MALIGNANT COMMENTS CLASSIFIER PROJECT

Submitted by:
Naina Joshi

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.

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

- https://medium.com/@dobko_m/nlp-text-data-cleaning-and-preprocessing-ea3ffe0406c1
- https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1

Articles that helped me in this project were as follows:

TF-IDF Vectorizerscikit-learn. Deep understanding TfidfVectorizer by… | by Mukesh Chaudhary | Medium

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.
- 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

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.

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

Here we are dealing with one main text columns which held some importance of the data and others shows the multiple types of behaviour inferred from the text. I prefer to select on focus more on the words which has great value of importance in the context. Countvector is the NLP terms I am going to apply on text columns. This converts the important words proper vectors with some weights.

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.

```
<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
```

```
Features Present in the Dataset:
 Index(['id', 'comment_text', 'malignant', 'highly_malignant', 'rud
e', '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
```

LIBRARIES:

```
# Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
# import More librarues
import string
import re
# packages from gensim
from gensim import corpora
from gensim.parsing.preprocessing import STOPWORDS
from gensim.utils import simple_preprocess

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

# packages from nltk
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk import pos_tag
```

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

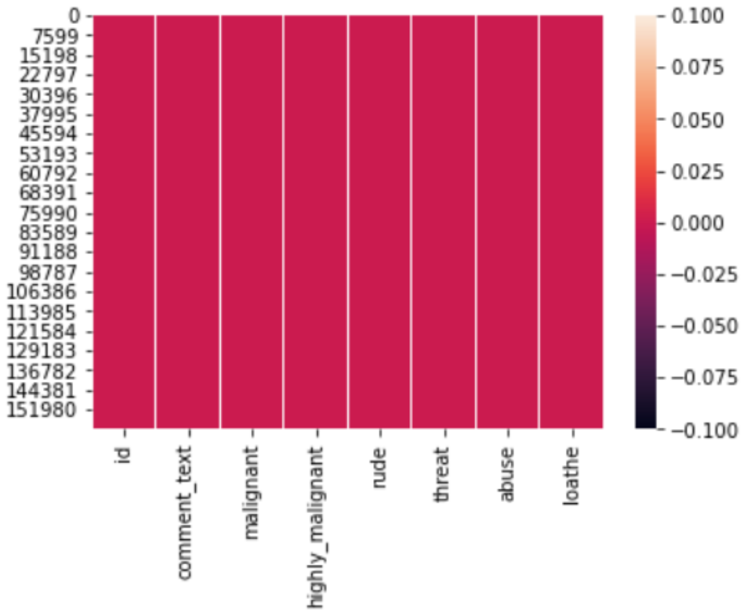
```python
# Importing libraries for model training

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier


from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
from sklearn.model_selection import GridSearchCV


# Importing evaluation metrics for 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
```

DATA PREPROCESSING DONE

After loading all the required libraries, we loaded the data into our jupyter notebook.

Feature Engineering has been used for cleaning of the data. We first did data cleaning. We first looked percentage of values missing in columns.

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



For Data pre-processing we did some data cleaning, where we used wordNetlemmatizerto 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.

```
#Creating a 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
```

```python
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
```

```python
# Cleaning  and storing the comments in a separate feature.
malignant_train["clean_comment_text"] = malignant_train["comment_text"].apply(lambda x: Processed_data(x))
```

```python
# Cleaning and storing the comments in a separate feature.
malignant_test["clean_comment_text"] = malignant_test["comment_text"].apply(lambda x: Processed_data(x))
```

```python
# Adding new feature clean_comment_length to store length of cleaned comments in clean_comment_text characters
malignant_train['clean_comment_length'] = malignant_train['clean_comment_text'].apply(lambda x: len(str(x)))
malignant_train.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	label	clean_comment_text	clean_comment_length
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264	0	explanation edits username hardcore metallica ...	129
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112	0	match background colour seemingly stuck thanks...	64
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233	0	trying edit constantly removing relevant infor...	112
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622	0	real suggestion improvement wondered section s...	315
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67	0	hero chance remember page	25

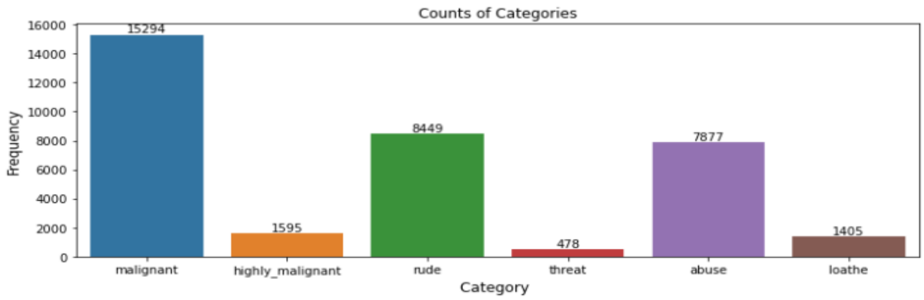
```python
malignant_test['clean_comment_length'] = malignant_test['clean_comment_text'].apply(lambda x: len(str(x)))
malignant_test.head()
```

	id	comment_text	comment_length	clean_comment_text	clean_comment_length
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367	bitch rule succesful whats hating mofuckas bit...	184
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50	title fine	10
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	54	source zawe ashton lapland	26
3	00017563c3f7919a	:If you have a look back at the source, the in...	205	look source information updated correct form g...	109
4	00017695ad8997eb	I don't anonymously edit articles at all.	41	anonymously edit article	24

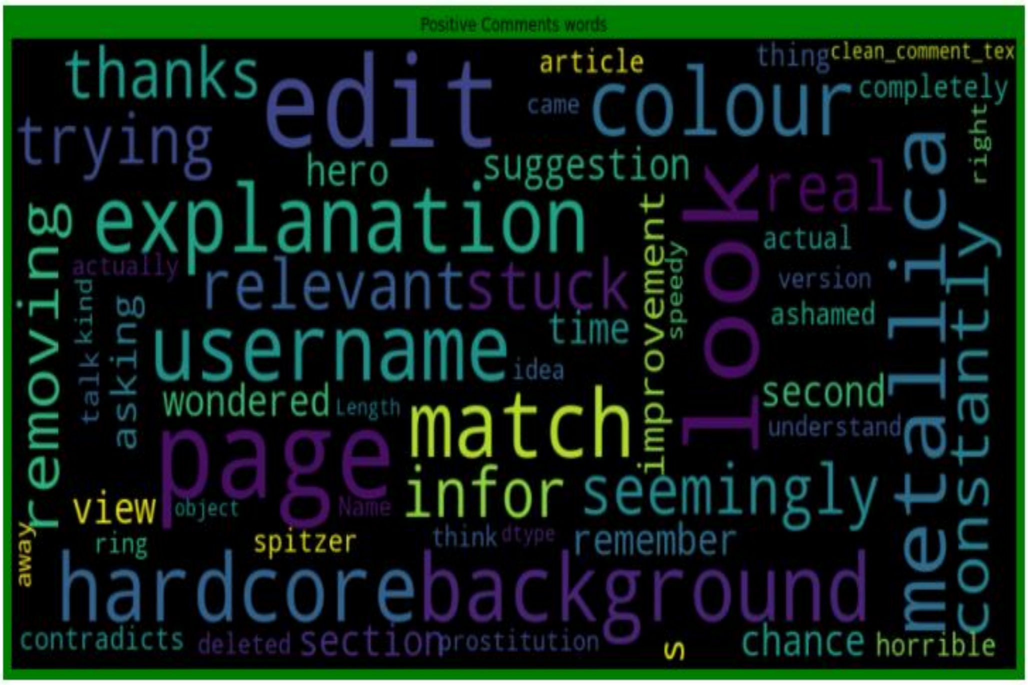
DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

EDA was performed by creating valuable insights using various visualization libraries.

