TOXIC COMMENTS CLASSIFICATION

IN SOCIAL NETWORKING

Submitted By

PENTAPATI ESWAR AKHIL

39110765

INDEX

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 behavior 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 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 able to analyses 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

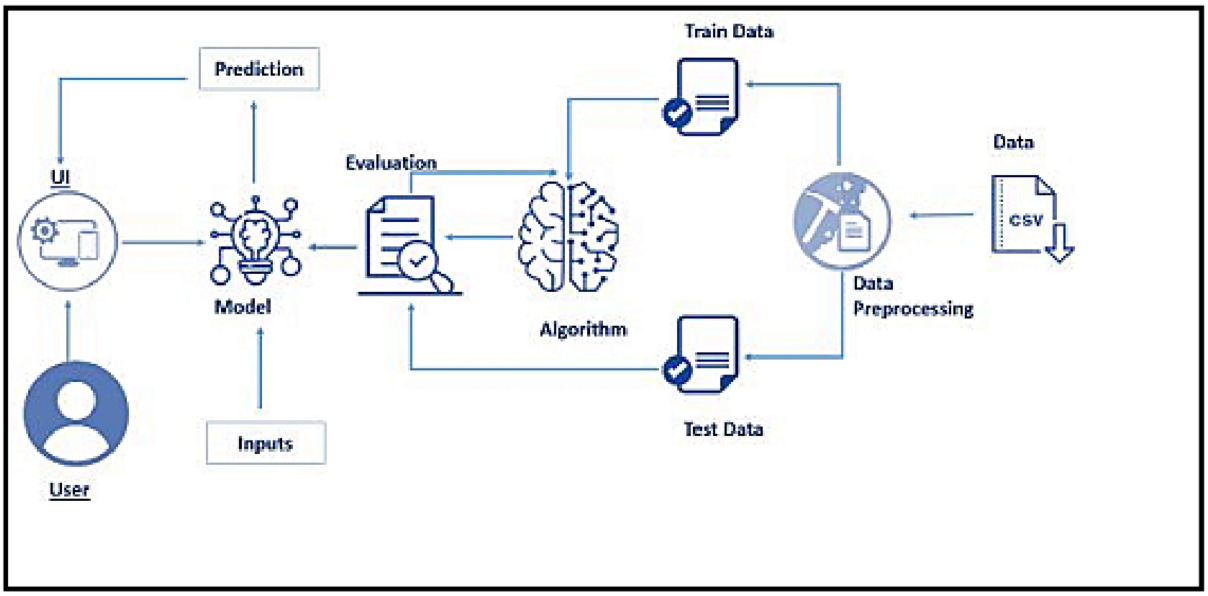
Bag of words statistics and bag of symbols statistics are the typical source information for the toxic comment's detection. Usually, the following statistics-based features are used: length of the comment, number of capital letters, number of exclamation marks, number of question marks, number of spelling 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

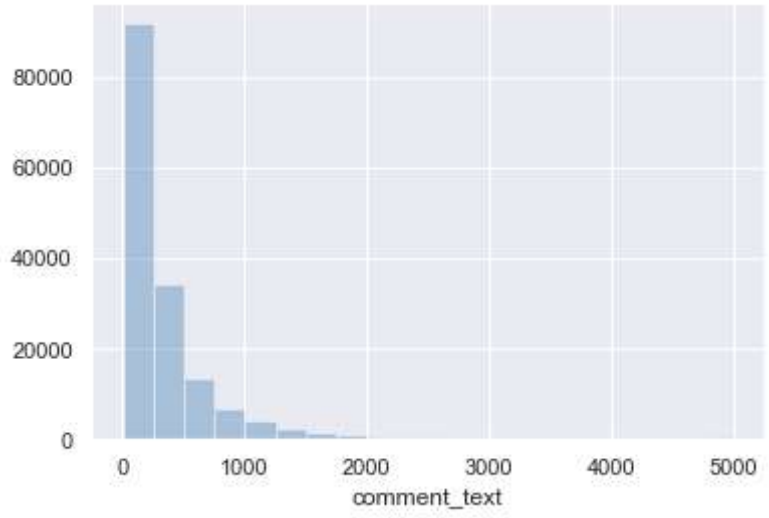


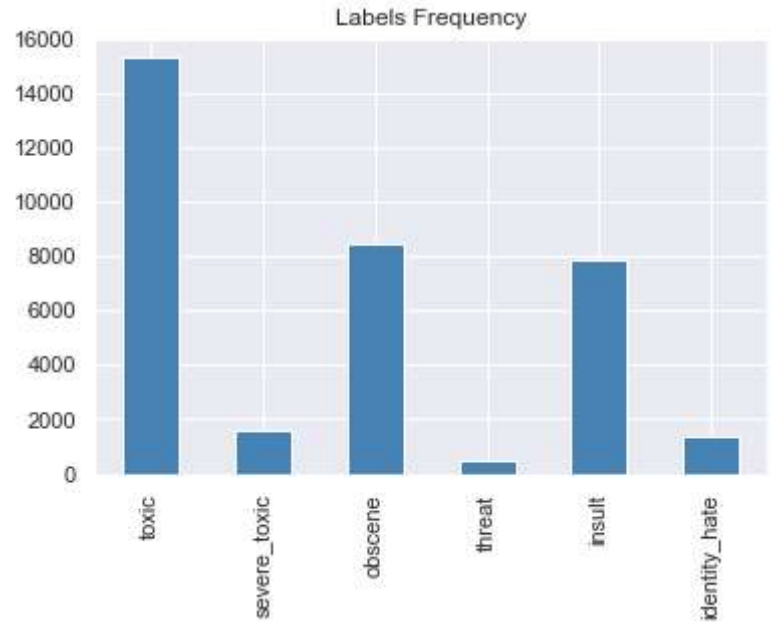
b. Hardware/Software Requirements

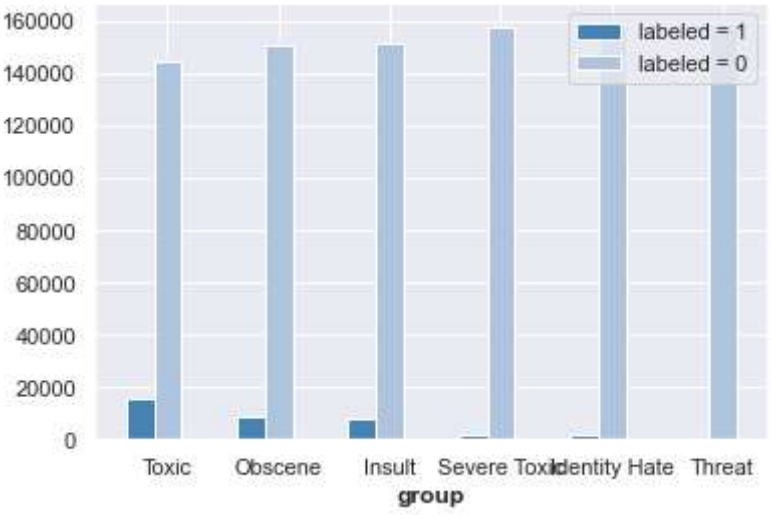
1. IDE: Jupyter Notebook, Spyder, Anaconda navigator
2. Programming Language (Back-end): Python 3.7
3. Front-end: HTML, CSS
4. Framework: Flask

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.



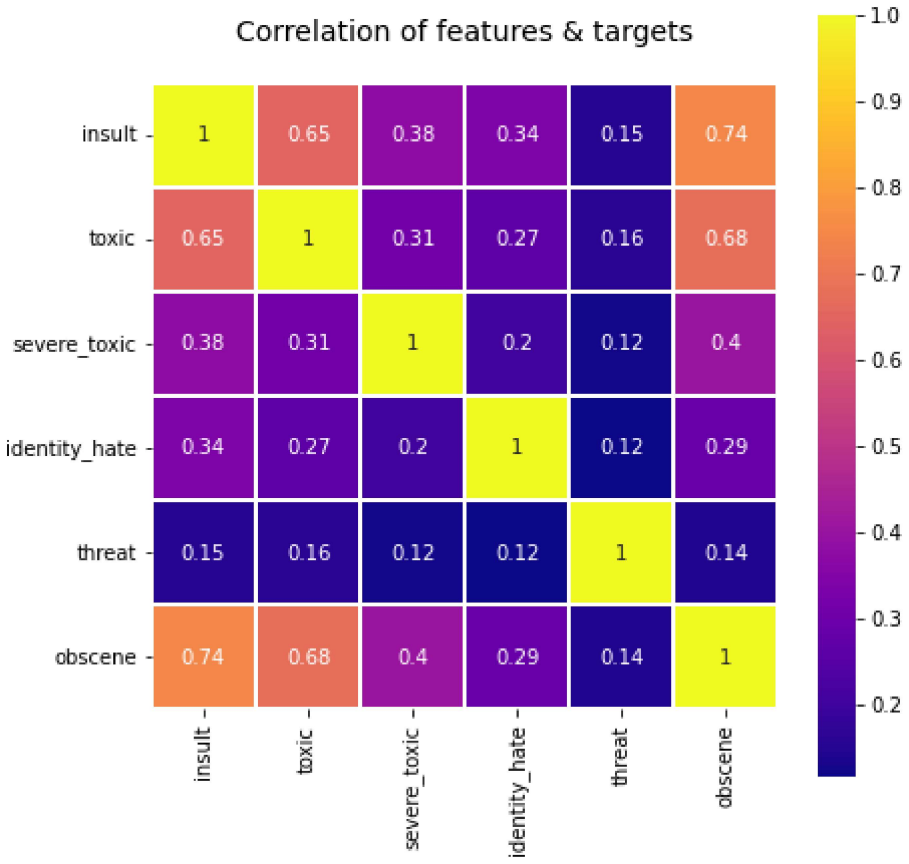




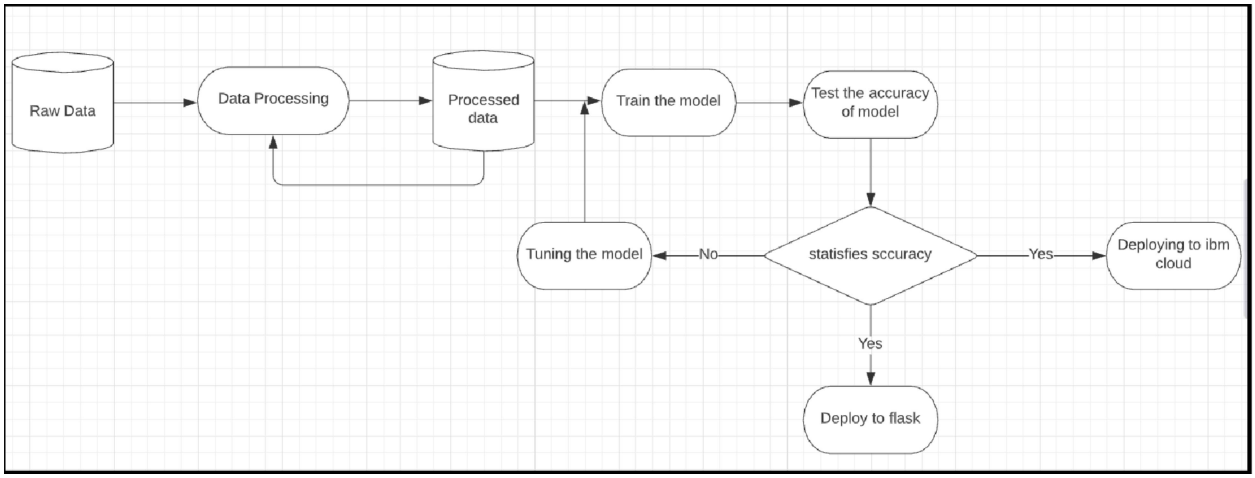
Only 39 percent of the toxic comments have only one label, and the majority have some sort of overlap. I believe that because of this, it will be much more difficult to train a classifier on specific labels than whether they are toxic. This ambiguity and the lack of explanation around it is what led 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.

