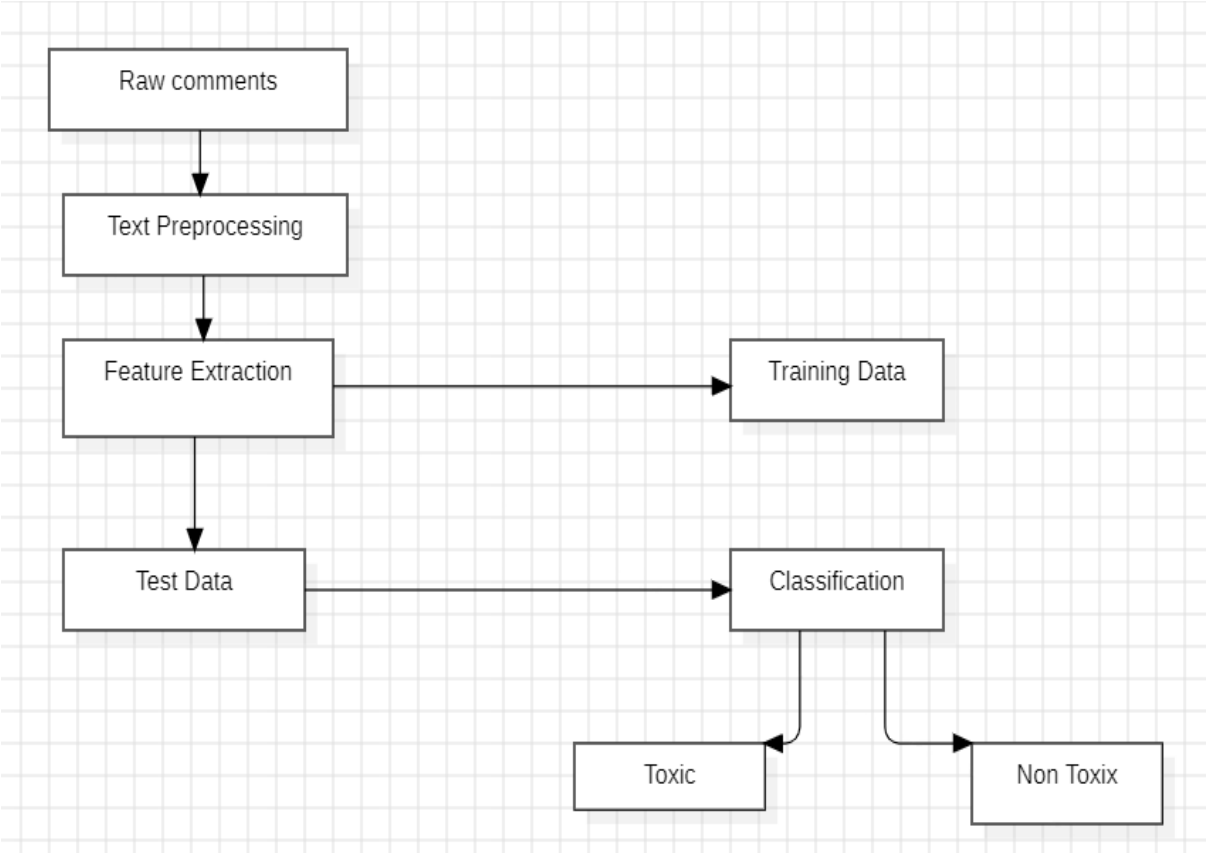

Fig 5.4.1 Model Architecture

IMPLEMENTATION

6. IMPLEMENTATION

6.1 WORKING DESCRIPTION

The system presented in the documentation addresses the crucial issue of detecting and managing toxic comments in social media platforms. It begins by assembling a diverse dataset encompassing a broad spectrum of social media comments, ensuring inclusivity across various user demographics, subjects, and contexts. This comprehensive dataset is then meticulously preprocessed, involving essential steps like data cleaning, tokenization, and feature extraction. These measures are pivotal in enhancing the data's quality and relevance for subsequent analysis.

The core of the system revolves around the thoughtful selection and rigorous training of machine learning models. Employing an array of established algorithms such as Logistic Regression, SVM, Naïve Bayes, Decision Trees, Random Forest, and KNN, the system learns to differentiate toxic from non-toxic comments. Each model undergoes meticulous evaluation, scrutinized by metrics like accuracy, precision, recall, and F1-score to ascertain its efficacy in accurately categorizing toxic content.

The architecture emphasizes text preprocessing, model selection, optimization, and validation to construct a robust framework capable of accurately identifying toxic comments. The system's performance undergoes stringent testing on separate datasets, meticulously assessing its real-world applicability and resilience against unseen data.

The ultimate goal of this system is to actively contribute to fostering healthier online interactions by combatting cyberbullying, promoting compliance with community standards, and fostering a safer online environment. By detecting and managing toxic comments effectively, the system endeavors to minimize the negative impact of such content, safeguarding users' mental well-being, and fostering responsible online discourse. Its implementation stands as a crucial step towards creating an inclusive and secure digital space where users feel empowered to engage positively while mitigating the detrimental effects of toxic behavior.

