FLIP ROBO

Malignant Comments Classifier Prediction Project

Submitted by:

Archana Kumari

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to all the teachers who have taught me Machine Learning and NLP. Because of the knowledge they had provided to me I am able to complete this project.

INTRODUCTION

- ## Business Problem Framing

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour. There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a call on anyone and affect them mentally leading to depression, mental illness, self-hatred
and suicidal thoughts.

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as un-offensive, but “u ae an idiot” is clearly offensive.

Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying

- ## Conceptual Background of the Domain Problem

In the past few years its seen that the cases related to social media hatred have increased exponentially. The social media is turning into a dark venomous pit for people now a days. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc.

In social media the people spreading or involved in such kind of activities uses filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.

The result of such activities can be dangerous. It gives mental trauma to the victims making their lives miserable. People who are not well aware of mental health online hate or cyber bullying become life threatening for them. Such cases are also at rise. It is also taking its toll on religions. Each and every day we can see an incident of fighting between people of different communities or religions due to offensive social media posts.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity has been identified as a major threat on online social media platforms. These kinds of activities must be checked for a better future

- ## Review of Literature

Aggression by text is a complex phenomenon, and different knowledge fields try to study and tackle this problem.

This analysis of related work focuses on a computer science perspective of aggression identification, a recent emerging area. Currently, the scientific study of automatic identification of aggressive text, using information technology techniques, is increasing. In this study, several related literature are used to express different types of aggression. Some of those are hate , cyber bullying (, abusive language , toxicity .

- ## Motivation for the Problem Undertaken

The project was the first provided to me by FlipRobo as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

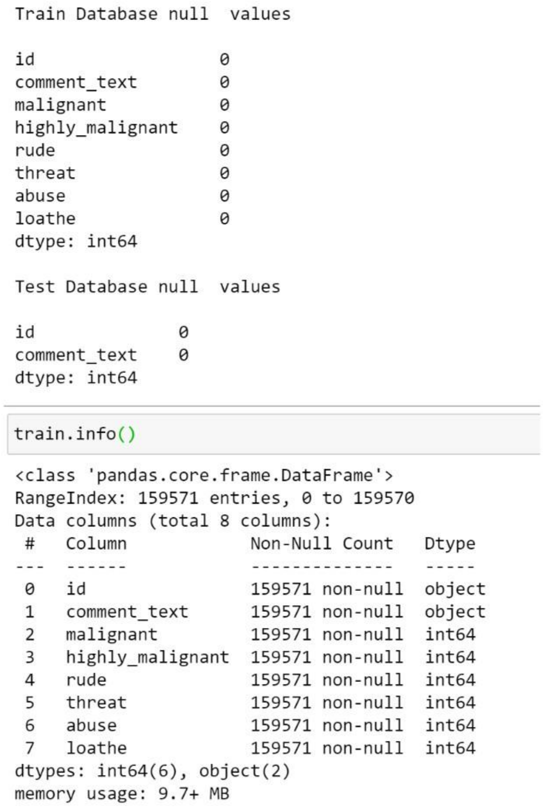
  - Project contain train and test dataset as well.
  - In train data set there are 159,571 rows and 8 columns.
  - There are no null values in the dataset

- Most of the data are numeric in nature which are binary.
- Comments is object in nature and consist of text

Data Sources and their formats

The data was provided by FlipRobo in CSV format. After loading the training dataset into Jupyter Notebook using Pandas and it can be seen that there are eight columns named as:
“ id, comment_text, “malignant, highly_malignant, rude, threat, abuse, loathe**”.**

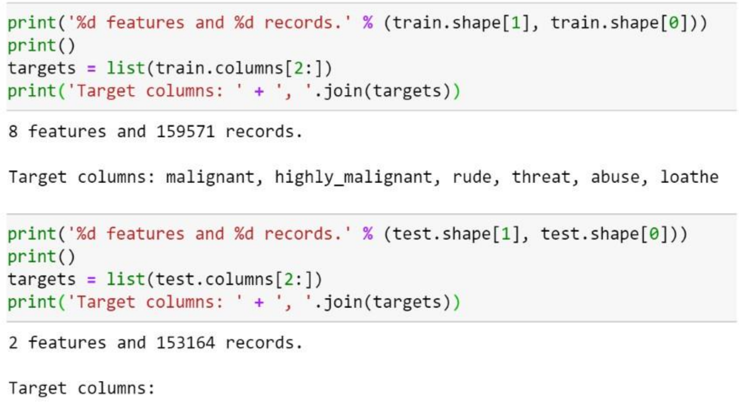
There are 8 columns in the dataset provided:
The description of each of the column is given below:
Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
Highly Malignant: It denotes comments that are highly malignant and hurtful.
Rude: It denotes comments that are very rude and offensive.
Threat: It contains indication of the comments that are giving any threat to someone.
Abuse: It is for comments that are abusive in nature.
Loathe: It describes the comments which are hateful and loathing in nature.
ID: It includes unique Ids associated with each comment text given.
Comment text: This column contains the comments extracted from various social media platforms.



```
Train Database null  values

id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64

Test Database null  values

id              0
comment_text    0
dtype: int64
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 2   malignant         159571 non-null  int64
 3   highly_malignant  159571 non-null  int64
 4   rude              159571 non-null  int64
 5   threat            159571 non-null  int64
 6   abuse             159571 non-null  int64
 7   loathe            159571 non-null  int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```



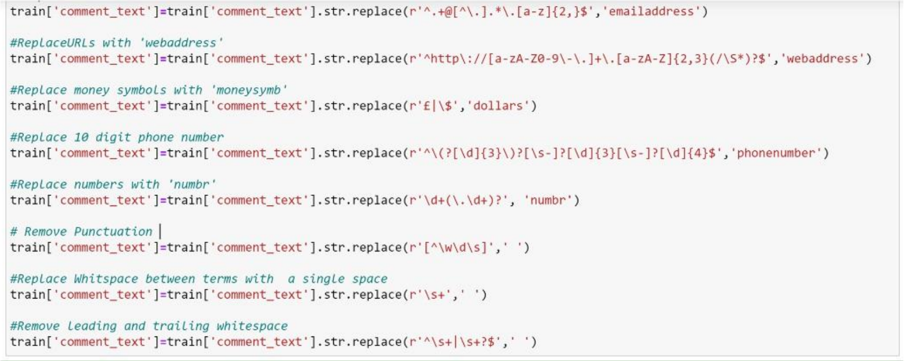
```
print('%d features and %d records.' % (train.shape[1], train.shape[0]))
print()
targets = list(train.columns[2:])
print('Target columns: ' + ', '.join(targets))
```

