Malignant Comments Classifier

Submitted by:

Raj Sharma

ACKNOWLEDGMENT

wish to express my sincere thanks to the following companies, without whom I would have not got opportunity to work on this project,

Data Trained Institute and Flip Robo Technology

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 must 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

Due the penetration of the internet in all domains of life which has led to increase of people’s participation actively and give remarks as an issue of communicating their concern/feedback/opinion in various online forums. Although most of the times these comments are helpful for the creator to

extemporize the substance that is being provided to people, but sometimes these may be abusive and create hatred-feeling among the people.

Thus, as these are openly available to the public which is being viewed from various sections of the society, people in different age groups, different communities and different socio-economic background, it becomes the prime responsibility of the content-creator (the host) to filter out these comments in order to stop the spread of negativity or hatred within people.

- ## Review of Literature

Detecting Toxic/Malignant comments has been a great challenge for the all the scholars in the field of research and development. This domain has drawn lot of interests not just because of the spread of hate but also people refraining people from participating in online forums which diversely affects for all the creators/content-providers to provide a relief to engage in a healthy public interaction which can be accessed by public without any hesitation.

- ## Motivation for the Problem Undertaken

In this project, I have tried to classify the comments/Texts which contains toxic or malignant keywords in the dataset, The objective behind working on this project is to build a strong filtering model which can be implemented on the current social media platforms where there is high chances of any offensive/negative interactions of users in the public comment section. This helps us to reduce the chances of spreading hate amongst people and stop cyberbully. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

Anyone can be a victim of online hate or cyberbully. The social media has become a dangerous place to dwell in. The use of abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity have significantly high negative impact on individual. We can use Machine Learning and NLP technologies to deal with such toxic comments. We were provided with two different datasets. One for training and another to test the efficiency of the model created using the training dataset. The training dataset provided here has a shape of 159571 rows and 8 columns. As it is a multiclass problem it has 6 dependent / target column. Here the target or the dependent variables named “malignant, highly_malignant, rude, threat, abuse, loathe” have two distinct values 0 and 1. Where 1 represents yes and 0 represents no for each class. As the target columns are giving binary outputs and all the independent variables has text so it is clear that it is a supervised machine learning problem where we can use the techniques of NLP and classification-based algorithms of Machine learning.

- ## 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 using df.head(), it can be seen that there are eight columns named as “ id, comment_text, “malignant, highly_malignant, rude, threat, abuse, loathe”

```
1 train = pd.read_csv('train.csv')
2
3 train
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

Similarly the test file can be load using pandas and the first five rows of the dataset can be seen using df.head().

```
test = pd.read_csv('test.csv')

test
```

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.
...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...

153164 rows × 2 columns

The datasets have no duplicated values or null values. Both the datset have no trace of any null or duplicated values.

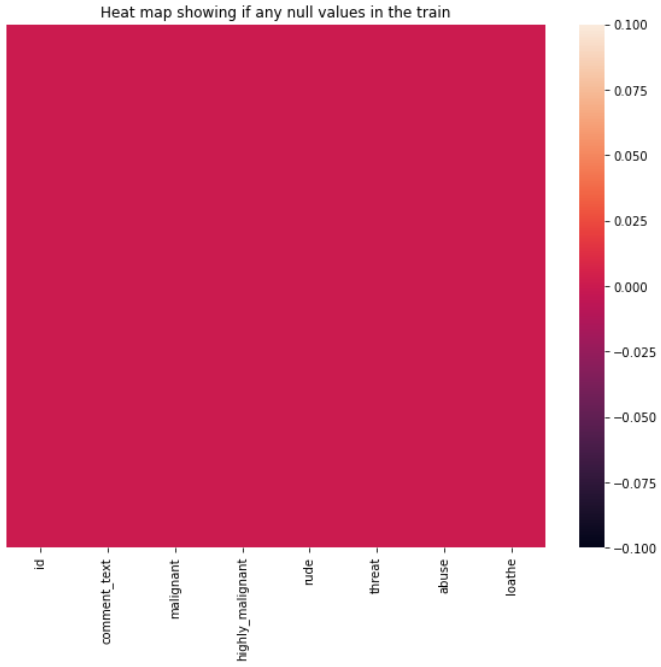
Checking for null values

