FLIP ROBO

MALIGNANT COMMENTS CLASSIFICATION

Submitted By-
Akanksha Amarnani

Acknowledgement

I would like to thank Almighty for giving me the confidence to pursue this project. Further, the concepts from DataTrained Academy guided me to complete the project.
In addition I would like to thank my mentor from Flip Robo Technology, Ms Khushboo Garg for clarifying my doubts and queries.

The references used for the completion of this project are-

- Malignant Comment Classification; Kaggle
- Toxic Comment Classification; Nupur Baghel; Medium.com

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 to build a model using Machine Learning in order 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-

The domain related concepts which help us in a better understanding are-

- Exploratory Data Analysis (EDA)- By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting. In addition in this scenario features can be added to the test dataset b analysing the train dataset

- Modeling- We apply Logistic Regression, KNeighbor Classifier, Decision Tree Classifier, Random Forest Classifier, Adaboost Classifier, Gradient Boosting Classifier and SVC to check if is the comment is malignant or not

- Regularization- Models are regularized and the parameters are hypertuned to enhance the efficiency of the models

Review of Literature-

Machine learning is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to extract useful information from data. Machine learning methods are appropriate in big data since attempting to manually process vast volumes of data would be impossible without the support of machines. Machine learning in computer science attempts to solve problems algorithmically rather than purely mathematically. Therefore, it is based on creating algorithms that permit the machine to learn. However, there are two general groups in machine learning which are supervised and unsupervised. Supervised is where the program gets trained on pre-determined set to be able to predict when a new data is given. Unsupervised is where the program tries to find the relationship and the hidden pattern between the data

The performance of the model build will be measured upon its accuracy to determine whether a comment is mlaignant or not, whereby, Label '1' indicates that the comment is malignant, while, Label '0' indicates that the comment is non-malignant. The features assessed will be such to determine which comment is categorized as highly malignant, rude, threat, abuse and loathe in the train dataset, and accordingly categorize the comments in the test dataset. This shall add features to the train dataset allowing the classification models a larger depth and uniformity to predict the malignant comments, as they would already had been trained on the train dataset. We implement and evaluate various learning methods on the provided dataset. However, proper EDA is to be kept in mind to avoid unbiased predictions. The data used in the experiment will be handled by using a combination of pre-processing methods to improve the prediction accuracy

Motivation for the Problem Undertaken-

The project involves classification and filtering malignant comments over the non-malignant ones. It also specifies whether the comment is highly malignant, rude, threat, abusive and loathe. Offensive comments and trolling has become a major part of cyber bullying. In the face of anonymity, people tend to demotivate and abuse people without any reason and background. Such offensive comments pollutes the vibe of social media as well as takes the content creators a toll on their mental health as well. This project is important not only in terms of enhancing ones skills in building machine learning models, but more so, as one’s morale responsibility towards the society. We all are responsible for the environment we create physically as well as digitally. Hence if a group of immature people are causing dirt to it by their trolling and offensive, rude comments, it is the other’s duty to clean up the dirt as well as create a blockage for it.

EDA Steps and Visualization

- The train datasheet is extracted and saved in a dataframe
- The shape of the dataframe is checked-

  There are 159571 rows and 8 columns

- **The columns are as follows-**
  - id
  - comment_text
  - malignant
  - highly_malignant
  - rude
  - threat
  - abuse
  - loathe

- The data type of each column is-
  - id - object
  - comment_text - object
  - malignant - int64
  - highly_malignant - int64
  - rude - int64
  - threat - int64
  - abuse - int64
  - loathe - int64
- There are no null values are present in the dataset

As the id is unique to all, its safe to drop this column

The data visualization, value counts encoding and imputation of null values for each column

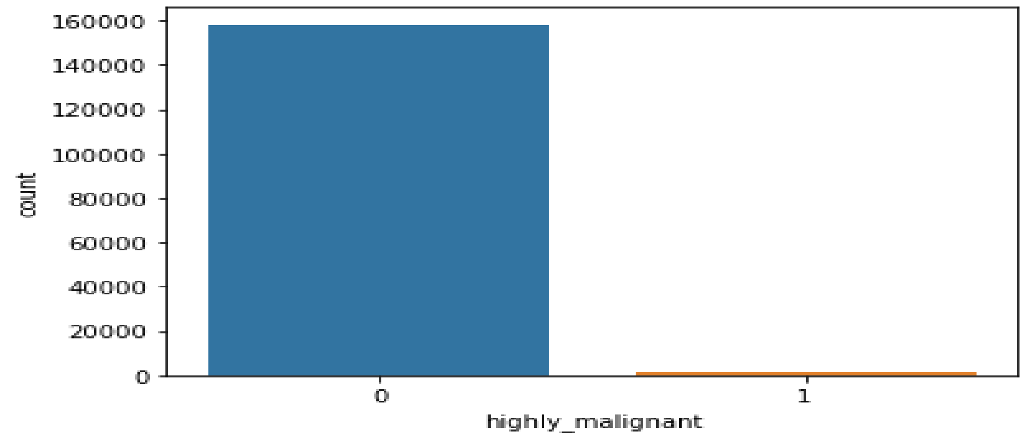
comment_text

It includes the comment text.

Encoding object data in numeric using Label Encoder

highly_malignant

Binary column with labels for highly malignant text..



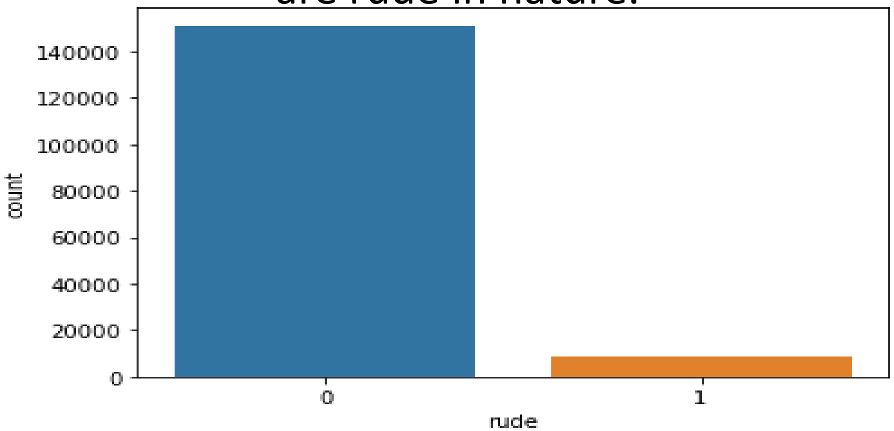
- 1595 comments are highly malignant

```
-----------Highly Malignant------------
COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK
---------------------------------------
-----------Highly Malignant------------
Stupid peace of shit stop deleting my stuff asshole go die and fall in a hole go to hell!
---------------------------------------
-----------Highly Malignant------------
you are a stupid fuck

and your mother's cunt stinks
---------------------------------------
-----------Highly Malignant------------
Hi

Im a fucking bitch.

50.180.208.181
---------------------------------------
```

- Words like fuck, fucking, cocksucker, fucker, motherfucking, motherfucker, suck, dick, ass, scum, asshole, bitch, kill, are most common in highly malignant comments

rude

Binary column with labels for comments that are rude in nature.



