Malignant Comments Classifier Project

Submitted by:

Amruta Shah

ACKNOWLEDGMENT

A unique opportunity like this comes very rarely. It is indeed a pleasure for me to have worked on this project.

The satisfaction that accompanies the successful completion of this project is incomplete without the mention of the people and references whose guidance has made it possible for me to complete this project.

I am grateful to my internship company **Flip Robo Technologies** with its ideals and inspiration for providing me with facilities that has made this project a success.

Also, I am grateful to my institute “Data Trained” for providing the opportunity to work as an intern in “Flip Robo Technologies” and giving me the great knowledge to complete this project.

I like to acknowledge the effort put in by our SME Khushboo Garg helping me understand the relevant concepts related to Data pre-processing and providing guidance when necessary.

Also based on below references I am able to encourage my knowledge time to time to improve my knowledge.

References:

- DecisionTreeClassifier
- Hyper-parameter Tuning
- SMOTE
- Getting started with XGBoost
- scikit learn

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.

- ## Conceptual Background of the Domain Problem

  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.

- ## Review of Literature

  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.

Data Set Description: The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'. The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

- 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.

- Motivation for the Problem Undertaken

  Motivation behind this project is that mainly focuses to prove that 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 Modelling of the Problem

  Mathematical Summary:
  Dimensions of Dataset: There are 8 columns and 159571 rows in this dataset.
  Null Values: There are no null values in this dataset.
  Skewness: Skewness is present in column.
  Statistical Summary: Standard deviation is very high in most of the columns.
  Special Character are present in both dataset (train and test).
  Target Variable: We have 6 target variables which we need to combine and work.

- Data Sources and their formats

  Data Sources: We have train and test dataset. We have taken from social media.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.

  Data Formats: Comment Text & Id columns are object datatype which we need to encode & pre-process.

- Data Pre-processing Done

  First of all, I have check for the duplicates & treated well.

Check for the null values by plotting the heatmap I can not find the same. Moving further I have convert all the data into lower case by using string. lower() method. Further , I have done some NLP text pre-processing like string punctuation, removing special character, removing emoji, remove stop words and meanwhile I have check for the text length and added into dataset. This pre-processing I have done on both train and test data set which is provided.
All steps are shown below.

```
#Converting all capital letter into small letters
df['comment_text'] =df['comment_text'].str.lower()
df['comment_text']
```

```
0         explanation\nwhy the edits made under my usern...
1         d'aww! he matches this background colour i'm s...
2         hey man, i'm really not trying to edit war. it...
3         "\nmore\ni can't make any real suggestions on ...
4         you, sir, are my hero. any chance you remember...
                                ...
159566    ":::::and for the second time of asking, when ...
159567    you should be ashamed of yourself \n\nthat is ...
159568    spitzer \n\numm, theres no actual article for ...
159569    and it looks like it was actually you who put ...
159570    "\nand ... i really don't think you understand...
Name: comment_text, Length: 159571, dtype: object
```

```
# Checking special characater if any
import string
alphabet = string.punctuation

# Storing the punctuation free text
df['clean_comment_text']=df['comment_text'].apply(lambda x:"".join([i for i in x if i not in alphabet]))
```

```
df.head(3)
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_comment_text
0	0000997932d777bf	explanation\nwhy the edits made under my usern...	0	0	0	0	0	0	264	explanation\nwhy the edits made under my usern...
1	000103f0d9cfb60f	d'aww! he matches this background colour i'm s...	0	0	0	0	0	0	112	daww he matches this background colour im seem...
2	000113f07ec002fd	hey man, i'm really not trying to edit war. it...	0	0	0	0	0	0	233	hey man im really not trying to edit war its j...

```
# Checking the text length after removing special character
df['str_re_length']=df['clean_comment_text'].str.len()
df['str_re_length']
```

```
# Removing some special character
df['clean_comment_text'] = df['clean_comment_text'].str.replace("4", "")
df['clean_comment_text'] = df['clean_comment_text'].str.replace("\n", "")
df['clean_comment_text'] = df['clean_comment_text'].str.replace("2", "")
```

```
# Checking the text length after removing special character
df['chr_length']=df['clean_comment_text'].str.len()
df['chr_length']
```

Stop word removal

```
Stopwords = set(stopwords.words('english') + ['u', 'ü', 'ur','im', 'dont', 'doin', 'ure'])

df['clean_comment_text']=df['clean_comment_text'].apply(lambda x:' '.join(term for term in x.split() if term not in Stopwords))
```

```
df['new_length']=df['clean_comment_text'].str.len()
df['new_length']
```

```
0         179
```

As we seen there are many target variable so lets gruop them into one for better prediction.

```
# Adding the all target variable with one column
df['total_malignant']=df[['malignant', 'highly_malignant', 'rude', 'threat', 'abuse','loathe']].sum(axis=1).astype(int)
```

```
df['total_malignant'].value_counts() #lets check the count for each class
```

```
0    143346
1      6360
3      4209
2      3480
4      1760
5       385
6        31
Name: total_malignant, dtype: int64
```

```
# Lets convert each class in to 1 & 0 as target variable
df['total_malignant']=df['total_malignant']>0
df['total_malignant'].value_counts() #lets check the count for each class
```

As our dataset has objective type data so covert them into int by using Encoding method as shown.

```
# Lets seaprate the catogorical data & numirical data
numerics=['int8','int16','int32','int64','float16','float32','float64']
cat_col=[] #empty list
features=df.columns.values.tolist()
for i in features:
    if df[i].dtype in numerics:
        continue
    cat_col.append(i)
cat_col
```

```
['id', 'clean_comment_text']
```

```
# Lets frist covert categorical data(type & column) into int
label = LabelEncoder()
for i in cat_col:
    data=label.fit_transform(df[i])
    pd.Series(data)
    df[i]=data
```

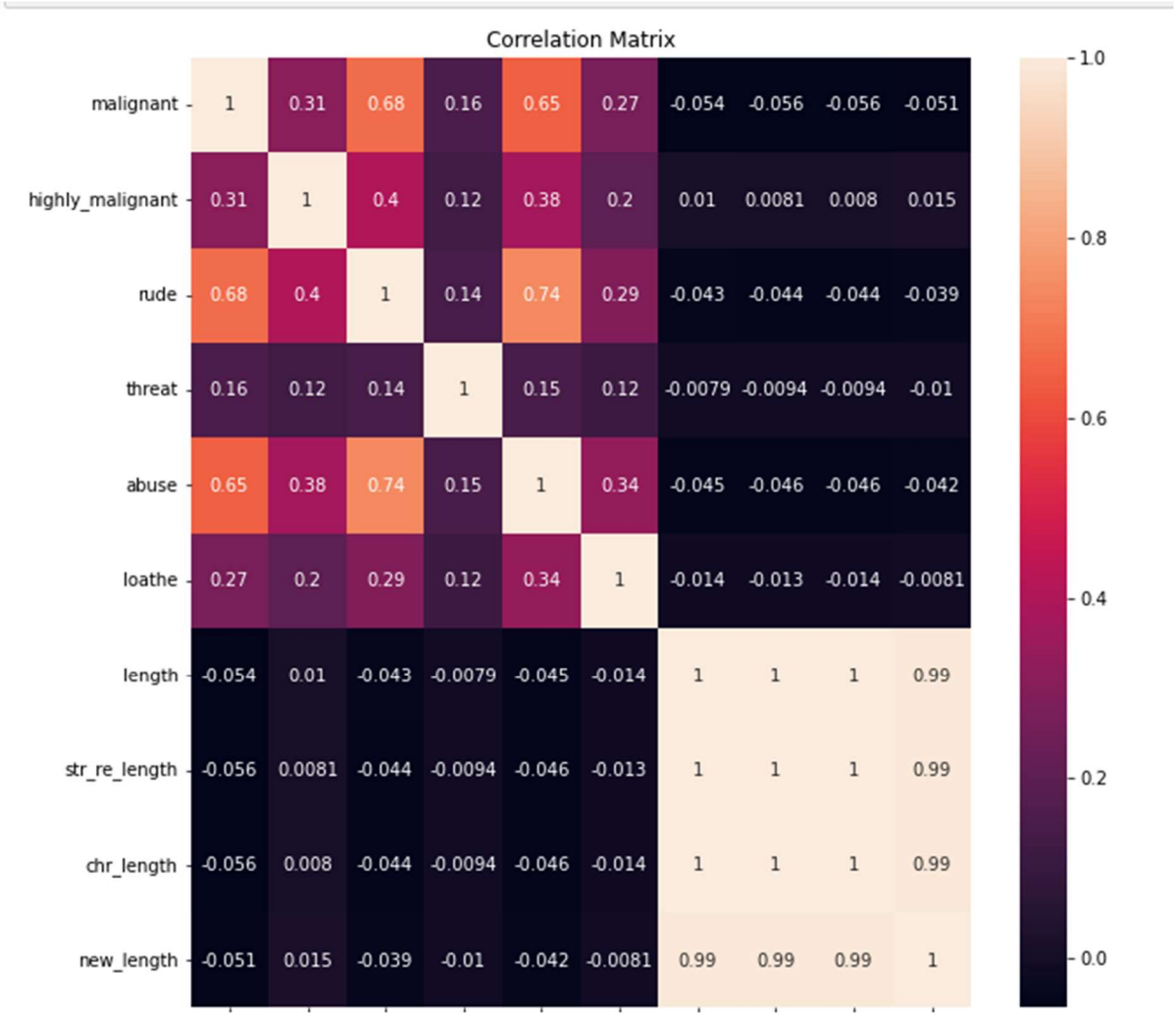
Handing Class imbalance problem:
I used Oversampling method, at last I used SMOTE method to handle the class imbalance problem.

