FLIP ROBO

Malignant Comments Classifier

Submitted by:

Yashna Shah

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to Sapna Verma, my SME (Subject Matter Expert), as well as Flip Robo Technologies, for allowing me to work on this project on Malignant Comments Classifier and for assisting me in conducting extensive research that allowed me to learn a lot of new things, particularly in the Natural Language Processing and Natural Language Toolkit sections.

In addition, I used a few other resources to assist me in working on the project. I made sure to learn from the samples and adjust things to fit my project's needs. The following are all of the external resources that were utilised to create this project:

1) https://www.google.com/

2) https://www.youtube.com/

3) https://scikit-learn.org/stable/user_guide.html

4) https://github.com/

5) https://www.kaggle.com/

6) https://medium.com/

7) https://towardsdatascience.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 that insults to third parties such as celebrities will be tagged as unoffensive, but "u are an idiot" is clearly offensive.
  Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

Conceptual Background of the Domain Problem

Online platforms and social media have evolved into places where individuals may freely discuss their beliefs without regard for race, and where people can share their thoughts and ideas with a large group of people.
Social media is Internet-based by design, allowing people to share material quickly via electronic means. Personal information, documents, movies, and images are all included in the content. Users interact with social media using web-based software or applications on a computer, tablet, or smartphone.

Some people on this massive internet platform or online community wilfully abuse others to prevent them from sharing their thoughts in a proper manner. They use filthy language to intimidate others, which is considered a form of ignominy in civilised society. When innocent people are intimidated by these mobs, they remain mute without saying anything. As a result, the disgusting mob's goal is excellently realised.

To address this issue, we are now developing a model that recognises all foul language and foul terms, with the goal of preventing these mobs from using foul language in online communities or perhaps blocking them from using foul language altogether.

Review of Literature

The purpose of the literature review is to:
1. Identify the foul words or foul statements that are being used.
2. Stop the people from using these foul languages in online public forum.

To solve this problem, we are now building a model using our machine language technique that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language. Comments on the internet are hotbeds of hate and venom. Machine learning may be used to combat online anonymity, which has created a new venue for hostility and hate speech. The issue we were attempting to address was the labelling of internet remarks that were hostile to other users.

Our objective is to create a prototype of an online hate and abuse comment classifier that can be used to categorise and manage hate and offensive remarks in order to prevent the spread of hatred and cyberbullying.

Motivation for the Problem Undertaken

With widespread usage of online social networks and its popularity, social networking platforms have given us incalculable opportunities than ever before, and its benefits are undeniable. Despite benefits, people may be humiliated, insulted, bullied, and harassed by anonymous users, strangers, or peers. In this study, we have proposed a cyberbullying detection framework to generate features from online content by leveraging a point wise mutual information technique. Based on these features, we developed a supervised machine learning solution for cyberbullying detection and multi-class categorization of its severity. Results from experiments with our proposed framework in a multi-class setting are promising both with respect to classifier accuracy and f-measure metrics. These results indicate that our proposed framework provides a feasible solution to detect cyberbullying behaviour and its severity in online social networks.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

Here in this project, we have been provided with two datasets namely train and test CSV files. I will build a machine learning model by using NLP using train dataset. And using this model we will make predictions for our test dataset. I will need to build multiple classification machine learning models. Before model building will need to perform all data pre-processing steps involving NLP. After trying different classification models with different hyper parameters then will select the best model out of it. Will need to follow the complete life cycle of data science that includes steps like -

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

Finally, we compared the results of proposed and baseline features with other machine learning algorithms. Findings of the comparison indicate the significance of the proposed features in cyberbullying detection.

Data Sources and their formats

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.

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 good amount of data exploration and derive some interesting features using the comments text column available. You need to build a model that can differentiate between comments and its categories.

• Data Preprocessing Done

The following pre-processing pipeline is required to be performed before building the classification model prediction:

1. Load dataset

2. There were no null values in our dataset.

3. Drop column id

4. Convert comment text to lower case and replace '\n' with single space.

5. Keep only text data ie. a-z' and remove other data from comment text.

6. Remove stop words and punctuations

7. Apply Stemming

8. Convert text to vectors using TfidfVectorizer

9. Saving the model

10. Predict values for multi class label

• Data Inputs- Logic- Output Relationships

I have analysed the input output logic with word cloud and I have word clouded the sentenced that as classified as foul language in every category. A tag/word cloud is a novelty visual representation of text data, typically used to depict keyword metadata on websites, or to visualize free form text. It’s an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance. These are the comments that belongs to different type so which the help of word cloud we can see if there is abuse comment which type of words it contains and similar to other comments as well.

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

Cyberbullying has become a growing problem in countries around the world. Essentially, cyberbullying doesn't differ much from the type of bullying that many children have unfortunately grown accustomed to in school. The only difference is that it takes place online.

- ## Hardware and Software Requirements and Tools Used

Hardware technology being used.

RAM	: 16 GB
CPU	: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
GPU	: intel Iris Graphics

Software technology being used.

Programming language	: Python
Distribution	: Anaconda Navigator
Browser based language shell	: Jupyter Notebook

Libraries/Packages specifically being used.
Pandas, NumPy, matplotlib, seaborn, scikit-learn, NLTK

Model/s Development and Evaluation

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

1. I checked through the entire training dataset for any kind of missing values information and all these pre - processing steps were repeated on the testing dataset as well.
2. Then we went ahead and took a look at the dataset information. Using the info method, we are able to confirm the non-null count details as well as the datatype information. We have a total of 8 columns out of which 2 columns have object datatype while the remaining 6 columns are of integer datatype.
3. Since there was no use of the "id" column I have dropped it.
4. Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP).

