FLIP ROBO

MALIGNANT COMMENTS CLASSIFICATION

Submitted by: Haindavi Chakravarthi

Internship: 23

ACKNOWLEDGMENT

- ➢ The successful completion of any work would be always be incomplete unless we mention the valuable cooperation and assistance of those people who were a source of constant guidance and encouragement, they served as bacon light and crowned our efforts with success.

- ➢ First of all, I would like to thank all my mentors in Data Trained and FlipRobo Technologies for this opportunity.

- ➢ I wish to express our sincere thanks to the above people, without whom I would not have been able to complete this project.

MALIGNANT COMMENTS CLASSIFICATION

- ✓ which also helped me in doing a lot of research and I came to know about so many new things on the Natural language processing project I am thankful to them.

- ✓ Secondly, I would also like to thank my parents who helped me a lot in finalizing this project within the limited time frame.

INTRODUCTION

- **Business Problem Framing**

  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.

  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.

- **Conceptual Background of the Domain Problem**

  The conceptual background of the problem depends on the language and words which make any comment toxic or neutral, our main motto is to find those words which are highly malignant.

- **Review of Literature**

  Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data, and other attributes. It is commonly conducted by data analysts using visual analytics tools, but it can also be done in more advanced statistical software, Python. Before it can analyze data collected by multiple data sources and stored in data warehouses, an organization must know how many cases are in a data set, what variables are included, how many missing values there are, and what general hypotheses the data is likely to support. An initial exploration of the data set can help answer these questions by familiarizing analysts with the data with which they are working. We divided the data 8:2 for Training and Testing purposes respectively.

- **Motivation for the Problem Undertaken**

  Every problem of Machine learning gives us chance to enhance and develop problem-solving skills. These Problems do's the same.
  When this real-life problem of predicting whether the comments to anybody is legal or not which kind of words are used and how much they are toxic to someone, whether to rely on old data or new data words are words they are accepted globally important and with help of A. I technology we make a completely new model of detection. As Data scientists it is our role to help and understand the market better with newer data, for constructing real-life helpful models for companies.

  In this project, we have a dataset of comments and their categories of how lethal they are, for that, we have to analyze those features which make comments toxic and help companies to build the detection models for filtration and stopping such comments for plotting.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  As for any basic model building, we have to understand the type of target variable, the data of the target variable is continued or classified.

  Data Analysis is always the difficult part, for better understanding different kinds of bar plots, distribution plots are created with the target Column for finding the insights of the dataset we have.

  Analytical Modelling always starts with the target variable we have, and in that case, our target variables are text comments first we have to filter them and make data clean then using different analysis tools select the list of toxic words which makes comments malignant, for that, we create some distribution plots with the target variable to understand which feature columns help to learn the model best and which feature columns reduce the accuracy of the model.

  And after finding the relation and correlation with the target variable we choose either Regression Model or Classification Model. Here in this problem, our target feature column is classified so we build our Machine Learning model on classification.

- **Data Sources and their formats**

  Data Set Description

  The data set contains the training set, which has approximately 1,59,000 samples, and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which include ‘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 an indication of the comments that are giving any threat to someone.

  Abuse: It is for abusive comments.

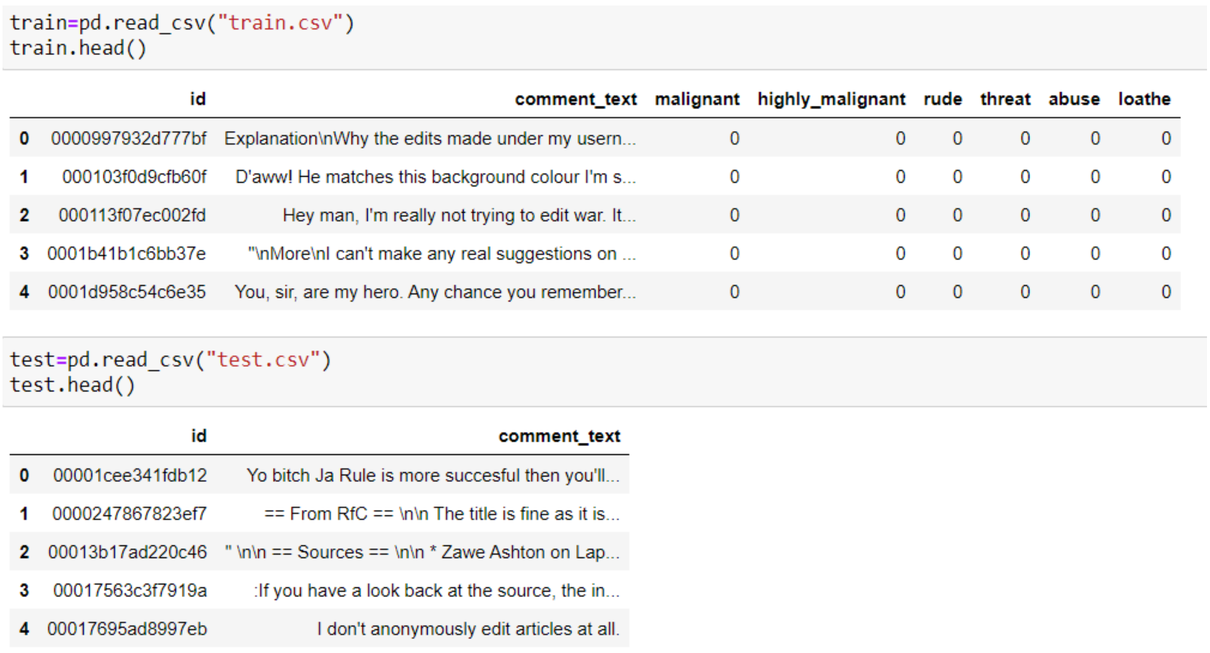
  Loathe: It describes the hateful comments and loathing in nature.

  ID: It includes unique Ids associated with each comment text given.

Comment text: This column contains the comments extracted from various social media platforms.

This project is more about exploration, feature engineering, and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do a good amount of data exploration and derive some interesting features using the comments text column available.

Data set looks as follows.



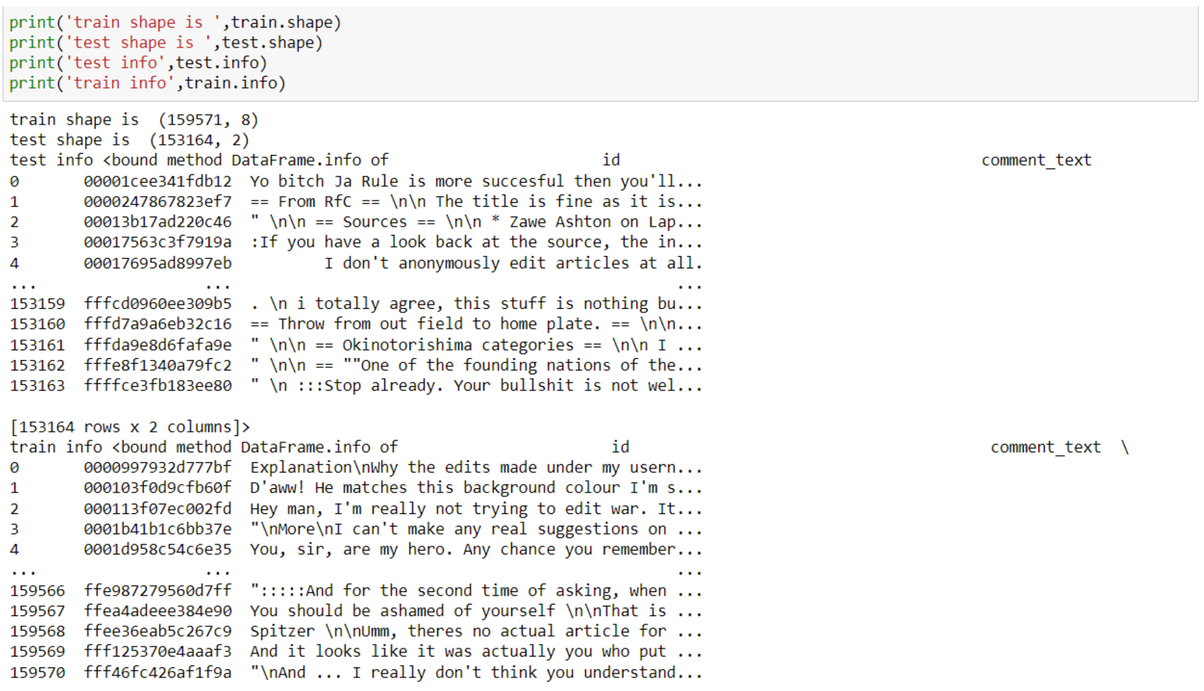
```
train=pd.read_csv("train.csv")
train.head()
```

