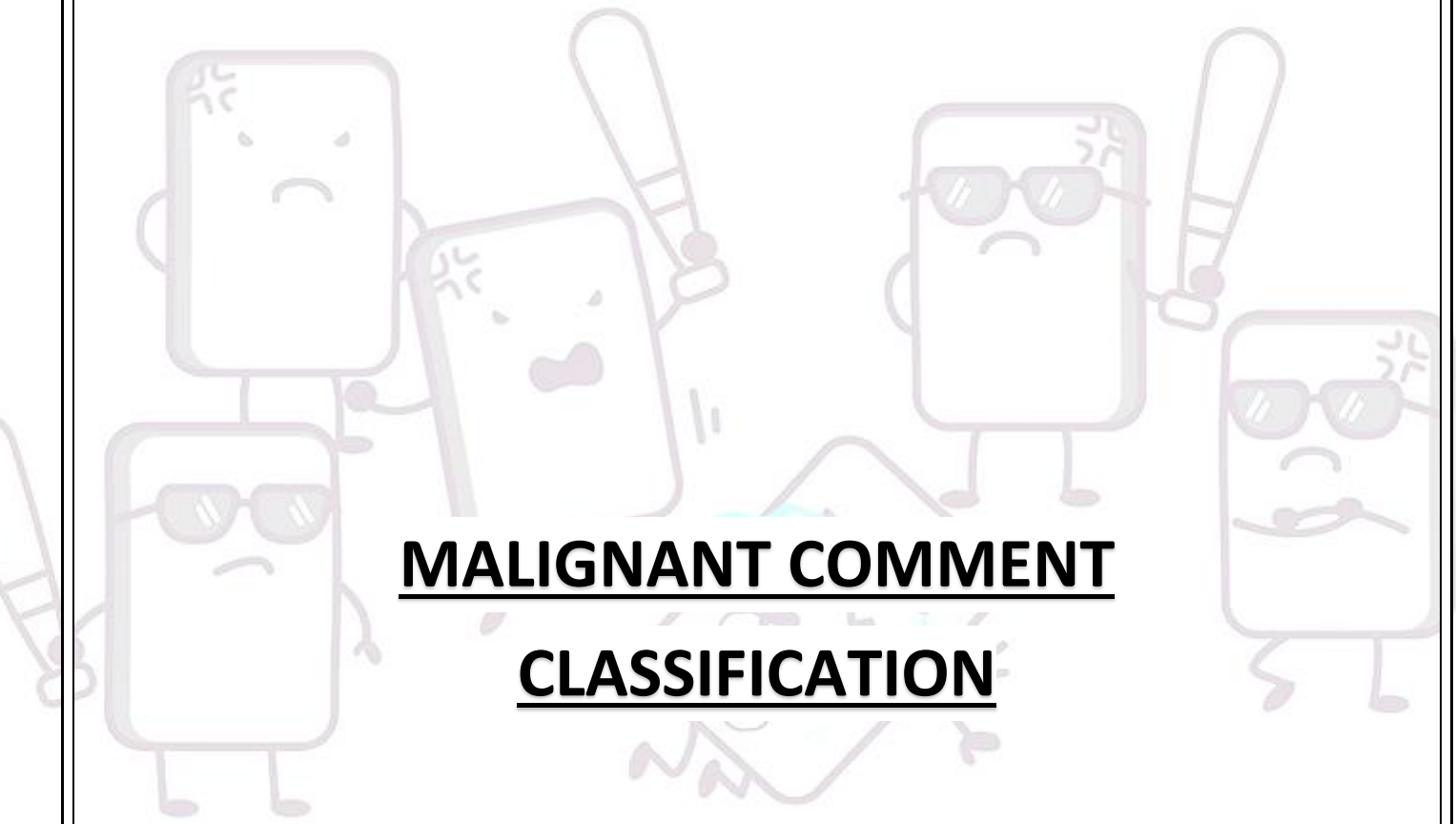


MALIGNANT COMMENT CLASSIFICATION

SUBMITTED BY

Abhijit Sarkar

ACKNOWLEDGMENT

I would like to express my gratitude for the opportunity and would like to thank Flip Robo Technologies for giving me an opportunity to work on this Given project. While going through this project I could Somewhere find the new modes of facts and a conceptual thinking and behaviour of the user behind the scene and in fact How these comments can hamper the brotherhood of the society. I am very grateful to DATA TRAINED team for providing me the adequate Trainings which actually helped me a lot to complete this project in the given time. I took help from Mr. Mohd Kashif where I faced the problem.

During the completion of the project, I found various issues working with NLP and I overcome those problem with the help of Google, You Tube Videos of Mr. Nair, Kaggle and Medium. I Used Sklearn library for solving this data set.

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.

➢ CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

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.

➢ REVIEW OF LITERATURE.

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection. Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour. There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and

affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts. Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but "u are an idiot" is clearly offensive.

➢ MOTIVATION FOR THE PROBLEM UNDERTAKEN

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. Here 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.

ANALYTICAL PROBLEM FRAMING

✓ Mathematical/ Analytical Modelling of the Problem

In this project, we will develop and evaluate the performance and classification prediction on the trained dataset and test dataset models based on comments which is provided. Once we get a best fit model, then we would apply on our test data. We will use various classification algorithm to predict our Target Variable. Let's have an overview of the Few algorithms we will use in predictions:

Logistic Regression:

Logistic regression analysis is valuable for predicting the likelihood of an event. It helps determine the probabilities between any two classes

Decision Tree Regression:

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

KNeighbors Classifier:

By default, the KNeighborsClassifier looks for the 5 nearest neighbors. We must explicitly tell the classifier to use Euclidean distance for determining the proximity between neighboring points.

Random Forest Classifier:

What is Random Forest classifier in Python? A random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

TF-IDF

Term frequency-inverse document frequency and it is a measure, used in the fields of information retrieval (IR) and machine learning, that can quantify the importance or relevance of string representations (words, phrases, lemmas, etc)

Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.

Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma. For instance, stemming the word 'Caring' would return 'Car'.

✓ Data Sources and their formats

There are two data-set in csv format: train and test dataset.

- 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.

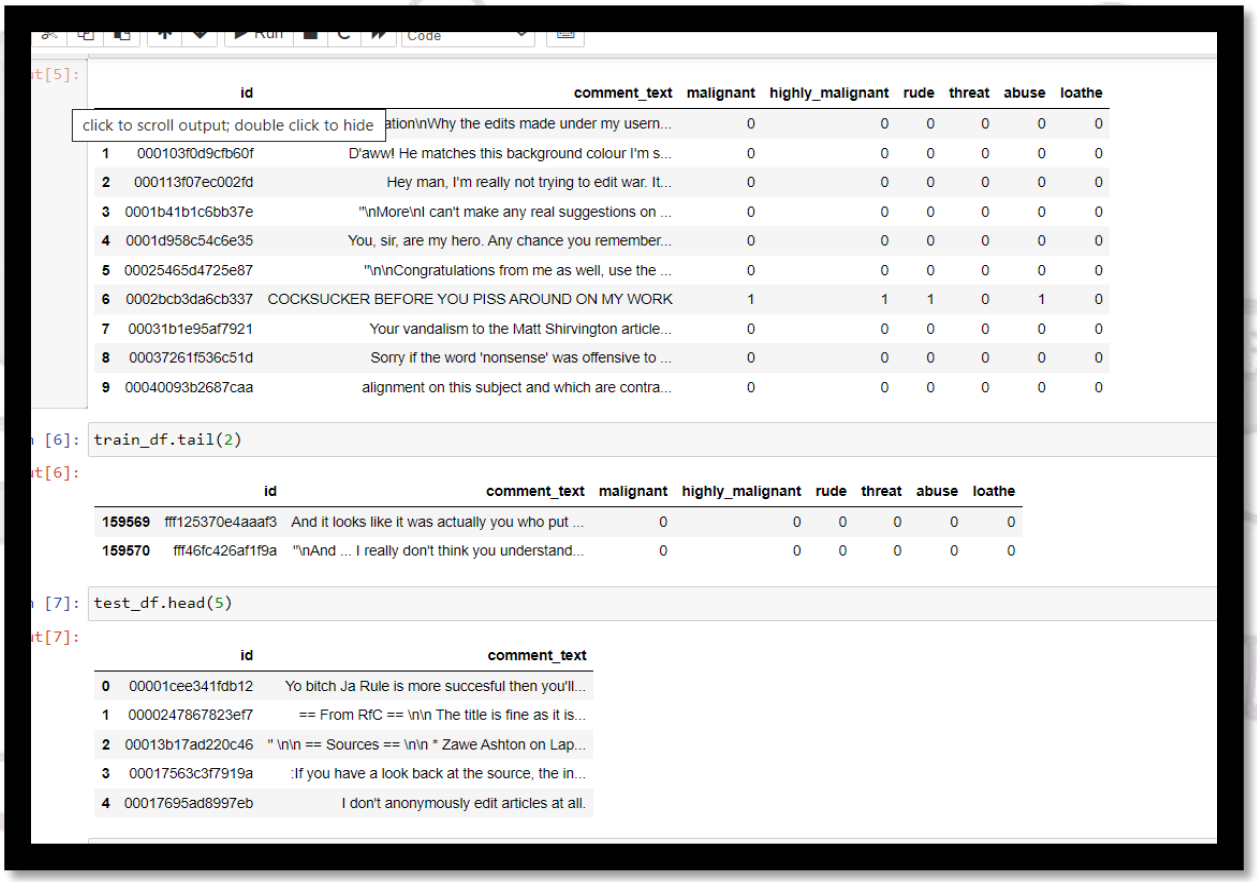
About Data Set

```
In [11]: train_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 159571 non-null  object
 1   comment_text       159571 non-null  object
 2   malignant          159571 non-null  int64
 3   highly_malignant   159571 non-null  int64
 4   rude               159571 non-null  int64
 5   threat             159571 non-null  int64
 6   abuse              159571 non-null  int64
 7   loathe             159571 non-null  int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB

In [12]: # About Test Data set
test_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153164 entries, 0 to 153163
Data columns (total 2 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   id             153164 non-null  object
 1   comment_text   153164 non-null  object
dtypes: object(2)
memory usage: 2.3+ MB
```

✓ Data Pre-processing

- Loading Both Data Set and checking top 5 with Data types
- Finding the unique values in categorical and numerical columns.
- Finding Data Missing percentage
- Finding nan values
- Finding duplicated values in columns rows.
- Feature Extraction.