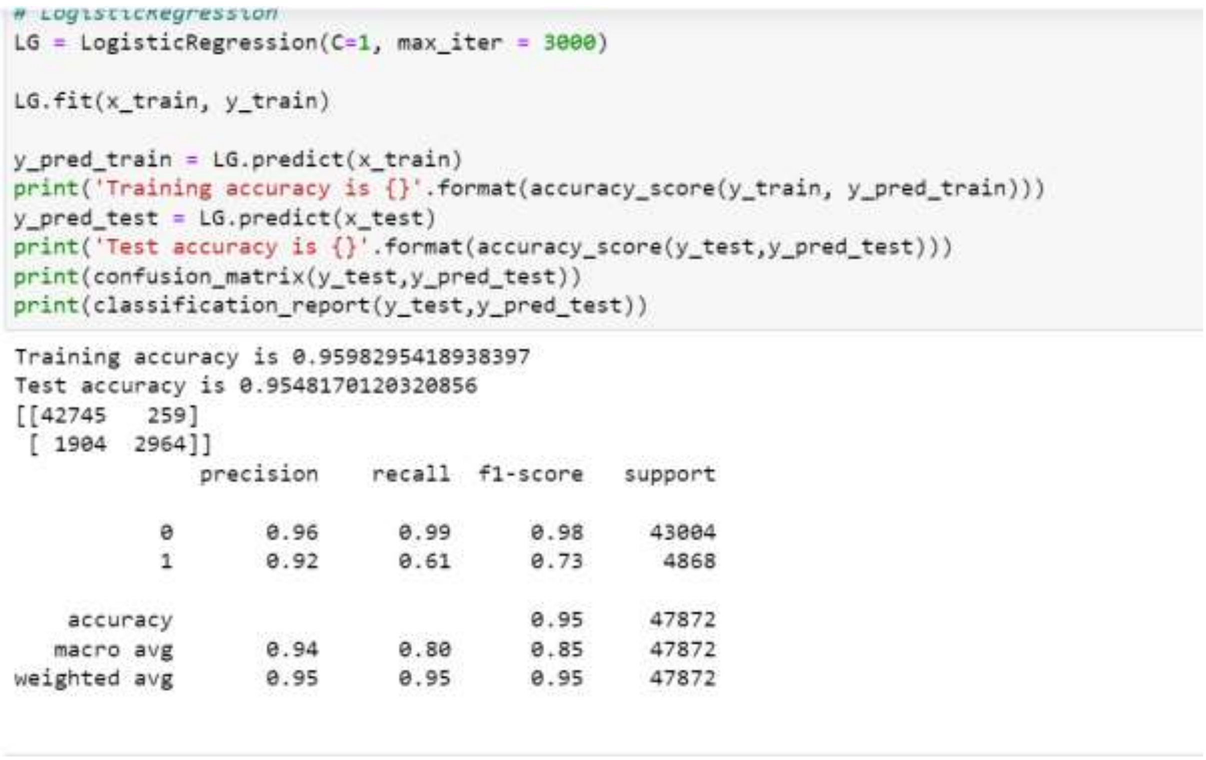
- ### Testing of Identified Approaches (Algorithms)

The complete list of all the algorithms used for the training and testing classification model are listed below:

1) Logistic Regression
2) Random Forest Classifier
3) Ada Boost Classifier
4) K Nearest Neighbors Classifier
5) Decision Tree Classifier

- ### Run and Evaluate selected models

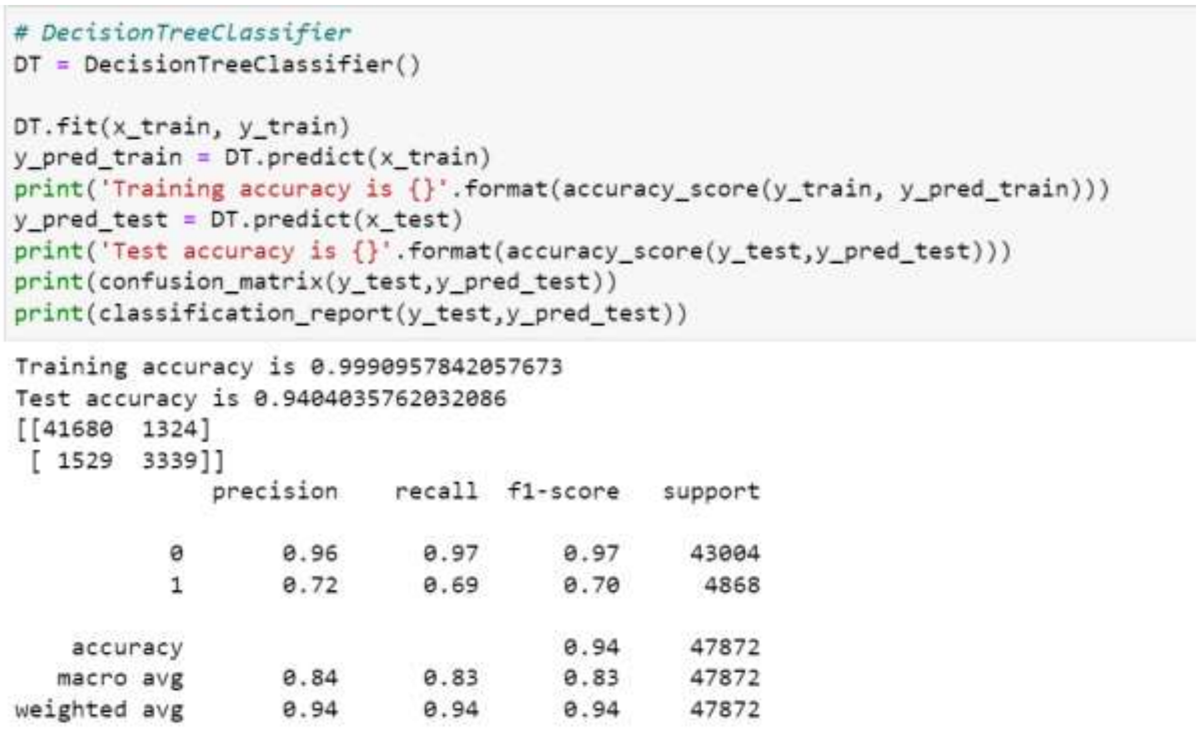
1. Logistic Regression



```
# 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 0.9598295418938397
Test accuracy is 0.9548170120320856
[[42745   259]
 [ 1904  2964]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     43004
           1       0.92      0.61      0.73      4868

    accuracy                           0.95     47872
   macro avg       0.94      0.80      0.85     47872
weighted avg       0.95      0.95      0.95     47872
```