```
8 features and 159571 records.

Target columns: malignant, highly_malignant, rude, threat, abuse, loathe
```

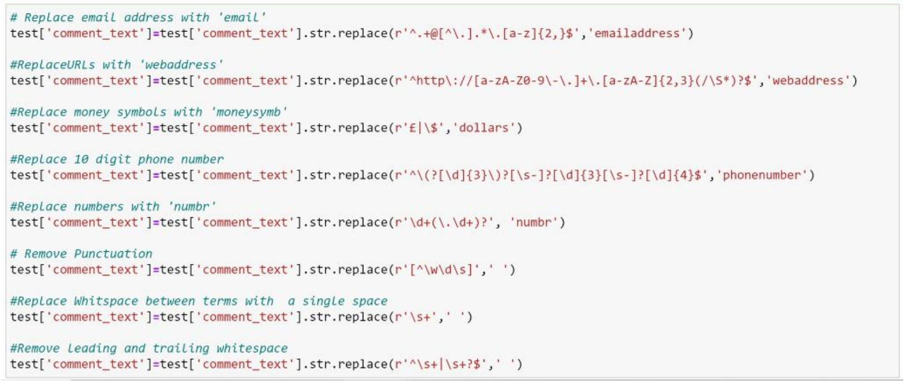
```
print('%d features and %d records.' % (test.shape[1], test.shape[0]))
print()
targets = list(test.columns[2:])
print('Target columns: ' + ', '.join(targets))
```

```
2 features and 153164 records.

Target columns:
```

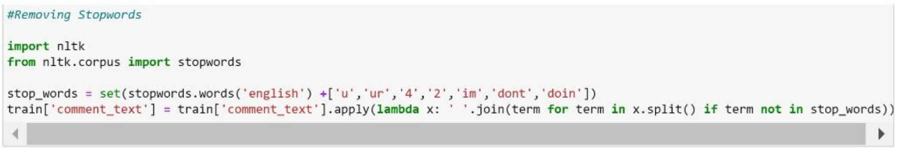
- Data Preprocessing Done



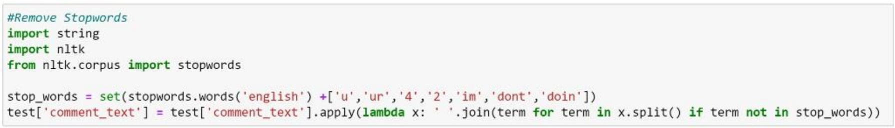
```
train['comment_text']=train['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

#ReplaceURLs with 'webaddress'
train['comment_text']=train['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')

#Replace money symbols with 'moneysymb'
train['comment_text']=train['comment_text'].str.replace(r'£|\$','dollars')

#Replace 10 digit phone number
train['comment_text']=train['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

#Replace numbers with 'numbr'
train['comment_text']=train['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')

# Remove Punctuation |
train['comment_text']=train['comment_text'].str.replace(r'[^\w\d\s]',' ')

#Replace Whitspace between terms with  a single space
train['comment_text']=train['comment_text'].str.replace(r'\s+',' ')

#Remove leading and trailing whitespace
train['comment_text']=train['comment_text'].str.replace(r'^\s+|\s+?$',' ')
```



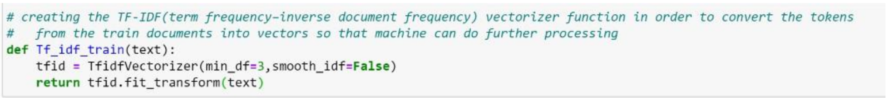
```
# Replace email address with 'email'
test['comment_text']=test['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

#ReplaceURLs with 'webaddress'
test['comment_text']=test['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')

#Replace money symbols with 'moneysymb'
test['comment_text']=test['comment_text'].str.replace(r'£|\$','dollars')