```
# Let's plot the counts of each category

plt.figure(figsize=(12,4))
ax = sns.barplot(counts.index, counts.values)
plt.title("Counts of Categories")
plt.ylabel('Frequency', fontsize=12)
plt.xlabel('Category ', fontsize=12)
rects = ax.patches
labels = counts.values
for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 5, label, ha=
plt.show()
```



Malignant Words:



NoN Malignant Words:

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

HARDWARE:

Device specifications

Mi NoteBook Horizon Edition 14

Device name	LAPTOP-ED8G2MH8
Processor	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
Installed RAM	8.00 GB (7.83 GB usable)
Device ID	05E09149-DB9B-49DE-88A4-9C13612E78F7
Product ID	00327-35882-06869-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Rename this PC

Windows specifications

Edition	Windows 10 Home Single Language
Version	1909
Installed on	29-09-2020
OS build	18363.1440

SOFTWARE:

Jupyter Notebook (Anaconda 3) – Python

Microsoft Excel 2019

MODEL/S DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

The dataset is loaded and stored in a data frame. We need to perform some text processing to remove unwanted words and characters from our text. I used the nltk library and the string library. Then the data was analysed and visualized to extract insights about the comments. The sentence in the cleaned data, were broken down into vectors using Tokenizer from Keras and each word was converted into sequence of integers. Comments are variable in length, some are one-word replies while others are vastly elaborated thoughts. To overcome this issue, we use Padding. With the help of padding, we can make the shorter sentences as long as the others by filling the shortfall by zeros, and on the other hand, we can trim the longer ones to the same length as the short ones . I used the “pad_sequences” function from the “Keras” library and, I fixed the sentence length at 200 words and applied pre padding (i.e. for shorter sentences, 0’s will be added at the beginning of the sequence vector) A model was built using Keras and Tensorflow. For our classification task, I used both CNN and LSTM neural networks. The model consisted of Embedding layer, which is responsible for embedding. MaxPool layer used to focus on the important features. Bi-directional LSTM was used for one forward and one backward network. Last layer consisted of Sigmoid layer, which will predict probabilities for each kind of features in our dataset. The training dataset was split into training and validation set. 20% of the training data was kept aside for validation. The model was compiled with various optimizers, amongst which adam performed better and metrics like loss and AUC were used to evaluate the model. The dataset was then fit on training data and validated on validation dataset. It gave a quite good AUC of about 98.3% with 2 epochs. The loss was also decreasing significantly with increase in epoch, and finally the model was used to predict on the testing dataset.

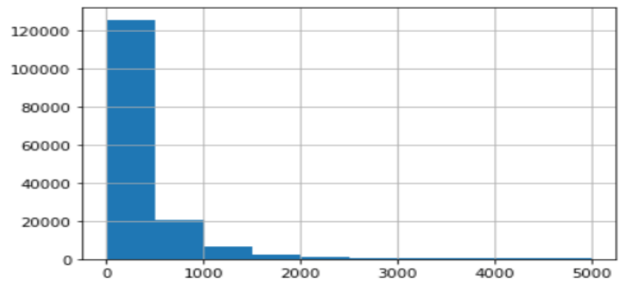
TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

```
# Creating instances for different Classifiers

LR=LogisticRegression()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
KNN=KNeighborsClassifier()
RFC=RandomForestClassifier()
GBC=GradientBoostingClassifier()
SV=SVC()
```

VISUALIZATIONS

```
# Let's plot the length in a histogram
lens.hist()
```

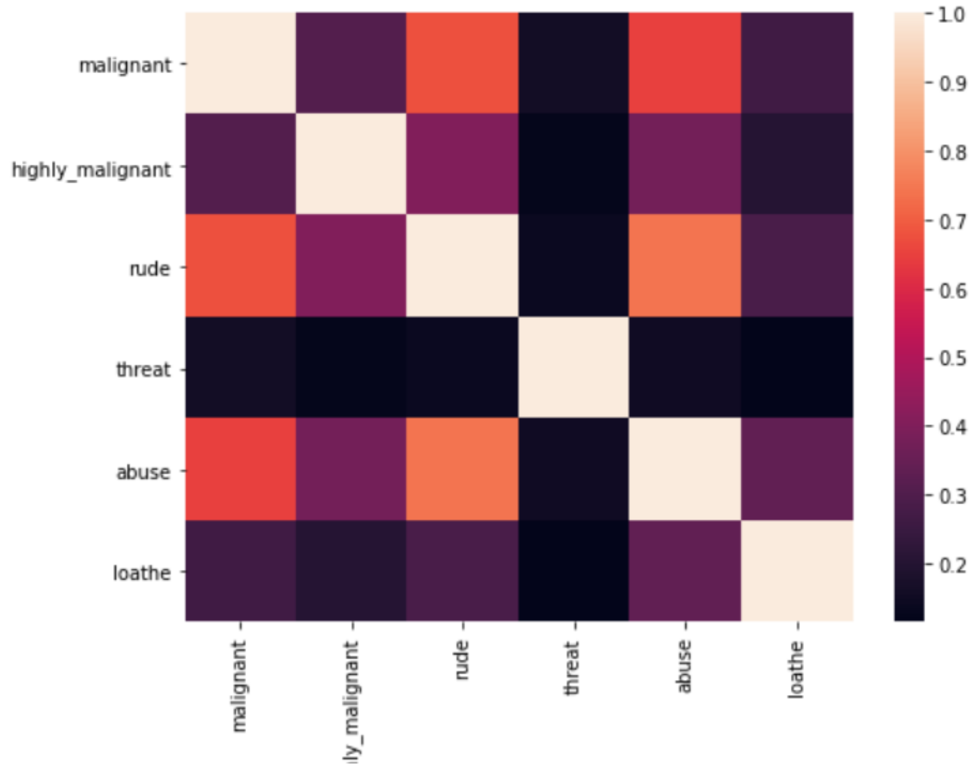
```
<AxesSubplot:>
```