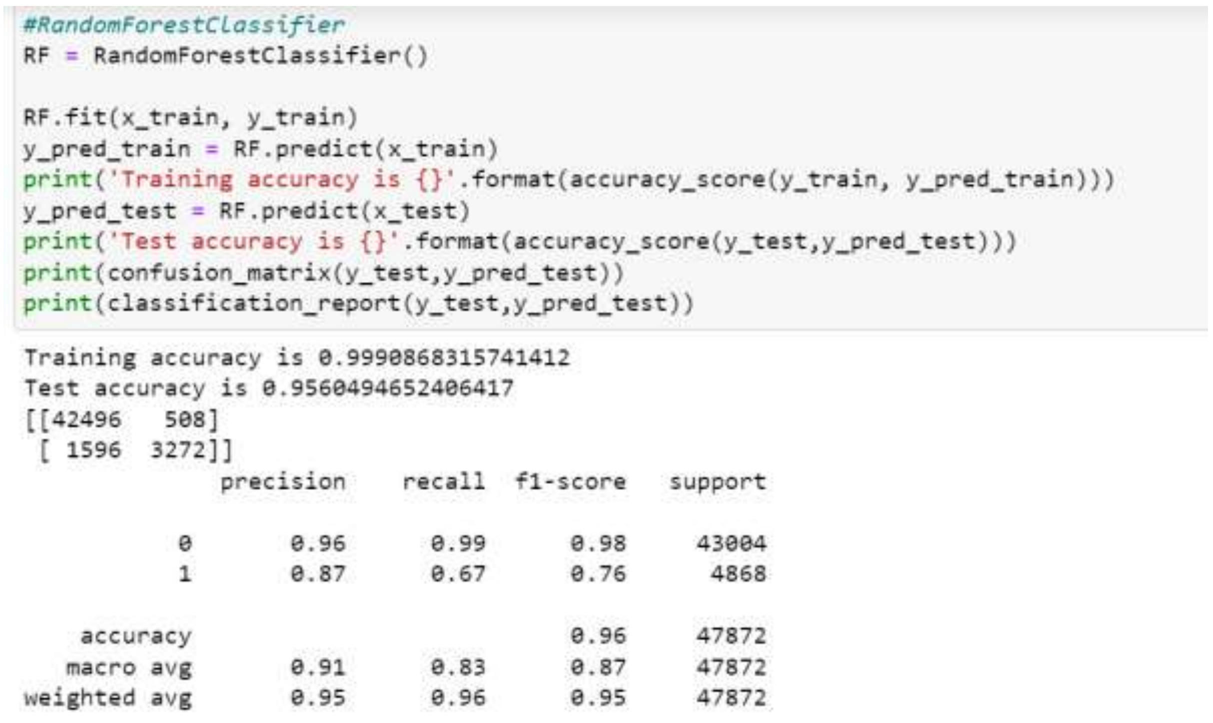
2. Decision Tree Classifier



```
# 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 0.9990957842057673
Test accuracy is 0.9404035762032086
[[41680  1324]
 [ 1529  3339]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     43004
           1       0.72      0.69      0.70      4868

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

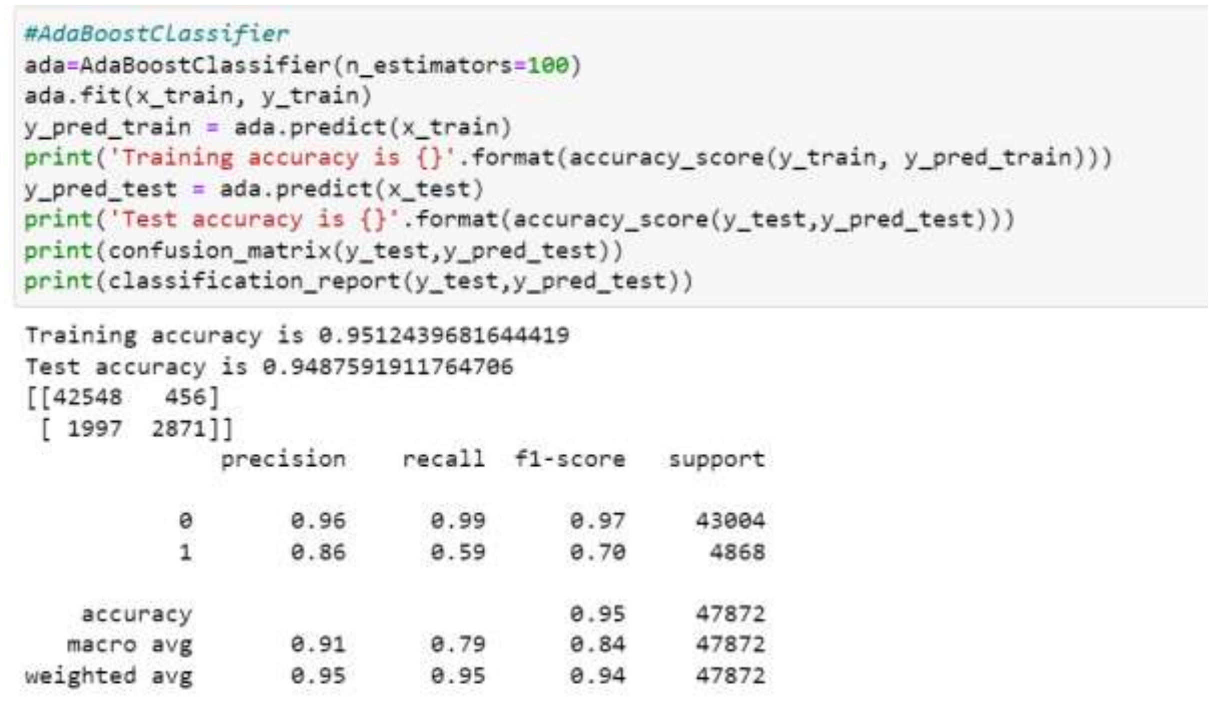
3. Random Forest Classifier



```
#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 0.9990868315741412
Test accuracy is 0.9560494652406417
[[42496   508]
 [ 1596  3272]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     43004
           1       0.87      0.67      0.76      4868