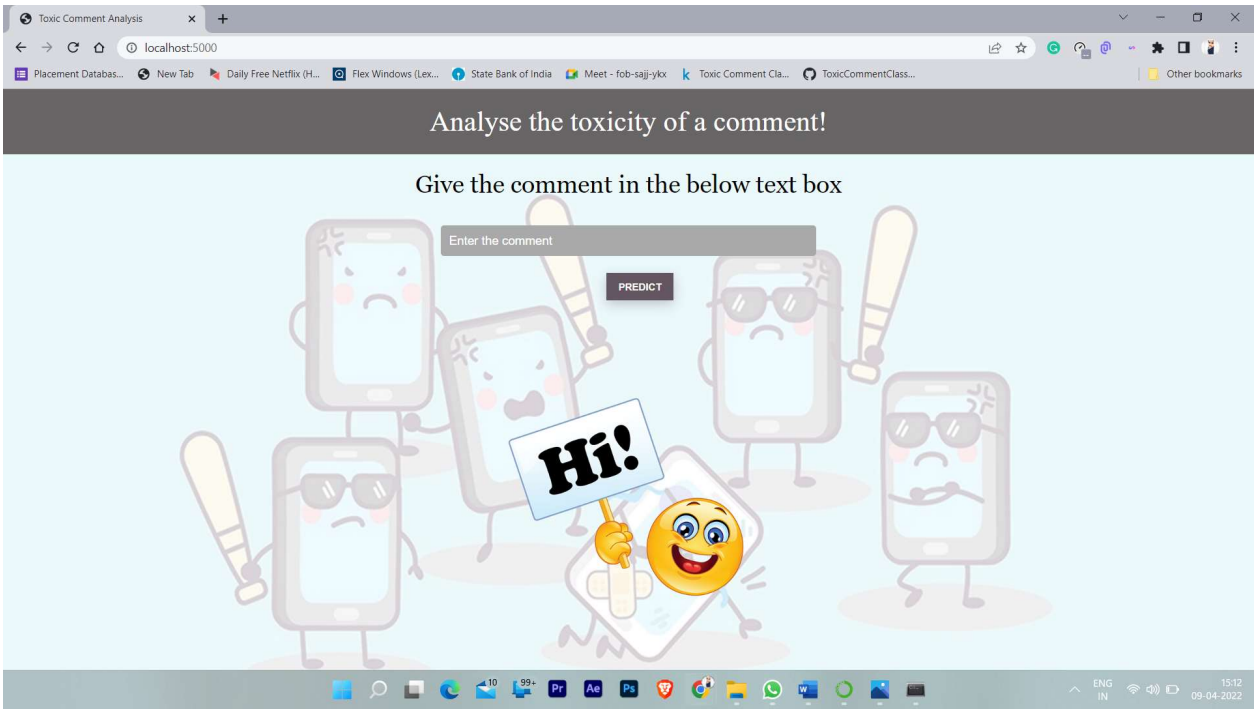


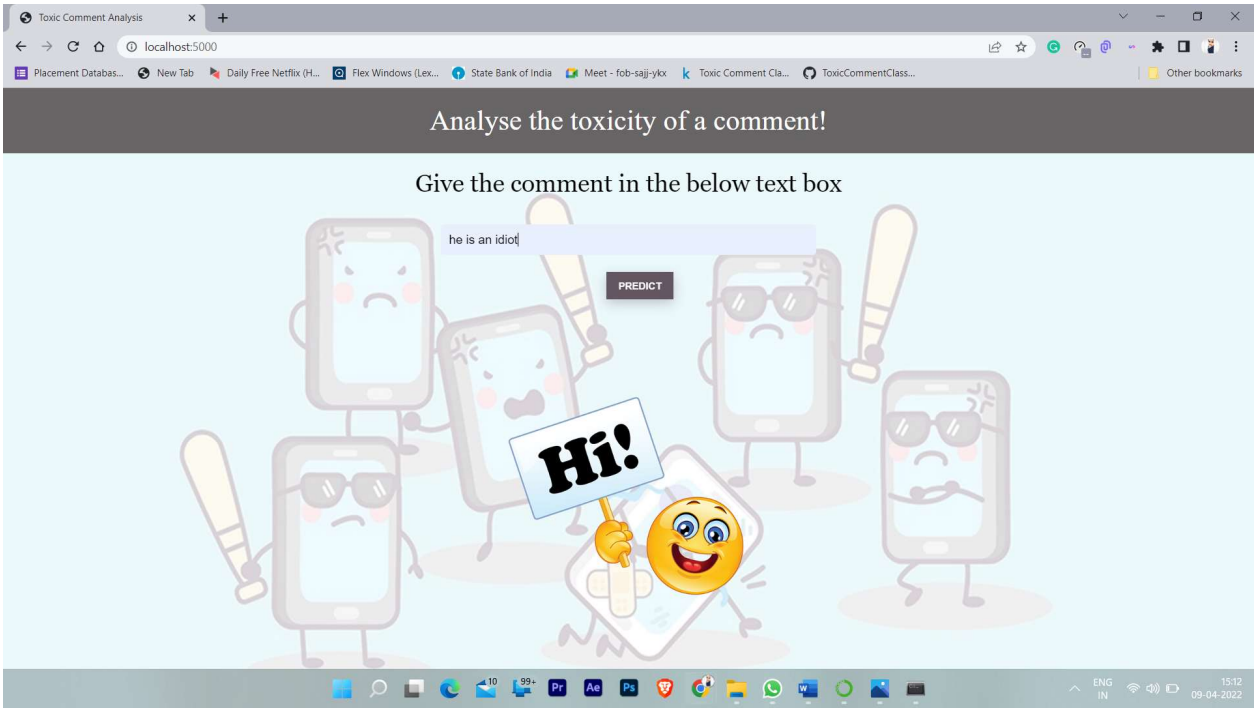
5. FLOW CHART



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 feel comments into the model that would be most effective. A few that I can think of are:

1. Short comments
2. Long comments
3. “Hot” threads where the rate of commenting is high and emotions may be high

11. BIBLIOGRAPHY

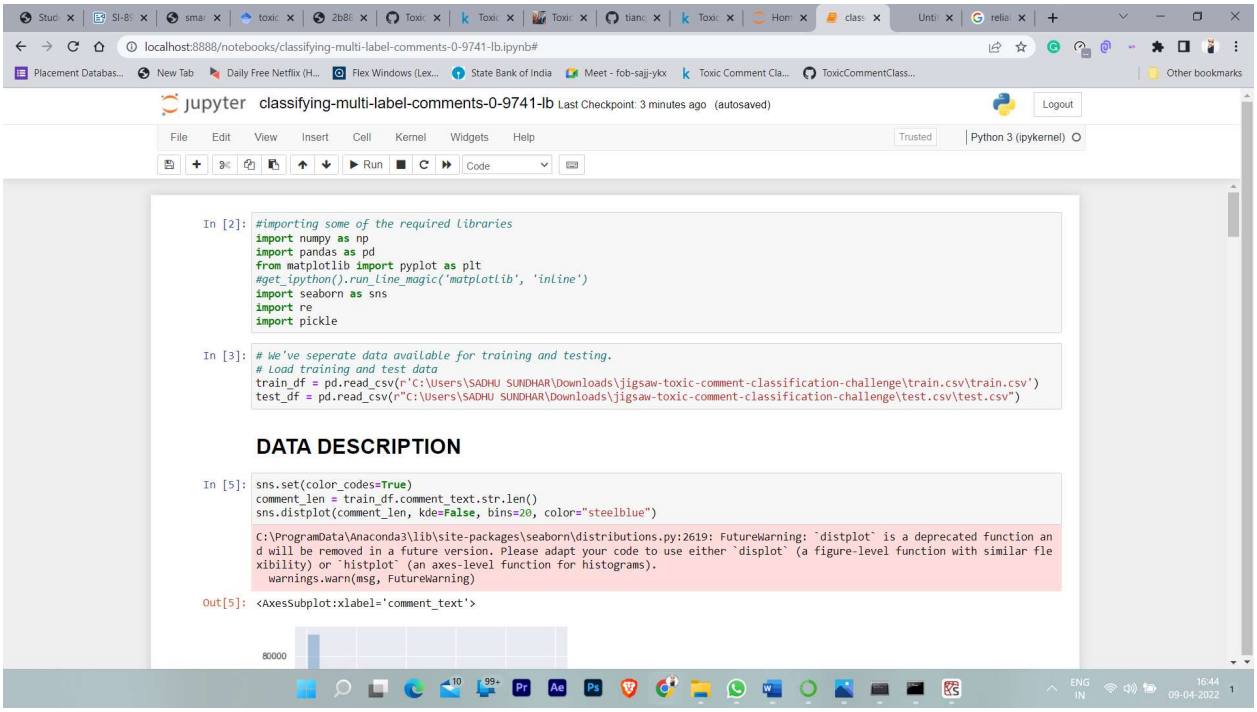
1. DATASET-- **https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data**

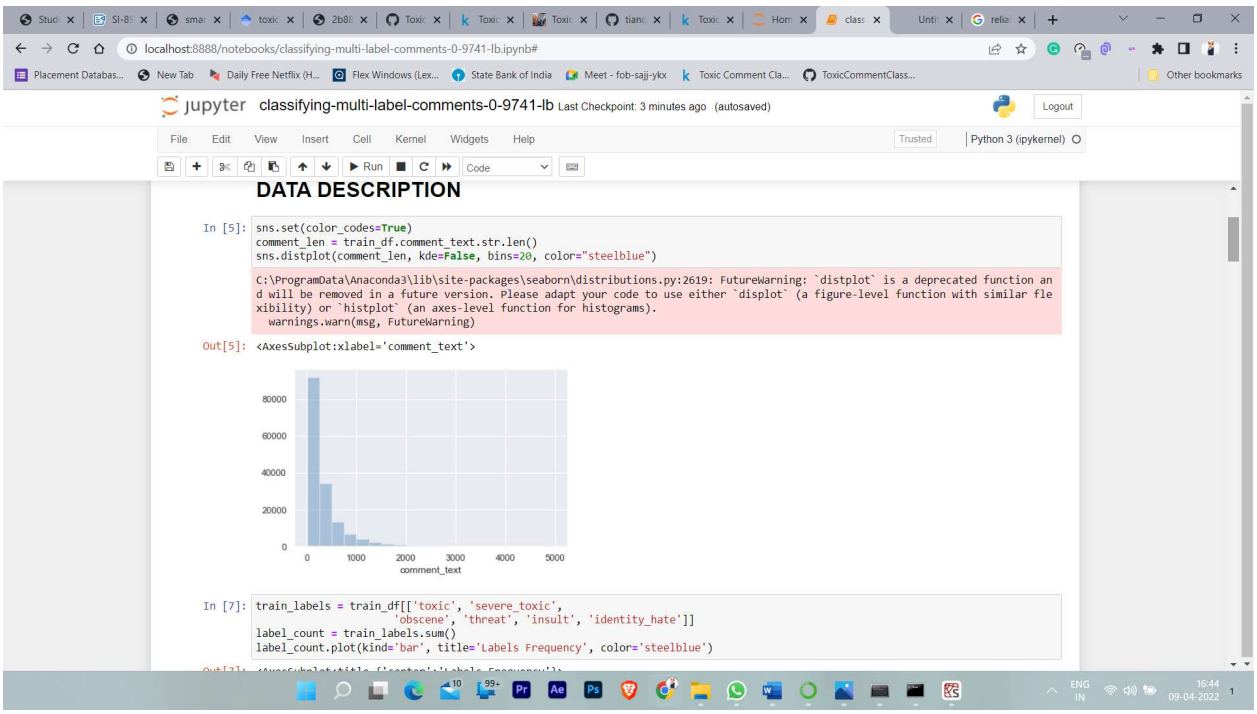
2. Toxic Comment Classification Group Project for MSDS621 Machine Learning at University of San Francisco

12. APPENDIX

a. Source code:

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





localhost:8888/notebooks/classifying-multi-label-comments-0-9741-lb.ipynb#

jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 4 minutes ago (unsaved changes)