```
# Let's check the correlation
malignant_train.corr()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128
abuse	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736
loathe	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000

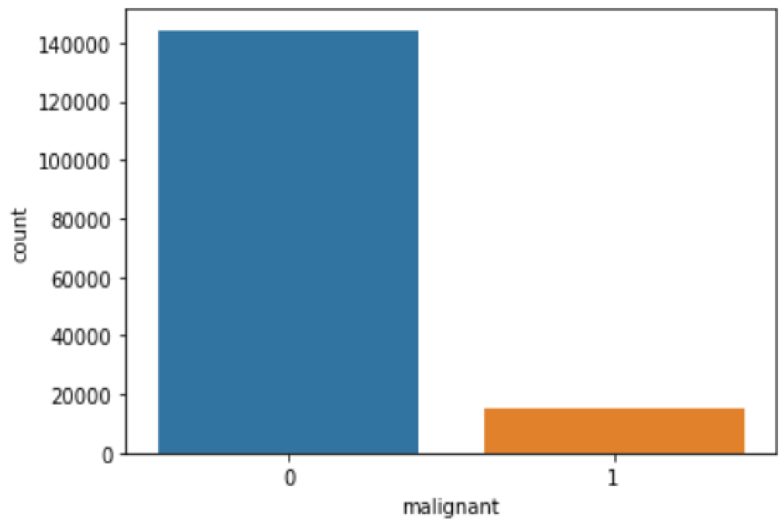
```
<matplotlib.axes._subplots.AxesSubplot at 0x2a606653490>
```

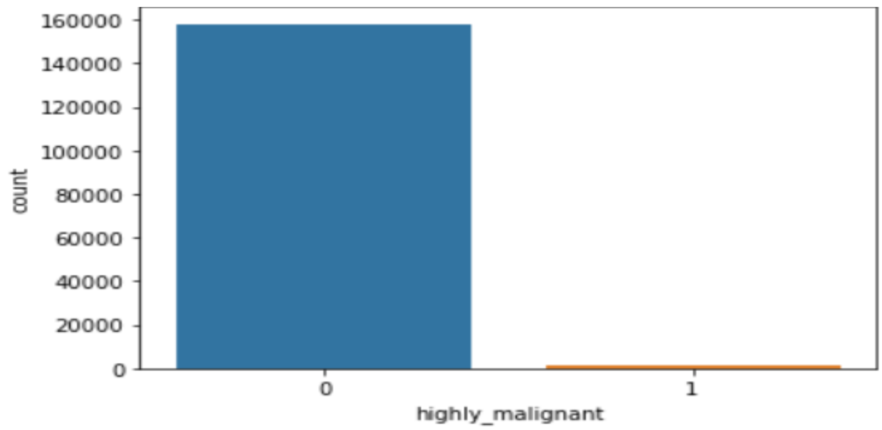


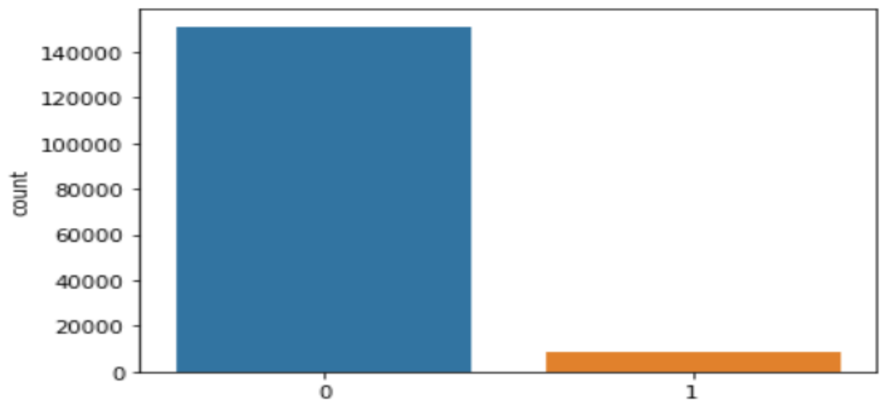
for i in features:

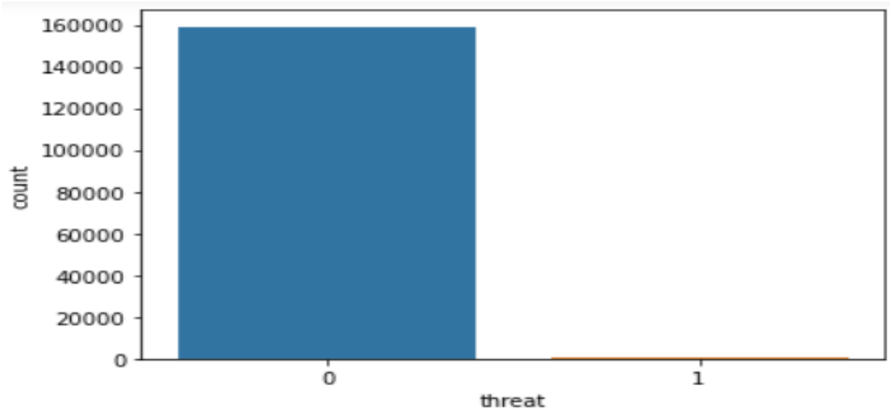
sns.countplot(malignant_train[i])

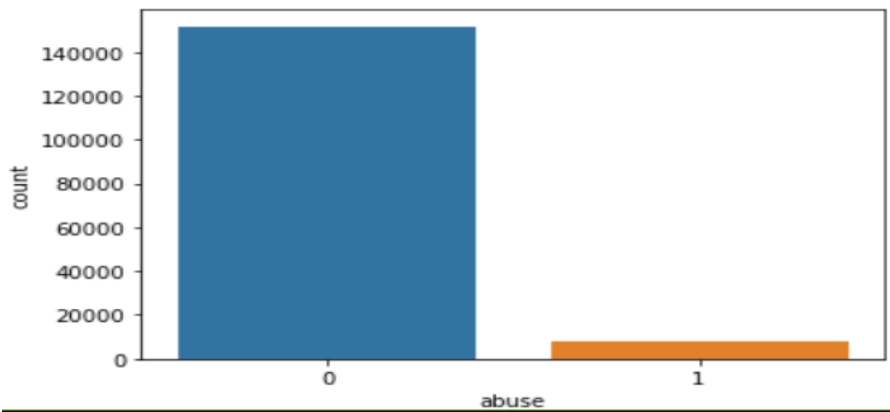
plt.show()