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

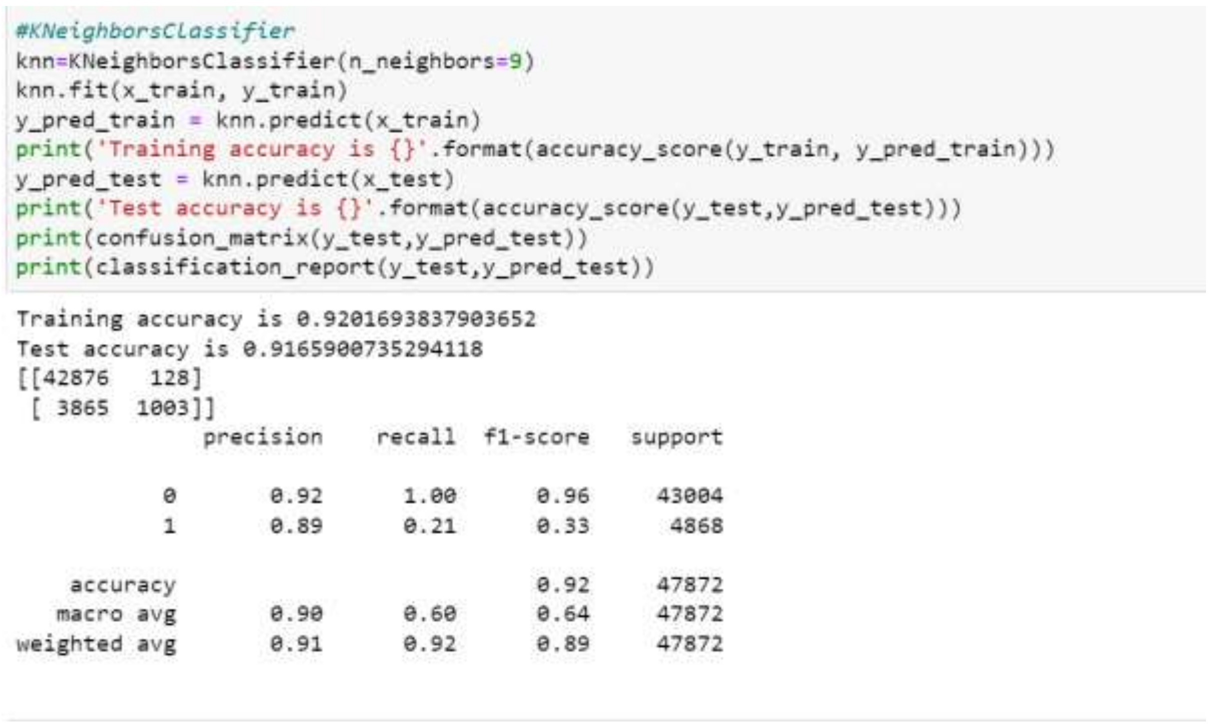
4. Ada Boost Classifier



```
#AdaBoostClassifier
ada=AdaBoostClassifier(n_estimators=100)
ada.fit(x_train, y_train)
y_pred_train = ada.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = ada.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 0.9512439681644419
Test accuracy is 0.9487591911764706
[[42548   456]
 [ 1997  2871]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.86      0.59      0.70      4868

    accuracy                           0.95     47872
   macro avg       0.91      0.79      0.84     47872
weighted avg       0.95      0.95      0.94     47872
```

5. KNN Classifier

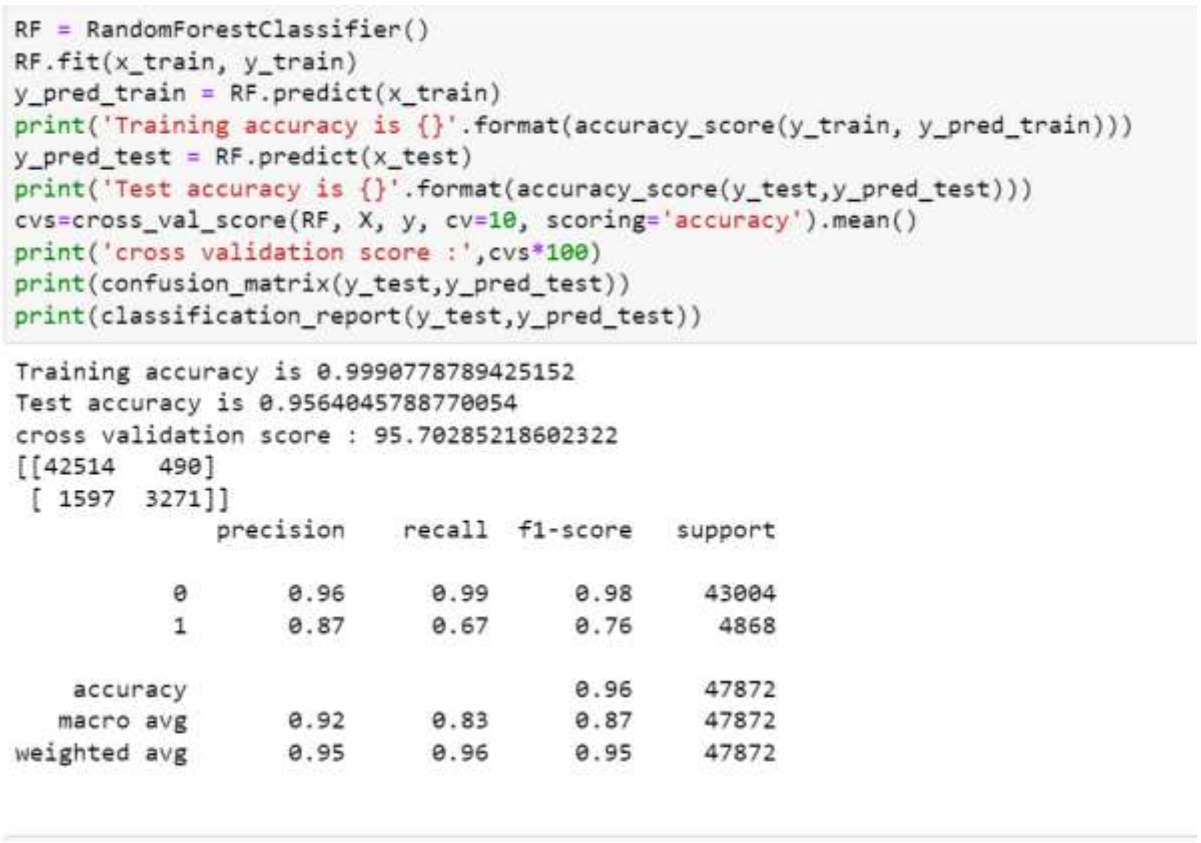

```
#KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=9)
knn.fit(x_train, y_train)
y_pred_train = knn.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = knn.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9201693837903652
Test accuracy is 0.9165900735294118
[[42876   128]
 [ 3865  1003]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     43004
           1       0.89      0.21      0.33      4868

    accuracy                           0.92     47872
   macro avg       0.90      0.60      0.64     47872
weighted avg       0.91      0.92      0.89     47872
```

Hence, here after evaluating all the models, we can see that Random Forest gives the best accuracy score.

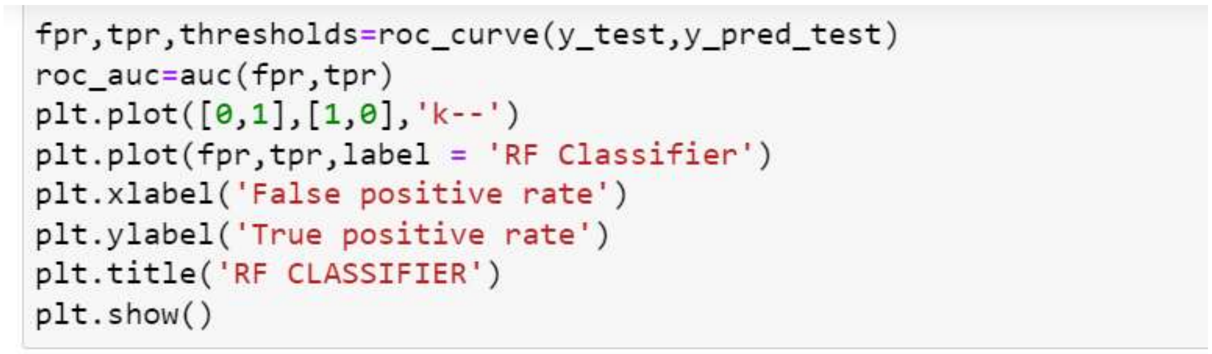
- ## Key Metrics for success in solving problem under consideration

