PROJECT REPORT ON

"MALIGNANT COMMENTS SCLASSIFIERS"

SUBMITTED BY

Ankit Dadarwala

ACKNOWLEDGMENT

I would like to express my special gratitude to "Flip Robo" team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzation skills. And I want to express my huge gratitude to **Mr. Shubham Yadav** (SME Flip Robo), he is the person who has helped me in tickets to get out of all the difficulties I faced while doing the project. And also inspired me in so many aspects and also encouraged me a lot with his valuable words and with his unconditional support I have ended up with a beautiful Project.

A huge thanks to my academic team "Data trained" who are the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank you for many other persons who has helped me directly or indirectly to complete the project.

INTRODUCTION

1.1 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.

1.2 Conceptual Background of the Domain Problem

Multi-label classification originated from the investigation of text categorisation problem, where each document may belong to several predefined topics simultaneously.

For example, *multi-class classification* makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time. Whereas, an instance of *multi-label classification* can be that a text might be about any of religion, politics, finance or education at the same time or none of these

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 used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying

1.3 Review of Literature

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.

1.4 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

Analytical Problem Framing

2.1 Mathematical/ Analytical modelling of the Problem

In this particular problem I have multilabel categorical variables as my target column and it was having of the different labels like malignant, rude, abuse, threat, loathe. So clearly it is a Multilabel Classification base problem and I have to use all classification algorithms while building the model. There were no null values in the dataset. To get better insight on the features I have used plotting like distribution plot, bar plot, Pie plot and Word cloud plot. Using the **TFIDF** vectorizer extract the 1,62,330 features from dataset. I have used all the linear regression and Tree based algorithms while building model then tunned the best model and saved the best model. At last, I have predicted the test data file using saved model.

2.2 Data Sources and their formats

The data was collected from my internship company – Flip Robo technologies in excel format. The sample data is provided to us from our client database. It is hereby given to us for this exercise. In order to built model for

online hate and abuse comment classifier which can used to classify hate and offensive comments.

Also, my dataset was having 1,59,571 rows and 8 columns including target. In this dataset have multilabel target columns have 6 target columns and comments text column that mean NLP sentiment analysis problem. We have to classify base on word's Verb, Noun, Adjective, Adverb etc..

Features Information:

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.

2.3 Data Pre-processing Done

1. As a first step I have imported required libraries and I have imported the dataset which was provided in excel format.
2. Then I did all the analysis like checking shape, value counts, info, duplicates value null values etc.
3. I have also dropped "id", column as I found they are useless.
4. While checking for null values I found null values in the dataset.
5. Make columns to check length of strings in particular comments.
6. Convert all text data into lower case.
7. Replace time date format to text format.
8. Replace all numerical no to text format.
9. Replace numeric IP address to text format.

10. Remove head and tail blank spaces.
11. Than make **CLAENTEXT** function to prepare text data, in that function remove string punctuation, tokenize comment in words, remove stop words like(I,me,u,your,have,had,etc…),than using POS TAG(part of speech) lemmatization the words.
12. Apply this function to comment_text column.
13. Again make column to check clean text length.
14. Using TFIDF Vectorize transform text to vectors and extract features for models

]:

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	text_length	clean_comment	clean_length
0	Explanation Why the edits made under my userna...	0	0	0	0	0	0	264	explanation edits make username hardcore metal...	165
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112	aww match background colour seemingly stuck th...	69
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233	hey man really try edit war guy constantly rem...	128
3	" More I can't make any real suggestions on im...	0	0	0	0	0	0	622	make real suggestion improvement wonder sectio...	352
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67	sir hero chance remember page	29
5	" Congratulations from me as well, use the to...	0	0	0	0	0	0	65	congratulation well use tool well talk	38
6	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0	44	cocksucker piss around work	27
7	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0	115	vandalism matt shirvington article revert plea...	52
8	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0	472	sorry word nonsense offensive anyway intend wr...	278
9	alignment on this subject and which are contra...	0	0	0	0	0	0	70	alignment subject contrary dulithgow	36

2.4 Data Inputs- Logic- Output Relationships

- ✓ Since I had one input column as text format and had clean the text from that extract features or vocabulary from that as input feature to train our models.
- ✓ Plotting some graph to see the malignant and non-malignant words and values counts of multilabel binary class
- ✓ As Output we get multilabel classification from models by analysing the words as comments are malignant, abuse, rude, threat, loathe or non-malignant