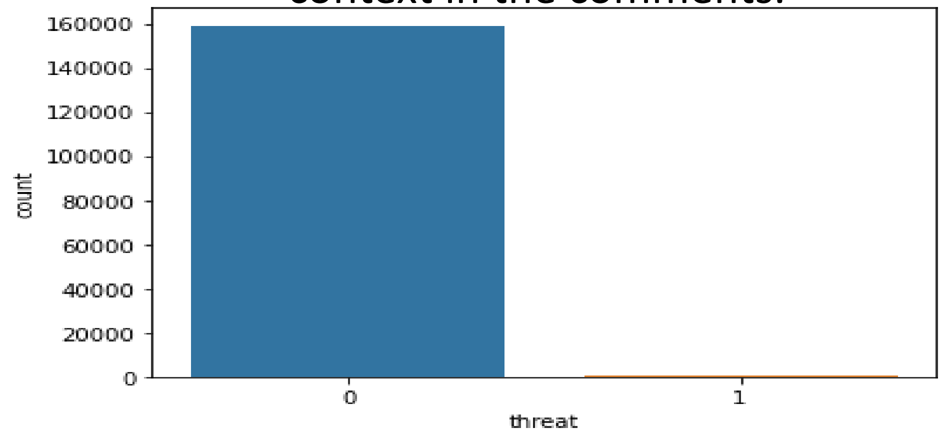
- 8449 comments are rude in nature

```
-----------------Rude-----------------
COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK
--------------------------------------
-----------------Rude-----------------
You are gay or antisemmitian?

Archangel WHite Tiger

Meow! Greetingshhh!

Uh, there are two ways, why you do erased my comment about WW2, that holocaust was brutally slaying of Jews and not gays/Gyps
ys/Slavs/anyone...

1 - If you are anti-semitian, than shave your head bald and go to the skinhead meetings!

2 - If you doubt words of the Bible, that homosexuality is a deadly sin, make a pentagram tatoo on your forehead go to the sa
tanistic masses with your gay pals!
```

- Words like cocksucker, gay, idiot, die, antisemmitian, motherfucker, fuck, fucking, hell, fucked, fucker, ass, shit, hell, dick, scum, asshole, bitch, schmucks, racists, dead, vandalism are some common words in rude comments

threat

Binary column with labels for threatening context in the comments.



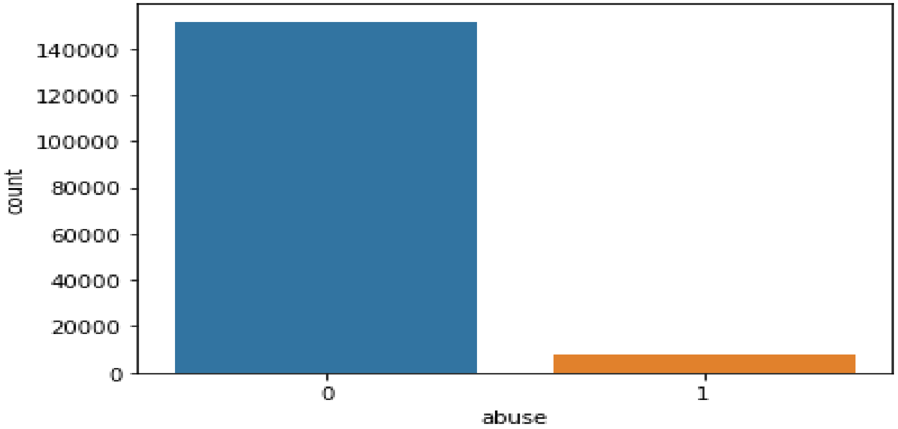
- 478 comments are threatening

```
-----------------Threat-----------------
Hi! I am back again!
Last warning!
Stop undoing my edits or die!
--------------------------------------
-----------------Threat-----------------
I think that your a Fagget get a oife and burn in Hell I hate you 'm sorry we cant have any more sex i'm running out of connd
oms
--------------------------------------
-----------------Threat-----------------
I'm also a sock puppet of this account...SUPRISE!!
-sincerely,
            The man that will track you down from the Internet and kill you
--------------------------------------
-----------------Threat-----------------
Fuck you, Smith. Please have me notified when you die. I want to dance on your grave.
--------------------------------------
```

- Words like die, kill, fuck, grave, bitch, kick, ass, burn, bitch, fucking are common words in threat comments

abuse

Binary column with labels with abusive behaviour.



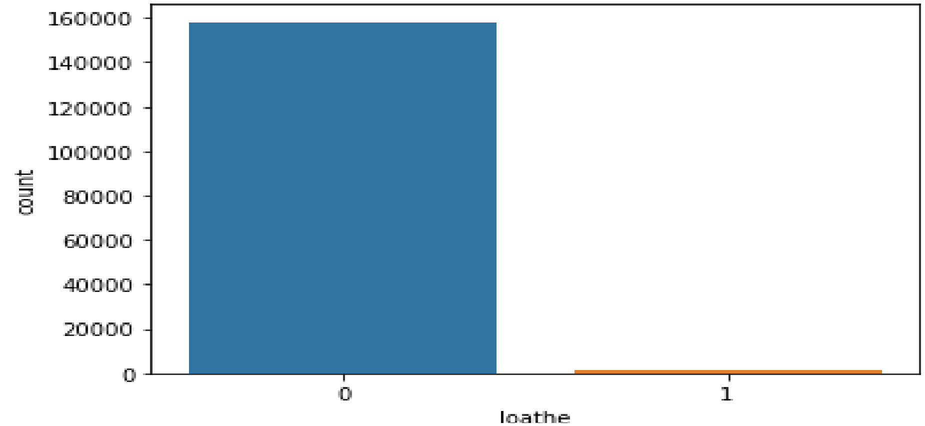
- 7877 comments are abusive

```
-----------------Abuse-----------------
COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK
----------------------------------------
-----------------Abuse-----------------
You are gay or antisemmitian?

Archangel WHite Tiger

Meow! Greetingshhh!

Uh, there are two ways, why you do erased my comment about WW2, that holocaust was brutally slaying of Jews and not gays/Gyps
ys/Slavs/anyone...

1 - If you are anti-semitian, than shave your head bald and go to the skinhead meetings!

2 - If you doubt words of the Bible, that homosexuality is a deadly sin, make a pentagram tatoo on your forehead go to the sa
tanistic masses with your gay pals!
```

- Words like gay, antisemmitian, cocksucker, fuck, ass, cunts, schmuck, kill, asshole, idiot are common words in abuse comments

loathe

Label to comments that are full of loathe and hatred.



- 1405 comments are full of loathe and hatred

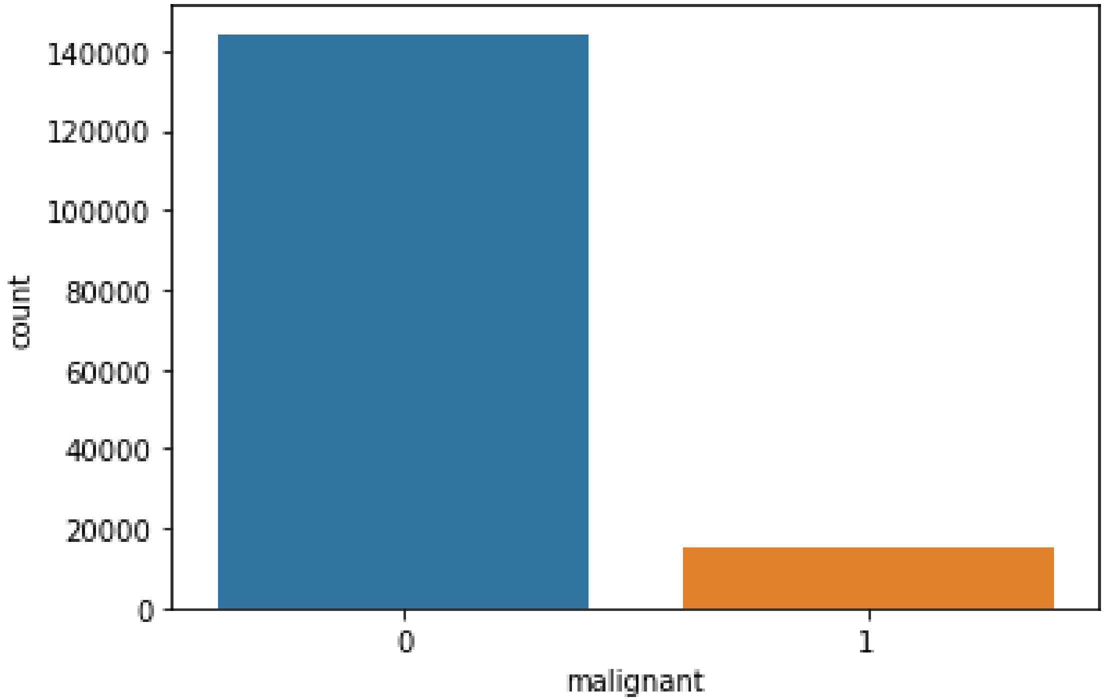
```
-----------------Loathe-----------------
You are gay or antisemmitian?

Archangel WHite Tiger

Meow! Greetingshhh!

Uh, there are two ways, why you do erased my comment about WW2, that holocaust was brutally slaying of Jews and not gays/Gyps
ys/Slavs/anyone...

1 - If you are anti-semitian, than shave your head bald and go to the skinhead meetings!

2 - If you doubt words of the Bible, that homosexuality is a deadly sin, make a pentagram tatoo on your forehead go to the sa
tanistic masses with your gay pals!

3 - First and last warning, you fucking gay - I won't appreciate if any more nazi shwain would write in my page! I don't wish
to talk to you anymore!
```