```
# We have imbalance dataset lets use SMOTE
# Lets use of Resampling Techniques to handle Imbalanced Data
from imblearn.over_sampling import SMOTE
from collections import Counter

ove_smp=SMOTE(0.90) # selecting 90% data for resampling
x_train_ns,y_train_ns=ove_smp.fit_resample(x_train,y_train)
print(Counter(y_train))
print(Counter(y_train_ns))
```

```
Counter({0: 100443, 1: 11256})
Counter({0: 100443, 1: 90398})
```

- Data Inputs- Logic- Output Relationships

  Multicollinearity:
  To check the Input output relation, I have plotted the heatmap and data.corr () technique to check the multicollinearity. As shown in below screenshot. And detected no multicollinearity.




```
df.corr()
```

	malignant	highly_malignant	rude	threat	abuse	loathe	length	str_re_length	chr_length	new_length
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009	-0.054413	-0.055664	-0.055868	-0.051310
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600	0.010175	0.008114	0.008017	0.014845
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867	-0.042910	-0.043699	-0.043900	-0.038972
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128	-0.007917	-0.009362	-0.009398	-0.010067
abuse	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736	-0.045027	-0.045764	-0.045918	-0.041729
loathe	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000	-0.013628	-0.013477	-0.013582	-0.008112
length	-0.054413	0.010175	-0.042910	-0.007917	-0.045027	-0.013628	1.000000	0.998731	0.998583	0.989935
str_re_length	-0.055664	0.008114	-0.043699	-0.009362	-0.045764	-0.013477	0.998731	1.000000	0.999943	0.990516
chr_length	-0.055868	0.008017	-0.043900	-0.009398	-0.045918	-0.013582	0.998583	0.999943	1.000000	0.990221
new_length	-0.051310	0.014845	-0.038972	-0.010067	-0.041729	-0.008112	0.989935	0.990516	0.990221	1.000000

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

  This Dataset is class imbalanced.

- Hardware and Software Requirements and Tools Used

Machine: Can use a laptop/desktop.

Operating system: Windows 11, Mac OS X 10.9 Mavericks or Higher

RAM & Processor: 4 GB+ RAM, i3 5th Generation 2.2 Ghz or equivalent/higher .

Tools Used: Jupyter Notebook, Microsoft Excel.

Libraries Used :

```
import pandas as pd        # for data manipulation

import numpy as np         # for mathematical calculations

import seaborn as sns      # for data visualization

import matplotlib.pyplot as plt  #for graphical analysis

%matplotlib inline

from scipy.stats import zscore # to remove outliers

from sklearn.preprocessing import StandardScaler  # for normalize the model

from sklearn.preprocessing import LabelEncoder  # to convert object into int

from sklearn.model_selection import train_test_split  # for train and test model

import warnings                    # to ignore any warnings

warnings.filterwarnings("ignore")

from sklearn import metrics  # for model evaluation

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report.
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

I have dropped the duplicated to avoid impact of the same on result.

I have used NLP text Pre-processing techniques to processes the data.

Used SMOTE method to deal with class imbalance problem.

- Testing of Identified Approaches (Algorithms)

I have selected 5 models for the prediction which is used for the training and testing.

1) Logistic Regression from sklearn.linear_model
2) DecisionTreeClassifier from sklearn.tree
3) RandomForestClassifier from sklearn.ensemble
4) ADABoostClassifier from sklearn.ensemble
5) XGBoostClassifier from sklearn.ensemble
6) GradientBoostClassifier from XGBoost
7) KNeighborsClassifier from sklearn.neighbors

- Run and Evaluate selected models

1) **Logistic Regression from sklearn.linear_model:**
Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is binary, which means there would be only two possible classes 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

2) **DecisionTreeClassifier from sklearn.tree:**
Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different

conditions. The two main entities of a tree are decision nodes, where the data is split and leaves, where we get the outcome.

3) **RandomForestClassifier from sklearn.ensemble:**

As we know that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

4) **ADABoostClassifier:**

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

5) **GradientBoostClassifier from sklearn.ensemble:**

The power of gradient boosting machines comes from the fact that they can be used on more than binary classification problems, they can be used on multi-class classification problems and even regression problems.

The Gradient Boosting Classifier depends on a loss function. A custom loss function can be used, and many standardized loss functions are supported by gradient boosting classifiers, but the loss function has to be differentiable. Gradient boosting systems have two other necessary parts: a weak learner and an additive component. Gradient boosting systems use decision trees as their weak learners.

6) **KNeighborsClassifier from sklearn.neighbors:**

The KNN (k-nearest neighbour) algorithm is a fundamental supervised machine learning algorithm used to solve regression and classification problem statements.

The K-Nearest Neighbour or the KNN algorithm is a machine learning algorithm based on the supervised learning model. The K-NN algorithm works by assuming that similar things exist close to each other. Hence, the K-NN algorithm utilises feature similarity between the new data points and the points in the training set (available cases) to predict the values of the new data points. In essence, the K-NN algorithm assigns a value to the latest data point based on how closely it resembles the points in the training set. K-NN algorithm finds application in both classification and regression problems but is mainly used for classification problems.

7) **XGBClassifier from XGBoost:**

XGBoost is short for "eXtreme Gradient Boosting." The "eXtreme" refers to speed enhancements such as parallel computing and cache awareness that makes XGBoost approximately 10 times faster than traditional Gradient Boosting. In addition, XGBoost includes a unique split-finding algorithm to optimise trees, along with built-in regularisation that reduces over-fitting. Generally speaking, XGBoost

is a faster, more accurate version of Gradient Boosting.

