FLIP ROBO

COMMENT TYPE PREDICTION MODEL

Machine Learning Project

Submitted by: Bhushan Kumar Sharma

Inter Data Scientist

Batch : Internship 21

2022

ACKNOWLEDGMENT

*****

I would like to express my special gratitude to "Flip Robo" team, who has given me this opportunity to deal with this dataset and it has helped me to improve my model building skills.

A huge thanks to my academic team "Datatrained" who is 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 to many other persons who has helped me directly or indirectly to complete the project.

Following are the external references which I used:

www.geeksforgeeks.org

www.stackoverflow.com

www.w3school.com

www.google.com

Datatrained Lectures

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.

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 inoffensive, 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**

There has been a remarkable increase in the cases of cyber bullying 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**

  - 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 inoffensive, 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 cyber bullying

- **Mathematical/ Analytical Modelling of the Problem**

In this particular problem I have multilevel 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 Multilevel 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 tuned the best model and saved the best model. At last, I have predicted the test data file using saved model.

In order to determine the regularization parameter, throughout the project in classification part, we would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFID in order to convert the tokens from the train documents into vectors so that machine can do further processing. I have used all the classification algorithms while building model then tuned the best model and saved the best model.

- **Dataset Description**

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

The data set contains the training set, which has approximately 1,59,571 samples and the test set which contains nearly 1,53,164 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.

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving malignant_train.csv to malignant_train.csv
```

```
import io
df = pd.read_csv(io.BytesIO(uploaded['malignant_train.csv']))
df.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

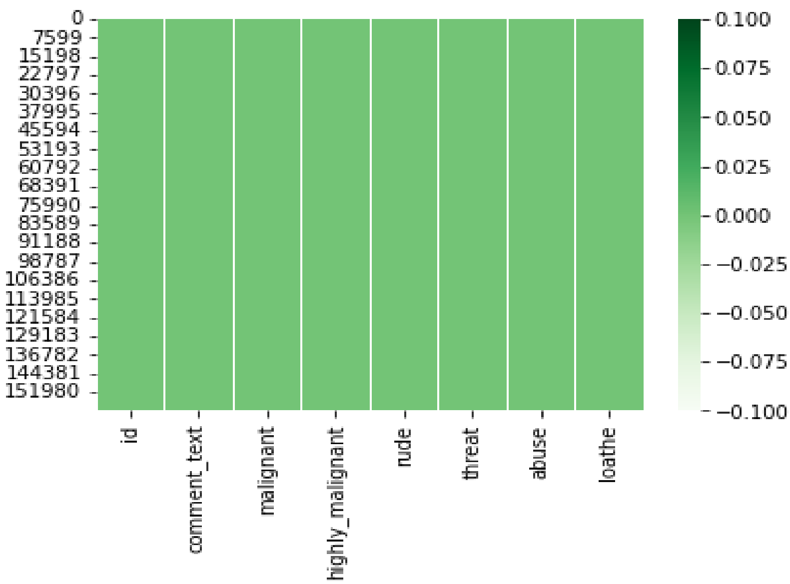
Data Information

```
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
```

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 and null values etc.
3. Make columns to check length of strings in particular comments
4. Created new columns for length to get amount of cleaned data records
5. Converted all text into lower case
6. Replace email address with email
7. Replace URL with Web address
8. Replace money symbols with “moneysymb” £:
9. Replace 10 digit phone numbers
10. Replace numbers with “number”
11. Remove punctuation
12. Replace whitespace between terms with a single space
13. Remove leading and trailing whitespace

- **Data Pre-processing:**
  - **Null values identification:**



Note: **As heatmap is clear, which indicating, no null values are present in the dataset.**

- *Created New columns for length to get amount of cleaned data records*

```
# New column for length of message
df['Length of comment_text'] = df['comment_text'].str.len()
df_test['Length of comment_text'] = df_test['comment_text'].str.len()
```

- *Converted all text into lower case:*

```
# Converting all msges into lower case
df['comment_text'] = df['comment_text'].apply(lambda x:x.lower())
df_test['comment_text'] = df_test['comment_text'].apply(lambda x:x.lower())
```