- Words like antisemmitian, schmucks, kill, fuck, gay, cock, bitch, ass, bastard, fucking are common words in loathe comments

Label

malignant

It is a column with binary values depicting which comments are malignant in nature.

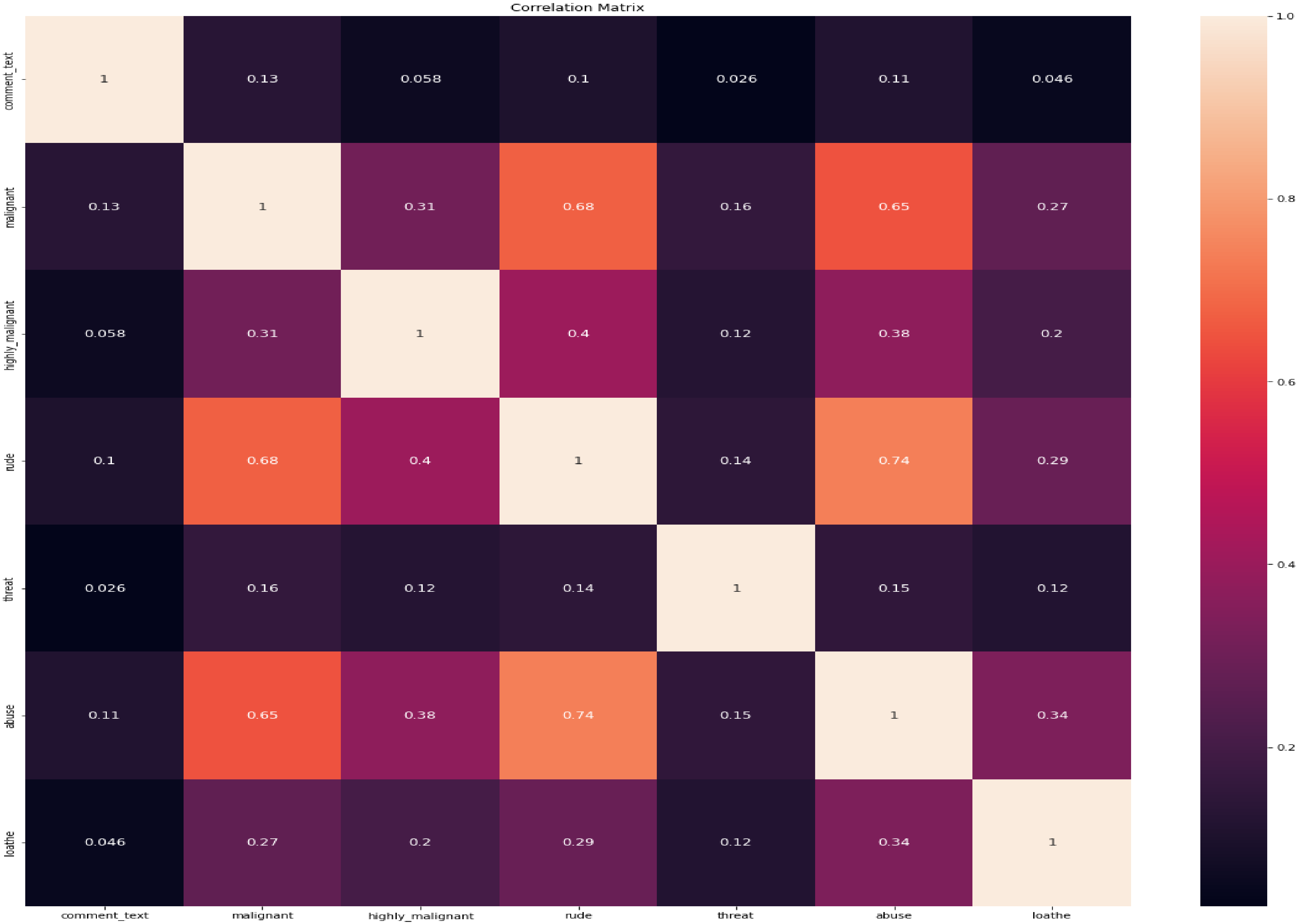


- 15294 comments are malignant in nature

- Statistical analysis using describe method-

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.00000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	79785.00000	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	46064.32424	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	39892.50000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	79785.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	119677.50000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	159570.00000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