	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

```
test=pd.read_csv("test.csv")
test.head()
```

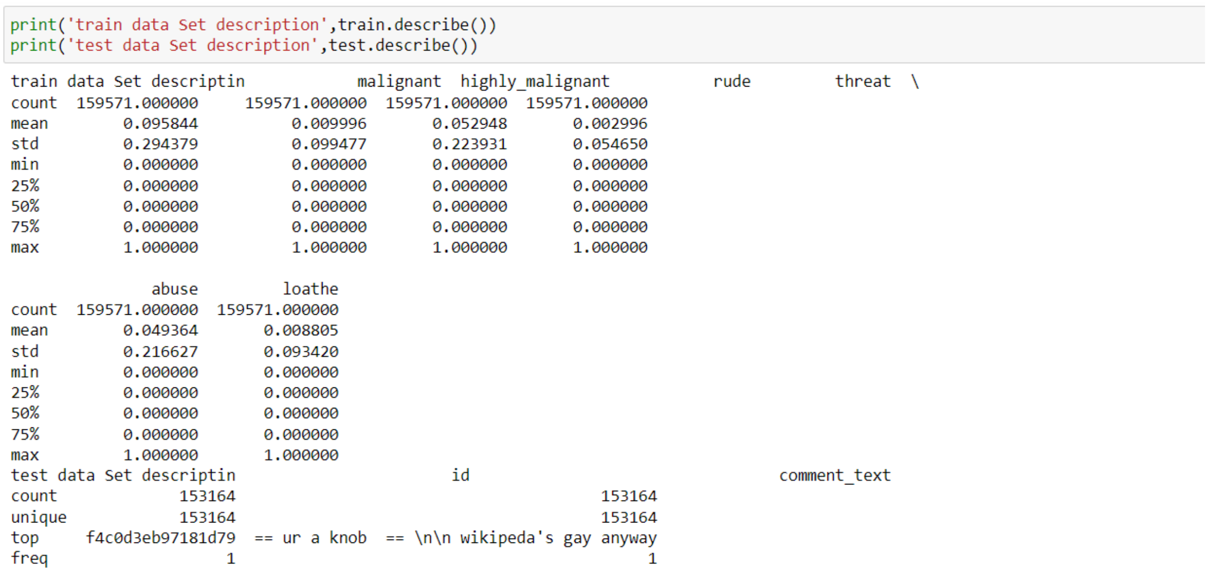
	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

Dataset Information looks as follows-



```
print('train shape is ',train.shape)
print('test shape is ',test.shape)
print('test info',test.info)
print('train info',train.info)
```

```
train shape is  (159571, 8)
test shape is  (153164, 2)
test info <bound method DataFrame.info of                          id                                       comment_text
0       00001cee341fdb12  Yo bitch Ja Rule is more succesful then you'll...
1       0000247867823ef7  == From RfC == \n\n The title is fine as it is...
2       00013b17ad220c46  " \n\n == Sources == \n\n * Zawe Ashton on Lap...
3       00017563c3f7919a  :If you have a look back at the source, the in...
4       00017695ad8997eb          I don't anonymously edit articles at all.
...                  ...                                                ...
153159  fffcd0960ee309b5  . \n i totally agree, this stuff is nothing bu...
153160  fffd7a9a6eb32c16  == Throw from out field to home plate. == \n\n...
153161  fffda9e8d6fafa9e  " \n\n == Okinotorishima categories == \n\n I ...
153162  fffe8f1340a79fc2  " \n\n == ""One of the founding nations of the...
153163  ffffce3fb183ee80  " \n :::Stop already. Your bullshit is not wel...

[153164 rows x 2 columns]>
train info <bound method DataFrame.info of                          id                                       comment_text  \
0       0000997932d777bf  Explanation\nWhy the edits made under my usern...
1       000103f0d9cfb60f  D'aww! He matches this background colour I'm s...
2       000113f07ec002fd  Hey man, I'm really not trying to edit war. It...
3       0001b41b1c6bb37e  "\nMore\nI can't make any real suggestions on ...
4       0001d958c54c6e35  You, sir, are my hero. Any chance you remember...
...                  ...                                                ...
159566  ffe987279560d7ff  ":::::And for the second time of asking, when ...
159567  ffea4adeee384e90  You should be ashamed of yourself \n\nThat is ...
159568  ffee36eab5c267c9  Spitzer \n\nUmm, theres no actual article for ...
159569  fff125370e4aaaf3  And it looks like it was actually you who put ...
159570  fff46fc426af1f9a  "\nAnd ... I really don't think you understand...
```

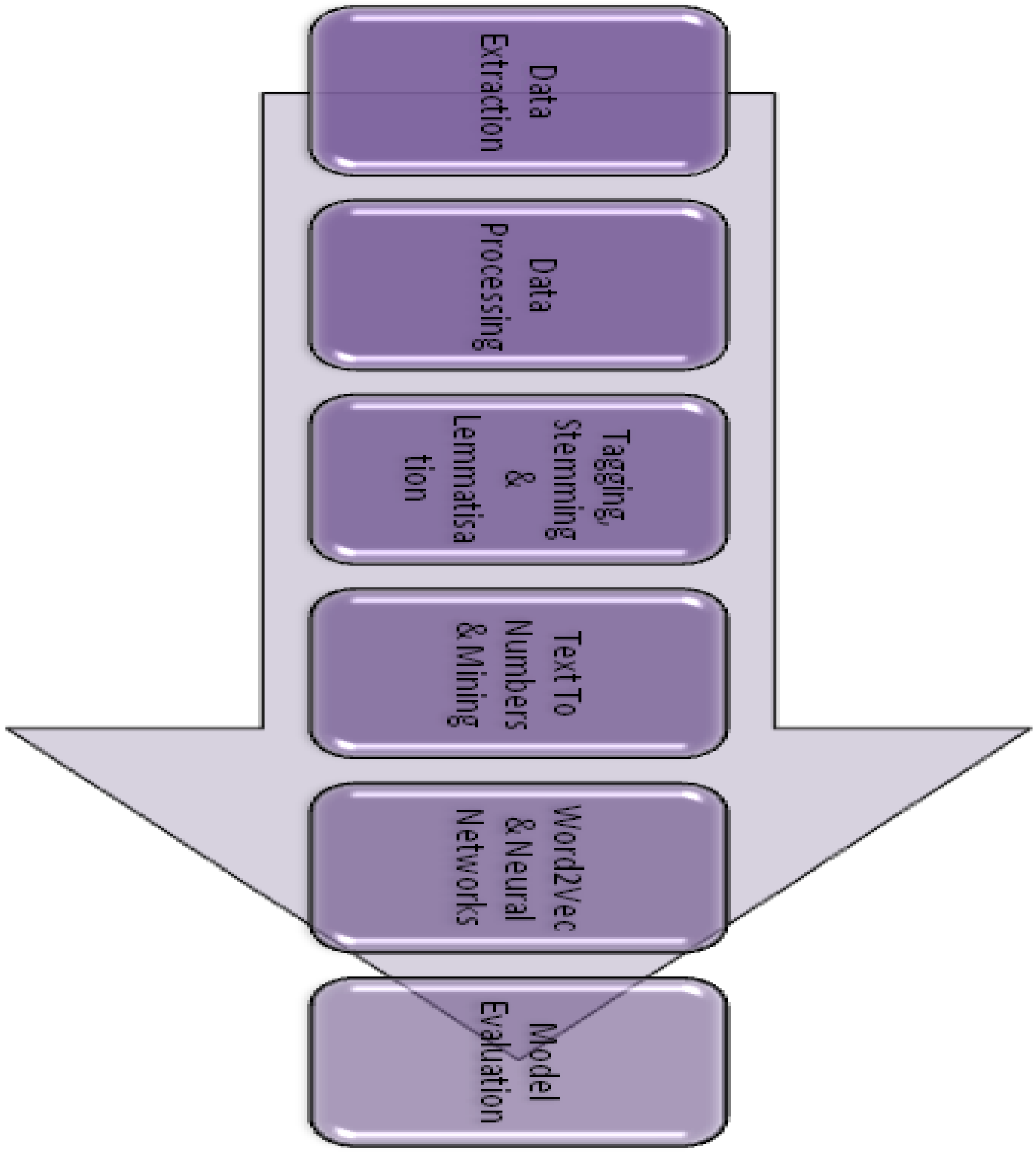


```
print('train data Set description',train.describe())
print('test data Set description',test.describe())
```

```
train data Set descriptin            malignant  highly_malignant           rude         threat  \
count  159571.000000     159571.000000  159571.000000  159571.000000
mean        0.095844          0.009996       0.052948       0.002996
std         0.294379          0.099477       0.223931       0.054650
min         0.000000          0.000000       0.000000       0.000000
25%         0.000000          0.000000       0.000000       0.000000
50%         0.000000          0.000000       0.000000       0.000000
75%         0.000000          0.000000       0.000000       0.000000
max         1.000000          1.000000       1.000000       1.000000

               abuse         loathe
count  159571.000000  159571.000000
mean        0.049364       0.008805
std         0.216627       0.093420
min         0.000000       0.000000
25%         0.000000       0.000000
50%         0.000000       0.000000
75%         0.000000       0.000000
max         1.000000       1.000000
test data Set descriptin                           id                      comment_text
count                 153164                                   153164
unique                153164                                   153164
top     f4c0d3eb97181d79  == ur a knob  == \n\n wikipeda's gay anyway
freq                       1                                        1
```