```
print(train.isnull().sum())

plt.figure(figsize=(10,8),facecolor='white')
sns.heatmap(train.isnull(),yticklabels=False)
plt.title('Heat map showing if any null values in the train')
plt.show()
```

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

We have seen that there are no null values in the dataset, we are good to proceed with some feature engineering techniques.



- **Data Pre-processing Done**

After the dataset is loaded and the shape, null values and duplicated values were checked then the data- set is further treated where the unwanted column “id” is removed from the training dataset as we will work on the columns like “comment_text, “malignant, highly_malignant, rude, threat, abuse, loathe”

```
#creating a column called 'normal_comment'
col = ['malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe']

train['normal_comment'] = 1- train[col].max(axis=1)
```

```
train.drop(columns=['id'],inplace=True,axis=1)

train['Row Length']=train['comment_text'].str.len()
```

```
print(train['normal_comment'].value_counts())

plt.figure(figsize=(8,8),facecolor='white')
plt.pie(train['normal_comment'].value_counts(),labels=[1,0])
plt.title('Graph showing normal comments vs malignant')
plt.legend(['Normal Comments','Malignant Comments'])
plt.show()
```

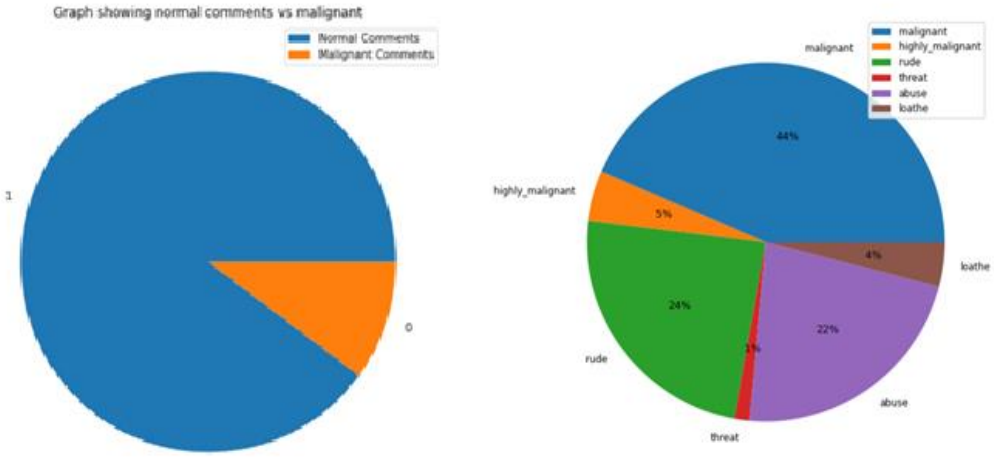
```
1    143346
0     16225
Name: normal_comment, dtype: int64
```

After removing the unwanted column, a new column named ‘normal’ was created in the training dataset which represents the statements not falling

under malignant, highly_malignant, rude, threat, abuse, loathe category or statements where values of malignant, highly_malignant, rude, threat, abuse, loathe are 0.

- **Data Inputs- Logic- Output Relationships**

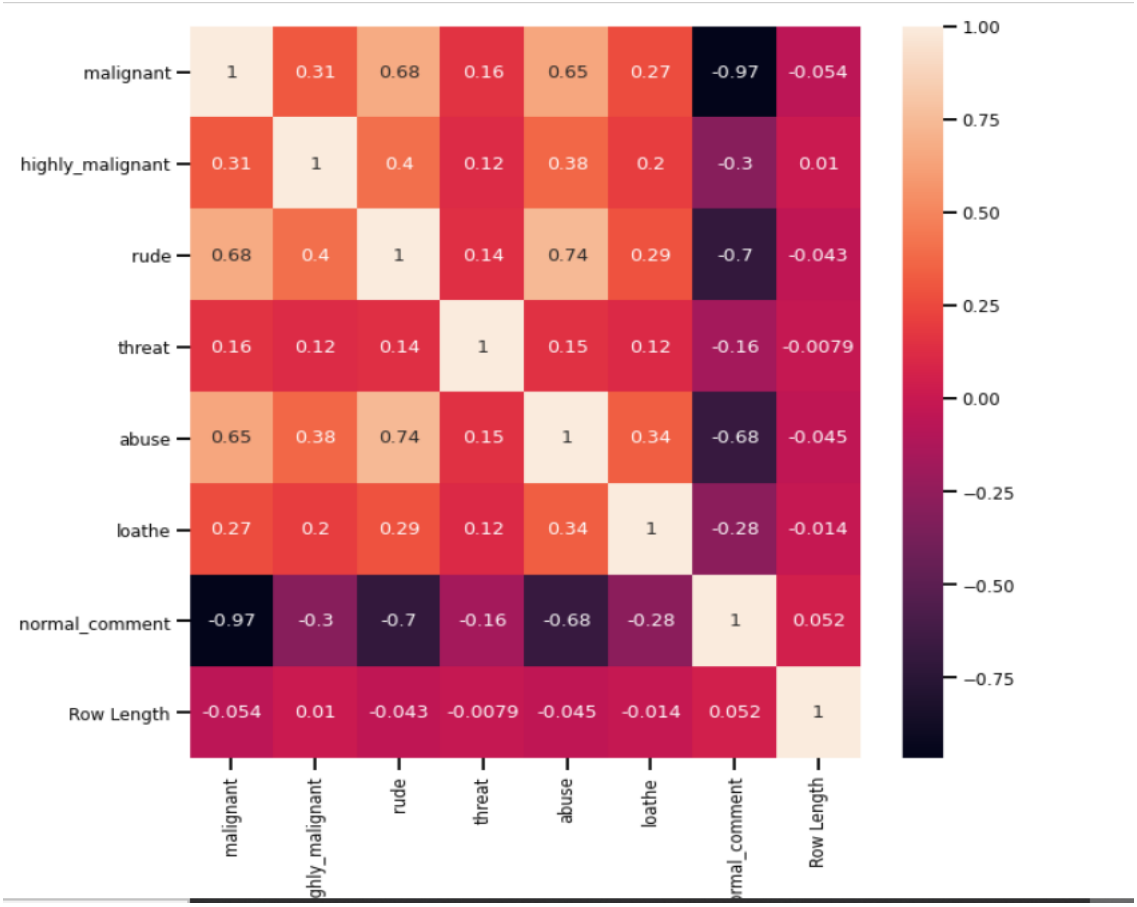
There are nearly 90% of the comments which are normal and 10 % of comments are classified malignant in training dataset.



There is no much skewness observed from the attributes, given as per below fig.

