FLIP ROBO

Malignant Comments Classifier Prediction Project

Submitted By –

Himanshu Soni

ACKNOWLEDGMENT

I would like to express my very great appreciation to my SME Mr. Sajid Chowdhary for providing me this opportunity with this project.

Also, I would like to thank:

- Data Trained Team

Research papers that helped me in this project was as follows:

- https://www.kaggle.com/surekharamireddy/malignant-comment-classification

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

- 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**

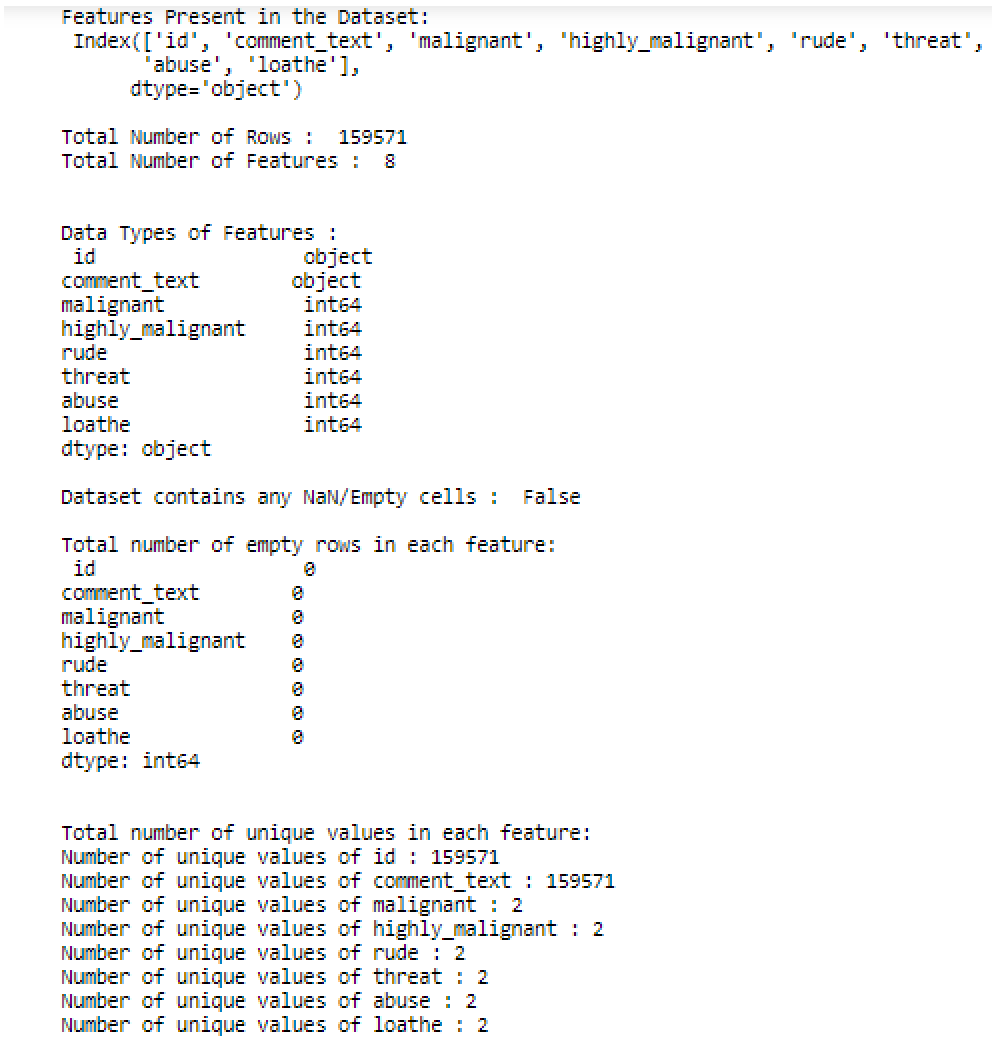
- The project was the first provided to me by FlipRobo as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively 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

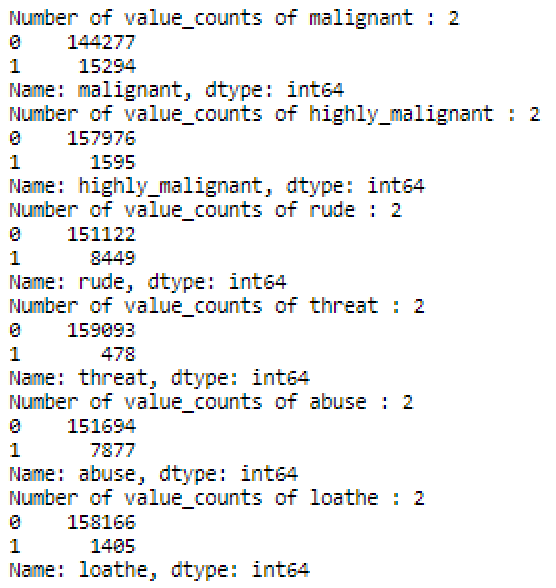
- **Data Sources and their formats**
- The data was provided by FlipRobo in CSV format. After loading the training dataset into Jupyter Notebook using Pandas and it can be seen that there are eight columns named as:
  “ id, comment_text, “malignant, highly_malignant, rude, threat, abuse, loathe**”.**

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.

