Final Project Report

“TOXIC COMMENT CLASSFICATION FOR SOCIAL MEDIA”

1.Introduction

1.1 Project overviews

Online harassment and toxic comments pose significant challenges on social media platforms, where escalating user bases and harmful content have led to emotional distress and community disruption. Conventional moderation solutions rely on manual review or historical data, often resulting in delayed responses to emerging toxic comment trends. Toxic Comment Classification uses machine learning algorithms to analyse real-time data from multiple sources—user reports, comment patterns, sentiment analysis and platform policies—providing accurate toxic comment detection.

These insights enable proactive moderation, improved user experience and optimized community management, resulting in safer online environments and reduced harm. The primary objective of Toxic Comment Classification is to develop an accurate and scalable machine learning model for identifying toxic comments, empowering social media moderators, administrators and policymakers to make data-driven decisions.

This system aims to overcome the limitations of traditional moderation methods by integrating real-time and historical data to offer dynamic detection of toxic comments.

1.2 Objectives

The main objective of "Toxic Comment Classification for Social Media" is to develop a machine learning-based system that accurately identifies and classifies toxic comments in real-time. The system aims to enhance social media moderation, reduce online harassment and improve user experience through dynamic and data-driven insights.

Specific Objectives:

1. Accurate Toxic Comment Detection:
   1. Build a reliable ML model to classify toxic comments based on historical and real-time data.
   2. Improve the precision of toxic comment detection compared to traditional moderation methods.

2. Real-Time Insights:
   1. Integrate live data feeds from user reports, comment patterns and platform policies to provide dynamic toxic comment detection.
   2. Continuously update the model to reflect current trends and language usage.

3. Optimized Moderation:
   1. Assist moderators in proactively identifying and addressing toxic comments.
   2. Enable efficient content management and minimize harmful content exposure.

4. Improved User Experience:
   1. Promote a safer online environment by reducing toxic interactions.

2. Enhance user engagement and satisfaction through proactive moderation.

5. Scalability and Adaptability:

1. Design a flexible solution that can scale across multiple social media platforms.
2. Ensure compatibility with emerging technologies and evolving language trends.

2. Project Initialization and Planning Phase

2.1 Define Problem Statements (Customer Problem Statement Template):

Managing online harassment and toxic comments on social media remains a challenge due to escalating user bases and harmful content. Traditional moderation methods are limited in scope and accuracy, leading to poor detection and response times. This results in emotional distress, community disruption and reputational damage.

PROBLEM STATEMENT (PS)	I AM (CUSTOMER)	I'M TRYING TO	BUT	BECAUSE	WHICH MAKES ME FEEL
PS-1	A social media moderator or administrator	Identify and address toxic comments efficiently	Current methods are manual and time-consuming	Growing user bases and complex comment patterns	Overwhelmed and concerned about user harm
PS-2	A social media user or community member	Feel safe and engage constructively online	Toxic comments are prevalent and unaddressed	Lack of effective moderation and real-time insights	Lack of effective moderation and real-time insights

I am	I'm trying to	But	Because	Which makes me feel
A social media moderator Or administrator	Identify and address toxic comments efficiently	Current methods are manual and time consuming	Growing user bases and complex comment patterns	Overwhelmed and concerned about user harm
I am	**I'm trying to**	**But**	**Because**	**Which makes me feel**
A social media community member	Feel safe and Engage Constructively online	Toxic comments are prevalent and unaddressed	Lack of Effective Moderation and real time insights	Lack of effective moderation and real-time insights

2.2 Project Proposal (Proposed Solution) template

This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel.

Project Overview

Objective	The primary objective of "Toxic Comment Classification for Social Media" is to develop an advanced, machine learning-based system for accurate and real-time detection of toxic comments, promoting a safer online environment.

Scope	This project focuses on developing a machine learning-based system for toxic comment classification using real-time and historical data from social media platforms. The project includes model development, validation and deployment through a web or mobile interface, offering real-time alerts and moderation recommendations. It supports community management optimization by aiding moderators in reducing harmful content and minimizing user harm.

Problem Statement

Description	Social media platforms struggle with increasing online harassment due to outdated moderation methods and inability to capture emerging toxic comment patterns. This results in emotional distress, community disruption and reputational damage. "Toxic Comment Classification for Social Media" aims to solve this by developing a machine learning-based system integrating real-time data from user reports, comment patterns and platform policies.
Impact	"Toxic Comment Classification for Social Media" enhances online safety by providing accurate toxic comment detection, reducing harm and minimizing user distress. Moderators optimize content management, leading to improved user experience. City planners and policymakers benefit from data-driven insights for informed decision-making.

Proposed System

Approach	The approach for "Toxic Comment Classification for Social Media" involves a systematic process beginning with the collection of real-time and historical data from social media platforms, user reports and comment patterns. This data undergoes cleaning and preprocessing to handle missing values and normalize features.
Key Features	"Toxic Comment Classification for Social Media" offers real-time toxic comment detection by integrating live data from user reports, comment patterns and platform policies. Its user-friendly web and mobile interface provides accurate moderation recommendations, alerts and actionable insights for proactive content management.