#Replace 10 digit phone number
test['comment_text']=test['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

#Replace numbers with 'numbr'
test['comment_text']=test['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')

# Remove Punctuation
test['comment_text']=test['comment_text'].str.replace(r'[^\w\d\s]',' ')

#Replace Whitspace between terms with  a single space
test['comment_text']=test['comment_text'].str.replace(r'\s+',' ')

#Remove leading and trailing whitespace
test['comment_text']=test['comment_text'].str.replace(r'^\s+|\s+?$',' ')
```



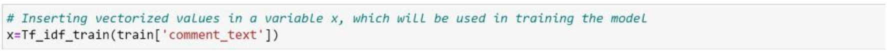
```
#Removing Stopwords

import nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english') +['u','ur','4','2','im','dont','doin'])
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```



```
#Remove Stopwords
import string
import nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english') +['u','ur','4','2','im','dont','doin'])
test['comment_text'] = test['comment_text'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```



```
# creating the TF-IDF(term frequency-inverse document frequency) vectorizer function in order to convert the tokens
#   from the train documents into vectors so that machine can do further processing
def Tf_idf_train(text):
    tfid = TfidfVectorizer(min_df=3,smooth_idf=False)
    return tfid.fit_transform(text)
```



```
# Inserting vectorized values in a variable x, which will be used in training the model
x=Tf_idf_train(train['comment_text'])
```

For Data pre-processing we did some data cleaning, and filter the words by removing stop words.
Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms.

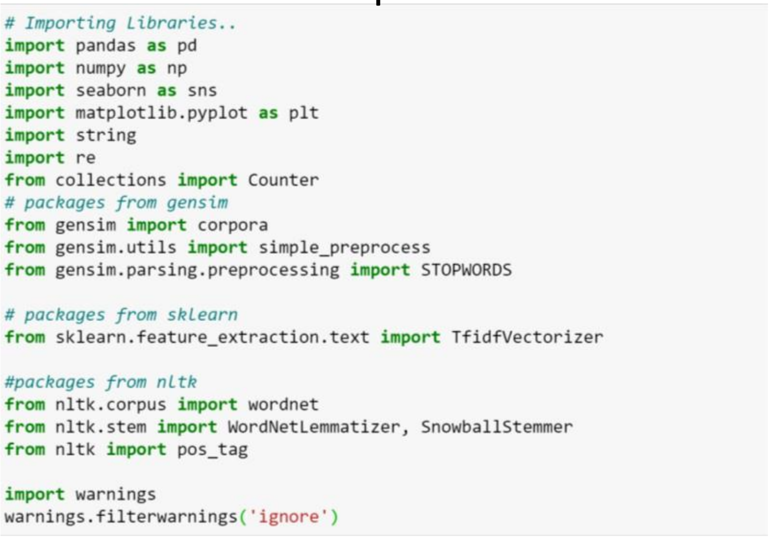
- ## Data Inputs- Logic- Output Relationships

There are features which are highly correlated.
It is possible because one comment can be classified into multiple categories.
One comment can be rude and abuse at same time.
Most of the text length are within 500 characters, with some up to 5,000 characters long.

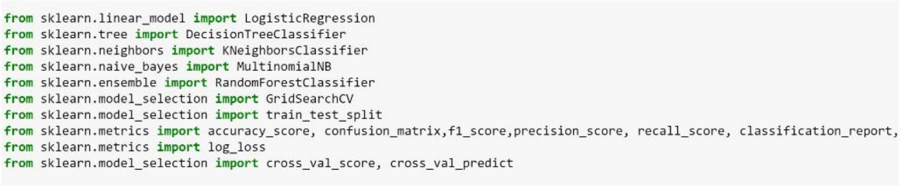
- ## State the set of assumptions (if any) related to the problem under consideration

For the easiness of the training and predicting I assumed and make a label of bad words. Means all the bad comments are classify as bad word and all good comments whose value are 0 are classified as good comments.
This will help the model to classify the comment easily also I had made a separate model which helps to predict the probability of comment text that in which column it has highest probability.

- ## Hardware and Software Requirements and Tools Used



```python
# Importing Libraries..
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import string
import re
from collections import Counter
# packages from gensim
from gensim import corpora
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS

# packages from sklearn
from sklearn.feature_extraction.text import TfidfVectorizer

#packages from nltk
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk import pos_tag