Loading Data set



t[5]:

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
		click to scroll output; double click to hide						
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
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

[6]: `train_df.tail(2)`

t[6]:

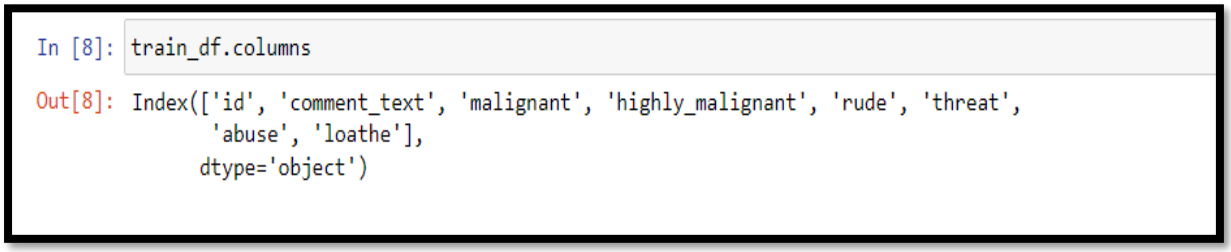
	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

[7]: `test_df.head(5)`

t[7]:

	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.

Columns checks:



```
In [8]: train_df.columns

Out[8]: Index(['id', 'comment_text', 'malignant', 'highly_malignant', 'rude', 'threat',
               'abuse', 'loathe'],
              dtype='object')
```

Unique:

Unique Values

In [13]:
```
# train dataset
unique_train=pd.DataFrame(data={"Unique Values":train_df.nunique()})
unique_train
```

Out[13]:

	Unique Values
id	159571
comment_text	159571
malignant	2
highly_malignant	2
rude	2
threat	2
abuse	2
loathe	2

In [14]:
```
# test data set:
unique_test= pd.DataFrame(data={"Unique Values":test_df.nunique()})
unique_test
```

Out[14]:

	Unique Values
id	153164
comment_text	153164

Shape Size

About Data Set

In [9]:
```
print("About Train Data Set :","\nDataType ",type(train_df),"\nShape ",train_df.shape,"\nSize ",train_df.size,)
```

```
About Train Data Set :
DataType  <class 'pandas.core.frame.DataFrame'>
Shape  (159571, 8)
Size  1276568
```

In [10]:
```
print("Test Data_set :","\nData Type",type(test_df),"\nTest Shape ",test_df.shape,"\nTest Size ",test_df.size)
```

```
Test Data_set :
Data Type <class 'pandas.core.frame.DataFrame'>
Test Shape  (153164, 2)
Test Size  306328
```

Null Values:

Finding Null values

In [19]:
```
train_null_values=pd.DataFrame(data={" Total Missing Values ":train_df.isnull().sum()," Missing Percentage ":train_df.isnull().s
train_null_values
```

Out[19]:

	Total Missing Values	Missing Percentage
comment_text	0	0.0
malignant	0	0.0
highly_malignant	0	0.0
rude	0	0.0
threat	0	0.0
abuse	0	0.0
loathe	0	0.0

Duplicated Rows:

```
In [16]: train_df.duplicated().sum()

Out[16]: 0
```

✓ Data Inputs- Logic- Output Relationships

I used the following to determine the relationship between variable:

- I have used Catplot and Count-plot for each pair of categorical features as well as Numerical Features.
- Used univariate, Bivariate, Multi-variate Graph to check for relations.
- Word Clouds was used to check the words in comments provided. Almost all the comments were checked.
- Pie plot along with bar graph is used to check the distribution. (In both the ways)
- Stop Words were added with Extra Words to unfold new relationship.
- Lemmatization was used to standardization
- Finally, Term Frequency-Inverse Document Frequency was used to convert the comments to vectors.

By the Use of these Graph and Tools,

I Uncovered the is a relationship between continuous numerical variable and The Target Variable.

✓ Hardware and Software Requirements and Tools Used

Hardware:

Device specifications — Copy

Device name	DataScientist
Processor	11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz 3.30 GHz
Installed RAM	16.0 GB (15.7 GB usable)
Device ID	4F7BEEF5-7469-44D4-B20A-DB8ACB2591EC
Product ID	00327-30000-00000-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

SOFTWARE USED:

- *I Used Jupiter Note Book.*
- *Microsoft Office 2020*
- *Windows 11 OS*

Library used: To run the program and to build the model we need some basic libraries as follows:

- *NumPy*
- *Pandas*
- *Seaborn*
- *Matplotlib*
- *SciPy*
- *Sklearn*
- *Pickle*
- *Imbalance Learn*
- *NLTK*

1) *import pandas as pd:*
   Pandas are a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array. This makes pandas a trusted ally in data science and machine learning.
2) *import NumPy as np:*
   NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3) *import seaborn as sns:*
   Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas' data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

4) *Import matplotlib.pyplot as plt:*
   matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

5. scipy:
SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

6. Sklearn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines.

7. Pickle

"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

8. IMB learn (SMOTE)

One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short.

9.NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

Model/s Development and Evaluation.

1. Identification of possible problem-solving approaches (methods)

Checked Total Numbers of Rows and Column

Checked All Column Name, Type of All Data, Null Values

Checked for special character present in dataset or not

Checked total number of unique values

Information about Data

Checked Description of Data and Dataset

Dropped irrelevant Columns

Replaced special characters and irrelevant data and checked all features through visualization.

Checked Descriptive and correlation of features

Converted all messages to lower case, replaced email addresses with 'email' and Replaced URLs with 'webaddress'

Replaced money symbols with 'moneysymb' (£ can by typed with ALT key + 156)

Replaced 10digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'

Replace Numbers with 'numbr', Punctation, extra space, leadning and trailing white space

Removed \n, removed stop words, Words of Sentence and Calculated length of sentence

Made one Target Column

Removed Total length

Checked the word which are offensive using Word Cloud

Checked the word which are not offensive using Word Cloud

Converted text into vectors using TF-IDF

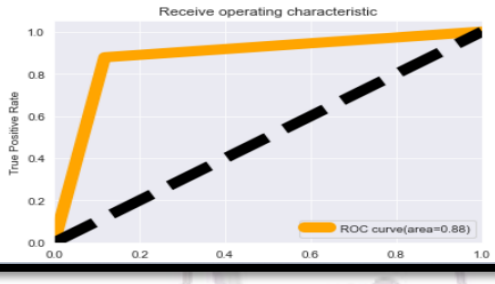
2. Testing of Identified Approaches (Algorithms)

```
In [85]: # linear_model,train test and metrics
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

         # Clssifiers
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.svm import SVC
         from sklearn.linear_model import SGDClassifier

         # cross Validation
         from sklearn.model_selection import cross_val_score

         # Ensamble
         from sklearn.ensemble import ExtraTreesClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import AdaBoostClassifier
         from sklearn.ensemble import GradientBoostingClassifier

         # neural network
         from sklearn.neural_network import MLPClassifier

         # hyper paramter
         from sklearn.model_selection import GridSearchCV
         import xgboost

In [103]: lg = LogisticRegression()
          dtc= DecisionTreeClassifier()
          knn= KNeighborsClassifier()
          svc= SVC()
          xg =xgb.Classifier()
          sgd= SGDClassifier()
          etc= ExtraTreesClassifier()
          rfc= RandomForestClassifier()
          ada= AdaBoostClassifier()
          gbc= GradientBoostingClassifier()
          mlp= MLPClassifier()
```

Run and evaluate selected models.

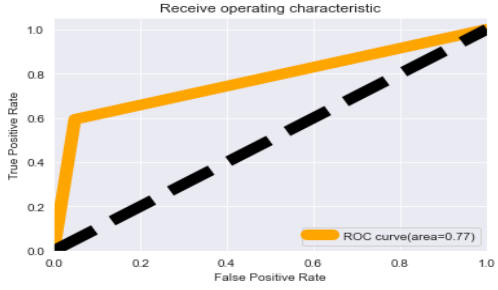
Linear Model

```
 LogisticRegression()
 ******************************
Training Score 0.8803721773859509
Accuracy Training Score = 0.8803721773859509  Accuracy Test Score = 0.8800816198398995

Training Confusion_Matrix
 [[100921  13625]
 [ 13812 100995]] Testing Confusion_Matrix
 [[25316  3484]
 [ 3392 25147]]
Classification Report
              precision    recall  f1-score   support

           0       0.88      0.88      0.88     28800
           1       0.88      0.88      0.88     28539

    accuracy                           0.88     57339
   macro avg       0.88      0.88      0.88     57339
weighted avg       0.88      0.88      0.88     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.8773247945530396 Training Score  0.8803721773859509 Testing Score  0.8800816198398995
A cross-fold  3 CV_mean  0.8778061473637213 Training Score  0.8803721773859509 Testing Score  0.8800816198398995
A cross-fold  4 CV_mean  0.8781270492375093 Training Score  0.8803721773859509 Testing Score  0.8800816198398995
A cross-fold  5 CV_mean  0.8780119870916027 Training Score  0.8803721773859509 Testing Score  0.8800816198398995
```



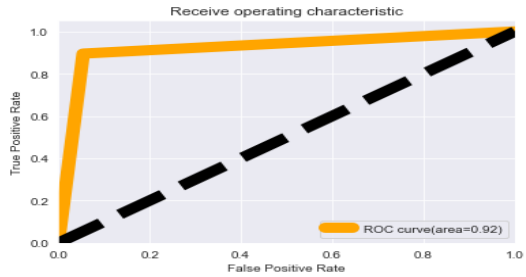
KNeighbors Classifier:

```
******************************
 KNeighborsClassifier()
 ******************************
Training Score 0.692391204824005
Accuracy Training Score = 0.692391204824005  Accuracy Test Score = 0.6557840213467273

Training Confusion_Matrix
 [[ 45604  68942]
 [  1609 113198]] Testing Confusion_Matrix
 [[ 9553 19247]
 [  490 28049]]
Classification Report
              precision    recall  f1-score   support

           0       0.95      0.33      0.49     28800
           1       0.59      0.98      0.74     28539

    accuracy                           0.66     57339
   macro avg       0.77      0.66      0.62     57339
weighted avg       0.77      0.66      0.62     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.6453371562513079 Training Score  0.692391204824005 Testing Score  0.6557840213467273
A cross-fold  3 CV_mean  0.654221254865849 Training Score  0.692391204824005 Testing Score  0.6557840213467273
A cross-fold  4 CV_mean  0.6553025546579605 Training Score  0.692391204824005 Testing Score  0.6557840213467273
A cross-fold  5 CV_mean  0.6606009374722274 Training Score  0.692391204824005 Testing Score  0.6557840213467273
```