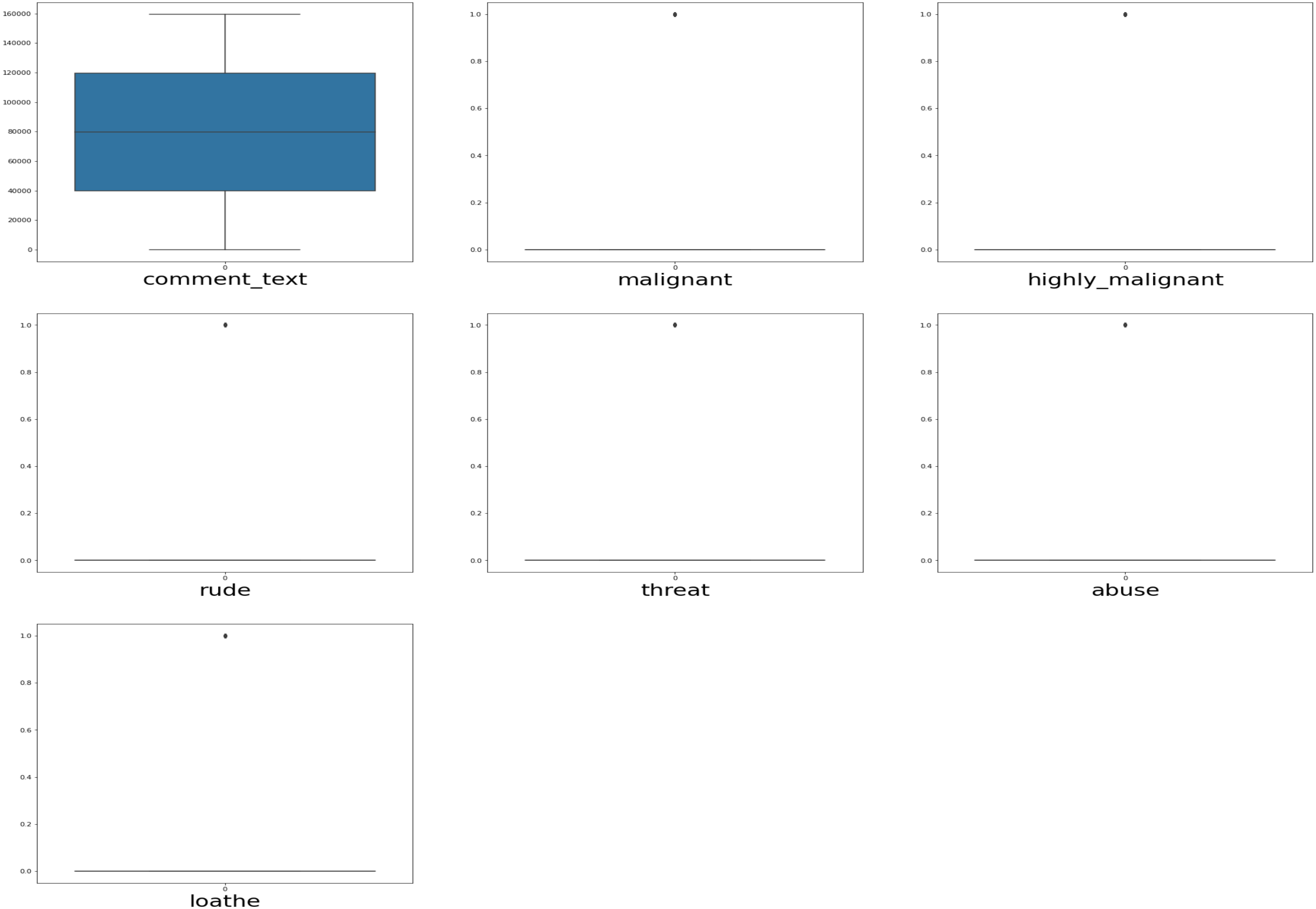
The Correlation Matrix using heatmap



- Correlation between the columns and the label ‘malignant’ using corr method-
    - malignant : 1.000000
    - rude : 0.676515
    - abuse : 0.647518
    - highly_malignant : 0.308619
    - loathe : 0.266009
    - threat : 0.157058
    - comment_text : 0.132016

- rude is 67% positively correlated to ‘malignant’
- abuse is 64% positively correlated to ‘malignant’
- highly_malignant is 30% positively correlated to ‘malignant’
- loathe is 26% positively correlated to ‘malignant’
- threat is 15% positively correlated to ‘malignant’
- comment_text is 13% positively correlated to ‘malignant’

- Visualizing outliers using boxplot method-



- Removing outliers using zscore method-

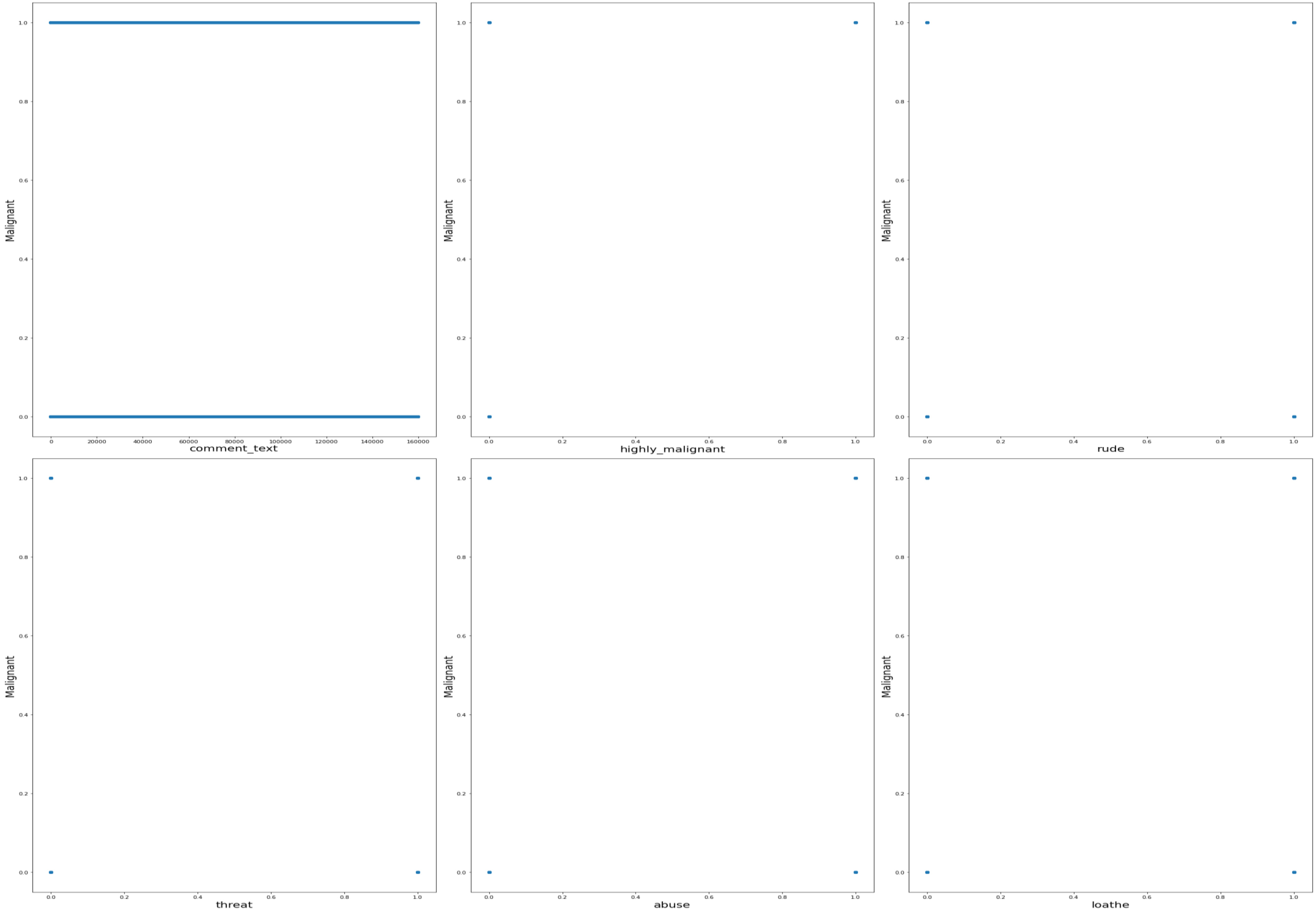
On removing the outliers the data loss is 10.17%, which is not acceptable, hence outliers are tolerated

- The dataset is divided into x_train (features) and y_train (label)-

The x_train contains all the features other than the label ‘malignant’

The y_train contains only the label ‘malignant’

• Visualizing relationship between features and label-



- The skewness observes in graphical analysis was confirmed by using the skew method-
    - threat : 1.818900e+01
    - loathe : 1.051592e+01
    - highly_malignant : 9.851722e+00
    - abuse : 4.160540e+00
    - rude : 3.992817e+00
    - comment_text : 1.282301e-19