Here we have described the train and test data where we can see that in train data the minimum value starts from 0 and even they 50% and 75% quantile are also 0

- **Data Preprocessing Done**
  This project is based on NLP and we have comments or string data for that we have to first convert comments into words then filter and cleaning of data using several libraries and finally convert them into int datatype.



1. **Data Extraction:** It is the first step of any model. most data is in CSV or excel format and for analyzing the data we use the data manipulation tool pandas and NumPy after that data is converted into a data frame for analyzing the data.

2. **Data Cleaning:** First we clean the data which have no use in prediction like the S.no column and Id, then we drop the data which has a high no of missing percentages.

3. **Stemming And Lemmatisation:** The next step is to tag the words via Text Part Of Speech Tagging. Additionally, we are ready to perform Stemming And Lemmatisation In The NLP Data Science Project.

4. **Text Mining Algorithms:** The text needs to be converted to numbers. Multiple algorithms compute and use the frequency of the words or group them to help us understand their hidden meanings.

5. **Word2Vec Algorithm:** Now the numerical data needs to be fed into a model so that we can start forecasting it. We can feed our data to a model. Word2Vec algorithm is gaining popularity. Let’s understand how to Predict Text Using

6. **Data transformation** is the process of changing the format, structure, or values of data; we use a labeled encoder for coding the object data into integer data.
7. **Data Reduction:** it is the process of finding the most correlated columns, and combining them because the machine does not understand which feature columns impact the most on accuracy.

8. **Data discretization** converts a large number of data values into smaller once, so that data evaluation and data management becomes very easy, using box plots is makes a clear understanding of the data.

9. **Evaluate NLP Model:** Now that the NLP algorithm has started to forecast text, the last step is about assessing the accuracy of the model. Learn How To Evaluate The Model Performance

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

  Regression and classification models are important tools for researchers in various fields. The application of these many-to-one mapping models is two-fold. First, they can be used for prediction. The output value or class of a (new) case can be predicted by applying the inferred mapping to the input variables of the case. Second, they inform us about the relationship between the input and the output. They specify how the input variables are (mathematically) interacting with each other to produce the output variable. The usefulness of the second application is, however, limited by the power of the human intellect. We suggest that the interpretation of these many-to-one mapping models is of utmost, yet undervalued, importance in many research fields.

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

  As we are trying to predict the malignant in comments and build a filtration model using M.L for predicting whether a comment is malignant or not for this we assume that the bad words are almost the same which makes any comments toxic.

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

  Python is widely used in scientific and numeric computing:
  SciPy is a collection of packages for mathematics, science, and engineering.
  Pandas are data analysis and modeling libraries.

  Natural Language Toolkit: NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries,

Libraries Used for this Project include –

- 1. Pandas
- 2. NumPy
- 3. Matplotlib
- 4. Seaborn
- 5. Scikit Learn
- 6. Nltk
- 7. WordNetLemmatizer
- 8. TF-IDF vactorrization

Model/s Development and Evaluation

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

After analyzing the dataset, I observe that many of the feature columns are int type and coment_text is string type. so first, we have to convert them into an integer so that the machine interprets the data and for that, we use the NLP toolkit for all the features columns.

Then find the correlation between the columns with target columns and delete the non-related feature columns.
We observe that the target column is skewed.
After converting text into int datatype and classes are defined.
The target column is classified so we start work on Classification models building.

- **Testing of Identified Approaches (Algorithms)**

List down all the algorithms used for the training and testing.

1. Logistic Regression
2. DecisionTreeClassifier
3. Random forest Regression.
4. Xgboost
5. AdaBoostClassifier
6. KNeighborsClassifier

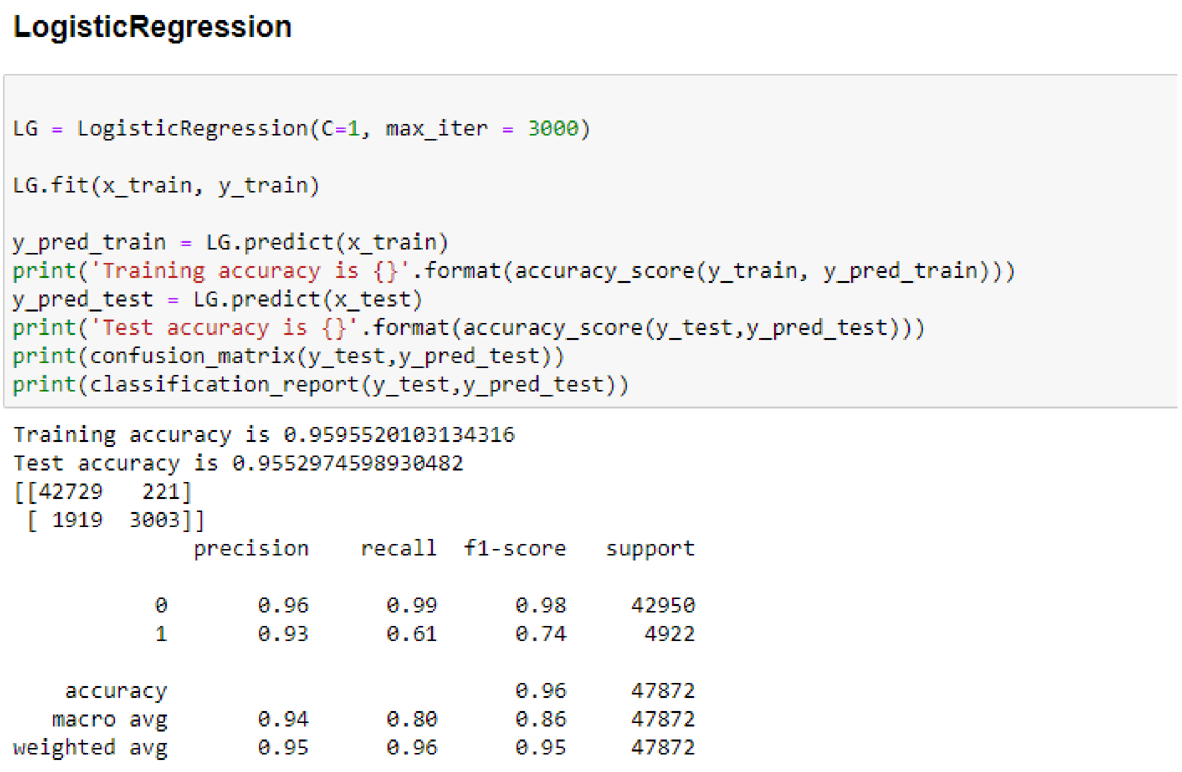
• Run and Evaluate selected models

Logistic Regression Model

• Logistic Regression is a machine learning algorithm based on supervised learning.

