

# **TOXIC COMMENTS CLASSIFICATION**

## **IN SOCIAL NETWORKING**

Submitted By

**GUNDUMOGULA SADHU SUNDHAR**

**39110360**

# **INDEX**

## **1. Introduction**

- a. Overview
- b. Purpose

## **1. Literature Survey**

- a. Existing Problem
- b. Proposed solution

## **2. Theoretical Analysis**

- a. Block Diagram
- b. Hardware/Software designing

## **3. Experimental investigations**

## **4. Flowchart**

## **5. Result**

## **7. Advantages & Disadvantages**

## **1. Applications**

## **2. Conclusion**

## **3. Future Scope**

## **4. Bibliography**

## **12. Appendix**

- a. Source Code
- b. UI output screen-shot

## **1. INTRODUCTION**

### **a. Overview:**

In recent years, the flow of data over the internet has grown dramatically, especially with the appearance of social networking sites. Social networks sometimes become a place for threats, insults, and other components of cyber bullying. A huge number of people are involved in online social networks.

Toxic comments are textual comments with threats, insults, obscene, racism, etc. In recent years there have been many cases in which authorities have arrested some users of social sites because of the negative (abusive) content of their pages. Hence, the protection of network users from anti-social behaviour is an important activity. One of the major tasks of such activity is automated detecting the toxic comments.

The project aims to build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate, we will use a dataset of comments from Wikipedia talk page edits, collected by Kaggle. Improvements to the current model

### **b. Purpose:**

We'll be able to understand the problem to classify if it is a toxic comment or a normal comment. We will be able to know how to pre-process/clean the data using different data pre-processing techniques. You will be able to analyse or get insights into data through visualization. Applying different algorithms according to the dataset and based on visualization. We will be able to know how to build a web application using the Flask framework

## **2.LITERATURE SURVEY**

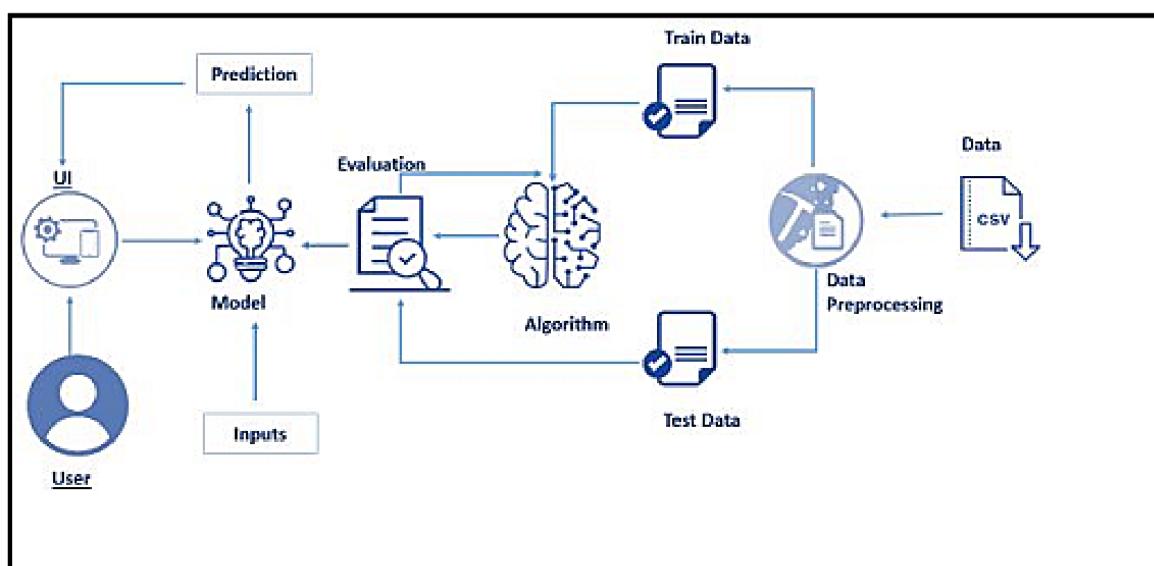
Bag of words statistics and bag of symbols statistics are the typical source information for the toxic comment's detection. Usually, the following statistics-based features are used: length of the comment, number of capital letters, number of exclamation marks, number of question marks, number of spelling errors, number of tokens with non-alphabet symbols, number of abusive, aggressive, and threatening words in the comment, etc.

## b. Proposed Solution :

The goal is to create a classifier model that can predict if input text is inappropriate (toxic). Explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments. Create a baseline score with a simple logistic regression classifier. Explore the effectiveness of multiple machine learning approaches and select the best for this problem. Select the best model and tune the parameters to maximize performance. Build a the final model with the best performing algorithm and parameters and test it on a holdout subset of the data.

## 2. THEORETICAL ANALYSIS

### 3.1 Block Diagram

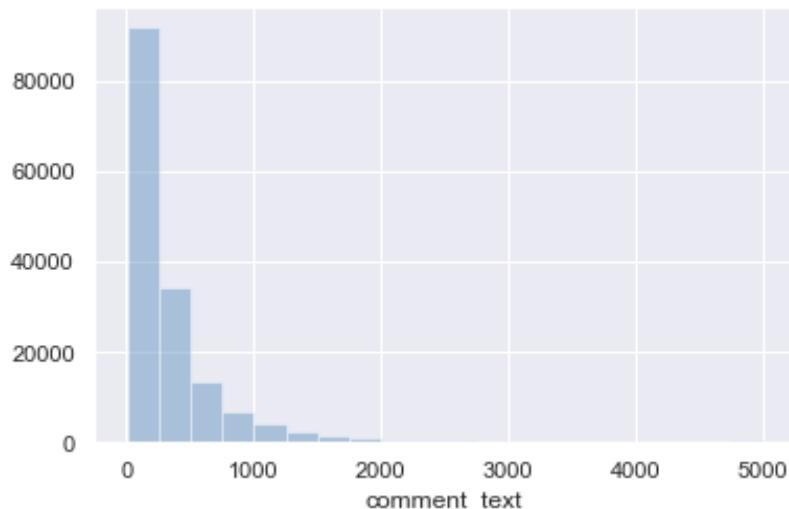


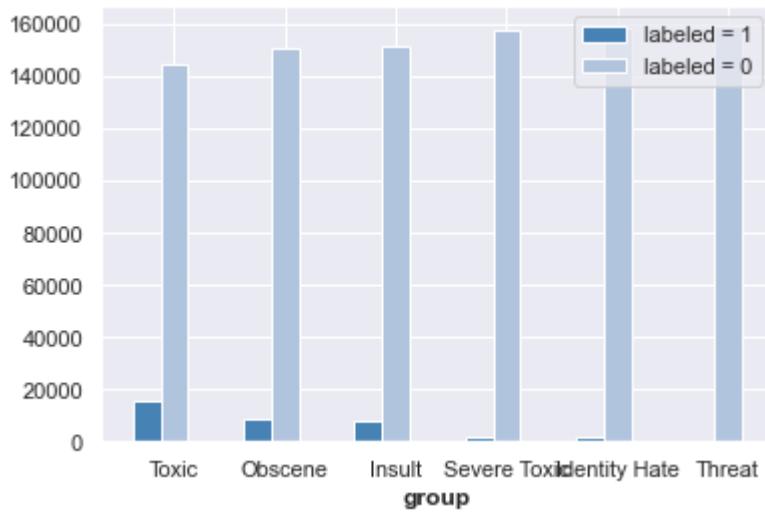
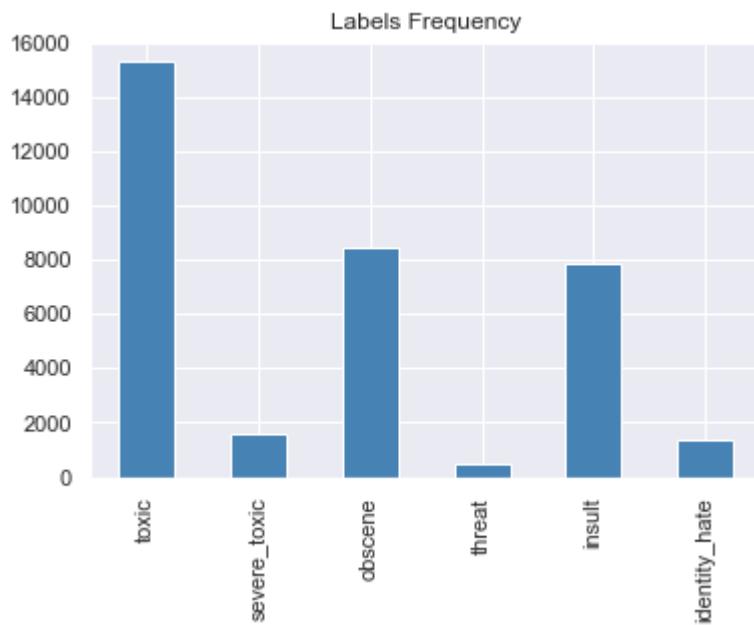
## b. Hardware/Software Requirements