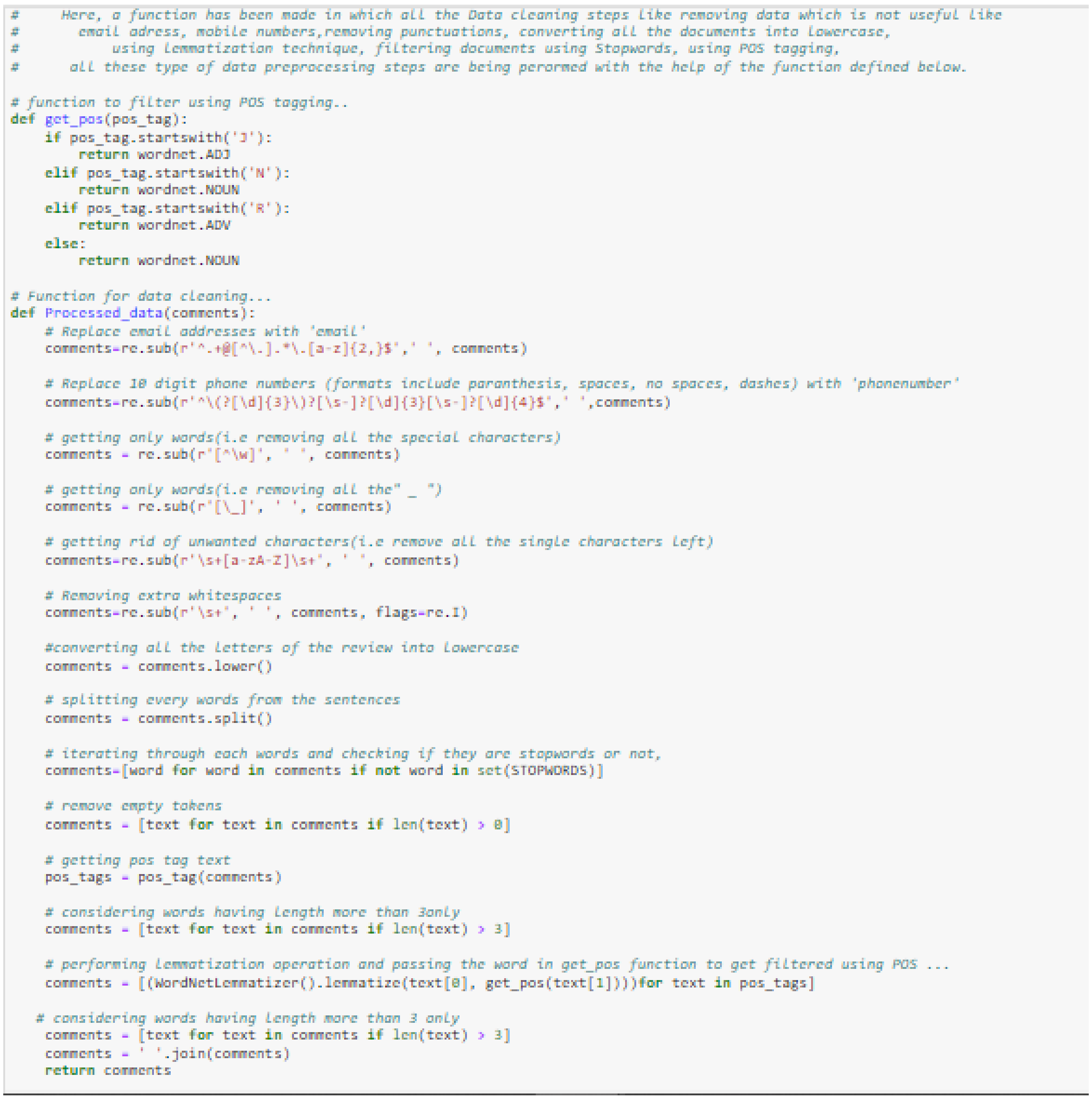


```
    Features Present in the Dataset:
     Index(['id', 'comment_text', 'malignant', 'highly_malignant', 'rude', 'threat',
           'abuse', 'loathe'],
          dtype='object')

    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

    Dataset contains any NaN/Empty cells :  False

    Total number of empty rows in each feature:
     id                  0
    comment_text        0
    malignant           0
    highly_malignant    0
    rude                0
    threat              0
    abuse               0
    loathe              0
    dtype: int64


    Total number of unique values in each feature:
    Number of unique values of id : 159571
    Number of unique values of comment_text : 159571
    Number of unique values of malignant : 2
    Number of unique values of highly_malignant : 2
    Number of unique values of rude : 2
    Number of unique values of threat : 2
    Number of unique values of abuse : 2
    Number of unique values of loathe : 2
```



```
Number of value_counts of malignant : 2
0    144277
1     15294
Name: malignant, dtype: int64
Number of value_counts of highly_malignant : 2
0    157976
1      1595
Name: highly_malignant, dtype: int64
Number of value_counts of rude : 2
0    151122
1      8449
Name: rude, dtype: int64
Number of value_counts of threat : 2
0    159093
1       478
Name: threat, dtype: int64
Number of value_counts of abuse : 2
0    151694
1      7877
Name: abuse, dtype: int64
Number of value_counts of loathe : 2
0    158166
1      1405
Name: loathe, dtype: int64
```