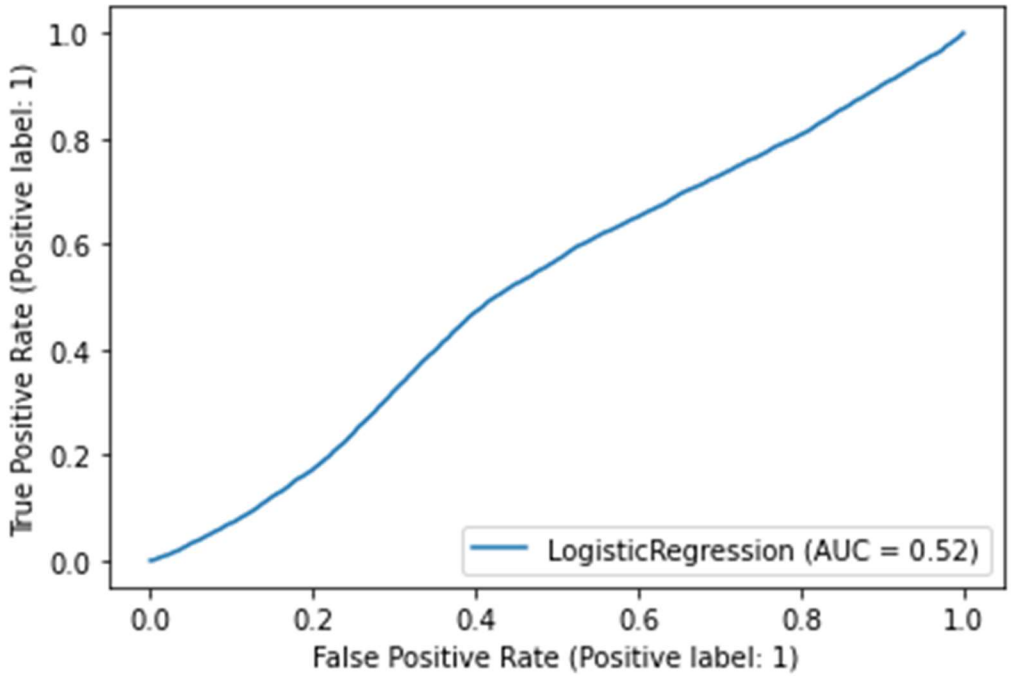
```
from sklearn.linear_model import LogisticRegression
LR= LogisticRegression()

from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()

from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
ada=AdaBoostClassifier()
gb=GradientBoostingClassifier()

from sklearn.ensemble import RandomForestClassifier
rfc= RandomForestClassifier()

import xgboost as xgb
xgb=xgb.XGBClassifier()

from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()

from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve,roc_auc_score, plot_roc_curve, auc

models=[]
models.append(('LogisticRegression', LR))
models.append(('DecisionTreeClassifier', dt))
models.append(('AdaBoostClassifier', ada))
models.append(('GradientBoostingClassifier', gb))
models.append(('RandomForestClassifier', rfc))
models.append(('XGBClassifier', xgb))
models.append(('KNeighborsClassifier', knn))
```

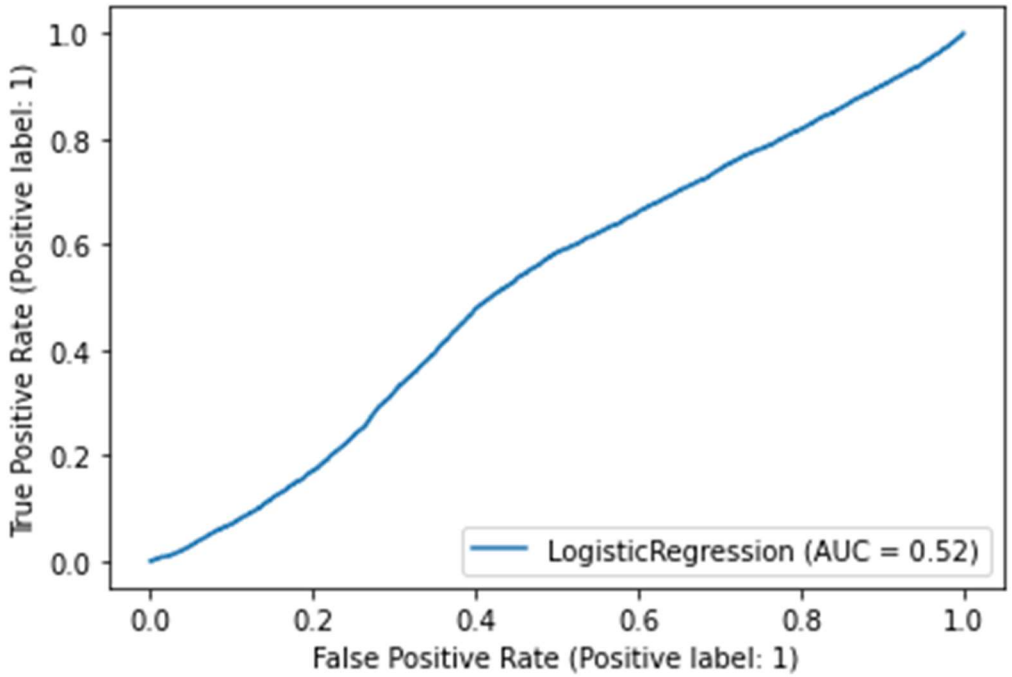
```
********** LogisticRegression **********
LogisticRegression()
Train Report: 0.5263177199867953
Test Report: 0.8962023729946524
Classification Report:               precision    recall  f1-score   suppor
t

           0       0.90      1.00      0.95     42903
           1       0.00      0.00      0.00      4969

    accuracy                           0.90     47872
   macro avg       0.45      0.50      0.47     47872
weighted avg       0.80      0.90      0.85     47872

Confusion Matrix: [[42903     0]
 [ 4969     0]]
Accuracy: 52.63 %
Standard Deviation: 0.00 %
```

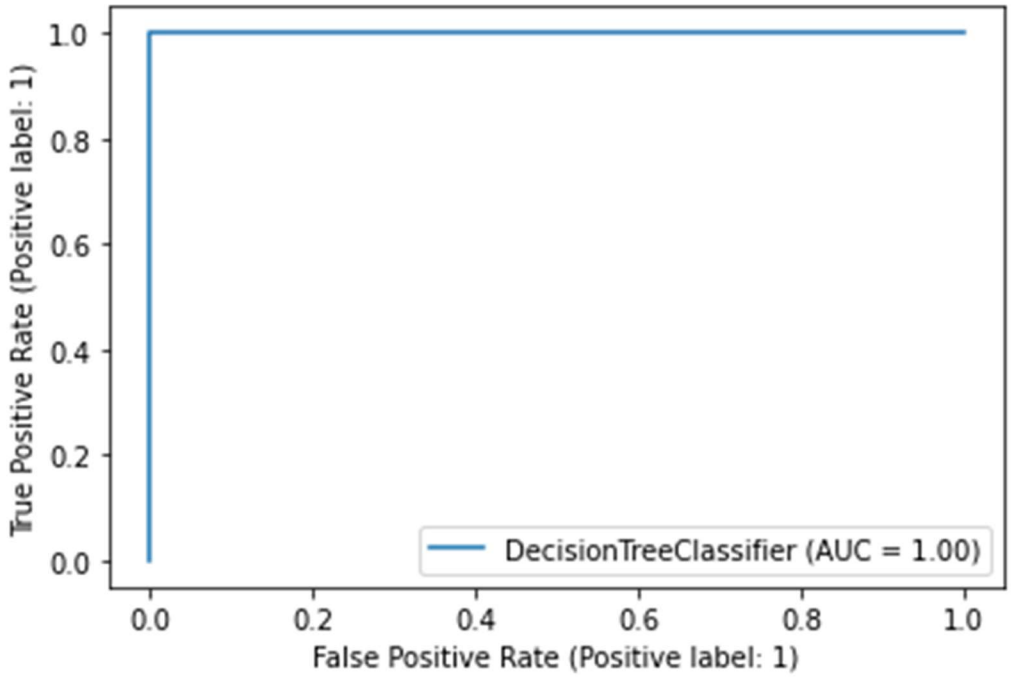
```
********** DecisionTreeClassifier **********
DecisionTreeClassifier()
Train Report: 1.0
Test Report: 0.7071983622994652
Classification Report:               precision    recall  f1-score   suppor
t

           0       0.92      0.74      0.82     42903
           1       0.16      0.44      0.24      4969

    accuracy                           0.71     47872
   macro avg       0.54      0.59      0.53     47872
weighted avg       0.84      0.71      0.76     47872

Confusion Matrix: [[31693 11210]
 [ 2807  2162]]
Accuracy: 70.86 %
Standard Deviation: 0.40 %


********** AdaBoostClassifier **********
AdaBoostClassifier()
Train Report: 0.6071494070980554
Test Report: 0.7574991644385026
Classification Report:               precision    recall  f1-score   suppor
t

           0       0.92      0.80      0.86     42903
           1       0.18      0.38      0.25      4969

    accuracy                           0.76     47872
   macro avg       0.55      0.59      0.55     47872
weighted avg       0.84      0.76      0.79     47872

Confusion Matrix: [[34374  8529]
 [ 3080  1889]]
Accuracy: 60.67 %
Standard Deviation: 0.04 %


********** GradientBoostingClassifier **********
GradientBoostingClassifier()
Train Report: 0.6366661252037036
Test Report: 0.7846966911764706
Classification Report:               precision    recall  f1-score   suppor
t
```