1. IDE: Jupyter Notebook, Spyder, Anaconda navigator
2. Programming Language (Back-end): Python 3.7
3. Front-end: HTML, CSS
4. Framework: Flask

## 4. EXPERIMENTAL INVESTIGATIONS

**Data Exploration** This dataset contains 159,571 comments from Wikipedia. The data consists of one input feature, the string data for the comments, and six labels for different categories of toxic comments: toxic, severe toxic, obscene, threat, insult, and identity hate. The figure on the following page contains a breakdown of how the labels are distributed throughout the dataset, including overlapping data. As you can see in the breakdown, while most comments with other labels are also toxic, not all of them are. Only “severe toxic” is clearly a subcategory of “toxic.” And it’s not close enough to be a labelling error. This suggests that “toxic” is not a catch-all label, but rather a subcategory in itself with a large amount of overlap. Because of this, I’m going to create a seventh label called “any\_label” to represent overall toxicity of a comment. From here on in, I’m going to refer to any labelled comments as toxic, and the specific “toxic” label (along with other labels) in quotation marks.



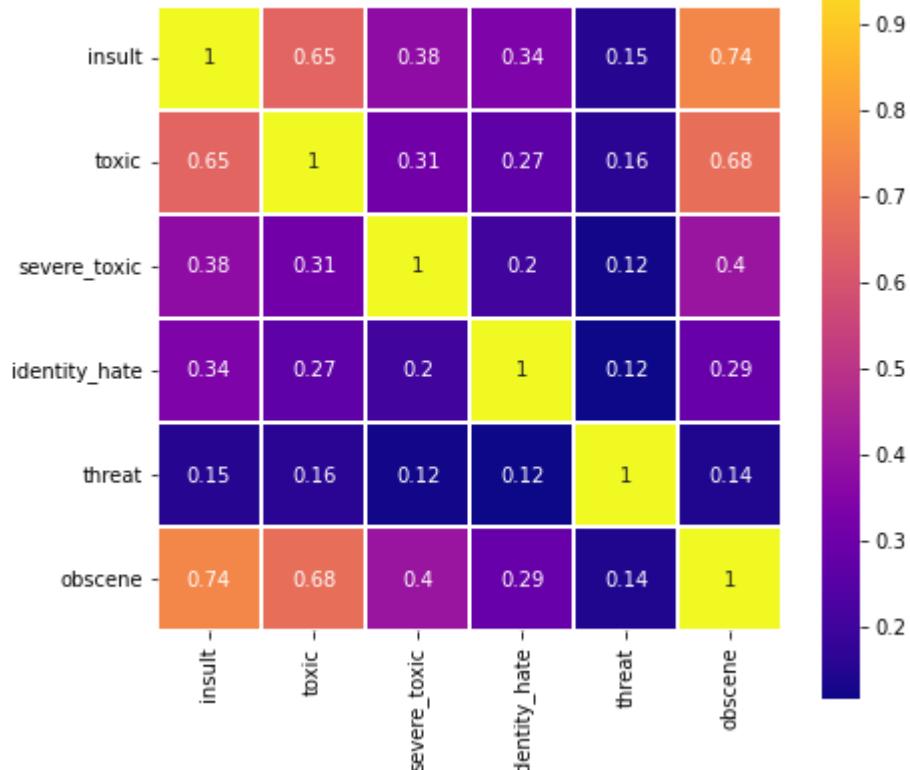


Only 39 percent of the toxic comments have only one label, and the majority have some sort of overlap. I believe that because of this, it will be much more difficult to train a classifier on specific labels than whether they are toxic. This ambiguity and the lack of explanation around it is what led me to select an aggregate label of general toxicity, what I've called "any\_label," as the target. Sixteen thousand two hundred twenty-five out of 159571 comments, or 10.17%, are classified as some category of toxic.

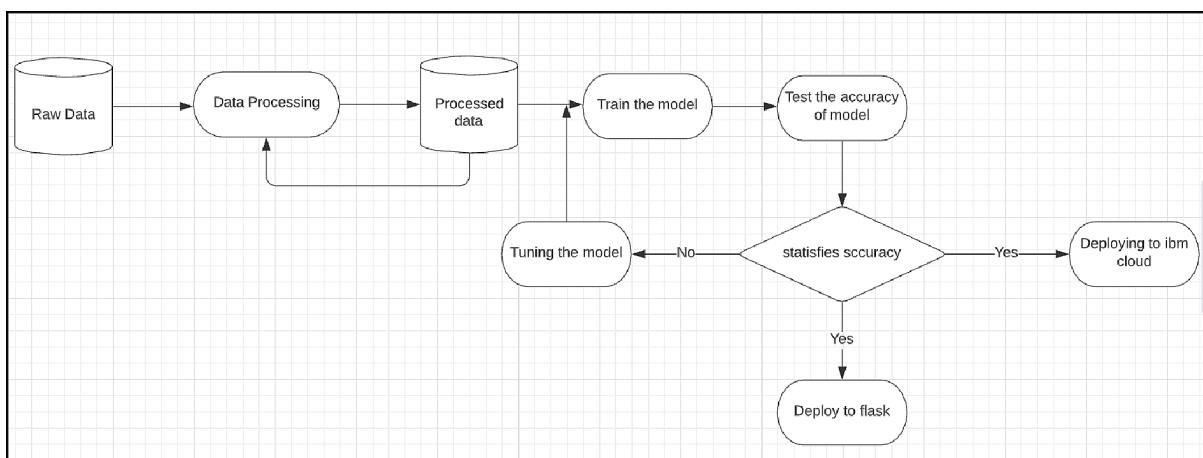
The correlation matrix below provides more insight into these overlapping categories. Threats are not likely to be severely toxic, nor are they likely to be racist or homophobic. But

insults are often obscene, and identity hate really doesn't have much overlap at all.