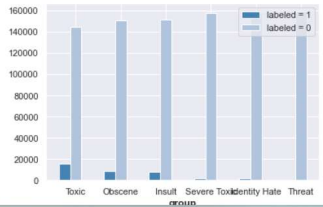
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Markdown

In [10]:
```
barWidth = 0.25

bars1 = [sum(train_df['toxic'] == 1), sum(train_df['obscene'] == 1), sum(train_df['insult'] == 1), sum(train_df['severe_toxic'] =
         sum(train_df['identity_hate'] == 1), sum(train_df['threat'] == 1)]
bars2 = [sum(train_df['toxic'] == 0), sum(train_df['obscene'] == 0), sum(train_df['insult'] == 0), sum(train_df['severe_toxic'] =
         sum(train_df['identity_hate'] == 0), sum(train_df['threat'] == 0)]

r1 = np.arange(len(bars1))
r2 = [x + barWidth for x in r1]

plt.bar(r1, bars1, color='steelblue', width=barWidth, label='labeled = 1')
plt.bar(r2, bars2, color='lightsteelblue', width=barWidth, label='labeled = 0')

plt.xlabel('group', fontweight='bold')
plt.xticks([r + barWidth for r in range(len(bars1))], ['Toxic', 'Obscene', 'Insult', 'Severe Toxic', 'Identity Hate',
                                                       'Threat'])
plt.legend()
plt.show()
```



localhost:8888/notebooks/classifying-multi-label-comments-0-9741-lb.ipynb#

jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 5 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Markdown

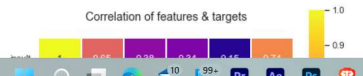
In [11]:
```
# In the training data, the comments are labelled as one or more of the
# six categories; toxic, severe toxic, obscene, threat, insult and identity hate.
# This is essentially a multi-label classification problem.
cols_target = ['insult','toxic','severe_toxic','identity_hate','threat','obscene']
```

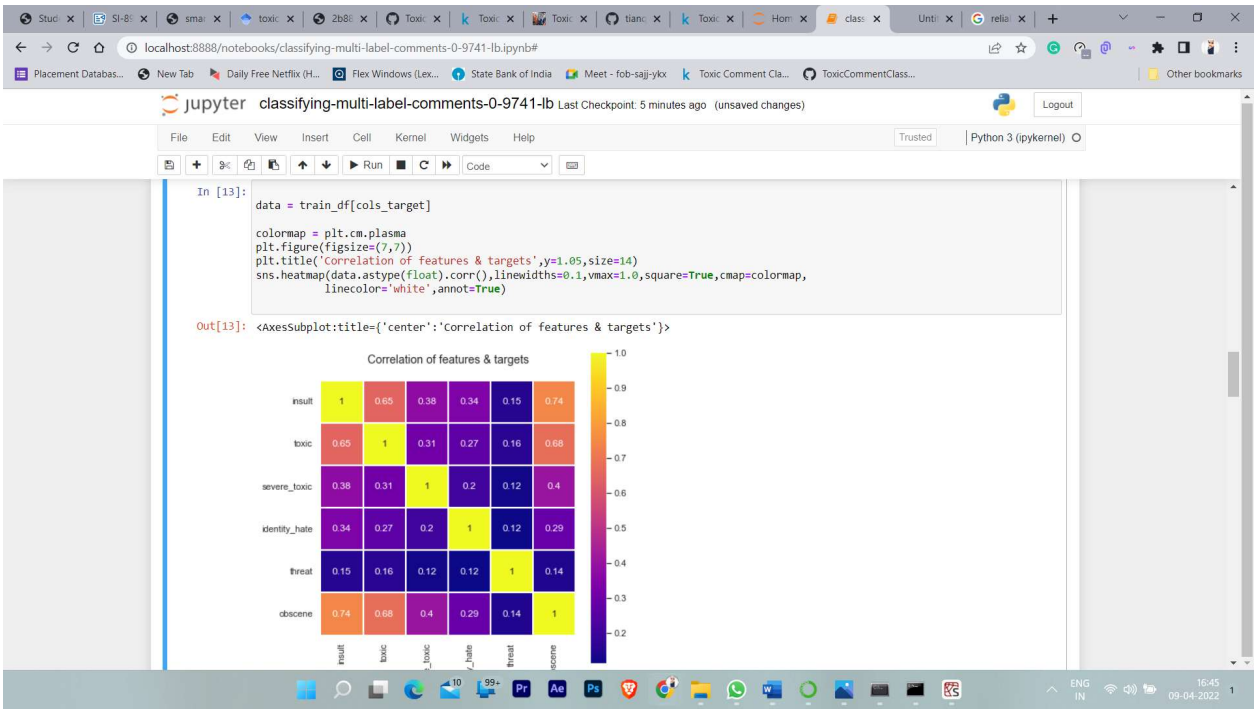
In [12]:
```
# check for null comments in test_df
print(test_df.isnull().any())
# no null values in test_df
```

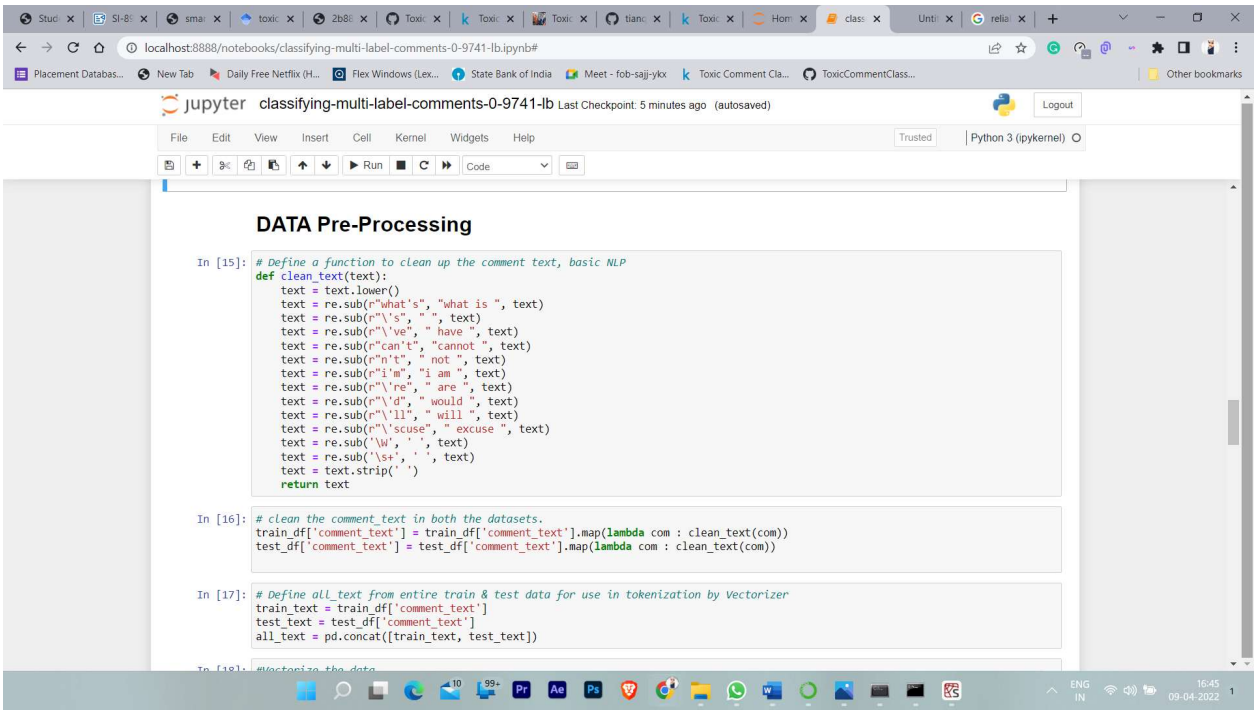
```
id              False
comment_text    False
dtype: bool
```

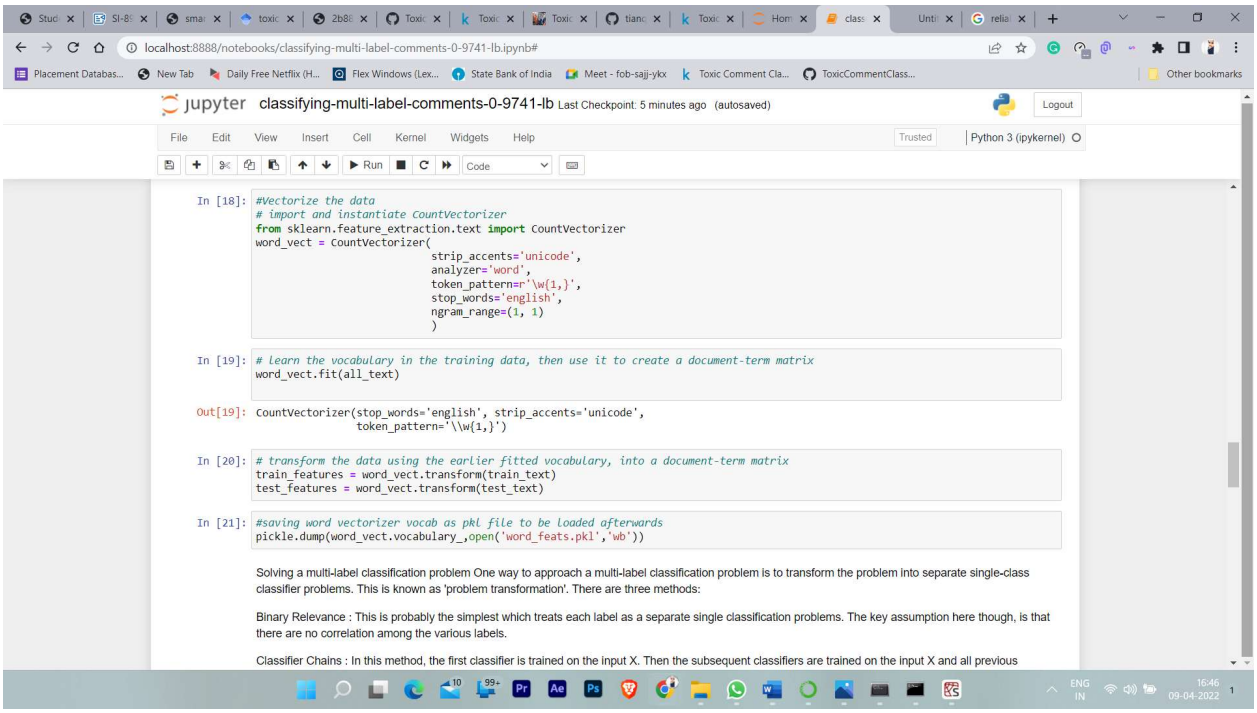
In [13]:
```
# All rows in the training and test data contain comments, so there's no need to clean
# up null fields.

# Next, let's examine the correlations among the target variables.
data = train_df[cols_target]

colormap = plt.cm.plasma
plt.figure(figsize=(7,7))
plt.title('Correlation of features & targets',y=1.05,size=14)
sns.heatmap(data.astype(float).corr(),linewidths=0.1,vmax=1.0,square=True,cmap=colormap,
            linecolor='white',annot=True)
# Indeed, it looks like some of the labels are higher correlated, e.g.
# insult-obscene has the highest at 0.74, followed by toxic-obscene and toxic-insult.
# This co-relation can be taken in use for other kind of algorithms, such as multi-chain labelling
```