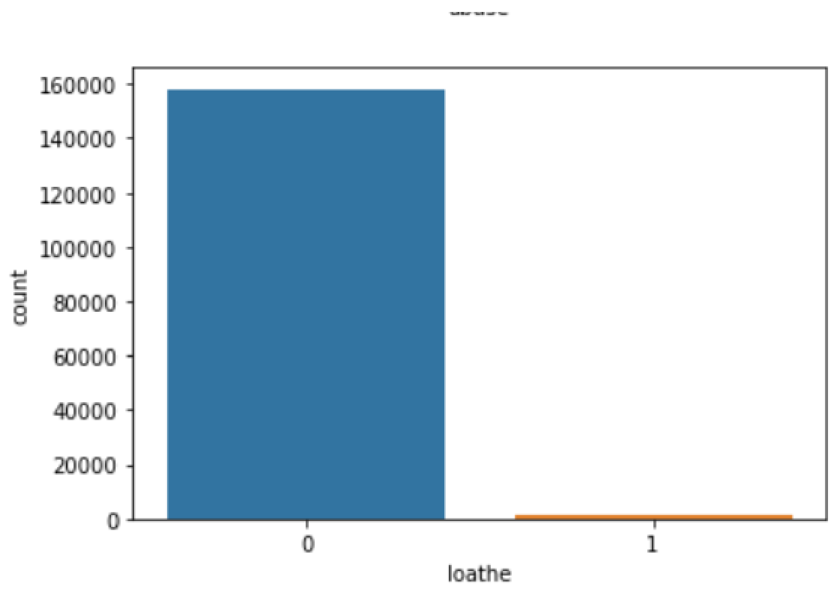



160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
highly_malignant

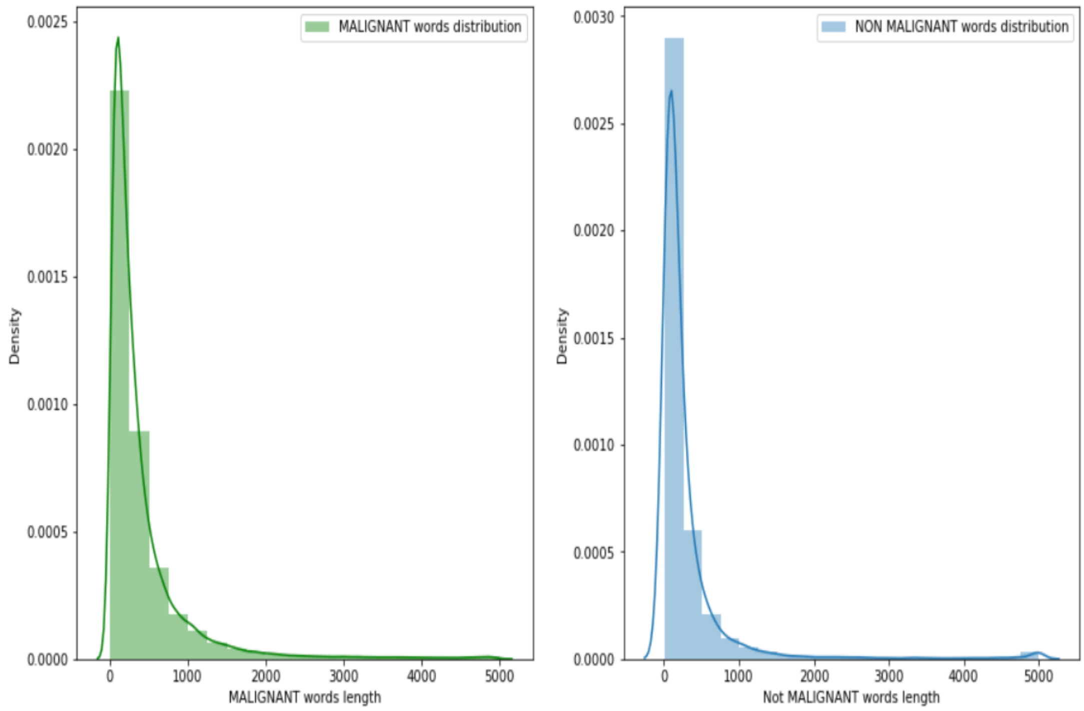

140000
120000
100000
80000
60000
40000
20000
0
count
0
1


160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
threat

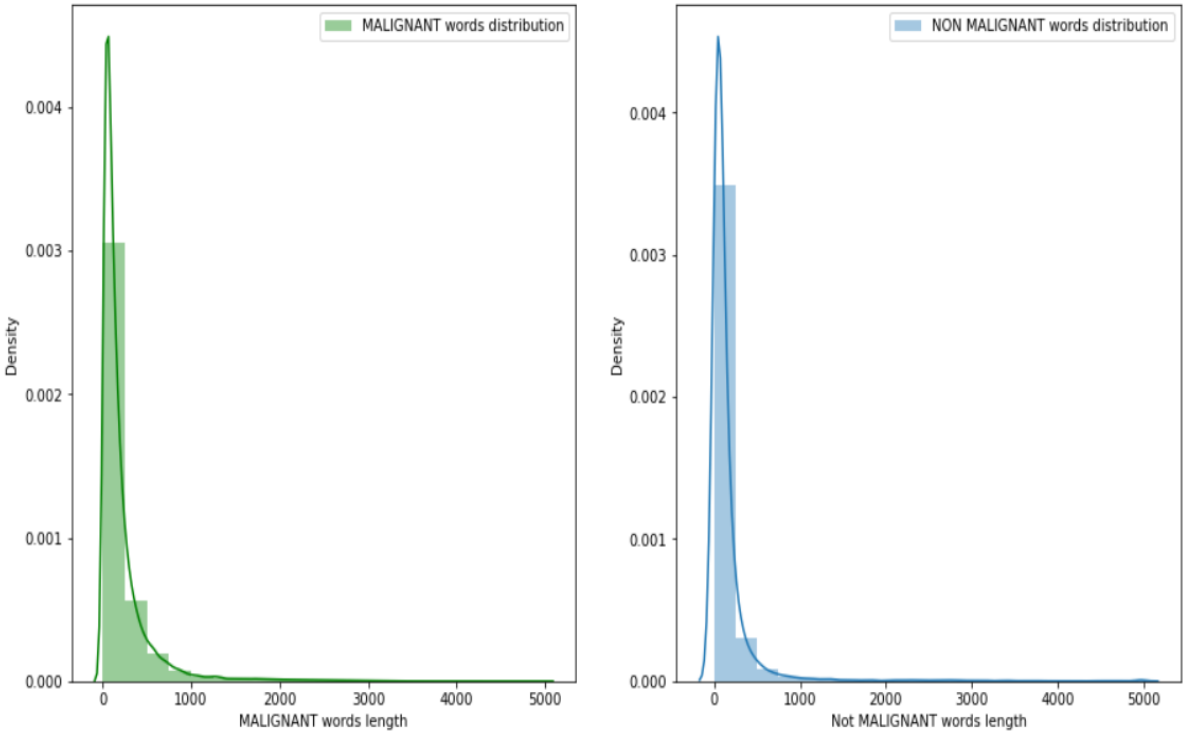

140000
120000
100000
80000
60000
40000
20000
0
count
0
1
abuse




```
# Comments length distribution BEFORE cleaning
f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(malignant_train[malignant_train['label']==0]['comment_length'],bins=20,ax=ax[0],label='MALIGNANT words distribution'

ax[0].set_xlabel('MALIGNANT words length')
ax[0].legend()

sns.distplot(malignant_train[malignant_train['label']==1]['comment_length'],bins=20,ax=ax[1],label='NON MALIGNANT words distribut
ax[1].set_xlabel('Not MALIGNANT words length')
ax[1].legend()

plt.show()
```



```python
# Comments length distribution after cleaning
f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(malignant_train[malignant_train['label']==0]['clean_comment_length'],bins=20,ax=ax[0],label='MALIGNANT words distri

ax[0].set_xlabel('MALIGNANT words length')
ax[0].legend()

sns.distplot(malignant_train[malignant_train['label']==1]['clean_comment_length'],bins=20,ax=ax[1],label='NON MALIGNANT words di
ax[1].set_xlabel('Not MALIGNANT words length')
ax[1].legend()

plt.show()
```



RUN AND EVALUATED SELECTED MODELS

```python