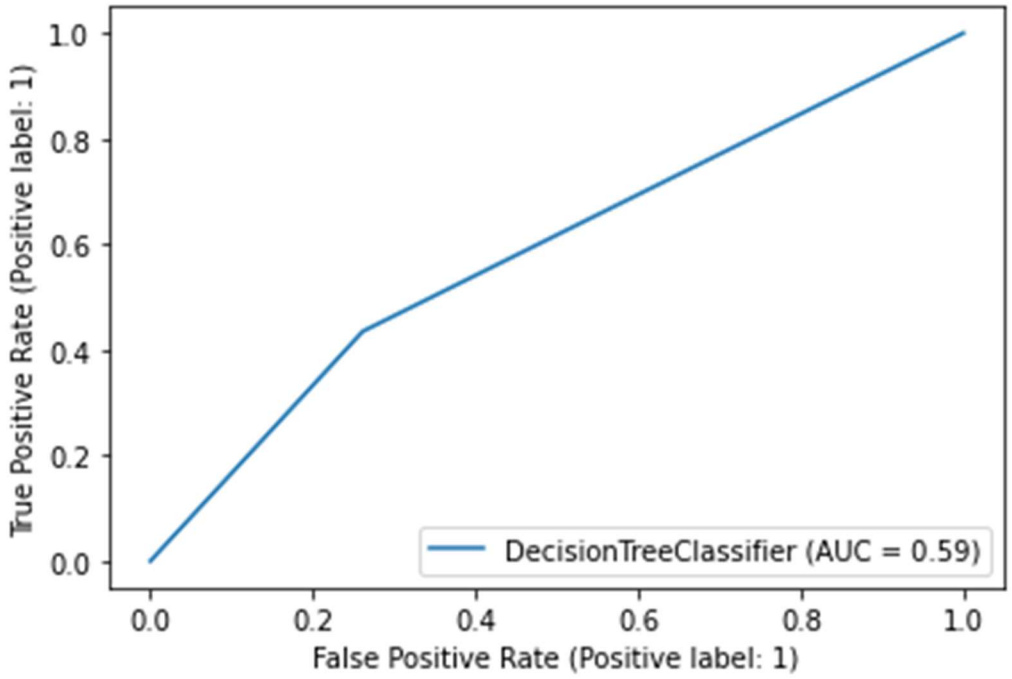
```
           0       0.92      0.83      0.87     42903
           1       0.21      0.40      0.28      4969

    accuracy                           0.78     47872
   macro avg       0.57      0.62      0.58     47872
weighted avg       0.85      0.78      0.81     47872

Confusion Matrix: [[35571  7332]
 [ 2975  1994]]
Accuracy: 63.07 %
Standard Deviation: 0.55 %


********** RandomForestClassifier **********
RandomForestClassifier()
Train Report: 0.999984280107524
Test Report: 0.7450701871657754
Classification Report:               precision    recall  f1-score   suppor
t

           0       0.92      0.79      0.85     42903
           1       0.18      0.39      0.24      4969

    accuracy                           0.75     47872
   macro avg       0.55      0.59      0.54     47872
weighted avg       0.84      0.75      0.78     47872

Confusion Matrix: [[33717  9186]
 [ 3018  1951]]
Accuracy: 75.73 %
Standard Deviation: 0.30 %


********** XGBClassifier **********
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise'
,
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight
=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state
=0,
```

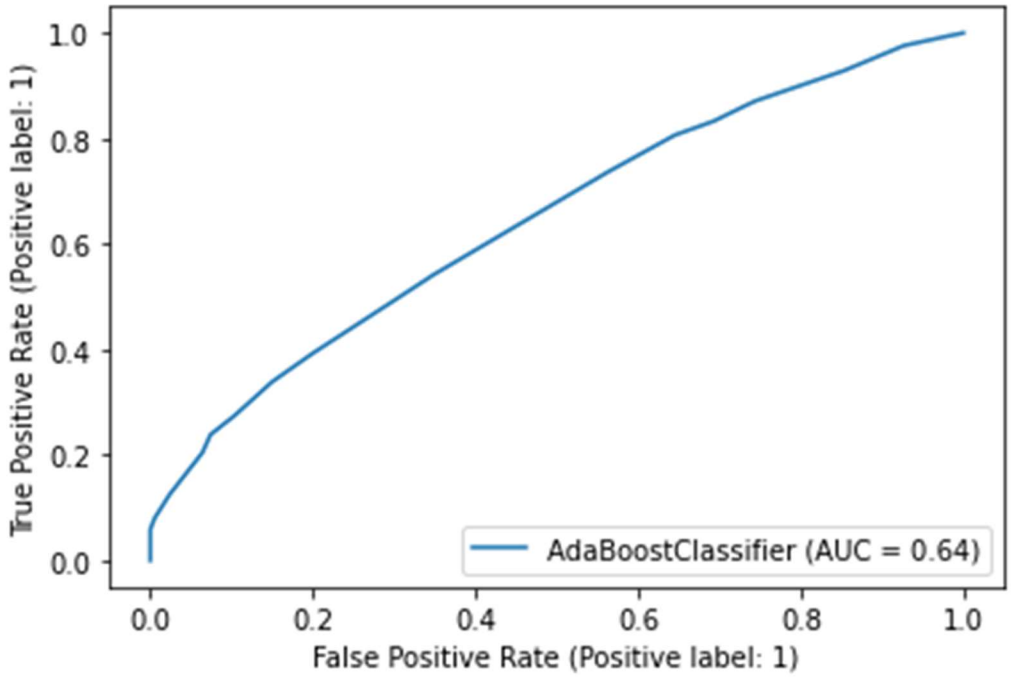
```
              reg_alpha=0, reg_lambda=1, ...)
Train Report: 0.7109111773675468
Test Report: 0.7349390040106952
Classification Report:               precision    recall  f1-score   suppor
t

           0       0.93      0.76      0.84     42903
           1       0.20      0.52      0.29      4969

    accuracy                           0.73     47872
   macro avg       0.57      0.64      0.56     47872
weighted avg       0.86      0.73      0.78     47872

Confusion Matrix: [[32611 10292]
 [ 2397  2572]]
Accuracy: 67.70 %
Standard Deviation: 0.11 %


********************** KNeighborsClassifier **********************
KNeighborsClassifier()
Train Report: 0.8510120990772423
Test Report: 0.6836146390374331
Classification Report:               precision    recall  f1-score   suppor
t

           0       0.91      0.72      0.80     42903
           1       0.14      0.41      0.21      4969

    accuracy                           0.68     47872
   macro avg       0.53      0.56      0.51     47872
weighted avg       0.83      0.68      0.74     47872

Confusion Matrix: [[30700 12203]
 [ 2943  2026]]
Accuracy: 74.54 %
Standard Deviation: 0.02 %
```

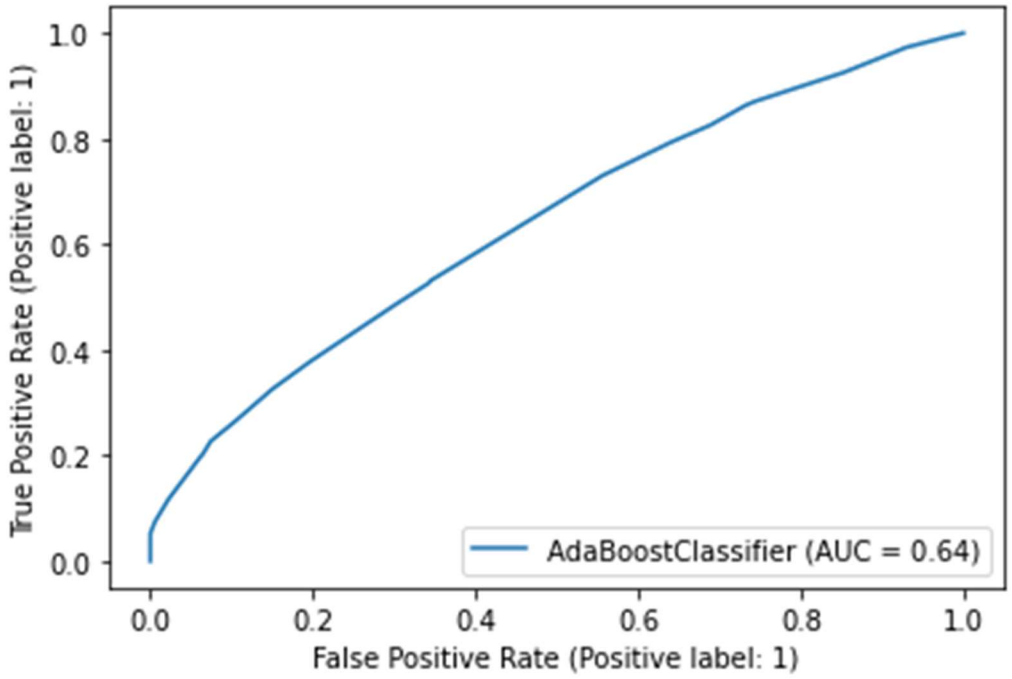

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
LogisticRegression (AUC = 0.52)
0.0
0.2
0.4
0.6
0.8
1.0