```
# checking the skewness for the features:
train.skew()
```

```
malignant            2.745854
highly_malignant     9.851722
rude                 3.992817
threat              18.189001
abuse                4.160540
loathe              10.515923
normal_comment      -2.635944
Row Length           4.121676
dtype: float64
```



- **Hardware and Software Requirements and Tools Used**

  Following are the recommended hardware requirement to build and run machine learning model.

  i. 7th generation (Intel Core i7 processor)

  ii. 8GB RAM / 16 GB RAM (recommended)

  We have used following software and tools for the machine learning model.

  ANACONDA

  Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split, GridSearchCV,cross_val_score
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,roc_auc_score,roc_curve,plot_roc_curve

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

The figure shows the important libraries I have imported to execute the project. I have used built in Data science libraries like pandas, NumPy, Visualization libraries like matplotlib and seaborn. Jupyter Notebook, a shareable notebook that combines live code, visualizations, and text.

Machine learning libraries like scikit-learn for data pre-processing, model selection, model evaluation, SciPy for standardizing& normalizing the data,

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

After getting a cleaned data TF-IDF vectorizer will be used. It'll help to transform the text data to feature vector which can be used as input in our modelling. The TFIDF stands for Term Frequency Inverse Document Frequency. It is a common algorithm to transform text into vectors or numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in.