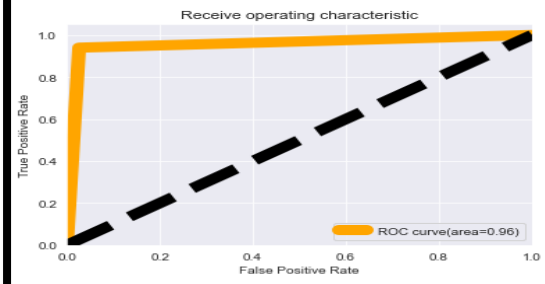
Model-Decision Tree Classifier:

```
*******************************
 DecisionTreeClassifier()
 *******************************
Training Score 0.9824855135969445
Accuracy Training Score = 0.9824855135969445  Accuracy Test Score = 0.918816163518722

Training Confusion_Matrix
 [[111415   3131]
 [   886 113921]] Testing Confusion_Matrix
 [[25627  3173]
 [ 1482 27057]]
Classification Report
              precision    recall  f1-score   support

           0       0.95      0.89      0.92     28800
           1       0.90      0.95      0.92     28539

    accuracy                           0.92     57339
   macro avg       0.92      0.92      0.92     57339
weighted avg       0.92      0.92      0.92     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.9037887349490046 Training Score  0.9824855135969445 Testing Score  0.918816163518722
A cross-fold  3 CV_mean  0.9134436956734056 Training Score  0.9824855135969445 Testing Score  0.918816163518722
A cross-fold  4 CV_mean  0.9183618656955896 Training Score  0.9824855135969445 Testing Score  0.918816163518722
A cross-fold  5 CV_mean  0.9198164399481437 Training Score  0.9824855135969445 Testing Score  0.918816163518722
```



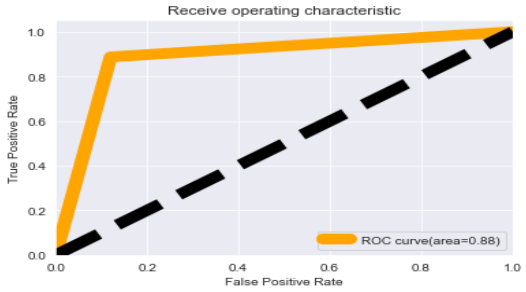
Model-Random Forest Classifier:

```
*******************************
 RandomForestClassifier()
 *******************************
Training Score 0.9824855135969445
Accuracy Training Score = 0.9824855135969445  Accuracy Test Score = 0.9560159751652453

Training Confusion_Matrix
 [[111317   3229]
 [   788 114019]] Testing Confusion_Matrix
 [[27012  1788]
 [  734 27805]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.94      0.96     28800
           1       0.94      0.97      0.96     28539

    accuracy                           0.96     57339
   macro avg       0.96      0.96      0.96     57339
weighted avg       0.96      0.96      0.96     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.9449409121984569 Training Score  0.9824855135969445 Testing Score  0.9560159751652453
A cross-fold  3 CV_mean  0.9525065226793911 Training Score  0.9824855135969445 Testing Score  0.9560159751652453
A cross-fold  4 CV_mean  0.9556876369065059 Training Score  0.9824855135969445 Testing Score  0.9560159751652453
A cross-fold  5 CV_mean  0.9570584818008084 Training Score  0.9824855135969445 Testing Score  0.9560159751652453
```



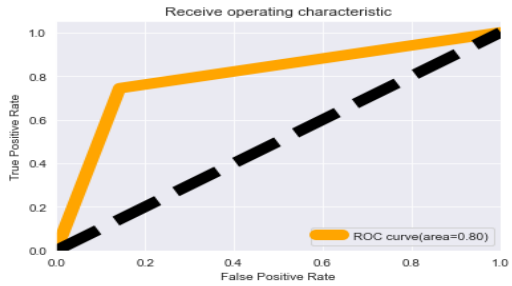
Model- XG-Boost Classifier

```
Training Score 0.8932126460085545
Accuracy Training Score = 0.8932126460085545  Accuracy Test Score = 0.8833778056819965

Training Confusion_Matrix
 [[102882  11664]
 [ 12828 101979]] Testing Confusion_Matrix
 [[25610  3190]
 [ 3497 25042]]
Classification Report
              precision    recall  f1-score   support

           0       0.88      0.89      0.88     28800
           1       0.89      0.88      0.88     28539

    accuracy                           0.88     57339
   macro avg       0.88      0.88      0.88     57339
weighted avg       0.88      0.88      0.88     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.8800978053102284 Training Score  0.8932126460085545 Testing Score  0.8833778056819965
A cross-fold  3 CV_mean  0.8818627656160617 Training Score  0.8932126460085545 Testing Score  0.8833778056819965
A cross-fold  4 CV_mean  0.8824522484059549 Training Score  0.8932126460085545 Testing Score  0.8833778056819965
A cross-fold  5 CV_mean  0.8829057495609712 Training Score  0.8932126460085545 Testing Score  0.8833778056819965
```



Model- ADA Boost Classifier

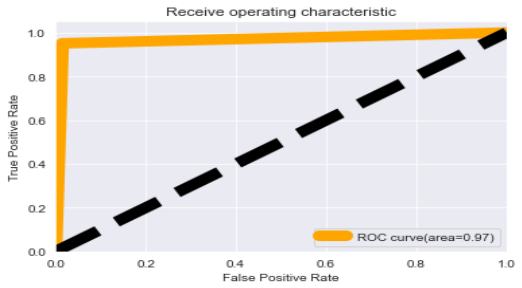
```
*******************************
 AdaBoostClassifier()
 *******************************
Training Score 0.7915135184628063
Accuracy Training Score = 0.7915135184628063  Accuracy Test Score = 0.7902125952667469

Training Confusion_Matrix
 [[ 80068  34478]
 [ 13339 101468]] Testing Confusion_Matrix
 [[20122  8678]
 [ 3351 25188]]
Classification Report
              precision    recall  f1-score   support

           0       0.86      0.70      0.77     28800
           1       0.74      0.88      0.81     28539

    accuracy                           0.79     57339
   macro avg       0.80      0.79      0.79     57339
weighted avg       0.80      0.79      0.79     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.7986305861342486 Training Score  0.7915135184628063 Testing Score  0.7902125952667469
A cross-fold  3 CV_mean  0.7904231719057386 Training Score  0.7915135184628063 Testing Score  0.7902125952667469
A cross-fold  4 CV_mean  0.7903743390119014 Training Score  0.7915135184628063 Testing Score  0.7902125952667469
A cross-fold  5 CV_mean  0.7921985910954913 Training Score  0.7915135184628063 Testing Score  0.7902125952667469
```



Model- Extra Trees Classifier

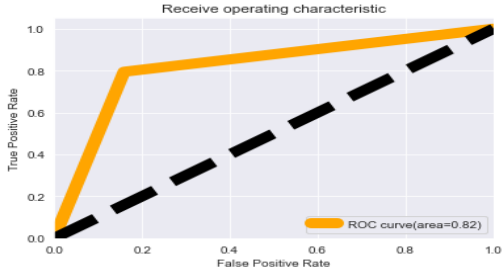
click to scroll output; double click to hide

```
 ********************************
Training Score 0.9824855135969445
Accuracy Training Score = 0.9824855135969445  Accuracy Test Score = 0.9673171837667207

Training Confusion_Matrix
 [[111415   3131]
 [   886 113921]] Testing Confusion_Matrix
 [[27344  1456]
 [  418 28121]]
Classification Report
              precision    recall  f1-score   support

           0       0.98      0.95      0.97     28800
           1       0.95      0.99      0.97     28539

    accuracy                           0.97     57339
   macro avg       0.97      0.97      0.97     57339
weighted avg       0.97      0.97      0.97     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.9607418414186654 Training Score  0.9824855135969445 Testing Score  0.9673171837667207
A cross-fold  3 CV_mean  0.9658239504415889 Training Score  0.9824855135969445 Testing Score  0.9673171837667207
A cross-fold  4 CV_mean  0.9675993749389589 Training Score  0.9824855135969445 Testing Score  0.9673171837667207
A cross-fold  5 CV_mean  0.9682621316680489 Training Score  0.9824855135969445 Testing Score  0.9673171837667207
```



Model- Gradient Classifier

```
 ********************************
 GradientBoostingClassifier()
 ********************************
Training Score 0.8184414417949624
Accuracy Training Score = 0.8184414417949624  Accuracy Test Score = 0.8159716772179494

Training Confusion_Matrix
 [[89777 24769]
 [16872 97935]] Testing Confusion_Matrix
 [[22510  6290]
 [ 4262 24277]]
Classification Report
              precision    recall  f1-score   support

           0       0.84      0.78      0.81     28800
           1       0.79      0.85      0.82     28539

    accuracy                           0.82     57339
   macro avg       0.82      0.82      0.82     57339
weighted avg       0.82      0.82      0.82     57339

 * * * * *
 MODEL CROSSVALIDATION
A cross-fold  2 CV_mean  0.8146233588659606 Training Score  0.8184414417949624 Testing Score  0.8159716772179494
A cross-fold  3 CV_mean  0.8154221254865849 Training Score  0.8184414417949624 Testing Score  0.8159716772179494
A cross-fold  4 CV_mean  0.8148082262497733 Training Score  0.8184414417949624 Testing Score  0.8159716772179494
A cross-fold  5 CV_mean  0.8165906600274525 Training Score  0.8184414417949624 Testing Score  0.8159716772179494
```



Hyper-Parameter

Selected Extra-Trees, Random-Forest for Hyper-Parameter Tunning as these models Works the best with better accuracy, and better F1 score.

Random Forest

- Cross-Fold 5
- CV_mean 0.96
- Training Score 0.98
- Testing Score 0.96

Extra Trees

- Cross-Fold 5
- CV_mean 0.97
- Training Score 0.98
- Testing Score 0.97

Random Forest Classifier

Model : Random Forest Classifier

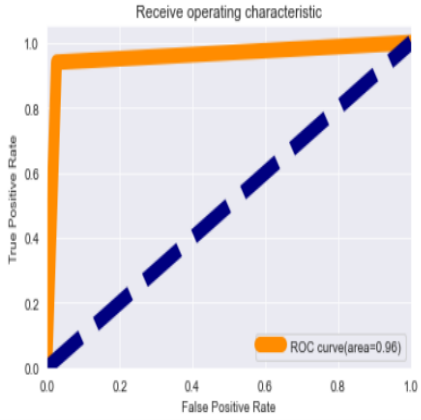
In [116]:
```
#random Forest Training and score
rfc=RandomForestClassifier(n_estimators=200, criterion='gini', max_features='sqrt')
rfc.fit(x_train,y_train)
rfc_score=rfc.score(x_train,y_train)

#predict random Forest
pred_train1=rfc.predict(x_train)
pred_test1=rfc.predict(x_test)

#result random Forest
print("Model Score ",rfc_score)
print("Accuracy Training Score ",accuracy_score(y_train,pred_train1)," Accuracy Test Score ",accuracy_score(y_test,pred_test1),"
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train1)," \nTesting Confusion_Matrix",confusion_matrix(y_test,
print("Classification Report \n",classification_report(y_test,pred_test1))
```

```
Model Score  0.9824855135969445
Accuracy Training Score  0.9824855135969445  Accuracy Test Score  0.955928774481592

Training Confusion_Matrix
 [[111322   3224]
 [   793 114014]]
Testing Confusion_Matrix [[27009  1791]
 [  736 27803]]
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.94      0.96     28800
           1       0.94      0.97      0.96     28539

    accuracy                           0.96     57339
   macro avg       0.96      0.96      0.96     57339
weighted avg       0.96      0.96      0.96     57339
```