- ## Data Pre-processing Done

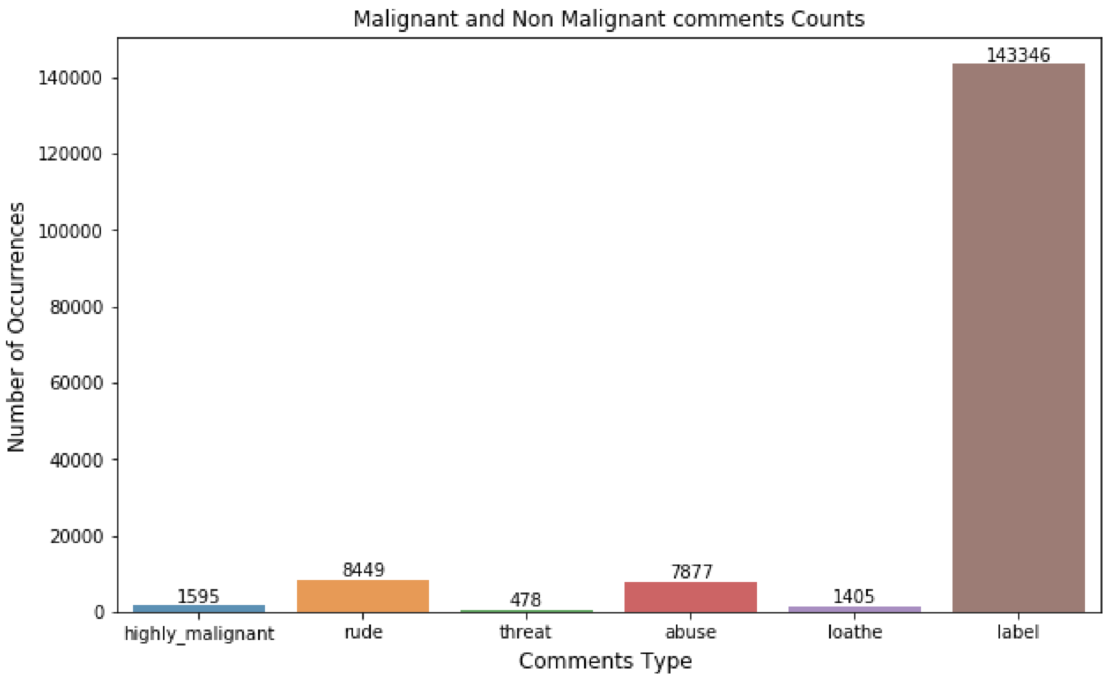


```python
#     Here, a function has been made in which all the Data cleaning steps like removing data which is not useful like
#       email adress, mobile numbers,removing punctuations, converting all the documents into lowercase,
#           using Lemmatization technique, filtering documents using Stopwords, using POS tagging,
#      all these type of data preprocessing steps are being peromed with the help 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
```

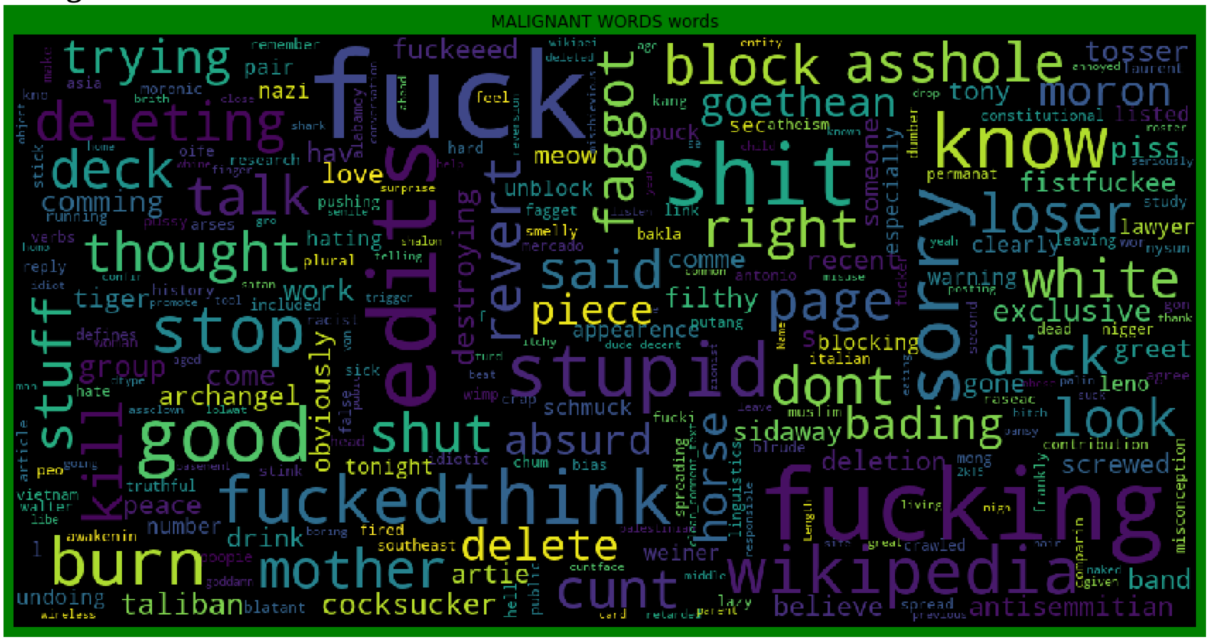
For Data pre-processing we did some data cleaning, where we used wordNet lemmatizer to clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizers and joined and return the filtered words.
Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms.