```
#  Convert text into vectors using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(train['comment_text'])
x = features
```

```
#dropping the id column from test dataset and converting the text into vectors

test.drop('id',axis=1,inplace=True)

test_data =tf_vec.fit_transform(test['comment_text'])
test_data

```

- **Testing of Identified Approaches (Algorithms)**
  Now the data obtained is clean and is having least multicollinearity, independent and balanced data. 'train_test_split' function in model selection is used for splitting data arrays into two subsets: for training data and for testing data. We train the model using the training set and then apply the model to the test set.

```
#dropping the id column from test dataset and converting the text into vectors

test.drop('id',axis=1,inplace=True)

test_data =tf_vec.fit_transform(test['comment_text'])
test_data

```

```
<153164x10000 sparse matrix of type '<class 'numpy.float64'>'
        with 2940344 stored elements in Compressed Sparse Row format>
```

```
y=train['bad']
X_train,X_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.30)
```

  As it is a multi-label classification problem, we will use Classification algorithms from sklearn like;
  Logistic Regression()
  Decision Tree Classifier ()
  Multinomial NB()
  Random Forest Classifier()
  SVM ()
  During modelling various metrices like f1_score, confusion matrix, accuracy score, classification report, roc curve, roc auc score, mean squared error, precision score, recall score, log loss will be used to determine the performance of the model. At each step at the end of a model a data frame will be generated which will show the performance of the model per class.
- **Run and Evaluate selected models**

Logistic Regression

```
Log_reg = LogisticRegression()
Log_reg.fit(X_train, y_train)
y_pred = Log_reg.predict(X_test)

print("\n The accuracy score of Model :", accuracy_score(y_test,y_pred))
print("\n The ROC AUC score of Model :", roc_auc_score(y_test,y_pred))


CV=cross_val_score(Log_reg,X_train, y_train,cv=5)
print("\n The CV score of Model :", CV.mean())

print("\n The confusion Matrix :\n ", confusion_matrix(y_test, y_pred))
print("\n The classification report:\n ",classification_report(y_test, y_pred))


print("*******************************************")
```

```
 The accuracy score of Model : 0.955276570855615

 The ROC AUC score of Model : 0.802294632116666

 The CV score of Model : 0.9541804391779747

 The confusion Matrix :
  [[42730   220]
 [ 1921  3001]]

 The classification report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.93      0.61      0.74      4922

    accuracy                           0.96     47872
   macro avg       0.94      0.80      0.86     47872
weighted avg       0.95      0.96      0.95     47872

*******************************************
```

Random forest

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_train = rf.predict(X_train)
y_pred = rf.predict(X_test)
print("*****************RESULTS*****************")
print("\n The accuracy score of train Model :", accuracy_score(y_train,y_pred_train))

print("\n The accuracy score test of Model :", accuracy_score(y_test,y_pred))
print("\n The ROC AUC score of Model :", roc_auc_score(y_test,y_pred))

CV=cross_val_score(rf,X_train, y_train,cv=5)
print("\n The CV score of Model :", CV.mean())

print("\n The confusion Matrix :\n ", confusion_matrix(y_test, y_pred))
print("\n The classification report:\n ",classification_report(y_test, y_pred))


print("*******************************************")
```

```
*****************RESULTS*****************

 The accuracy score of train Model : 0.9988719684151156

 The accuracy score test of Model : 0.9564463569518716

 The ROC AUC score of Model : 0.8365853389713052

 The CV score of Model : 0.9562216373458815

 The confusion Matrix :
  [[42412   538]
 [ 1547  3375]]

 The classification report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.69      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.84      0.87     47872
weighted avg       0.95      0.96      0.95     47872

*******************************************
```