import warnings
warnings.filterwarnings('ignore')
```



```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,f1_score,precision_score, recall_score, classification_report,
from sklearn.metrics import log_loss
from sklearn.model_selection import cross_val_score, cross_val_predict
```

Above are the libraries which I used to pre-process, predict and visualize the project :

Pandas - It is used to play with data frame and helps in to get more insight of the data, like describing the data and the types of the all features.

Seaborn, Matplotlib – Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots and so on. Seaborn on the other hand, provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Read the use case and search from the google to know more about the use case and in which field or domain it will applicable more.

  As data set is already there, so this step is excluded.

  Data Preparation includes the data cleaning and describing the data which I followed text pre-processing.

  EDA involves the visualization which is helpful to get more insight from the data and get to know about the mostly use comment and check the most frequent words.

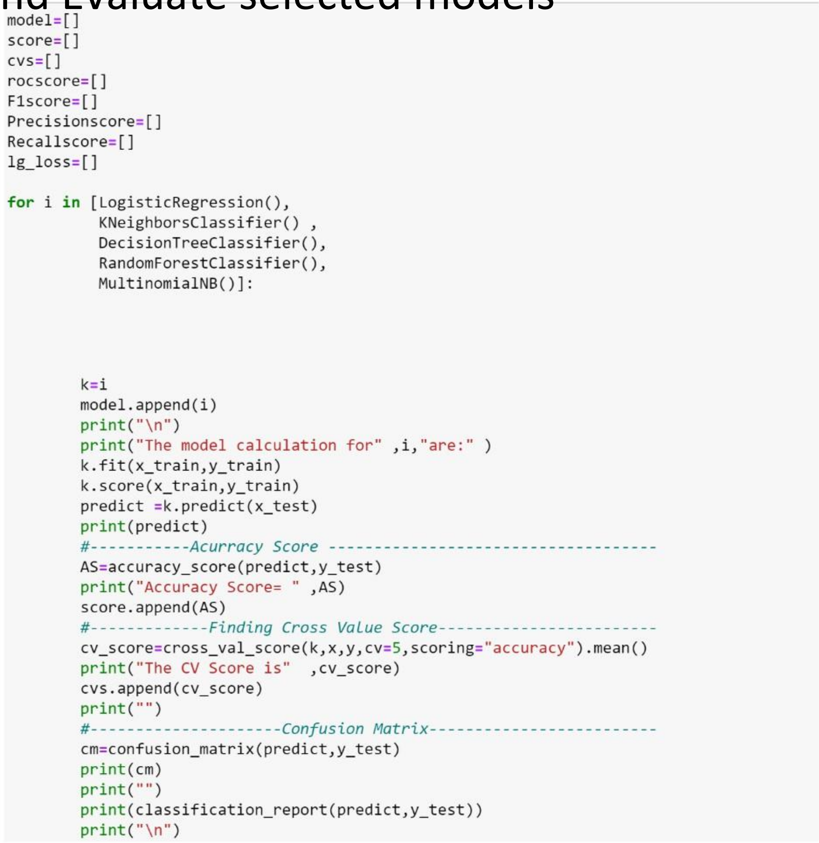
  Modelling involves creating the model with suitable algorithm which provide the best result, I tried multiple algo and apply hyper-parameter tuning.

  Model Evaluation, for this I used confusion metrices and mainly focus on False negative and tried to reduce the False negative which is type 2 error and on F1 score as dataset is imbalance.

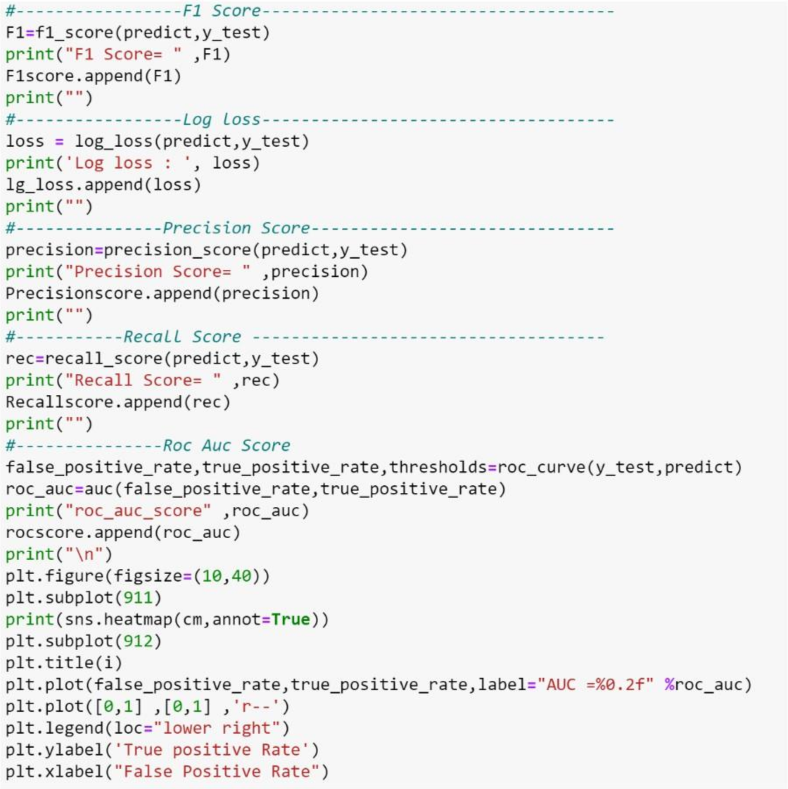
- Testing of Identified Approaches (Algorithms)
  - lr=LogisticRegression()
  - mnb=MultinomialNB()
  - rfc=RandonForestClassifier()

- dtc=DecisionTreeClassifier()
- knc=KNeighborsClassifier()

- Run and Evaluate selected models

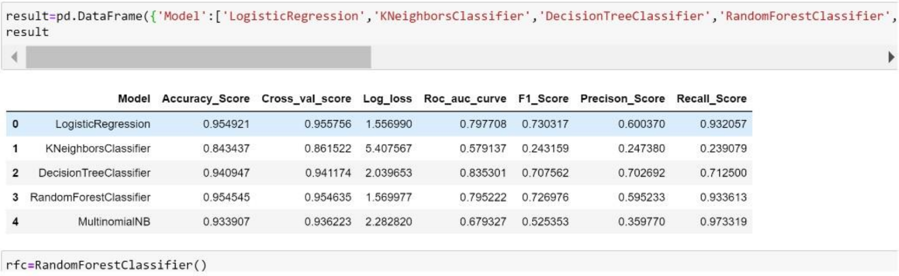