Resource Requirements

Hardware-

Resource Type	Description	Specification/Allocation
Computing Resources	GPUs for model training	2 x NVIDIA V100 GPUs
Memory	RAM for processing large datasets	32 GB RAM
Storage	Disk space for models and logs	2 TB SSD

Software-

Resource Type	Description	Specification/Allocation
Frameworks	Python frameworks	Flask, Fast API
Libraries	Additional machine learning tools	TensorFlow, PyTorch
Development Environment	IDE and version control tools	Jupyter Notebook, Git

Data-

Resource Type	Description	Specification/Allocation
Data	Social media platforms (e.g., Twitter, Facebook)	CSV, JSON or SQL database

2.3 Initial Project Planning Template

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint1	User Registration and Login	USN-1	Authentication & Account Setup	2	High	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	16 Aug 2024	18 Aug 2024
Sprint1	User Registration and Login	USN-2	Email Verification	1	High	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	16 Aug 2024	18 Aug 2024

Sprint1	User Registration and Login	USN-3	User Authentication & Login	1	Low	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	16 Aug 2024	18 Aug 2024
Sprint2	Data Preprocessing and Model Selection	USN-4	Data Cleaning, Encoding	2	Medium	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	20 Aug 2024	25 Aug 2024
Sprint2	Data Preprocessing and Model Selection	USN-5	Model Training & Tuning	5	High	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	20 Aug 2024	25 Aug 2024
Sprint3	Model Evaluation and Web Deployment	USN-8	Performance Evaluation	3	Medium	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	01 Sep 2024	5 Sep 2024
Sprint3	Model Evaluation and Web Deployment	USN-9	Backend Development	4	High	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	01 Sep 2024	5 Sep 2024
Sprint4	Web Application Testing & Deployment	USN-10	Testing & Deployment	5	Medium	Sadhika, Srikanth, Nagarjuna, Ganesh, Karishma	6 Sep 2024	10 Sep 2024

3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan & Raw Data Sources Identification Template

Elevate your online safety strategy with the Data Collection plan and Raw Data Sources report for "Toxic Comment Classification for Social Media", ensuring meticulous data curation and integrity for informed decision-making in every moderation and policy endeavor.

Data Collection Plan Template

Section	Description
Project Overview	"Toxic Comment Classification for Social Media" aims to develop a machine learning-based solution for real-time detection and classification of toxic comments. By integrating data from social media platforms, user reports and natural language processing techniques, the project seeks to provide accurate predictions of toxic comment patterns and support proactive moderation.
Data Collection Plan	• Collect comments from social media platforms (e.g., Twitter, Facebook) to analyse language patterns and toxicity. • Gather user reports on toxic comments to train machine learning models. • Collect data on user interactions (e.g., likes, replies) to understand comment context.
Raw Data Sources Identified	The raw data sources report ensures data integrity by meticulously curating sources from social media platforms (APIs and web scraping), user-generated content (comments, posts, reviews), moderation tools and user feedback, NLP libraries and research papers, and online safety and harassment datasets.

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions

Smart Internz Platform	This dataset includes data about toxic comments, and types of toxic comments.	https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data	CSV	2.2 MB	Public

Data Collection and Preprocessing Phase

3.2 Data Quality Report Template

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset: Jigsaw Toxic Comment Classification Challenge	Inconsistent text formatting	Minor	Standardize text formatting using preprocessing techniques.

Data Collection and Preprocessing Phase

3.3 Data Exploration and Preprocessing Template (Toxic Comment Analysis System)

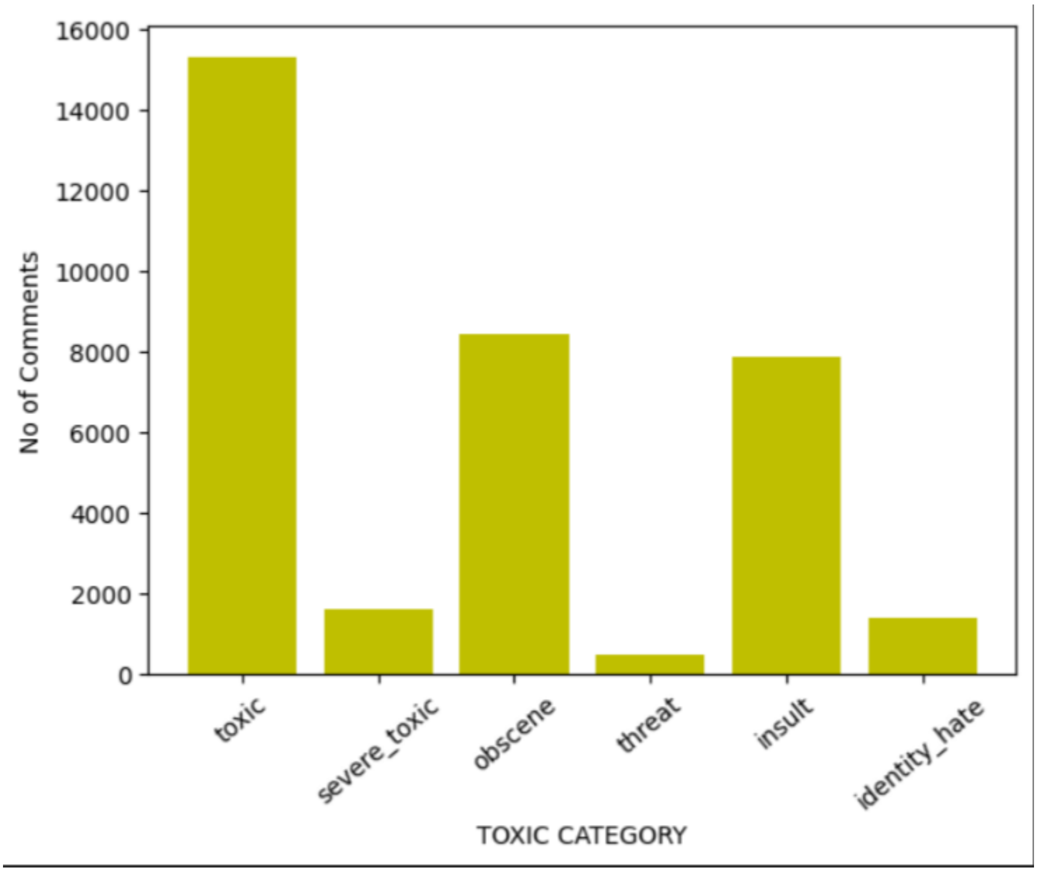
Our system's data exploration and preprocessing phase identifies relevant data sources, detects quality issues such as missing values and duplicates, and implements effective resolution plans to ensure accurate and reliable analysis. This crucial step enables our toxic comment analysis system to maintain data integrity, guaranteeing informed decision-making for proactive moderation and safer online interactions.