• It performs a regression task. Regression models a target prediction value based on independent variables.

• It is mostly used for finding out the relationship between variables and forecasting.



LogisticRegression

```
LG = LogisticRegression(C=1, max_iter = 3000)

LG.fit(x_train, y_train)

y_pred_train = LG.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = LG.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9595520103134316
Test accuracy is 0.9552974598930482
[[42729   221]
 [ 1919  3003]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.93      0.61      0.74      4922

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

Conclusion of the Logistic Regression.

Observations:

1. This Logistic Regression Performs with 96% accuracy for predicting labels.
2. We use the best-fit line.
3. from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

```
[[28459   157]
 [ 1251  2048]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     28616
           1       0.93      0.62      0.74      3299

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

from above we easily find out that

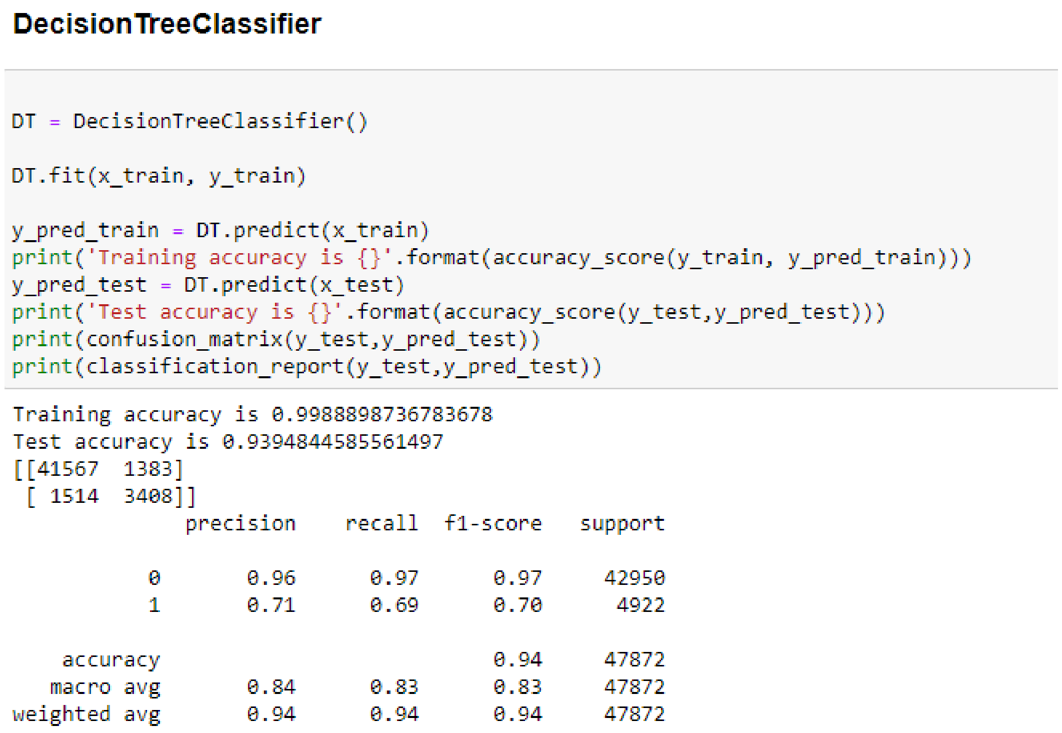
Precision, recall,fl-score from the above plotting.

Our model performs well on the initial level

2. DecisionTreeClassifier

DecisionTreeClassifier is a class capable of performing multi-class classification on a   dataset.

As with other classifiers, DecisionTreeClassifier takes as input two arrays: an array X,   sparse or dense, of shape (n_samples, n_features) holding the training samples, and an array Y of integer values, shape (n_samples,), holding the class labels for the training samples: `



DecisionTreeClassifier

```
DT = DecisionTreeClassifier()

DT.fit(x_train, y_train)

y_pred_train = DT.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = DT.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988898736783678
Test accuracy is 0.9394844585561497
[[41567  1383]
 [ 1514  3408]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.71      0.69      0.70      4922

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.83     47872
weighted avg       0.94      0.94      0.94     47872
```

Observations:

1. This Decision Tree classifier Performs with 94% accuracy for predicting frauds.
2. After predicting and plotting the predicted data on the best fit line we observe that DT-C is not so accurate.
3. CV is not well. And does not give accurate results.

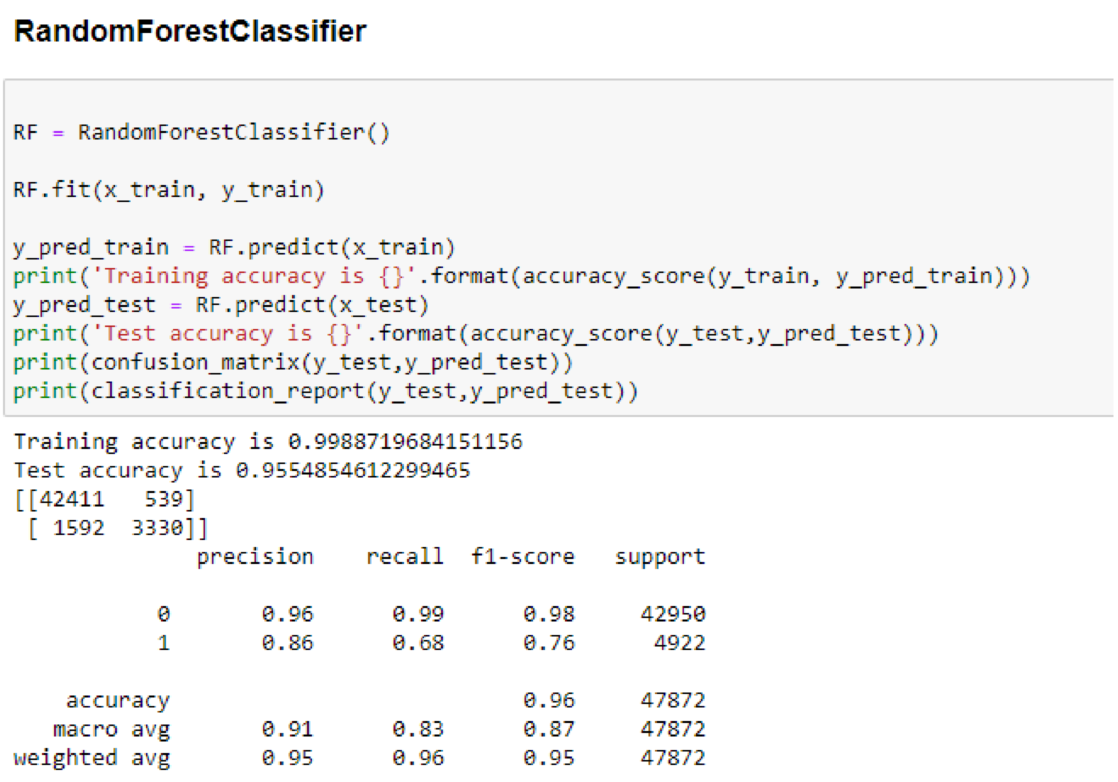
```
print(accuracy_score(y_test,pred_decision))
print(confusion_matrix(y_test,pred_decision))
print(classification_report(y_test,pred_decision))
```

```
[[27719   897]
 [ 1012  2287]]
              precision    recall  f1-score   support
```

```
           0       0.96      0.97      0.97     28616
           1       0.72      0.69      0.71      3299

    accuracy                           0.94     31915
   macro avg       0.84      0.83      0.84     31915
weighted avg       0.94      0.94      0.94     31915

0.9401848660504465 accuracy score.
```

RandomForestClassifier

A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.