```
model=[]
score=[]
cvs=[]
rocscore=[]
F1score=[]
Precisionscore=[]
Recallscore=[]
lg_loss=[]

for i in [LogisticRegression(),
          KNeighborsClassifier() ,
          DecisionTreeClassifier(),
          RandomForestClassifier(),
          MultinomialNB()]:




        k=i
        model.append(i)
        print("\n")
        print("The model calculation for" ,i,"are:" )
        k.fit(x_train,y_train)
        k.score(x_train,y_train)
        predict =k.predict(x_test)
        print(predict)
        #-----------Acurracy Score -----------------------------------
        AS=accuracy_score(predict,y_test)
        print("Accuracy Score= " ,AS)
        score.append(AS)
        #------------Finding Cross Value Score-----------------------
        cv_score=cross_val_score(k,x,y,cv=5,scoring="accuracy").mean()
        print("The CV Score is"  ,cv_score)
        cvs.append(cv_score)
        print("")
        #--------------------Confusion Matrix------------------------
        cm=confusion_matrix(predict,y_test)
        print(cm)
        print("")
        print(classification_report(predict,y_test))
        print("\n")
```



```
#-----------------F1 Score------------------------------------
F1=f1_score(predict,y_test)
print("F1 Score= " ,F1)
F1score.append(F1)
print("")
#-----------------Log Loss------------------------------------
loss = log_loss(predict,y_test)
print('Log loss : ', loss)
lg_loss.append(loss)
print("")
#---------------Precision Score-------------------------------
precision=precision_score(predict,y_test)
print("Precision Score= " ,precision)
Precisionscore.append(precision)
print("")
#-----------Recall Score ------------------------------------
rec=recall_score(predict,y_test)
print("Recall Score= " ,rec)
Recallscore.append(rec)
print("")
#---------------Roc Auc Score
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,predict)
roc_auc=auc(false_positive_rate,true_positive_rate)
print("roc_auc_score" ,roc_auc)
rocscore.append(roc_auc)
print("\n")
plt.figure(figsize=(10,40))
plt.subplot(911)
print(sns.heatmap(cm,annot=True))
plt.subplot(912)
plt.title(i)
plt.plot(false_positive_rate,true_positive_rate,label="AUC =%0.2f" %roc_auc)
plt.plot([0,1] ,[0,1] ,'r--')
plt.legend(loc="lower right")
plt.ylabel('True positive Rate')
plt.xlabel("False Positive Rate")
```

- ## Key Metrics for success in solving problem under consideration.



```
result=pd.DataFrame({'Model':['LogisticRegression','KNeighborsClassifier','DecisionTreeClassifier','RandomForestClassifier',
result
```

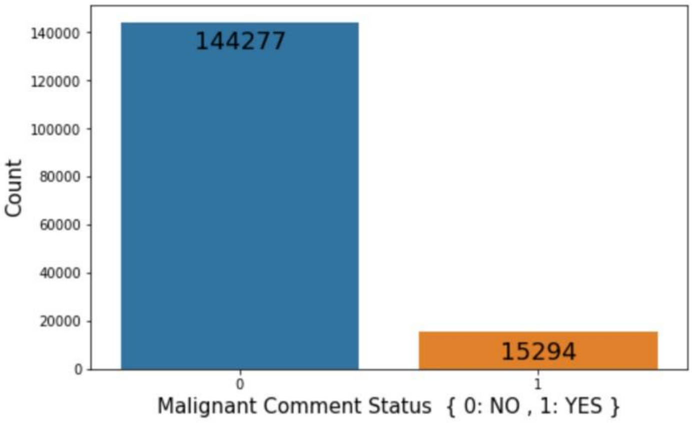
	Model	Accuracy_Score	Cross_val_score	Log_loss	Roc_auc_curve	F1_Score	Precison_Score	Recall_Score
0	LogisticRegression	0.954921	0.955756	1.556990	0.797708	0.730317	0.600370	0.932057
1	KNeighborsClassifier	0.843437	0.861522	5.407567	0.579137	0.243159	0.247380	0.239079
2	DecisionTreeClassifier	0.940947	0.941174	2.039653	0.835301	0.707562	0.702692	0.712500
3	RandomForestClassifier	0.954545	0.954635	1.569977	0.795222	0.726976	0.595233	0.933613
4	MultinomialNB	0.933907	0.936223	2.282820	0.679327	0.525353	0.359770	0.973319