Data Overview-

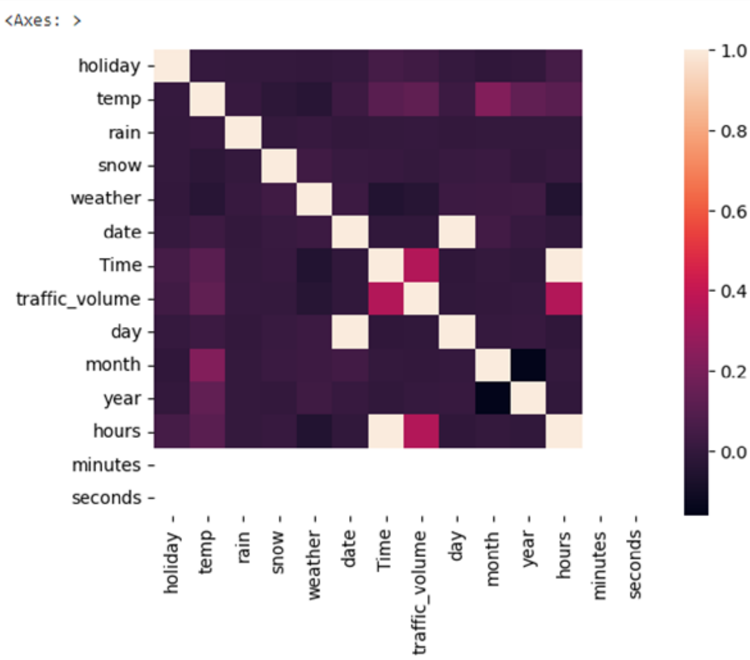
	toxic	severe_toxic	obscene	threat	insult	identity_hate
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

159571 rows × 6 columns

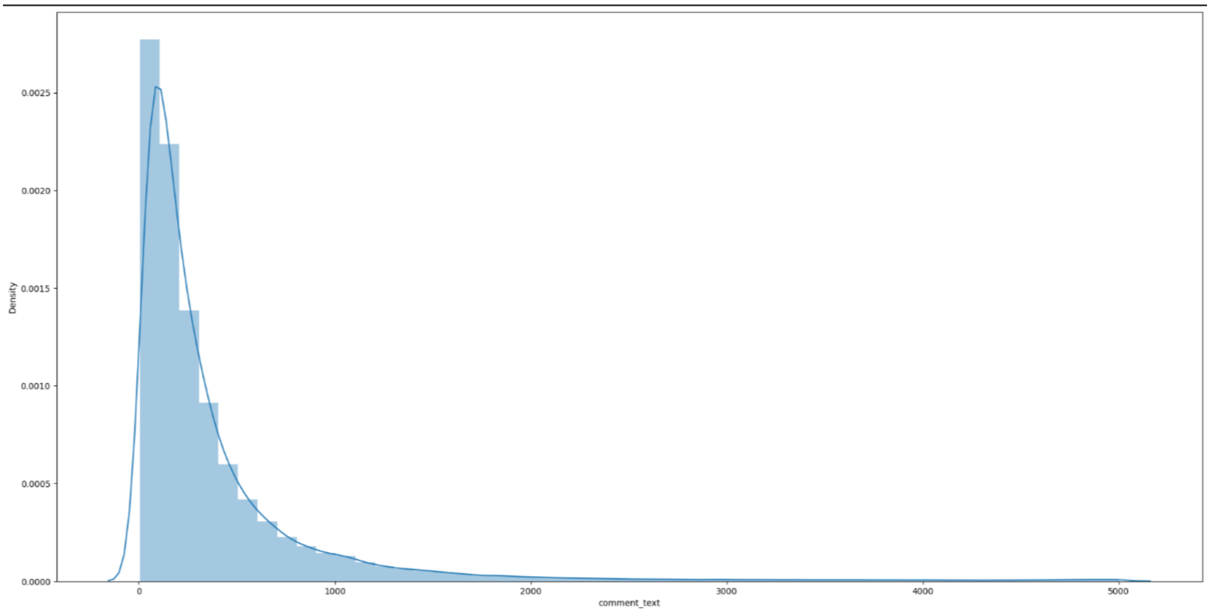
Univariate Analysis-



Bivariate Analysis-



Multivariate Analysis –



Outliers and Anomalies-

***

Data Preprocessing Code Screenshots

Reading the Dataset-

```
comment_train = pd.read_csv("train.csv")

comment_train.head()
```

✓ 0.7s

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

Preprocessing the data-

```
import nltk #importing nltk
nltk.download('stopwords') #downlaoding the stopwords
stop = stopwords.words('english') #As our training dataset is in only one langu
puncutations = string.punctuation #From string package we can retrieve the list
stop = stop + list(puncutations) #adding the list of puntautions on the array w
stop
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\saidi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Cleaning the Dataset-

```
comment_test['Cleaned_data'] = comment_test['comment_text'].apply(cleaning)
```