Correlation of features & targets

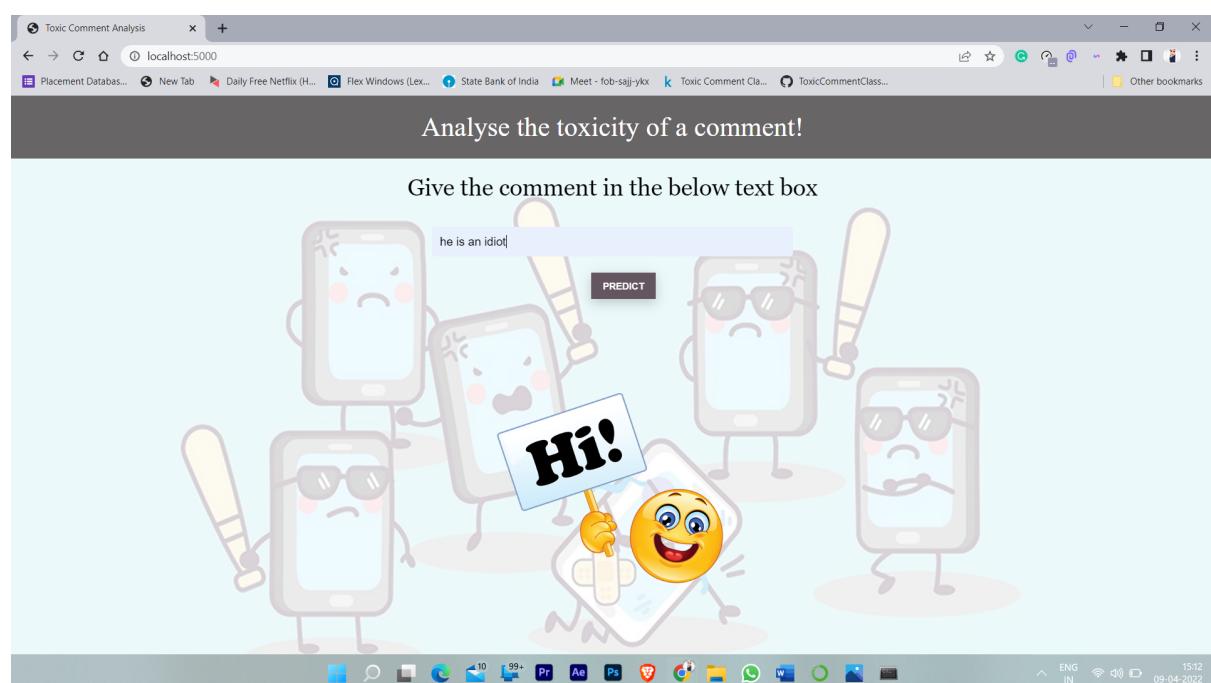
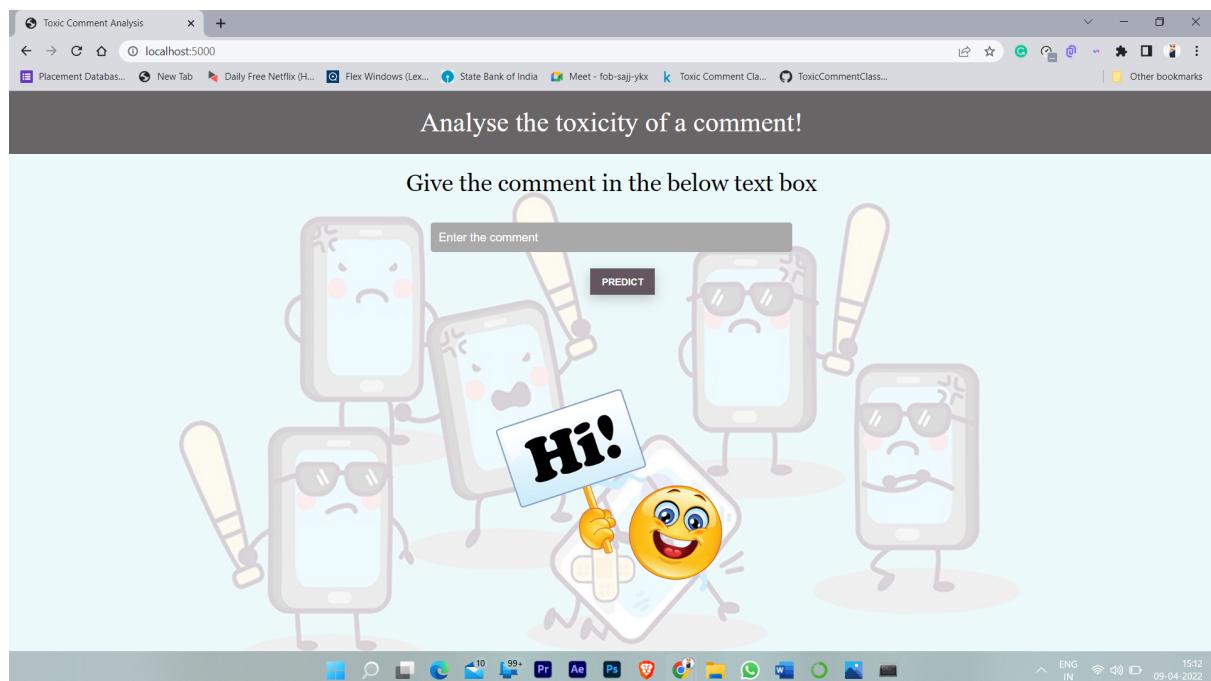


## 5. FLOW CHART



## 6. RESULT

We have successfully built the UI interface for classification of toxic comments, in which user can input any comment and then while pressing predict the user can find the comment is toxic or clean , this UI will help them to find the comment is toxic or clean .



## **7. ADVANTAGES**

1. The interface is user friendly and easy to use and understand even to a person who has very little/no knowledge .
2. This model uses a huge amount of data the results generated are accurate and reliable.
3. Easy to use & has a user-friendly interface.
4. Results can be improved by training data to our choice of parameter.
5. We can easily do the classification of different comments whether the comment is toxic or clean using natural language processing.

## **DISADVANTAGES**

1. we can't predict the comment if it is different language
2. the modal can't determine properly if the comment is big

## **8. APPLICATIONS**

1. using this modal we can identify the toxic comments
2. Time saving & cost-efficient method as we don t need to read every comment
3. by using this modal we can skip the toxic comments
4. As the same way we can find the person who handles the toxic comments and we can block them
5. using this modal we can clear the unwanted and toxic comments
6. By clearing the waste/toxic data we can reduce the data storage.
7. As this modal is reliable we can use in different sectors to classified the comments.

## 9. CONCLUSION

A Machine Learning model, has been developed to **classification of toxic comments in social networking**. A simple, efficient and a versatile model is built keeping in mind the diversity of data, computational complexity and overhead involved in making API calls to the model for prediction. The product can increase the accuracy of finding the toxic comments in social network. It helps to classified the comments and identify the toxic comments.

## 10. FUTURE SCOPE

1. Recurrent neural networks, despite their increased overhead, could be a very effective solution if GPU resources are available for quick predictions
2. We could use a simple decision tree to feed comments into the model that would be most effective. A few that I can think of are:
  - Short comments
  - Long comments
  - “Hot” threads where the rate of commenting is high and emotions may be high

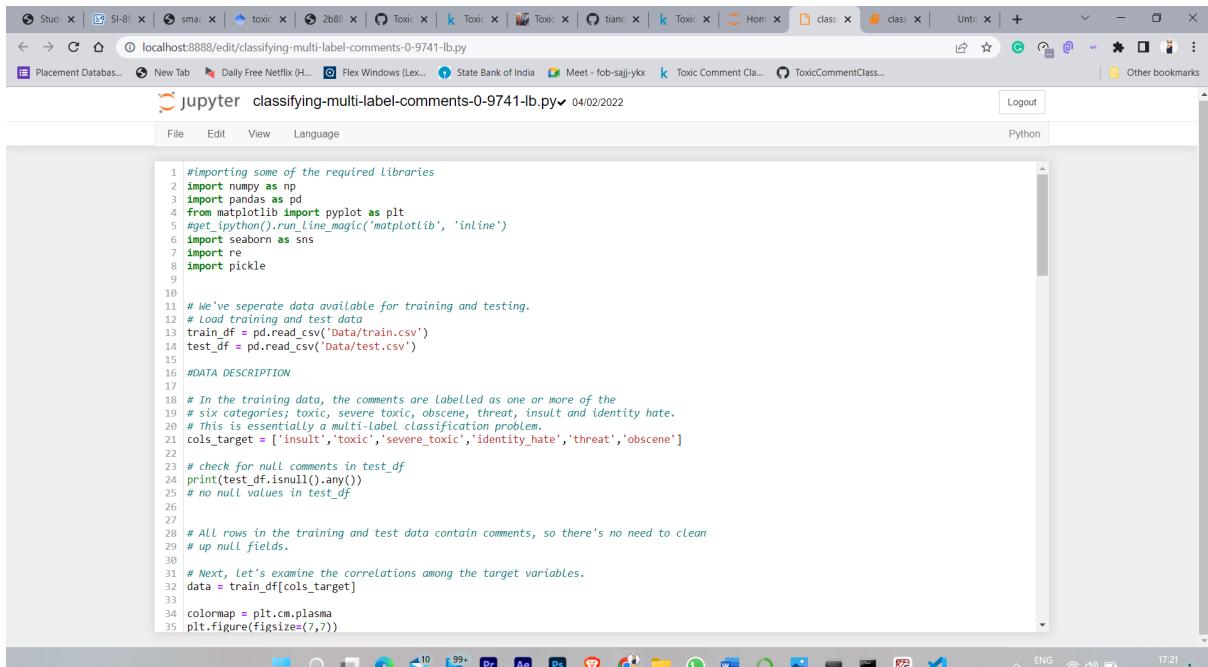
## 11. BIBLIOGRAPHY

- DATASET- <https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data>
- Toxic Comment Classification Group Project for MSDS621 Machine Learning at University of San Francisco