# Creating instances for different Classifiers

LR=LogisticRegression()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
KNN=KNeighborsClassifier()
RFC=RandomForestClassifier()
GBC=GradientBoostingClassifier()
SV=SVC()
```

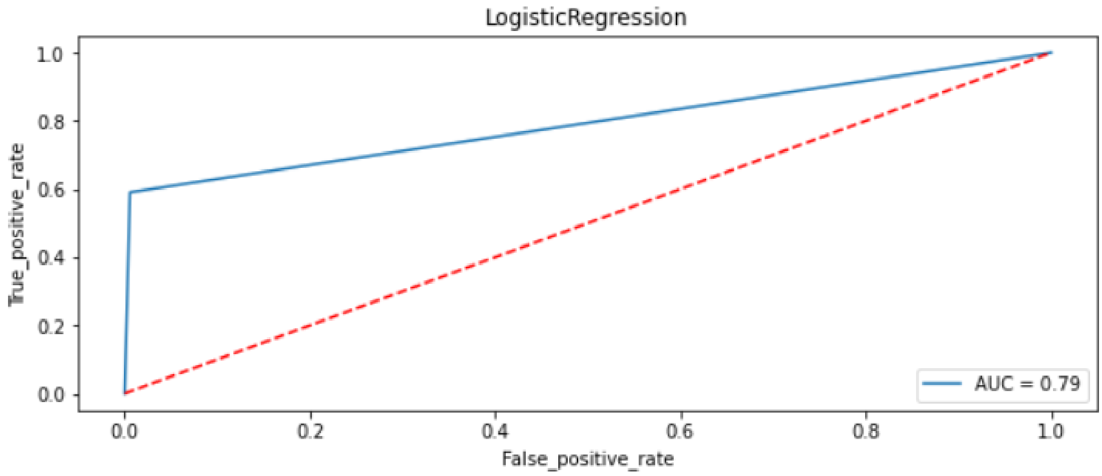
```python
# Creating a list model where all the models will be appended for fur
models=[]
models.append(('LogisticRegression',LR))
models.append(('MultinomialNB',MNB))
models.append(('DecisionTreeClassifier',DT))
models.append(('KNeighborsClassifier',KNN))
models.append(('RandomForestClassifier',RFC))
models.append(('GradientBoostingClassifier',GBC))
models.append(('SVC',SV))
```

```python
# Lists to store model name, Learning score, Accuracy score, cross_va

Model=[]
Score=[]
Acc_score=[]
cvs=[]
rocscore=[]
lg_loss=[]

# For Loop to Calculate Accuracy Score, Cross Val Score, Classificati

for name,model in models:
    print(name)
    Model.append(name)
    print(model)

    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30
    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)
```

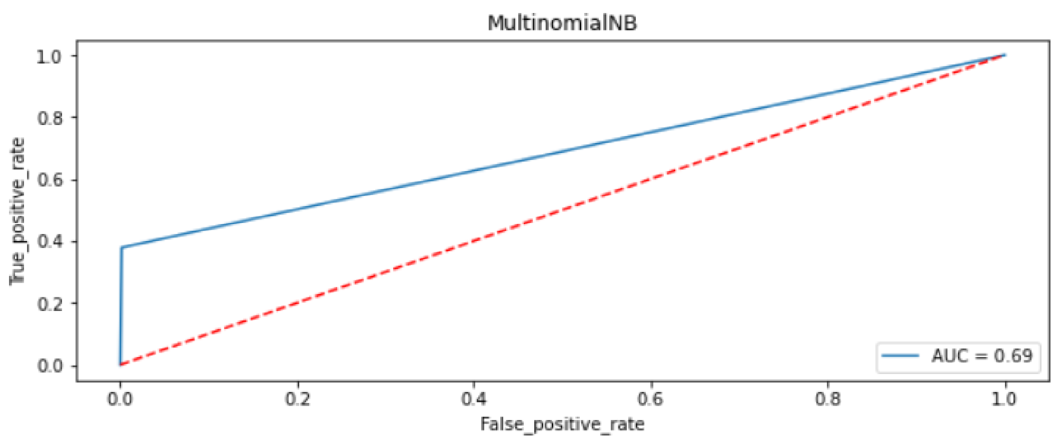
```python
# Cross_val_score
    cv_score=cross_val_score(model,x,y,cv=5,scoring='roc_auc').mean()
    print('Cross Val Score : ', cv_score)
    cvs.append(cv_score*100)

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

```
# Log Loss
    loss = log_loss(y_test,y_pred)
    print('Log loss : ', loss)
    lg_loss.append(loss)

