MALIGNANT COMMENTS CLASSIFIER PROJECT

Submitted by:

Kishan Verma

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to my SME, Mr. Shubham Yadav for his constant guidance. Some of the reference sources are as follows:

- Stack Overflow
- Medium.com
- scikit-learn.org
- Python official documentation

Contents

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 toll 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 unoffensive, but “u are 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

REVIEW OF LITERATURE

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.

MOTIVATION FOR THE PROBLEM UNDERTAKEN

The project is 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

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 and loathe**.**

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.

```
Features Present in the Dataset:
 Index(['id', 'comment_text', 'malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe'],
      dtype='object')

Total Number of Rows :  159571
Total Number of Features :  8


Data Types of Features :
 id                  object
comment_text        object
malignant            int64
highly_malignant     int64
rude                 int64
threat               int64
abuse                int64
loathe               int64
dtype: object
```

DATA PREPROCESSING DONE

For Data pre-processing we did some data cleaning, where we used wordNet lemmatizer to clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizers and joined and return the filtered 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.

```
#     Here I have made a function in which all the Data cleaning steps like removing data which is not useful like
#       email adress, mobile numbers,removing punctuations, converting all the documents into lowercase,
#           using lemmatization technique, filtering documents using Stopwords, using POS tagging,
#      all these type of data preprocessing steps are being perormed with th ehelp of the function defined below.

# function to filter using POS tagging..
def get_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

# Function for data cleaning...
def Processed_data(comments):
    # Replace email addresses with 'email'
    comments=re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', comments)

    # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
    comments=re.sub(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',' ',comments)

    # getting only words(i.e removing all the special characters)
    comments = re.sub(r'[^\w]', ' ', comments)

    # getting only words(i.e removing all the" _ ")
    comments = re.sub(r'[\_]', ' ', comments)

    # getting rid of unwanted characters(i.e remove all the single characters left)
    comments=re.sub(r'\s+[a-zA-Z]\s+', ' ', comments)

    # Removing extra whitespaces
    comments=re.sub(r'\s+', ' ', comments, flags=re.I)

    #converting all the letters of the review into lowercase
    comments = comments.lower()

    # splitting every words from the sentences
    comments = comments.split()

    # iterating through each words and checking if they are stopwords or not,
    comments=[word for word in comments if not word in set(STOPWORDS)]

    # remove empty tokens
    comments = [text for text in comments if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(comments)

    # considering words having length more than 3only
    comments = [text for text in comments if len(text) > 3]

    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS ...
    comments = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]

   # considering words having length more than 3 only
    comments = [text for text in comments if len(text) > 3]
    comments = ' '.join(comments)
    return comments
```

```
df["clean_comment_text"] = df["comment_text"].apply(lambda x: Processed_data(x))
```

```
# checking the clean comments content..
df["clean_comment_text"]
```

```
0         explanation edits username hardcore metallica ...
1         match background colour seemingly stuck thanks...
2         trying edit constantly removing relevant infor...
3         real suggestion improvement wondered section s...
4                                 hero chance remember page
                               ...
159566    second time asking view completely contradicts...
159567                      ashamed horrible thing talk page
159568    spitzer there actual article prostitution ring...
159569       look like actually speedy version deleted look
159570    think understand came idea right away kind com...
Name: clean_comment_text, Length: 159571, dtype: object
```

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE:

HP Pavilion X360

SOFTWARE:

Jupyter Notebook (Anaconda 3) – Python 3

Microsoft Office 365 Package

LIBRARIES USED:

```
# 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')
```

```
# Importing useful libraries for model training

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression,PassiveAggressiveClassifier

# Model selection libraries...
from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
from sklearn.model_selection import GridSearchCV


# Importing some metrics we can use to evaluate our model performance....
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import log_loss

# Creating instances for different Classifiers

LR=LogisticRegression()
MNB=MultinomialNB()
PAC=PassiveAggressiveClassifier()
DT=DecisionTreeClassifier()