- ## Data Inputs- Logic- Output Relationships

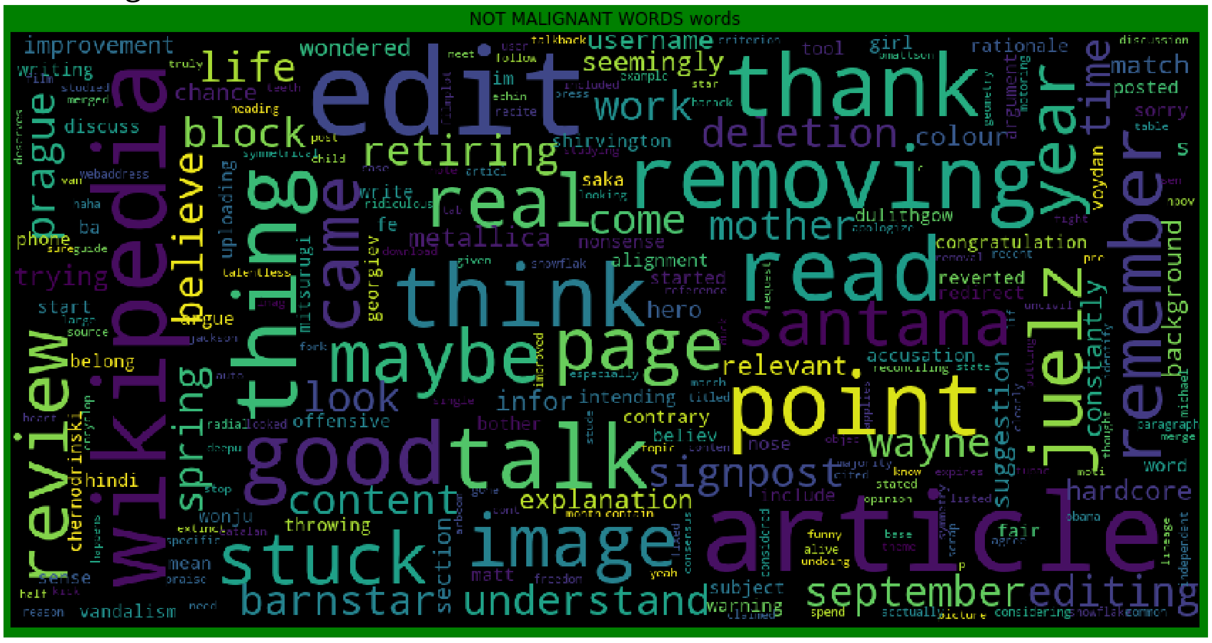
Comments categories counts:



Malignant Words:



Not Malignant Words:



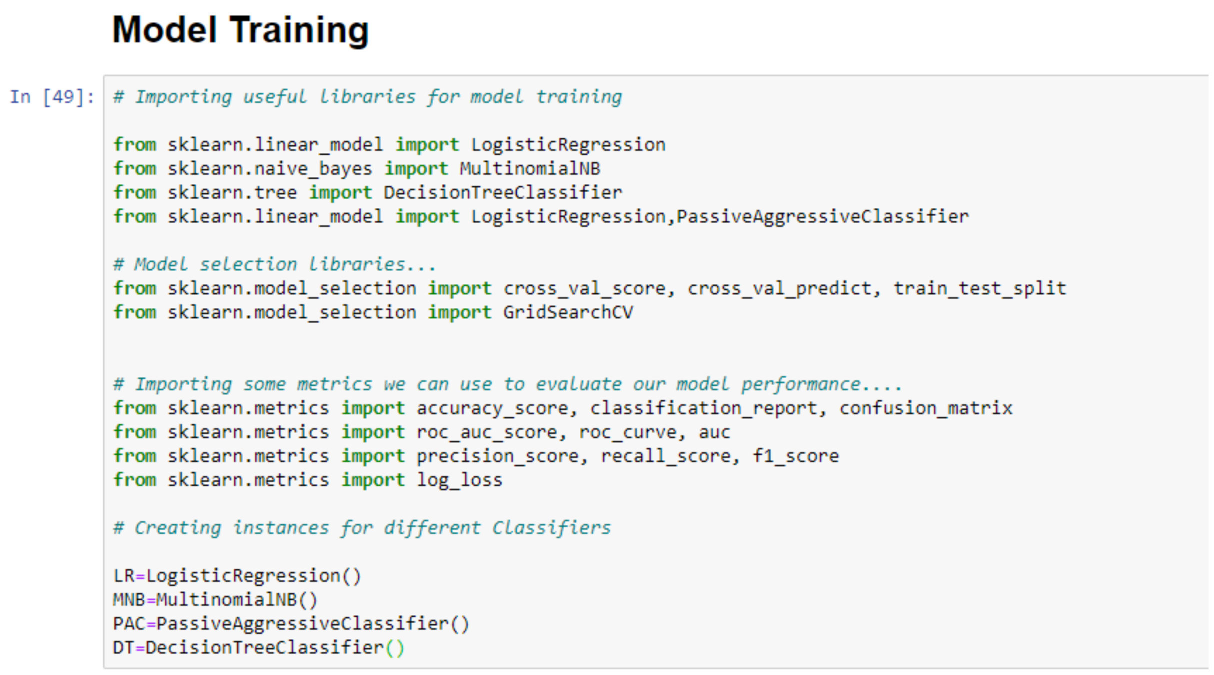
From the above we can see that most frequent words for both Malignant and Non Malignant category.

Hardware and Software Requirements and Tools Used

- Hardware: 12GB RAM, 64-bit, 7th gen i3 processor.
- Software: MS-Excel, Jupyter Notebook, python 3.7.