1. Cross Validation

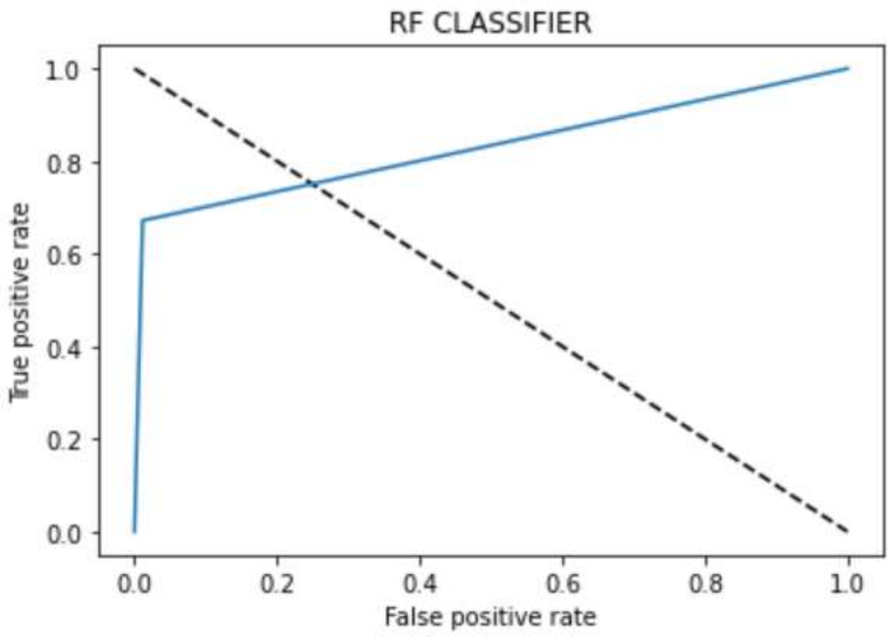

```
RF = RandomForestClassifier()
RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
cvs=cross_val_score(RF, X, y, cv=10, scoring='accuracy').mean()
print('cross validation score :',cvs*100)
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9990778789425152
Test accuracy is 0.9564045788770054
cross validation score : 95.70285218602322
[[42514   490]
 [ 1597  3271]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     43004
           1       0.87      0.67      0.76      4868

    accuracy                           0.96     47872
   macro avg       0.92      0.83      0.87     47872
weighted avg       0.95      0.96      0.95     47872
```

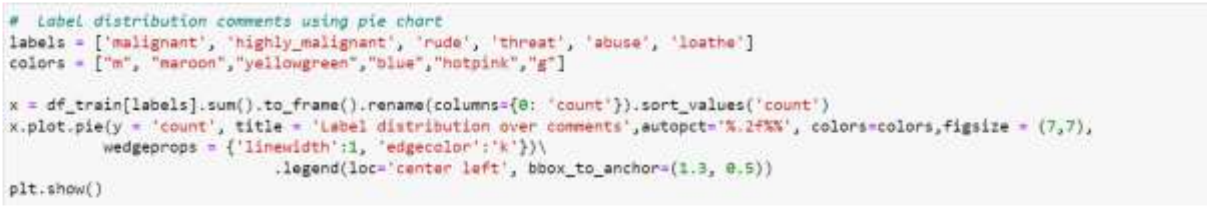
2. AUC ROC Curve



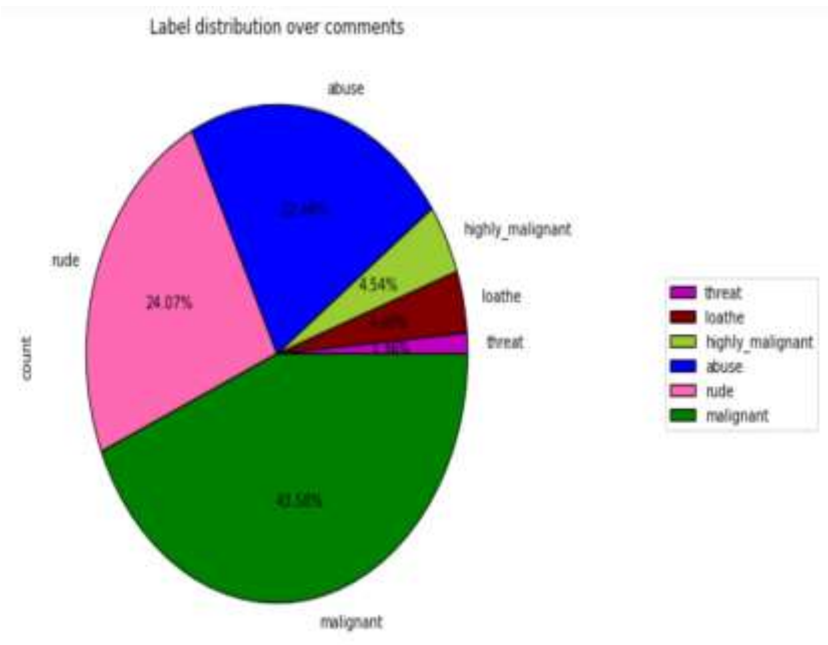
```
fpr,tpr,thresholds=roc_curve(y_test,y_pred_test)
roc_auc=auc(fpr,tpr)
plt.plot([0,1],[1,0],'k--')
plt.plot(fpr,tpr,label = 'RF Classifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('RF CLASSIFIER')
plt.show()
```



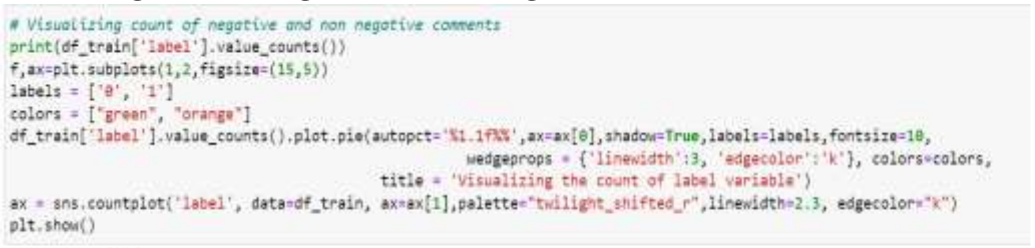
- Visualizations
  1. Label distribution comments using Pie Chart



```
#  Label distribution comments using pie chart
labels = ['malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe']
colors = ["m", "maroon","yellowgreen","blue","hotpink","g"]

x = df_train[labels].sum().to_frame().rename(columns={0: 'count'}).sort_values('count')
x.plot.pie(y = 'count', title = 'Label distribution over comments',autopct='%.2f%%', colors=colors,figsize = (7,7),
          wedgeprops = {'linewidth':1, 'edgecolor':'k'})\
                              .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
plt.show()
```