6.2 IMPLEMENTATION

Implementing the code for real-time toxic comment detection involves integrating the developed model into a live environment to perform predictions on incoming comments or text data. Here's a high-level overview of how this can be achieved:

6.2.1. Model Serialization: Serialize or save the trained machine learning model(s) into a file format (e.g., using Python's `pickle` or `joblib`).

6.2.2. API or Web Service: Develop an API or web service using frameworks like Flask or Django in Python. Create endpoints to receive incoming text data/comments for prediction.

6.2.3. Data Preprocessing: Implement the same preprocessing steps (tokenization, lowercasing, noise removal, vectorization) used during training on the incoming text data.

6.2.4. Prediction Function: Load the serialized model into memory when the application starts. Write a function to preprocess the incoming text and make predictions using the loaded model.

6.2.5. Real-Time Integration: Connect the prediction function to the API endpoints. When new comments/text data arrive, call the prediction function to classify them as toxic or non-toxic.

6.2.6. Response Handling: Send the prediction results (toxicity label or probability) back as a response to the incoming request.

6.2.7. Deployment and Scaling: Deploy the API or web service on a server or cloud platform to make it accessible online. Scale the application based on traffic and usage demands.

6.2.8. Monitoring and Maintenance: Implement monitoring mechanisms to track the model's performance in real-time. Continuously monitor and maintain the system, applying updates or retraining the model periodically to adapt to evolving patterns in toxic comments.

6.3 SOURCE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
import matplotlib.pyplot as plt
import numpy as np
from tkinter import ttk
from tkinter import filedialog
import pandas as pd
from sklearn.model_selection import train_test_split
from string import punctuation
from nltk.corpus import stopwords
import nltk
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from sklearn import svm
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import hamming_loss

main = Tk()
main.title("Classification of Online Toxic Comments Using Machine Learning
Algorithms")
main.geometry("1300x1200")
```

```
global filename
global X, Y1, Y2, Y3, Y4, Y5, Y6
accuracy = []
global X_train, X_test, y_train, y_test
loss = []

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

textdata = []
global classifier1,classifier2,classifier3,classifier4,classifier5,classifier6
global tfidf_vectorizer

def cleanPost(doc):
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()]
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    tokens = ' '.join(tokens)
    return tokens

def uploadDataset():
    global filename
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="Dataset")
    text.insert(END,filename+" loaded\n")



def preprocess():
    textdata.clear()
```

```
global X, Y1, Y2, Y3, Y4, Y5, Y6
Y1 = []
Y2 = []
Y3 = []
Y4 = []
Y5 = []
Y6 = []
text.delete('1.0', END)
dataset = pd.read_csv(filename,encoding='iso-8859-1',nrows = 300)
for i in range(len(dataset)):
    msg = dataset._get_value(i, 'comment_text')
    toxic = dataset._get_value(i, 'toxic')
    severe_toxic = dataset._get_value(i, 'severe_toxic')
    obscene = dataset._get_value(i, 'obscene')
    threat = dataset._get_value(i, 'threat')
    insult = dataset._get_value(i, 'insult')
    identity_hate = dataset._get_value(i, 'identity_hate')
    msg = str(msg)
    msg = msg.strip().lower()
    Y1.append(int(toxic))
    Y2.append(int(severe_toxic))
    Y3.append(int(obscene))
    Y4.append(int(threat))
    Y5.append(int(insult))
    Y6.append(int(identity_hate))
    clean = cleanPost(msg)
    textdata.append(clean)
    text.insert(END,clean+"\n")
Y1 = np.asarray(Y1)
Y2 = np.asarray(Y2)
Y3 = np.asarray(Y3)
Y4 = np.asarray(Y4)
Y5 = np.asarray(Y5)
Y6 = np.asarray(Y6)
```

```
def countVector():
    global X, Y1, Y2, Y3, Y4, Y5, Y6
    global tfidf_vectorizer
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    stopwords=stopwords = nltk.corpus.stopwords.words("english")
    tfidf_vectorizer = TfidfVectorizer(stop_words=stopwords, use_idf=True,
smooth_idf=False, norm=None, decode_error='replace')
    tfidf = tfidf_vectorizer.fit_transform(textdata).toarray()
    df = pd.DataFrame(tfidf, columns=tfidf_vectorizer.get_feature_names_out())
    text.insert(END,str(df))
    print(df.shape)
    df = df.values
    X = df[:, 0:df.shape[1]]
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y1 = Y1[indices]
    Y2 = Y2[indices]
    Y3 = Y3[indices]
    Y4 = Y4[indices]
    Y5 = Y5[indices]
    Y6 = Y6[indices]
    X_train, X_test, y_train, y_test = train_test_split(X, Y1, test_size=0.2,random_state=0)
    text.insert(END,"\n\nTotal Comments found in dataset : "+str(len(X))+"\n")
    text.insert(END,"Total records used to train machine learning algorithms :
"+str(len(X_train))+"\n")
    text.insert(END,"Total records used to test machine learning algorithms  :
"+str(len(X_test))+"\n")


def train(Xdata,Ydata,cls):
    X_train, X_test, y_train, y_test = train_test_split(Xdata, Ydata,
```