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 nltk
nltk.download('averaged_perceptron_tagger')

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

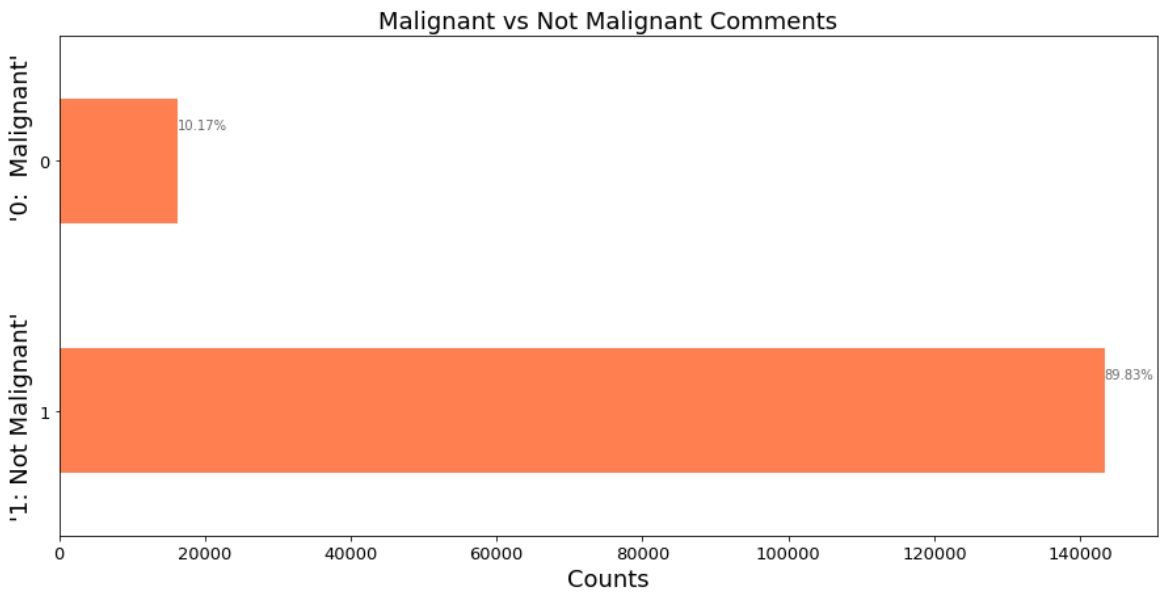


Model Training

```
In [49]: # 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()
```

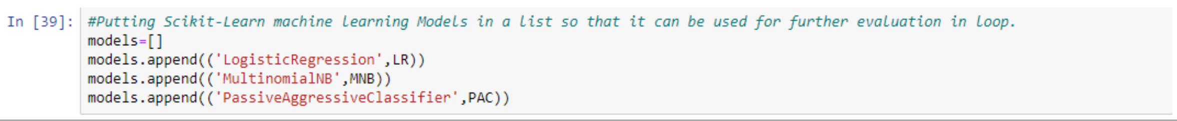
Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods).**