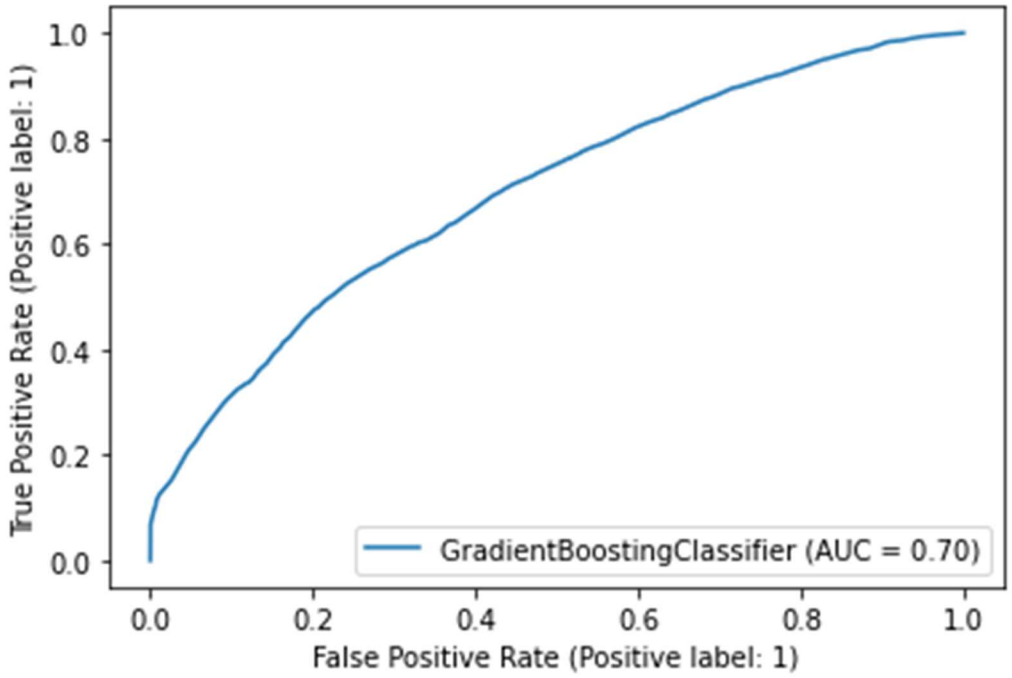

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
LogisticRegression (AUC = 0.52)
0.0
0.2
0.4
0.6
0.8
1.0

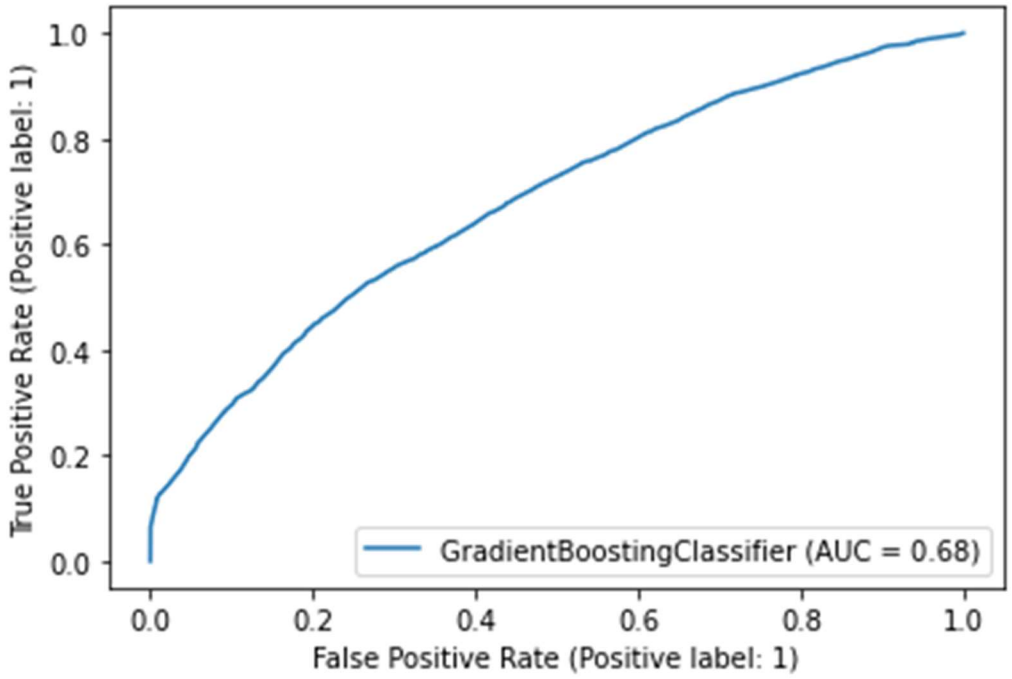

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
DecisionTreeClassifier (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

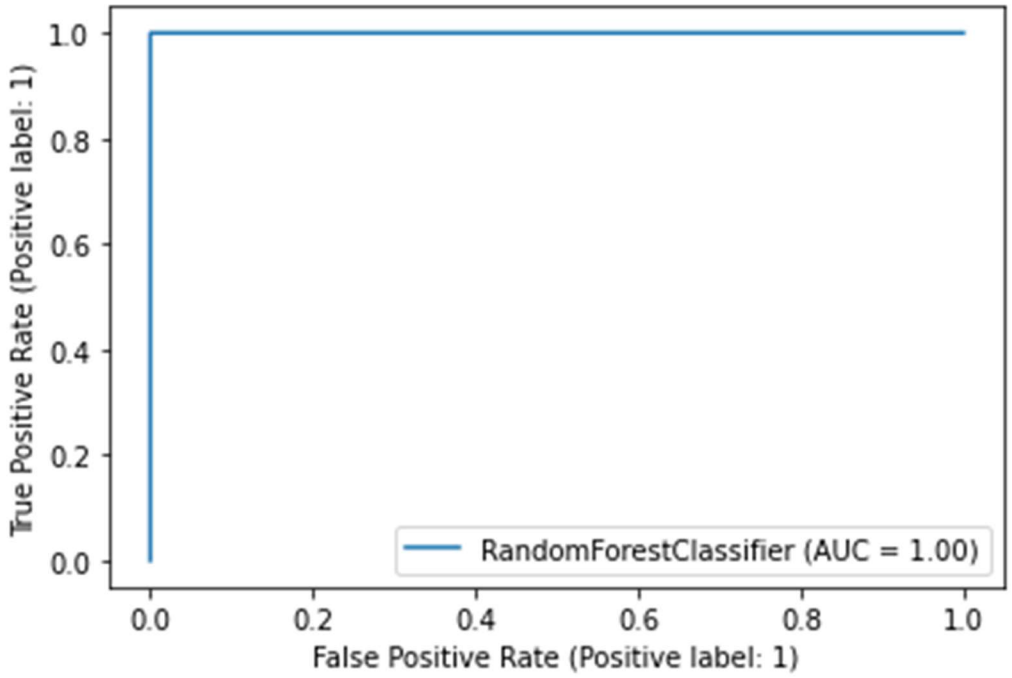

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
DecisionTreeClassifier (AUC = 0.59)
0.0
0.2
0.4
0.6
0.8
1.0

