MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

Harneet Kaur rehsi

ACKNOWLEDGMENT

I want to thank my SME Khusboo Garg and Flip Robo Techniques to express our sincere thanks to the following people, without whom we would not have been able to complete this project.

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. . This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred, and suicidal thoughts.

Review of Literature: In this project, Our goal is to build a prototype of online hate and abuse comment classifier which can be used to classify hate and offensive comments so that they can be controlled and restricted from spreading hatred and cyberbullying. These are some columns in our data 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse', and 'Loathe'.

Motivation for the Problem Undertaken: There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlash 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.

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem:** Data contains 159571 rows and 8 columns. columns contain object datatype and int datatype as well. With the help of value_count, we have seen the value each column contains. Then used describe the method to check the health of the dataset. IsNull for checking if data contain some Nan values. used some EDA methods for a better understanding of the data. Then check the skewness of the data. with the help of replace method, I have replaced some data containing addresses, email, etc with meaningful data. And replaced numbers with numbers. Store all the targets in one single column. Convert text into vectors then split the data using train test split and used 4 different classification models.

Data Sources and their formats: Data contains 159571 rows and 8 columns respectively. containing all the necessary details.

	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

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

Data Preprocessing Done:
As data contain some object type data with the help of replace method, I have replaced some data containing addresses, email, etc with meaningful data. And replaced numbers with numbers. Store all the targets in one single column. Convert text into vectors and used all these methods for better model prediction.

State the set of assumptions (if any) related to the problem under consideration:
In this data set we have taken assumption as :

In every columns : 0 = NO , 1 = YES

Hardware and Software Requirements and Tools Used:
These are some libraries I have used for data cleaning , Visualization and model building.

- import pandas as pd
- import numpy as np
- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import warnings
- warnings.filterwarnings('ignore')

- from sklearn.model_selection import train_test_split
- from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
- from sklearn.metrics import roc_curve,roc_auc_score,auc

- from sklearn.linear_model import LogisticRegression
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import AdaBoostClassifier
- import xgboost
- from nltk.stem import WordNetLemmatizer
- import nltk
- from nltk.corpus import stopwords
- import string
- import nltk
- from sklearn.feature_extraction.text import TfidfVectorizer

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods):

- import pandas as pd
- import numpy as np
- import seaborn as sns
- import matplotlib.pyplot as plt
- %matplotlib inline
- import warnings
- warnings.filterwarnings('ignore')
- from nltk.stem import WordNetLemmatizer
- import nltk
- from nltk.corpus import stopwords
- import string
- import nltk
- from sklearn.feature_extraction.text import TfidfVectorizer

Testing of Identified Approaches (Algorithms)

- from sklearn.model_selection import train_test_split
- from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
- from sklearn.metrics import roc_curve,roc_auc_score,auc
- from sklearn.linear_model import LogisticRegression
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import AdaBoostClassifier

- import xgboost

Run and Evaluate selected models :

```
# LogisticRegression

lr = LogisticRegression(C=1, max_iter = 3000)

lr.fit(x_train, y_train)

y_pred_train = lr.predict(x_train)
print('Training accuracy    :',format(accuracy_score(y_train, y_pred_train)))
print('-'*50)
y_pred_test = lr.predict(x_test)
print('Test accuracy        :',format(accuracy_score(y_test,y_pred_test)))
print('-'*50)
print('confusion matrix     :',confusion_matrix(y_test,y_pred_test))
print('-'*50)
print('Classification Report :',classification_report(y_test,y_pred_test))
```

Result :

```
Training accuracy     : 0.960675565582503
--------------------------------------------------
Test accuracy         : 0.9575748080839731
--------------------------------------------------
confusion matrix      : [[28642   172]
 [ 1182  1919]]
--------------------------------------------------
Classification Report :               precision    recall  f1-score   supp
ort

           0       0.96      0.99      0.98     28814
           1       0.92      0.62      0.74      3101

    accuracy                           0.96     31915
   macro avg       0.94      0.81      0.86     31915
weighted avg       0.96      0.96      0.95     31915
```

```
#AdaBoostClassifier

ada =AdaBoostClassifier(n_estimators=100)

ada.fit(x_train, y_train)

y_pred_train = ada.predict(x_train)
print('Training accuracy     :',format(accuracy_score(y_train, y_pred_train)))
print('-'*50)
y_pred_test = ada.predict(x_test)
print('Test accuracy         :',format(accuracy_score(y_test,y_pred_test)))
print('-'*50)
print('confusion matrix      :',confusion_matrix(y_test,y_pred_test))
print('-'*50)
print('Classification Report :',classification_report(y_test,y_pred_test))
```