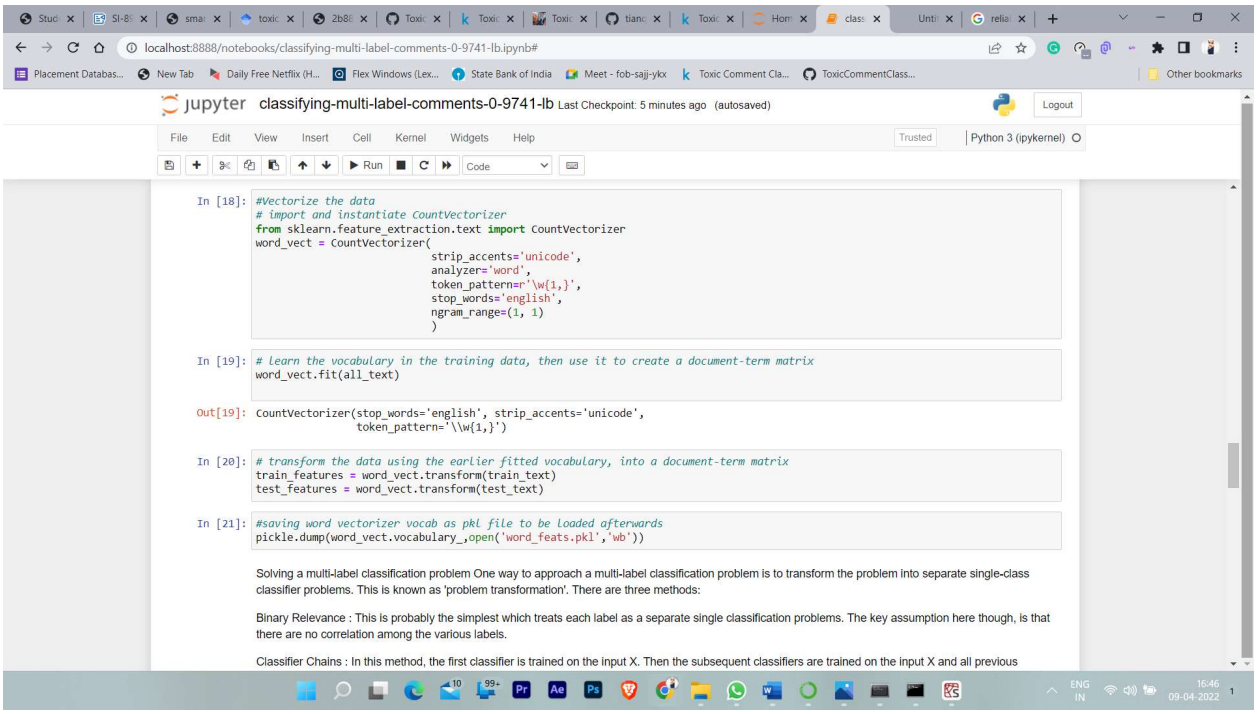
Out[13]: <AxesSubplot:title={'center':'Correlation of features & targets'}>

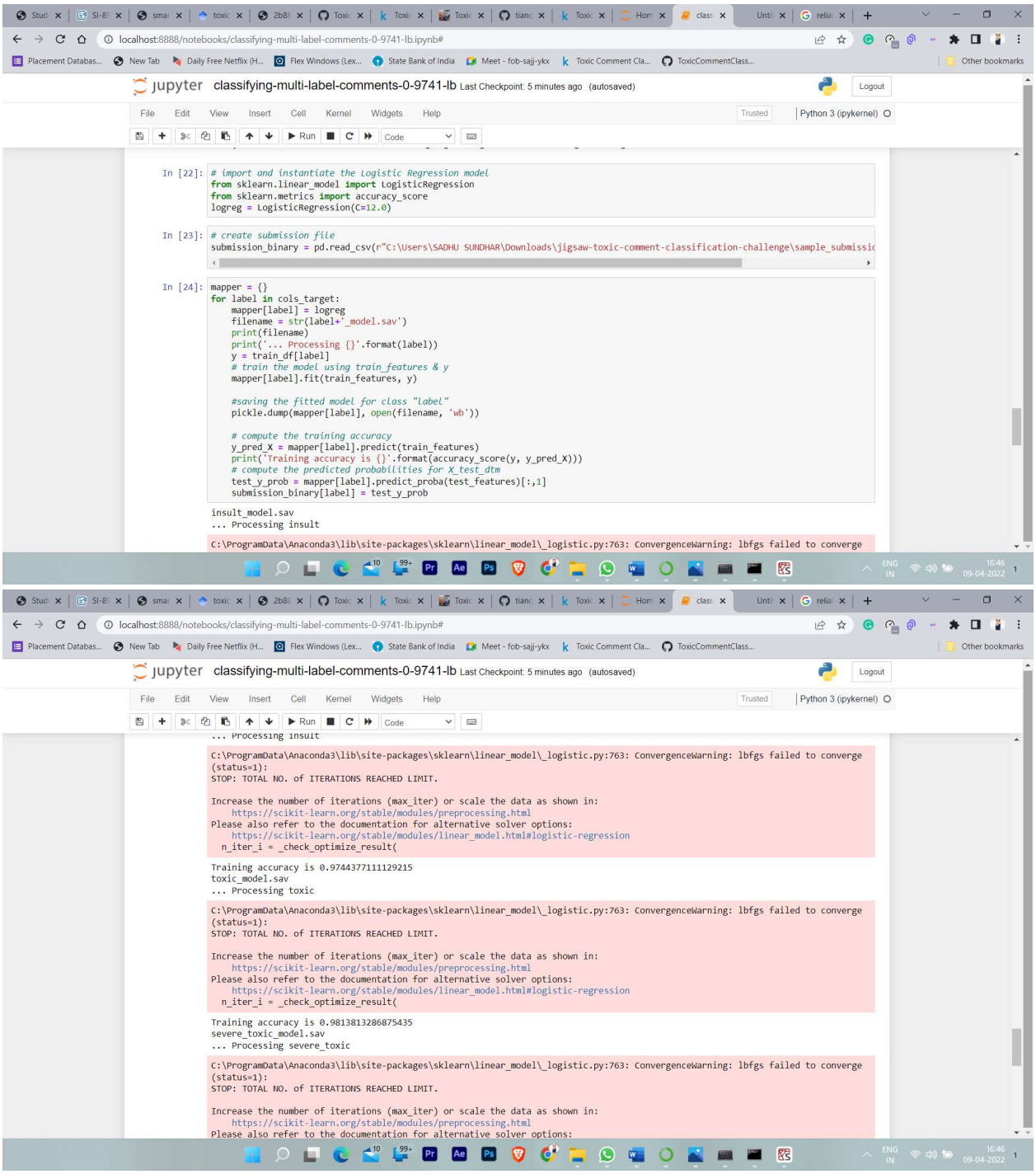



localhost:8888/notebooks/classifying-multi-label-comments-0-9741-lb.ipynb#
jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 5 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [13]:
data = train_df[cols_target]
colormap = plt.cm.plasma
plt.figure(figsize=(7,7))
plt.title('Correlation of features & targets',y=1.05,size=14)
sns.heatmap(data.astype(float).corr(),linewidths=0.1,vmax=1.0,square=True,cmap=colormap,
linecolor='white',annot=True)
Out[13]: <AxesSubplot:title={'center':'Correlation of features & targets'}>
Correlation of features & targets
insult 1 0.65 0.38 0.34 0.15 0.74
toxic 0.65 1 0.31 0.27 0.16 0.68
severe_toxic 0.38 0.31 1 0.2 0.12 0.4
identity_hate 0.34 0.27 0.2 1 0.12 0.29
threat 0.15 0.16 0.12 0.12 1 0.14
obscene 0.74 0.68 0.4 0.29 0.14 1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2


localhost:8888/notebooks/classifying-multi-label-comments-0-9741-lb.ipynb#
jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 5 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
DATA Pre-Processing
In [15]:
# Define a function to clean up the comment text, basic NLP
def clean_text(text):
text = text.lower()
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", "cannot ", 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)
text = re.sub('\W', ' ', text)
text = re.sub('\s+', ' ', text)
text = text.strip(' ')
return text
In [16]:
clean the comment_text in both the datasets.
train_df['comment_text'] = train_df['comment_text'].map(lambda com : clean_text(com))
test_df['comment_text'] = test_df['comment_text'].map(lambda com : clean_text(com))
In [17]:
# Define all_text from entire train & test data for use in tokenization by Vectorizer
train_text = train_df['comment_text']
test_text = test_df['comment_text']
all_text = pd.concat([train_text, test_text])


localhost:8888/notebooks/classifying-multi-label-comments-0-9741-lb.ipynb#
jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 5 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [18]:
#Vectorize the data
# import and instantiate CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
word_vect = CountVectorizer(
strip_accents='unicode',
analyzer='word',
token_pattern=r'\w{1,}',
stop_words='english',
ngram_range=(1, 1)
)
In [19]:
# learn the vocabulary in the training data, then use it to create a document-term matrix
word_vect.fit(all_text)
Out[19]:
CountVectorizer(stop_words='english', strip_accents='unicode',
token_pattern='\\w{1,}')
In [20]:
# transform the data using the earlier fitted vocabulary, into a document-term matrix
train_features = word_vect.transform(train_text)
test_features = word_vect.transform(test_text)
In [21]:
#saving word vectorizer vocab as pkl file to be loaded afterwards
pickle.dump(word_vect.vocabulary_,open('word_feats.pkl','wb'))
Solving a multi-label classification problem One way to approach a multi-label classification problem is to transform the problem into separate single-class classifier problems. This is known as 'problem transformation'. There are three methods:
Binary Relevance : This is probably the simplest which treats each label as a separate single classification problems. The key assumption here though, is that there are no correlation among the various labels.
Classifier Chains : In this method, the first classifier is trained on the input X. Then the subsequent classifiers are trained on the input X and all previous


localhost:8888/notebooks/classifying-multi-label-comments-0-9741-lb.ipynb#
jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 5 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [18]:
#Vectorize the data
# import and instantiate CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
word_vect = CountVectorizer(
strip_accents='unicode',
analyzer='word',
token_pattern=r'\w{1,}',
stop_words='english',
ngram_range=(1, 1)
)
In [19]:
# learn the vocabulary in the training data, then use it to create a document-term matrix
word_vect.fit(all_text)
Out[19]:
CountVectorizer(stop_words='english', strip_accents='unicode',
token_pattern='\\w{1,}')
In [20]:
# transform the data using the earlier fitted vocabulary, into a document-term matrix
train_features = word_vect.transform(train_text)
test_features = word_vect.transform(test_text)
In [21]:
#saving word vectorizer vocab as pkl file to be loaded afterwards
pickle.dump(word_vect.vocabulary_,open('word_feats.pkl','wb'))
Solving a multi-label classification problem One way to approach a multi-label classification problem is to transform the problem into separate single-class classifier problems. This is known as 'problem transformation'. There are three methods:
Binary Relevance : This is probably the simplest which treats each label as a separate single classification problems. The key assumption here though, is that there are no correlation among the various labels.
Classifier Chains : In this method, the first classifier is trained on the input X. Then the subsequent classifiers are trained on the input X and all previous



localhost:8888/notebooks/classifying-multi-label-comments-0-9741-lb.ipynb#

jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 5 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