- *Various operations perform to clean the comment content.*

```
# Replace email address with email:[
df['comment_text'] = df['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'email_address')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'email_address')
```

```
# Replace URL with 'webaddress'
df['comment_text'] = df['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$', 'webaddress')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$', 'webaddress')
```

```
# Replace money symbols with 'moneysymb' (£ can be typed with ALT key + 156)
df['comment_text'] = df['comment_text'].str.replace(r'£|\$', 'dollers')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'£|\$', 'dollers')
```

```
# Replace 10 digit phone numbers (formats include paranthesis spaces, no spaces, dashed) with 'phonenumber'
df['comment_text'] = df['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$', 'phone_number')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$', 'phone_number')
```

```
# Replace numbers with 'number'
df['comment_text'] = df['comment_text'].str.replace(r'\d+(\.\d+)?', 'number')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'\d+(\.\d+)?', 'number')
```

```
# Remove punctuation
df['comment_text'] = df['comment_text'].str.replace(r'[^\w\d\s]', ' ')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'[^\w\d\s]', ' ')
```

```
# Replace whitespace between terms with a single space:
df['comment_text'] = df['comment_text'].str.replace(r'\s+', ' ')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'\s+', ' ')
```

```
# Remove leading and trailing whitespace
df['comment_text'] = df['comment_text'].str.replace(r'^\s+|\s+?$', ' ')
df_test['comment_text'] = df_test['comment_text'].str.replace(r'^\s+|\s+?$', ' ')
```

- *Now, Applied operation to remove stop words*

```
# Remove stopwords
import nltk
nltk.download('stopwords')

import string
from nltk.corpus import stopwords
stop_words = stopwords.words('english') + ['u', 'ur', '4', '2','im','dont', 'doin', 'ure']
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'you
```

```
df['comment_text'] = df['comment_text'].apply(lambda x : ' '.join(word for word in x.split() if word not in stop_words ))
df_test['comment_text'] = df_test['comment_text'].apply(lambda x : ' '.join(word for word in x.split() if word not in stop_words ))
```

```
# New column length after removing unwanted crupt data
df['Len of clean comment'] = df.comment_text.str.len()
df_test['Len of clean comment'] = df_test.comment_text.str.len()
```

By finding this Len of clean "Comment" we can see the content weightage, how much data cleaned by processing various operation to convert into this desired form.

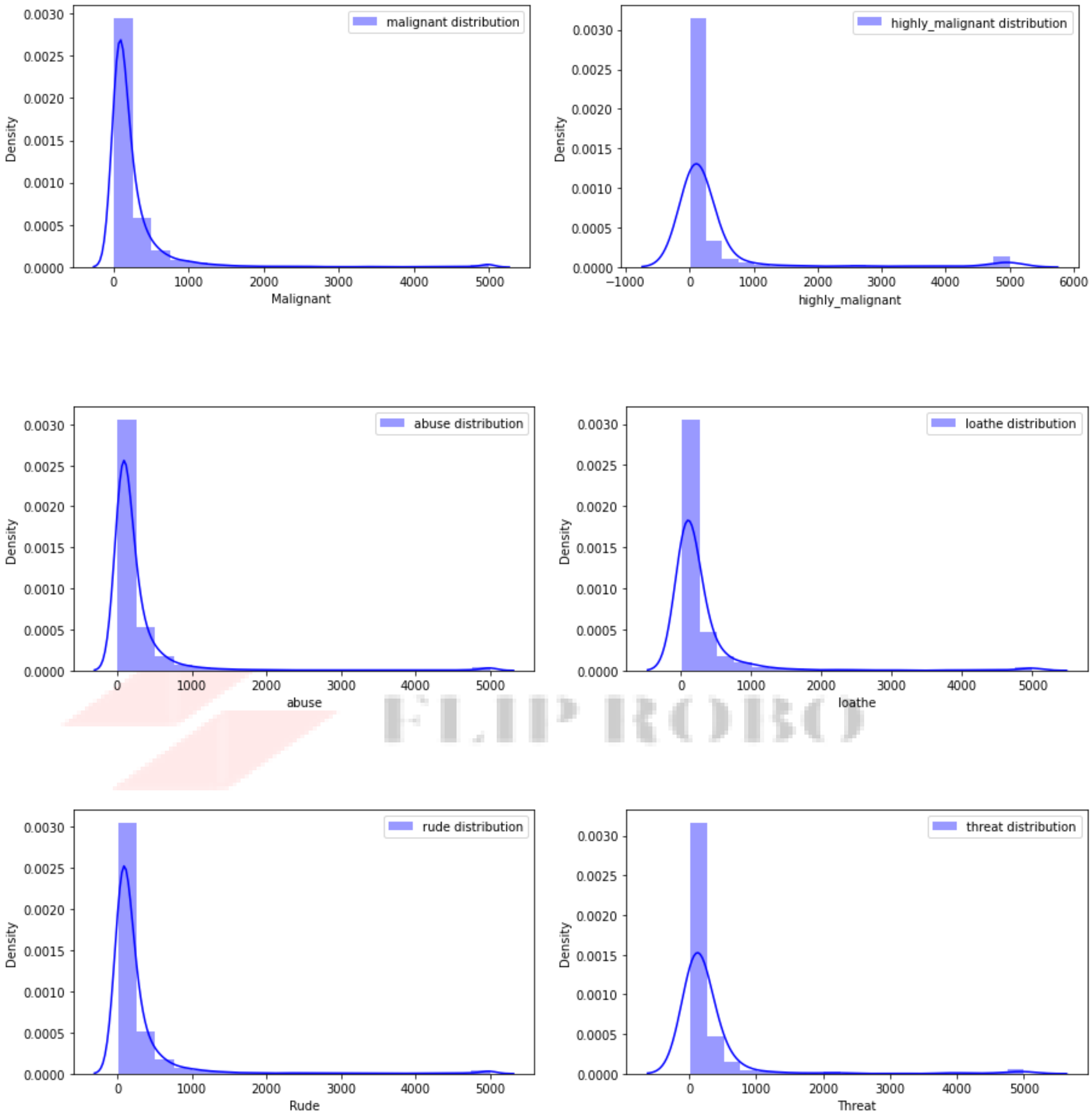
```
# total reduced length
print("Original Length: ",df['Length of comment_text'].sum())
print("Cleaned Length: ",df['Len of clean comment'].sum())
```

```
Original Length:  62893130
Cleaned Length:  40723981
```