2.6 Hardware and Software Requirements and Tools Used

- ✓ Device name: HP Pavilion
- ✓ Processor: AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
- ✓ RAM: 8.00 GB
- ✓ System type: 64-bit operating system, x64-based processor
- ✓ Jupyter NoteBooks Version : 6.4.3
- ✓ Python3 vesrion : 3.8.8
- ✓

❖ **Libraries required** :-

- ✓ import pandas as pd
- ✓ import numpy as np
- ✓ import matplotlib.pyplot as plt
- ✓ import seaborn as sns
- ✓ import warnings
  warnings.filterwarnings('ignore')
- ✓ import re # regular expression
- ✓ import string
- ✓ import nltk
- ✓ from nltk.corpus import stopwords # for stop words
- ✓ from nltk.stem import WordNetLemmatizer # for lemmatization
- ✓ from nltk.tokenize import word_tokenize # for word tokenize
- ✓ from nltk import pos_tag # for part of speech
- ✓ from wordcloud import WordCloud
- ✓ from sklearn.feature_extraction.text import TfidfVectorizer
- ✓ from sklearn.naive_bayes import MultinomialNB
- ✓ from sklearn.model_selection import train_test_split,cross_val_score
- ✓ from sklearn.multiclass import OneVsRestClassifier
- ✓ from sklearn.metrics import
  accuracy_score,confusion_matrix,classification_report,roc_curve,roc_auc_score,f1_score,auc,hamming_loss
- ✓ from sklearn.linear_model import LogisticRegression
- ✓ from sklearn.tree import DecisionTreeClassifier
- ✓ from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, Bagging Classifier

- ✓ from xgboost import XGBClassifier
- ✓ from yellowbrick.text import FreqDistVisualizer
- ✓ import pickle

Data Analysis and Visualization

3.1 Testing of Identified Approaches (Algorithms)

- **OneVsRestClassifier**

  - Traditional two-class and multi-class problems can both be cast into multi-label ones by restricting each instance to have only one label. On the other hand, the generality of multi-label problems inevitably makes it more difficult to learn. An intuitive approach to solving multi-label problem is to decompose it into multiple independent binary classification problems (one per category).

  - In an "one-to-rest" strategy, one could build multiple independent classifiers and, for an unseen instance, choose the class for which the confidence is maximized.

  - The main assumption here is that the labels are *mutually exclusive*. You do not consider any underlying correlation between the classes in this method.

  - For instance, it is more like asking simple questions, say, "*is the comment toxic or not*", "*is the comment threatening or not?*", etc. Also there might be an extensive case of overfitting here, since most of the comments are unlabeled, i,e., most of the comments are clean comments.

Since multi-labels was my target and it was a classifiaction , so this particular problem was Classfication problem. And I have used all classifier algorithms to build my model. By looking into the difference of Accuracy score and F1-score micro or weighted and also hamming loss. To get the best model we have to run through multiple models and to avoid the confusion of overfitting we select best random state. Below are the list of Classifier algorithms I have used in my project.

1. Logistic Regression
2. MultinomialNB
3. XGB Classifier
4. Decision Tree Classifer
5. Random Forest Classifier
6. Bagging Classifier
7. AdaBoost Classifier
8. OneVsRestClassifier

3.2 Key Metrics for success in solving problem under consideration

1. **Micro-averaging (Label based measures):**To measure a multi-class classifier we have to average out the classes somehow. There are two different methods of doing this called micro-averaging and macro-averaging.in micro-averaging all TPs, TNs, FPs and FNs for each class are summed up and then the average is taken.

   . Microaveraging Precision $Prc^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FPs(c_i)}$

   . Microaveraging Recall $Rcl^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FNs(c_i)}$

2. **Macro-averaging:** Macro-averaging method can be used when you want to know how the system performs overall across the sets of data. You should not come up with any specific decision with this average. On the other hand, micro-averaging can be a useful measure when your dataset varies in size.