```
test_size=0.2,random_state=0)
    cls.fit(X_train, y_train)
    predict = cls.predict(X_test)
    acc = accuracy_score(y_test,predict)*100
    loss = hamming_loss(y_test,predict)*100
    return acc,loss


def train1(Xdata,Ydata,cls):
    X_train, X_test, y_train, y_test = train_test_split(Xdata, Ydata,
test_size=0.2,random_state=0)
    cls.fit(Xdata, Ydata)
    predict = cls.predict(X_test)
    acc = accuracy_score(y_test,predict)*100
    loss = hamming_loss(y_test,predict)*100
    return acc,loss


def runSVM():
    global X, Y1, Y2, Y3, Y4, Y5, Y6
    text.delete('1.0', END)
    accuracy.clear()
    loss.clear()
    cls1 = svm.SVC()
    acc1,loss1 = train(X,Y1,cls1)
    cls2 = svm.SVC()
    acc2,loss2 = train(X,Y2,cls2)
    cls3 = svm.SVC()
    acc3,loss3 = train(X,Y3,cls3)
    cls4 = svm.SVC()
    acc4,loss4 = train(X,Y4,cls4)
    cls5 = svm.SVC()
    acc5,loss5 = train(X,Y5,cls5)
    cls6 = svm.SVC()
```

```
    acc6,loss6 = train(X,Y6,cls6)
    acc = acc1 + acc2 + acc3 + acc4 + acc5 + acc6
    acc = (acc / 600) * 100
    lossValue = loss1 + loss2 + loss3 + loss4 + loss5 + loss6
    accuracy.append(acc)
    lossValue = lossValue / 600
    loss.append(lossValue)
    text.insert(END,"SVM Accuracy : "+str(acc)+"\n")
    text.insert(END,"SVM Hamming Loss : "+str(lossValue)+"\n\n")


def runLR():
    global X, Y1, Y2, Y3, Y4, Y5, Y6
    cls1 = LogisticRegression()
    acc1,loss1 = train(X,Y1,cls1)
    cls2 = LogisticRegression()
    acc2,loss2 = train(X,Y2,cls2)
    cls3 = LogisticRegression()
    acc3,loss3 = train(X,Y3,cls3)
    cls4 = LogisticRegression()
    acc4,loss4 = train(X,Y4,cls4)
    cls5 = LogisticRegression()
    acc5,loss5 = train(X,Y5,cls5)
    cls6 = LogisticRegression()
    acc6,loss6 = train(X,Y6,cls6)
    acc = acc1 + acc2 + acc3 + acc4 + acc5 + acc6
    acc = (acc / 600) * 100
    accuracy.append(acc)
    lossValue = loss1 + loss2 + loss3 + loss4 + loss5 + loss6
    lossValue = lossValue / 600
    loss.append(lossValue)
    text.insert(END,"Logistic Regression Accuracy : "+str(acc)+"\n")
    text.insert(END,"Logistic Regression Hamming Loss : "+str(lossValue)+"\n\n")
```

```
def runNB():
    global X, Y1, Y2, Y3, Y4, Y5, Y6
    cls1 = GaussianNB()
    acc1,loss1 = train(X,Y1,cls1)
    cls2 = GaussianNB()
    acc2,loss2 = train(X,Y2,cls2)
    cls3 = GaussianNB()
    acc3,loss3 = train(X,Y3,cls3)
    cls4 = GaussianNB()
    acc4,loss4 = train(X,Y4,cls4)
    cls5 = GaussianNB()
    acc5,loss5 = train(X,Y5,cls5)
    cls6 = GaussianNB()
    acc6,loss6 = train(X,Y6,cls6)
    acc = acc1 + acc2 + acc3 + acc4 + acc5 + acc6
    acc = (acc / 600) * 100
    accuracy.append(acc)
    lossValue = loss1 + loss2 + loss3 + loss4 + loss5 + loss6
    lossValue = lossValue / 600
    loss.append(lossValue)
    text.insert(END,"Naive Bayes Accuracy : "+str(acc)+"\n")
    text.insert(END,"Naive Bayes Hamming Loss : "+str(lossValue)+"\n\n")

def runDecisionTree():
    global X, Y1, Y2, Y3, Y4, Y5, Y6
    cls1 = DecisionTreeClassifier()
    acc1,loss1 = train(X,Y1,cls1)
    cls2 = DecisionTreeClassifier()
    acc2,loss2 = train(X,Y2,cls2)
    cls3 = DecisionTreeClassifier()
    acc3,loss3 = train(X,Y3,cls3)
    cls4 = DecisionTreeClassifier()
    acc4,loss4 = train(X,Y4,cls4)
```

```
    cls5 = DecisionTreeClassifier()
    acc5,loss5 = train(X,Y5,cls5)
    cls6 = DecisionTreeClassifier()
    acc6,loss6 = train(X,Y6,cls6)
    acc = acc1 + acc2 + acc3 + acc4 + acc5 + acc6
    acc = (acc / 600) * 100
    accuracy.append(acc)
    lossValue = loss1 + loss2 + loss3 + loss4 + loss5 + loss6
    lossValue = lossValue / 600
    loss.append(lossValue)
    text.insert(END,"Decision Tree Accuracy : "+str(acc)+"\n")
    text.insert(END,"Decision Tree Hamming Loss : "+str(lossValue)+"\n\n")