```
comment_test
```

	id	comment_text	Cleaned_data
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	yo bitch ja rule succesful ever whats hat sad ...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	rfc title fine imo
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	source zawe ashton lapland
3	00017563c3f7919a	:If you have a look back at the source, the in...	look back source information update correct fo...
4	00017695ad8997eb	I don't anonymously edit articles at all.	anonymously edit article
...	...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...	totally agree stuff nothing long crap
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...	throw field home plate get faster throw cut ma...
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...	okinotorishima categories see change agree cor...
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...	one found nations eu germany law return quite ...
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...	stop already bullshit welcome fool think kind ...

153164 rows × 3 columns

Feature Engineering-

Attached codes in the final submission.

4.Model Development Phase Template

4.1 Feature Selection Report Template

In our upcoming update, each proposed feature for the Toxic Comment Analysis System will be paired with a concise description, empowering users to make informed decisions. Users will indicate whether they accept or reject each feature, providing clear reasoning behind their selection. This streamlined process facilitates collaborative decision-making, enhances transparency in feature selection and ensures stakeholder alignment.

Feature	**Description**	**Selected (Yes/No)**	**Reasoning**
Comment Length	Indicates the number of characters in a comment.	Yes	Comment length can impact toxicity, with longer comments potentially containing more harmful content.
Sentiment Analysis	Evaluates comment sentiment (positive, negative, neutral).	Yes	Sentiment analysis helps identify toxic comments with negative sentiment.
Profanity Detection	Identifies profane language in comments.	Yes	Profanity is a strong indicator of toxic comments.
Named Entity Recognition (NER)	Identifies individuals, organizations and locations.	Yes	NER helps detect personal attacks and toxic references.
Part-of-Speech (POS) Analysis	Examines comment grammar and syntax.	Yes	POS analysis assists in identifying toxic language patterns.

Comment Date	Records the date of comment posting.	No	Date alone may not provide significant insight into toxicity.
Comment Time	Records the time of comment posting.	No	Time may not directly impact comment toxicity.

Model Development Phase Template

4.2 Model Selection Report

Model	Description	Performance Metric (e.g., Accuracy F1 Score)
Logistic Regression	A linear model that predicts probabilities of toxic comments.	Accuracy score=82%
Linear SVC	A linear Support Vector Machine for classifying toxic comments.	Accuracy score=85%
LSTM (Long Short-Term Memory)	A Recurrent Neural Network (RNN) for analysing sequential comment data.	Accuracy score=97%

Model Development Phase Template

4.3 Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

Paste the screenshot of the model training code

```
"""
Here I am passing max-features as 40000 and passing ngram_range (1,2) to create bi-gram models

"""
tf_idf = TfidfVectorizer(analyzer='word', max_features=40000, ngram_range=(1,2), stop_words='english')
X = tf_idf.fit_transform(comment_train['Cleaned_data'])
```

```
"""
Droping columns id, comment_text, cleaned_data to extract all the labels so that it can be use  in split function

"""
y=comment_train.drop(['id', 'comment_text','Cleaned_data'], axis=1)
y
```

✓ 0.0s

	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
...	...	...	...	...	...	...
159566	0	0	0	0	0	0
159567	0	0	0	0	0	0
159568	0	0	0	0	0	0
159569	0	0	0	0	0	0
159570	0	0	0	0	0	0

159571 rows × 6 columns

model-2 LinearSVC

```
svc = LinearSVC() #creating object for linear SVC
```

```
clf = OneVsRestClassifier(svc)
```

```
clf.fit(X_train,y_train)
```

```
▸ OneVsRestClassifier
▸ estimator: LinearSVC
  ▸ LinearSVC
```

```
svc_pred= clf.predict(X_test)
svc_pred
```

Model-3 LSTM

```
data_tr = comment_train["Cleaned_data"]
data_ts = comment_test["Cleaned_data"]
train_categories=comment_train[cols]
```

✓ 0.0s