KNN

```
knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

y_pred_train = knn_clf.predict(X_train)
y_pred = knn_clf.predict(X_test)
print("*****************RESULTS*****************")
print("\n The accuracy score of train Model :", accuracy_score(y_train,y_pred_train))

print("\n The accuracy score test of Model :", accuracy_score(y_test,y_pred))
print("\n The ROC AUC score of Model :", roc_auc_score(y_test,y_pred))

CV=cross_val_score(knn_clf,X_train, y_train,cv=5)
print("\n The CV score of Model :", CV.mean())

print("\n The confusion Matrix :\n ", confusion_matrix(y_test, y_pred))
print("\n The classification report:\n ",classification_report(y_test, y_pred))


print("*******************************************")
```

```
*****************RESULTS*****************

 The accuracy score of train Model : 0.9289519154155363

 The accuracy score test of Model : 0.9174883021390374

 The ROC AUC score of Model : 0.6326489653022542

 The CV score of Model : 0.9171970995640825

 The confusion Matrix :
  [[42573   377]
 [ 3573  1349]]

 The classification report:
               precision    recall  f1-score   support

           0       0.92      0.99      0.96     42950
           1       0.78      0.27      0.41      4922

    accuracy                           0.92     47872
   macro avg       0.85      0.63      0.68     47872
weighted avg       0.91      0.92      0.90     47872

*******************************************
```

Multinomial NB

```
mnn_clf = MultinomialNB()
mnn_clf.fit(X_train, y_train)
y_pred_train = mnn_clf.predict(X_train)
y_pred = mnn_clf.predict(X_test)
print("*****************RESULTS*****************")
print("\n The accuracy score of train Model :", accuracy_score(y_train,y_pred_train))

print("\n The accuracy score test of Model :", accuracy_score(y_test,y_pred))
print("\n The ROC AUC score of Model :", roc_auc_score(y_test,y_pred))

CV=cross_val_score(mnn_clf,X_train, y_train,cv=5)
print("\n The CV score of Model :", CV.mean())

print("\n The confusion Matrix :\n ", confusion_matrix(y_test, y_pred))
print("\n The classification report:\n ",classification_report(y_test, y_pred))


print("*******************************************")
```

```
*****************RESULTS*****************

 The accuracy score of train Model : 0.9513334944807026

 The accuracy score test of Model : 0.9472342914438503

 The ROC AUC score of Model : 0.7588672369286836

 The CV score of Model : 0.9471526225477417

 The confusion Matrix :
  [[42778   172]
 [ 2354  2568]]

 The classification report:
               precision    recall  f1-score   support

           0       0.95      1.00      0.97     42950
           1       0.94      0.52      0.67      4922

    accuracy                           0.95     47872
   macro avg       0.94      0.76      0.82     47872
weighted avg       0.95      0.95      0.94     47872

*******************************************
```

SVC

```
svc_clf = SVC()
svc_clf.fit(X_train, y_train)
y_pred_train = svc_clf.predict(X_train)
y_pred = svc_clf.predict(X_test)
print("*****************RESULTS*****************")
print("\n The accuracy score of train Model :", accuracy_score(y_train,y_pred_train))

print("\n The accuracy score test of Model :", accuracy_score(y_test,y_pred))
print("\n The ROC AUC score of Model :", roc_auc_score(y_test,y_pred))

CV=cross_val_score(svc_clf,X_train, y_train,cv=5)
print("\n The CV score of Model :", CV.mean())

print("\n The confusion Matrix :\n ", confusion_matrix(y_test, y_pred))
print("\n The classification report:\n ",classification_report(y_test, y_pred))


print("*******************************************")
```

```
*****************RESULTS*****************

 The accuracy score of train Model : 0.9804384998970448

 The accuracy score test of Model : 0.9565299131016043

 The ROC AUC score of Model : 0.810908127203466

 The CV score of Model : 0.9561142099342262

 The confusion Matrix :
  [[42702   248]
 [ 1833  3089]]

 The classification report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.93      0.63      0.75      4922

    accuracy                           0.96     47872
   macro avg       0.94      0.81      0.86     47872
weighted avg       0.96      0.96      0.95     47872

*******************************************
```

- # Key Metrics for success in solving problem under consideration

An key/evaluation metric quantifies the performance of a predictive model This typically. Following are the metrics I have used to evaluate the model performance.

1. Confusion Matrix:

The confusion matrix provides a more insightful picture based on the counts of test records correctly and incorrectly predicted by the model, and what type of errors are being made.