- This skewness was removed using the power transformer

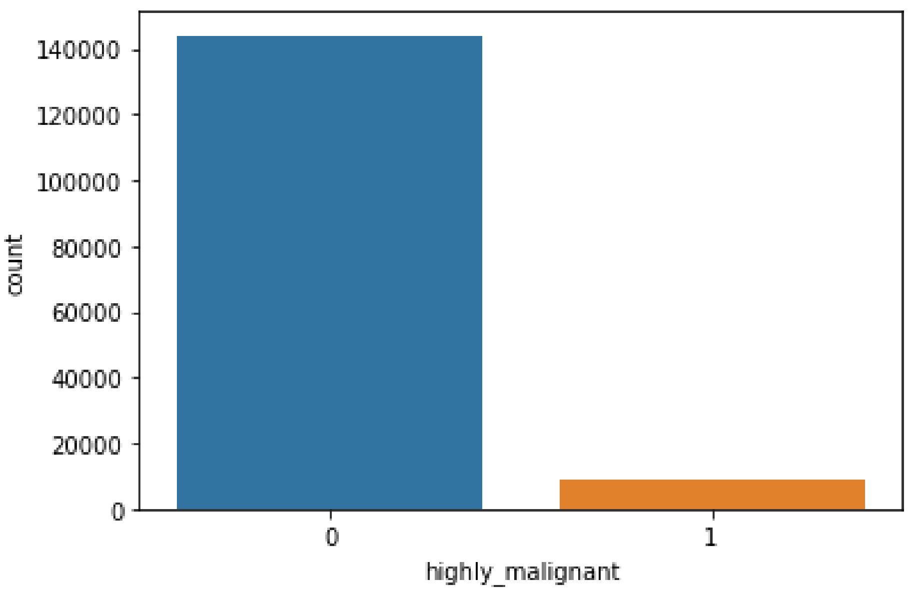
- The x_train(features) were scaled using the Standard Scaler

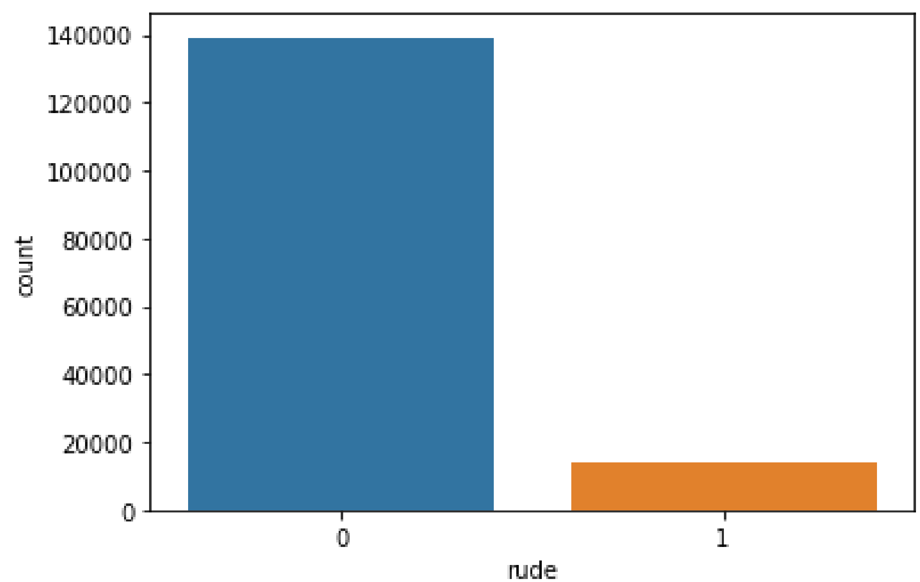
- The test datasheet and saved in another dataframe

- The shape of x_test is 153164 rows and 2 columns

highly malignant column is added by filtering comments containing the words common in highly malignant comments in the train dataset

rude column is added by filtering comments containing the words common in rude comments in the train dataset

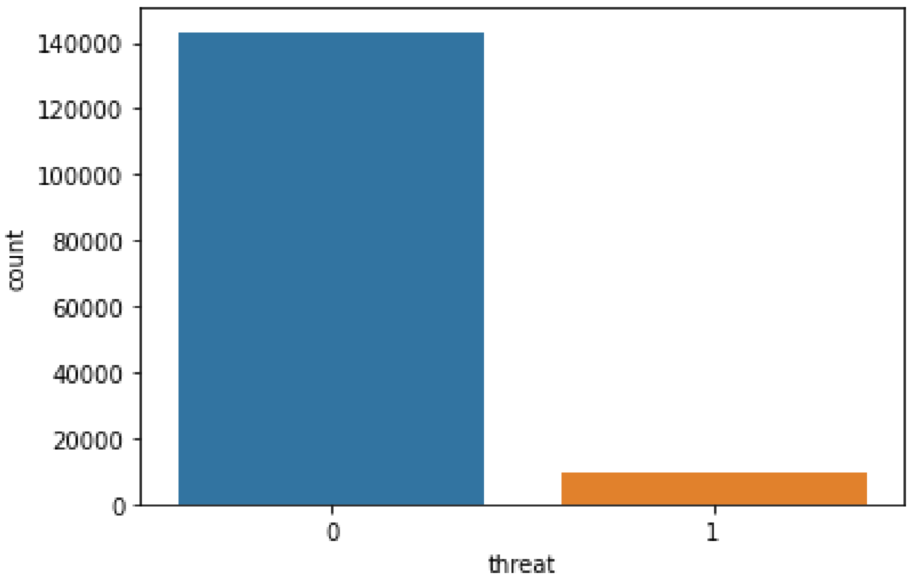




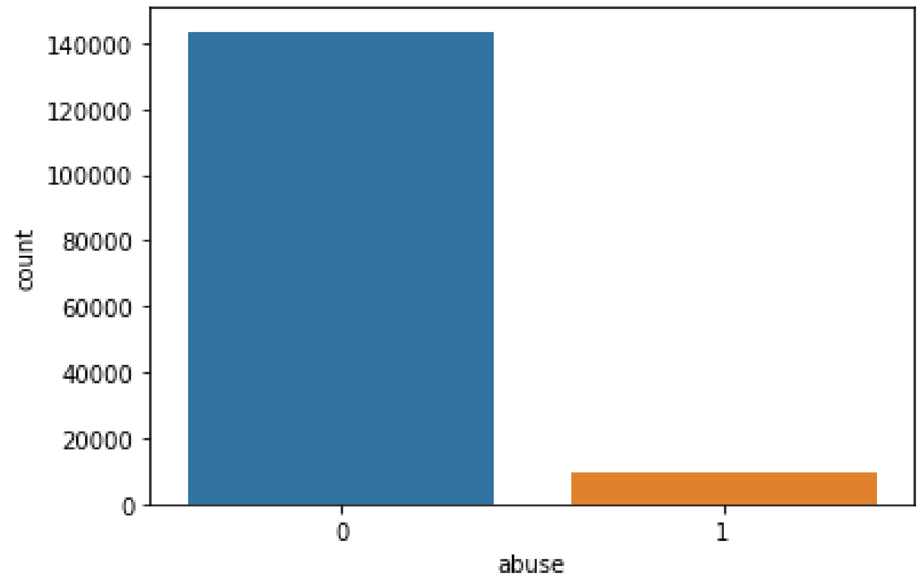
- 9111 comments are highly malignant
- 13876 comments are rude in nature

threat column is added by filtering comments containing the words common in threat comments in the train dataset

abuse column is added by filtering comments containing the words common in abuse comments in the train dataset

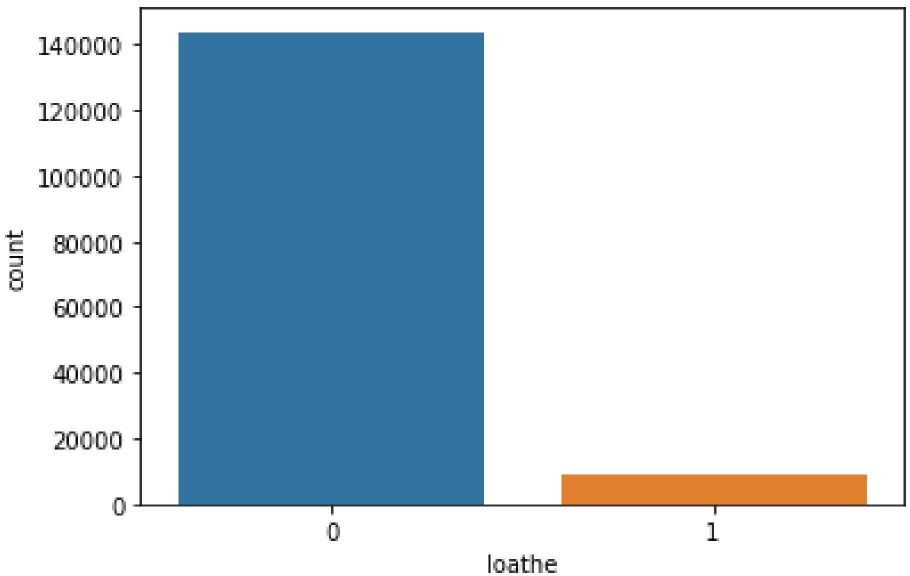


- 9851 comments are threatening



- 9510 comments are abusive in nature

loathe column is added by filtering comments containing the words common in loathe comments in the train dataset