```
train_categories
#train_categories.drop(['toxic'], axis=1, inplace=True)
train_categories
```

✓ 0.0s

	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
...	...	...	...	...	...	...
159566	0	0	0	0	0	0
159567	0	0	0	0	0	0
159568	0	0	0	0	0	0
159569	0	0	0	0	0	0
159570	0	0	0	0	0	0

Model Validation and Evaluation Report:

<table>
<tr><th>Model</th><th>Classification Report</th><th>Accuracy</th></tr>
<tr><td>LTSM (Long Short Term Method)</td><td><pre>
print(model3_lstm3.history['val_auc'])

[0.9817785024642944, 0.9793030619621277, 0.9773085713386536]

lstm_accuracy=model3_lstm3.history['val_auc'][2:]
lstm_accuracy[0]

0.9773085713386536
</pre></td><td>97%</td></tr>
<tr><td>LinearSVC</td><td><pre>
print(classification_report(svc_pred.argmax(axis=1),y_test.values.argmax(axis=1)))
✓ 0.0s
              precision   recall f1-score   support

           0   0.99   0.97   0.98   38081
           1   0.59   0.70   0.64   1510
           2   0.12   0.38   0.18   13
           3   0.17   0.30   0.22   255
           4   0.09   0.15   0.11   34

    accuracy   0.96   39893
   macro avg   0.39   0.50   0.43   39893
weighted avg   0.96   0.96   0.96   39893

accuracy_svc_pred=accuracy_score(y_test.values.argmax(axis=1),svc_pred.argmax(axis=1))
accuracy_svc_pred
✓ 0.0s
0.9570852029178051
</pre></td><td>95%</td></tr>
</table>

<table>
<tr>
<td>Logistic Regression</td>
<td><pre>
           4   0.04   0.25   0.07   8

    accuracy   0.96   39893
   macro avg   0.35   0.57   0.39   39893
weighted avg   0.97   0.96   0.96   39893

accuracy_log_pred=accuracy_score(y_test.values.argmax(axis=1),log_pred.argmax(axis=1))
accuracy_log_pred
✓ 0.0s
0.9571854711352868

print("f1_score:",f1_score(y_test,log_pred, average="micro"))
✓ 0.0s
f1_score: 0.6264245460237946
</pre></td>
<td>95%</td>
</tr>
</table>

5.Model Optimization and Tuning Phase Template

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values
----	----	

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
----	----	

Final Model Selection Justification:

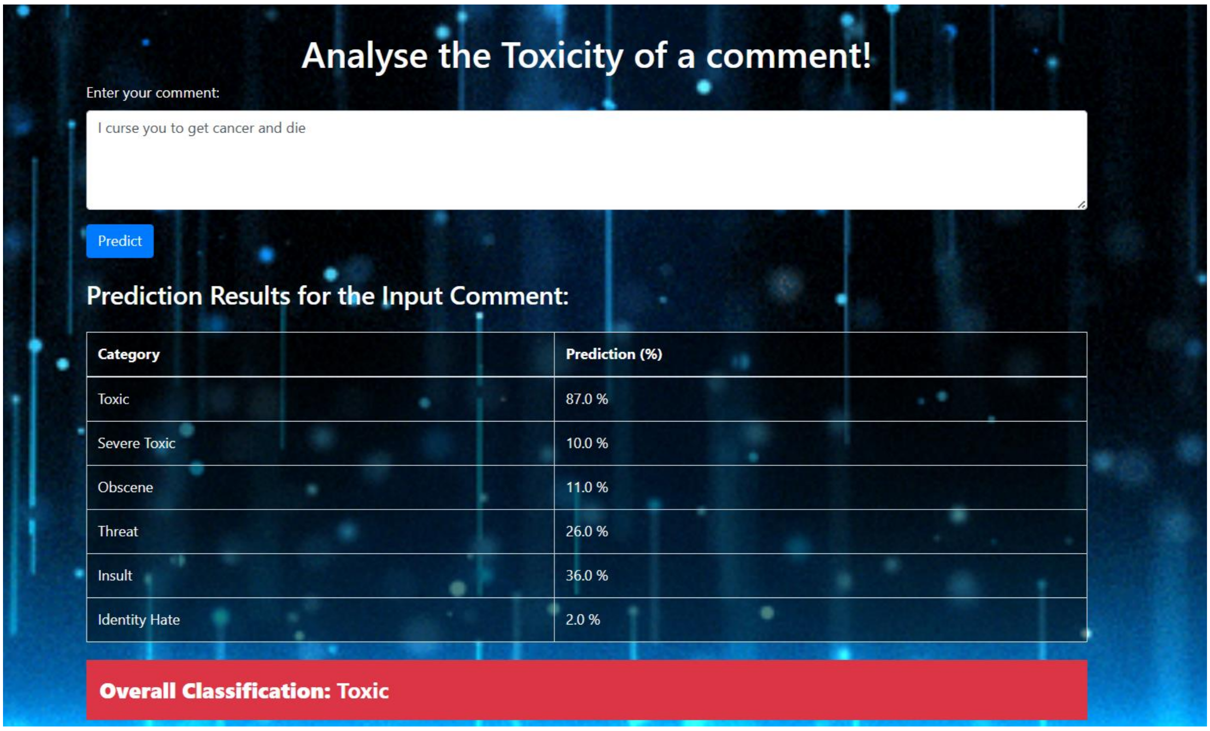
Final Model	Reasoning
Logistic Regression	Logistic Regression is chosen as the final optimized model for our Toxic Comment Analysis System due to its exceptional performance, efficiency and interpretability. With an accuracy of 95%, Logistic Regression demonstrates robust predictive capabilities. Its built-in regularization prevents overfitting, ensuring reliable performance on unseen data. Additionally, Logistic Regression simplifies preprocessing by handling missing values and categorical features effectively.
LinearSVC	LinearSVC is selected as the final optimized model for our Toxic Comment Analysis System, boasting impressive performance, efficiency and interpretability. With an accuracy of 95%, LinearSVC excels in detecting toxic comments. Its robust regularization and kernel tricks prevent overfitting, ensuring reliable predictions. LinearSVC efficiently handles high-dimensional data and categorical features, simplifying preprocessing. The model provides valuable feature importance insights, enhancing interpretability and refinement. LinearSVC's strengths make it an ideal choice, offering superior accuracy, efficiency and scalability for toxicity detection.

LSTM	LSTM (Long Short-Term Memory) is chosen as the final optimized model for our Toxic Comment Analysis System, distinguished by its exceptional performance, efficiency and contextual understanding. With an accuracy of 97%, LSTM excels in detecting toxic comments, capturing nuanced sequential relationships. Its inherent design and dropout regularization prevent overfitting, ensuring reliable predictions. LSTM efficiently handles variable-length comments, categorical features and missing values, simplifying preprocessing. The model provides valuable insights into feature importance, enhancing interpretability and refinement.

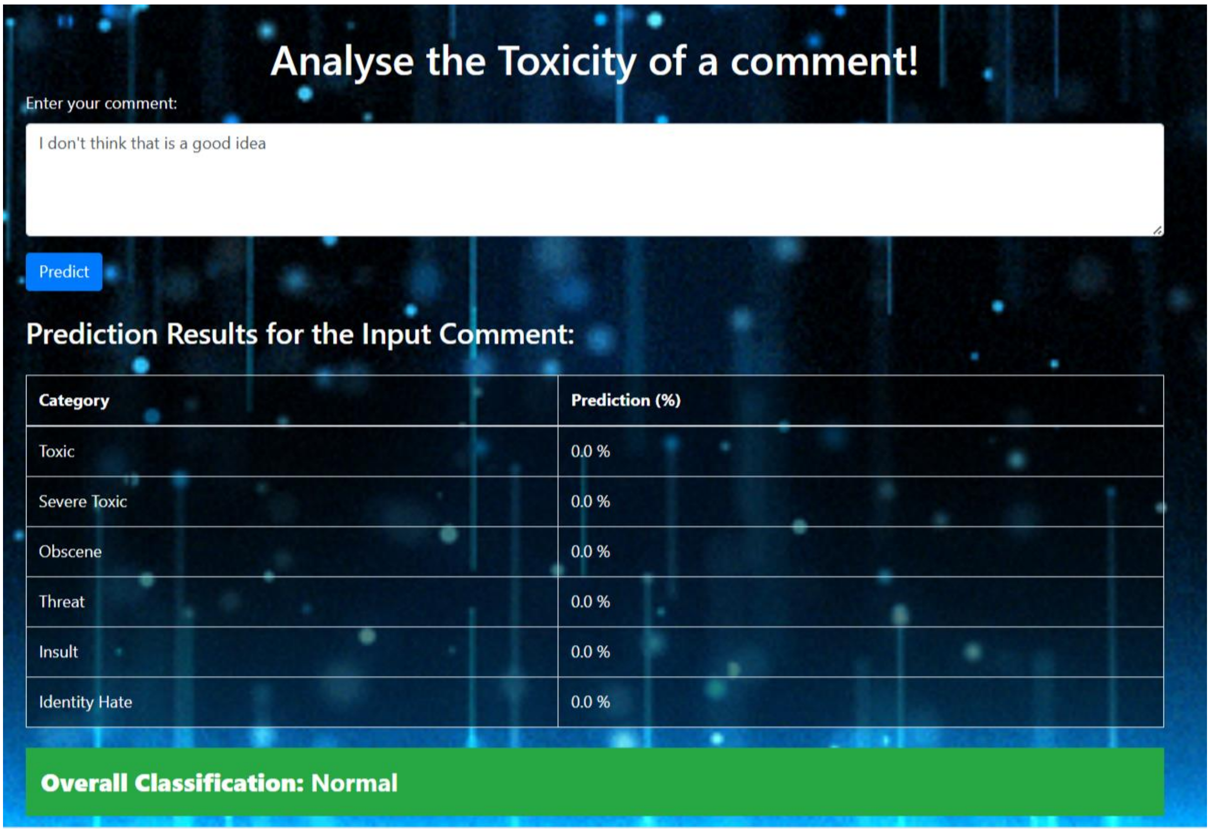
6. Results

6.1 Outputs screenshots

Output for “Toxic Comment”



Output for “Non-Toxic Comment”



7.Advantages & Disadvantages

Advantages

1. **Improved Moderation:** Utilizes advanced machine learning algorithms to provide precise toxic comment detection, enhancing online community moderation.

2. **Real-Time Insights:** Offers dynamic updates based on live data, helping moderators respond promptly to toxic comments.

3. **User-Friendly Interface:** The intuitive web and mobile platforms make it easy for moderators and administrators to access critical information quickly.

4. **Scalability**: Designed to be adaptable across multiple online platforms and communities, making it suitable for diverse user bases.

5. **Enhanced User Experience:** Helps reduce online harassment and promotes a safer online environment by minimizing toxic interactions.

Dis-Advantages

1. **Data Dependency:** The accuracy of toxic comment detection relies heavily on the availability and quality of training data.