Result:

```
Training accuracy     : 0.9513928056652253
--------------------------------------------------
Test accuracy         : 0.9501174996083347
--------------------------------------------------
confusion matrix      : [[28565   249]
 [ 1343  1758]]
--------------------------------------------------
Classification Report :               precision    recall  f1-score   supp
ort

           0       0.96      0.99      0.97     28814
           1       0.88      0.57      0.69      3101

    accuracy                           0.95     31915
   macro avg       0.92      0.78      0.83     31915
weighted avg       0.95      0.95      0.95     31915
```

```
# xgboost

xgb = xgboost.XGBClassifier()

xgb.fit(x_train, y_train)

y_pred_train = xgb.predict(x_train)
print('Training accuracy     :',format(accuracy_score(y_train, y_pred_train)))
print('-'*50)
y_pred_test = xgb.predict(x_test)
print('Test accuracy         :',format(accuracy_score(y_test,y_pred_test)))
print('-'*50)
print('confusion matrix      :',confusion_matrix(y_test,y_pred_test))
print('-'*50)
print('Classification Report :',classification_report(y_test,y_pred_test))
```

Result :

```
Training accuracy     : 0.9613100833490005
--------------------------------------------------
Test accuracy         : 0.9544414851950493
--------------------------------------------------
confusion matrix      : [[28637   177]
 [ 1277  1824]]
--------------------------------------------------
Classification Report :               precision    recall  f1-score   supp
ort

           0       0.96      0.99      0.98     28814
           1       0.91      0.59      0.72      3101

    accuracy                           0.95     31915
   macro avg       0.93      0.79      0.85     31915
weighted avg       0.95      0.95      0.95     31915
```

DecisionTreeClassifier

dt = DecisionTreeClassifier()

dt.fit(x_train, y_train)

y_pred_train = dt.predict(x_train)

print('Training accuracy   :',format(accuracy_score(y_train, y_pred_train)))

print('-'*50)

y_pred_test = dt.predict(x_test)

print('Test accuracy   :',format(accuracy_score(y_test,y_pred_test)))

print('-'*50)

print('confusion matrix   :',confusion_matrix(y_test,y_pred_test))

print('-'*50)