Model Graph:

Model Graph

In [123]:
```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds= roc_curve(pred_test1,y_test)
roc_auc = auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=10,label='ROC curve(area=%0.2f)' % roc_auc)

plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receive operating characteristic")
plt.legend(loc="lower right")
plt.show()
```



SAVING MODEL

Random Forest working the best at an accuracy of +96%

Saving Model

In [135]:
```
import pickle
filename='malignant.pkl'
pickle.dump(rfc,open(filename,'wb'))
```

Saving Model and Loading Model:

In [137]:
```
# loading pack file
pickled_model= pickle.load(open(filename,'rb'))
result=pickled_model.score(x_test,y_test)
result*100
```

Out[137]: 95.5928774481592

In [139]:
```
array=np.array(y_test)
conclude=pd.DataFrame([pickled_model.predict(x_test)[:],array[:]],index=['Predicted','Original'])
conclude
```

Out[139]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Predicted	0	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0	0	1	0	1	1	1	1	0	0	1	1	1	0	0	1	0
Original	0	0	0	1	0	0	1	0	0	1	1	1	0	0	1	1	0	0	1	1	0	1	0	0	1	0	1	1	1	1	0	0	1	1	1	0	0	1	0

VERIFICATION WITH TESTING MODEL

```
In [143]: #test data (comments) converted to vectors
          testing_data = tf_vec.fit_transform(test_df["comment_text"])

          #applying testing on test data set
          prediction=rfc.predict(testing_data)
          prediction

          test_df['label'] = prediction
```

```
In [144]: test_df.head(5)
```

Out[144]:

	comment_text	New_length	label
0	yo bitch rule succesful ever whats hating sad ...	221	0
1	rfc title fine imo	18	0
2	source zawe ashton lapland	26	0
3	look back source information updated correct f...	109	0
4	anonymously edit article	24	0

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION.

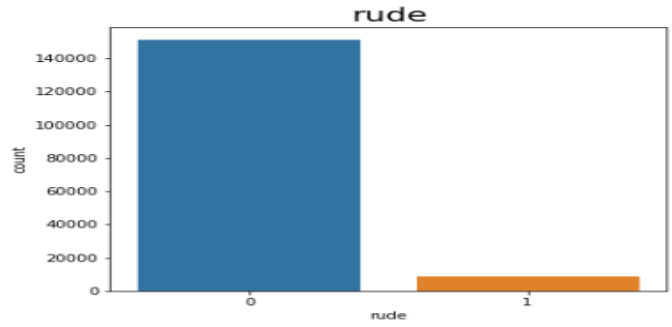
The Following Metrics were Used:

- Accuracy Score
- Classification Report
- Confusion Matrix

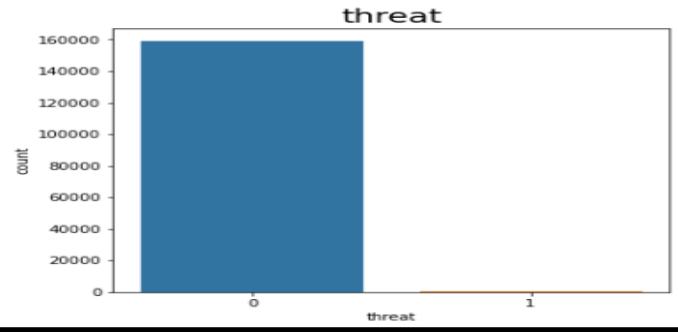
```
#result random Forest
print("Model Score ",rfc_score)
print("Accuracy Training Score ",accuracy_score(y_train,pred_train1)," Accuracy Test Score ",accuracy_score(y_test,pred_test1),"
print("Training Confusion_Matrix \n",confusion_matrix(y_train,pred_train1)," \nTesting Confusion_Matrix",confusion_matrix(y_test,
print("Classification Report \n",classification_report(y_test,pred_test1))
```

Visualizations: Uni-Variate Analysis

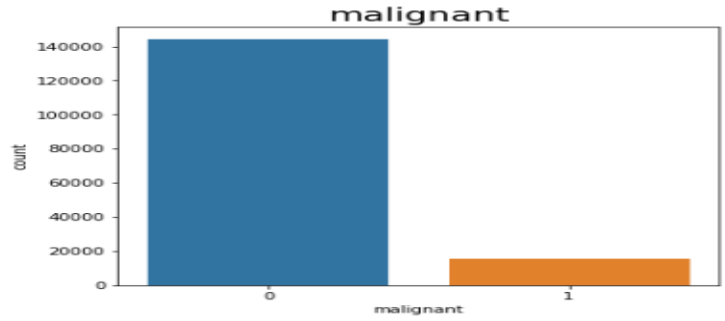
```
rude
  0     151122
1        8449
Name: rude, dtype: int64
```



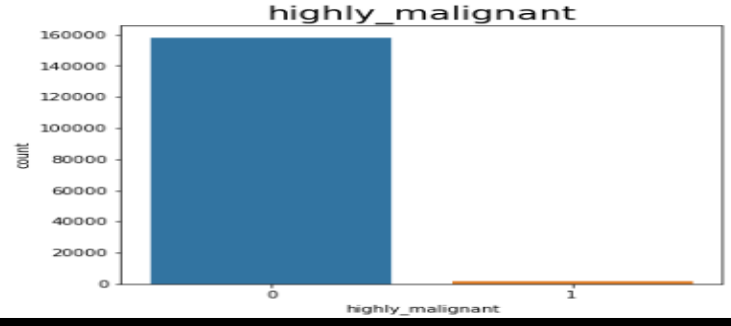

```
threat
  0     159093
1         478
Name: threat, dtype: int64
```




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

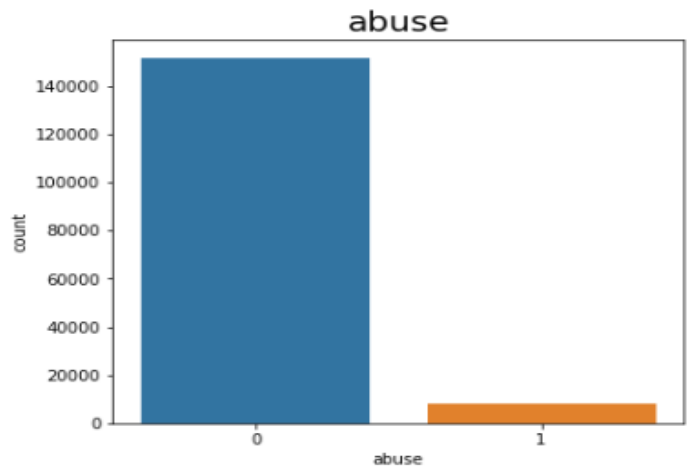



```
highly_malignant
  0     157976
1        1595
Name: highly_malignant, dtype: int64
```

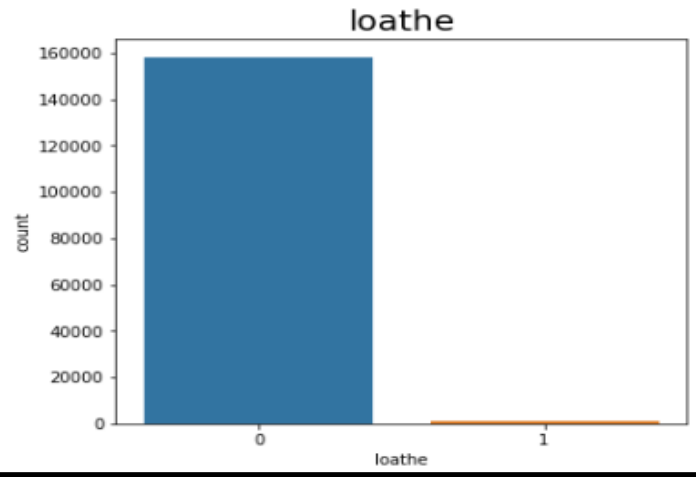


```
abuse
 0    151694
1      7877
```

click to scroll output; double click to hide



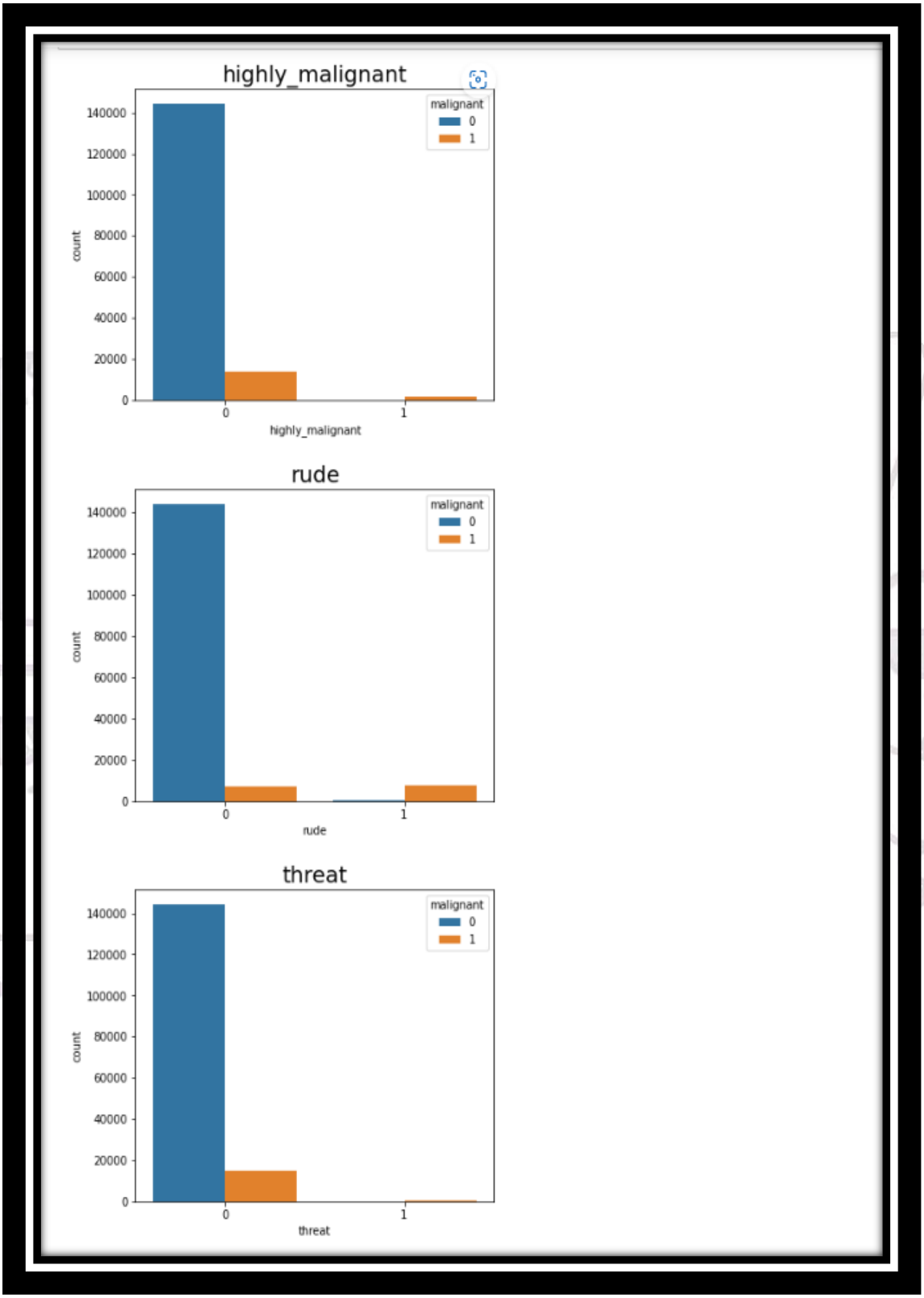

```
loathe
 0    158166
1      1405
Name: loathe, dtype: int64
```