# Classification Report
    print('Classification Report:\n',classification_report(y_test,y_p
    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 = %0.2
    plt.plot([0,1],[0,1],'r--')
    plt.legend(loc='lower right')
    plt.ylabel('True_positive_rate')
    plt.xlabel('False_positive_rate')
```

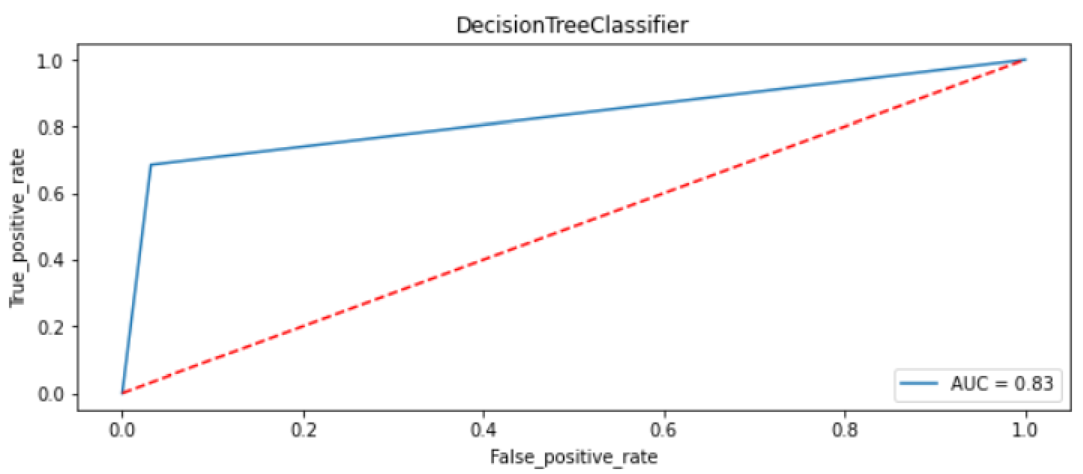
```
LogisticRegression
LogisticRegression()
Learning Score :  0.9577704366198444
Accuracy Score :  0.9531458890374331
Cross Val Score :  0.9640642647812614
roc auc score :  0.7923891030143992
Log loss :  1.618287837423593
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.92      0.59      0.72      4868

    accuracy                           0.95     47872
   macro avg       0.94      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872



Confusion Matrix:
 [[42754   250]
 [ 1993  2875]]
```



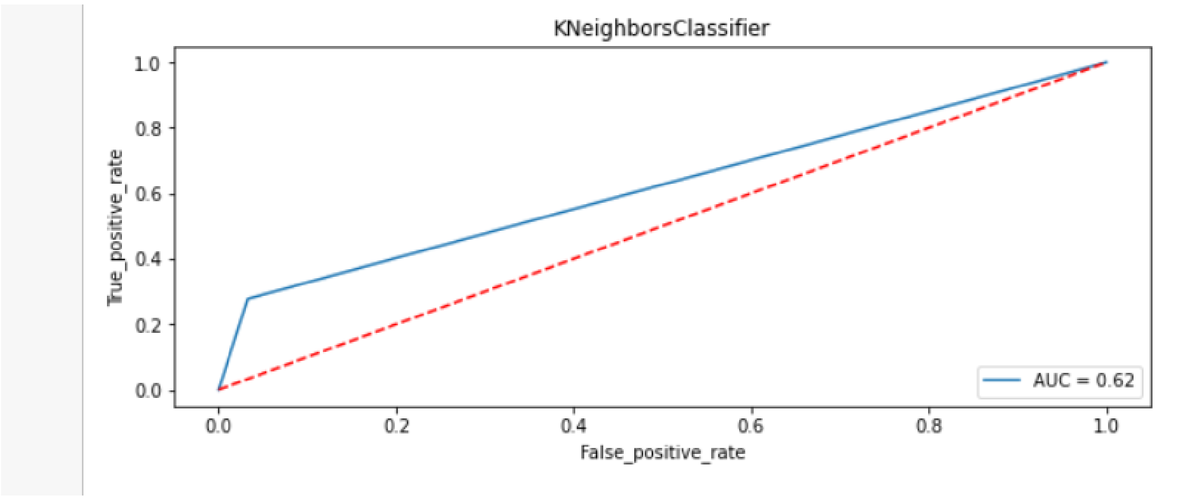

```
MultinomialNB
MultinomialNB()
Learning Score :  0.9397487891565726
Accuracy Score :  0.9354737633689839
Cross Val Score :  0.9264906705491673
roc auc score :  0.6884622511658735
Log loss :  2.22865831088146
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.97     43004
           1       0.97      0.38      0.54      4868

    accuracy                           0.94     47872
   macro avg       0.95      0.69      0.75     47872
weighted avg       0.94      0.94      0.92     47872



Confusion Matrix:
 [[42941    63]
 [ 3026  1842]]
```



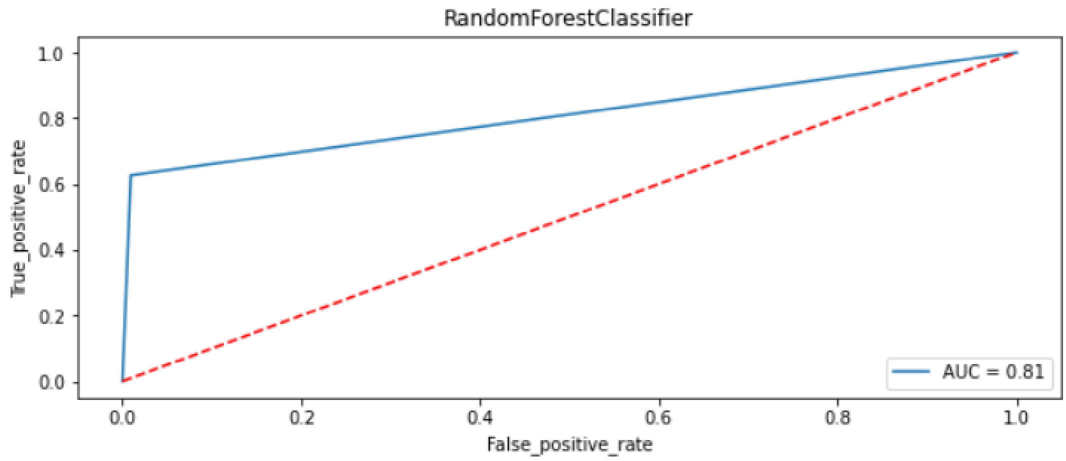
```
DecisionTreeClassifier
DecisionTreeClassifier()
Learning Score :  0.9982631894645431
Accuracy Score :  0.9397560160427807
Cross Val Score :  0.8339469045185884
roc auc score :  0.829112416746389
Log loss :  2.080776474118198
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.71      0.69      0.70      4868