```
rfc=RandomForestClassifier()
```

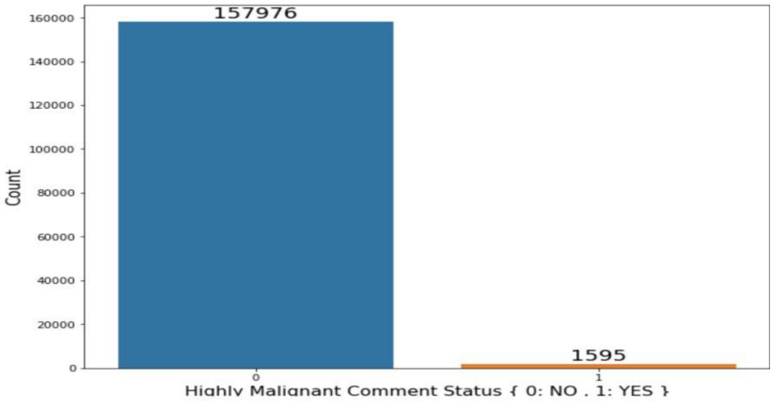
Key Metrices used were the Accuracy Score, Crossvalidation Score and AUC & ROC Curve as this was binary classification. From the above we can see that there are various models out of which we few gave good accuracy score as more than 90%,

- ## Visualizations

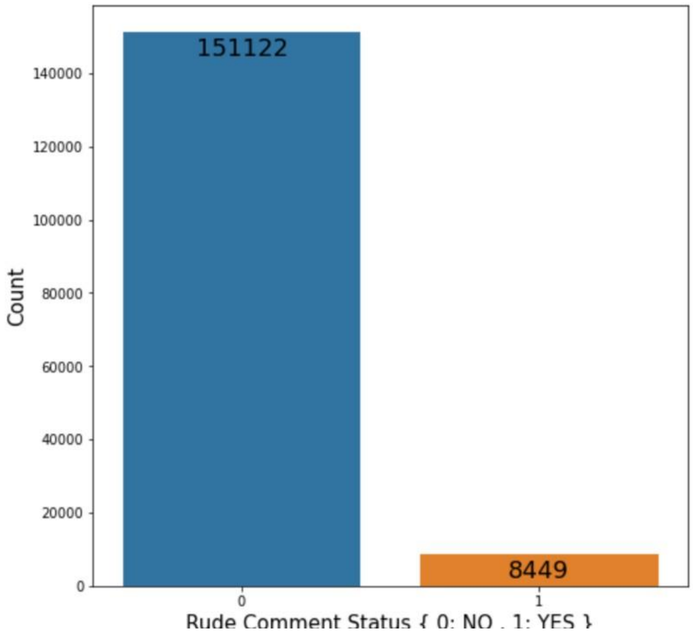
Let's see the Count plot of Malignant comments Status



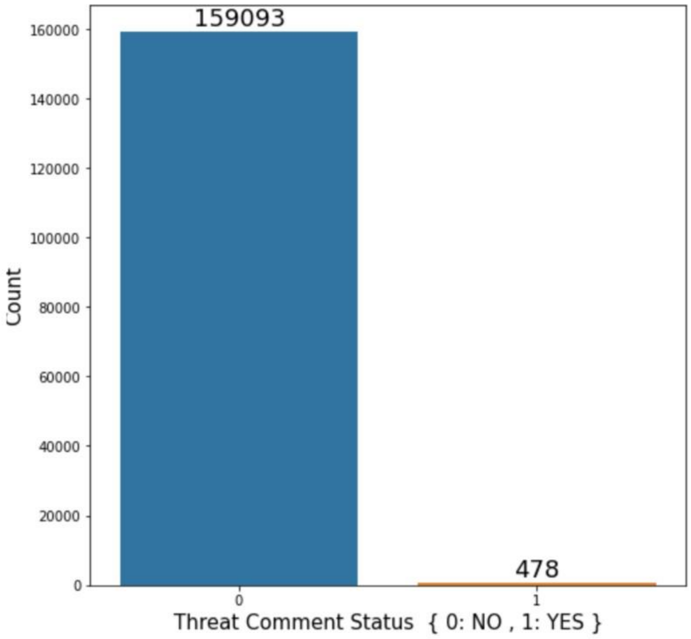
Let's see the Count plot of Highly Malignant comments status.



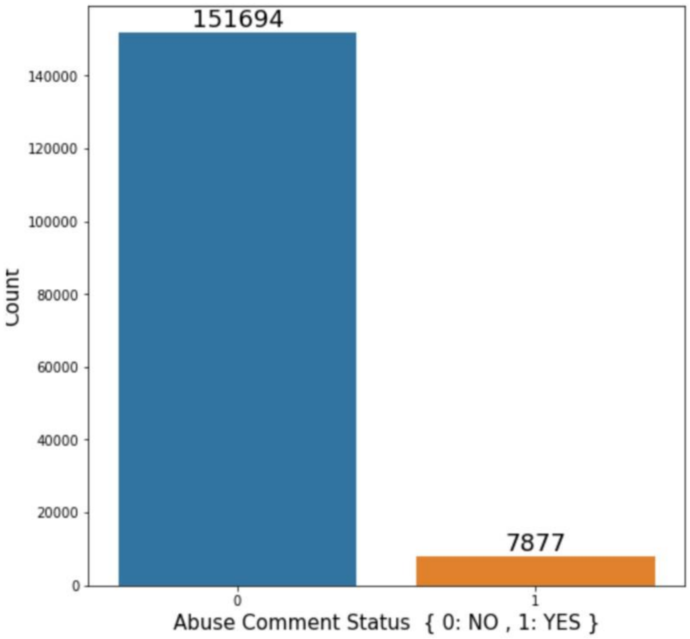
Let's see the Count plot of Rude comment Status



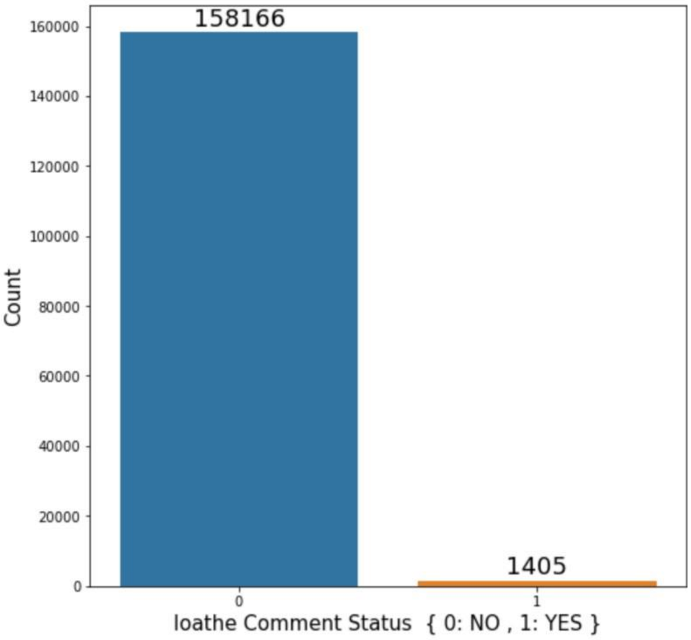
Let’s see the Count-plot of Threat Comments Status



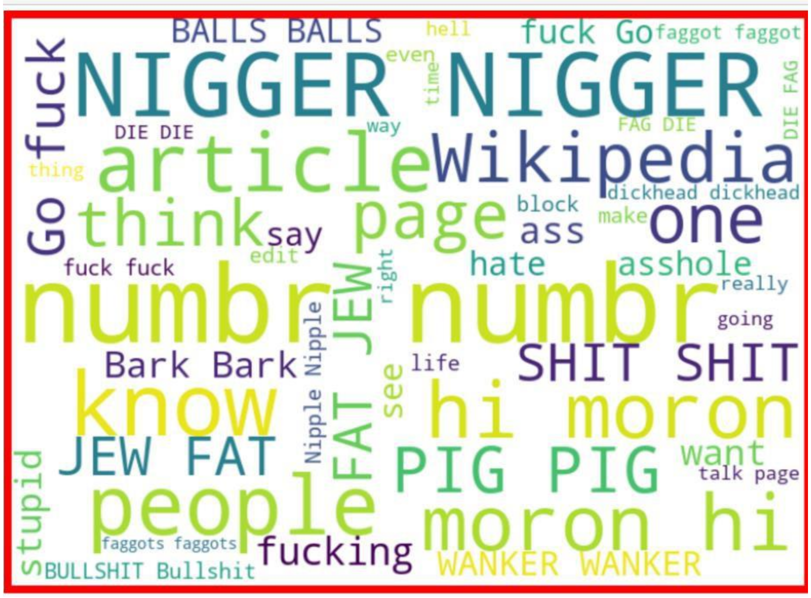
Let’s see the Countplot of Abuse Comment Status



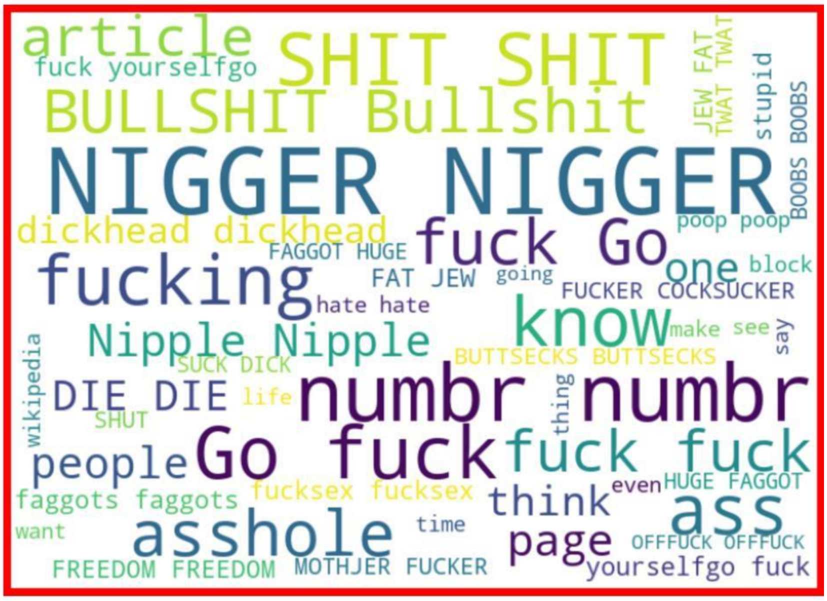
Let’s see the Count-plot of loathe Comment Status



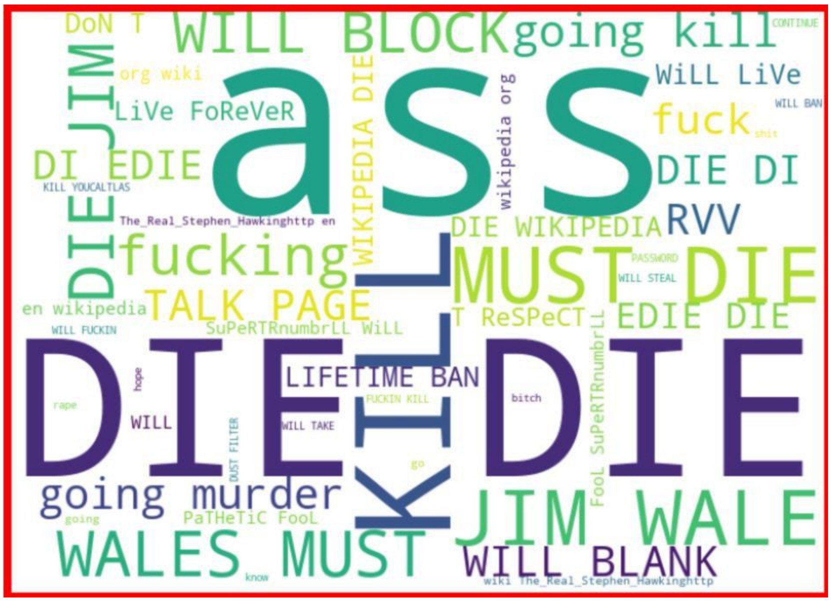
Let’s see the loud words in Comments which are marked as Malignant