- 9217 comments are full of hatred and loathe

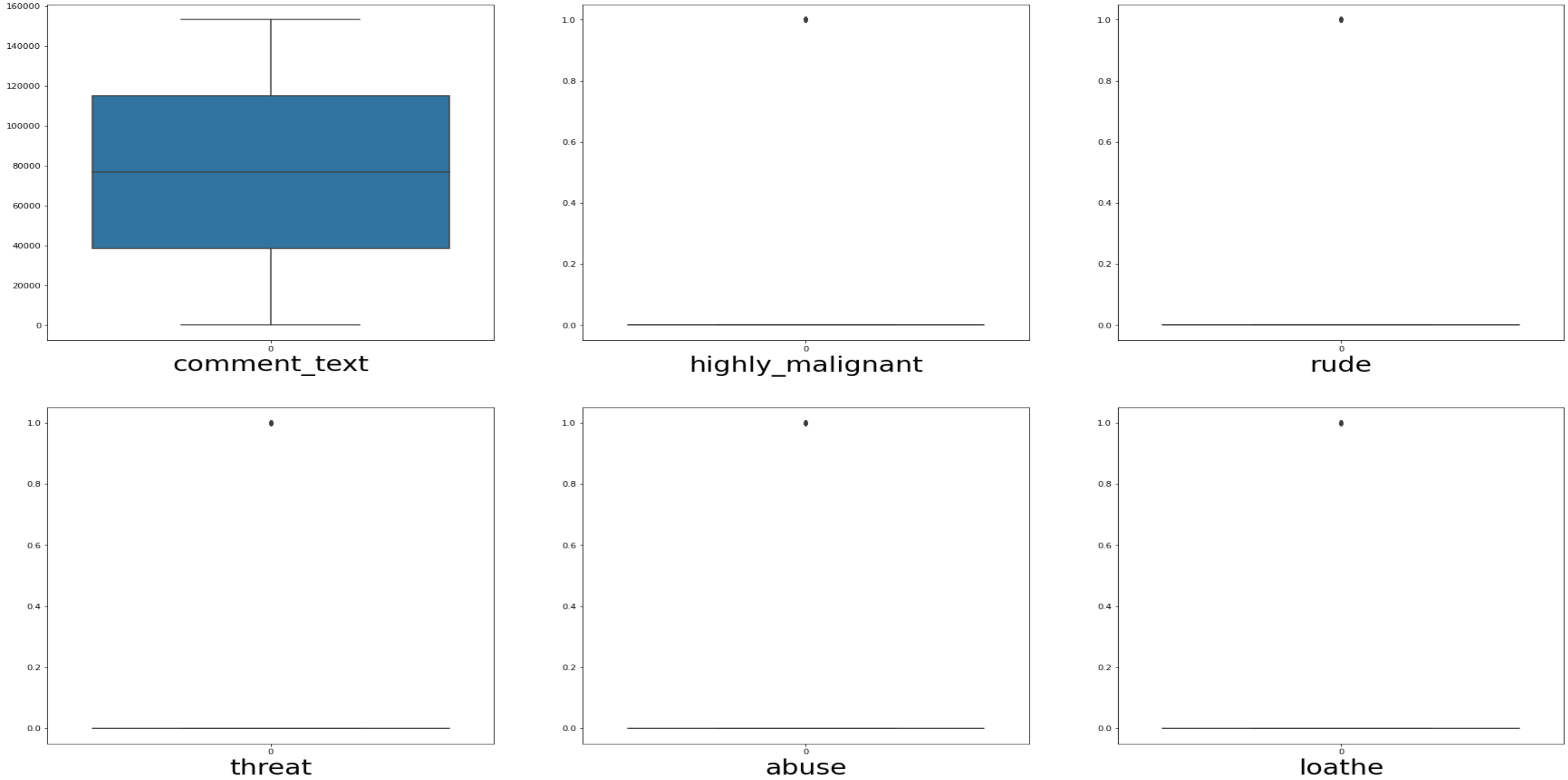
comment_text

It includes the comment text.

Encoding object data in numeric using Label Encoder

As the id is unique to all, its safe to drop this column

- Visualizing outliers using boxplot method-



- Removing outliers using zscore method-

On removing the outliers the data loss is 10.27%, which is not acceptable, hence outliers are tolerated

- The skewness observes in graphical analysis was confirmed by using the skew method-
  - highly_malignant : 3.724837e+00
  - loathe : 3.698899e+00
  - abuse : 3.629329e+00
  - threat : 3.552049e+00
  - rude : 2.852688e+00
  - comment_text : 1.510704e-18

- This skewness was removed using the power transformer

- The x_test were scaled using the Standard Scaler

Software Requirements-

- Jupyter Notebook – Interface for the program
- Pandas – for datafram working
- Numpy – to deal with null data
- matplotlib.pyplot – for data visualization
- Seaborn - for data visualization
- Warnings- to omit warnings
- sklearn.preprocessing – to import powertransform
- scipy.stats – to import zscore
- Zscore- to remove outliers
- power_transform- to remove the skewness in the data
- sklearn.linear_model- to import Logistic Regression model,
- Logistic Regression – to use Logistic Regression model
- sklearn.metrics- to import accuracy, confusion matric and classification report, plot_roc_curve
- sklearn.neighbors- to import Kneighbours CLassifierModel
- Kneighbours-Classifier to use Kneighbours model
- sklearn.tree- to import DecisionTreeClassifier
- DecisionTreeClassifier- to use DecisionTreeClassifier Model
- sklearn.ensemble – to import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
- RandomForestClassifier- to use RandomForestClassifier model
- AdaBoostClassifier- to use AdaBoostClassifier model
- GradientBoostingClassifier- to use GradientBoostingClassifier model
- sklearn.svm – to import SVC
- SVC- to use SVC model
- sklearn.model_selection- to import cross_val_score
- cross_val_score- to check for overfitting and underfitting
- sklearn.model_selection- to import GridSearchCV
- GridSearchCV to enhance the working of the model by manipulating the parameters
- plot_roc_curve- to plot ROC_AUC plot

Model/s Development and Evaluation

The x_train, y_train and x_test were applied on different models as follows

Logistic Regression Model

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

LR=LogisticRegression()
LR.fit(x_train, y_train)
predlr=LR.predict(x_train)
print("Accuracy ",accuracy_score(y_train, predlr)*100)      #accuracy score
print(confusion_matrix(y_train,predlr))
print(classification_report(y_train,predlr))
```

```
Accuracy  95.83006937350773
[[143390    887]
 [  5767   9527]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98    144277
           1       0.91      0.62      0.74     15294

    accuracy                           0.96    159571
   macro avg       0.94      0.81      0.86    159571
weighted avg       0.96      0.96      0.95    159571
```

- The **Accuracy** for y_train and pred_train(data predicted on x_train) is **95.83%**

- The **Confusion Matrix** for y_train and pred_train(data predicted on x_train) is –

	True Positive	False Positive	
	143390	**887**	
False Negative	**5767**	**9527**	True Negative

- The **Classification Report** for y_train and pred_train(data predicted on x_train) is

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98    144277
           1       0.91      0.62      0.74     15294

    accuracy                           0.96    159571
   macro avg       0.94      0.81      0.86    159571
weighted avg       0.96      0.96      0.95    159571
```