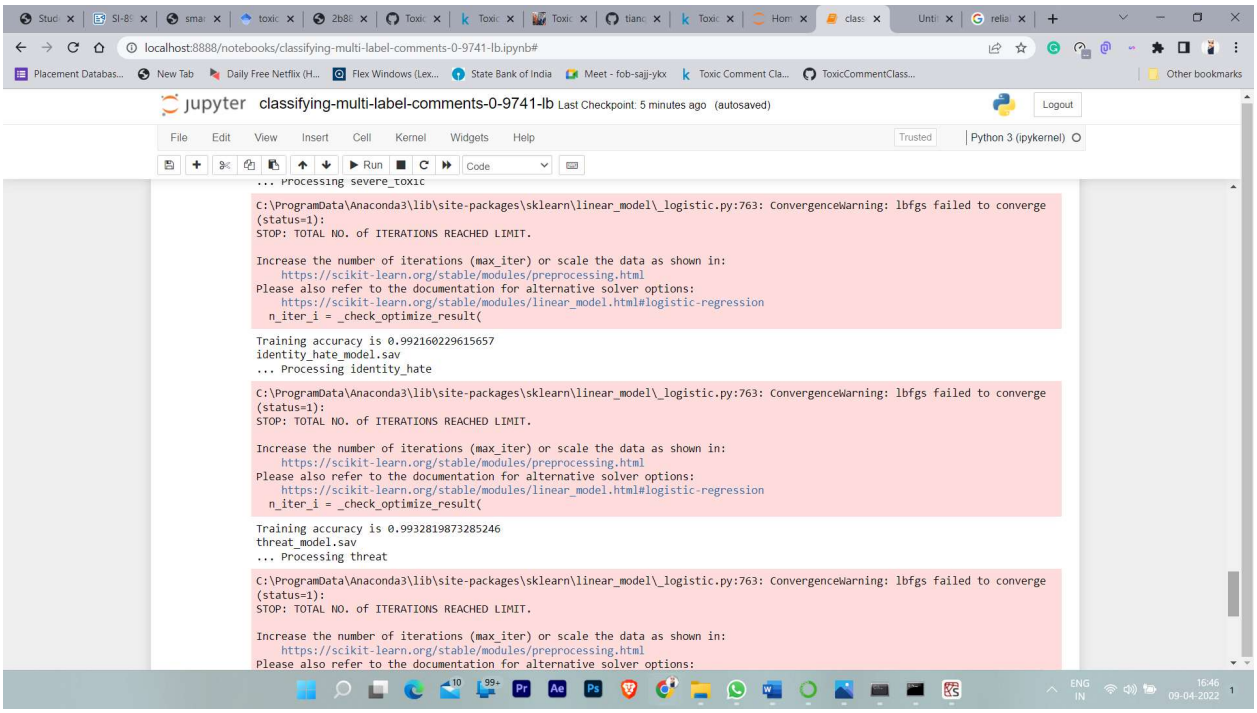
Run Code

In [22]:
```
# import and instantiate the Logistic Regression model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
logreg = LogisticRegression(C=12.0)
```

In [23]:
```
# create submission file
submission_binary = pd.read_csv(r"C:\Users\SADHU SUNDHAR\Downloads\jigsaw-toxic-comment-classification-challenge\sample_submissio
```

In [24]:
```
mapper = {}
for label in cols_target:
    mapper[label] = logreg
    filename = str(label+'_model.sav')
    print(filename)
    print('... Processing {}'.format(label))
    y = train_df[label]
    # train the model using train_features & y
    mapper[label].fit(train_features, y)

    #saving the fitted model for class "label"
    pickle.dump(mapper[label], open(filename, 'wb'))

    # compute the training accuracy
    y_pred_X = mapper[label].predict(train_features)
    print('Training accuracy is {}'.format(accuracy_score(y, y_pred_X)))
    # compute the predicted probabilities for X_test_dtm
    test_y_prob = mapper[label].predict_proba(test_features)[:,1]
    submission_binary[label] = test_y_prob
```

```
insult_model.sav
... Processing insult
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Training accuracy is 0.9744377111129215
toxic_model.sav
... Processing toxic
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Training accuracy is 0.9813813286875435
severe_toxic_model.sav
... Processing severe_toxic
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```



jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 5 minutes ago (autosaved)

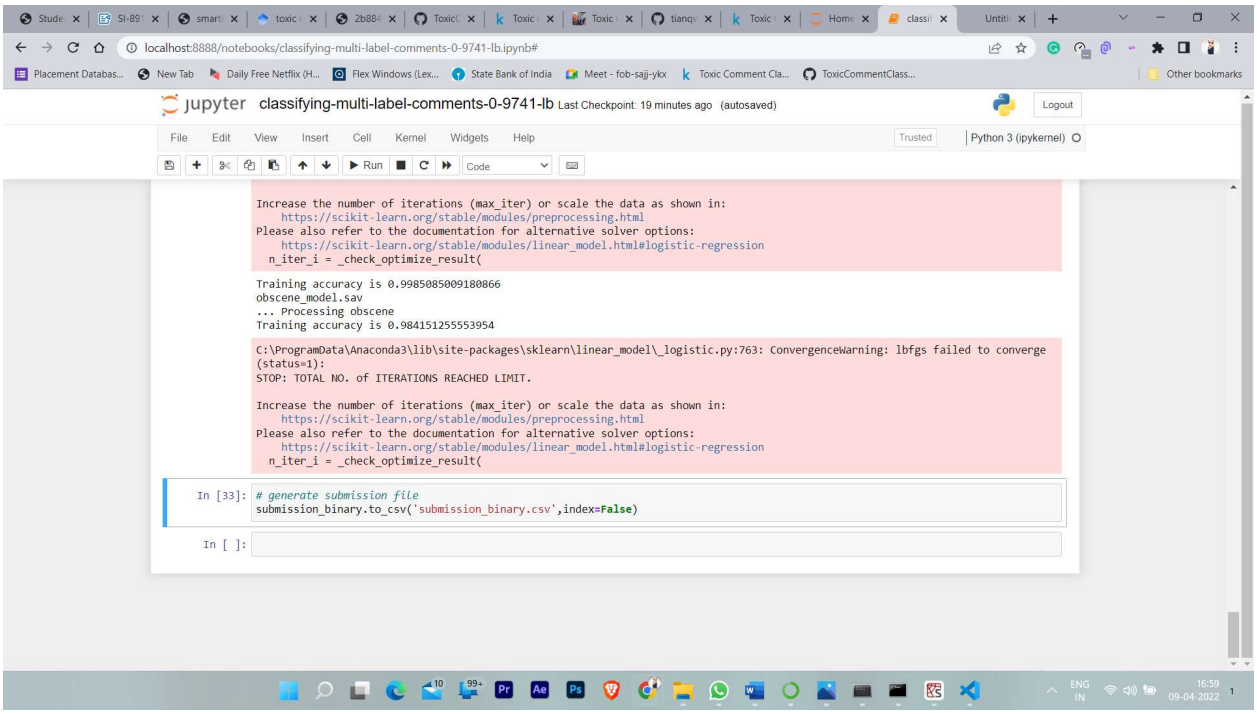
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

```
... Processing severe_toxic
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Training accuracy is 0.992160229615657
identity_hate_model.sav
... Processing identity_hate
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Training accuracy is 0.9932819873285246
threat_model.sav
... Processing threat
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```



jupyter classifying-multi-label-comments-0-9741-lb Last Checkpoint: 19 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