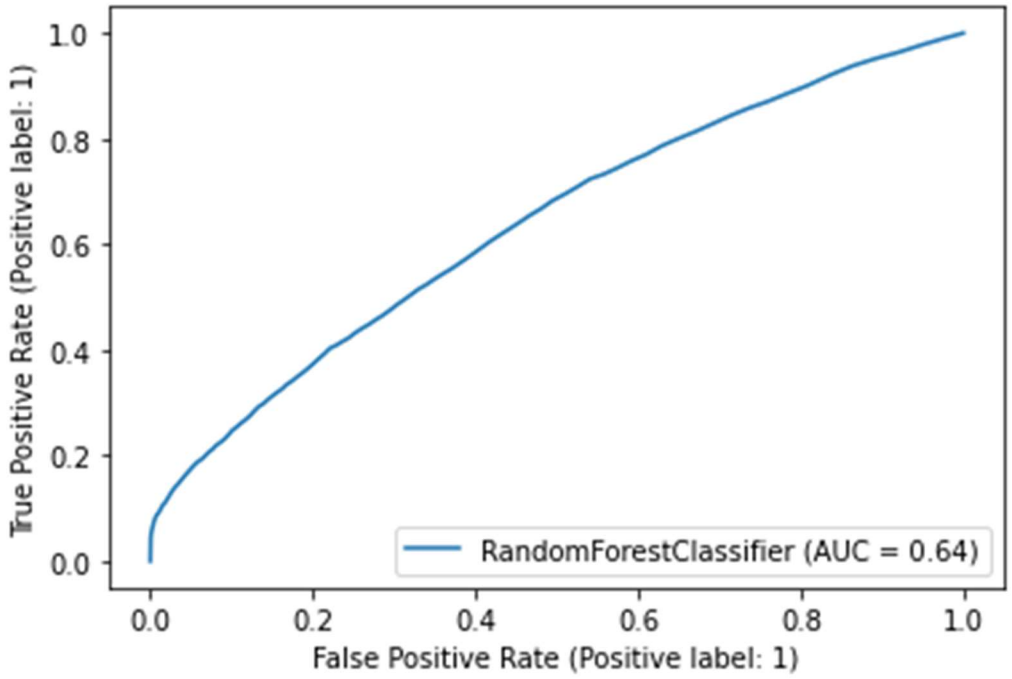

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
AdaBoostClassifier (AUC = 0.64)
0.0
0.2
0.4
0.6
0.8
1.0

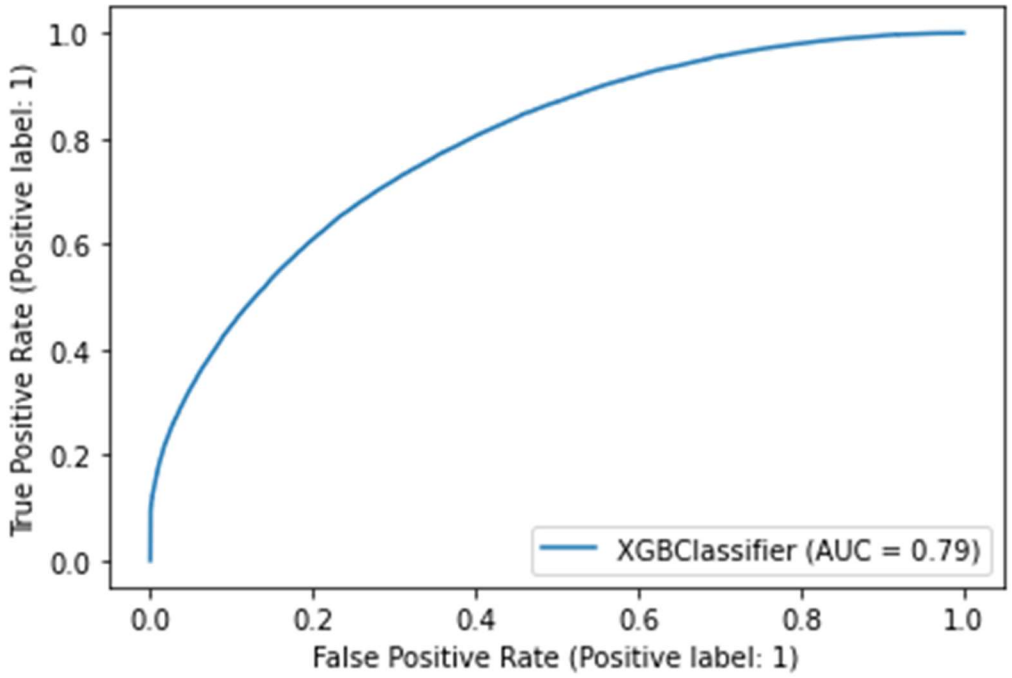

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
AdaBoostClassifier (AUC = 0.64)
0.0
0.2
0.4
0.6
0.8
1.0

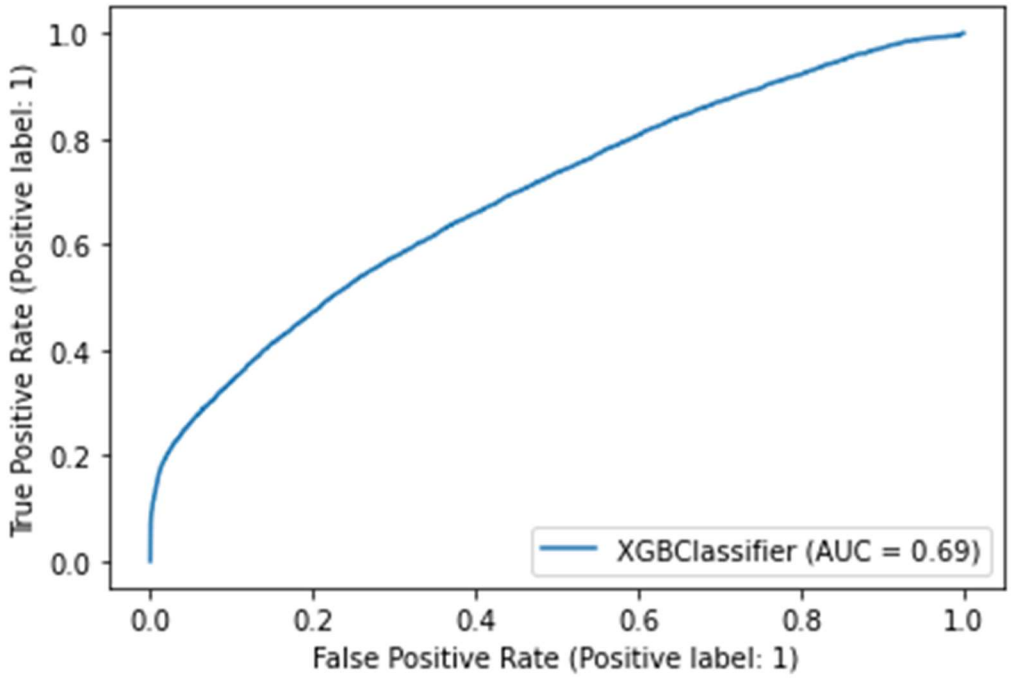

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
GradientBoostingClassifier (AUC = 0.70)
0.0
0.2
0.4
0.6
0.8
1.0

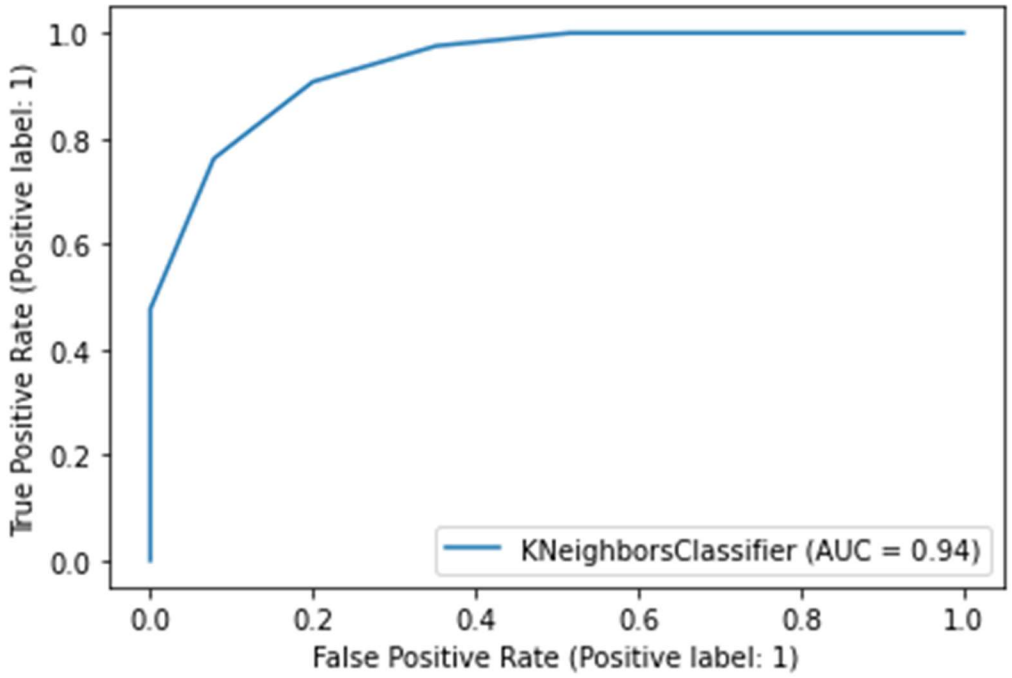

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
GradientBoostingClassifier (AUC = 0.68)
0.0
0.2
0.4
0.6
0.8
1.0