KNeighbors Classifier Model

```
from sklearn.neighbors import KNeighborsClassifier

kn=KNeighborsClassifier()
kn.fit(x_train, y_train)
predkn=kn.predict(x_train)
print("Accuracy ",accuracy_score(y_train, predkn)*100)     #accuracy score
print(confusion_matrix(y_train,predkn))
print(classification_report(y_train,predkn))
```

```
Accuracy  96.10894210100834
[[143309    968]
 [  5241  10053]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98    144277
           1       0.91      0.66      0.76     15294

    accuracy                           0.96    159571
   macro avg       0.94      0.83      0.87    159571
weighted avg       0.96      0.96      0.96    159571
```

- The **Accuracy** for y_train and pred_train(data predicted on x_train) is **96.11%**

- The **Confusion Matrix** for y_train and pred_train(data predicted on x_train) is –

	True Positive	False Positive	
	143309	**968**	
False Negative	**5241**	**10053**	True Negative

- The **Classification Report** for y_train and pred_train(data predicted on x_train) is

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98    144277
           1       0.91      0.66      0.76     15294

    accuracy                           0.96    159571
   macro avg       0.94      0.83      0.87    159571
weighted avg       0.96      0.96      0.96    159571
```

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()
dt.fit(x_train, y_train)
preddt=dt.predict(x_train)
print("Accuracy ",accuracy_score(y_train, preddt)*100)     #accuracy score
print(confusion_matrix(y_train,preddt))
print(classification_report(y_train,preddt))
```

```
Accuracy  100.0
[[144277      0]
 [     0  15294]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    144277
           1       1.00      1.00      1.00     15294

    accuracy                           1.00    159571
   macro avg       1.00      1.00      1.00    159571
weighted avg       1.00      1.00      1.00    159571
```

- The **Accuracy** for y_train and pred_train(data predicted on x_train) is **100%**

- The **Confusion Matrix** for y_train and pred_train(data predicted on x_train) is –

	True Positive	False Positive	
	144277	**0**	
False Negative	**0**	**15294**	True Negative

- The **Classification Report** for y_train and pred_train(data predicted on x_train) is

	precision	recall	f1-score	support
0	1.00	1.00	1.00	144277
1	1.00	1.00	1.00	15294
accuracy			1.00	159571
macro avg	1.00	1.00	1.00	159571
weighted avg	1.00	1.00	1.00	159571

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()
rf.fit(x_train, y_train)
predrf=rf.predict(x_train)
print("Accuracy ",accuracy_score(y_train, predrf)*100)     #accuracy score
print(confusion_matrix(y_train,predrf))
print(classification_report(y_train,predrf))
```

```
Accuracy  99.99122647598875
[[144276      1]
 [    13  15281]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    144277
           1       1.00      1.00      1.00     15294

    accuracy                           1.00    159571
   macro avg       1.00      1.00      1.00    159571
weighted avg       1.00      1.00      1.00    159571
```

- The **Accuracy** for y_train and pred_train(data predicted on x_train) is **99.99%**

- The **Confusion Matrix** for y_train and pred_train(data predicted on x_train) is –

	True Positive	False Positive	
	144276	**1**	
False Negative	**13**	**15281**	True Negative

- The **Classification Report** for y_train and pred_train(data predicted on x_train) is

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    144277
           1       1.00      1.00      1.00     15294

    accuracy                           1.00    159571
   macro avg       1.00      1.00      1.00    159571
weighted avg       1.00      1.00      1.00    159571
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier

ada=AdaBoostClassifier()
ada.fit(x_train, y_train)
predada=ada.predict(x_train)
print("Accuracy ",accuracy_score(y_train, predada)*100)      #accuracy score
print(confusion_matrix(y_train,predada))
print(classification_report(y_train,predada))
```

```
Accuracy  95.48163513420359
[[143629    648]
 [  6562   8732]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98    144277
           1       0.93      0.57      0.71     15294

    accuracy                           0.95    159571
   macro avg       0.94      0.78      0.84    159571
weighted avg       0.95      0.95      0.95    159571
```

- The **Accuracy** for y_train and pred_train(data predicted on x_train) is **95.48%**

- The **Confusion Matrix** for y_train and pred_train(data predicted on x_train) is –

	True Positive	False Positive	
	143629	**648**	
False Negative	**6562**	**8732**	True Negative

- The **Classification Report** for y_train and pred_train(data predicted on x_train) is

	precision	recall	f1-score	support
0	0.96	1.00	0.98	144277
1	0.93	0.57	0.71	15294
accuracy			0.95	159571
macro avg	0.94	0.78	0.84	159571
weighted avg	0.95	0.95	0.95	159571

Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier

gbdt= GradientBoostingClassifier()
gbdt.fit(x_train, y_train)
gbdt_pred=gbdt.predict(x_train)
print("Accuracy ",accuracy_score(y_train, gbdt_pred)*100)      #accuracy score
print(confusion_matrix(y_train,gbdt_pred))
print(classification_report(y_train,gbdt_pred))
```

```
Accuracy  95.87268363299096
[[143363    914]
 [  5672   9622]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98    144277
           1       0.91      0.63      0.75     15294

    accuracy                           0.96    159571
   macro avg       0.94      0.81      0.86    159571
weighted avg       0.96      0.96      0.96    159571
```

- The **Accuracy** for y_train and pred_train(data predicted on x_train) is **95.87%**

- The **Confusion Matrix** for y_train and pred_train(data predicted on x_train) is –

	True Positive	False Positive	
	143363	**914**	
False Negative	**5672**	**9622**	True Negative

- The **Classification Report** for y_train and pred_train(data predicted on x_train) is

	precision	recall	f1-score	support
0	0.96	0.99	0.98	144277
1	0.91	0.63	0.75	15294
accuracy			0.96	159571
macro avg	0.94	0.81	0.86	159571
weighted avg	0.96	0.96	0.96	159571

SVC

```
from sklearn.svm import SVC

svc=SVC()
svc.fit(x_train, y_train)
ad_pred=svc.predict(x_train)
print("Accuracy ",accuracy_score(y_train, ad_pred)*100)     #accuracy score
print(confusion_matrix(y_train,ad_pred))
print(classification_report(y_train,ad_pred))
```

```
Accuracy  95.86579014983926
[[143346    931]
 [  5666   9628]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98    144277
           1       0.91      0.63      0.74     15294

    accuracy                           0.96    159571
   macro avg       0.94      0.81      0.86    159571
weighted avg       0.96      0.96      0.96    159571
```

SVC

- The **Accuracy** for y_train and pred_train(data predicted on x_train) is **95.86%**

- The **Confusion Matrix** for y_train and pred_train(data predicted on x_train) is –

	True Positive	False Positive	
	143346	**931**	
False Negative	**5666**	**9628**	True Negative

- The **Classification Report** for y_train and pred_train(data predicted on x_train) is

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.98    144277
           1       0.91      0.63      0.74     15294

    accuracy                           0.96    159571
   macro avg       0.94      0.81      0.86    159571
weighted avg       0.96      0.96      0.96    159571
```