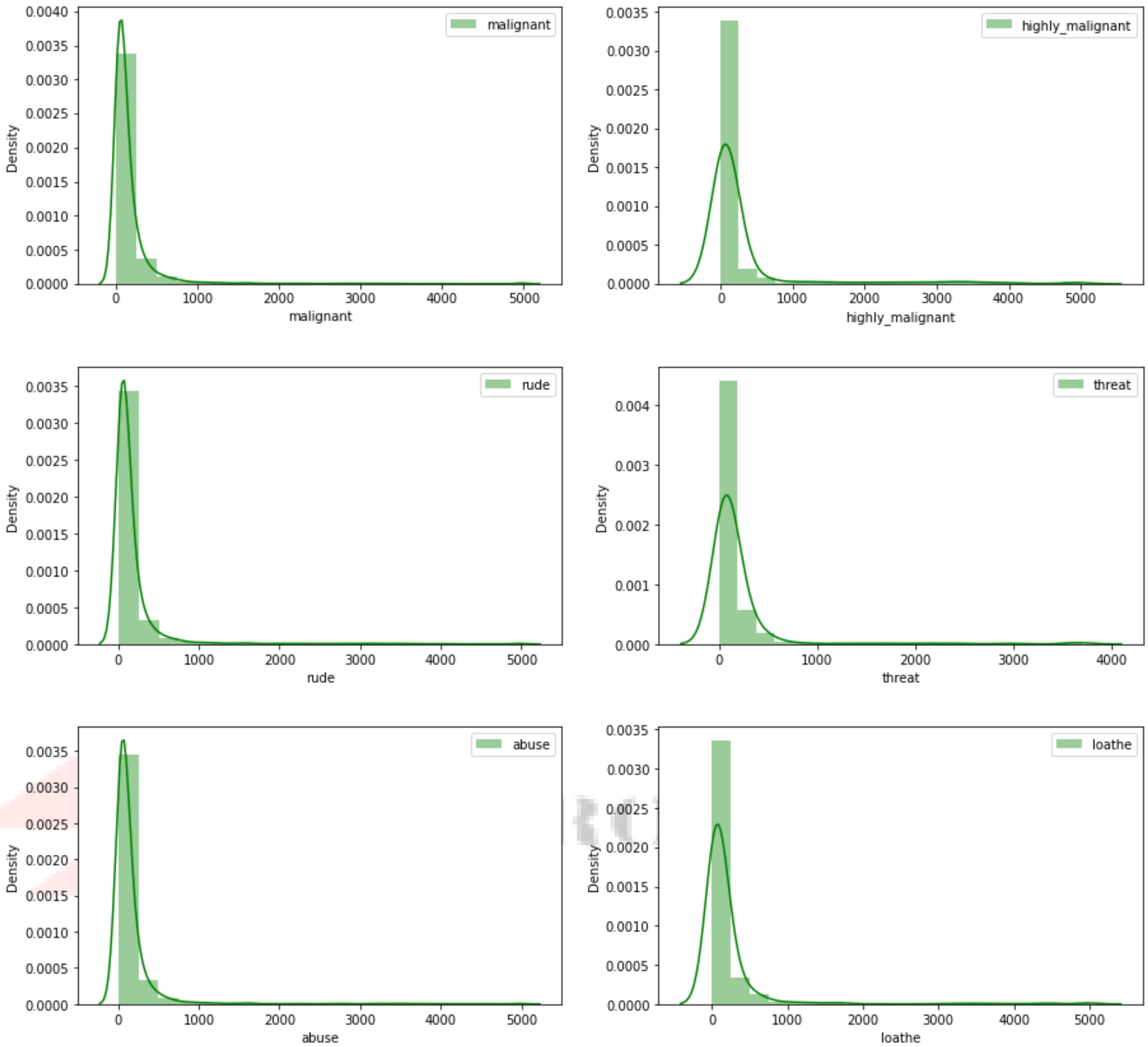
```
# total reduced length
print("Original Length: ",df_test['Length of comment_text'].sum())
print("Cleaned Length: ",df_test['Len of clean comment'].sum())
```

```
Original Length:  55885733
Cleaned Length:  36136856
```