RandomForestClassifier

```
RF = RandomForestClassifier()

RF.fit(x_train, y_train)

y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

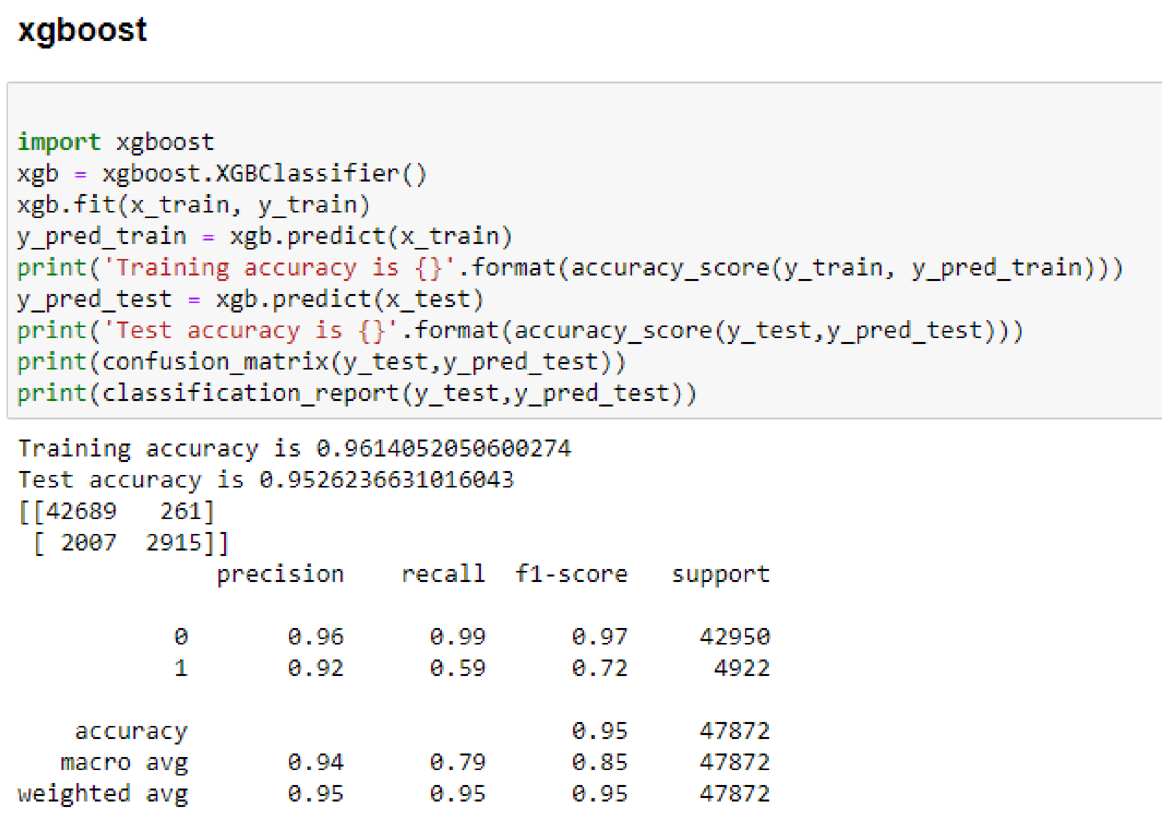
```
Training accuracy is 0.9988719684151156
Test accuracy is 0.9554854612299465
[[42411   539]
 [ 1592  3330]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.68      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.83      0.87     47872
weighted avg       0.95      0.96      0.95     47872
```

Conclusion:

```
[[28239   377]
 [ 1063  2236]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     28616
           1       0.86      0.68      0.76      3299

    accuracy                           0.95     31915
   macro avg       0.91      0.83      0.87     31915
weighted avg       0.95      0.95      0.95     31915

0.9548801503994987 Accuracy Score
Random forest performs the same as DTC.
```

<u>XGBoost Classifier</u>

XGBoost provides a wrapper class to allow models to be treated like classifiers or regressors in the sci-kit-learn framework. This means we can use the full sci-kit-learn library with XGBoost models. The XGBoost model for classification is called **XGBClassifier**.We can create and fit it into our training dataset. Models are fit using the sci-kit-learn API and the **model.fit()** function.Parameters for training the model can be passed to the model in the constructor.



xgboost

```
import xgboost
xgb = xgboost.XGBClassifier()
xgb.fit(x_train, y_train)
y_pred_train = xgb.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = xgb.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9614052050600274
Test accuracy is 0.9526236631016043
[[42689   261]
 [ 2007  2915]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     42950
           1       0.92      0.59      0.72      4922

    accuracy                           0.95     47872
   macro avg       0.94      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872
```

```
[[28431   185]
 [ 1333  1966]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     28616
           1       0.91      0.60      0.72      3299

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

```
0.9524361585461382 Accuracy score
```

- **<u>Key Metrics for success in solving the problem under consideration</u>**

  Confusion matrix, Mean Absolute Error, Mean Squared Error, Root Mean Square Error

  This matrix helps to understand the model more deeply.

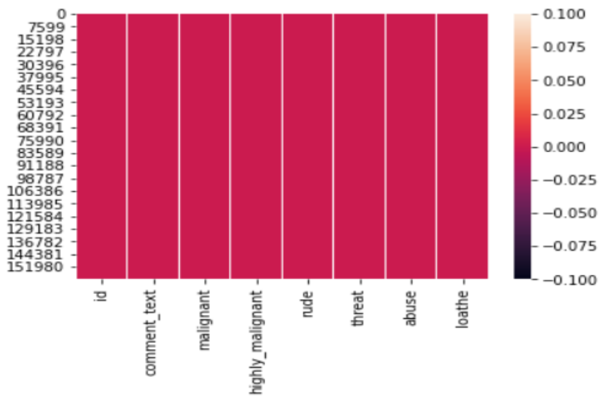
- **<u>Visualizations</u>**

  Data visualization is the graphical representation of information and data. By using charts, plots, and graphs data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

  In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

```
# checking null values
print(train.isnull().sum())
print(sns.heatmap(train.isnull()))
```

```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
AxesSubplot(0.125,0.125;0.62x0.755)
```

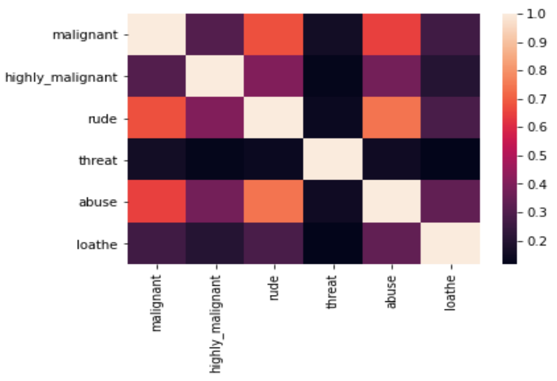


As clearly we analyze that no-null values are present.

Correlation:

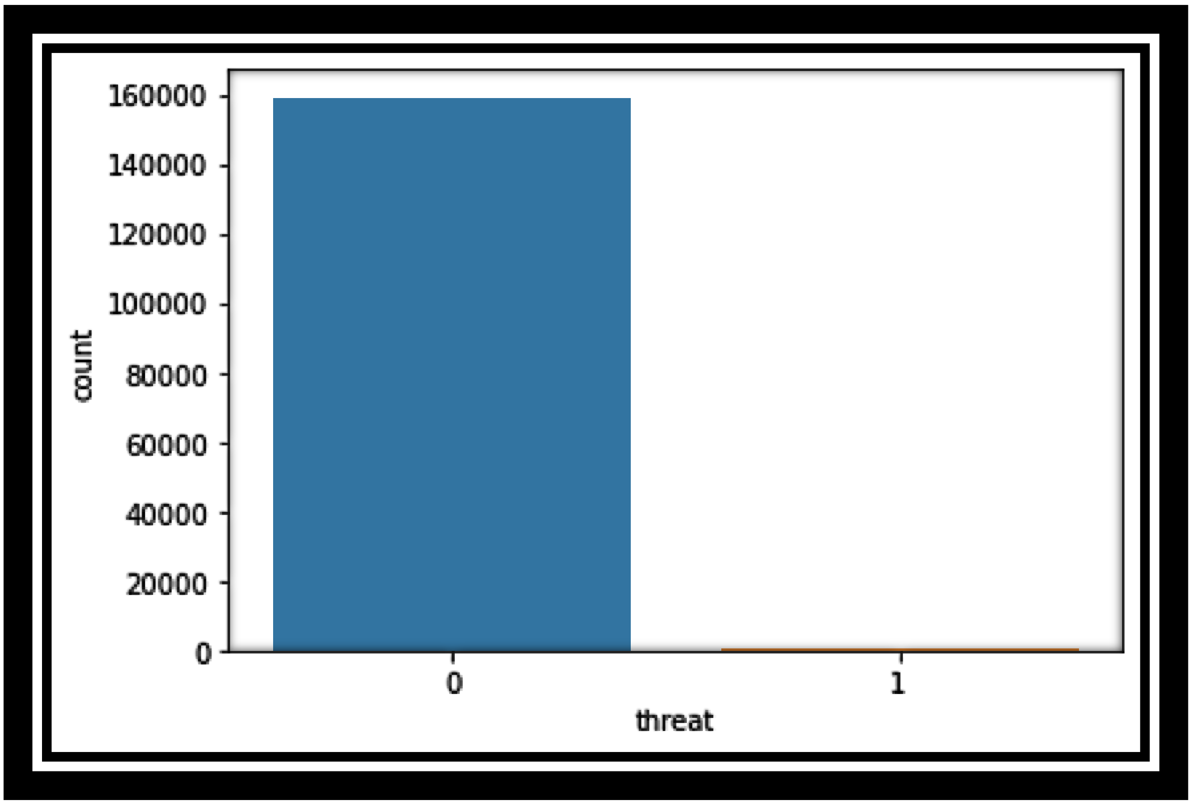
```
print(train.corr())
print(sns.heatmap(train.corr()))
```