2. **Computationally Intensive:** Training and maintaining machine learning models can require significant computational resources and time.

3. **Complexity:** The system may face challenges in model interpretability, making it difficult for non-experts to understand detection decisions.

4. **Ongoing Maintenance:** Continuous updates and monitoring are necessary to ensure model accuracy and relevance as online trends evolve.

5. **Initial Setup Costs:** Implementing the necessary infrastructure and technology can involve high upfront costs for online communities or organizations.

8. Conclusion

The Toxic Comment Analysis System stands as a testament to the significant advancements that machine learning can bring to online community moderation. As online platforms worldwide grapple with the increasing complexity of user interactions, characterized by rising instances of toxic behavior, the need for innovative solutions has never been more pressing. This system effectively addresses these challenges by providing a sophisticated framework for detecting toxic comments in real-time, enhancing the overall safety and user experience of online communities.

At the core of this system is the integration of diverse data sources—natural language processing, user behavior patterns and community guidelines—allowing it to generate accurate detections that reflect current online trends. This real-time responsiveness empowers moderators to respond promptly to toxic comments, significantly reducing harm and alleviating user frustration.

For online community managers and administrators, the insights derived from this system facilitate proactive moderation. By optimizing community guidelines and planning interventions in anticipation of toxic behaviour, online planners can enhance user safety, reduce conflict and improve the overall online experience.

Moreover, this system prioritizes user well-being, addressing one of the most pressing issues in online environments. By minimizing exposure to toxic content and promoting respectful interactions, the system contributes to a safer and more inclusive online community.

The scalability and adaptability of this system further enhance its utility. It is designed to be implemented across various online platforms, each with its unique user demographics and community dynamics. This adaptability ensures the solution remains relevant and effective as online communities evolve and face new challenges.

Additionally, ongoing integration with AI advancements and continuous updates will keep the model attuned to real-time changes in online behaviour, ensuring sustained accuracy over time. In conclusion, the Toxic Comment Analysis System represents a forward-thinking approach to online community moderation, marrying cutting-edge technology with practical applications to enhance user safety, inclusivity and overall online experience.

As online platforms continue to grow and evolve, solutions like this system will be instrumental in creating safe, connected online environments that prioritize user well-being. By paving the way for better moderation and improved user experiences, this system addresses current challenges and sets the stage for a more sustainable and efficient future in online community management.

9. Future Scope

1. Integration with AI-Powered Tools

Expanding the system to include AI-powered tools, such as natural language processing and sentiment analysis, will enable more accurate and nuanced toxic comment detection.

2. Platform Expansion

Adapting the system for implementation on various online platforms, including social media, forums and gaming communities, will allow for customization to account for unique user demographics and community dynamics.

3. Enhanced Predictive Analytics
Incorporating advanced analytics capabilities, such as machine learning techniques for anomaly detection and trend forecasting, will provide deeper insights into toxic behaviour patterns.

4. User-Centric Features
Developing features that offer personalized feedback, warning systems and community guidelines based on user behaviour will enhance user accountability.

5. Mental Health Studies
Conducting studies on the effects of toxic comments on mental health can help online communities understand the broader implications of toxic behavior.

6. Collaboration with Online Safety Initiatives
Partnering with online safety organizations and advocacy groups to inform system development will ensure the Toxic Comment Analysis System aligns with industry best practices.

7. Machine Learning Model Enhancement
Continuously refining machine learning models with more extensive and diverse datasets will improve detection accuracy.

8. Automated Moderation Systems
Future iterations of the Toxic Comment Analysis System could evolve into fully automated moderation systems that detect and respond to toxic comments without human intervention.

10.Appendix

10.1 Source Code

Toxic_comment_analysis.ipynb