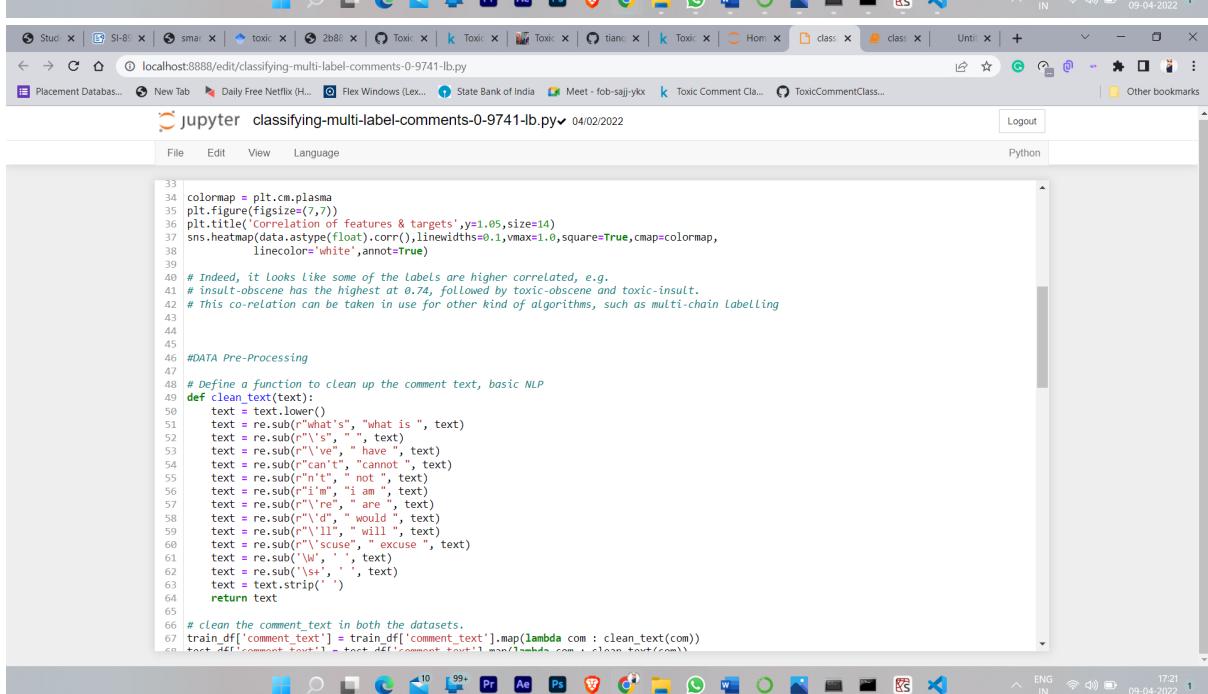
## 12. APPENDIX

### a. Source code:

#### classifying-multi-label-comments-0-9741-lb.py



```
1 #importing some of the required Libraries
2 import numpy as np
3 import pandas as pd
4 from matplotlib import pyplot as plt
5 #get_ipython().run_line_magic('matplotlib', 'inline')
6 import seaborn as sns
7 import re
8 import pickle
9
10 # We've separate data available for training and testing.
11 # Load training and test data
12 train_df = pd.read_csv('Data/train.csv')
13 test_df = pd.read_csv('Data/test.csv')
14
15 #DATA DESCRIPTION
16
17 # In the training data, the comments are labelled as one or more of the
18 # six categories; toxic, severe_toxic, obscene, threat, insult and identity hate.
19 # This is essentially a multi-label classification problem.
20 cols_target = ['insult','toxic','severe_toxic','identity_hate','threat','obscene']
21
22 # check for null comments in test_df
23 print(test_df.isnull().any())
24 # no null values in test_df
25
26
27 # All rows in the training and test data contain comments, so there's no need to clean
28 # up null fields.
29
30 # Next, let's examine the correlations among the target variables.
31 data = train_df[cols_target]
32
33 colormap = plt.cm.plasma
34 plt.figure(figsize=(7,7))
```



```
35
36 sns.heatmap(data.astype(float).corr(), linewidths=0.1, vmax=1.0, square=True, cmap=colormap,
37             linecolor='white', annot=True)
38
39 # Indeed, it looks like some of the labels are higher correlated, e.g.
40 # insult-obscene has the highest at 0.74, followed by toxic-obscene and toxic-insult.
41 # This co-relation can be taken in use for other kind of algorithms, such as multi-chain Labelling
42
43
44
45
46 #DATA Pre-Processing
47
48 # Define a function to clean up the comment text, basic NLP
49 def clean_text(text):
50     text = text.lower()
51     text = re.sub(r"what's", " what is ", text)
52     text = re.sub(r"\'", " ', ", text)
53     text = re.sub(r"\'ve", " have ", text)
54     text = re.sub(r"\can't", " cannot ", text)
55     text = re.sub(r"\n't", " not ", text)
56     text = re.sub(r"\i m", " i am ", text)
57     text = re.sub(r"\re", " are ", text)
58     text = re.sub(r"\d", " would ", text)
59     text = re.sub(r"\ll", " will ", text)
60     text = re.sub(r"\scuse", " excuse ", text)
61     text = re.sub(r"\W", " ', ", text)
62     text = re.sub(r"\s+", " ', ", text)
63     text = text.strip(' ')
64     return text
65
66 # clean the comment_text in both the datasets.
67 train_df['comment_text'] = train_df['comment_text'].map(lambda com : clean_text(com))
68 test_df['comment_text'] = test_df['comment_text'].map(lambda com : clean_text(com))
```