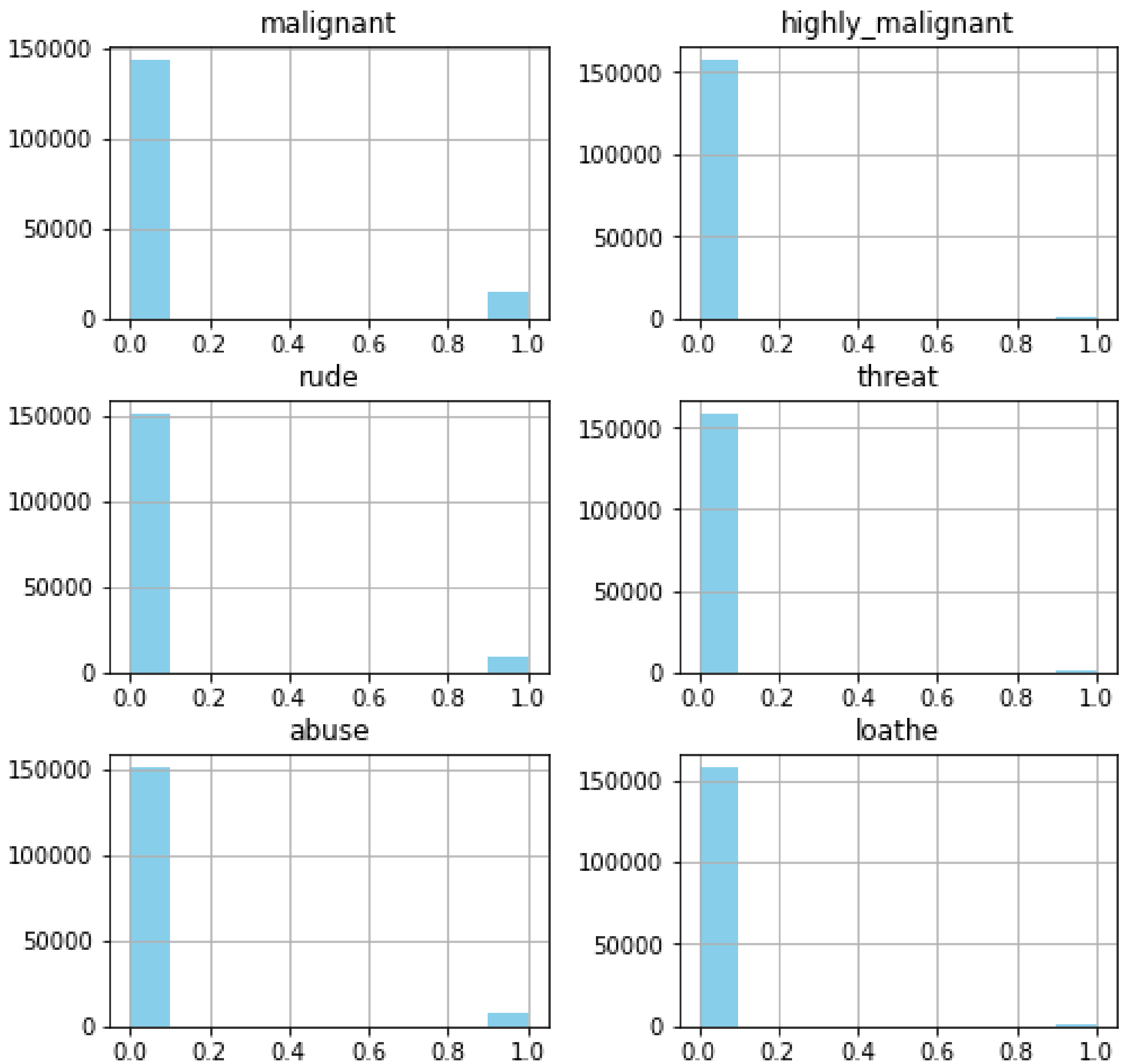
print('Classification Report :',classification_report(y_test,y_pred_test))

Result:

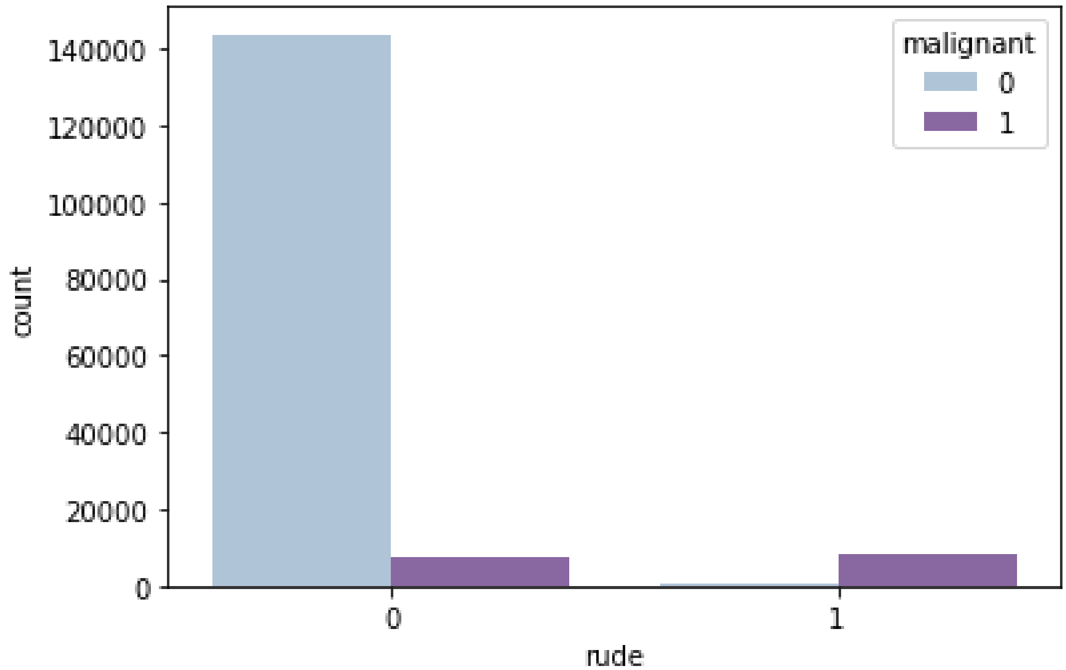
```
Training accuracy     : 0.9987152973616594
--------------------------------------------------
Test accuracy         : 0.9399028669904433
--------------------------------------------------
confusion matrix      : [[27902   912]
 [ 1006  2095]]
--------------------------------------------------
Classification Report :               precision    recall  f1-score   supp
ort

           0       0.97      0.97      0.97     28814
           1       0.70      0.68      0.69      3101

    accuracy                           0.94     31915
   macro avg       0.83      0.82      0.83     31915
weighted avg       0.94      0.94      0.94     31915
```