def runRandomForest():
    global classifier1,classifier2,classifier3,classifier4,classifier5,classifier6
    global X, Y1, Y2, Y3, Y4, Y5, Y6
    cls1 = RandomForestClassifier()
    acc1,loss1 = train1(X,Y1,cls1)
    cls2 = RandomForestClassifier()
    acc2,loss2 = train1(X,Y2,cls2)
    cls3 = RandomForestClassifier()
    acc3,loss3 = train1(X,Y3,cls3)
    cls4 = RandomForestClassifier()
    acc4,loss4 = train1(X,Y4,cls4)
    cls5 = RandomForestClassifier()
    acc5,loss5 = train1(X,Y5,cls5)
    cls6 = RandomForestClassifier()
    acc6,loss6 = train1(X,Y6,cls6)
    classifier1 = cls1
    classifier2 = cls2
    classifier3 = cls3
    classifier4 = cls4
    classifier5 = cls5
    classifier6 = cls6
```

```
    acc = acc1 + acc2 + acc3 + acc4 + acc5 + acc6
    acc = (acc / 600) * 100
    accuracy.append(acc)
    lossValue = loss1 + loss2 + loss3 + loss4 + loss5 + loss6
    lossValue = lossValue / 600
    loss.append(lossValue)
    text.insert(END,"Random Forest Accuracy : "+str(acc)+"\n")
    text.insert(END,"Random Forest Hamming Loss : "+str(lossValue)+"\n\n")

def runKNN():
    global X, Y1, Y2, Y3, Y4, Y5, Y6
    cls1 = KNeighborsClassifier(n_neighbors = 2)
    acc1,loss1 = train(X,Y1,cls1)
    cls2 = KNeighborsClassifier(n_neighbors = 2)
    acc2,loss2 = train(X,Y2,cls2)
    cls3 = KNeighborsClassifier(n_neighbors = 2)
    acc3,loss3 = train(X,Y3,cls3)
    cls4 = KNeighborsClassifier(n_neighbors = 2)
    acc4,loss4 = train(X,Y4,cls4)
    cls5 = KNeighborsClassifier(n_neighbors = 2)
    acc5,loss5 = train(X,Y5,cls5)
    cls6 = KNeighborsClassifier(n_neighbors = 2)
    acc6,loss6 = train(X,Y6,cls6)
    acc = acc1 + acc2 + acc3 + acc4 + acc5 + acc6
    acc = (acc / 600) * 100
    accuracy.append(acc)
    lossValue = loss1 + loss2 + loss3 + loss4 + loss5 + loss6
    lossValue = lossValue / 600
    loss.append(lossValue)
    text.insert(END,"KNN Accuracy : "+str(acc)+"\n")
    text.insert(END,"KNN Hamming Loss : "+str(lossValue)+"\n\n")



def graph():
```

```
    df = pd.DataFrame([['SVM','Accuracy',accuracy[0]],['SVM','Hamming Loss',loss[0]],
                       ['Logistic Regression','Accuracy',accuracy[1]],['Logistic
Regression','Hamming Loss',loss[1]],
                       ['Naive Bayes','Accuracy',accuracy[2]],['Naive Bayes','Hamming
Loss',loss[2]],
                       ['Decision Tree','Accuracy',accuracy[3]],['Decision Tree','Hamming
Loss',loss[3]],
                       ['Random Forest','Accuracy',accuracy[4]],['Random Forest','Hamming
Loss',loss[4]],
                       ['KNN','Accuracy',accuracy[5]],['KNN','Hamming Loss',loss[5]],
                      ], columns=['Parameters', 'Algorithms', 'Value'])

    pivot_df = df.pivot(index="Parameters", columns="Algorithms", values="Value")
    pivot_df.plot(kind='bar')
    plt.show()



def predict():
    global classifier1,classifier2,classifier3,classifier4,classifier5,classifier6
    testfile = filedialog.askopenfilename(initialdir="Dataset")
    testData = pd.read_csv(testfile)
    text.delete('1.0', END)
    for i in range(len(testData)):
        msg = testData._get_value(i, 'comment')
        review = msg.lower()
        review = review.strip().lower()
        review = cleanPost(review)
        testReview = tfidf_vectorizer.transform([review]).toarray()
        predict1 = classifier1.predict(testReview)
        predict2 = classifier2.predict(testReview)
        predict3 = classifier3.predict(testReview)
        predict4 = classifier4.predict(testReview)
        predict5 = classifier5.predict(testReview)
```

```
        predict6 = classifier6.predict(testReview)
        if predict1 == 0 and predict2 == 0 and predict3 == 0 and predict4 == 0 and predict5
== 0 and predict6 == 0:
            text.insert(END,msg+" [NOT CONTAINS TOXIC Comments]\n\n")
        elif predict1 == 1:
            text.insert(END,msg+" [CONTAINS TOXIC Comments]\n\n")
        elif predict2 == 1:
            text.insert(END,msg+" [CONTAINS SEVERE_TOXIC Comments]\n\n")
        elif predict3 == 1:
            text.insert(END,msg+" [CONTAINS OBSCENE Comments]\n\n")
        elif predict4 == 1:
            text.insert(END,msg+" [CONTAINS THREAT Comments]\n\n")
        elif predict5 == 1:
            text.insert(END,msg+" [CONTAINS INSULT Comments]\n\n")
        elif predict6 == 1:
            text.insert(END,msg+" [CONTAINS IDENTITY_HATE Comments]\n\n")


font = ('times', 15, 'bold')
title = Label(main, text='Classification of Online Toxic Comments Using Machine
Learning Algorithms')
title.config(bg='darkviolet', fg='gold')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 13, 'bold')
ff = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Toxic Comments Dataset",
command=uploadDataset)
uploadButton.place(x=20,y=100)
uploadButton.config(font=ff)
```