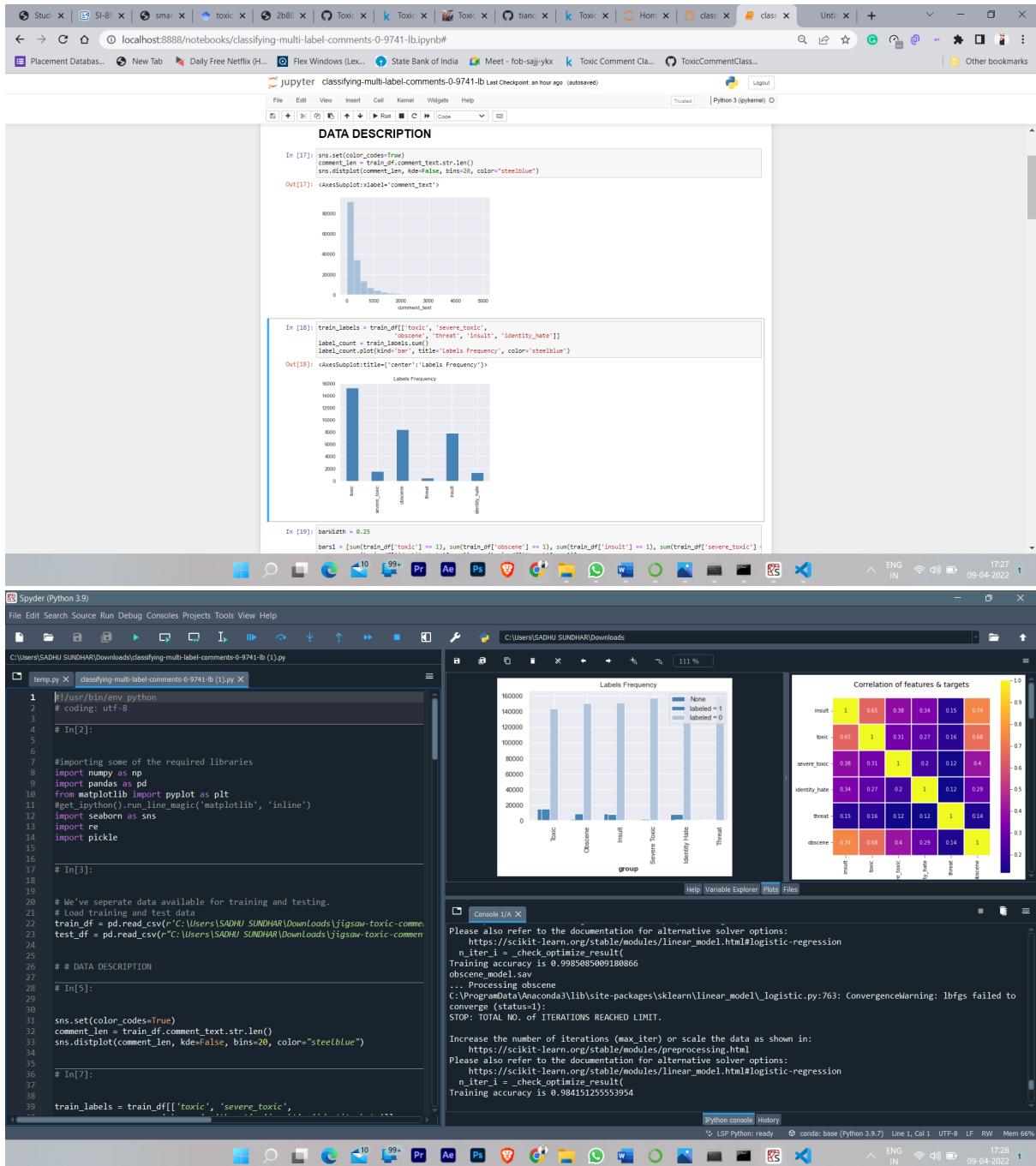
The image shows two screenshots of a Jupyter Notebook interface running on a Windows operating system. Both screenshots have a similar layout with a top navigation bar, a code editor area, and a bottom taskbar.

**Screenshot 1 (Top):**

```
65 # clean the comment_text in both the datasets.
66 train_df['comment_text'] = train_df['comment_text'].map(lambda com : clean_text(com))
67 test_df['comment_text'] = test_df['comment_text'].map(lambda com : clean_text(com))
68
69 # Define all_text from entire train & test data for use in tokenization by Vectorizer
70 train_text = train_df['comment_text']
71 test_text = test_df['comment_text']
72 all_text = pd.concat([train_text, test_text])
73
74 # Vectorize the data
75 # import and instantiate CountVectorizer
76 from sklearn.feature_extraction.text import CountVectorizer
77 word_vect = CountVectorizer(
78     strip_accents='unicode',
79     analyzer='word',
80     token_pattern=r'(\\w+)', 
81     stop_words='english',
82     ngram_range=(1, 1)
83 )
84
85
86
87 # Learn the vocabulary in the training data, then use it to create a document-term matrix
88 word_vect.fit(all_text)
89
90 # transform the data using the earlier fitted vocabulary, into a document-term matrix
91 train_features = word_vect.transform(train_text)
92 test_features = word_vect.transform(test_text)
93
94
95 #saving word vectorizer vocab as pkl file to be loaded afterwards
96 pickle.dump(word_vect.vocabulary_,open("word_feats.pkl",'wb'))
97
98
99
```

**Screenshot 2 (Bottom):**

```
120 # Model Building and Fitting
121
122 # import and instantiate the Logistic Regression model
123 from sklearn.linear_model import LogisticRegression
124 from sklearn.metrics import accuracy_score
125 logreg = LogisticRegression(C=12.0)
126
127 # create submission file
128 submission_binary = pd.read_csv('sample_submission.csv')
129
130 mapper = {}
131 for label in cols_target:
132     mapper[label] = logreg
133     filename = str(label+'_model.sav')
134     print(filename)
135     print('... Processing {}'.format(label))
136     y = train_df[label]
137     # train the model using train_features & y
138     mapper[label].fit(train_features, y)
139
140 # saving the fitted model for class "label"
141 pickle.dump(mapper[label], open(filename, 'wb'))
142
143 # compute the training accuracy
144 y_pred_X = mapper[label].predict(train_features)
145 print('Training accuracy is {}'.format(accuracy_score(y, y_pred_X)))
146 # compute the predicted probabilities for X_test dtm
147 test_y_prob = mapper[label].predict_proba(test_features)[:,1]
148 submission_binary[label] = test_y_prob
149
150
151 # generate submission file
152 submission_binary.to_csv('submission_binary.csv', index=False)
153
154
```



## testing comment toxicity.py

```
C:\Users\SADHU SUNDHAR>Desktop>sadhu pt2> testing comment toxicity.py > clean_text
1 import numpy as np
2 import pandas as pd
3 import re
4 from sklearn.feature_extraction.text import CountVectorizer
5 import pickle
6
7 loaded=CountVectorizer(decode_error='replace',vocabulary=pickle.load(open('word_features.pkl','rb')))
8
9 #test_comment='Food was really boring'
10 #test_comment=test_comment.split('delimiter')
11 #result=loaded.transform(test_comment)
12 #print(result)
13
14 #train_df = pd.read_csv('train.csv')
15 #test_df = pd.read_csv('test.csv')
16
17 user_df = pd.DataFrame(columns = ['comment_text'])
18 user_df
19
20 inp_comment = str(input("enter a comment : "))
21 inp_comment
22
23 new_row = {'comment_text':inp_comment}
24 #append a new row to the dataframe formed by user inputs
25 user_df = user_df.append(new_row,ignore_index = True)
26
27
28 def clean_text(text):
29     text = text.lower()
30     text = re.sub(r"What's", " what is ", text)
31     text = re.sub(r"\'s", " ", text)
32     text = re.sub(r"\ve", " have ", text)
33     text = re.sub(r"can't", " cannot ", text)
34     text = re.sub(r"n't", " not ", text)
35     text = re.sub(r"I'm", " i am ", text)
36     text = re.sub(r"\re", " are ", text)
37     text = re.sub(r"\d", " would ", text)
38
39     text = re.sub(r"\d", " would ", text)
40     text = re.sub(r"\ll", " will ", text)
41     text = re.sub(r"\scuse", " excuse ", text)
42     text = re.sub(r'\W', ' ', text)
43     text = re.sub(r"\s+", ' ', text)
44     text = text.strip(' ')
45
46     # clean the comment_text in train_df [Thanks to Pukkit Jha for the useful pointer.]
47     #train_df['comment_text'] = train_df['comment_text'].map(lambda com : clean_text(com))
48
49     # clean the comment_text in test_df [Thanks, Pukkit Jha.]
50     #test_df['comment_text'] = test_df['comment_text'].map(lambda com : clean_text(com))
51
52     # clean the comment_text in user_df [Thanks, Pukkit Jha.]
53     user_df['comment_text'] = user_df['comment_text'].map(lambda com : clean_text(com))
54     user_df
55
56     #train_text = train_df['comment_text']
57     #test_text = test_df['comment_text']
58     user_text = user_df['comment_text']
59
60
61     #train_features = loaded.transform(train_text)
62     # transform the test data using the earlier fitted vocabulary, into a document-term matrix
63     #test_features = loaded.transform(test_text)
64
65     user_features = loaded.transform(user_text)
66     print(user_features)
67
68     cols_target = ['obscene', 'insult', 'toxic', 'severe_toxic', 'identity_hate', 'threat']
69
70     from sklearn.linear_model import LogisticRegression
71     from sklearn.metrics import accuracy_score
72
73     #create submission file
```