The confusion matrix is useful for measuring Recall (also known as Sensitivity), Precision, Specificity, Accuracy, and, most importantly, the AUC-ROC Curve.

2. Sensitivity:

It measures how many observations out of all positive observations have we classified as positive. It tells us how many fraudulent transactions we recalled from all fraudulent transactions.

3. Precision:

It measures how many observations predicted as positive are in fact positive. Taking our fraud detection example, it tells us what ratio of transactions correctly classified as fraudulent.

4. Accuracy:

It measures how many observations, both positive and negative, were correctly classified.

5. F1 Score:

A good F1 score means that we have low false positives and low false negatives, so we're correctly identifying real threats, and we are not disturbed by false alarms.

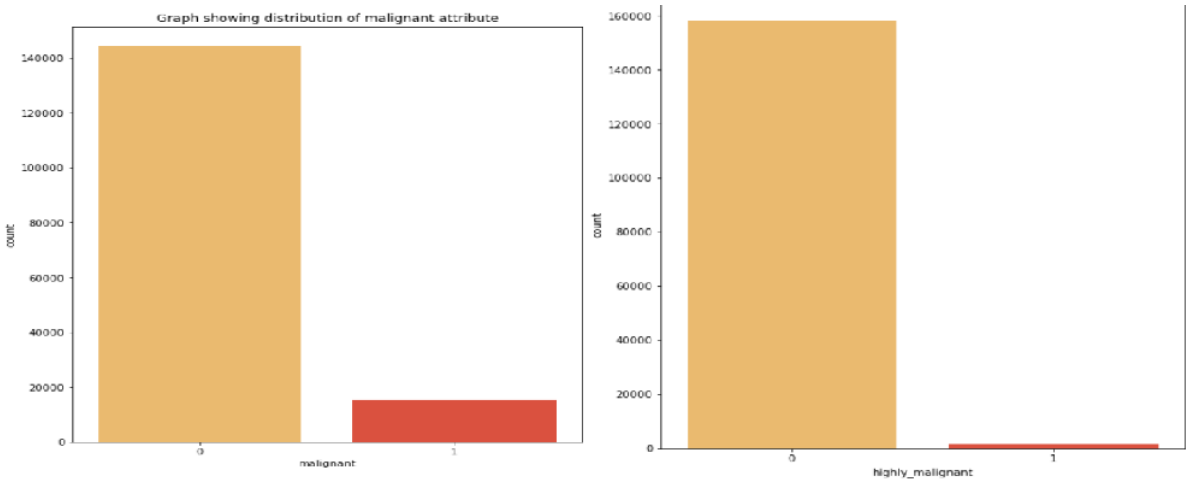
6. Cross Validation score:

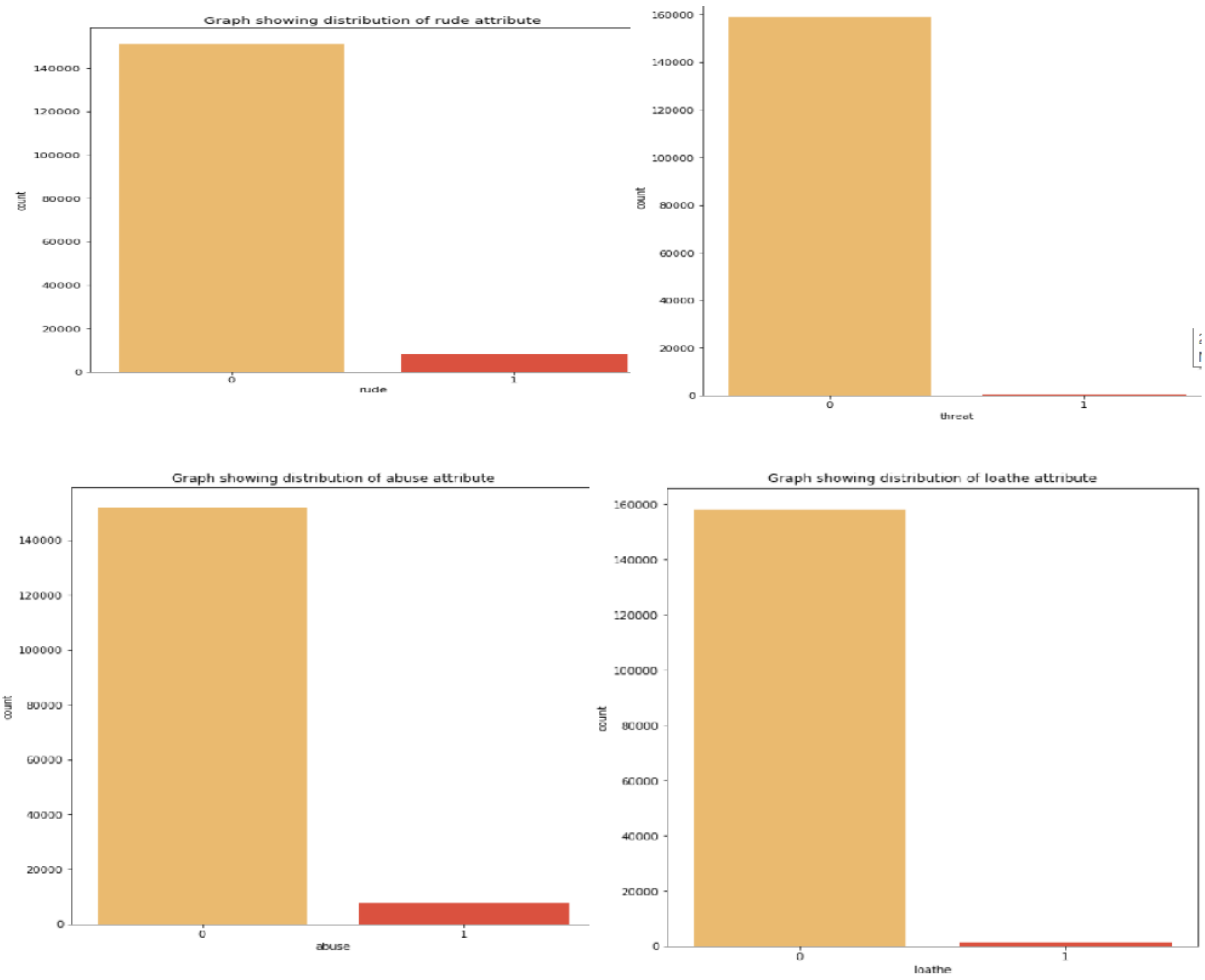
It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

- **Visualizations**

Visualization plays a crucial role in EDA as well as during modelling. It gives a better idea about the things going on beautifully.

We have used matplotlib and seaborn to interpret the relationship, we have plotted the graph using countplot to know how the data is distributed.





- # Interpretation of the Results

Basing on the result obtained ‘Support Vector Machine’ Classifier have performed well and has given better result as compared to other models so it has been selected as final model and it will be saved using pickle library.

Saving the BEST MODEL

We can see from the ROC curve that, SVC is giving the highest area which proves that it is the best model.

```
:  1  import pickle
   2
   3  file = open('Malignant Comments classifier.pkl', 'wb')
   4  pickle.dump(svc_clf, file)
```

CONCLUSION

- ## Key Findings and Conclusions of the Study

  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

  The power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important steps, tried to make comments shorter and all the necessary keywords in it.
  Various algorithms I used in this dataset and to get out best result and save that model. The best algorithm is SVC.

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

  Additionally, the followings are some suggested studies to be considered as future work in this area:

  a) We suggest a plan to improve the NLP classifiers: first by using other algorithms which such as Support Vector Clustering (SVC) and Convolutional Neural Networks (CNN); secondly, extend the classifiers to the overall goal which is multi-label classifiers. in the current study, the problem simplified into two classes but it worth to pursue a main goal which is 6 classes of comments.

  b) We also suggest using SVM for text processing and text classification. It requires a grid search for hyper-parameter tuning to get the best results.

  c) using Other DNN techniques (CNN)) because some recently published papers have shown that CNN proves to have a very high performance for various NLP tasks.