```

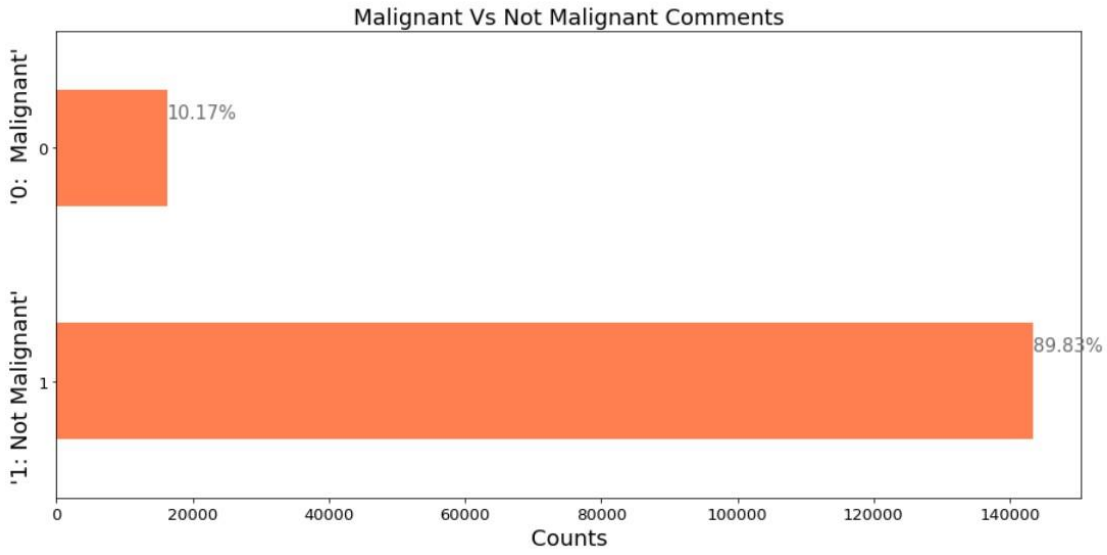
MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

```
# Tokenizing Documents..
data=[]
from nltk.tokenize import word_tokenize
for j,i in enumerate(df['clean_comment_text']):
    a=word_tokenize(i,'english')
    data.append(a)
```

```
# Makin Word dictionary...
dictionary = corpora.Dictionary(data)
print(dictionary)
```

```
Dictionary(167144 unique tokens: ['closure', 'doll', 'edits', 'explanation', 'hardcore']...)
```



TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

The algorithms we used for the training and testing are as follows:-

- Logistic Regression
- Decision Tree Classifier
- Multinomial NB
- Passive Aggressive Classifier

RUN AND EVALUATE SELECTED MODELS

```
#     Putting Scikit-Learn machine learning Models in a list so that it can be used for further evaluation in loop.
models=[]
models.append(('LogisticRegression',LR))
models.append(('MultinomialNB',MNB))
models.append(('PassiveAggressiveClassifier',PAC))
models.append(('DecisionTreeClassifier',DT))
```

```
#        Function which will find best Random State and then calculate Maximum Accuracy Score corresponding to it
#                                  and print accuracy score in one go.
def max_acc_score(clf,x,y):
    max_acc_score=0
    final_r_state=0
    for r_state in range(42,100):
        x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=r_state,stratify=y)
        clf.fit(x_train,y_train)
        y_pred=clf.predict(x_test)
        acc_score=accuracy_score(y_test,y_pred)
        if acc_score > max_acc_score:
            max_acc_score=acc_score
            final_r_state=r_state
    print('Max Accuracy Score corresponding to Random State ', final_r_state, 'is:', max_acc_score)
    print('\n')
    return final_r_state
```

```
#     Lists to store model name, Learning score, Accuracy score, cross_val_score, Auc Roc score .
Model=[]
Score=[]
Acc_score=[]
cvs=[]
rocscore=[]
lg_loss=[]
#            For Loop to Calculate Accuracy Score, Cross Val Score, Classification Report, Confusion Matrix

for name,model in models:
    print('**************************',name,'******************************')
    print('\n')
    Model.append(name)
    print(model)
    print('\n')

     #        Now here I am calling a function which will calculate the max accuracy score for each model
     #                                  and return best random state.
    r_state=max_acc_score(model,x,y)
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=r_state,stratify=y)
    model.fit(x_train,y_train)
#..............Learning Score...........
    score=model.score(x_train,y_train)
    print('Learning Score : ',score)
    Score.append(score*100)
    y_pred=model.predict(x_test)
    acc_score=accuracy_score(y_test,y_pred)
    print('Accuracy Score : ',acc_score)
    Acc_score.append(acc_score*100)

#................Finding Cross_val_score..................
    cv_score=cross_val_score(model,x,y,cv=10,scoring='roc_auc').mean()
    print('Cross Val Score : ', cv_score)
    cvs.append(cv_score*100)