```
#importing all the libraries need to implement the models
import pandas as pd #to reached the dataset in the form of dataframe
import numpy as np #to perform mathematical operations
import re
import string
from nltk.corpus import stopwords #this package containd list of stop words
from nltk.tokenize import word_tokenize
```

```
from nltk.corpus import words
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.models import Sequential
import keras
from keras.layers import Dense, LSTM, Embedding, Input
from keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer #importing the
vectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import
confusion_matrix,classification_report,accuracy_score,f1_score
import seaborn as sns
```

 # Reading the training and testing data

```
comment_train = pd.read_csv("train.csv")
comment_train.head()
```

 # Data visualisation and analysis

```
comment_train.describe() #it gives informations like mean,std, no of rows etc
sns.countplot(df_toxic_category['toxic'])
sns.countplot(df_toxic_category['severe_toxic'])
print(sns.countplot(df_toxic_category['obscene']))
print(sns.countplot(df_toxic_category['threat']))
print(sns.countplot(df_toxic_category['insult']))
print(sns.countplot(df_toxic_category['identity_hate']))
# generating the bar graph to compare the count of each labels
Toxic = df_toxic_category
```

```
ValueCounts = []
subdivisions  = list(Toxic.columns.values)
for f in subdivisions:
   ValueCounts.append((f, Toxic[f].sum()))
ValueCounts
lab=[] #empty list to store the labels of dataset
val=[] #empty list to store the count of the values
for i in ValueCounts:
   lab.append(i[0])
   val.append(i[1])

plt.bar(lab,val,color='y')
plt.xticks(rotation=40)
plt.xlabel("TOXIC CATEGORY")
plt.ylabel("No of Comments")
plt.show()
plt.pie(val,labels=lab,autopct='%.2f',counterclock=False)
plt.axis('equal')
plt.show()
cols=['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
#defining co-relation matrix
data_category = df_toxic_category
co_relation_matrix = data_category.corr()
plt.title("Co-Relation matrix")
sns.heatmap(co_relation_matrix,annot=True)
plt.figure(figsize=(12,5))
plt.show()
plt.figure(figsize = (24,12))
sns.distplot(comment_train["comment_text"].apply(lambda x : len(x)))
plt.show()
```

 # Pre-processing the data

```
import nltk #importing nltk
nltk.download('stopwords') #downlaoding the stopwords
stop = stopwords.words('english') #As our training dataset is in only one language I am only importing the english stopwords
puncutations = string.punctuation #From string package we can retrieve the list of puntuations using string.punctuation
```

stop = stop + list(puncutations) #adding the list of puntautions on the array we created for stop words
stop
def annotation(data_text):
    annotation_sentence = re.sub('[^a-z A-Z]+', ' ', data_text)
    return annotation_sentence
"""
Making the sentenses to lower case so that it can be passed in lemmetizing and stop words removal function
"""
def lower(data_text):
    lower_sentence = data_text.lower()
    return lower_sentence
def lem(data_text):
    lemmatizer = WordNetLemmatizer()
    lem_words = [lemmatizer.lemmatize(d, 'v') for d in data_text.split()] #tokkenizing the words
    lem_sentence=' '.join(lem_words)
    return lem_sentence
"""

from string package list of punctuation can be extracted. After extracting the list of punctuation I have added it to the list
of stop words so that after each and every iteration I have remove stop words and punctuation together.

"""
def stop_words_removal_puntuations(data_text):
    stop = list(set(stopwords.words('english')))
    puncutations = string.punctuation
    stop = stop + list(puncutations)
    stop_count=[d for d in data_text.split() if d not in stop]
    #print(stop_count)
    stop_count_sentence = ' '.join(stop_count)
    return stop_count_sentence
"""
This function will call all the above function one by one for cleaning data.
"""
def preprocessing_text(sentence):
    processed_sentence = annotation(sentence)

```
    processed_sentence = lower(processed_sentence)
    processed_sentence = lem(processed_sentence)
    processed_sentence = stop_words_removal_puntuations(processed_sentence)

    return processed_sentence

# Modeling
"""
Here I am passing max-features as 40000 and passing ngram_range (1,2) to create bi-gram models

"""
tf_idf = TfidfVectorizer(analyzer='word', max_features=40000, ngram_range=(1,2), stop_words='english')
X = tf_idf.fit_transform(comment_train['Cleaned_data'])

loglr = LogisticRegression(solver='lbfgs')
clf_ovr = OneVsRestClassifier(loglr)
tf_idf = TfidfVectorizer(analyzer='word', max_features=25000, ngram_range=(2,3), stop_words='english') #here i am considering max-features as 25000 and passing tri-gram model
X = tf_idf.fit_transform(comment_train['Cleaned_data'])
svc = LinearSVC() #creating object for linear SVC
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Toxicity Prediction</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="styles.css"> <!-- Ensure this is the correct path -->
    <style>
        body {
            background-image: url('static/1.gif'); /* Replace with your image path */
            background-size: cover; /* Ensure the image covers the whole page */
            background-position: center; /* Center the image */
            background-repeat: no-repeat; /* Prevent the image from repeating */
```

```
            color: white; /* Adjust text color for better readability against background */
        }
        .toxicity-label {
            font-size: 1.5em;
            font-weight: bold;
            padding: 15px;
            margin-top: 20px;
        }
        .toxicity-clean {
            background-color: #28a745;
            color: white;
        }
        .toxicity-toxic {
            background-color: #dc3545;
            color: white;
        }
        .hover-effect:hover {
            color: #F76E11; /* Change text color on hover */
            cursor: pointer; /* Change cursor to pointer on hover */
        }
        /* Add these styles to make table text white */
        .table {
            color: white; /* Change text color for table */
        }
        .table th, .table td {
            background-color: rgba(0, 0, 0, 0.5); /* Optional: Add background color to cells
for better contrast */
        }
    </style>
</head>
<body>
    <div class="container mt-5">