The screenshot displays two separate Microsoft Windows desktop environments, each featuring a Visual Studio Code (VS Code) window. The top window shows a Python script named `testing comment toxicity.py`. The script reads a file named `#SUBMISSION_COLUMNS.csv` and processes comments based on toxicity levels. It uses a trained model to predict probabilities for labels like 'obscene', 'insult', 'toxic', 'severe\_toxic', 'identity\_hate', and 'threat'. If the probability for any of these labels is above 0.5, it is considered toxic. The output text is either "The comment is clean" or "the comment is toxic". The bottom window shows a terminal session running the same script. The terminal output shows the script's execution and the resulting toxicity predictions for various input comments. Both windows have standard Windows taskbars at the bottom.

```
File Edit Selection View Go Run Terminal Help testing comment toxicity.py - Visual Studio Code

Get Started < testing comment toxicity.py >
C:\Users>SADHU SUNDHAR>Desktop>sadhu pt2 > testing comment toxicity.py > clean_text
76
77 1st= []
78 mapper = {}
79 for label in cols_target:
80     filename = str(label+'_.model.sav')
81     filename
82     model = pickle.load(open(filename, 'rb'))
83
84     print('... Processing {}'.format(label))
85     user_y_prob = model.predict_proba(user_features)[:,1]
86     print(label,":",user_y_prob[0])
87     1st.append([label,user_y_prob])
88 #submission_binary[label] = test_y_prob
89 print(1st)
90 #[['obscene', array([0.05674509])],
91 #[['insult', array([0.07231278])], ['toxic', array([0.10434529])], ['severe_toxic', array([0.01385649])], ['identity_hate', array([0.01463043])], ['threat', array([0.0048
92
93 final=[]
94 for i in 1st:
95     if i[1]>0.5:
96         final.append(i[0])
97
98 if not len(final):
99     text = "The comment is clean"
100 else:
101     text= "the comment is toxic"
102
103 print(text)
104
105
106
107
108
109
110
111
112

Ln 36, Col 42 Spaces: 4 UTF-8 CRLF Python 3.9.7 (base:conda) 17:42 09-04-2022 1
```

```
File Edit Selection View Go Run Terminal Help testing comment toxicity.py - Visual Studio Code

Get Started < testing comment toxicity.py >
C:\Users>SADHU SUNDHAR>Desktop>sadhu pt2 > testing comment toxicity.py > ...
1 import numpy as np
2 import pandas as pd
3 import re
4 from sklearn.feature_extraction.text import CountVectorizer
5 import pickle
6
7 loaded=CountVectorizer(decode_error='replace',vocabulary=pickle.load(open('word_features.pkl','rb')))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\SADHU SUNDHAR & C:/ProgramData/Anaconda3/python.exe "c:/Users/SADHU SUNDHAR/Desktop/sadhu pt2/testing comment toxicity.py"
enter a comment : he is an idiot
(0, 104085) 1
... Processing obscene
obscene : 0.20713966072594417
... Processing insult
insult : 0.7006829806577194
... Processing toxic
toxic : 0.9858620000000001
... Processing severe_toxic
severe_toxic : 0.021324154969935092
... Processing identity_hate
identity_hate : 0.013324584695474614
... Processing threat
threat : 0.0017120708631557866
[['obscene', array([0.20713966]), ['insult', array([0.70068298])], ['toxic', array([0.98586200])], ['severe_toxic', array([0.02132415])], ['identity_hate', array([0.01332458])], ['thr
eat', array([0.00171207])]]
the comment is toxic
PS C:\Users\SADHU SUNDHAR >
```

LN 4, COL 60 SPACES: 4 UTF-8 CRLF PYTHON 3.9.7 (BASE:CONDA) 17:46 09-04-2022 1

## commentApp.py

```
C:\> Users > SADHU SUNDHAR > Desktop > pt 2 > Commit-Toxicity-Multi-Class-Classification-main > Commit-Toxicity-Multi-Class-Classification-main > Flask > commentApp.py > ...  
1 import numpy as np  
2 import pandas as pd  
3 import re  
4 from sklearn.feature_extraction.text import CountVectorizer  
5 import pickle  
6 from sklearn.linear_model import logisticRegression  
7 from sklearn.metrics import accuracy_score  
8 from flask import Flask, request, jsonify, render_template, url_for  
9  
10 loaded=CountVectorizer(decode_error='replace',vocabulary=pickle.load(open('word_features.pkl','rb')))  
11  
12 app = Flask(__name__)  
13  
14 def clean_text(text):  
15     text = text.lower()  
16     text = re.sub(r'what\'s', "what is ", text)  
17     text = re.sub(r'\s+', " ", text)  
18     text = re.sub(r've', " have ", text)  
19     text = re.sub(r'can\'t', "cannot ", text)  
20     text = re.sub(r'n\'t', " not ", text)  
21     text = re.sub(r'i\'m", " i am ", text)  
22     text = re.sub(r'\re', " are ", text)  
23     text = re.sub(r'\d', " would ", text)  
24     text = re.sub(r'\ll', " will ", text)  
25     text = re.sub(r'\scuse", " excuse ", text)  
26     text = re.sub('W', ' ', text)  
27     text = re.sub('\'', ' ', text)  
28     text = text.strip('')  
29  
30     return text  
31  
32 @app.route('/')  
33 def landingpage():  
34     img_url = url_for('static', filename = 'images/hello.png')  
35     print(img_url)  
36     flag=0  
37     return render_template('toxic.html', flag=flag)  
Ln 1, Col 1 | Spaces: 4 | UTF-8 | CRLF | Python | 3.9.7 (base:conda) | 17:52 | 09-04-2022 |  
  
C:\> Users > SADHU SUNDHAR > Desktop > pt 2 > Commit-Toxicity-Multi-Class-Classification-main > Commit-Toxicity-Multi-Class-Classification-main > Flask > commentApp.py > ...  
37  
38 @app.route('/predict')  
39 @app.route('/', methods = ['GET','POST'])  
40 def predict():  
41     if request.method == 'GET':  
42         img_url = url_for('static',filename = 'images/hello.png')  
43         return render_template('toxic.html', url=img_url)  
44     if request.method == 'POST':  
45         comment = request.form['comment']  
46         new_row = {'comment_text':comment}  
47         user_df = pd.DataFrame(columns = ['comment_text'])  
48         user_df = user_df.append(new_row,ignore_index = True)  
49         user_df['comment_text'] = user_df['comment_text'].map(lambda com : clean_text(com))  
50         user_text = user_df['comment_text']  
51         user_features = loaded.transform(user_text)  
52         cols_target = ['obscene', 'insult', 'toxic', 'severe_toxic', 'identity_hate', 'threat']  
53         list= []  
54         mapper = {}  
55         for label in cols_target:  
56             filename = str(label)+'_model.sav'  
57             print(filename)  
58             model = pickle.load(open(filename, 'rb'))  
59             print('... Processing {}'.format(label))  
60             user_y_prob = model.predict_proba(user_features)[:,1]  
61             print(label,":",user_y_prob[0])  
62             list.append([label,user_y_prob])  
63         print(list)  
64  
65         final=[ ]  
66         flag=0  
67         for i in list:  
68             if i[1]>0.5:  
69                 final.append(i[0])  
70                 flag=2  
71         if not len(final):  
72             text = "Yaaay!! The comment is clean"  
73             img_url = url_for('static',filename = 'images/happy.png')  
Ln 1, Col 1 | Spaces: 4 | UTF-8 | CRLF | Python | 3.9.7 (base:conda) | 17:52 | 09-04-2022 |
```

```
53     lst = []
54     mapper = {}
55     for label in cols_target:
56         filename = str(label) + '_model.sav'
57         print(filename)
58         model = pickle.load(open(filename, 'rb'))
59         print('... Processing {}'.format(label))
60         user_y_prob = model.predict_proba(user_features)[:,1]
61         print(label, ":", user_y_prob[0])
62         lst.append([label, user_y_prob])
63     print(lst)
64
65     final = []
66     flag=0
67     for i in lst:
68         if i[1]>0.5:
69             final.append(i[0])
70             flag+=1
71     if not len(final):
72         text = "Yayy!! The comment is clean"
73         img_url = url_for('static',filename = 'images/happy.png')
74         flag=1
75         print(img_url)
76     else:
77         text="The comment is "
78         for i in final:
79             text+=i+
80             img_url = url_for('static',filename = 'images/toxic.png')
81         print(text)
82     return render_template('toxic.html',ypred = text,url= img_url,flag=flag)
83
84 if __name__ == '__main__':
85     app.run(host = 'localhost', debug = False , threaded = False)
86
87
```

commentApp.py - Visual Studio Code

C: > Users > SADHU SUNDHAR > Desktop > pt 2 > Commit-Toxicity-Multi-Class-Classification-main > Commit-Toxicity-Multi-Class-Classification-main > Flask > commentApp.py > ...