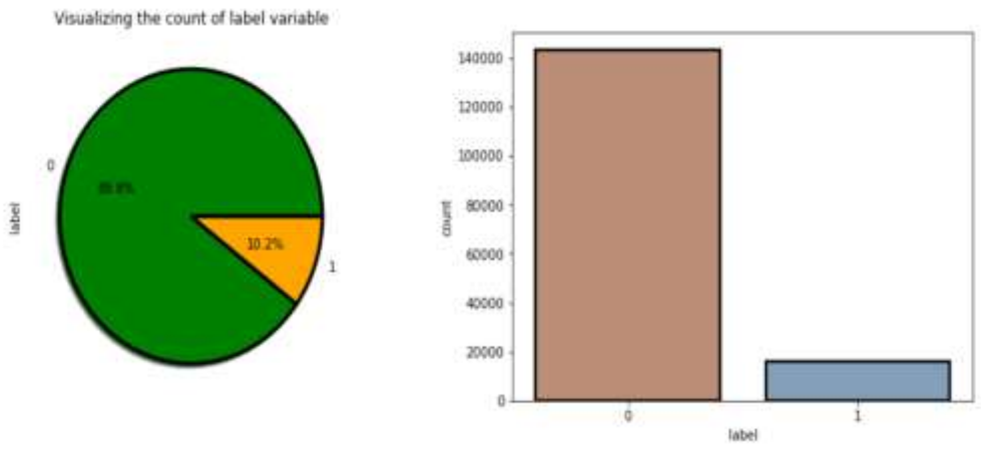


2. Visualizing count of negative and non negative comments

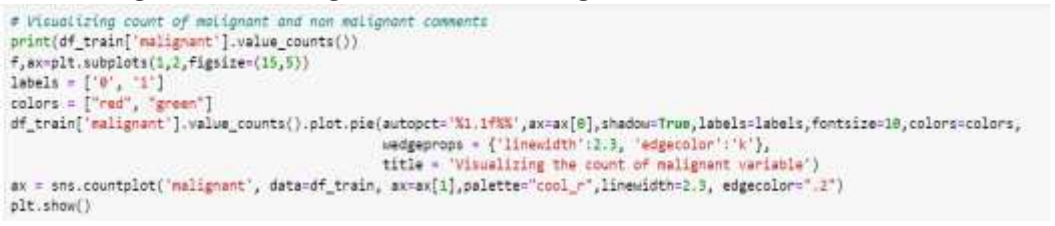


```
# Visualizing count of negative and non negative comments
print(df_train['label'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['0', '1']
colors = ["green", "orange"]
df_train['label'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,
                                           wedgeprops = {'linewidth':3, 'edgecolor':'k'}, colors=colors,
                                  title = 'Visualizing the count of label variable')
ax = sns.countplot('label', data=df_train, ax=ax[1],palette="twilight_shifted_r",linewidth=2.3, edgecolor="k")
plt.show()
```

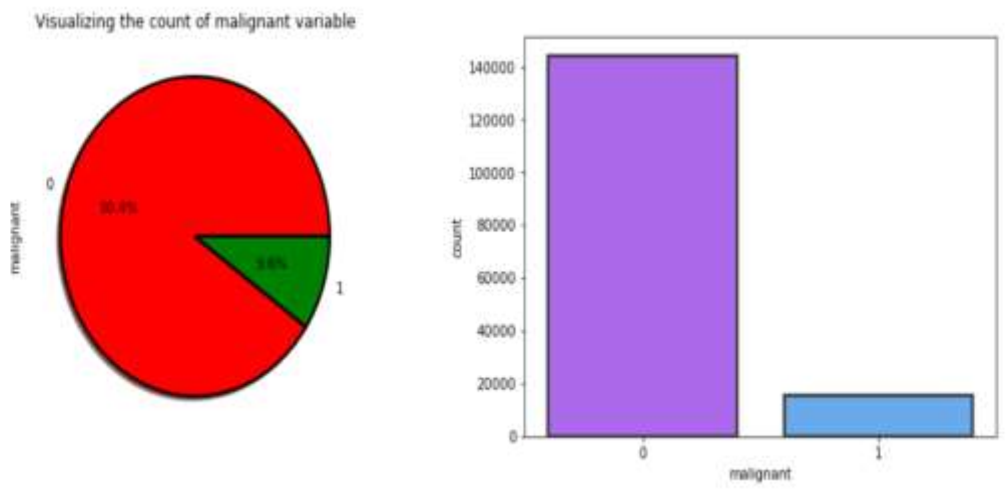
```
0    143346
1     16225
Name: label, dtype: int64
```



3. Visualizing count of malignant and non-malignant comments



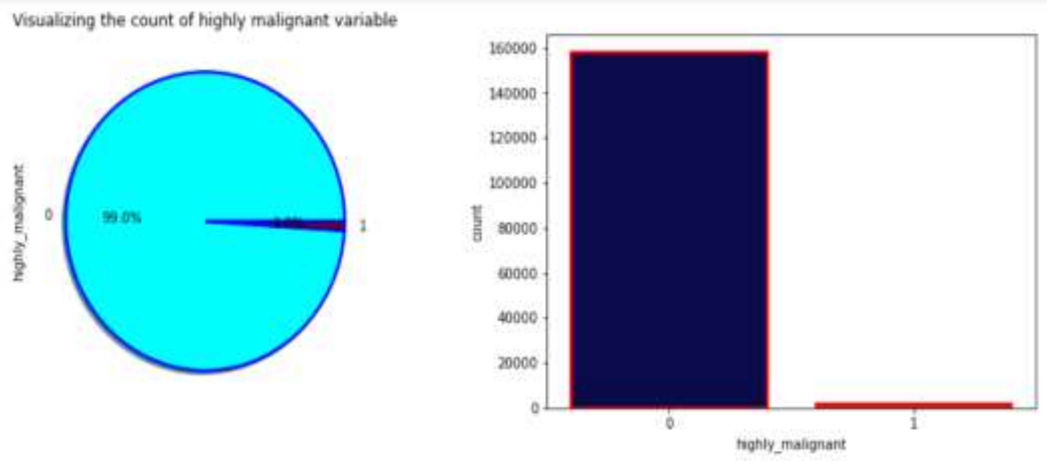
```
# Visualizing count of malignant and non malignant comments
print(df_train['malignant'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['0', '1']
colors = ["red", "green"]
df_train['malignant'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,colors=colors,
                                           wedgeprops = {'linewidth':2.3, 'edgecolor':'k'},
                                           title = 'Visualizing the count of malignant variable')
ax = sns.countplot('malignant', data=df_train, ax=ax[1],palette="cool_r",linewidth=2.3, edgecolor=".2")
plt.show()
```