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



Confusion Matrix:
 [[41628  1376]
 [ 1508  3360]]
```



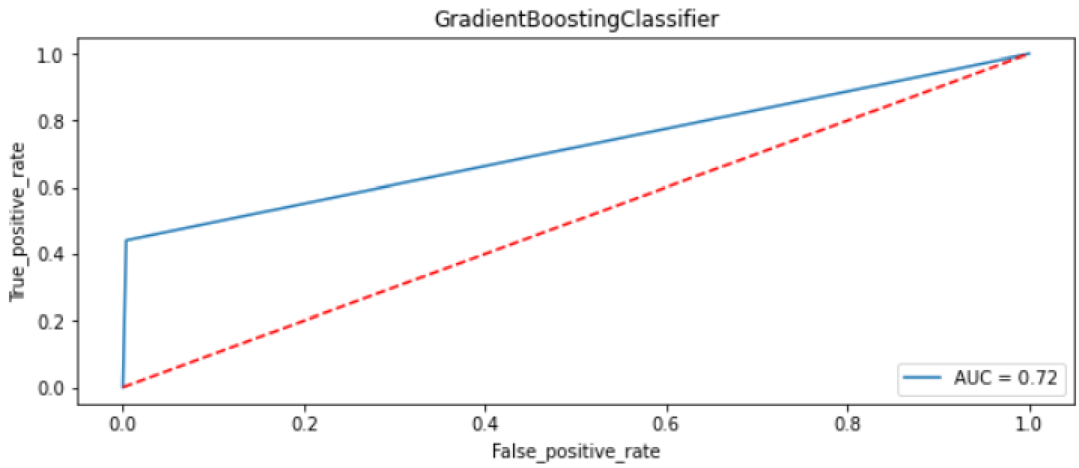
```
KNeighborsClassifier
KNeighborsClassifier()
Learning Score :  0.9235355732817662
Accuracy Score :  0.8970170454545454
Cross Val Score :  0.690548573731317
roc auc score :  0.6223346481995865
Log loss :  3.5569288406182857
Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.97      0.94     43004
           1       0.49      0.28      0.35      4868

    accuracy                           0.90     47872
   macro avg       0.71      0.62      0.65     47872
weighted avg       0.88      0.90      0.88     47872



Confusion Matrix:
 [[41591  1413]
 [ 3517  1351]]
```



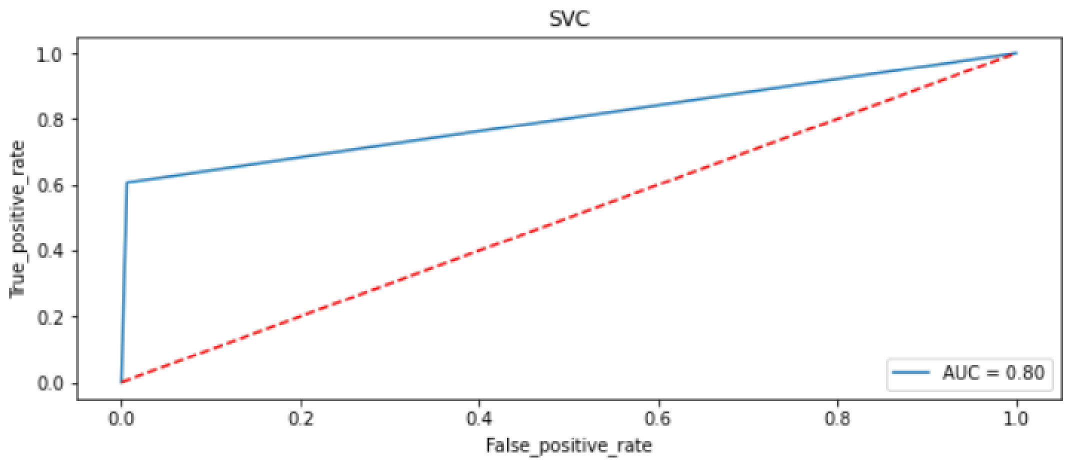
```
RandomForestClassifier
RandomForestClassifier()
Learning Score :  0.9982363315696648
Accuracy Score :  0.9535636697860963
Cross Val Score :  0.9549471952417437
roc auc score :  0.8081060488000315
Log loss :  1.6038607069865174
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.88      0.63      0.73      4868

    accuracy                           0.95     47872
   macro avg       0.92      0.81      0.85     47872
weighted avg       0.95      0.95      0.95     47872



Confusion Matrix:
 [[42604   400]
 [ 1823  3045]]
```



```
GradientBoostingClassifier
GradientBoostingClassifier()
Learning Score :  0.9418795154835764
Accuracy Score :  0.9394635695187166
Cross Val Score :  0.8899085169813693
roc auc score :  0.7157312839446935
Log loss :  2.0908566914537396
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     43004
           1       0.94      0.43      0.59      4868

    accuracy                           0.94     47872
   macro avg       0.94      0.72      0.78     47872
weighted avg       0.94      0.94      0.93     47872



Confusion Matrix:
 [[42857   147]
 [ 2751  2117]]
```



```
SVC
SVC()
Learning Score :  0.9812352841117646
Accuracy Score :  0.9545872326203209
Cross Val Score :  0.9627966041119127
roc auc score :  0.800295965283312
Log loss :  1.5685057440313812
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     43004
           1       0.92      0.61      0.73      4868

    accuracy                           0.95     47872
   macro avg       0.94      0.80      0.85     47872
weighted avg       0.95      0.95      0.95     47872



Confusion Matrix:
 [[42745   259]
 [ 1915  2953]]
```



```
# Displaying scores :
results=pd.DataFrame({'Model': Model,'Learning Score': Score,'Accuracy Score': Acc_score,'Cross Val Score':cvs,'Auc_score':rocsc
results
```

	Model	Learning Score	Accuracy Score	Cross Val Score	Auc_score	Log_Loss
0	LogisticRegression	95.777044	95.314589	96.406494	79.238910	1.618288
1	MultinomialNB	93.974879	93.547376	92.649067	68.846225	2.228658
2	DecisionTreeClassifier	99.826319	93.975602	83.394690	82.911242	2.080776
3	KNeighborsClassifier	92.353557	89.701705	69.054857	62.233465	3.556929
4	RandomForestClassifier	99.823633	95.356367	95.494720	80.810605	1.603861
5	GradientBoostingClassifier	94.187952	93.946357	88.990852	71.573128	2.090857
6	SVC	98.123528	95.458723	96.279660	80.029597	1.568506

Looking at all the Scores, I have selected Random Forest

Hyperparameter Tuning - Random Forest

FINAL MODEL

```
from sklearn.model_selection import RandomizedSearchCV
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=.30,stratify=y)
parameters={'bootstrap': [True, False],
 'max_depth': [10, 50, 100, None],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [100, 300, 500, 800, 1200]}


LG=LogisticRegression()