Logistic Regression fits best among all other algorithms
As Logistic Regression is giving: 0.97 accuracy

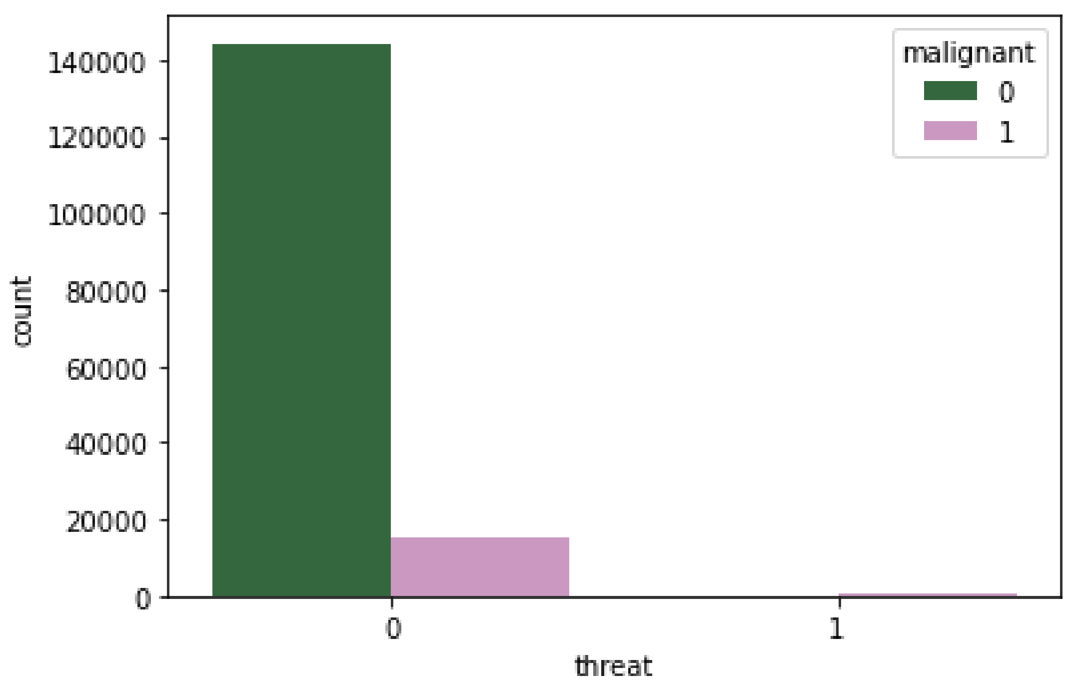
Visualizations