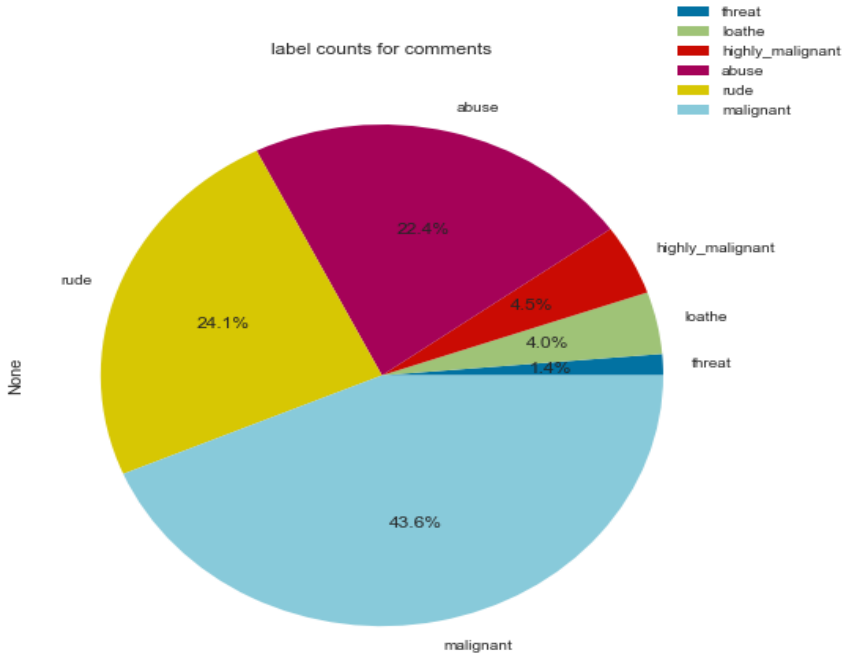
- Macroaveraging Precision $Prc^{macro}(D) = \frac{\sum_{c_i \in \mathcal{C}} Prc(D, c_i)}{|\mathcal{C}|}$

- Macroaveraging Recall $Rec^{macro}(D) = \frac{\sum_{c_i \in \mathcal{C}} Rcl(D, c_i)}{|\mathcal{C}|}$

3. **Hamming loss (Example base measure):**In simplest of terms, Hamming-Loss is the fraction of labels that are incorrectly predicted, i.e., the fraction of the wrong labels to the total number of labels.

$\frac{1}{|N| \cdot |L|} \sum_{i=1}^{|N|} \sum_{j=1}^{|L|} \text{xor}(y_{i,j}, z_{i,j})$, where $y_{i,j}$ is the target and $z_{i,j}$ is the prediction.