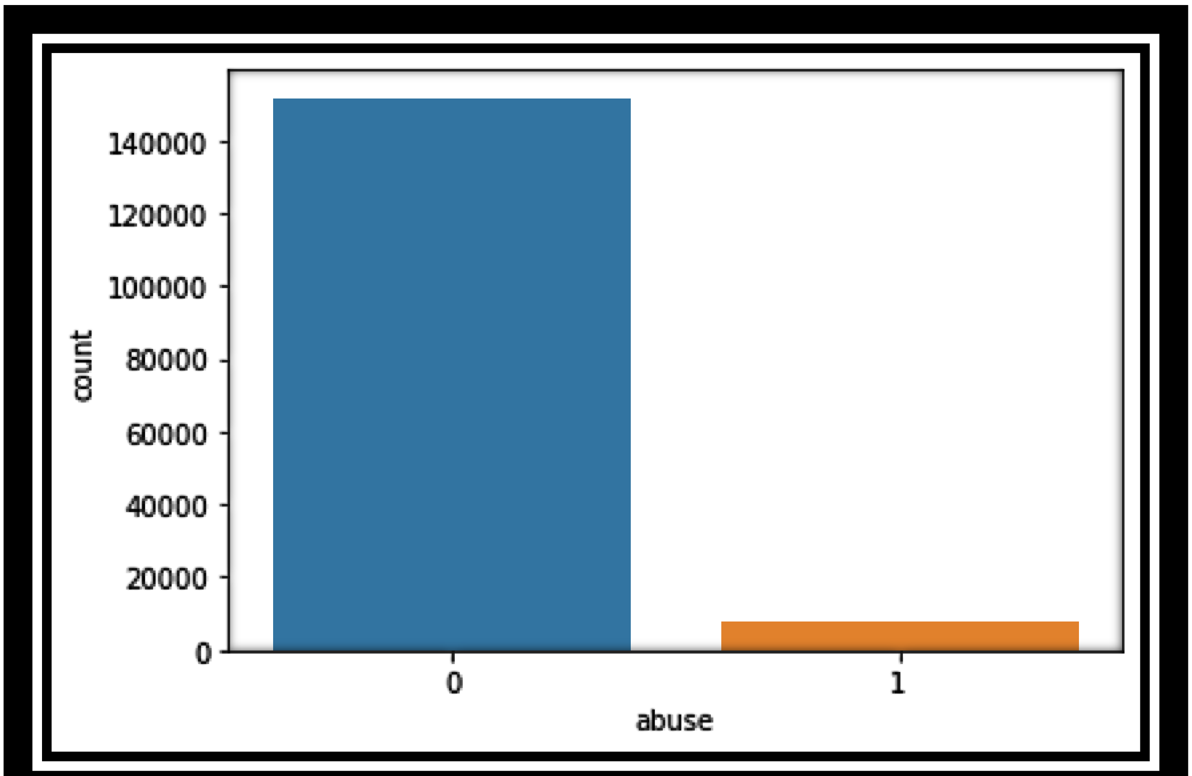
```
                  malignant  highly_malignant      rude    threat     abuse  \
malignant          1.000000          0.308619  0.676515  0.157058  0.647518
highly_malignant   0.308619          1.000000  0.403014  0.123601  0.375807
rude               0.676515          0.403014  1.000000  0.141179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

                    loathe
malignant         0.266009
highly_malignant  0.201600
rude              0.286867
threat            0.115128
abuse             0.337736
loathe            1.000000
AxesSubplot(0.125,0.125;0.62x0.755)
```