# Applying Randomized Search CV for hyperparameter tuning with scoring= "accuracy"
rand = RandomizedSearchCV(estimator = RFC, param_distributions = parameters,
                           n_iter = 10, cv = 3, verbose=2, random_state=42, n_jobs = -1,scoring='accuracy')
rand.fit(x_train,y_train)
rand.best_params_
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
{'n_estimators': 500,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_depth': 100,
 'bootstrap': False}
```

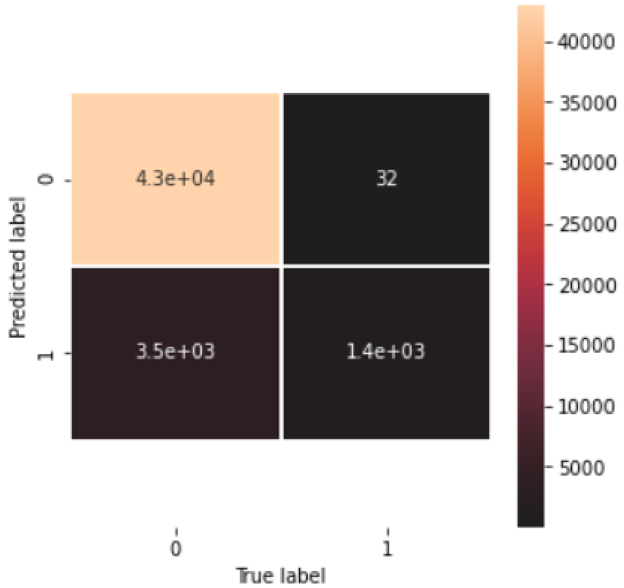
```
RFC=RandomForestClassifier(n_estimators= 500,
                           min_samples_split= 2,
                           min_samples_leaf=1,
                           max_depth= 100,
                           bootstrap= False)
```

```
RFC.fit(x_train,y_train)
RFC.score(x_train,y_train)
pred=RFC.predict(x_test)
print('Accuracy Score:',accuracy_score(y_test,pred))
print('Log loss : ', log_loss(y_test,pred))
print('Confusion Matrix:',confusion_matrix(y_test,pred))
print('Classification Report:','\n',classification_report(y_test,pred)
```

```
Accuracy Score: 0.926199030748663
Log loss :  2.548995709189865
Confusion Matrix: [[42972    32]
 [ 3501  1367]]
Classification Report:
               precision    recall  f1-score   support

           0       0.92      1.00      0.96     43004
           1       0.98      0.28      0.44      4868

    accuracy                           0.93     47872
   macro avg       0.95      0.64      0.70     47872
weighted avg       0.93      0.93      0.91     47872
```

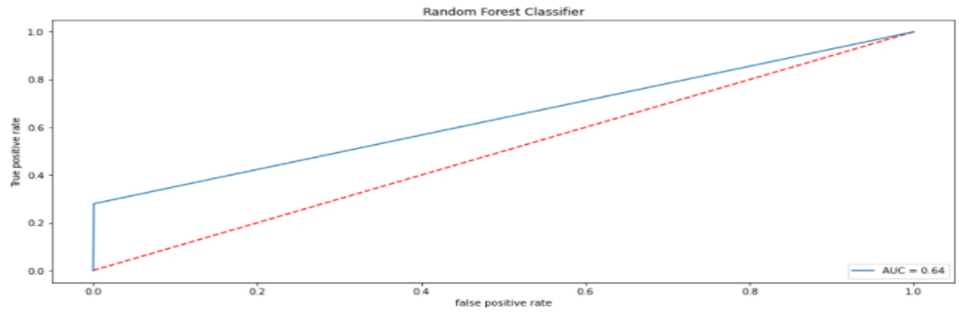
```
# Confusion matrix Visualization
fig, ax =plt.subplots(figsize=(5,5))
sns.heatmap(confusion_matrix(y_test, pred),annot=True,linewidths=1,center=0)
plt.xlabel("True label")
plt.ylabel("Predicted label")
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

```
(2.5, -0.5)
```



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

```
Text(0.5, 1.0, 'Random Forest Classifier')
```




```python
def Tf_idf_test(text):
    tfid = TfidfVectorizer(max_features=43194,smooth_idf=False)
    return tfid.fit_transform(text)
```

PREDICTION

```python
x_testing_data=Tf_idf_test(malignant_test['clean_comment_text'])
```

```python
x_testing_data.shape
```

```
(153164, 43194)
```

```python
Prediction=RFC.predict(x_testing_data)
malignant_test['Predicted values']=Prediction
malignant_test
```

	id	comment_text	comment_length	clean_comment_text	clean_comment_length	Predicted values
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367	bitch rule succesful whats hating mofuckas bit...	184	0
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50	title fine	10	0
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	54	source zawe ashton lapland	26	0
3	00017563c3f7919a	:If you have a look back at the source, the in...	205	look source information updated correct form g...	109	0
4	00017695ad8997eb	I don't anonymously edit articles at all.	41	anonymously edit article	24	0
...	...	...	...	...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...	60	totally agree stuff long crap	29	0
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...	198	throw field home plate faster throwing direct ...	85	0
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...	423	okinotorishima category change agree correct g...	212	0
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...	502	founding nation germany return similar israel ...	275	0
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...	141	stop bullshit welcome fool think kind explinat...	54	0

```
malignant_test['Predicted values'].value_counts()
```

```
0    153119
1        45
Name: Predicted values, dtype: int64
```

```
malignant_test[malignant_test['Predicted values']==1].head(20)
```

	id	comment_text	comment_length	clean_comment_text	clean_comment_length	Predicted values
805	0153f7856280e9ad	::::That entry made a lot of sense to me. As I...	372	entry sense replying time came desk noticed wa...	174	1
3914	06b13661ec5c3e6b	" \n\n ==Pelestinain Red Crescent Society and ...	990	pelestinain crescent society terrorism think p...	521	1
4568	07c5816cf1c0ffec	::Would you like to write up the Hegassen scro...	274	like write hegassen scroll entry publish soon ...	138	1
8358	0e02a435ccf5d6d1	== Franklin on Stalin == \n\n Possibly of inte...	382	franklin stalin possibly recently provided lin...	236	1
23370	26ffa274edf86566	==Ruud Lubbers entry== \n Hi Cary: What is hap...	485	ruud lubber entry cary happening page posted t...	219	1
25131	29e223fac14d609b	== Incorrectly titled articles by == \n\n You...	726	incorrectly titled article posted original wel...	324	1
25589	2ab006339fc4fdee	::::I use Google Earth. One has the option of ...	169	google earth option getting degree decimal for...	88	1
34462	394855c528d7c0d1	== Dude == \n\n We should form a rock band. Do...	147	dude form rock band prick pissed kissed opposite	48	1
34824	39ed57532158962a	" \n\n About your Third Opinion request: The	385	opinion request request dispute removed	239	1

```
malignant_test.to_csv('Malignant_Predict.csv')
```

```
# Pickle file.
import joblib
joblib.dump(RFC,'Malignant_Predict.pkl')
```

```
['Malignant_Predict.pkl']
```

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 (more than 13 hrs) 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.