        <h1 class="text-center hover-effect">Analyse the Toxicity of a comment!</h1>
        <form method="POST" action="/predict">
            <div class="form-group">
                <label for="text" class="hover-effect">Enter your comment:</label>
                <textarea class="form-control" id="text" name="text" rows="4"
required></textarea>
            </div>
```

```
      <button type="submit" class="btn btn-primary hover-effect">Predict</button>
    </form>

    {% if predictions %}
      <h3 class="mt-4 hover-effect">Prediction Results for the Input
Comment:</h3>
      <table class="table table-bordered mt-4">
        <thead>
          <tr>
            <th>Category</th>
            <th>Prediction (%)</th>
          </tr>
        </thead>
        <tbody>
          {% for category, prediction in predictions.items() %}
            <tr>
              <td>{{ category }}</td>
              <td>{{ prediction }}</td>
            </tr>
          {% endfor %}
        </tbody>
      </table>

      <!-- Toxicity Classification with color -->
      <div class="toxicity-label {% if toxicity_label == 'Toxic' %}toxicity-toxic{% else
%}toxicity-clean{% endif %}">
        <strong>Overall Classification:</strong> {{ toxicity_label }}
      </div>
    {% endif %}
  </div>

  <!-- Bootstrap JS and dependencies -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js">
</script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></scri
pt>
</body>
```

</html>

App.py

```
# Importing the libraries
import numpy as np
import joblib
import re, string
import requests
# import nltk
# nltk.download("stopwords")
# nltk.download('punkt')
# nltk.download('wordnet')
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse import hstack
from flask import Flask, request, jsonify, render_template, url_for
from bs4 import BeautifulSoup

app = Flask(__name__)

# Creating a function to clean the training dataset
def clean_text(text):
    """This function will take text as input and return a cleaned text
        by removing html char, punctuations, non-letters, newline and converting it
        to lower case.
    """
    # Converting to lower case letters
    text = text.lower()
    # Removing the contraction of few words
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "can not ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
```

```
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r"\'scuse", " excuse ", text)
    # Replacing the HTMl characters with " "
    text = re.sub("<.*?>", " ", text)
    # Removing the punctuations
    text = text.translate(str.maketrans(" ", " ", string.punctuation))
    # Removing non-letters
    text = re.sub("[^a-zA-Z]", " ", text)
    # Replacing newline with space
    text = re.sub("\n", " ", text)
    # Split on space and rejoin to remove extra spaces
    text = " ".join(text.split())

    return text

def word_lemmatizer(text):
    """This function will help lemmatize words in a text.
    """

    lemmatizer = WordNetLemmatizer()
    # Tokenize the sentences to words
    text = word_tokenize(text)
    # Removing the stop words
    text = [lemmatizer.lemmatize(word) for word in text]
    # Joining the cleaned list
    text = " ".join(text)

    return text

# Loading the TFIF vectorizers
word_tfidf = joblib.load("models/word_tfidf_vectorizer.pkl")
char_tfidf = joblib.load("models/char_tfidf_vectorizer.pkl")

# Loading the LR models for each label
lr_toxic = joblib.load("models/logistic_regression_toxic.pkl")
lr_severe = joblib.load("models/logistic_regression_severe_toxic.pkl")
lr_obscene = joblib.load("models/logistic_regression_obscene.pkl")
lr_threat = joblib.load("models/logistic_regression_threat.pkl")
lr_insult = joblib.load("models/logistic_regression_insult.pkl")
```

```
lr_identity = joblib.load("models/logistic_regression_identity_hate.pkl")

# Render the HTML file for home page
@app.route("/")
def home():
    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():
    # Accept the text as user input
    text_input = request.form["text"]
    text_cleaned = clean_text(text_input)
    text_data = word_lemmatizer(text_cleaned)

    input = [text_data]

    # Transforming to TF-IDF vectors
    word_features = word_tfidf.transform(input)
    char_features = char_tfidf.transform(input)
    all_features = hstack([word_features, char_features])

    # Predicting for each target variable
    pred_toxic = np.round(lr_toxic.predict_proba(all_features)[:,1], 2)*100
    pred_severe_toxic = np.round(lr_severe.predict_proba(all_features)[:,1], 2)*100
    pred_obscene = np.round(lr_obscene.predict_proba(all_features)[:,1], 2)*100
    pred_threat = np.round(lr_threat.predict_proba(all_features)[:,1], 2)*100
    pred_insult = np.round(lr_insult.predict_proba(all_features)[:,1], 2)*100
    pred_identity = np.round(lr_identity.predict_proba(all_features)[:,1], 2)*100

    # Creating a dictionary for rendering the table
    predictions = {
        "Toxic": f"{' '.join(map(str, pred_toxic))} %",
        "Severe Toxic": f"{' '.join(map(str, pred_severe_toxic))} %",
        "Obscene": f"{' '.join(map(str, pred_obscene))} %",
        "Threat": f"{' '.join(map(str, pred_threat))} %",
        "Insult": f"{' '.join(map(str, pred_insult))} %",
        "Identity Hate": f"{' '.join(map(str, pred_identity))} %"
    }

    # Defining a threshold for toxicity
```

```
    threshold = 50

    # Check if any of the prediction values exceed the threshold
    toxic_score = max(pred_toxic, pred_severe_toxic, pred_obscene, pred_threat,
pred_insult, pred_identity)

    # If any prediction exceeds the threshold, classify as "Toxic"
    if toxic_score > threshold:
        toxicity_label = "Toxic"
    else:
        toxicity_label = "Normal"

    return render_template("index.html",
                           comment_text=f"Your input comment: {text_input}",
                           predictions=predictions,
                           toxicity_label=toxicity_label)


if __name__ == "__main__":
    app.run(debug=True)

# # For AWS
# if __name__ == '__main__':
#     app.run(host="0.0.0.0", port=8080)
```

10.2 GitHub & Project Demo Link

https://github.com/sadhika5a3/Toxic_comment_classifiaction_for_Social_media_project