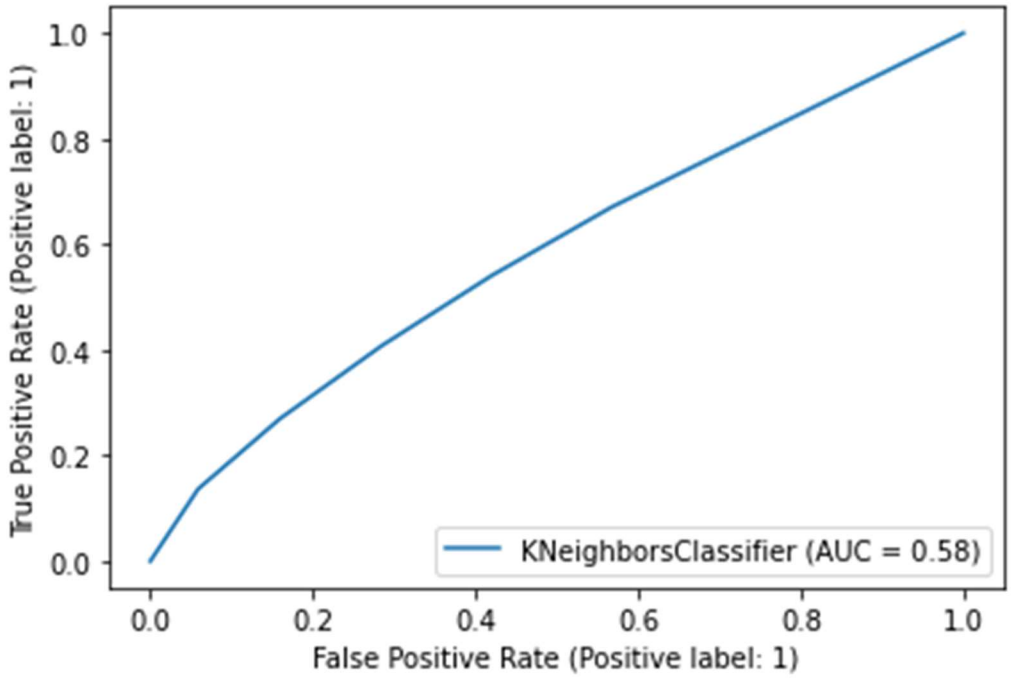

True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
RandomForestClassifier (AUC = 0.64)
0.0
0.2
0.4
0.6
0.8
1.0


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.79)
0.0
0.2
0.4
0.6
0.8
1.0


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
XGBClassifier (AUC = 0.69)
0.0
0.2
0.4
0.6
0.8
1.0





- Key Metrics for success in solving problem under consideration

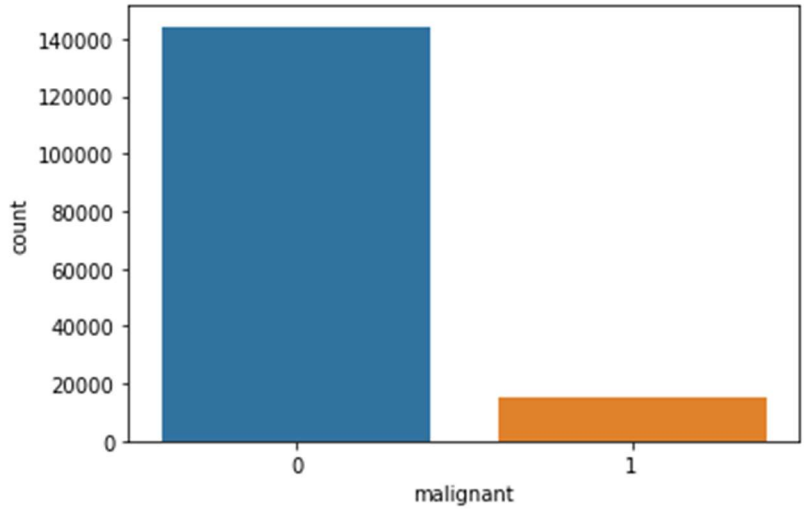
  I have used accuracy score, classification report & confusion matrix as metrices.

We got **our** best model looking at accuracy & confusion matrix is **XGBClassifier** with Kfold cross validation method with the accuracy score of 73.64% with hypertuning parameter RandomisedSearchCV.

- Visualizations:

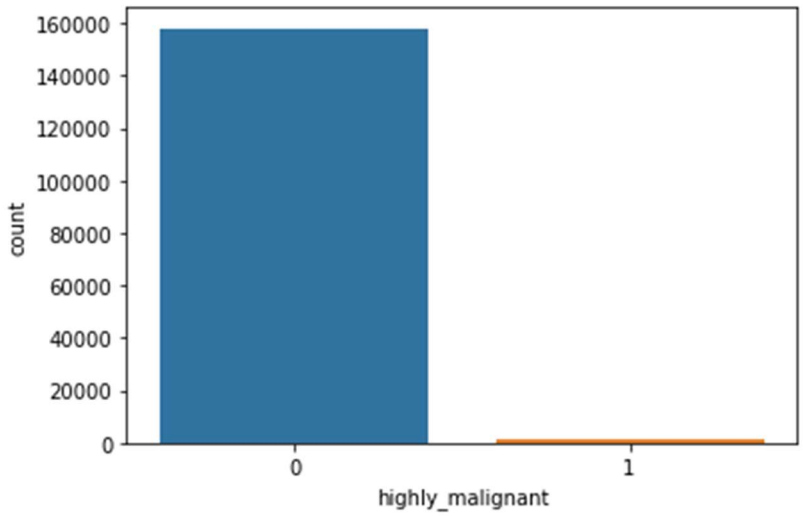
Univariate Plots

```
#plot each class frequency
sns.countplot(x='malignant',data=df)
plt.show()
print(df['malignant'].value_counts())
```



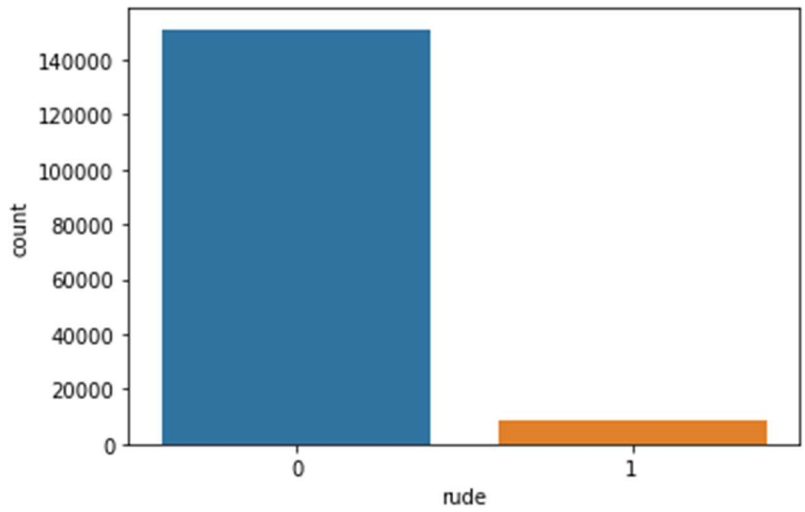

```
0    144277
1     15294
Name: malignant, dtype: int64
```

```
#plot each class frequency
sns.countplot(x='highly_malignant',data=df)
plt.show()
print(df['highly_malignant'].value_counts())
```