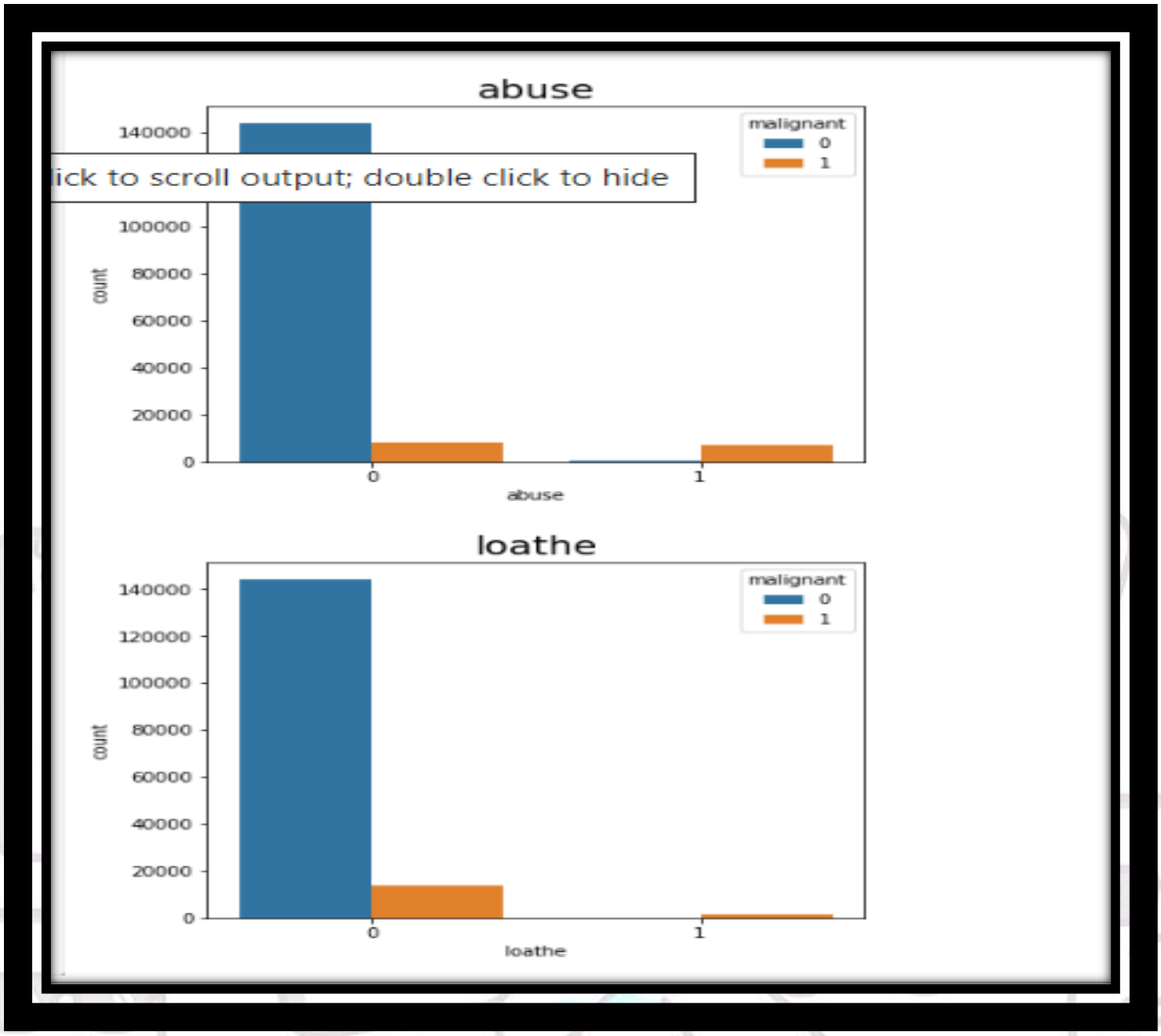


Observations:

- Column Malignant shows 1,44,277 are non-Malignant and 15294 is Malignant.
- Highly Malignant column shows 1,57,976 are non-Malignant and 1595 is highly Malignant
- Rude column shows 151122 are not rude whereas 8449 are rude
- Threat column shows 159093 are not a threat but 478 shows threat
- abuse column shows 151694 are not an abuse and 7877 shows abuse
- loathe column shows 158166 are not a loathe and 1405 shows loath