```

```
#.............Roc auc score.........................
    false_positive_rate,true_positive_rate, thresholds=roc_curve(y_test,y_pred)
    roc_auc=auc(false_positive_rate, true_positive_rate)
    print('roc auc score : ', roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')

    loss = log_loss(y_test,y_pred)
    print('Log loss : ', loss)
    lg_loss.append(loss)
    print('\n')

#.............Classification Report.........................
    print('Classification Report:\n',classification_report(y_test,y_pred))
    print('\n')

    print('Confusion Matrix:\n',confusion_matrix(y_test,y_pred))
    print('\n')
```

```
************************* LogisticRegression ****************************


LogisticRegression()


Max Accuracy Score corresponding to Random State  44 is: 0.9543574532085561


Learning Score :  0.9570721313530112
Accuracy Score :  0.9543574532085561
Cross Val Score :  0.9647437145554436
roc auc score :  0.7934277979300925


Log loss :  1.5764709396342023


Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.59      0.72      4868
           1       0.96      1.00      0.98     43004

    accuracy                           0.95     47872
   macro avg       0.95      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872



Confusion Matrix:
 [[ 2879  1989]
 [  196 42808]]
```

```
************************** MultinomialNB ****************************


MultinomialNB()


Max Accuracy Score corresponding to Random State  54 is: 0.9368524398395722


Learning Score :  0.9389072417837223
Accuracy Score :  0.9368524398395722
Cross Val Score :  0.9294912255583274
roc auc score :  0.6948768767912668


Log loss :  2.1810889674255955


Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.39      0.56      4868
           1       0.94      1.00      0.97     43004

    accuracy                           0.94     47872
   macro avg       0.95      0.69      0.76     47872
weighted avg       0.94      0.94      0.92     47872



Confusion Matrix:
 [[ 1904  2964]
 [   59 42945]]
```

```
************************** PassiveAggressiveClassifier ****************************


PassiveAggressiveClassifier()


Max Accuracy Score corresponding to Random State  58 is: 0.947735628342246


Learning Score :  0.9897313315249018
Accuracy Score :  0.9460018382352942
Cross Val Score :  0.9359602149598304
roc auc score :  0.8312225661376249


Log loss :  1.8650558733232439


Classification Report:
               precision    recall  f1-score   support

           0       0.76      0.69      0.72      4868
           1       0.96      0.98      0.97     43004

    accuracy                           0.95     47872
   macro avg       0.86      0.83      0.85     47872
weighted avg       0.94      0.95      0.94     47872



Confusion Matrix:
 [[ 3345  1523]
 [ 1062 41942]]
```

```
************************** DecisionTreeClassifier ****************************


DecisionTreeClassifier()


Max Accuracy Score corresponding to Random State  87 is: 0.941949364973262


Learning Score :  0.9982721420961692
Accuracy Score :  0.9420120320855615
Cross Val Score :  0.8346283576844614
roc auc score :  0.8343758433508737


Log loss :  2.0028579103214796


Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.70      0.71      4868
           1       0.97      0.97      0.97     43004

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.84     47872
weighted avg       0.94      0.94      0.94     47872



Confusion Matrix:
 [[ 3404  1464]
 [ 1312 41692]]
```