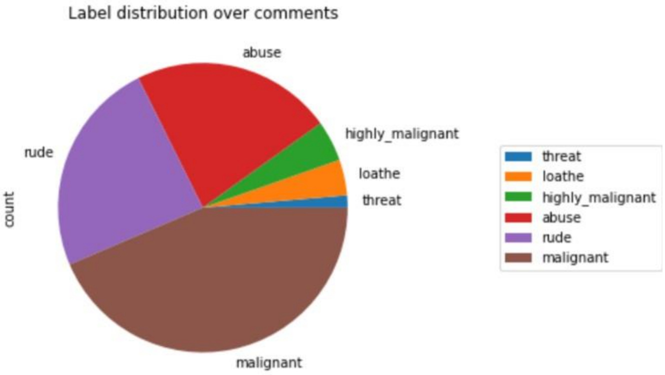
Let's see the loud words in Comments which are marked as rude.



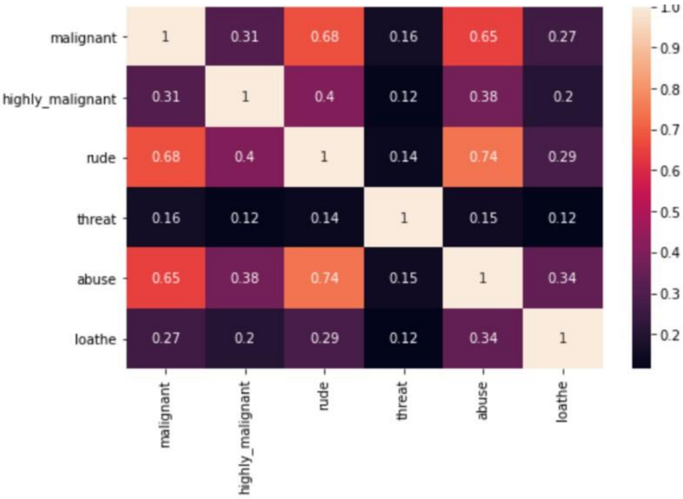
Let's see the loud words in Comments which are marked as threat.



Making Pie chart of 'malignant', 'threat' ,'abuse' ,'loathe', 'highly_malignant', 'rude' .



Correlation heatmap :



- ## Interpretation of the Results

- From the above visualization and matrices found that the Random Forest Classifier performed the best Accuracy Score of**i.e. 95.45%.**

CONCLUSION

- ## Key Findings and Conclusions of the Study

Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.

From the above analysis the below mentioned results were achieved which depicts the chances and conditions of a comment being a hateful comment or a normal comment.

With the increasing popularity of social media, more and more people consume feeds from social media and due differences they spread hate comments to instead of love and harmony. It has strong negative impacts on individual users and broader society.

- ## Learning Outcomes of the Study in respect of Data Science

It is possible to classify the comments content into the required categories of Malignant and Non Malignant. However, using this kind of project an awareness can be created to know what is good and bad. It will help to stop spreading hatred among people.

- ## Limitations of this work and Scope for Future Work

Machine Learning Algorithms like Decision Tree Classifier took enormous amount of time to build the model and Ensemble techniques were taking a lot more time thus I have not included Ensemble models

Every effort has been put on it for perfection but nothing is perfect and this project is of no exception. There are certain areas which can be enhanced. Comment detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field

It has future scope in various use cases likewise in election, social media etc, where every day there are multi offensive comments spread.

So, in the future it may use very well to easily classify the comments as bad or good