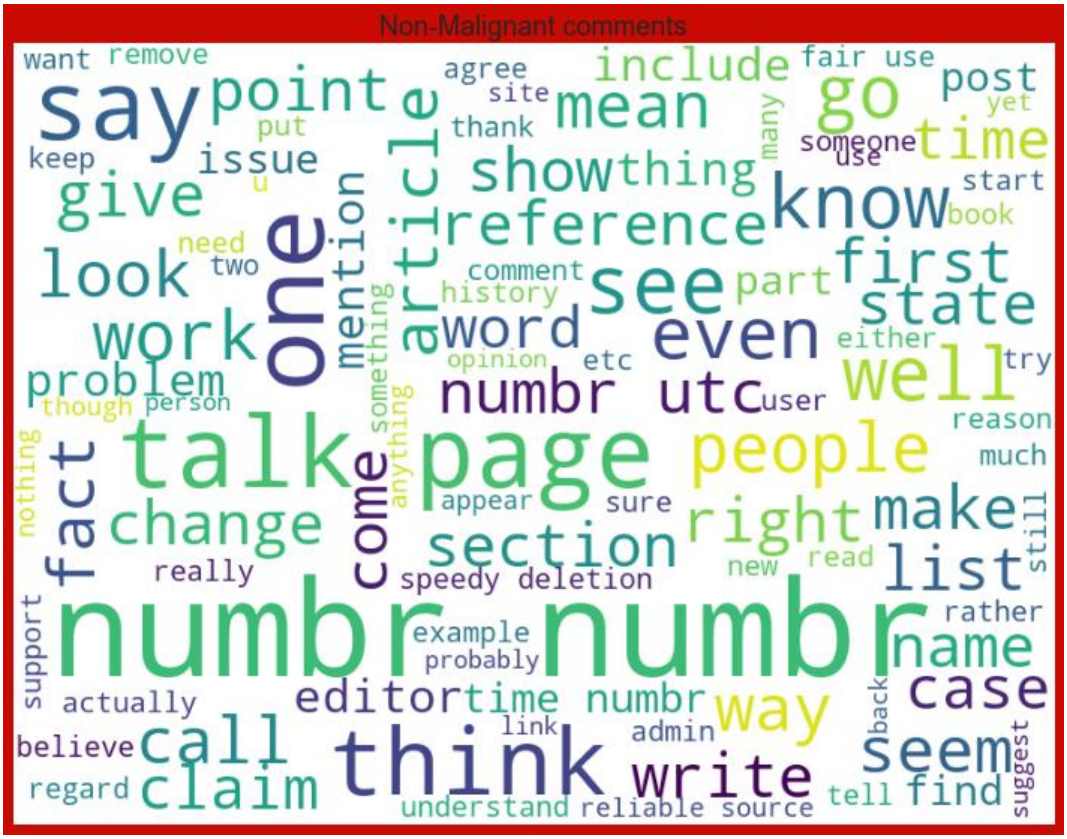
3.3 Visualizations



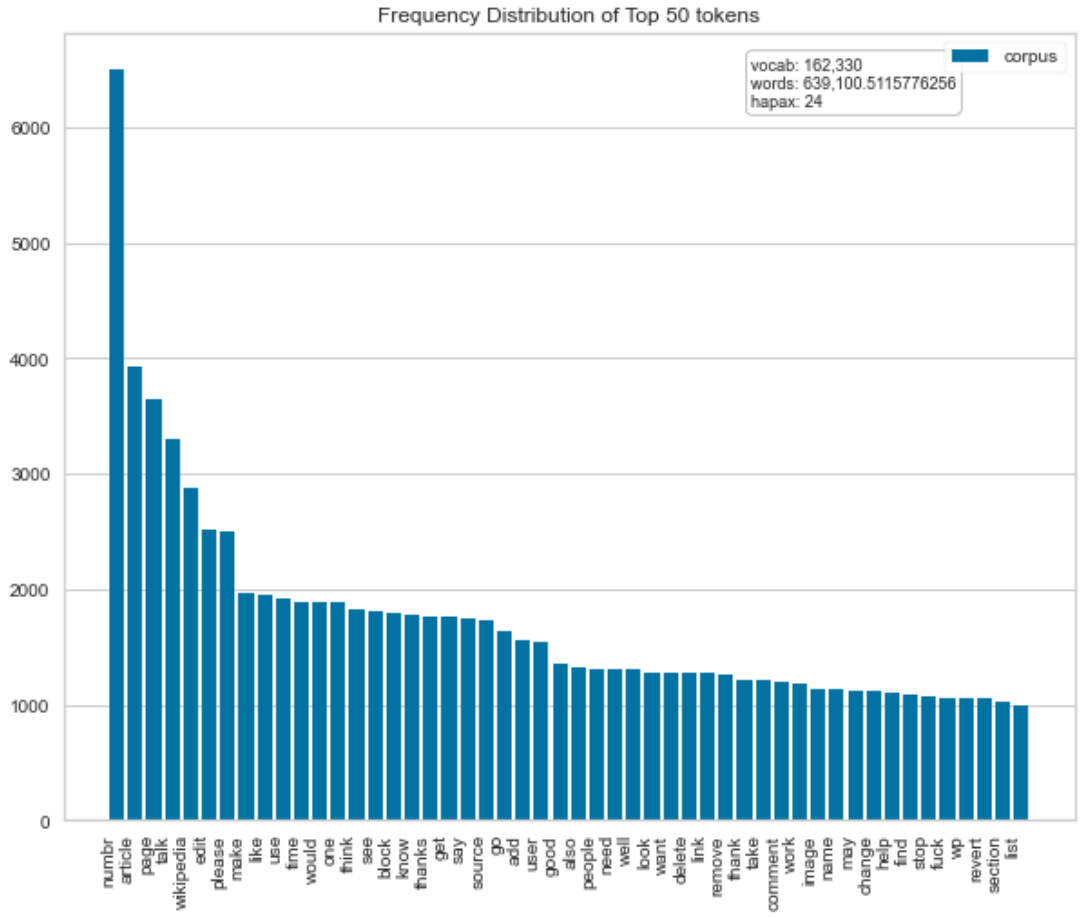
As show in fig pie chart of label counts for comments most of comments rate as malignant 44%, Abuse – 22.4%, Rude – 24.1% other remaining are in 11% distributed.