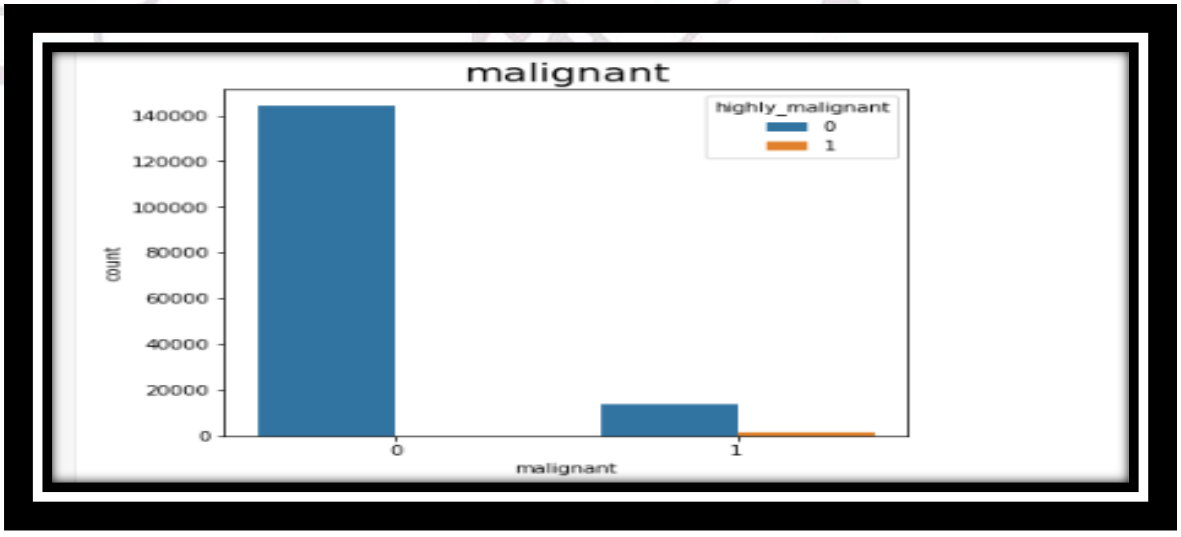
Bivariate Analysis

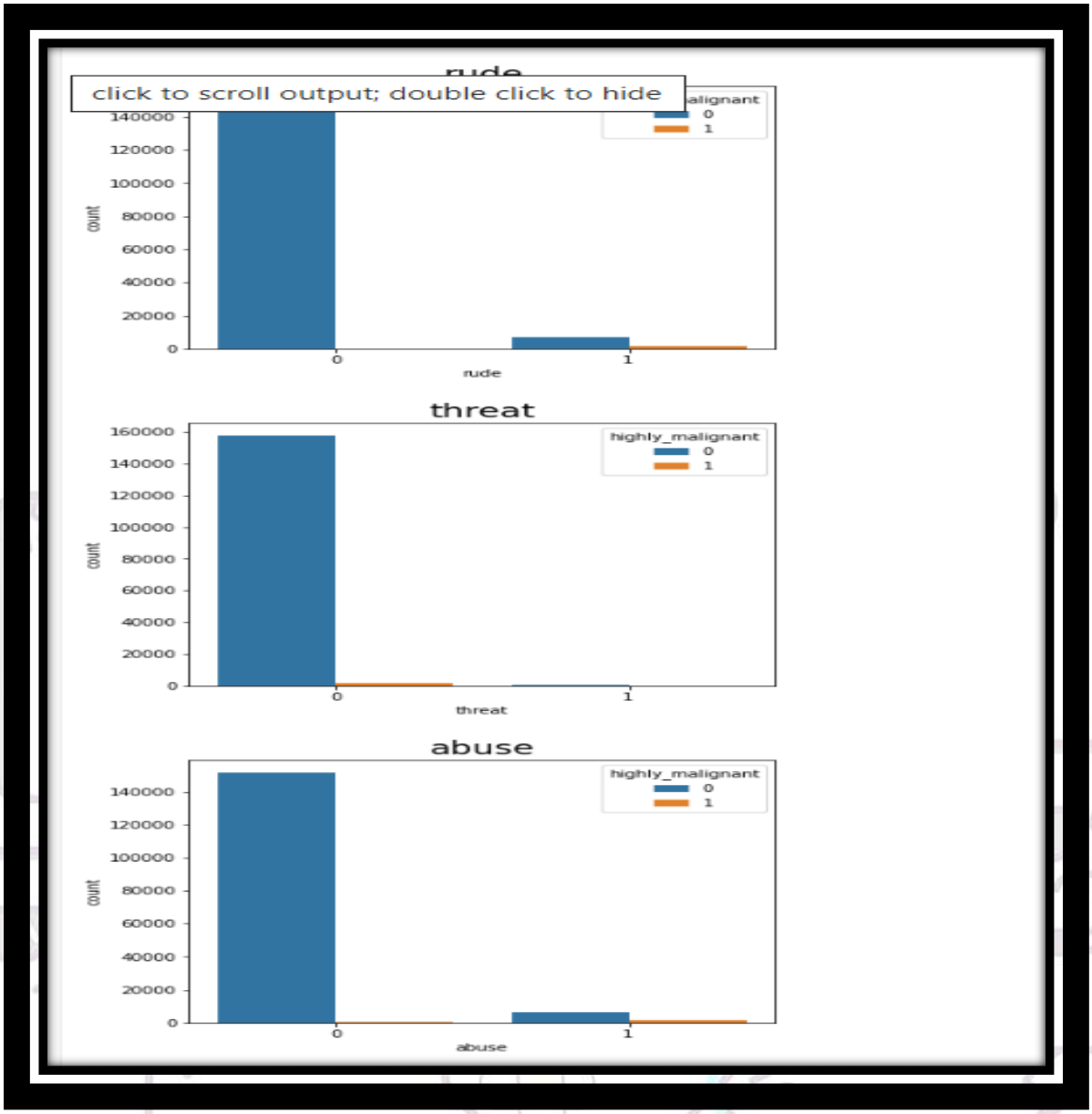


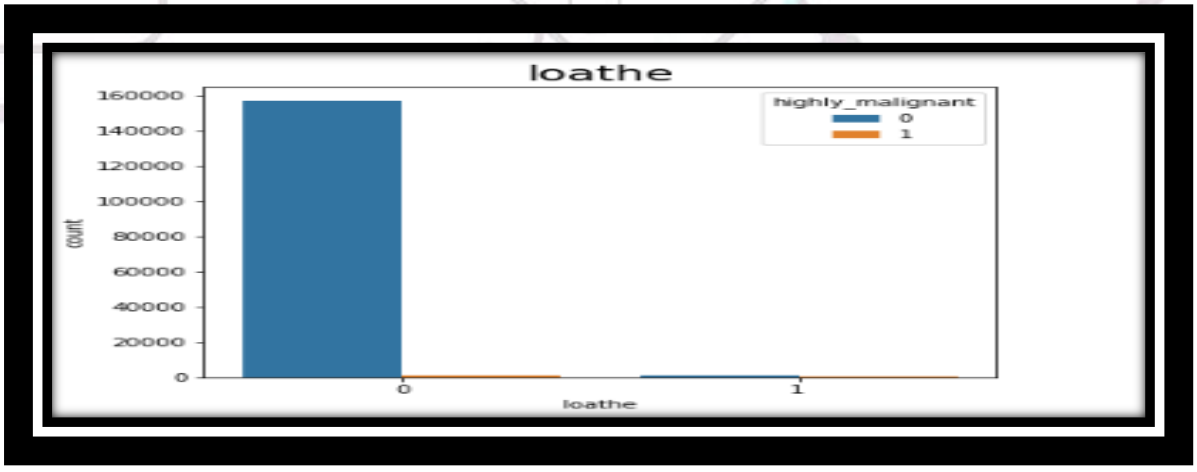


Observation:

- Non-Highly_Malignant,Rude,Non-threat,Non-abuse,Non-loathe are highly malignant.

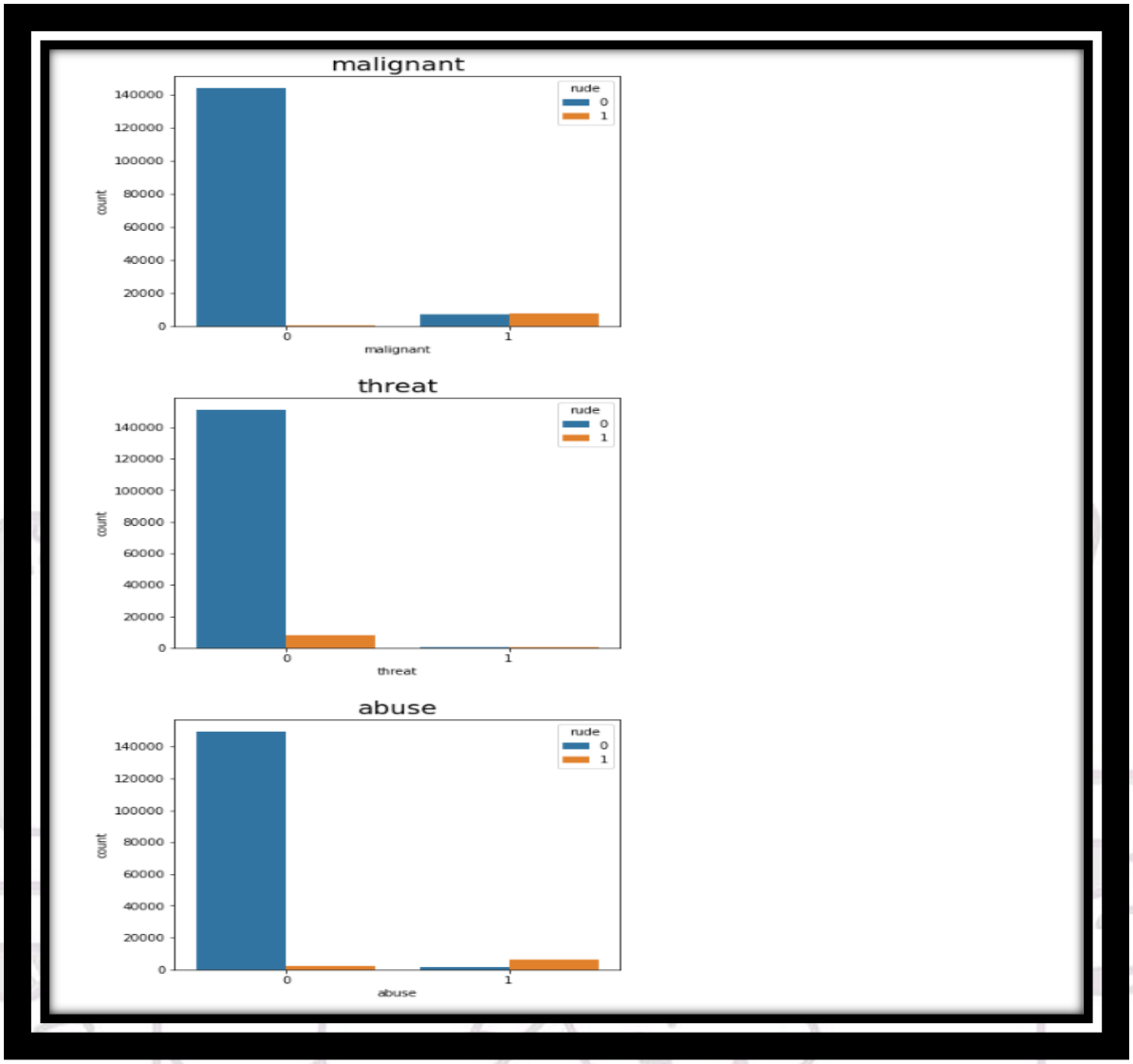


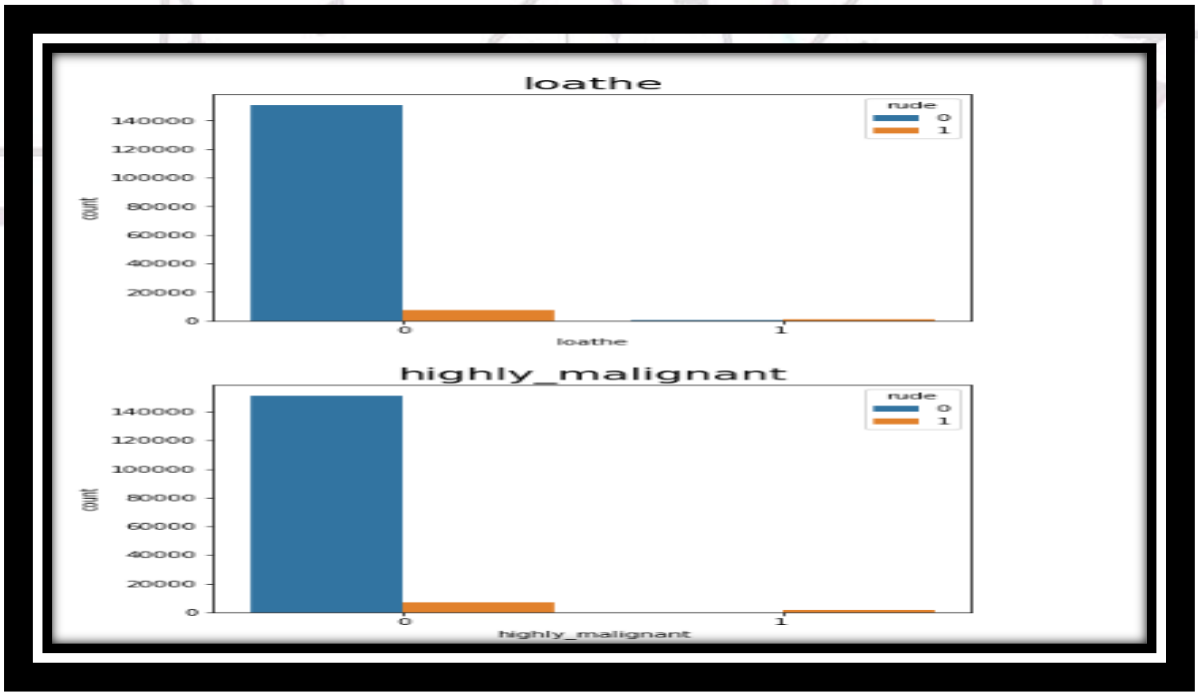




Observation:

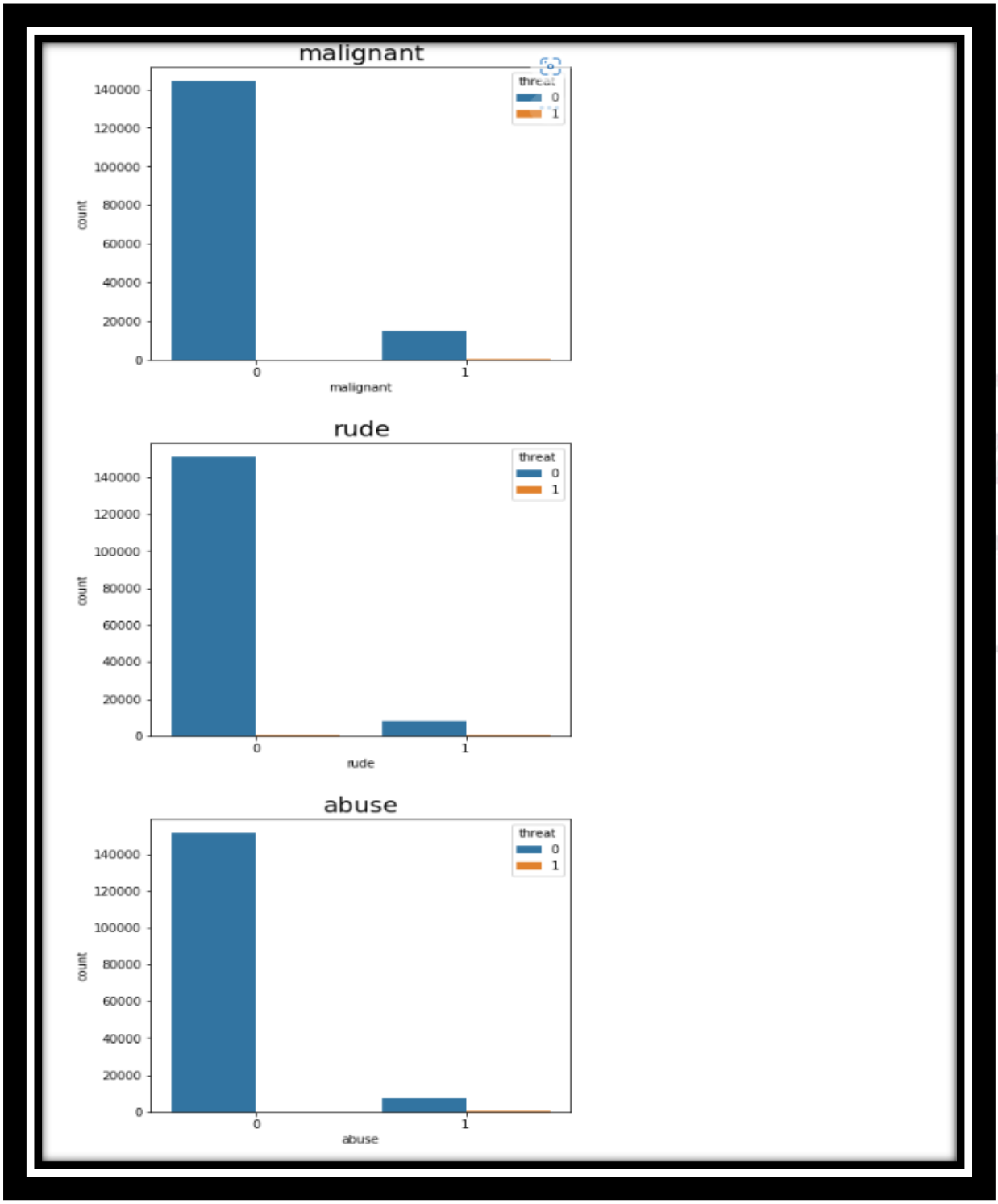
- Maximum malignant, rude are not highly malignant.
- Maximum non-threat, non-lathe, non-abuse Column shows high malignant.

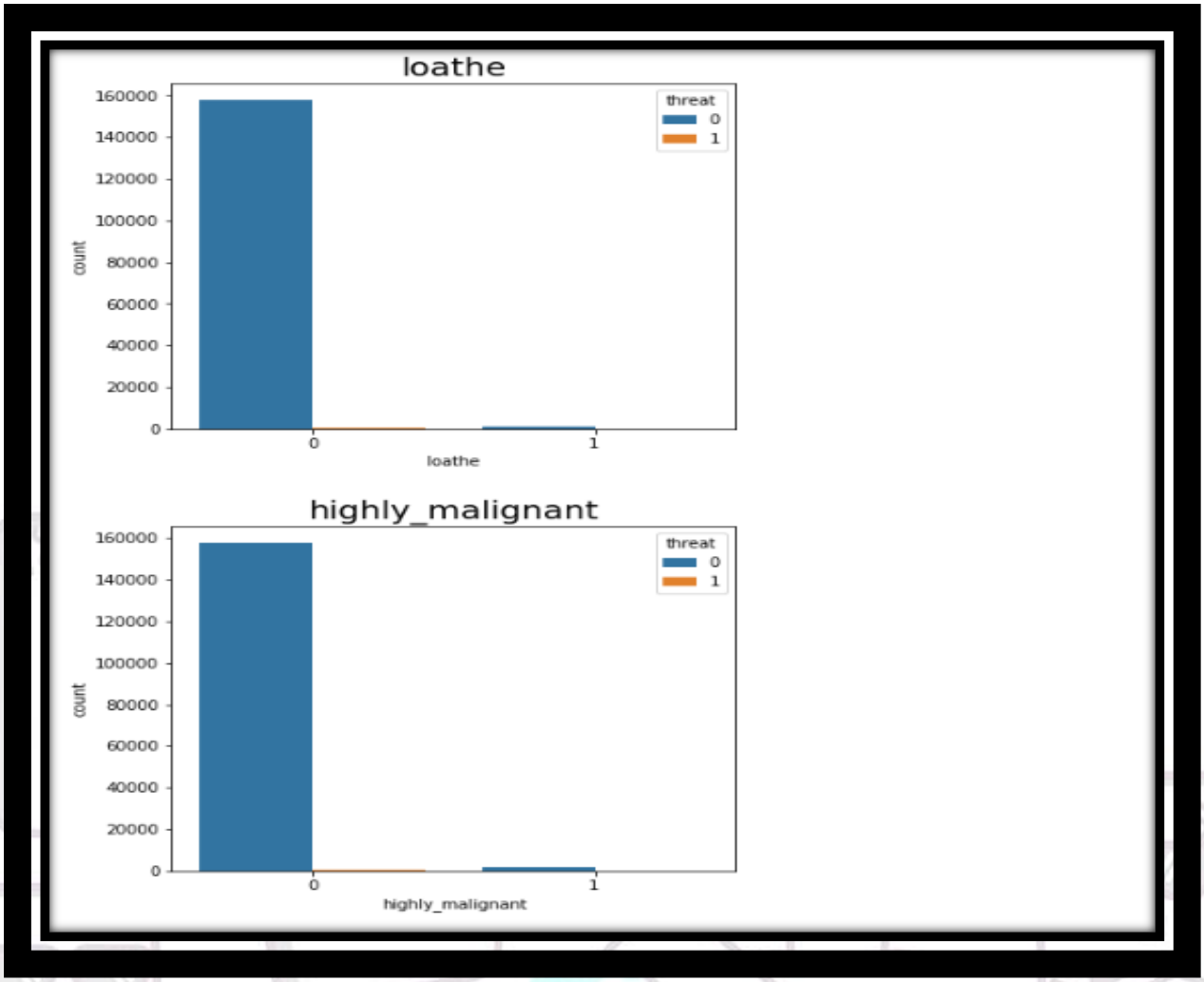

malignant
rude
0
1
count
malignant
threat
rude
0
1
count
threat
abuse
rude
0
1
count
abuse


loathe
rude
0
1
count
loathe
highly_malignant
rude
0
1
count
highly_malignant

OBSERVATION SHOWS

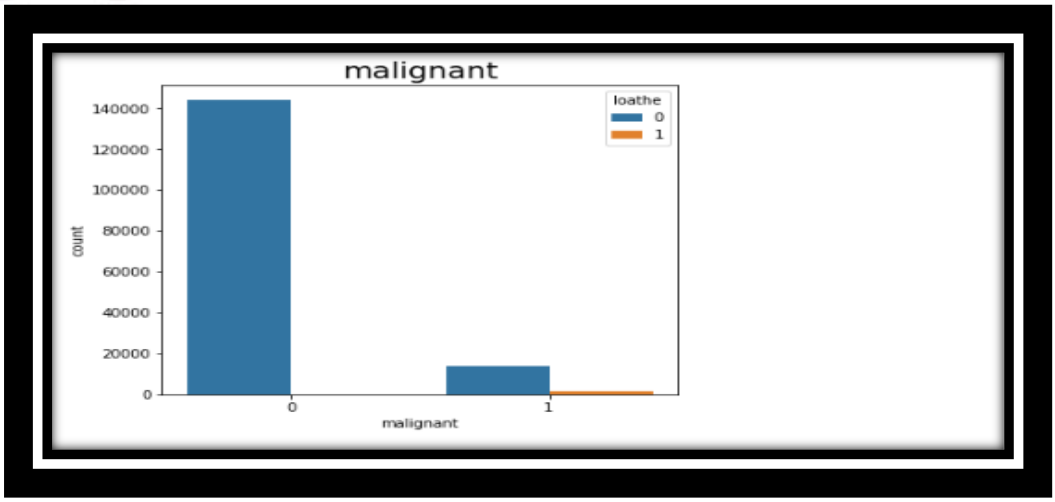
- Maximum malignant are rude
- Maximum non threat are rude
- Maximum abuse are rude
- Maximum non-loathe are rude
- Maximum non highly-malignant are rude.