4. Visualizing count of highly malignant and normal comments

```
# Visualizing count of highly malignant and normal comments
print(df_train['highly_malignant'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['0', '1']
colors = ["cyan", "maroon"]
df_train['highly_malignant'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,
                                                     wedgeprops = {'linewidth':2.3, 'edgecolor':'b'},
                                                     colors=colors,title = 'Visualizing the count of highly malignant variable')
ax = sns.countplot('highly_malignant', data=df_train, ax=ax[1],palette="ocean",linewidth=2.3, edgecolor="r")
plt.show()
```

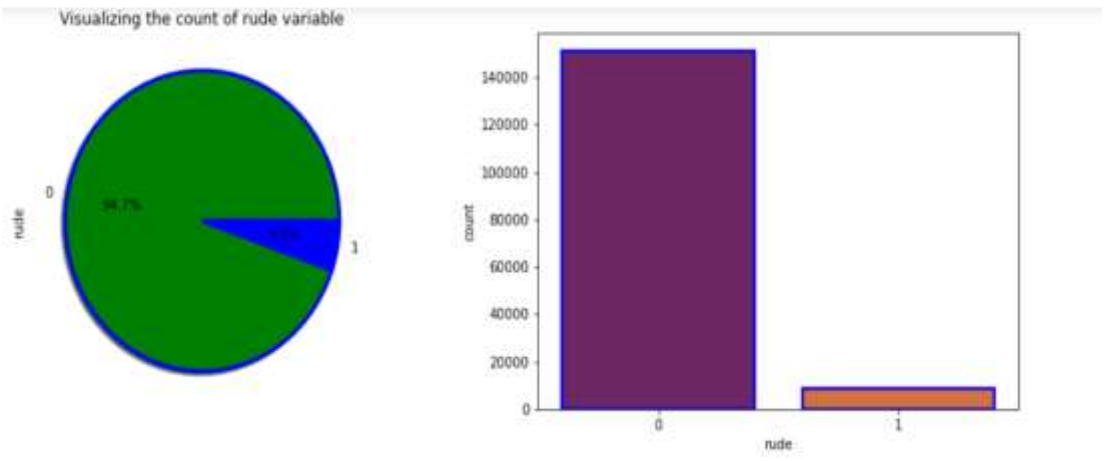
```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```



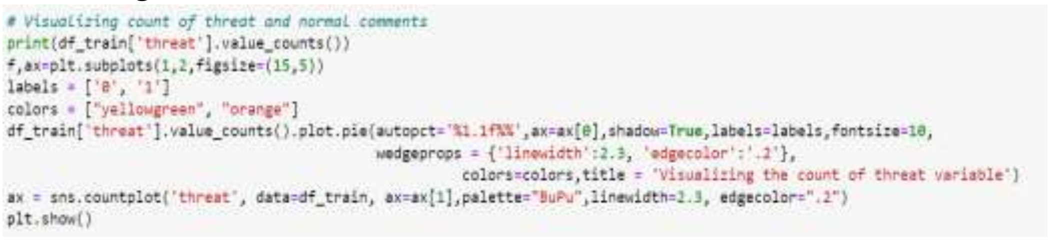
5. Visualizing count of rude and normal comments