Malignant, Abuse, Rude, Threat comments words cloud

Non-Malignant comments words cloud



Frequency distribution of top 50 token words



3.4 Run and evaluate select models

1). MultinomialNB

```
# Naive bayes Multinomial

naive=MultinomialNB()
clf=OneVsRestClassifier(naive)
clf.fit(train_x,train_y)
y_pred_nv=clf.predict(test_x)
print('train score =>\t',clf.score(train_x,train_y)*100)
print('accuracy score =>\t',accuracy_score(test_y,y_pred_nv)*100)
print('f1 score =>\t',f1_score(test_y,y_pred_nv,average='weighted')*100)
print('Classification report =>\n',classification_report(test_y,y_pred_nv,target_names=y.columns))
```

```
train score =>   90.07242678985487
accuracy score =>        89.74557152406418
f1 score =>      18.271272072127605
Classification report =>
                   precision    recall  f1-score   support

       malignant       0.99      0.17      0.28      4695
highly_malignant       0.00      0.00      0.00       491
           rude        0.98      0.10      0.18      2544
          threat       0.00      0.00      0.00       154
           abuse       0.92      0.03      0.07      2387
          loathe       0.00      0.00      0.00       442

       micro avg       0.98      0.10      0.19     10713
       macro avg       0.48      0.05      0.09     10713
    weighted avg       0.87      0.10      0.18     10713
     samples avg       0.02      0.01      0.01     10713
```

2). Logistic Regression

```
# Logistic regression
lg=LogisticRegression()
clf1=OneVsRestClassifier(lg)
clf1.fit(train_x,train_y)
y_pred_lg=clf1.predict(test_x)
print('train score =>\t',clf1.score(train_x,train_y)*100)
print('accuracy score =>\t',accuracy_score(test_y,y_pred_lg)*100)
print('f1 score =>\t',f1_score(test_y,y_pred_lg,average='weighted')*100)
print('Classification report =>\n',classification_report(test_y,y_pred_lg,target_names=y.columns))
```

```
train score =>   92.49411364470586
accuracy score =>        91.68407419786097
f1 score =>      65.06558608113785
Classification report =>
                precision    recall  f1-score   support

       malignant       0.93      0.58      0.72      4695
highly_malignant       0.55      0.24      0.34       491
           rude        0.90      0.61      0.73      2544
          threat       0.73      0.07      0.13       154
           abuse       0.82      0.49      0.61      2387
          loathe       0.70      0.15      0.24       442

       micro avg       0.88      0.53      0.66     10713
       macro avg       0.77      0.36      0.46     10713
    weighted avg       0.87      0.53      0.65     10713
     samples avg       0.05      0.05      0.05     10713
```

---

3). Decision Tree classifier

```
# Decision tree classifier
clf2=OneVsRestClassifier(dtc)
clf2.fit(train_x,train_y)
y_pred=clf2.predict(test_x)
print('train score =>\t',clf2.score(train_x,train_y)*100)
print('accuracy score =>\t',accuracy_score(test_y,y_pred)*100)
print('f1 score =>\t',f1_score(test_y,y_pred,average='weighted')*100)
print('Classification report =>\n',classification_report(test_y,y_pred,target_names=y.columns))
```

```
train score =>   99.91136894690194
accuracy score =>        89.41343582887701
f1 score =>      65.33183244580941
Classification report =>
                precision    recall  f1-score   support

       malignant       0.73      0.67      0.70      4695
highly_malignant       0.36      0.27      0.31       491
           rude        0.76      0.74      0.75      2544
          threat       0.33      0.21      0.26       154
           abuse       0.61      0.59      0.60      2387
          loathe       0.47      0.34      0.40       442

       micro avg       0.68      0.63      0.66     10713
       macro avg       0.54      0.47      0.50     10713
    weighted avg       0.68      0.63      0.65     10713
     samples avg       0.06      0.06      0.06     10713
```