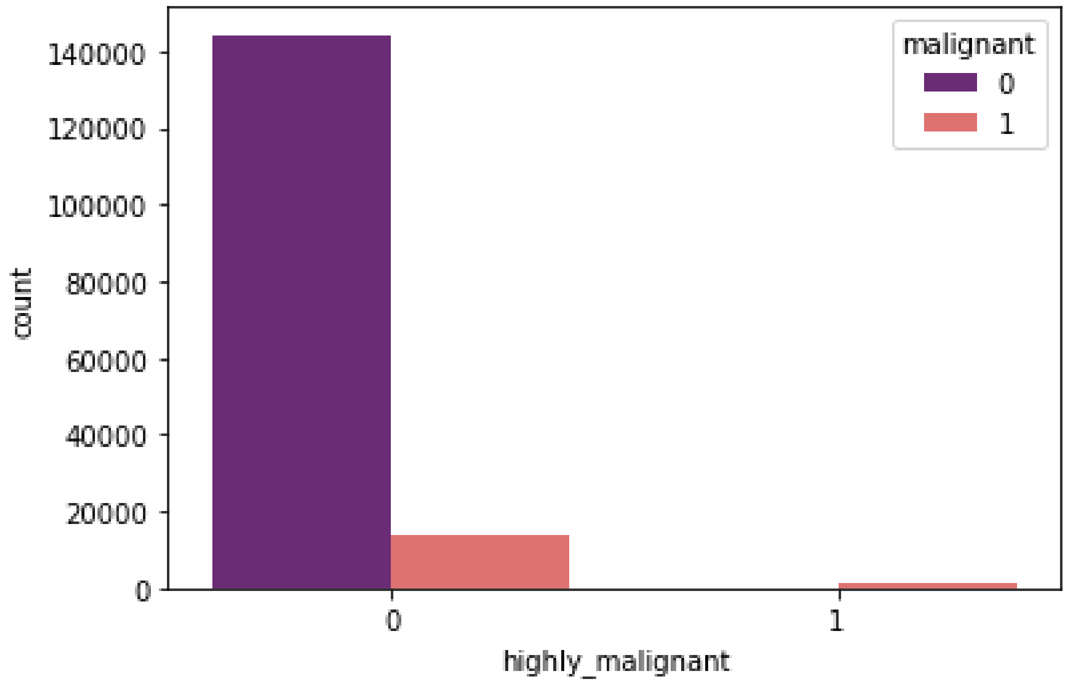
Applied Histogram on the dataset



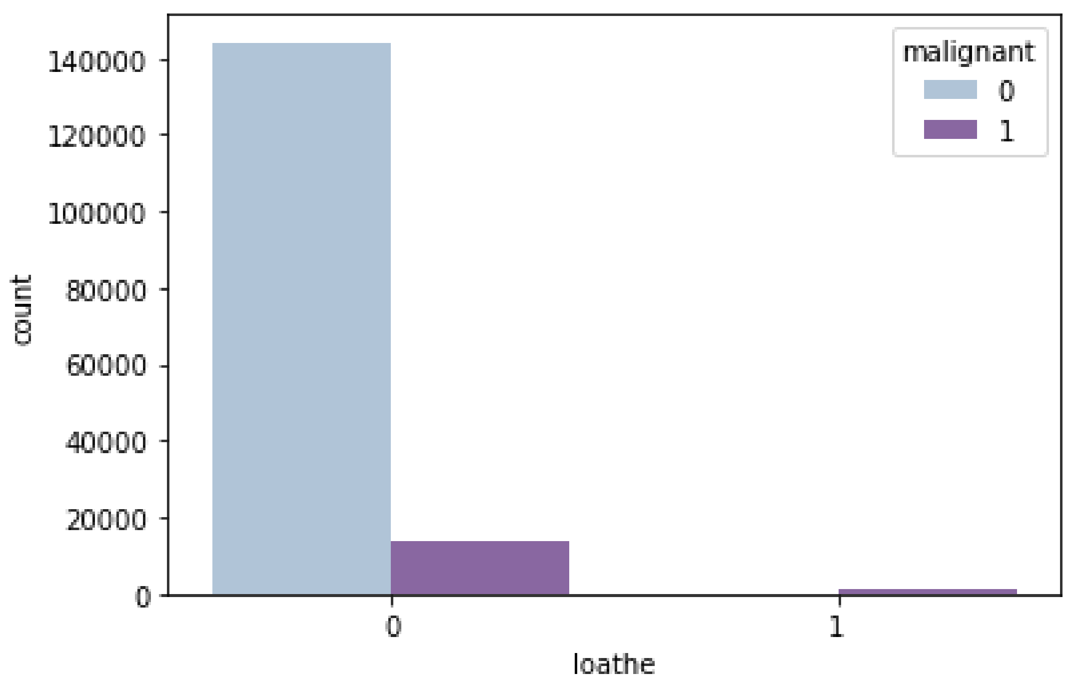
Used Countplot for checking Malignant and rude column dataset