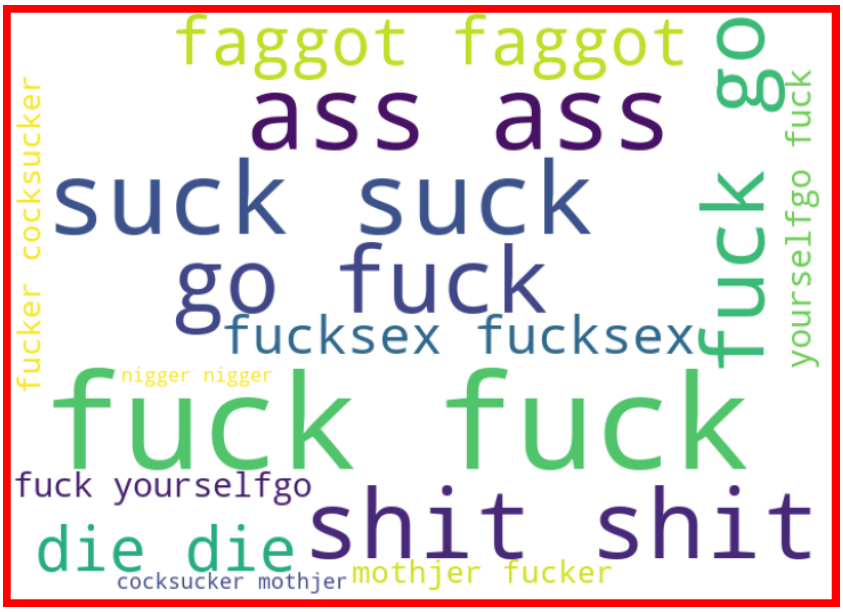
- ***Review text Distribution before cleaning***