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.7 (base: conda) ⚡ 09-04-2022 1

File Edit Selection View Go Run Terminal Help commentApp.py

commentApp.py - Visual Studio Code

C: > Users > SADHU SUNDHAR > Desktop > pt 2 > Commit-Toxicity-Multi-Class-Classification-main > Commit-Toxicity-Multi-Class-Classification-main > Flask > commentApp.py > ...

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.7 (base: conda) ⚡ 09-04-2022 1

File Edit Selection View Go Run Terminal Help commentApp.py

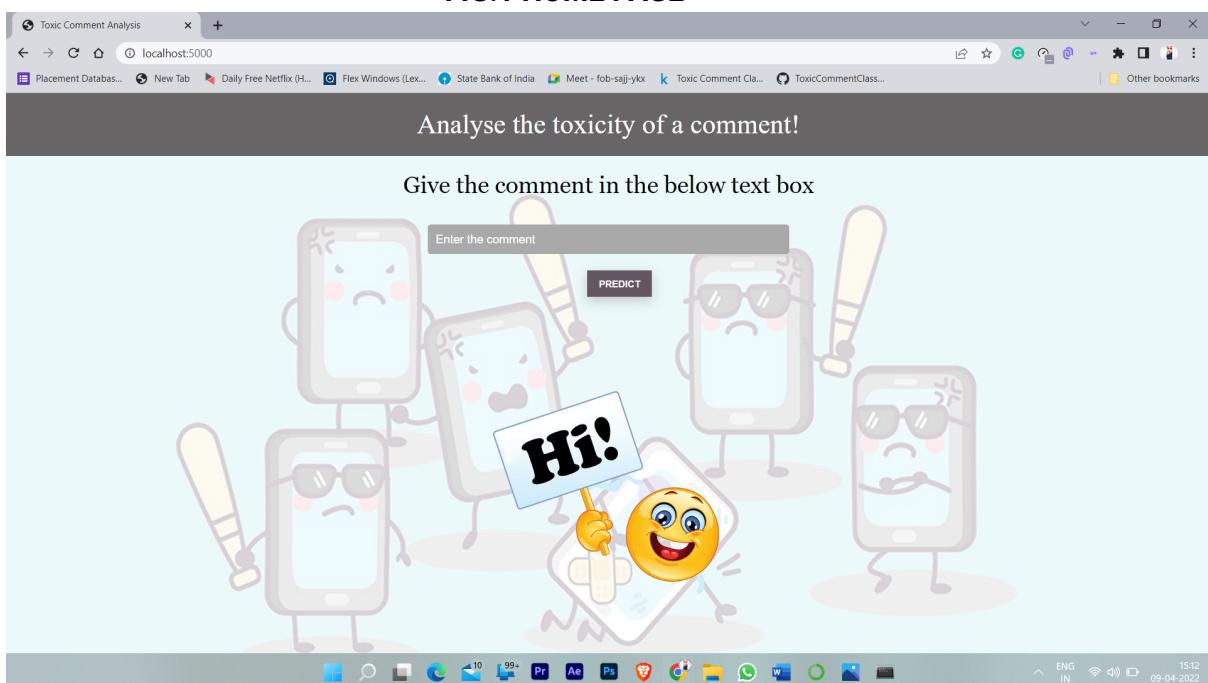
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
fication-main/Flask/commentApp.py"