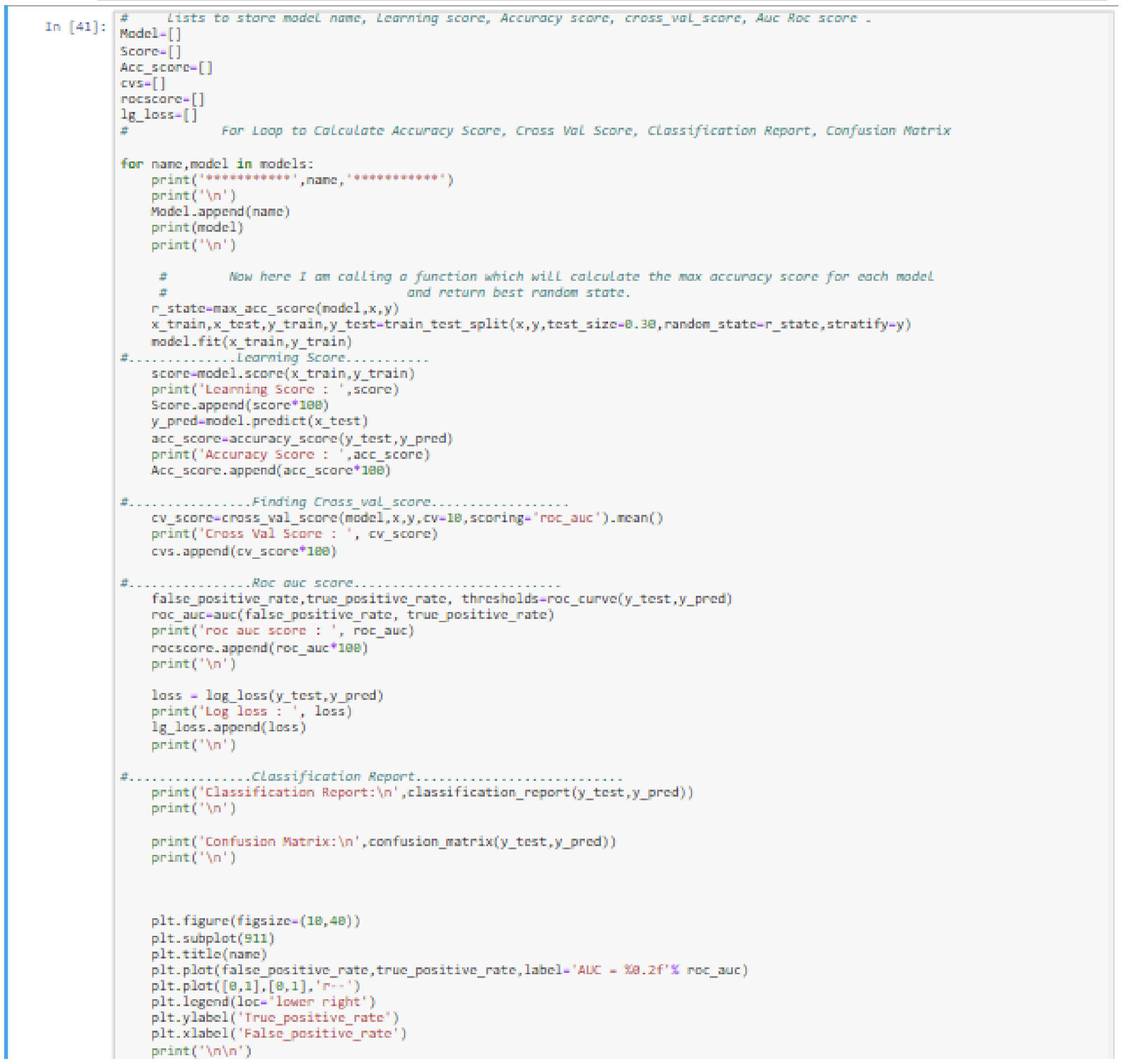


- **Testing of Identified Approaches (Algorithms)**
  - LR=LogisticRegression()
  - MNB=MultinomialNB()
  - PAC=PassiveAggressiveClassifier()

Run and Evaluated selected models

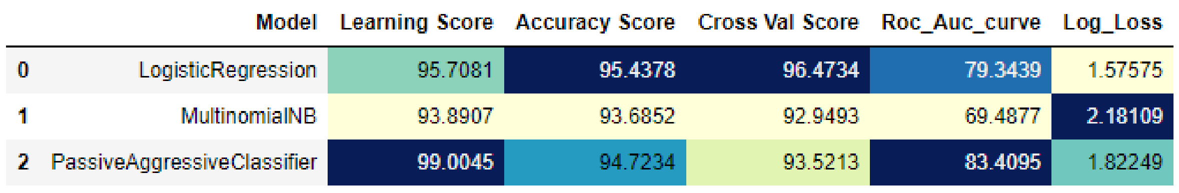


```
In [39]: #Putting Scikit-Learn machine Learning Models in a list so that it can be used for further evaluation in Loop.
         models=[]
         models.append(('LogisticRegression',LR))
         models.append(('MultinomialNB',MNB))
         models.append(('PassiveAggressiveClassifier',PAC))
```



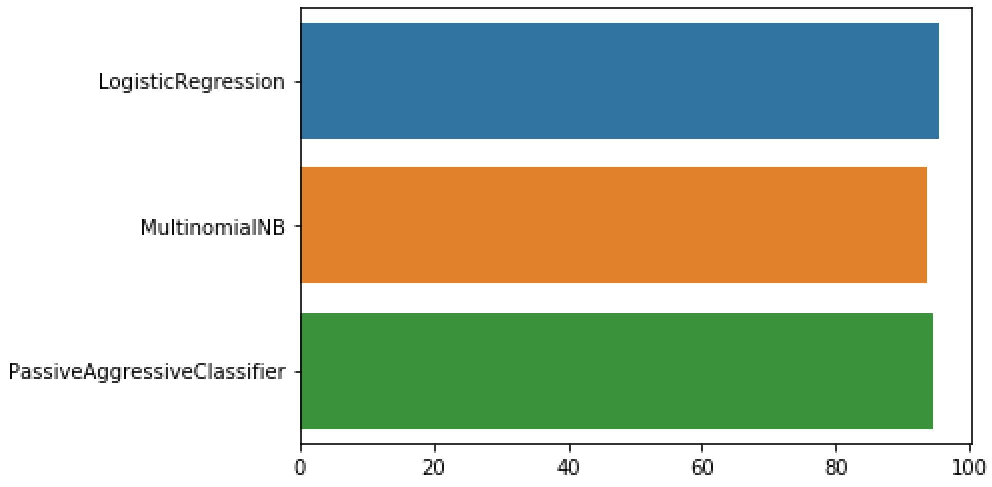
```
In [41]: #     Lists to store model name, Learning score, Accuracy score, cross_val_score, Auc Roc score .
         Model=[]
         Score=[]
         Acc_score=[]
         cvs=[]
         rocscore=[]
         lg_loss=[]
         #             For Loop to Calculate Accuracy Score, Cross Val Score, 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')



             plt.figure(figsize=(10,40))
             plt.subplot(911)
             plt.title(name)
             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')
             print('\n\n')
```

▪ Key Metrics for success in solving problem under consideration



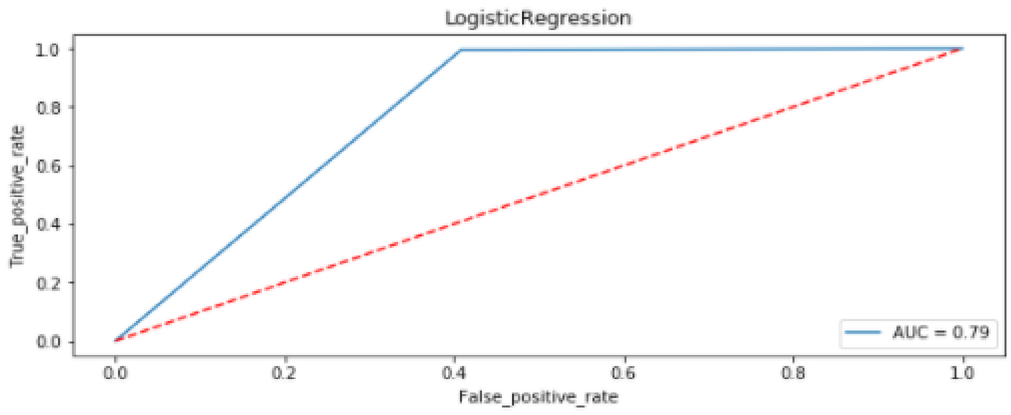
	Model	Learning Score	Accuracy Score	Cross Val Score	Roc_Auc_curve	Log_Loss
0	LogisticRegression	95.7081	95.4378	96.4734	79.3439	1.57575
1	MultinomialNB	93.8907	93.6852	92.9493	69.4877	2.18109
2	PassiveAggressiveClassifier	99.0045	94.7234	93.5213	83.4095	1.82249