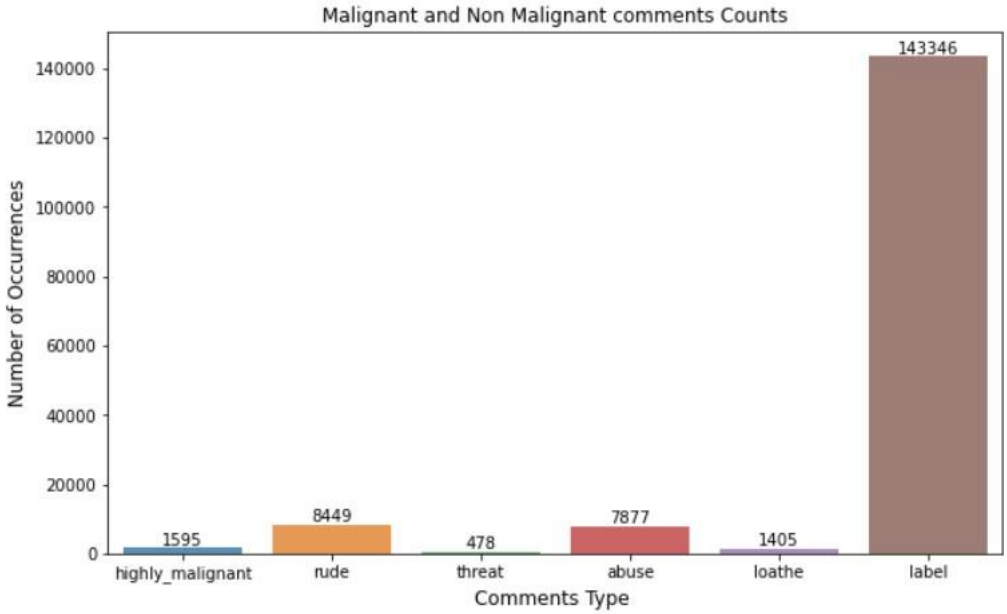
KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

```
# Making a Dataframe comprises of Differnt Calculated Scores :
result=pd.DataFrame({'Model': Model,'Learning Score': Score,'Accuracy Score': Acc_score,'Cross Val Score':cvs,
                     'Roc_Auc_curve':rocscore,'Log_Loss':lg_loss})
result.style.background_gradient(cmap='YlGnBu')
```

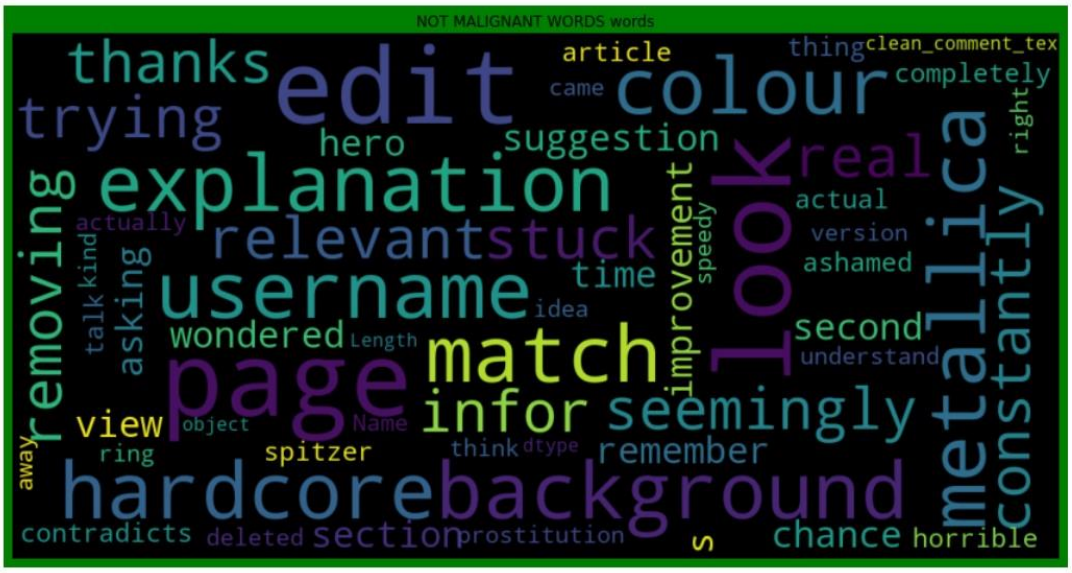
39]:

	Model	Learning Score	Accuracy Score	Cross Val Score	Roc_Auc_curve	Log_Loss
0	LogisticRegression	95.707213	95.435745	96.474371	79.342780	1.576471
1	MultinomialNB	93.890724	93.685244	92.949123	69.487688	2.181089
2	PassiveAggressiveClassifier	98.973133	94.600184	93.596021	83.122257	1.865056
3	DecisionTreeClassifier	99.827214	94.201203	83.462836	83.437584	2.002858