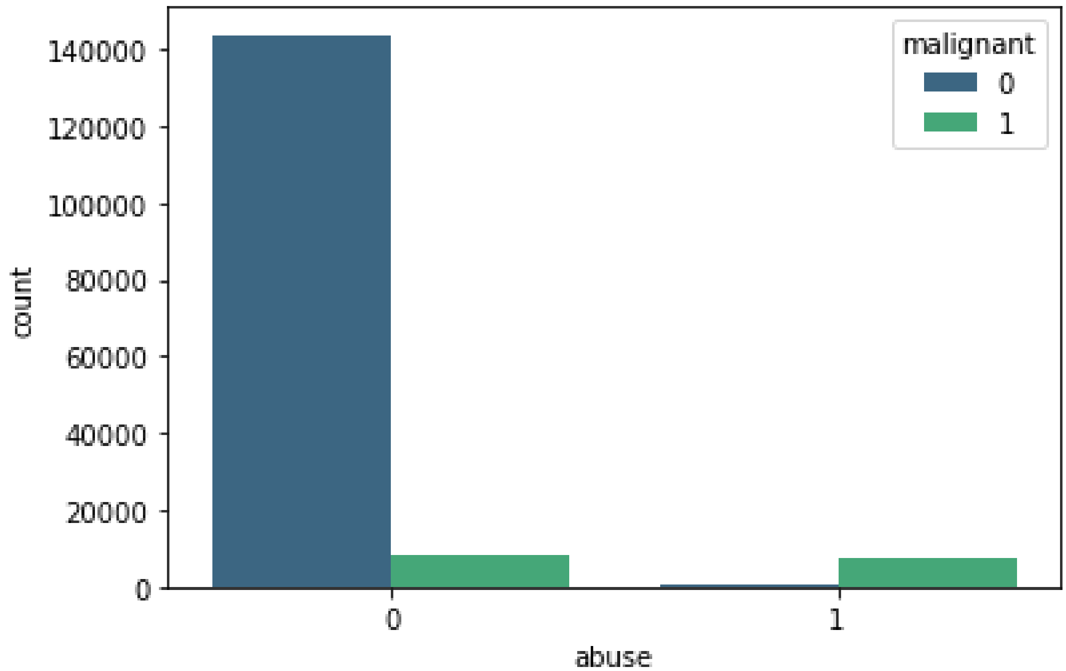
Used Countplot for checking Malignant and threat column dataset



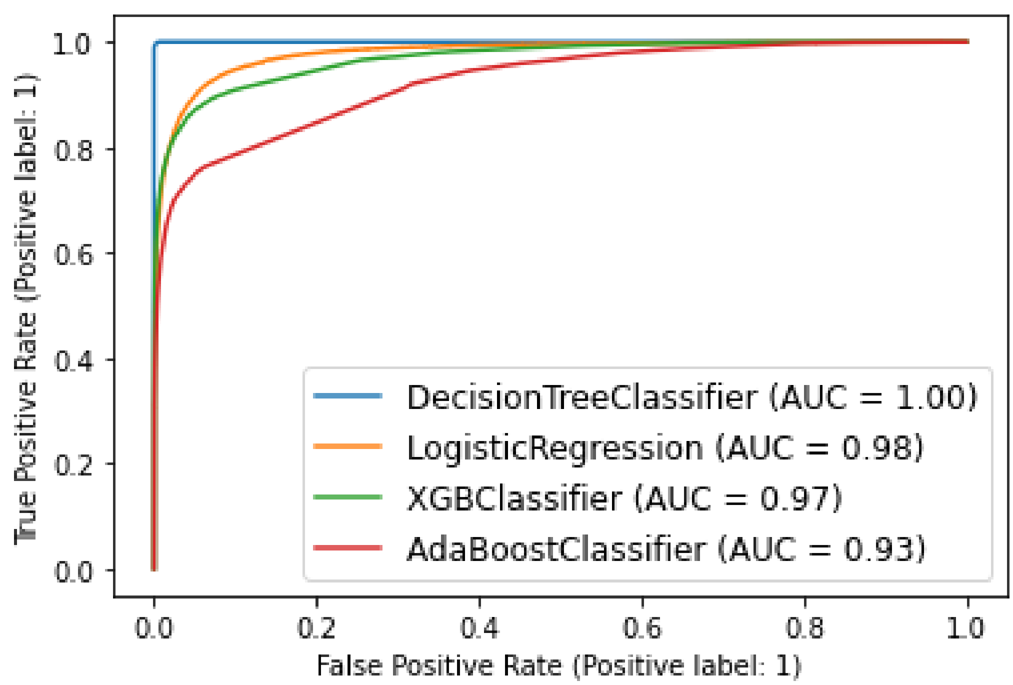
Used Countplot for checking Malignant and highly malignant column dataset