Key Metrices used were the Accuracy Score, Cross validation 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%,



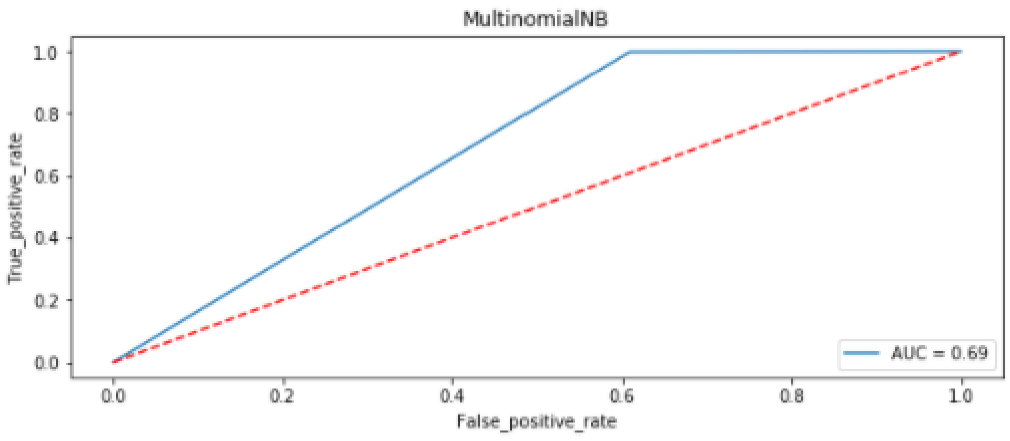
- **Visualizations:**
  - Logistic regression:

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


LogisticRegression(C=1.0, class_weight=None, dual=False, fit
_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_i
ter=100,
                   multi_class='warn', n_jobs=None, penalty=
'l2',
                   random_state=None, solver='warn', tol=0.0
001, verbose=0,
                   warm_start=False)


Max Accuracy Score corresponding to Random State  44 is: 0.9
543783422459893

Learning Score :  0.9570810839846373
Accuracy Score :  0.9543783422459893
Cross Val Score :  0.9647343896169192
roc auc score :  0.7934394247555041

Log loss :  1.5757494578411937

Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.59      0.73      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]
 [  195 42809]]
```



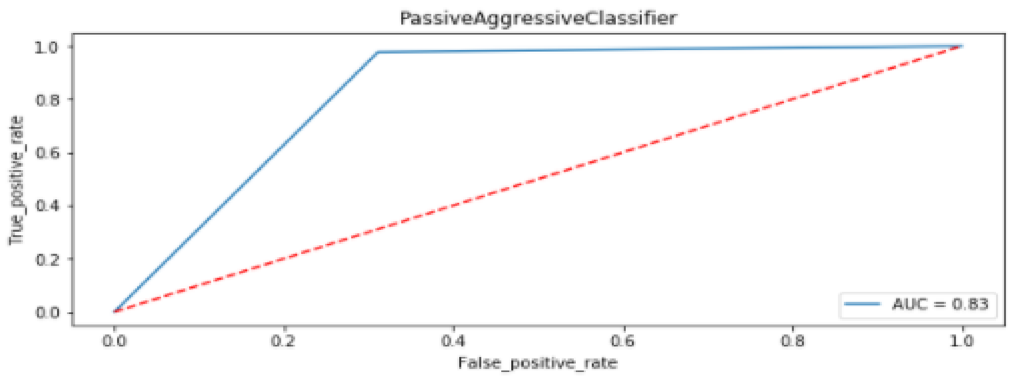
MultiNomial NB:

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


MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)


Max Accuracy Score corresponding to Random State  54 is: 0.936852439839
5722


Learning Score :  0.9389072417837223
Accuracy Score :  0.9368524398395722
Cross Val Score :  0.9294930890404292
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]]
```