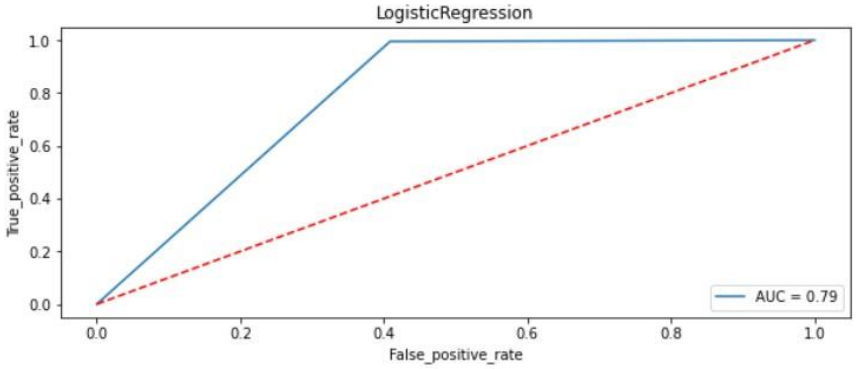
VISUALIZATIONS

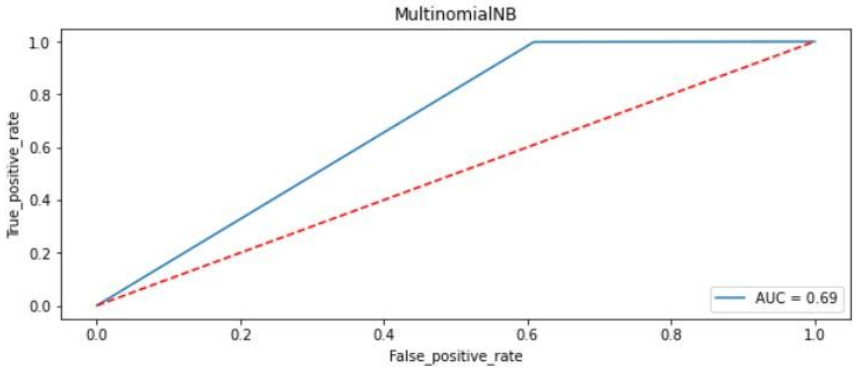


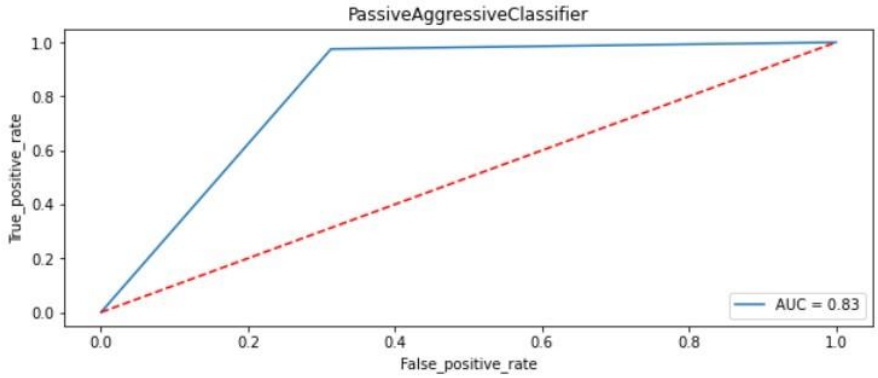

```
# NOT Malignant words in training data
Display_wordcloud(df['clean_comment_text'][df['label']==1],"NOT MALIGNANT WORDS")
```

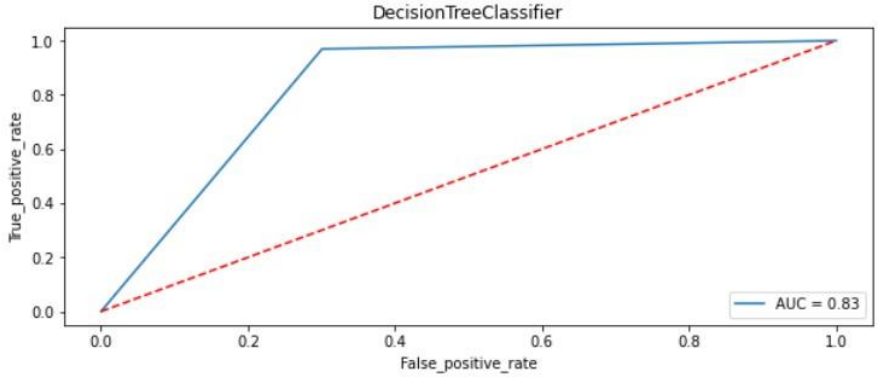


```
# Malignant words in training data
Display_wordcloud(df['clean_comment_text'][df['label']==0],"MALIGNANT WORDS")
```