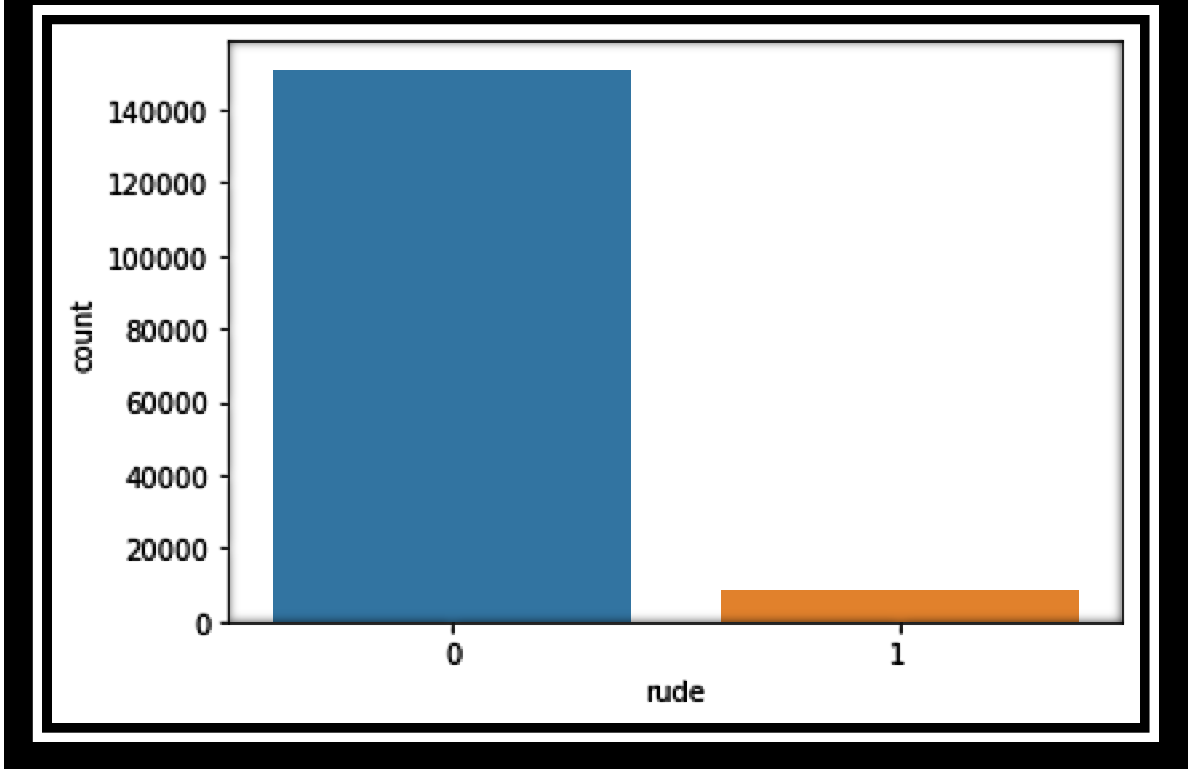


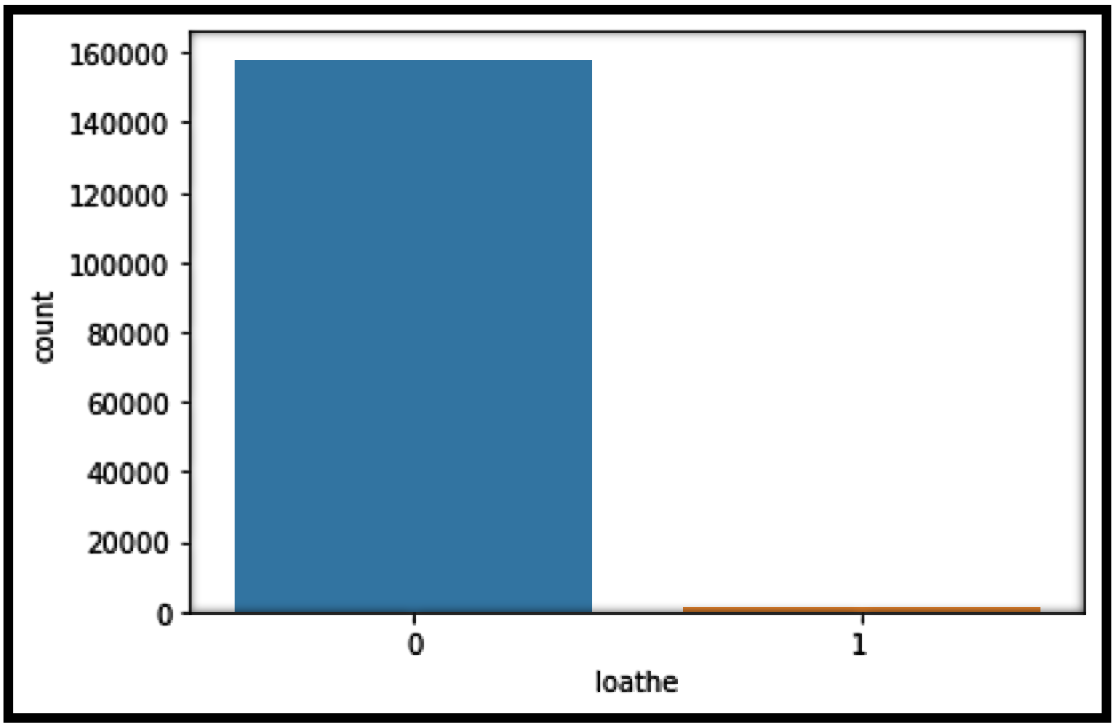
No feature columns are negatively co-related with target columns.