```
processButton = Button(main, text="Preprocess Dataset", command=preprocess)
processButton.place(x=20,y=150)
processButton.config(font=ff)

cvButton = Button(main, text="Apply Count Vectorizer", command=countVector)
cvButton.place(x=20,y=200)
cvButton.config(font=ff)

svmButton = Button(main, text="Run SVM Algorithm", command=runSVM)
svmButton.place(x=20,y=250)
svmButton.config(font=ff)

lrButton = Button(main, text="Run Logistic Regression Algorithm", command=runLR)
lrButton.place(x=20,y=300)
lrButton.config(font=ff)

nbButton = Button(main, text="Run Naive Bayes Algorithm", command=runNB)
nbButton.place(x=20,y=350)
nbButton.config(font=ff)

dtButton = Button(main, text="Run Decision Tree Algorithm",
command=runDecisionTree)
dtButton.place(x=20,y=400)
dtButton.config(font=ff)

rfButton = Button(main, text="Run Random Forest Algorithm",
command=runRandomForest)
rfButton.place(x=20,y=450)
rfButton.config(font=ff)

knnButton = Button(main, text="Run KNN Algorithm", command=runKNN)
knnButton.place(x=20,y=500)
knnButton.config(font=ff)
```

```
graphButton = Button(main, text="Accuracy Comparison Graph", command=graph)
graphButton.place(x=20,y=550)
graphButton.config(font=ff)

predictButton = Button(main, text="Predict Toxic Comments from Test Data",
command=predict)
predictButton.place(x=20,y=600)
predictButton.config(font=ff)


font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=110)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=360,y=100)
text.config(font=font1)