PassiveAggressiveClassifier
True_positive_rate
False_positive_rate
AUC = 0.83
0.0
0.2
0.4
0.6
0.8
1.0


DecisionTreeClassifier
True_positive_rate
False_positive_rate
AUC = 0.83
0.0
0.2
0.4
0.6
0.8
1.0

INTERPRETATION OF THE RESULTS

From the above visualization and matrices found that the Passive Aggressive Classifier performed the best AUC_ROC_SCORE.

```
# Using PassiveAggressiveClassifier for final model...
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=57,test_size=.30,stratify=y)
PAC=PassiveAggressiveClassifier()
PAC.fit(x_train,y_train)
PAC.score(x_train,y_train)
PACpred=PAC.predict(x_test)
print('Accuracy Score:',accuracy_score(y_test,PACpred))
print('Log loss : ', log_loss(y_test,PACpred))
print('Confusion Matrix:',confusion_matrix(y_test,PACpred))
print('Classification Report:','\n',classification_report(y_test,PACpred))
```

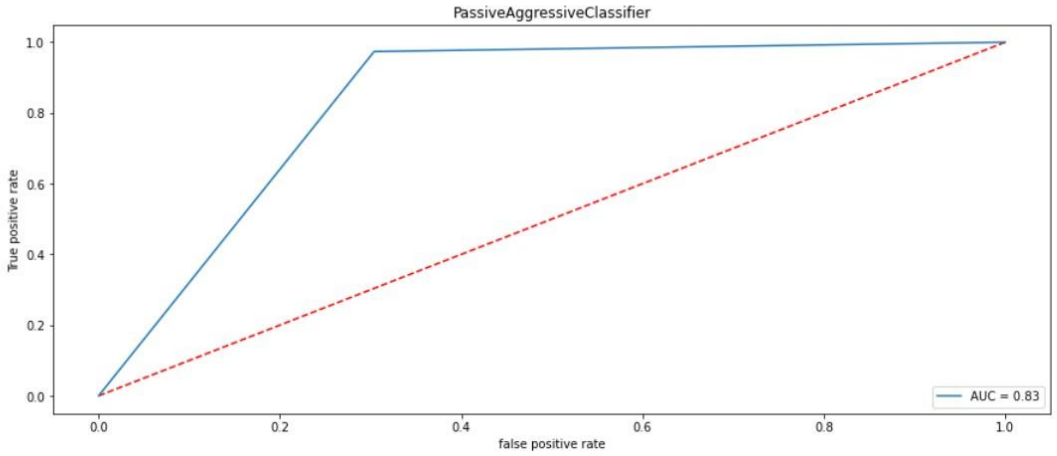
```
Accuracy Score: 0.9453960561497327
Log loss :  1.8859781103963864
Confusion Matrix: [[ 3389  1479]
 [ 1135 41869]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.70      0.72      4868
           1       0.97      0.97      0.97     43004

    accuracy                           0.95     47872
   macro avg       0.86      0.83      0.85     47872
weighted avg       0.94      0.95      0.94     47872
```

```
# Roc-Auc score
f,ax = plt.subplots(figsize = (15,6))
# Calculate fpr, tpr and thresholds
fpr, tpr, thresholds = roc_curve(y_test, PACpred)
ax.plot([0,1],[0,1],'r--')
ax.plot(fpr,tpr,label='AUC = %0.2f'% roc_auc_score(y_test, PACpred))
ax.legend(loc='lower right')
ax.set_xlabel('false positive rate')
ax.set_ylabel('True positive rate')
ax.set_title('PassiveAggressiveClassifier')
```

```
42]: Text(0.5, 1.0, 'PassiveAggressiveClassifier')
```



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.

Using Hyper-parameter tuning would have resulted in some more accuracy.

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.

THANK YOU