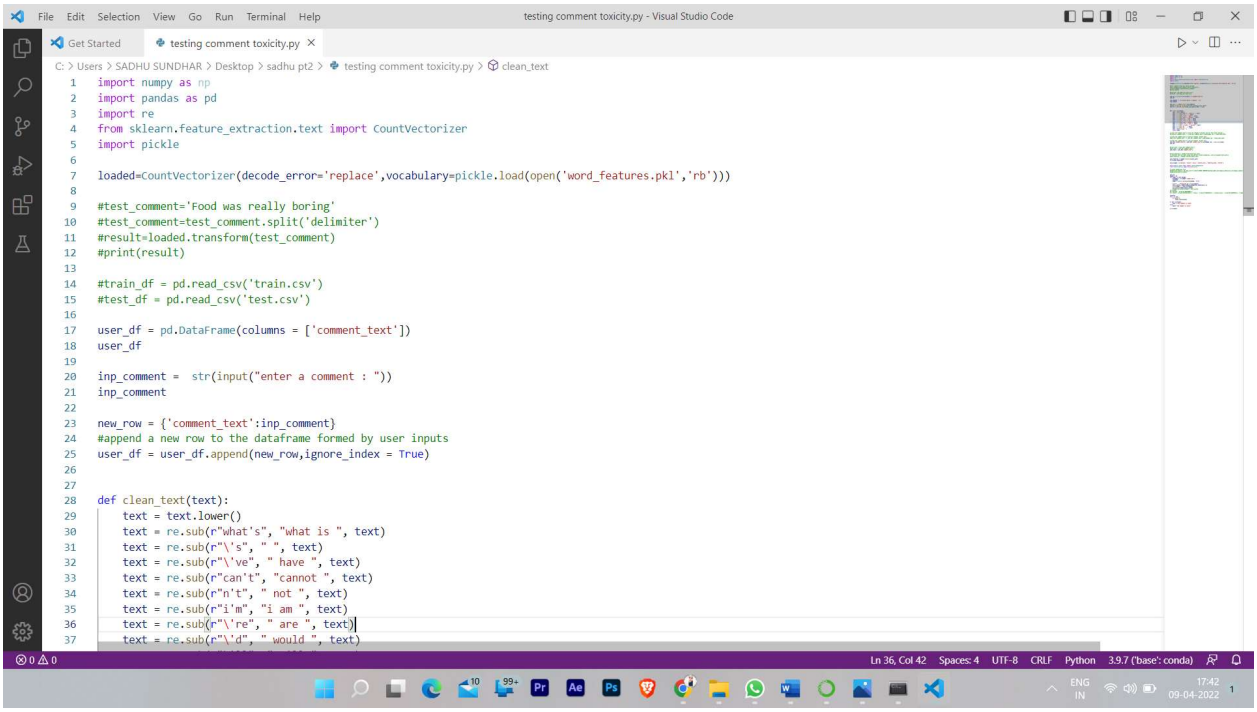
Run Code

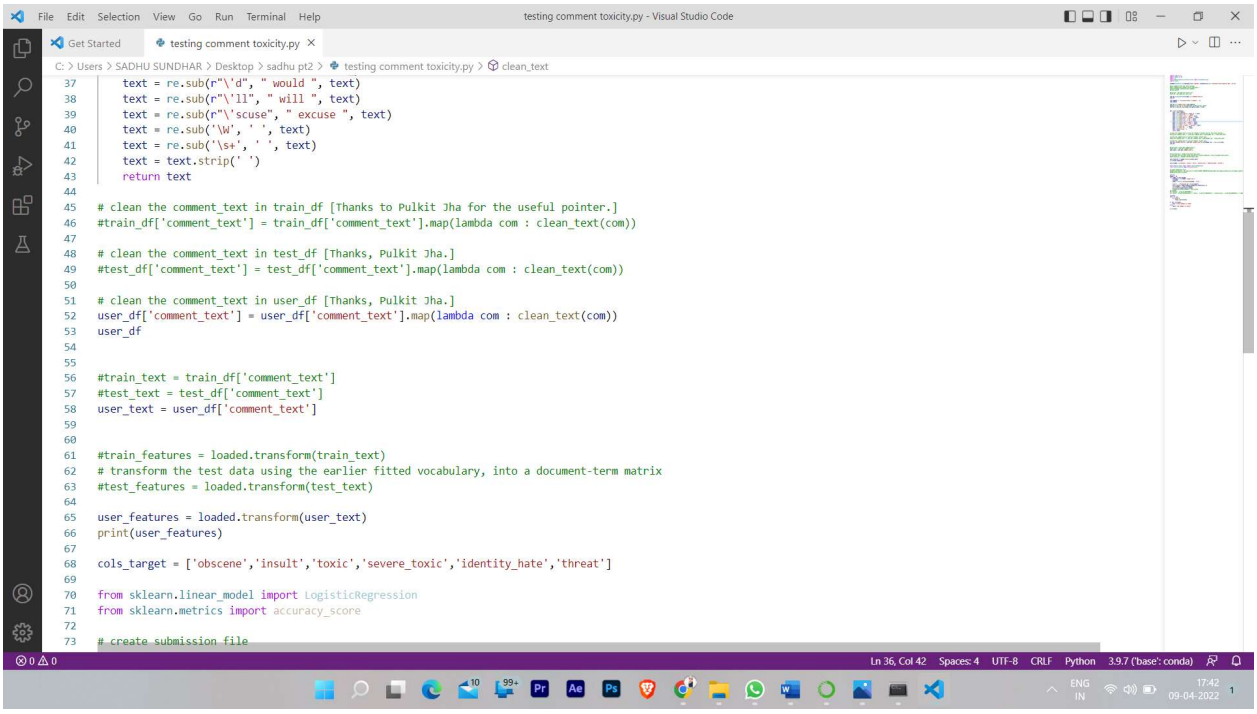
```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

Training accuracy is 0.9985085009180866
obscene_model.sav
... Processing obscene
Training accuracy is 0.984151255553954
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

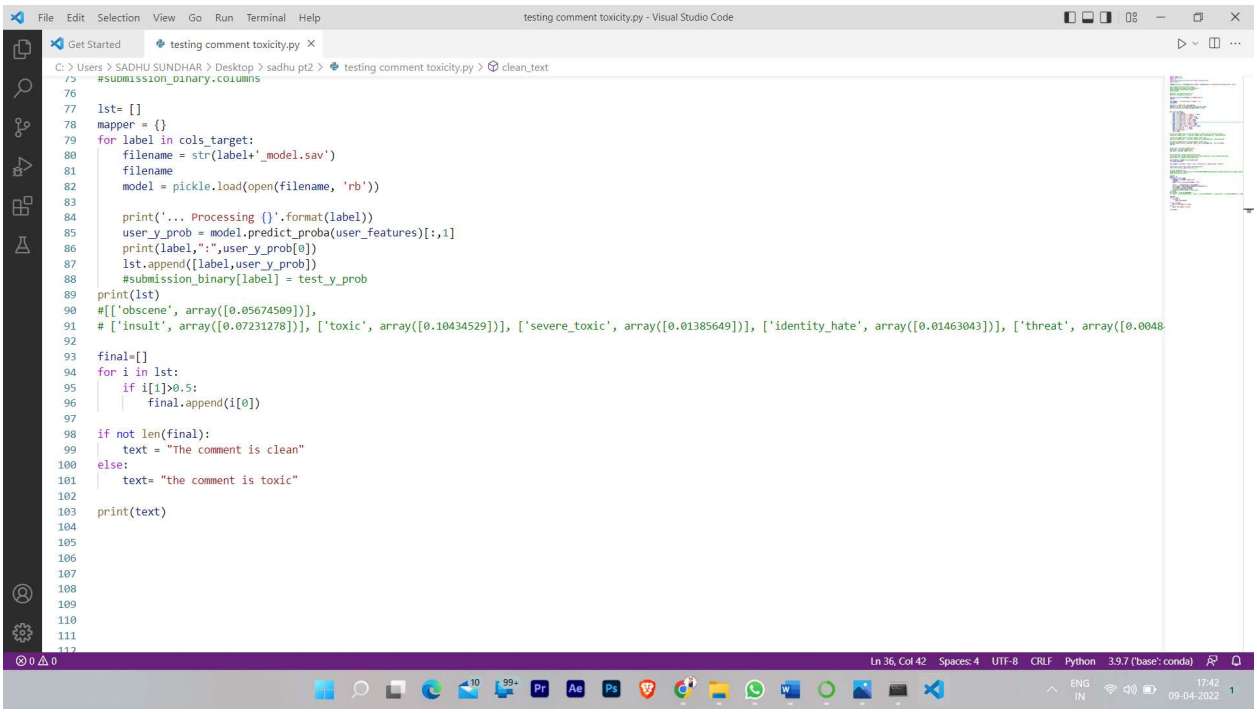
In [33]:
```
# generate submission file
submission_binary.to_csv('submission_binary.csv',index=False)
```

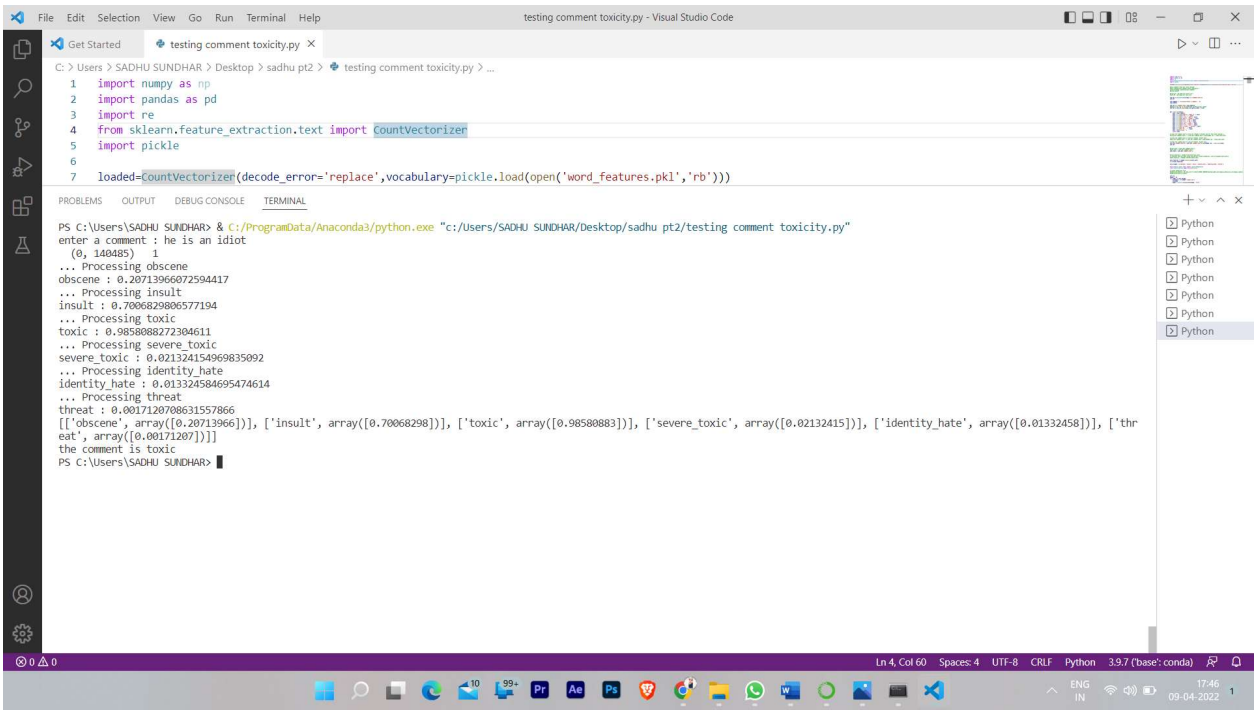
In [ ]:

testing comment toxicity.py

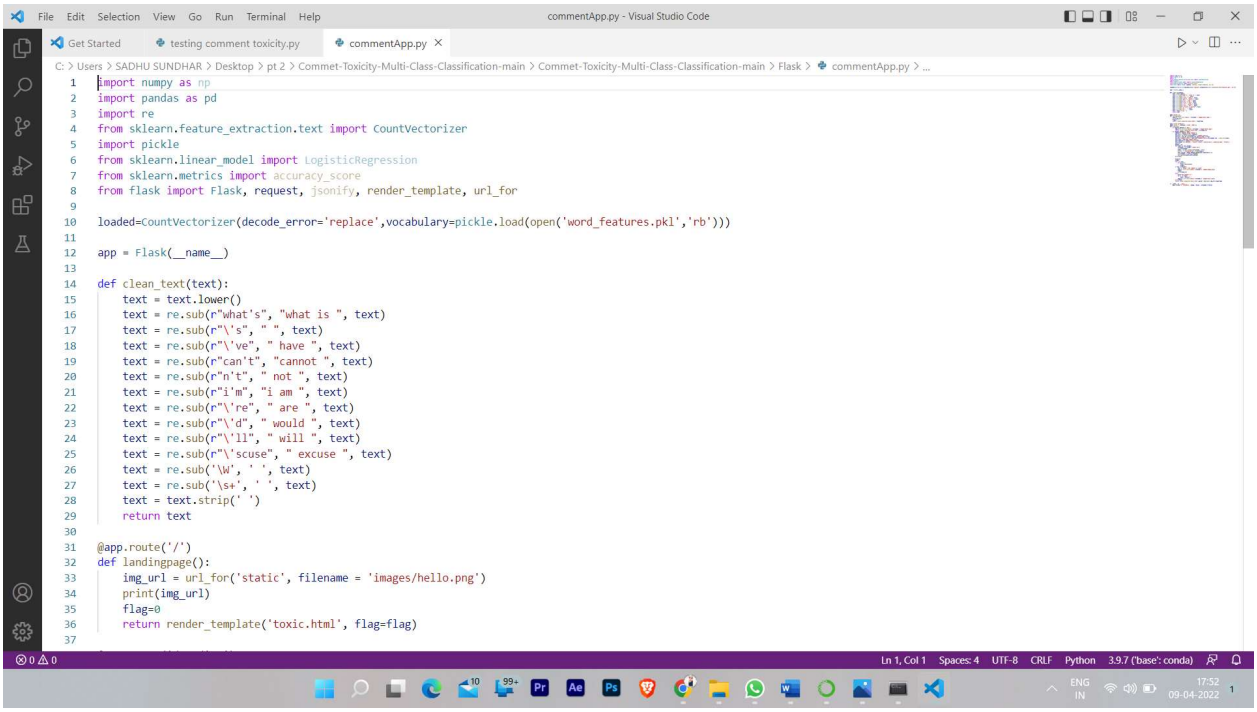




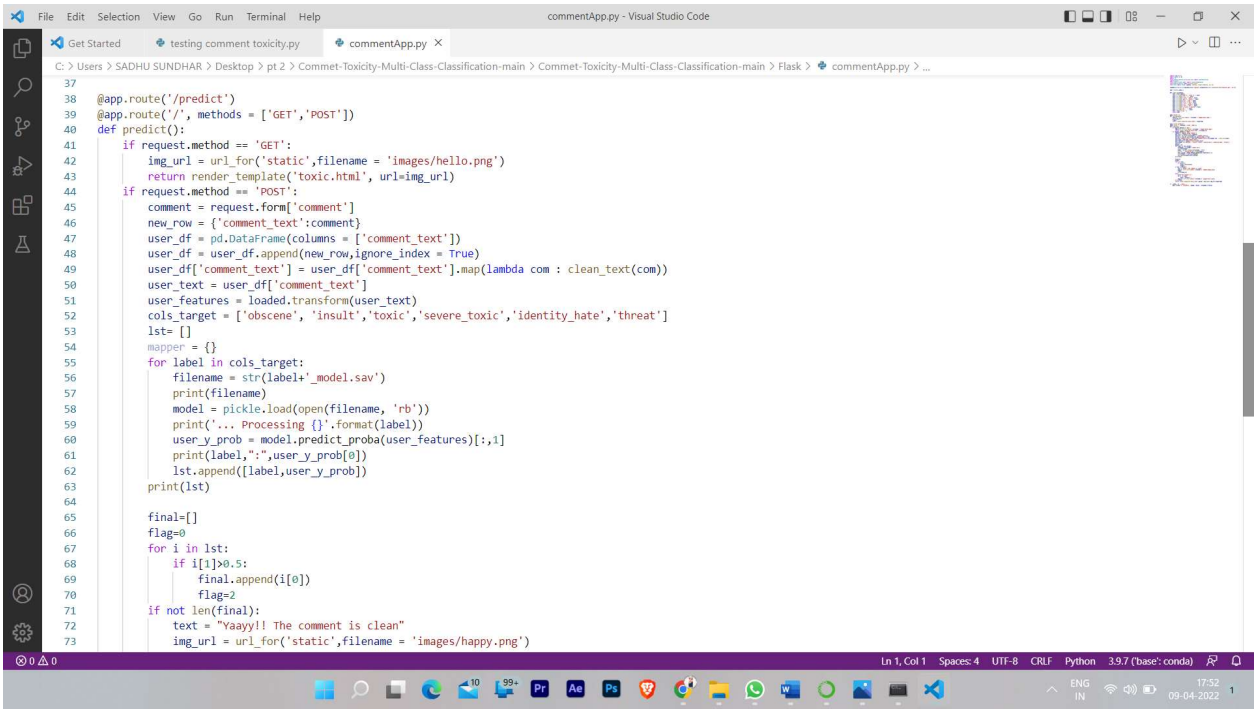

File Edit Selection View Go Run Terminal Help
testing comment toxicity.py - Visual Studio Code
Get Started
testing comment toxicity.py
C: > Users > SADHU SUNDHAR > Desktop > sadhu pt2 > testing comment toxicity.py > clean_text
#submission_binary.columns
lst= []
mapper = {}
for label in cols_target:
filename = str(label+'_model.sav')
filename
model = pickle.load(open(filename, 'rb'))
print('... Processing {}'.format(label))
user_y_prob = model.predict_proba(user_features)[:,1]
print(label,":",user_y_prob[0])
lst.append([label,user_y_prob])
#submission_binary[label] = test_y_prob
print(lst)
#[['obscene', array([0.05674509])],
# ['insult', array([0.07231278])], ['toxic', array([0.10434529])], ['severe_toxic', array([0.01385649])], ['identity_hate', array([0.01463043])], ['threat', array([0.0048
final=[]
for i in lst:
if i[1]>0.5:
final.append(i[0])
if not len(final):
text = "The comment is clean"
else:
text= "the comment is toxic"
print(text)
Ln 36, Col 42 Spaces: 4 UTF-8 CRLF Python 3.9.7 ('base': conda)
ENG IN
17:42
09-04-2022


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 > ...
import numpy as np
import pandas as pd
import re
from sklearn.feature_extraction.text import CountVectorizer
import pickle
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, 140485) 1
... Processing obscene
obscene : 0.20713966072594417
... Processing insult
insult : 0.7006829806577194
... Processing toxic
toxic : 0.9858088272304611
... Processing severe_toxic
severe_toxic : 0.021324154969835092
... Processing identity_hate
identity_hate : 0.013324584695474614
... Processing threat
threat : 0.0017120708631557866
[['obscene', array([0.20713966])], ['insult', array([0.70068298])], ['toxic', array([0.98580883])], ['severe_toxic', array([0.02132415])], ['identity_hate', array([0.01332458])], ['thr
eat', array([0.00171207])]]
the comment is toxic
PS C:\Users\SADHU SUNDHAR>
Python
Ln 4, Col 60 Spaces: 4 UTF-8 CRLF Python 3.9.7 ('base': conda)
ENG IN
17:46
09-04-2022

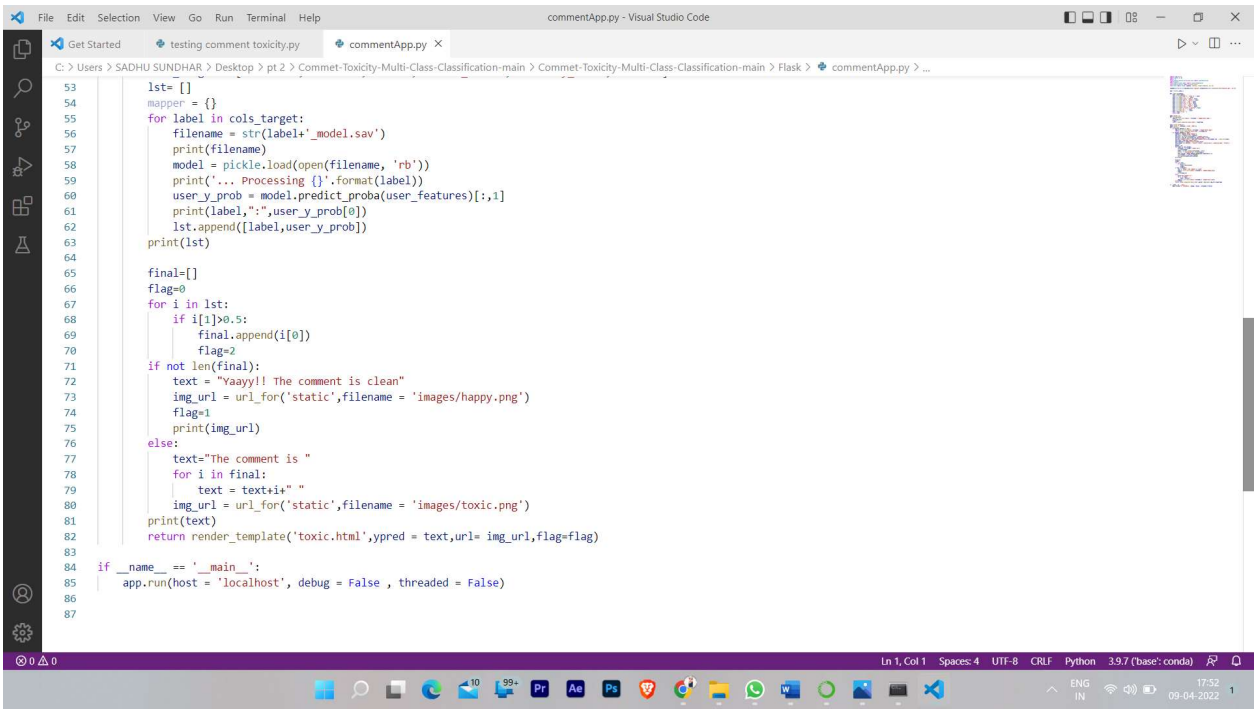
commentApp.py

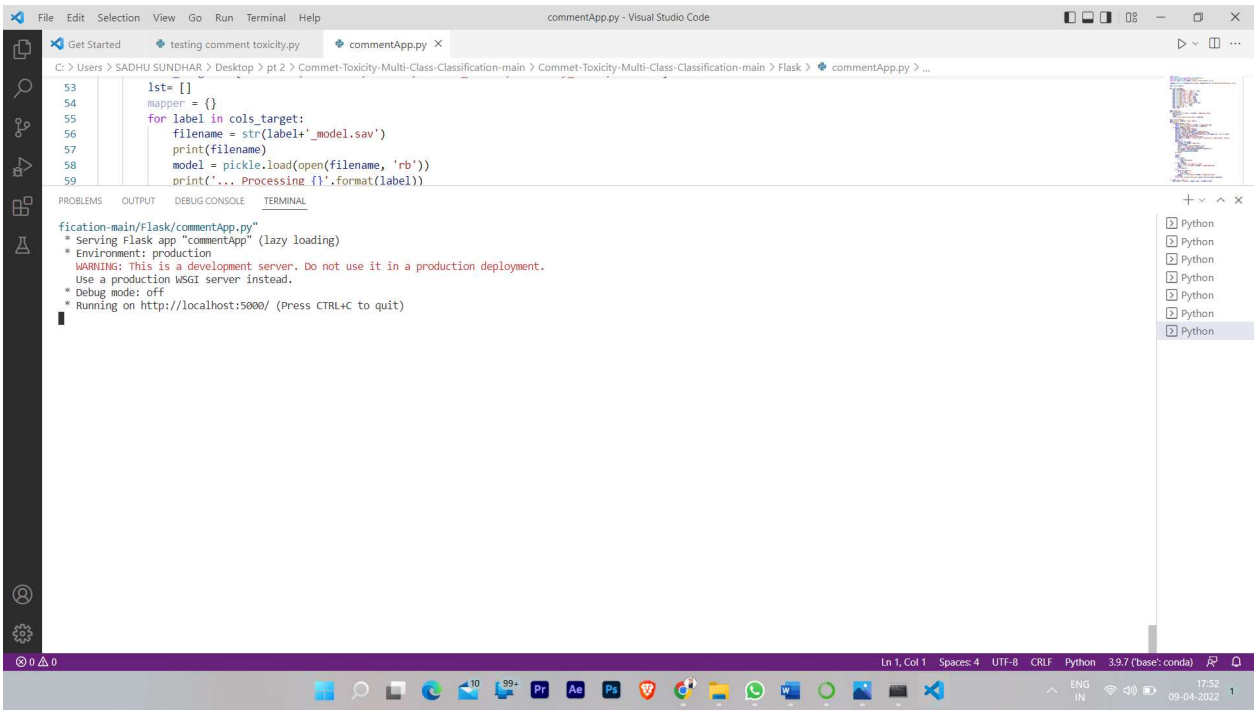