Cross Validation

```
from sklearn.model_selection import cross_val_score

#validation accuracy
scr=cross_val_score(LR,x_train,y_train,cv=5)
print("Cross validation score of Logistic Regression: ", scr.mean())
```

```
Cross validation score of Logistic Regression:  0.9582881636 7055
```

```
scr2=cross_val_score(kn,x_train,y_train,cv=5)
print("Cross validation score of KNeighbor Classifier: ", scr2.mean())
```

```
Cross validation score of KNeighbor Classifier:  0.956652533921 4799
```

```
scr3=cross_val_score(dt,x_train,y_train,cv=5)
print("Cross validation score of Decision Tree Classifier: ", scr3.mean())
```

```
Cross validation score of Decision Tree Classifier:  0.9289156675383203
```

```
scr4=cross_val_score(rf,x_train,y_train,cv=5)
print("Cross validation score of Random Forest Classifier: ", scr4.mean())
```

```
Cross validation score of Random Forest Classifier:  0.9291600733999099
```

```
scr5=cross_val_score(ada,x_train,y_train,cv=5)
print("Cross validation score of Ada Boost Classifier: ", scr5.mean())
```

```
Cross validation score of Ada Boost Classifier:  0.954603284003968 4
```

```
scr6=cross_val_score(gbdt,x_train,y_train,cv=5)
print("Cross validation score of Gradient Boost Classifier: ", scr6.mean())
```

```
Cross validation score of Gradient Boost Classifier:  0.9586140399827 95
```

Model	**Cross Validation Score**
Logistic Regression	0.9582
KNeighbor Classifier	0.9566
Decision Tree Classifier	0.9289
Random Forest Classifier	0.9291
Ada Boost Classifier	0.9546
Gradient Boost Classifier	0.9586

- Random Forest Classifier is performing better, hence it is carried forward

Hyperparameter tuned Random Forest Classifier Model

```
RandomForestClassifier()
```

```
RandomForestClassifier()
```

```
from sklearn.model_selection import GridSearchCV

#Creating parameter list to pass in GridSearchCV
parameters={'max_features':['auto','sqrt','log2'], 'max_depth':[4,5,6,7,8], 'criterion':['gini', 'entropy']}
```

```
GCV=GridSearchCV(RandomForestClassifier(), parameters, cv=5, scoring="accuracy")
GCV.fit(x_train,y_train)       #fitting data in the model
GCV.best_params_               #printing the best parameters found in GridSearchCV
```

```
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt'}
```

```
GCV.best_estimator_
```

```
RandomForestClassifier(max_depth=5, max_features='sqrt')
```

```
GCV_pred=GCV.best_estimator_.predict(x_train)        #Predicting with best parameters
accuracy_score(y_train,GCV_pred)
```

```
0.9587143027241792
```

```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(GCV.best_estimator_,x_train,y_train)
plt.title("ROC AUC Plot")
plt.show()
```

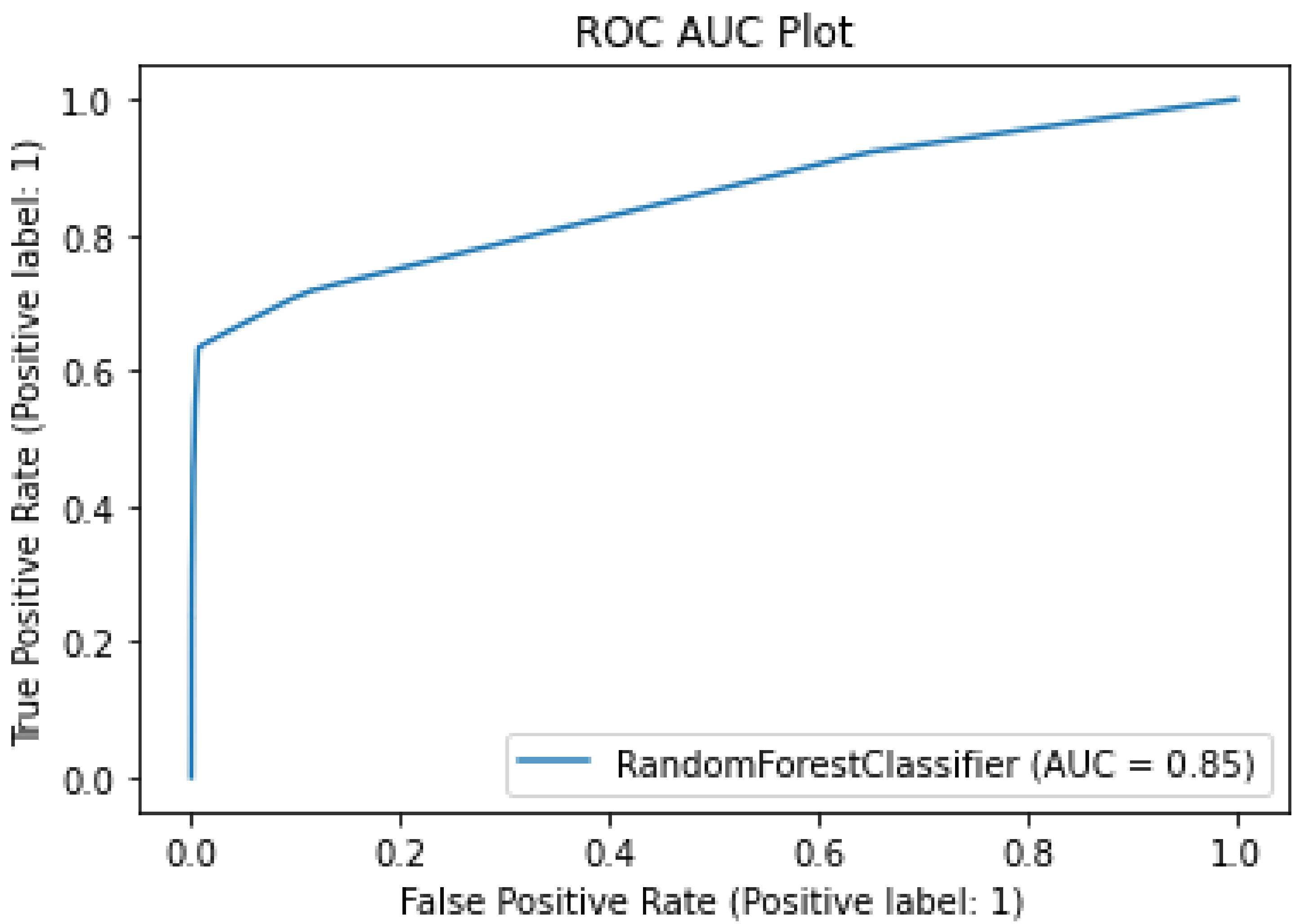
- Random Forest Classifier Model is hyperparameter tuned using GridSearchCV

- The best parameters for criterion, max_depth and max_features are found as follows-
  - **criterion: gini**
  - **max_depth: 5**
  - **Max_features: sqrt**

- Applying the above found best parameters on Random Forest Classifier Model, the following was obtained-

  - The **Accuracy** for target test and pred_test(data predicted on features_test) is **95.87%**

The ROC AUC Plot on the Hyperparameter tuned Random Forest Classifier Model



- **Final Accuracy** is **95.87%** and **AUC score** is **85%,** which depicts that our model is working well

- The test data (x_test) is fit into the Hyperparameter tuned Random Forest Classifier Model and the comments are classified as malignant (1) or non-malignant (0)

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

CONCLUSION

- The EDA analysis of the data is essential as it helps to understand the relationship between the target and features. Especially in this scenario, we get the liberty to add features in the test dataset by the aid of EDA analysis.
- The models should be used properly, as their regularization /hyperparamter tuning is highly advisable for the best outcome.
- The project imparted key knowledge about the kind of comments circulated in the social media environment. It also helped to understand the pattern of such comments and allow building a model to classify the malignant ones.
- The limitation of the solution is that it predicts if the comment is malignant or not. However, it does not provide the information about the person commenting it, or the account used to post such comments. However, a comment id was provided, so after classifying the malignant comments, the id can be traced back, and the accounts involved in such trolling activities can be blocked. Further, the IP address can traced for such accounts and can be put on red flags, in order to disallow such anonymous mis-handlers from creating new accounts with the intention to post malignant comments