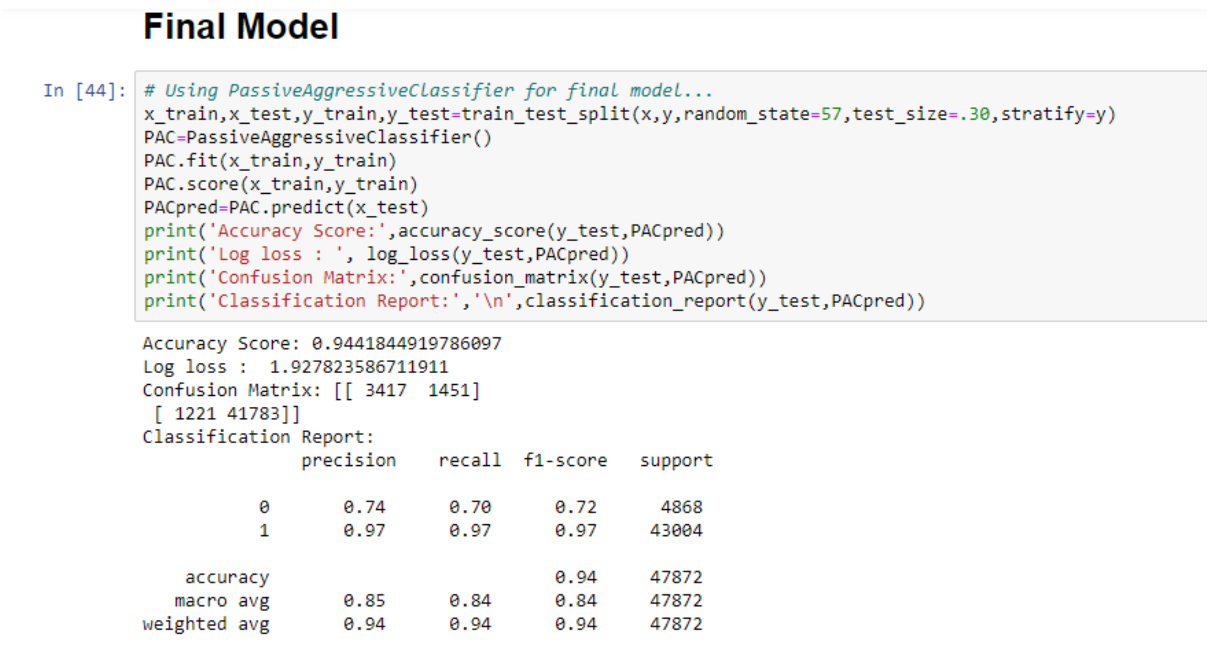
Passive Aggressive Classifier:

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


PassiveAggressiveClassifier(C=1.0, average=False, class_weight=None,
                            early_stopping=False, fit_intercept=True,
                            loss='hinge', max_iter=1000, n_iter_no_chan
ge=5,
                            n_jobs=None, random_state=None, shuffle=Tru
e,
                            tol=0.001, validation_fraction=0.1, verbose
=0,
                            warm_start=False)


Max Accuracy Score corresponding to Random State  98 is: 0.947192513368
9839


Learning Score :  0.990044673631814
Accuracy Score :  0.9472342914438503
Cross Val Score :  0.9352125615399485
roc auc score :  0.8340945830878357


Log loss :  1.822488046668037


Classification Report:
               precision    recall  f1-score   support

           0       0.77      0.69      0.73      4868
           1       0.97      0.98      0.97     43004

    accuracy                           0.95     47872
   macro avg       0.87      0.83      0.85     47872
weighted avg       0.95      0.95      0.95     47872


Confusion Matrix:
 [[ 3369  1499]
 [ 1027 41977]]
```



After all this process conclusion is that Passive Aggressive Classifier is giving accuracy of 94.72%. So, now I am making a final model using Passive Aggressive Classifier.

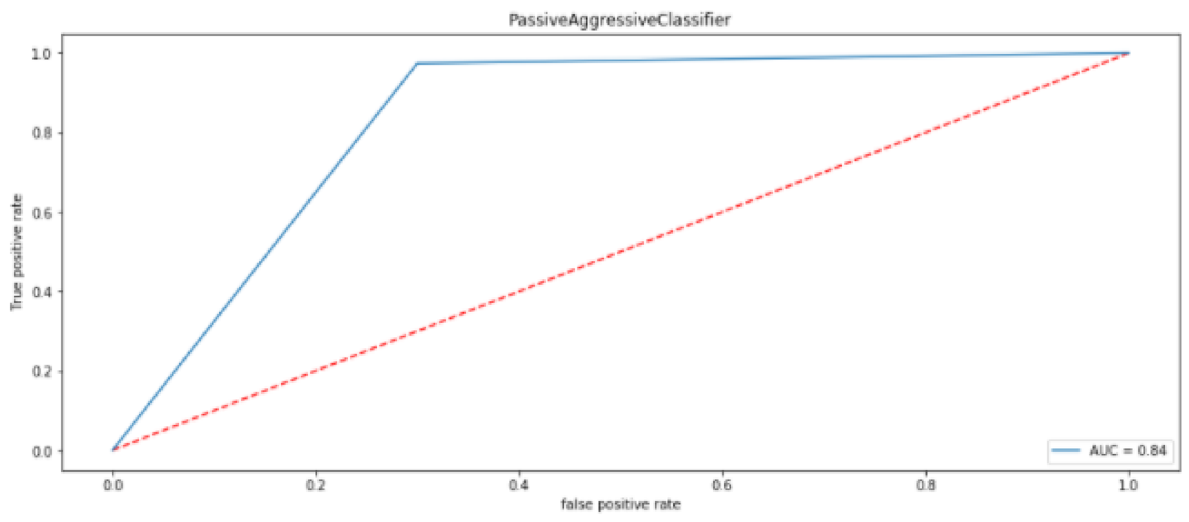
- **Final Model:**



Final Model

```
In [44]: # 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: 0.9441844919786097
Log loss :  1.927823586711911
Confusion Matrix: [[ 3417  1451]
 [ 1221 41783]]
Classification Report:
               precision    recall  f1-score   support

           0       0.74      0.70      0.72      4868
           1       0.97      0.97      0.97     43004

    accuracy                           0.94     47872
   macro avg       0.85      0.84      0.84     47872
weighted avg       0.94      0.94      0.94     47872
```



After all this process conclusion is that Passive Aggressive Classifier is giving accuracy of 94.72%, and with AUC_ROC score 0f 84%. So, now I am making a final model using Passive Aggressive Classifier.

Interpretation of the Results

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

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.