```
import numpy as np
import pandas as pd
import re
from sklearn.feature_extraction.text import CountVectorizer
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from flask import Flask, request, jsonify, render_template, url_for

loaded=CountVectorizer(decode_error='replace',vocabulary=pickle.load(open('word_features.pkl','rb')))

app = Flask(__name__)

def clean_text(text):
    text = text.lower()
    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", "cannot ", 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)
    text = re.sub('\W', ' ', text)
    text = re.sub('\s+', ' ', text)
    text = text.strip(' ')
    return text

@app.route('/')
def landingpage():
    img_url = url_for('static', filename = 'images/hello.png')
    print(img_url)
    flag=0
    return render_template('toxic.html', flag=flag)
```



```
@app.route('/predict')
@app.route('/', methods = ['GET','POST'])
def predict():
    if request.method == 'GET':
        img_url = url_for('static',filename = 'images/hello.png')
        return render_template('toxic.html', url=img_url)
    if request.method == 'POST':
        comment = request.form['comment']
        new_row = {'comment_text':comment}
        user_df = pd.DataFrame(columns = ['comment_text'])
        user_df = user_df.append(new_row,ignore_index = True)
        user_df['comment_text'] = user_df['comment_text'].map(lambda com : clean_text(com))
        user_text = user_df['comment_text']
        user_features = loaded.transform(user_text)
        cols_target = ['obscene', 'insult','toxic','severe_toxic','identity_hate','threat']
        lst= []
        mapper = {}
        for label in cols_target:
            filename = str(label+'_model.sav')
            print(filename)
            model = pickle.load(open(filename, 'rb'))
            print('... Processing {}'.format(label))
            user_y_prob = model.predict_proba(user_features)[:,1]
            print(label,":",user_y_prob[0])
            lst.append([label,user_y_prob])
        print(lst)