- ***Review Distribution after cleaning***



- ***Visualization*:**
  - **Word Cloud for Highly Malignant:**

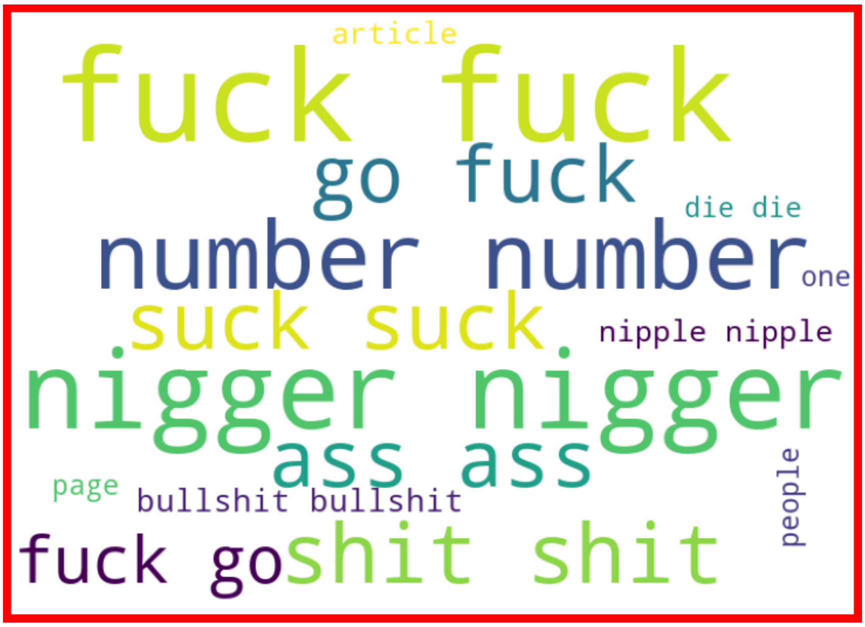


  - **Word Cloud for Abuse:**

- **Word Cloud for threat :**

- **Word Cloud for Loathe:**

- **Word Cloud for Rude:**



➢ **Apply TfidVectorizer to the Combined Review column**

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from scipy.sparse import csr_matrix
```

```
tf_vec = TfidfVectorizer()
df.columns
```

```
Index(['id', 'comment_text', 'malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe', 'Length of comment_text', 'Len of clean comment'],
      dtype='object')
```

```
tf_vec = TfidfVectorizer()
tf_vec.fit(df['comment_text'])
```

```
TfidfVectorizer()
```

```
features = tf_vec.transform(df['comment_text'])
```

```
x = features
```

```
x.shape
```

```
(159571, 180221)
```

```
y = df[['malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe']]
```

ectorization for Testing data

```
df_test.columns
```

```
Index(['id', 'comment_text', 'Length of comment_text', 'Len of clean comment'], dtype='object')
```

```
features2 = tf_vec.transform(df_test['comment_text'])
```

```
test_x = features2
```

```
test_x.shape
```

```
(153164, 180221)
```

```
print('x shape: ', x.shape)
print('y shape: ', y.shape)
```

```
x shape:  (159571, 180221)
y shape:  (159571, 6)
```

➢ Machine Learning:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from scipy.sparse import csr_matrix
from sklearn.metrics import multilabel_confusion_matrix, f1_score
from sklearn.multiclass import OneVsRestClassifier
naive = MultinomialNB()
```

 # MultinomialNB Model Performance:

```
Accuracy Score 0.8977690508021391
Classification Report:
              precision    recall  f1-score   support

           0       0.16      0.98      0.28       759
           1       0.00      0.00      0.00         5
           2       0.10      0.96      0.18       262
           3       0.01      0.33      0.01         3
           4       0.03      0.87      0.06        90
           5       0.00      0.00      0.00         5

   micro avg       0.10      0.96      0.18      1124
   macro avg       0.05      0.52      0.09      1124
weighted avg       0.13      0.96      0.23      1124
 samples avg       0.01      0.02      0.01      1124
```

```
Confusion Matrix:
 [[[43210    14]
  [ 3903   745]]

 [[47351     5]
  [  516     0]]

 [[45290    11]
  [ 2320   251]]

 [[47720     2]
  [  149     1]]

 [[45469    12]
  [ 2313    78]]

 [[47435     5]
  [  432     0]]]
```

DecisionTreeClassifier:

```
Accuracy Score 0.8938210227272727
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.72      0.70      4428
           1       0.23      0.34      0.28       354
           2       0.77      0.77      0.77      2576
           3       0.19      0.30      0.24        96
           4       0.61      0.63      0.62      2341
           5       0.32      0.45      0.38       313

   micro avg       0.65      0.68      0.66     10108
   macro avg       0.47      0.53      0.50     10108
weighted avg       0.66      0.68      0.67     10108
 samples avg       0.06      0.06      0.06     10108
```

```
Confusion Matrix:
 [[[41986  1238]
  [ 1458  3190]]

 [[47123   233]
  [  395   121]]

 [[44696   605]
  [  600  1971]]

 [[47655    67]
  [  121    29]]

 [[44609   872]
  [  922  1469]]

 [[47267   173]
  [  292   140]]]
```