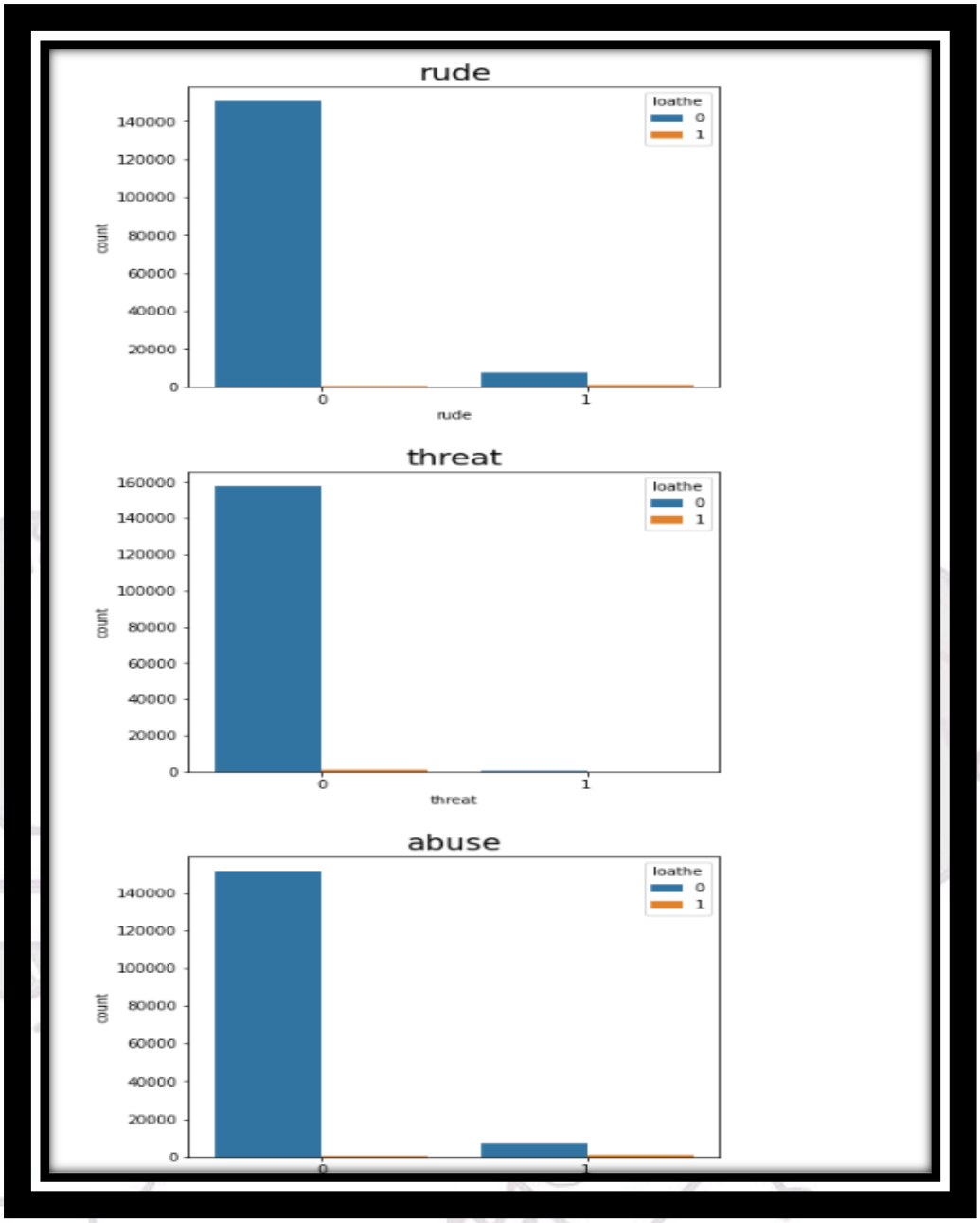


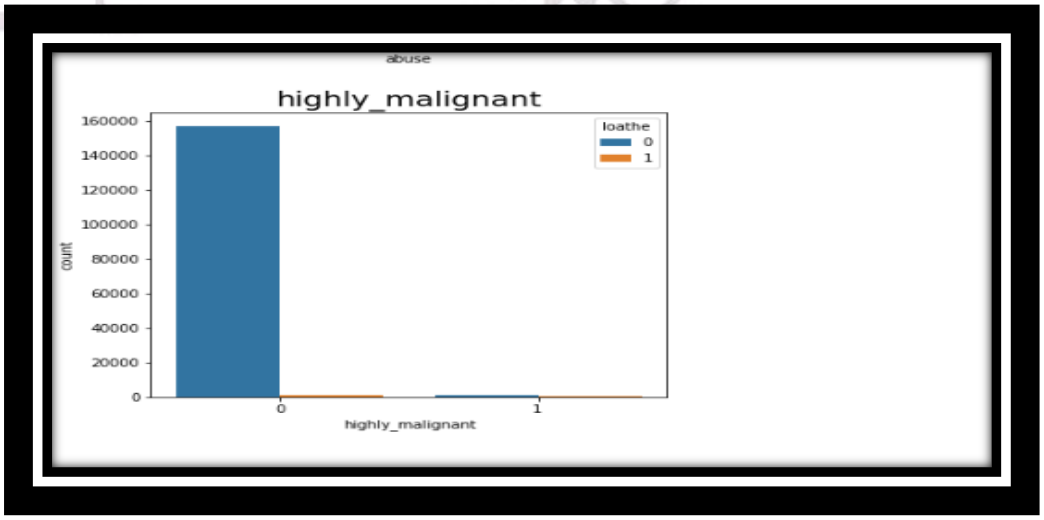


OBSERVATION

- Maximum malignant, rude, abuse, highly malignant comment showing low threat
- Non rude, loathe, highly malignant column shows they are a possible threat




rude
loathe
0
1
count
rude
threat
loathe
0
1
count
threat
abuse
loathe
0
1
count


abuse
highly_malignant
loathe
0
1
count
highly_malignant

OBSERVATION SHOWS:

1. Malignant are not loathe.
2. Rude are less loathe
3. Non loathe having Maximum threat.
4. Abuse is less loathed.

 # Descriptive Analysis

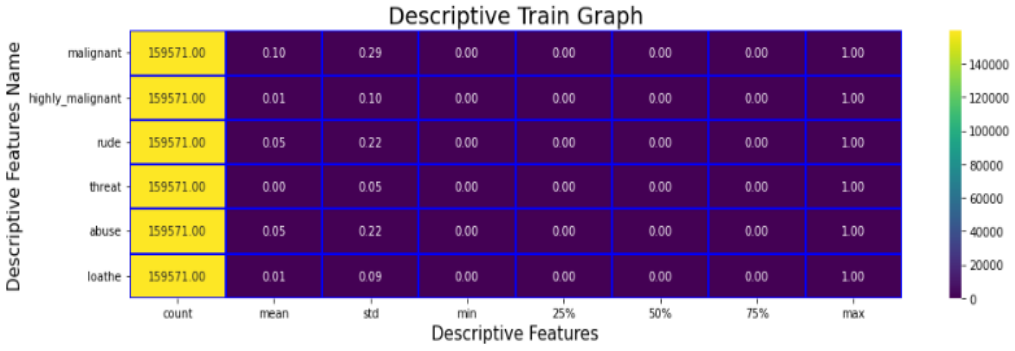
Descriptive Analysis :

In [37]:
```
train_df.describe()
```

Out[37]:

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [38]:
```
plt.figure(figsize=(16,4))
sns.heatmap(train_df.describe().T,annot=True,linewidths=0.2,linecolor='blue',fmt="0.2f",cmap="viridis")
plt.title("Descriptive Train Graph",fontsize=20)
plt.xlabel("Descriptive Features",fontsize=15)
plt.ylabel("Descriptive Features Name", fontsize=15)
plt.show()
```



Observation Shows:

1. Null Values - No Null Values.
2. Right Skewness - malignant highly_malignant rude threat abuse loathe
3. Left Skewness - Null
4. Std - Null
5. Outlier - Null

Correlation Graph:

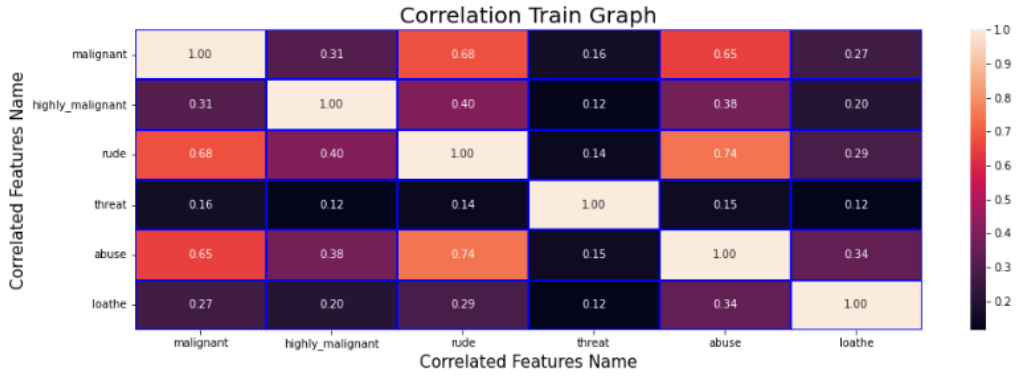
Correlation Graph

In [39]:
```
train_df.corr()
```

Out[39]:

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

In [40]:
```
plt.figure(figsize=(16,5))
sns.heatmap(train_df.corr().T,annot=True,linewidths=0.2,linecolor='blue',fmt="0.2f",cmap="rocket")
plt.title("Correlation Train Graph",fontsize=20)
plt.xlabel("Correlated Features Name",fontsize=15)
plt.ylabel("Correlated Features Name", fontsize=15)
plt.show()
```



Few Observation from Column:

- Malignant is 100 percentage correlated to itself.
- Higly_Malignant is 31 percentage correlated with malignant
- Rude is 68 percentage correlated with malignant
- Threat is 16 percentage correlated with malignant
- Abuse is 65 percentage correlated with malignant
- loathe is 27 percentage correlated with malignant

Note:

- Rude is 65 percentage positive highly correlated
- Threat is worse correlated.

➢ Interpretation of the Results

- This dataset was very special as it had a separate data type dataset having comments as target columns.
- Firstly, the datasets were having no null values.
- The data set had null duplicated rows.
- I found maximum label data columns.
- Maximum Cat Plot and Count plot was used followed to determine the condition and relation with one and another.
- I notice a huge number of symbols were present into comment texts which was cleaned and letter processed with the help of NLP (nltk) library like (stop words, lemmatization was used on the data set, finally TF-IDF was used to convert them to the vectors.)
- Scaling technique (By lemmatization) was done on both the train dataset and test data set, this has a good impact like it will help the model not to get biased.
- We have to use multiple models while building model using train dataset as to get the best model out of it.
- Extra Trees and Random Forest were the best among all the models. Result received with both model is approx. above 95 percent plus.
- However Extra Trees Shows good accuracy score and CV_Score close to testing score, hence this Shows our model is performing extremely well.
- Finally selected Extra Tree Regressor and saved the model and finally printed the score and finally compared the data with the test data set.
- Later the Test Data Set is vectorised and Test was applied to check the outcome.

CONCLUSION

➢ Key Findings and Conclusions of the Study

By, the help of Training Data set I have made a MALIGNANT COMMENTS CLASSIFICATION Model. We have done EDA, cleaned data and Visualized Data on the test Data Set. While cleaning the data Set it is analysed that ID column is irrelevant so I dropped that column. After that we have done prediction on basis of Data using Data Pre-processing, Checked Correlation, removed email addresses, URLs, money symbols, 10digit phone numbers, Punctation, extra space, leaning and trailing white space, ¥n, stop words, converted text into vectors using TF-IDF and at last train our data by splitting our data through train-test split process.

We worked with various classifiers while using these given models and finally selected best model which was giving best accuracy, F1 score and CV score. Random Forest Classifier model was selected and Hyper Tunning was used with Grid Search CV, And Best was Saved on the basis of Accuracy score, F1 score and CV score and at last I used that model to compare with predicted and Actual test data.

Thus, our project Stands completed.

And saved the model as filename= 'malignant.pkl'

➢ Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle as it contains all types of data in it. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in Malignant Comment Classification.

The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove missing value and to replace null value and zero values with their respective mean, median or mode.

This study is an exploratory Data Analysis has attempted to use machine learning algorithms in finding the probability of Malignant and Non-Malignant comments in each model. To conclude the application of Machine Learning in prediction is still at an early stage. I hope this study has moved a small step ahead in providing solution to the companies.

The Challenges I faced was when I was not much aware of NLP and I used various sources like YOU-Tube, Kaggle and Medium and NLTK library where I faced problem. I faced issues even at the time of binning process. Even The Algorithm was working huge time to complete the test and CV_Score generating was again time taking in each of the cases.

Finally, I had to run the test twice and thrice with different standardization process. I was actually thinking to get best out of those.

However, I finally achieved a good model out of this.

➢ Limitations of this work and Scope for Future Work

This model doesn't predict future probability. The future will be unpredictable at all times due to this, the risk in malignant classification remains an import factor. This Model can predict for a time period and needs to be updated as per need or on yearly basis to stop social media platforms from being more malignant. Machine can classify the malignancy of the comments but the intensity of the user remains unpredictable.

The best way to be future ready to get the modal updated once or twice as per the market standard.

Thank-you