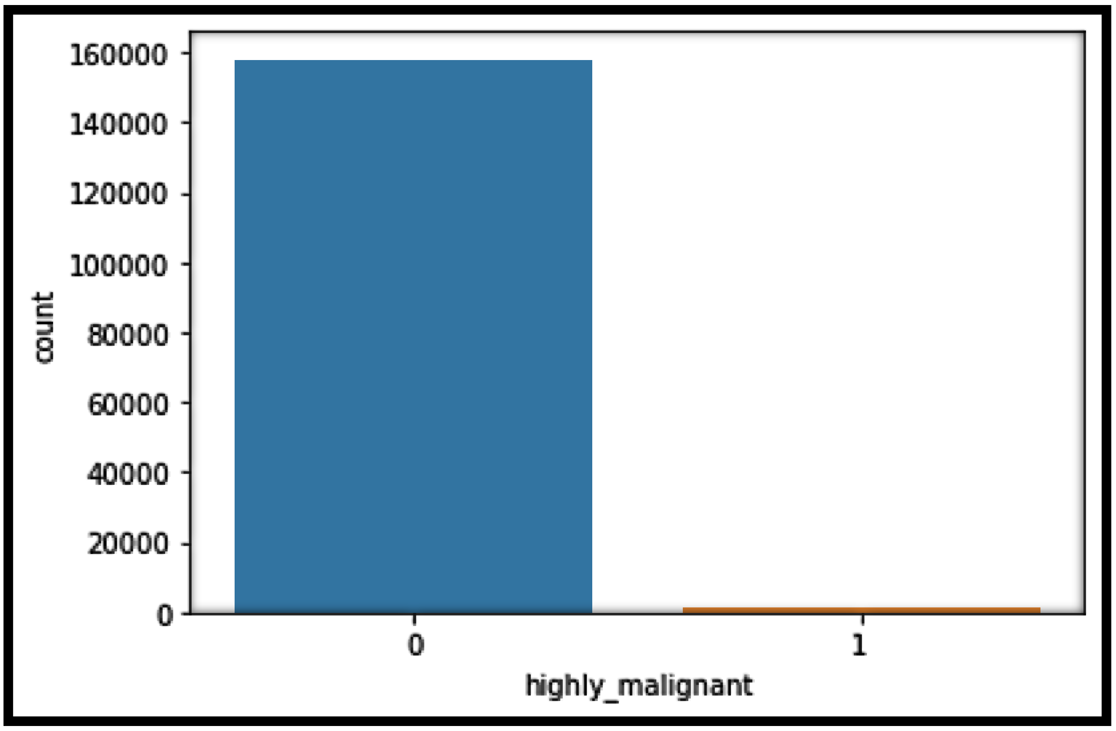
Abuse is highly related to malignant

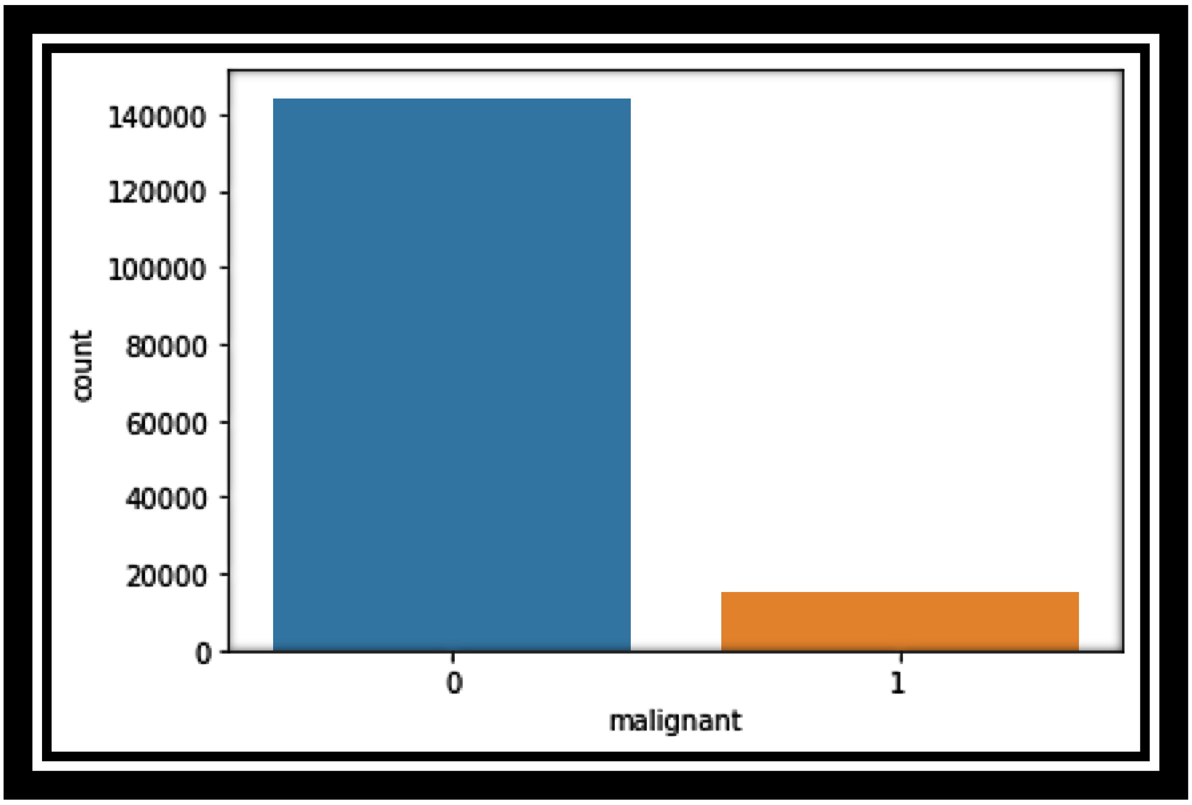



count
140000
120000
100000
80000
60000
40000
20000
0
0
1
abuse


count
140000
120000
100000
80000
60000
40000
20000
0
0
1
rude


160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
loathe


160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
highly_malignant

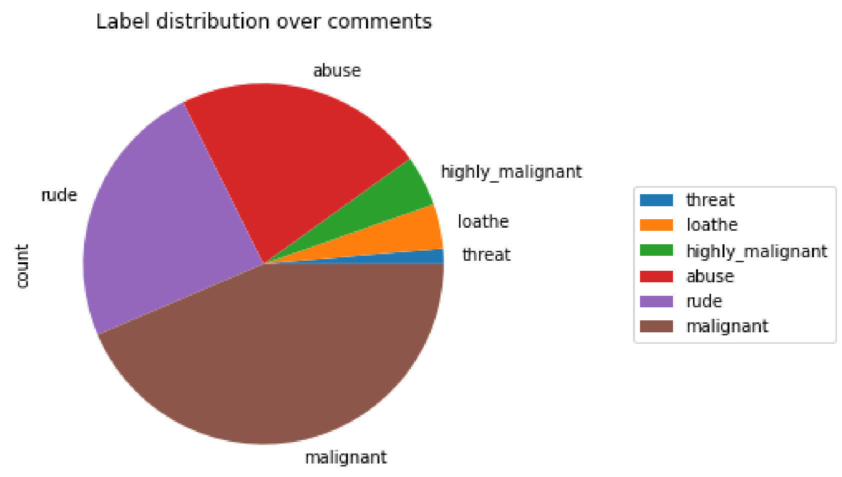


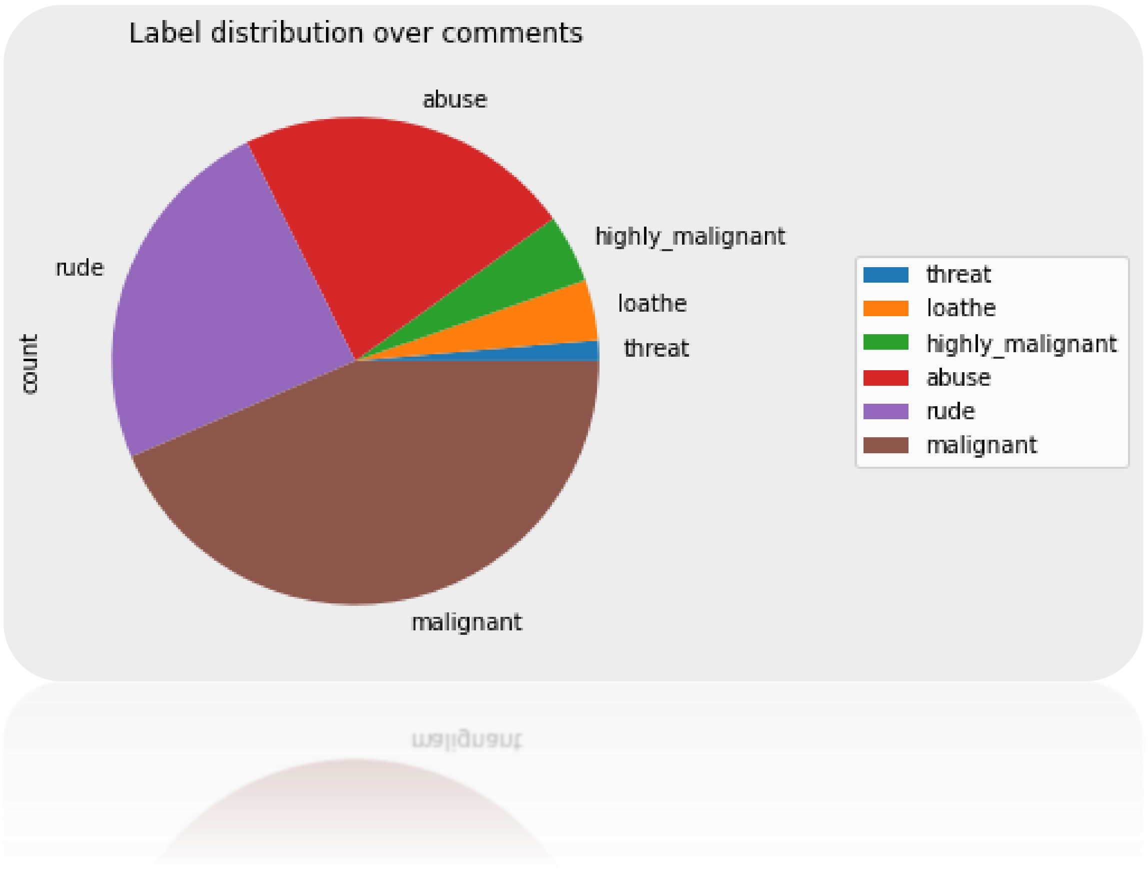
Distribution plotting of different features columns.

And from the above plotting, it is mentioned that the count of normal comments is more than as compared to toxic comments.

```
cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df_distribution = train[cols_target].sum()\
                            .to_frame()\
                            .rename(columns={0: 'count'})\
                            .sort_values('count')

df_distribution.plot.pie(y='count',
                                      title='Label distribution over comments',
                                      figsize=(5, 5))\
                            .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

```
<matplotlib.legend.Legend at 0x225e9c71d60>
```





Plotting of sections of different feature columns and we observe that malignant comments are more as compared to remaining one.

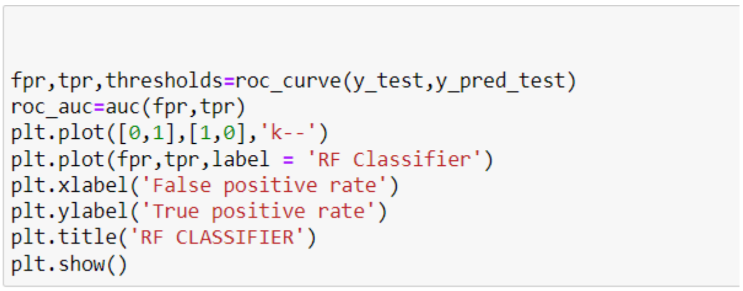
- **Interpretation of the Results**

List of accuracy scores of different classification models

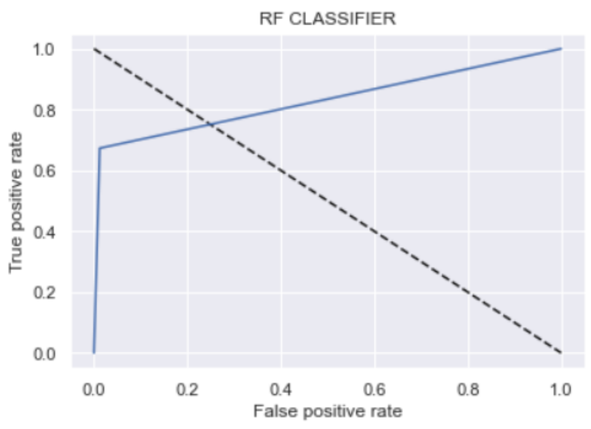
logistic Regression:- 0.96

Decision Tree Classifier:- 0.94

Random Forest classifier:- 0.96
xgboost:- 0.95
AdaBoostClassifier:- 0.95
KNeighborsClassifier:- 0.92



```
fpr,tpr,thresholds=roc_curve(y_test,y_pred_test)
roc_auc=auc(fpr,tpr)
plt.plot([0,1],[1,0],'k--')
plt.plot(fpr,tpr,label = 'RF Classifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('RF CLASSIFIER')
plt.show()
```



observations:

1. Roc curve area is 0.9 which means that our model is distingui -shed between the malignant comment or not.

2. our model understands that label 0 is 96% different than label 1, which is good.

3. It means there is a 96% chance that the model will be able to d istinguish between positive class and negative class.

CONCLUSION

We tested out different models for training and testing on our given dataset and after all the training and testing on different models we find out that <u>Random Forest Regression</u> performed above all.

- **Key Findings and Conclusions of the Study**

  So, our Aim is achieved as we have successfully ticked
  all our parameters as mentioned in our Aim Column. It is seen
  that nltk libraries help us to find the outcomes. All the feature colum ns are positively co-related.
  Our model accuracy is over 96% which is good in terms of initial mo del building.

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

  Throughout this kernel, we put into practice many of the strategies for predicting whether comments are malignant or not. We philosophized about the variables, we analyzed 'Comment _text' alone and with the most correlated variables, we tested some of the fundamental statistical assumptions and we even transformed text data into int32 type. That's a lot of work that Python helped us make easier.

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

  As we do lots of research and our data set is quite big which helps us to find different toxic words from around the world which makes this study successful still as generations are evolved and continuously discovering new slag words which make comments toxic. we don’t rely on this data for too long continuously adding more stopwords in the nltk library for helping us to make more powerful models and make us future-ready.