main.config(bg='forestgreen')
main.mainloop()
```

TESTING

7. TESTING

7.1. UNIT TESTING:

Unit Testing is a fundamental aspect of the software development lifecycle, focusing on the validation of individual units or components of code in isolation. Each unit, typically a function, method, or class, is subjected to tests to ensure its correctness and proper functioning. The primary objective of Unit Testing is to identify and rectify errors or bugs in these isolated components before integrating them into the broader system.

In the context of the Toxic Comment Classification System, Unit Testing plays a crucial role in scrutinizing specific functions responsible for various tasks. For instance, the tokenization function, which is responsible for splitting text into individual tokens, would undergo Unit Testing to verify its accuracy. Similarly, functions related to text preprocessing, vectorization, and model training would undergo meticulous testing to ensure their individual reliability.

These tests involve providing input to the units and verifying that the output aligns with the expected outcomes. Various testing frameworks, such as Python's `unittest` or `pytest`, are commonly employed for automating and organizing Unit Tests. By implementing Unit Testing, developers can detect and address issues early in the development process, contributing to the overall robustness and reliability of the system.

7.2. INTEGRATION TESTING:

Integration Testing builds upon Unit Testing by evaluating the interactions and interfaces between integrated components or modules. It aims to ensure that different parts of the system operate seamlessly when combined. In the context of the Toxic Comment Classification System, Integration Testing becomes crucial when examining how preprocessing, model training, and prediction functions integrate and communicate.

Integration Tests assess whether the output produced by one component correctly aligns with the input expected by another. This process involves evaluating the interoperability of

different modules to identify and resolve issues that may arise from their integration. Ensuring that the integrated system behaves cohesively is paramount for the overall success of the Toxic Comment Classification System.

Testing frameworks, such as JUnit in Java or Pytest in Python, often provide tools and structures for organizing and executing Integration Tests. These tests serve as a bridge between isolated Unit Tests and the broader acceptance criteria, providing insights into the system's behavior when components are combined.

7.3. ACCEPTANCE TESTING:

Acceptance Testing represents the phase where the system's compliance with specified requirements and user expectations is rigorously evaluated. It serves as a validation mechanism to ensure that the system aligns with predefined criteria, user stories, or broader business needs. In the realm of the Toxic Comment Classification System, Acceptance Testing becomes especially significant in confirming that the system accurately identifies and categorizes diverse comments as toxic or non-toxic.

These tests are often scenario-based, simulating real-world usage to verify if the system satisfies user-defined criteria or business goals. For instance, a range of comments with varying levels of toxicity might be presented to the system to observe how accurately it classifies them. By conducting Acceptance Testing, developers can ascertain whether the system is ready for deployment and whether it meets the intended functionality and user expectations.

Automated tools like Selenium for web applications or Cucumber for behavior-driven development can streamline the execution and documentation of Acceptance Tests. These tests act as a final gatekeeper before deploying the system, providing confidence that it will perform as intended in a real-world context.

7.4. TESTING ON OUR SYSTEM:

The culmination of Unit Testing, Integration Testing, and Acceptance Testing occurs in the phase of Testing on our System. This involves executing various testing methodologies on

the fully implemented Toxic Comment Classification System, providing a comprehensive evaluation of its behavior, functionality, and alignment with the defined objectives.

During this phase, the system is tested as a whole, and the documentation of tests conducted, their outcomes, and the system's behavior under different scenarios becomes pivotal. It offers insights into how the individual components, validated through Unit and Integration Testing, function together in the integrated environment. This holistic approach ensures that the system meets the specified requirements, functions reliably, and aligns with user expectations.

Testing on Our System involves monitoring and analyzing the results obtained from the various testing methodologies. It provides a comprehensive understanding of the system's strengths, weaknesses, and overall readiness for deployment. This phase serves as a critical step in the software development lifecycle, contributing to the creation of robust and dependable software systems.

RESULT

8. RESULT

Fig 8.1 Upload Toxic Comments Dataset

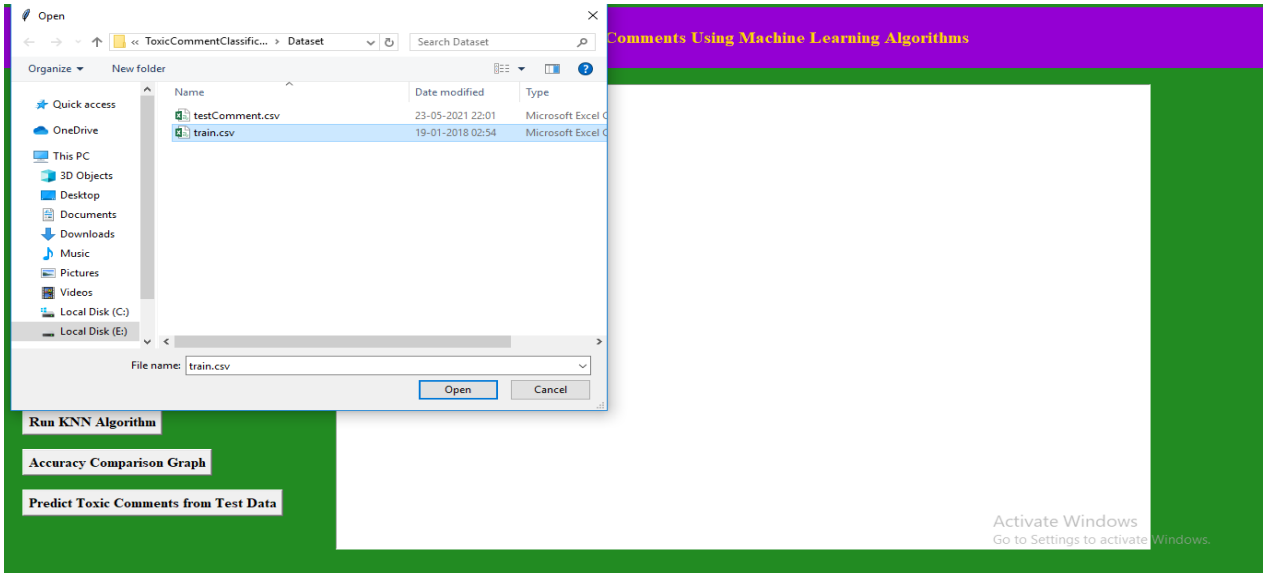

Fig 8.2 train.csv

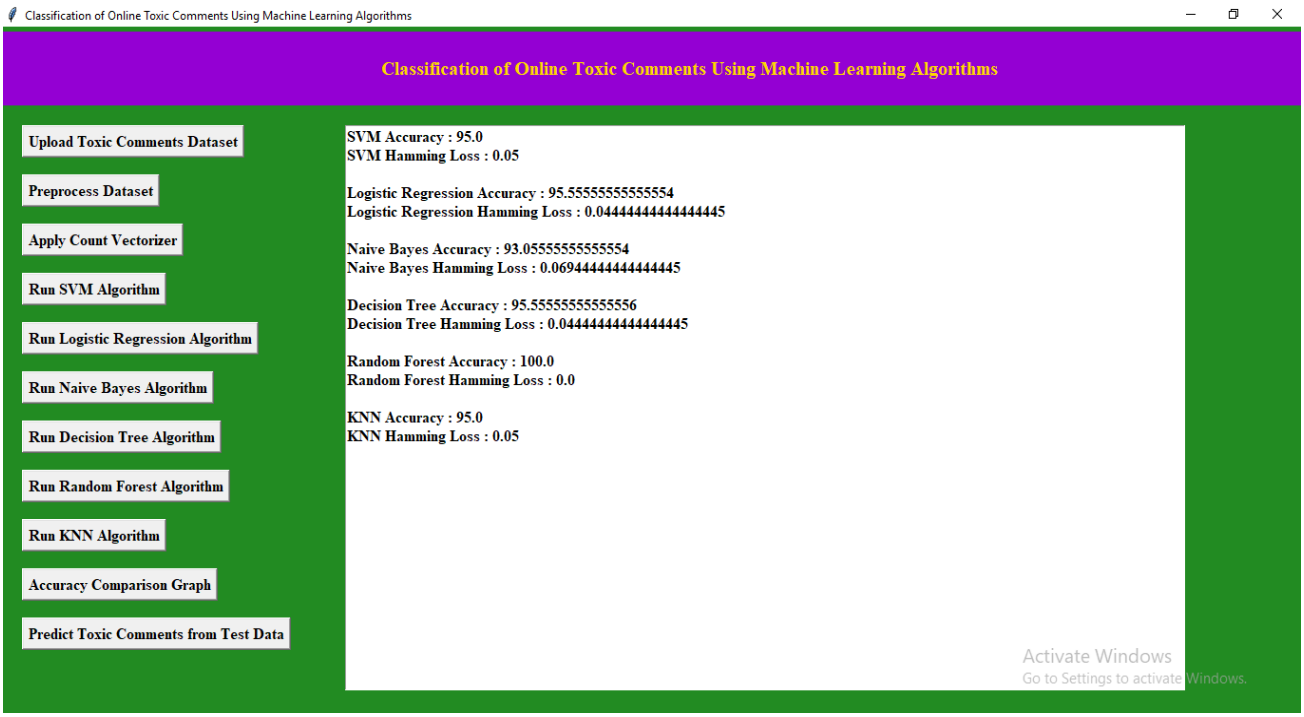

Fig 8.3 Accuracy and Hamming Loss

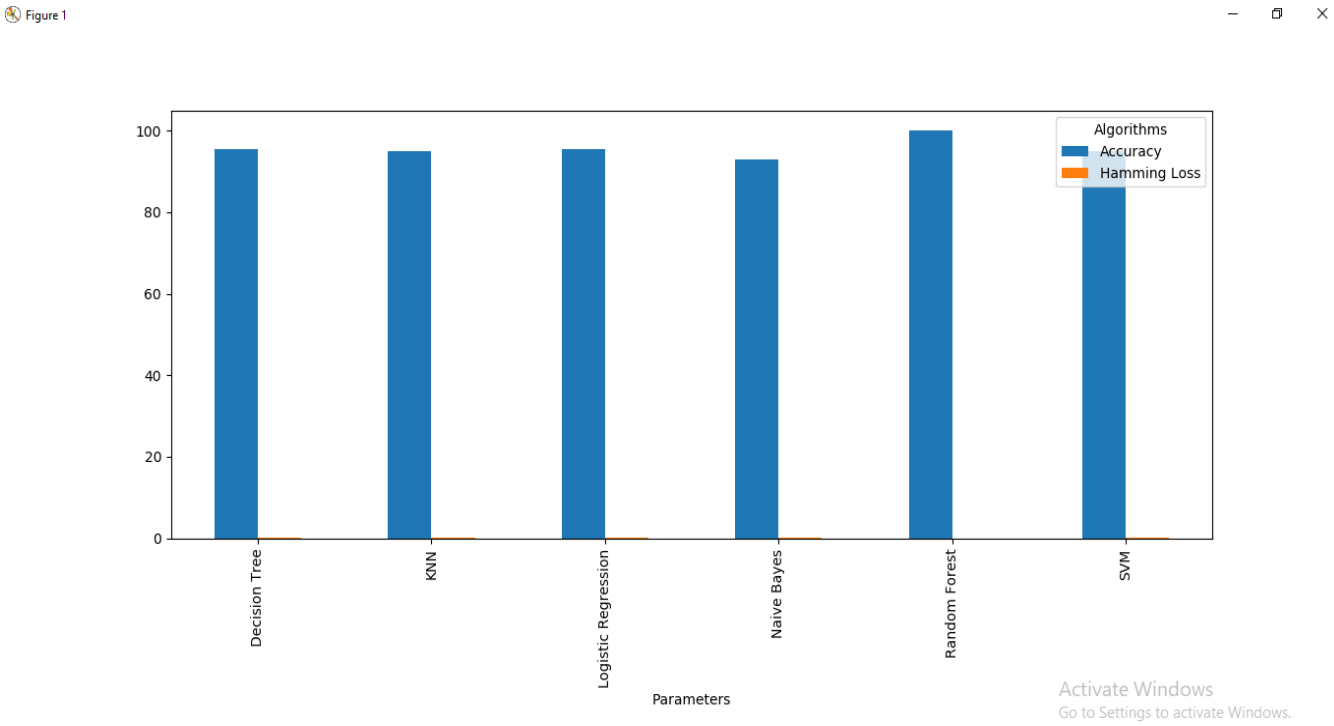

Fig 8.4 Accuracy Comparison Graph

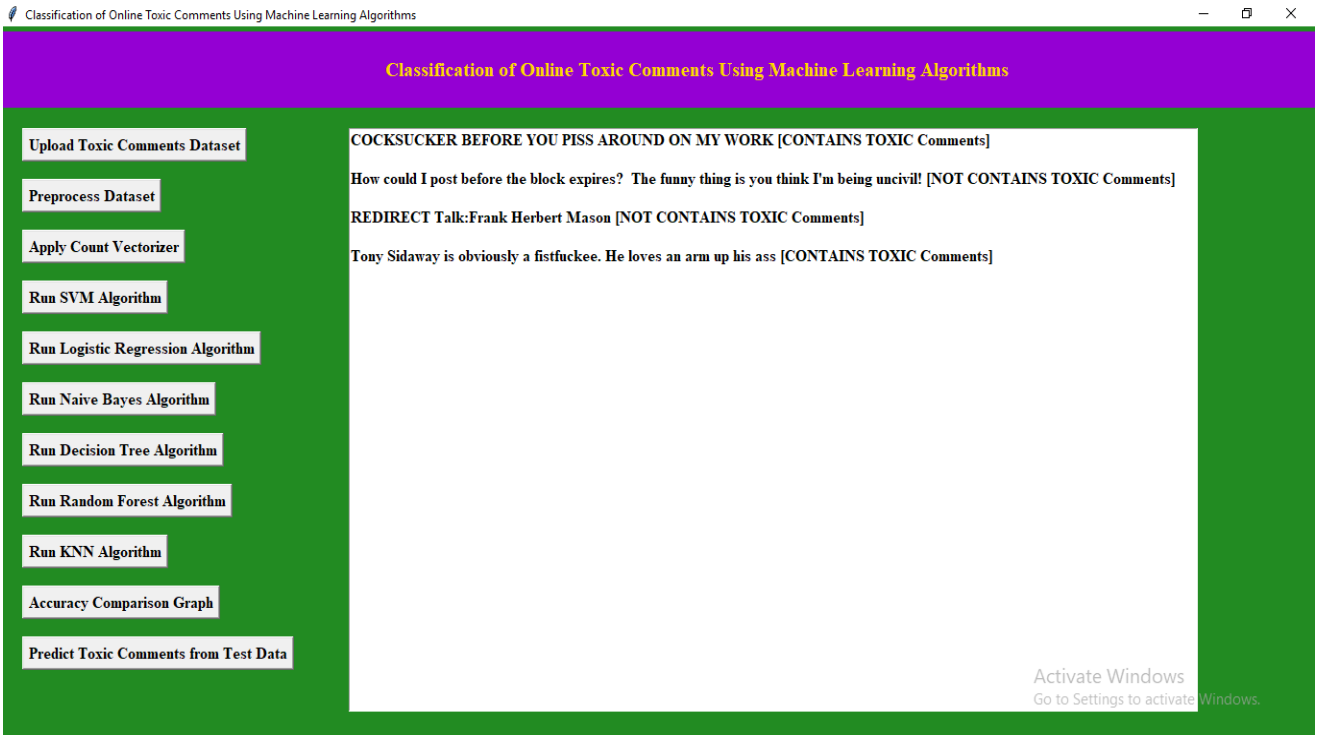

Fig 8.5 Final Result

CONCLUSION

9. CONCLUSION