---

4). Random Forest Classifier

```
# Random forest classifier
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(random_state=1)
clf3=OneVsRestClassifier(rfc)
clf3.fit(train_x,train_y)
y_pred1=clf3.predict(test_x)
print('train score =>\t',clf3.score(train_x,train_y)*100)
print('accuracy score =>\t',accuracy_score(test_y,y_pred1)*100)
print('f1 score =>\t',f1_score(test_y,y_pred1,average='weighted')*100)
print('Classification report =>\n',classification_report(test_y,y_pred1,target_names=y.columns))
```

```
train score =>   99.90778789425153
accuracy score =>        91.34149398395722
f1 score =>      60.79232430142319
Classification report =>
                precision    recall  f1-score   support

       malignant       0.94      0.53      0.68      4695
highly_malignant       0.43      0.05      0.08       491
           rude        0.91      0.60      0.73      2544
          threat       0.60      0.04      0.07       154
           abuse       0.82      0.44      0.57      2387
          loathe       0.62      0.07      0.12       442

       micro avg       0.90      0.48      0.63     10713
       macro avg       0.72      0.29      0.38     10713
    weighted avg       0.87      0.48      0.61     10713
     samples avg       0.05      0.04      0.04     10713
```

---

5). XGBOOST Classifier

```
# XGB classifier
xg=XGBClassifier(disable_default_eval_metric=1)
clf4=OneVsRestClassifier(xg)
clf4.fit(train_x,train_y)
y_pred2=clf4.predict(test_x)
print('train score =>\t',clf4.score(train_x,train_y)*100)
print('accuracy score =>\t',accuracy_score(test_y,y_pred2)*100)
print('f1 score =>\t',f1_score(test_y,y_pred2,average='weighted')*100)
print('Classification report =>\n',classification_report(test_y,y_pred2,target_names=y.columns))
```

```
train score =>   94.06261470559271
accuracy score =>        91.61096256684492
f1 score =>      68.26609530556574
Classification report =>
                  precision    recall  f1-score   support

       malignant       0.91      0.59      0.72      4695
highly_malignant       0.45      0.21      0.29       491
            rude       0.86      0.73      0.79      2544
          threat       0.59      0.22      0.32       154
           abuse       0.78      0.57      0.66      2387
          loathe       0.70      0.30      0.42       442

       micro avg       0.84      0.58      0.69     10713
       macro avg       0.71      0.44      0.53     10713
    weighted avg       0.83      0.58      0.68     10713
     samples avg       0.05      0.05      0.05     10713
```

6). Ada Boost Classifier

```
# Adaboost classifier
ad=AdaBoostClassifier(random_state=1)
clf6=OneVsRestClassifier(ad)
clf6.fit(train_x,train_y)
y_pred4=clf6.predict(test_x)
print('train score =>\t',clf6.score(train_x,train_y)*100)
print('accuracy score =>\t',accuracy_score(test_y,y_pred4)*100)
print('f1 score =>\t',f1_score(test_y,y_pred4,average='weighted')*100)
print('Classification report =>\n',classification_report(test_y,y_pred4,target_names=y.columns))
```

```
train score =>   91.16375258507237
accuracy score =>        90.74197860962568
f1 score =>      61.74004832520418
Classification report =>
                  precision    recall  f1-score   support

       malignant       0.87      0.55      0.67      4695
highly_malignant       0.48      0.29      0.36       491
            rude       0.87      0.61      0.72      2544
          threat       0.38      0.15      0.21       154
           abuse       0.78      0.39      0.52      2387
          loathe       0.54      0.32      0.40       442

       micro avg       0.82      0.50      0.62     10713
       macro avg       0.65      0.39      0.48     10713
    weighted avg       0.81      0.50      0.62     10713
     samples avg       0.05      0.04      0.04     10713
```

7). Bagging Classifier

```
# Bagging classifier
bg=BaggingClassifier()
clf7=OneVsRestClassifier(bg)
clf7.fit(train_x,train_y)
y_pred5=clf7.predict(test_x)
print('train score =>\t',clf7.score(train_x,train_y)*100)
print('accuracy score =>\t',accuracy_score(test_y,y_pred5)*100)
print('f1 score =>\t',f1_score(test_y,y_pred5,average='weighted')*100)
print('Classification report =>\n',classification_report(test_y,y_pred5,target_names=y.columns))
```

```
train score =>   98.44940420236529
accuracy score =>        90.83180147058823
f1 score =>      67.8213194038188
Classification report =>
                  precision    recall  f1-score   support

       malignant       0.82      0.67      0.74      4695
highly_malignant       0.40      0.15      0.22       491
            rude       0.81      0.76      0.78      2544
          threat       0.51      0.13      0.21       154
           abuse       0.67      0.59      0.63      2387
          loathe       0.61      0.31      0.41       442

       micro avg       0.76      0.63      0.69     10713
       macro avg       0.63      0.44      0.50     10713
    weighted avg       0.75      0.63      0.68     10713
     samples avg       0.06      0.06      0.06     10713
```