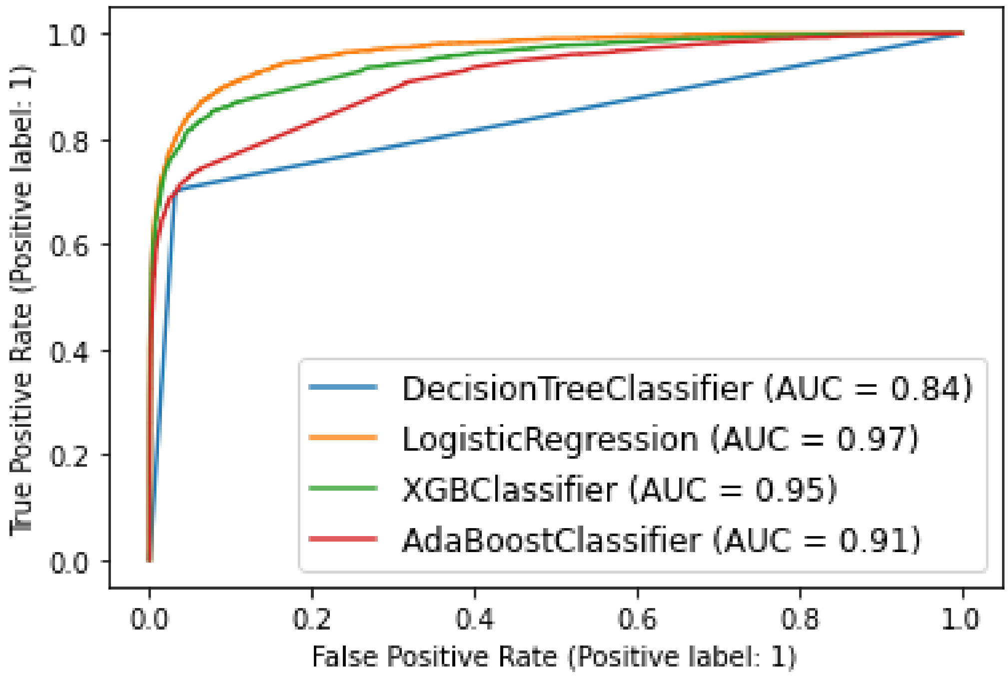
Used Countplot for checking Malignant and loathe column dataset



Used Countplot for checking Malignant and abuse column dataset



Checking the score of different models on training data that which model fit best



Checking the score of different models on testing data that which model fit best

Logistic Regression fits best among all other models
As Logistic Regression is given:

0.97 score while testing
0.98 score while training

Interpretation of the Results:: After visualizing the data I have concluded that under all these columns :

- **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.

As value of 0 as No are more than value of 1 as Yes in these columns. Majority of these texts 0 that is are Not malignant. Logistic Regression fits best among all other models
As Logistic Regression is given:
0.97 score while testing
0.98 score while training

CONCLUSION

Key Findings and Conclusions of the Study :

As value of 0 as No are more than value of 1 as Yes in these columns. Majority of these texts 0 that is are Not malignant. Logistic Regression fits best among all other models As Logistic Regression is given:
0.97 score while testing
0.98 score while training

Learning Outcomes of the Study in respect of Data Science:

As data contain some object type data with the help of replace method, I have replaced some data containing addresses, email, etc with meaningful data. And replaced numbers with numbers. Store all the targets in one single column. Convert text into vectors and used all these methods for better model prediction. As value of 0 as No are more than value of 1 as Yes in these columns. Majority of these texts 0 that is are Not malignant. Logistic Regression fits best among all other models As Logistic Regression is given:
0.97 score while testing
0.98 score while training