We have discussed six Machine learning techniques i.e., logistic regression, Naive Bayes, decision tree, random forest, KNN classification, and SVM classifier, and compared their hamming loss, accuracy, and log loss in this paper. Now after proper analysis, we can say that in terms of hamming loss, logistic regression performs best because in that case, our hamming loss is least, while in terms of accuracy, logistic regression performs best because accuracy is best in that model in comparison to other ones and terms of log loss, random forest works best due to least possible log loss in that model. So, our final model selection will be based on the combination of hamming loss and accuracy. Since we got the maximum accuracy i.e., 89.46 % and least possible hamming loss i.e., 2.43 % in case of the logistic regression model. We will select the logistic regression model as our final machine learning technique since it works best for our data.

BIBLOGRAPHY

10.BIBLOGRAPHY

[1] H. M. Saleem, K. P. Dillon, S. Benesch, and D. Ruths, “A Web of Hate: Tackling Hateful Speech in Online Social Spaces,” 2017, [Online]. Available: http://arxiv.org/abs/1709.10159.

[2] M. Duggan, “Online harassment 2017,” Pew Res., pp. 1–85, 2017, Doi: 202.419.4372.

[3] M. A. Walker, P. Anand, J. E. F. Tree, R. Abbott, and J. King, “A corpus for research on deliberation and debate,” Proc. 8th Int. Conf. Lang. Resour. Eval. Lr. 2012, pp. 812–817, 2012.

[4] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, “Antisocial behavior in online discussion communities,” Proc. 9th Int. Conf. Web Soc. Media, ICWSM 2015, pp. 61–70, 2015.

[5] B. Mathew et al., “Thou shalt not hate: Countering online hate speech,” Proc. 13th Int. Conf. Web Soc. Media, ICWSM 2019, no. August, pp. 369–380, 2019.

[6] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, “Abusive language detection in online user content,” 25th Int. World Wide Web Conf. WWW 2016, pp. 145–153, 2016, doe: 10.1145/2872427.2883062.

[7] E. K. Ikonomakis, S. Kotsiantis, and V. Tampakas, “Text Classification Using Machine Learning Techniques,” no. August, 2005.

[8] M. R. Murty, J. V. . Murthy, and P. Reddy P.V.G.D, “ Text Document Classification based on Least Square Support Vector Machines with Singular Value Decomposition,” Int. J. Comput. Appl., vol. 27, no. 7, pp. 21–26, 2011, doi: 10.5120/3312-4540.

[9] E. Wulczyn, N. Thain, and L. Dixon, “Ex machina: Personal attacks seen at scale,” 26th Int. World Wide Web Conf. WWW 2017, pp. 1391– 1399, 2017, doi: 10.1145/3038912.3052591.

[10] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, “Deceiving Google’s Perspective API Built for Detecting Toxic Comments,” 2017, [Online]. Available: http://arxiv.org/abs/1702.08138.