        final=[]
        flag=0
        for i in lst:
            if i[1]>0.5:
                final.append(i[0])
                flag=2
        if not len(final):
            text = "Yaayy!! The comment is clean"
            img_url = url_for('static',filename = 'images/happy.png')
```


File Edit Selection View Go Run Terminal Help
commentApp.py - Visual Studio Code
Get Started
testing comment toxicity.py
commentApp.py
C: > Users > SADHU SUNDHAR > Desktop > pt 2 > Commet-Toxicity-Multi-Class-Classification-main > Commet-Toxicity-Multi-Class-Classification-main > Flask > commentApp.py > ...
lst= []
mapper = {}
for label in cols_target:
filename = str(label+'_model.sav')
print(filename)
model = pickle.load(open(filename, 'rb'))
print('... Processing {}'.format(label))
user_y_prob = model.predict_proba(user_features)[:,1]
print(label,":",user_y_prob[0])
lst.append([label,user_y_prob])
print(lst)
final=[]
flag=0
for i in lst:
if i[1]>0.5:
final.append(i[0])
flag=2
if not len(final):
text = "Yaayy!! The comment is clean"
img_url = url_for('static',filename = 'images/happy.png')
flag=1
print(img_url)
else:
text="The comment is "
for i in final:
text = text+i+" "
img_url = url_for('static',filename = 'images/toxic.png')
print(text)
return render_template('toxic.html',ypred = text,url= img_url,flag=flag)
if __name__ == '__main__':
app.run(host = 'localhost', debug = False , threaded = False)
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.7 ('base': conda)
ENG IN
17:52 09-04-2022


File Edit Selection View Go Run Terminal Help
commentApp.py - Visual Studio Code
Get Started
testing comment toxicity.py
commentApp.py
C: > Users > SADHU SUNDHAR > Desktop > pt 2 > Commet-Toxicity-Multi-Class-Classification-main > Commet-Toxicity-Multi-Class-Classification-main > Flask > commentApp.py > ...
lst= []
mapper = {}
for label in cols_target:
filename = str(label+'_model.sav')
print(filename)
model = pickle.load(open(filename, 'rb'))
print('... Processing {}'.format(label))
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
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.7 ('base': conda)
ENG IN
17:52 09-04-2022

b. UI Output Screenshot

FIG:1 HOME PAGE

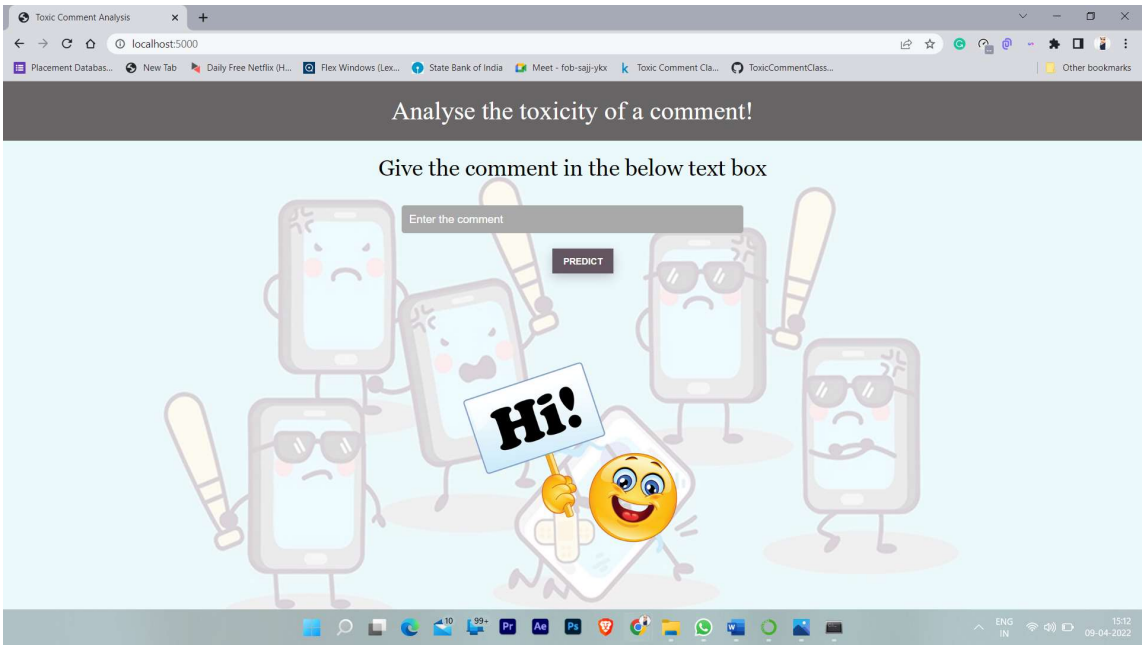


FIG:2 ENTER A TOXIC COMMENT

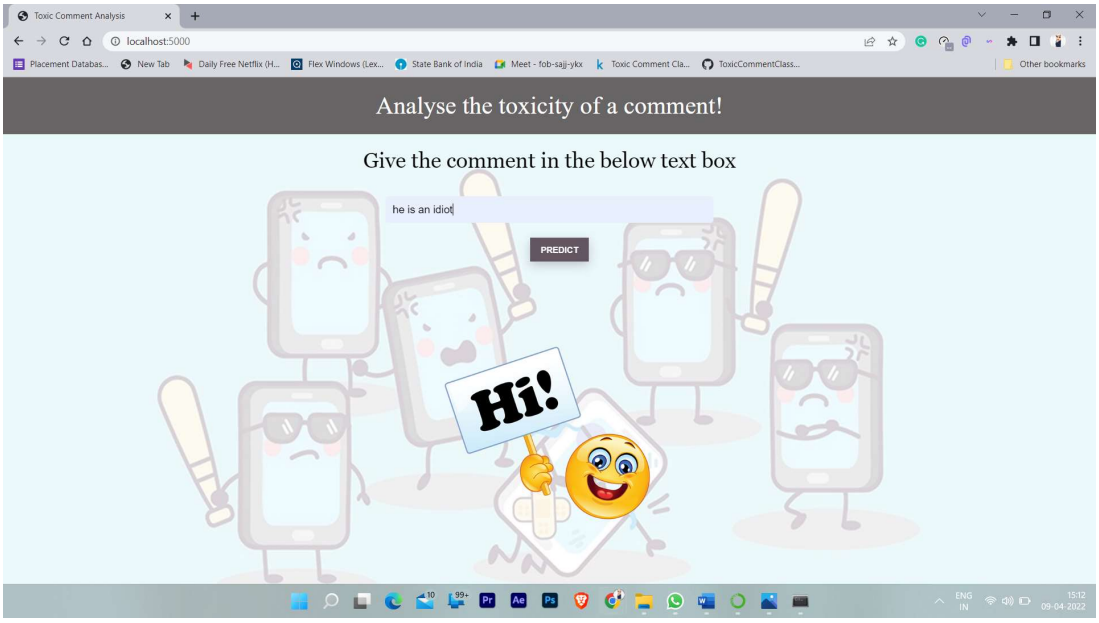


FIG:3 SHOWING OUTPUT THE COMMENT IS TOXIC

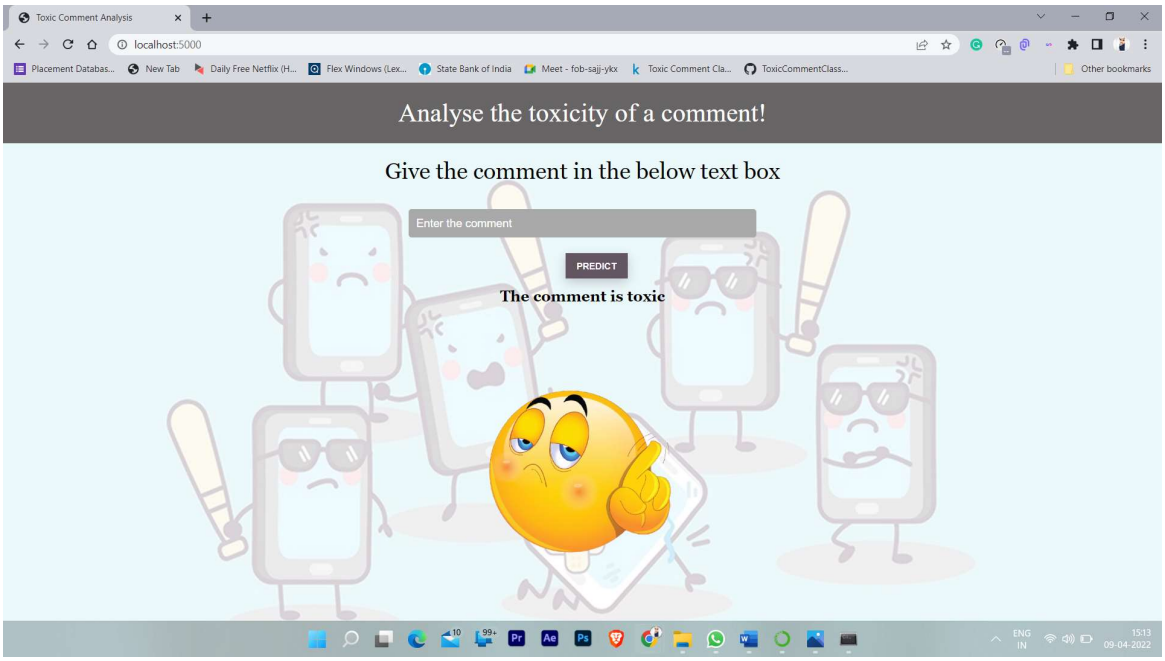


FIG:4 ENTERING NORMAL COMMENT

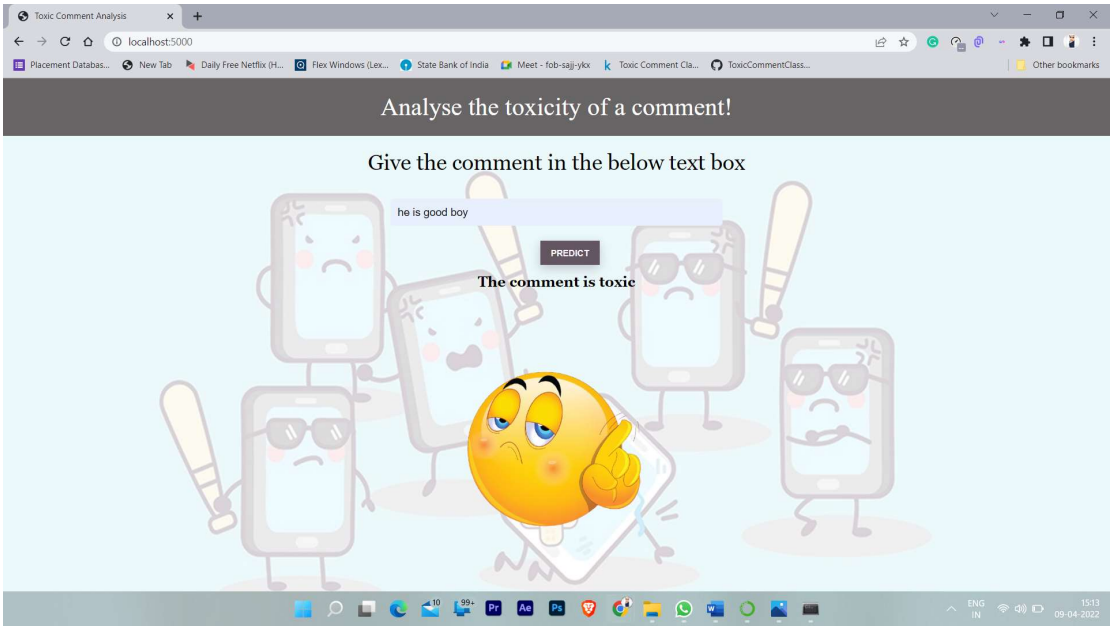
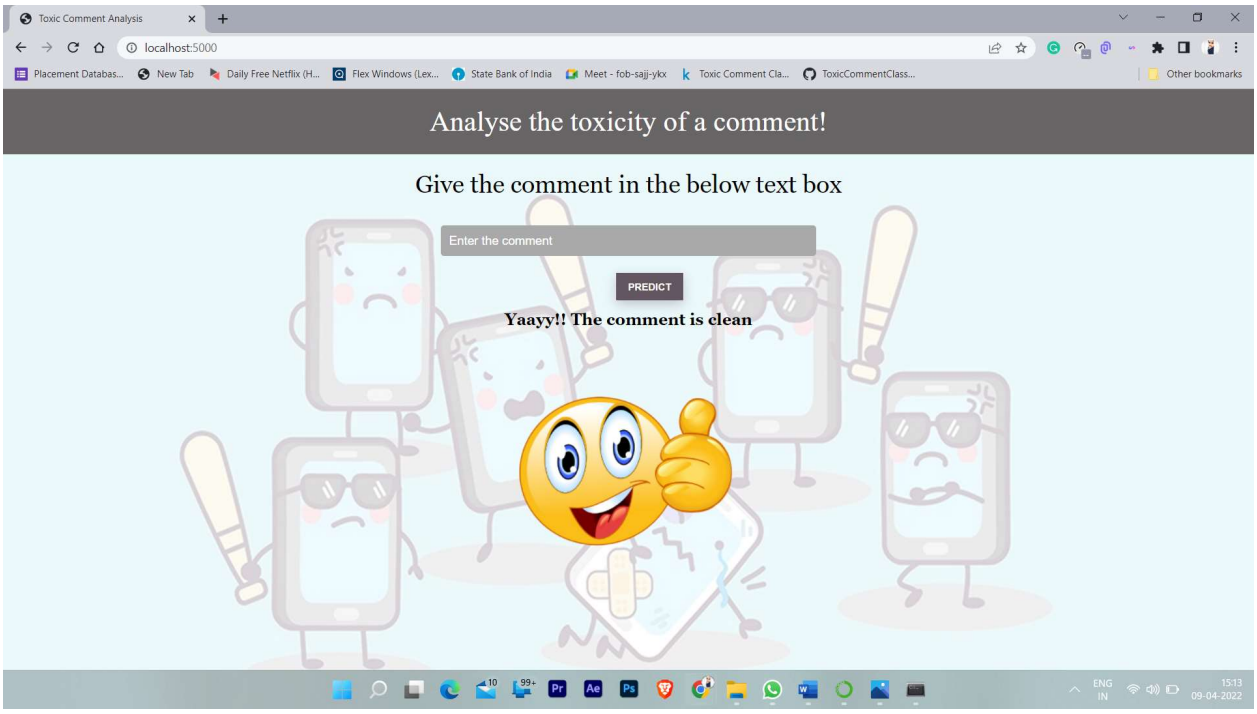


FIG:3 SHOWING OUTPUT THE COMMENT IS CLEAN



THANK YOU

Submitted By

PENTAPATI ESWAR AKHIL

39110765