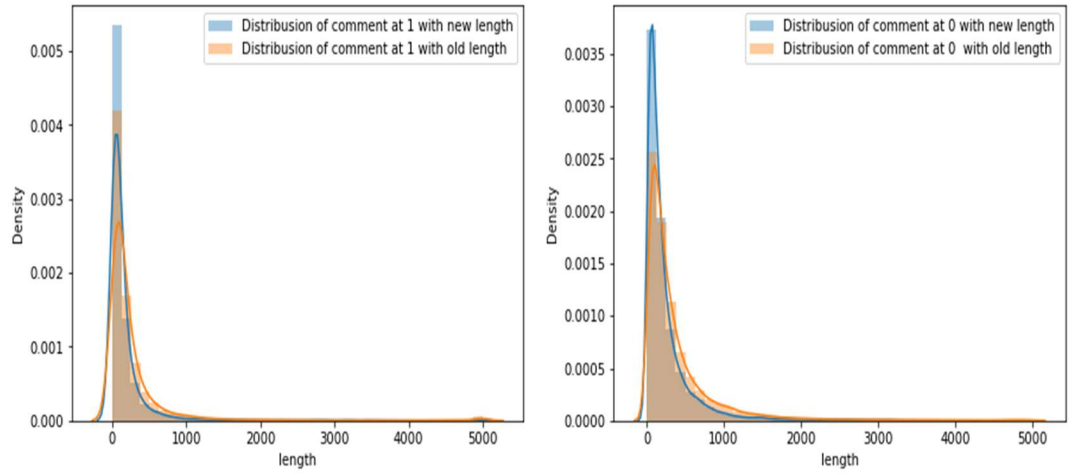

```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```

```
#plot each class frequency
sns.countplot(x='rude',data=df)
plt.show()
print(df['rude'].value_counts())
```

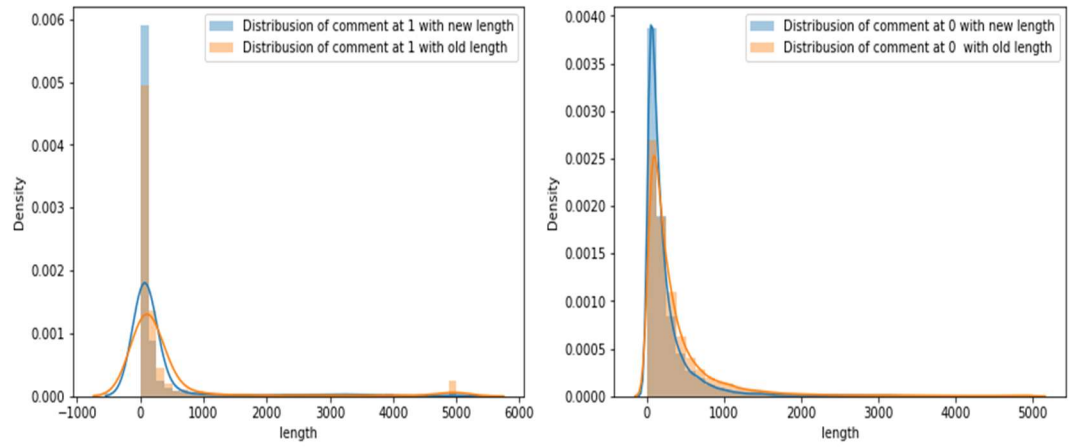



```
0    151122
1      8449
Name: rude, dtype: int64
```

```
#Bivariant graph
f,ax=plt.subplots(1,2, figsize =(15, 5))
sns.distplot(df[df['malignant']==1]['new_length'], ax=ax[0], bins=40, label='Distribusion of comment at 1 with new length').leger
sns.distplot(df[df['malignant']==0]['new_length'], ax=ax[1], bins=40,label='Distribusion of comment at 0 with new length').legend
sns.distplot(df[df['malignant']==1]['length'], ax=ax[0], bins=40, label='Distribusion of comment at 1 with old length').legend()
sns.distplot(df[df['malignant']==0]['length'], ax=ax[1], bins=40,label='Distribusion of comment at 0  with old length').legend()
plt.show()
```




```
#Bivariant graph
f,ax=plt.subplots(1,2, figsize =(15, 5))
sns.distplot(df[df['highly_malignant']==1]['new_length'], ax=ax[0], bins=40, label='Distribusion of comment at 1 with new length
sns.distplot(df[df['highly_malignant']==0]['new_length'], ax=ax[1], bins=40,label='Distribusion of comment at 0 with new length'
sns.distplot(df[df['highly_malignant']==1]['length'], ax=ax[0], bins=40, label='Distribusion of comment at 1 with old length').le
sns.distplot(df[df['highly_malignant']==0]['length'], ax=ax[1], bins=40,label='Distribusion of comment at 0  with old length').le
plt.show()
```



- Interpretation of the Results

  Looking at the accuracy i m selecting XGBoostClassifier further to get better accuracy I have used **RandomizedSearchCV** hyper tuning parameter.

```
# Hyper tuning by using RandomizedSearchCV With XGB
from sklearn.model_selection import RandomizedSearchCV

para={'n_estimators':[15,10,12],'gamma':[0.25,0.001,0.3,0.092],'max_depth':[25,20,11,21,8,9],'random_state':[20,10,30]}
rand=RandomizedSearchCV(estimator=xgb, cv=5,param_distributions=para)
rand.fit(x_train,y_train)

rand.best_params_
```

```
{'random_state': 20, 'n_estimators': 10, 'max_depth': 11, 'gamma': 0.092}
```

```
#Model no.5
import xgboost as xgb

xgb=xgb.XGBClassifier(random_state=20, n_estimators= 15, max_depth= 25, gamma= 0.001)

xgb.fit(x_train_ns,y_train_ns)
y_pred=xgb.predict(x_train_ns)
AS=accuracy_score(y_train_ns,y_pred)
print("Train Report:",AS*100)
pred=xgb.predict(x_test)
AS2=accuracy_score(y_test,pred)
print("Test Report:",AS2*100)
CR=classification_report(y_test,pred)
print("Classification Report:",CR)
CM=confusion_matrix(y_test,pred)
print("Confusion Matrix:", CM)

accuracies= cross_val_score(xgb, x_train_ns, y_train_ns, cv=2, scoring='accuracy')
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
cv_score.append(accuracies.mean()*100)
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))

plot_roc_curve(xgb,x_train_ns,y_train_ns)
auc = metrics.roc_auc_score(y_train_ns,y_pred) #calculate AUC of model
print('roc_auc_train',auc*100)  #print AUC score

plot_roc_curve(xgb,x_test, y_test)
auc1 = metrics.roc_auc_score(y_test, pred) #calculate AUC of model
print('roc_auc_test',auc1*100) #print AUC score
```

Saving the model:

```
#save model
import pickle
Filename='Finalized_model_Malignant_Comments.pickle'
pickle.dump(xgb,open(Filename,'wb'))
```

After finalising the model, I have load Test data set and do all the data pre-processing steps as mentioned above and do the prediction by using saved model.

```
df1["predictions"] = df1[['clean_comment_text','id']].apply(lambda s: dt.predict(s.values[None])[0], axis=1)
```

```
df1["predictions"].value_counts()
```

```
0    111725
1     41439
Name: predictions, dtype: int64
```

```
df1["predictions"]
```

```
0         0
1         0
2         0
3         0
4         0
         ..
153159    0
153160    1
153161    0
153162    0
153163    0
Name: predictions, Length: 153164, dtype: int32
```

CONCLUSION

- ## Key Findings and Conclusions of the Study

  The key findings, inferences, observations from the whole problem are that I found out how to handle the dataset which is class imbalanced and use for metrics like f1 score. I observed that data with least correlation with target variable do not impact the result.

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

  Visualization is very useful for detection of outliers in particular

  columns, distribution of data, Heat map for correlation and null values. I choose XGBClassifier with hypertune RandomisedSearchCV parameter algorithm as my final model since it was having least difference between its cross-validation score and testing accuracy.

  I faced the challenges in data pre-processing & visualizing all the columns at once when there are some object datatypes were in the columns & special character are present in the columns.

- Limitations of this work and Scope for Future Work

  Limitations of this solutions is that I could have used hyperparameter tuning for selected model or algorithm, but no change in score.

  Future scope of this project is that we can use to predict the malignant comments to online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others comment's to avoid mentally leading to depression, mental illness, self-hatred and suicidal thoughts.