8). Hamming loss:

```
]: # hamming loss of all models
print('hamming loss naive ==>',hamming_loss(test_y,y_pred_nv))
print('hamming loss logistic ==>',hamming_loss(test_y,y_pred_lg))
print('hamming loss dtc ==>',hamming_loss(test_y,y_pred))
print('hamming loss rf ==>',hamming_loss(test_y,y_pred1))
print('hamming loss xg ==>',hamming_loss(test_y,y_pred2))
print('hamming loss ad ==>',hamming_loss(test_y,y_pred4))
print('hamming loss bg ==>',hamming_loss(test_y,y_pred5))
```

```
hamming loss naive ==> 0.03348860851158645
hamming loss logistic ==> 0.02025192179144385
hamming loss dtc ==> 0.02466647170231729
hamming loss rf ==> 0.021418226381461677
hamming loss xg ==> 0.019649621212121212
hamming loss ad ==> 0.022782976827094473
hamming loss bg ==> 0.021174520944741534
```

9). Jaccard score:

Sr.NO	Model name	Jaccard score
1	MultinomialNB	6.975
2	Logistic Regression	39.70
3	Decision tree classifier	38.62
4	Random forest classifier	36.02
5	XGB Classifier	41.69
6	Ada boost classifier	35.19
7	Bagging Classifier	42.62

10). Saving Model for Prediction

=> I have saved my best model using .pkl as follows**. We save Bagging Classifier with OneVsRestClassifier**

```
# Saving model
import pickle
pickle.dump(clf7,open('comment_analysis.pkl','wb'))
bagging_model=pickle.load(open('comment_analysis.pkl','rb'))
```

```
# Use test data for testing
test1=pd.read_csv('comment test.csv',encoding='utf-8')
```

11). Prediction of Test data file by saving model

```
# Predict test data
predition=bagging_model.predict(test_tf)
predition
```

```
array([[1, 0, 1, 0, 1, 0],
       [1, 0, 1, 0, 1, 0],
       [0, 0, 0, 0, 0, 0],
       ...,
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0]])
```

```
# Making predicted test data frame
pred_data=pd.DataFrame(predition,index=test1['id'],columns=['malignant','highly_malignant','rude','threat','abuse','loathe'])
pred_data
```

id	malignant	highly_malignant	rude	threat	abuse	loathe
00001cee341fdb12	1	0	1	0	1	0
0000247867823ef7	1	0	1	0	1	0
00013b17ad220c46	0	0	0	0	0	0
00017563c3f7919a	0	0	0	0	0	0
00017695ad8997eb	0	0	0	0	0	0
...	...	...	...	...	...	...
fffcd0960ee309b5	1	0	0	0	0	0
fffd7a9a6eb32c16	0	0	0	0	0	0
fffda9e8d6fafa9e	0	0	0	0	0	0
fffe8f1340a79fc2	0	0	0	0	0	0

3.5 Interpretation of the Results

- ✓ Firstly, the datasets were not having duplicates values and Null values.
- ✓ And proper plotting for proper type of features will help us to get better insight on the data.
- ✓ It is multi-label classification problem
- ✓ And we have to use multiple metrics like Accuracy score, F1-micro score, Hamming loss, Jaccard score which will help us to decide the best model.
- ✓ I found bagging classifier as the best model with **90.83% Accuracy score, hamming loss of 0.021, Jaccard score of 42.62%**
- ✓ At last I have predicted **Test data set** using saved model. It was good!! that I was able to get proper binary classification

CONCLUSION

4.1 Key Findings and Conclusions of the Study

In this project report, we have used machine learning algorithms to predict the multi-label class. We have mentioned the step-by-step procedure to analyse the dataset and feature extraction. These feature set were then given as an input to algorithms. Hence we calculated the performance of each model using different performance metrics and compared them based on these metrics. Then we have also saved the best model and predicted the label.

4.2 Limitations of this work and Scope for Future Work

- ✓ The same problem can be solved using deep learning.
- ✓ For more speed we could use decision trees and for a reasonable trade-off between speed and accuracy we could also opt for ensemble models.
- ✓ In test data set some comments are in random symbols so we have encoding them into UTF-8 but not sure about their result predict by model.