" Serving Flask app "commentApp" (lazy loading)
" Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://localhost:5000/ (Press CTRL+C to quit)
```

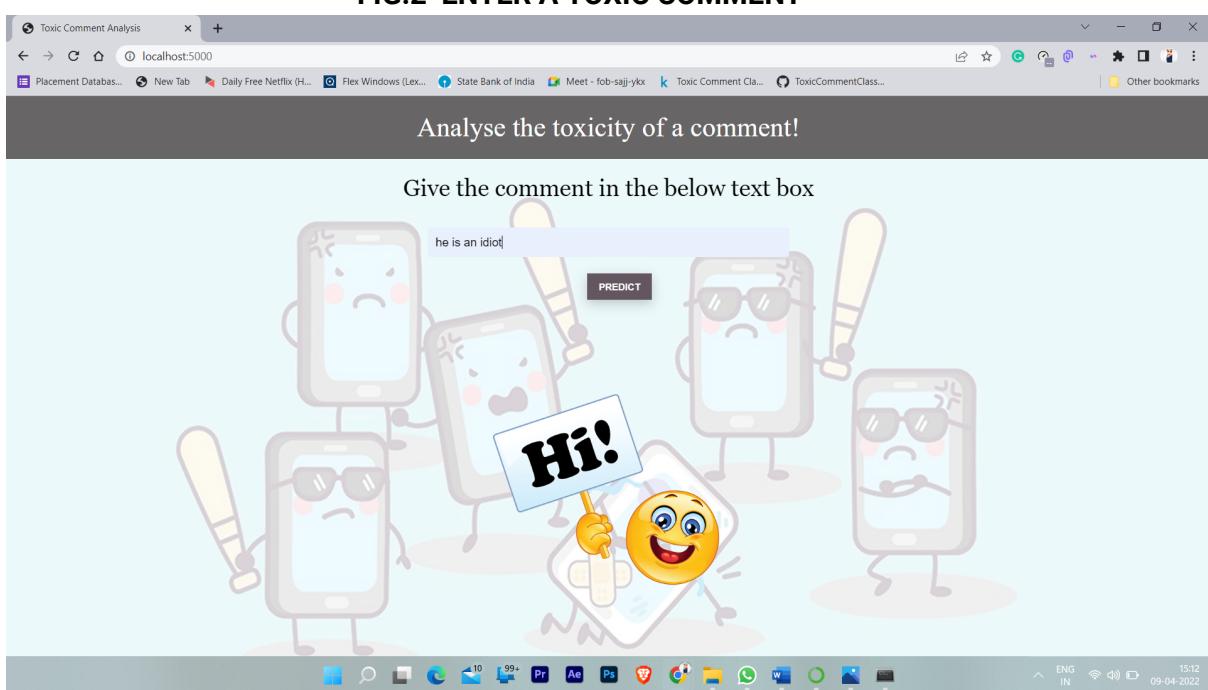
Python Python Python Python Python Python Python

## b. UI Output Screenshot:

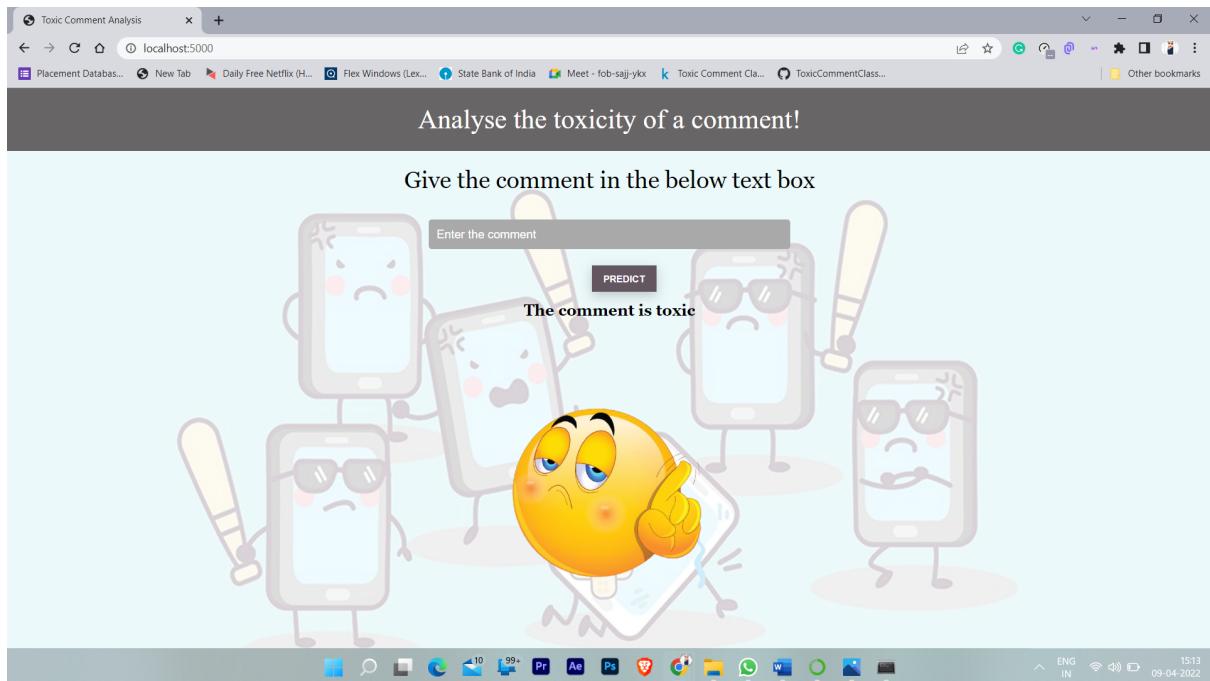
**FIG:1 HOME PAGE**



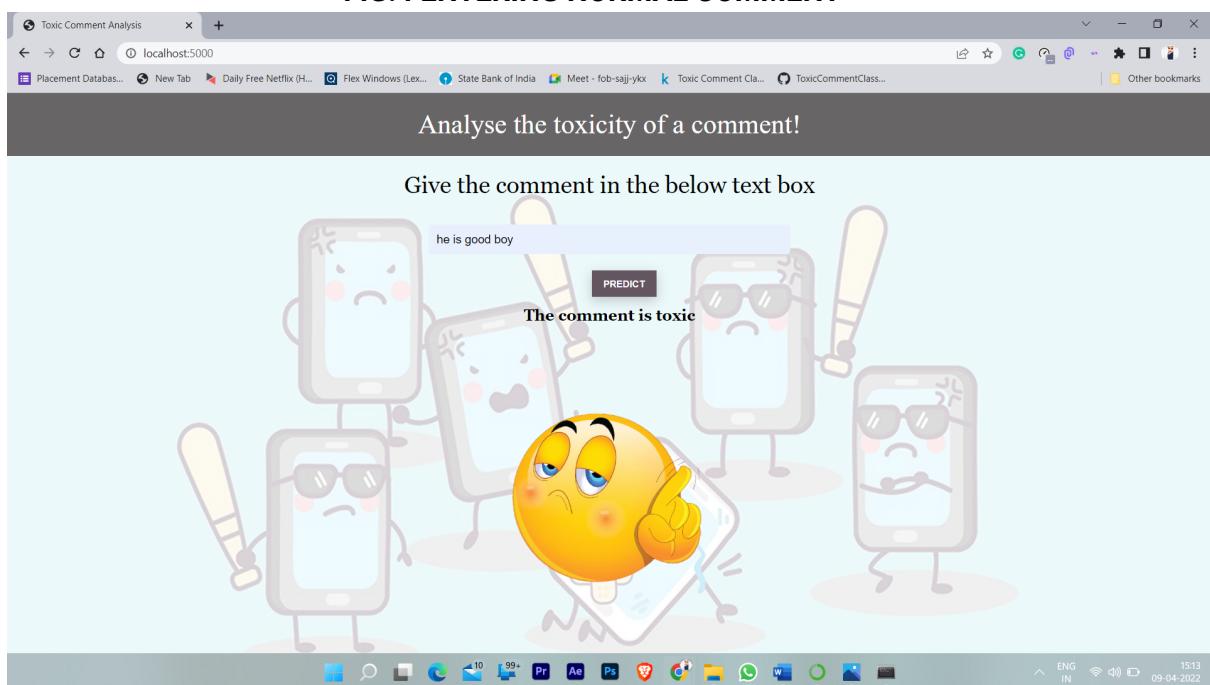
**FIG:2 ENTER A TOXIC COMMENT**



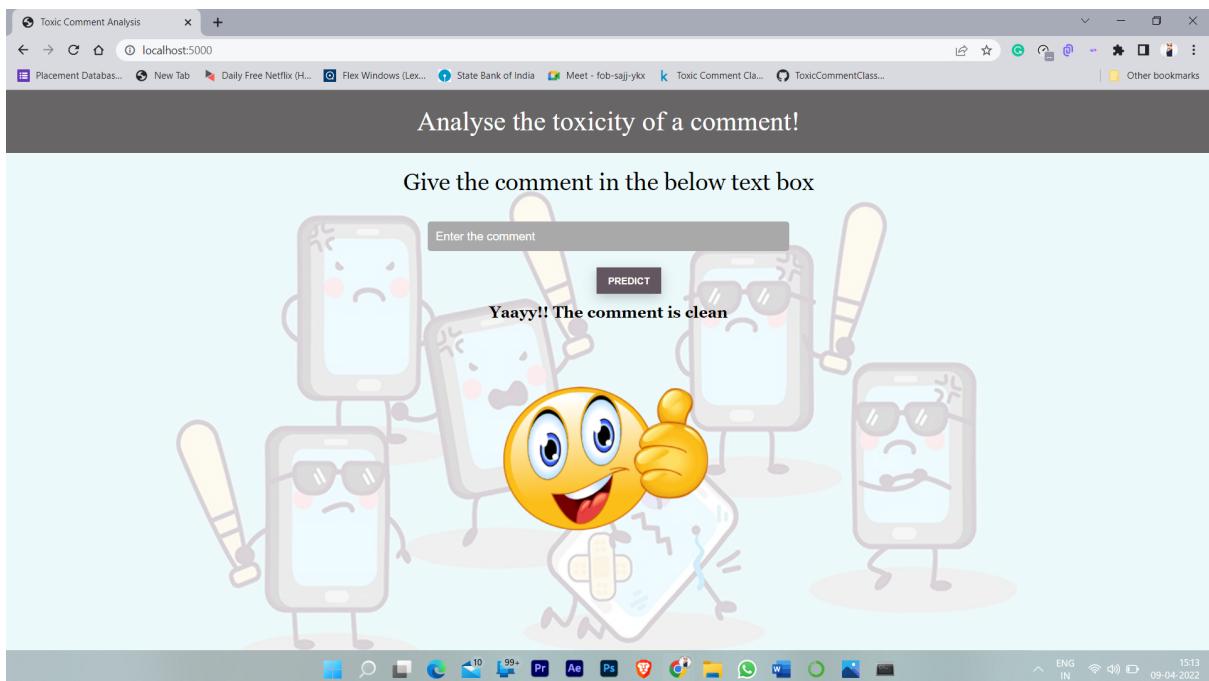
**FIG:3 SHOWING OUTPUT THE COMMENT IS TOXIC**



**FIG:4 ENTERING NORMAL COMMENT**



**FIG:3 SHOWING OUTPUT THE COMMENT IS CLEAN**



# THANK YOU

Submitted By  
**GUNDUMOGULA SADHU SUNDHAR**  
**39110360.**