```
# Visualizing count of rude and normal comments
print(df_train['rude'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['0', '1']
colors = ["green", "blue"]
df_train['rude'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,
                                       wedgeprops = {'linewidth':3, 'edgecolor':'b'},
                                                 colors=colors,title = 'Visualizing the count of rude variable')
ax = sns.countplot('rude', data=df_train, ax=ax[1],palette="inferno",linewidth=2.3, edgecolor="b")
plt.show()
```

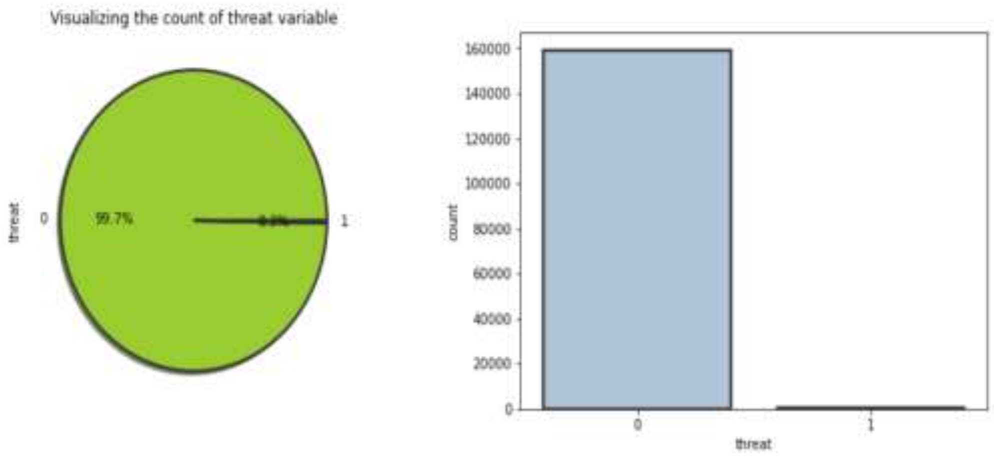
```
0    151122
1      8449
Name: rude, dtype: int64
```



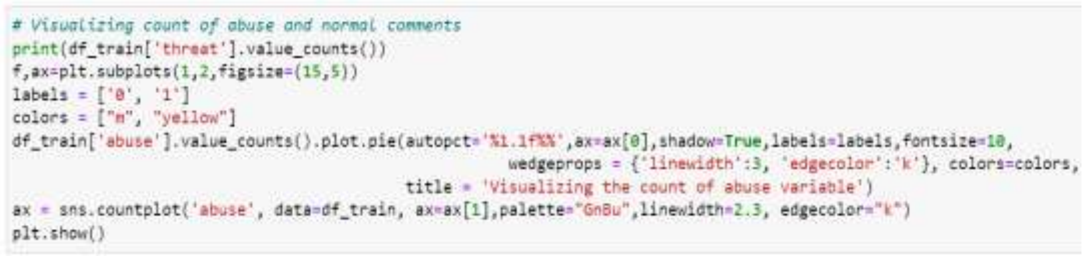
6. Visualizing count of threat and normal comments



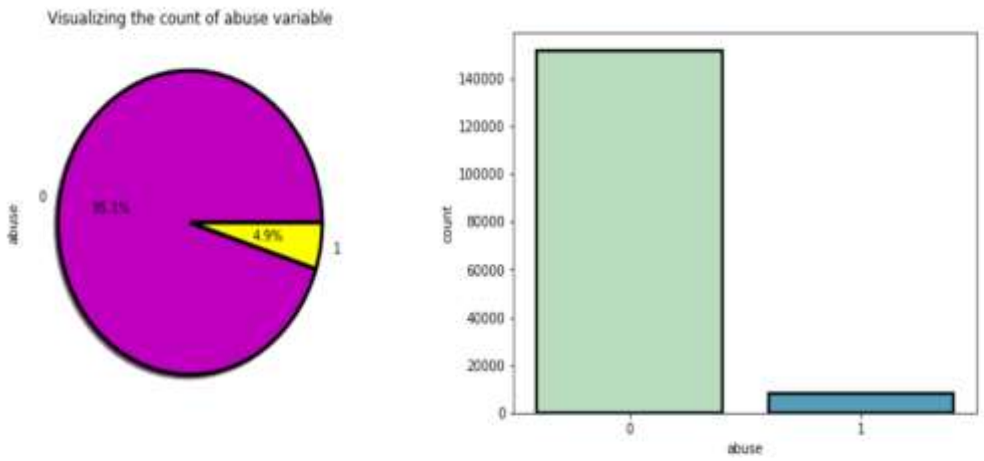
```
# Visualizing count of threat and normal comments
print(df_train['threat'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['0', '1']
colors = ["yellowgreen", "orange"]
df_train['threat'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,
                                           wedgeprops = {'linewidth':2.3, 'edgecolor':'.2'},
                                                   colors=colors,title = 'Visualizing the count of threat variable')
ax = sns.countplot('threat', data=df_train, ax=ax[1],palette="BuPu",linewidth=2.3, edgecolor=".2")
plt.show()
```



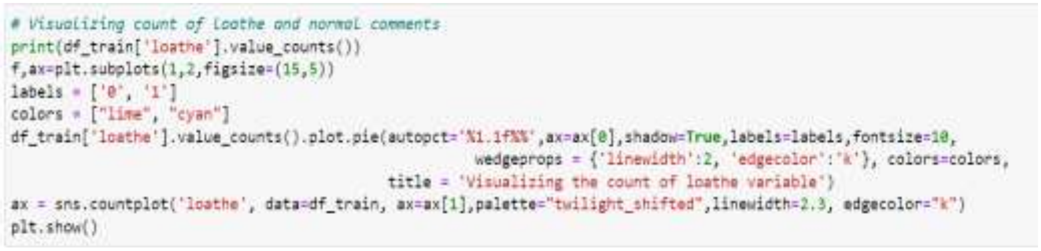
7. Visualizing count of abuse and normal comments



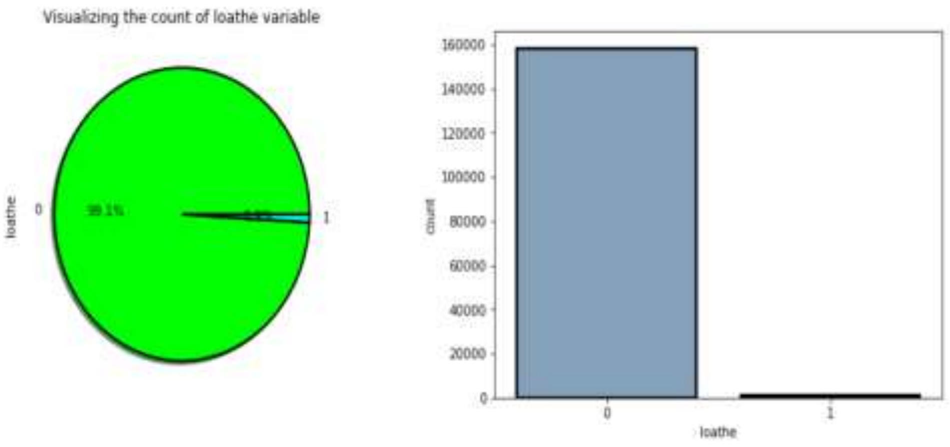
```
# Visualizing count of abuse and normal comments
print(df_train['threat'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['0', '1']
colors = ["m", "yellow"]
df_train['abuse'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,
                                                       wedgeprops = {'linewidth':3, 'edgecolor':'k'}, colors=colors,
                                             title = 'Visualizing the count of abuse variable')
ax = sns.countplot('abuse', data=df_train, ax=ax[1],palette="GnBu",linewidth=2.3, edgecolor="k")
plt.show()
```



8. Visualizing count of loathe and normal comments



```
# Visualizing count of loathe and normal comments
print(df_train['loathe'].value_counts())
f,ax=plt.subplots(1,2,figsize=(15,5))
labels = ['0', '1']
colors = ["lime", "cyan"]
df_train['loathe'].value_counts().plot.pie(autopct='%1.1f%%',ax=ax[0],shadow=True,labels=labels,fontsize=10,
                                            wedgeprops = {'linewidth':2, 'edgecolor':'k'}, colors=colors,
                                    title = 'Visualizing the count of loathe variable')
ax = sns.countplot('loathe', data=df_train, ax=ax[1],palette="twilight_shifted",linewidth=2.3, edgecolor="k")
plt.show()
```



• Interpretation of the Results

1. Starting with univariate analysis, with the help of count plot it was found that dataset is imbalanced with having higher number of records for normal comments than bad comments.
2. Moving further with word cloud it was found that malignant comments consists of words like fuck, nigger, moron, hate, suck etc. highly_malignant comments consists of words like ass, fuck, bitch, shit, die, suck, faggot etc. rude comments consists of words like nigger, ass, fuck, suck, bullshit, bitch etc. threat comments

consists of words like die, must die, kill, murder etc. abuse comments consists of words like moron, nigger, fat, jew, bitch etc. and loathe comments consists of words like nigga, stupid, nigger, die, gay, cunt etc.

CONCLUSION

- ## Key Findings and Conclusions of the Study

The finding of the study is that only few users over online use unparliamentary language. And most of these sentences have more stop words and are being quite long. As discussed before few motivated disrespectful crowds use these foul languages in the online forum to bully the people around and to stop them from doing these things that they are not supposed to do. Our study helps the online forums and social media to induce a ban to profanity or usage of profanity over these forums.

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

Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stopwords. We were also able to learn to convert strings into vectors through hash vectorizer. In this project we applied different evaluation metrics like log loss, hamming loss besides accuracy.