#Kneighbor Bayes

```
Accuracy Score 0.8333054812834224
Classification Report:
              precision    recall  f1-score   support

           0       0.23      0.24      0.24      4533
           1       0.09      0.40      0.15       116
           2       0.20      0.24      0.22      2162
           3       0.06      0.82      0.11        11
           4       0.14      0.76      0.24       438
           5       0.06      0.60      0.10        40

   micro avg       0.19      0.28      0.23      7300
   macro avg       0.13      0.51      0.18      7300
weighted avg       0.22      0.28      0.23      7300
 samples avg       0.02      0.02      0.02      7300
```

```
Confusion Matrix:
 [[[39782  3442]
  [ 3557  1091]]

 [[47286    70]
  [  470    46]]

 [[43663  1638]
  [ 2047   524]]

 [[47720     2]
  [  141     9]]

 [[45376   105]
  [ 2058   333]]

 [[47424    16]
  [  408    24]]]
```

Logistc Regression

```
Accuracy Score 0.9159634024064172
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.93      0.70      2799
           1       0.21      0.58      0.31       187
           2       0.60      0.92      0.73      1664
           3       0.09      0.78      0.17        18
           4       0.47      0.84      0.60      1342
           5       0.15      0.72      0.24        88

   micro avg       0.51      0.89      0.65      6098
   macro avg       0.35      0.79      0.46      6098
weighted avg       0.53      0.89      0.67      6098
 samples avg       0.04      0.05      0.05      6098
```

```
Confusion Matrix:
 [[[43032   192]
  [ 2041  2607]]

 [[47277    79]
  [  408   108]]

 [[45175   126]
  [ 1033  1538]]

 [[47718     4]
  [  136    14]]

 [[45263   218]
  [ 1267  1124]]

 [[47415    25]
  [  369    63]]]
```

Applied Boosting Techniques:

RandomForestClassifier

```
Accuracy Score 0.9136238302139037
Classification Report:
              precision    recall  f1-score   support

           0       0.55      0.93      0.69      2769
           1       0.04      0.39      0.08        59
           2       0.63      0.91      0.74      1791
           3       0.03      0.71      0.06         7
           4       0.45      0.83      0.58      1302
           5       0.05      0.68      0.09        31

   micro avg       0.50      0.89      0.64      5959
   macro avg       0.29      0.74      0.38      5959
weighted avg       0.55      0.89      0.67      5959
 samples avg       0.04      0.05      0.04      5959
```

```
Confusion Matrix:
 [[[43026   198]
  [ 2077  2571]]

 [[47320    36]
  [  493    23]]

 [[45131   170]
  [  950  1621]]

 [[47720     2]
  [  145     5]]

 [[45256   225]
  [ 1314  1077]]

 [[47430    10]
  [  411    21]]]
```

GradientBoostingClassifier

```
Accuracy Score 0.9091535762032086
Classification Report:
              precision    recall  f1-score   support

           0       0.41      0.94      0.57      2025
           1       0.18      0.48      0.26       192
           2       0.60      0.90      0.72      1714
           3       0.13      0.27      0.17        70
           4       0.43      0.82      0.56      1249
           5       0.28      0.51      0.36       238

   micro avg       0.44      0.86      0.58      5488
   macro avg       0.34      0.65      0.44      5488
weighted avg       0.46      0.86      0.59      5488
 samples avg       0.03      0.04      0.04      5488
```

```
Confusion Matrix:
 [[[43096   128]
  [ 2751  1897]]

 [[47257    99]
  [  423    93]]

 [[45137   164]
  [ 1021  1550]]

 [[47671    51]
  [  131    19]]

 [[45253   228]
  [ 1370  1021]]

 [[47324   116]
  [  310   122]]]
```

AdaBoosting :

```
Accuracy Score 0.9076913435828877
Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.86      0.64      2737
           1       0.25      0.45      0.32       282
           2       0.60      0.90      0.72      1724
           3       0.27      0.51      0.35        79
           4       0.38      0.81      0.52      1132
           5       0.27      0.54      0.36       219

   micro avg       0.48      0.83      0.61      6173
   macro avg       0.38      0.68      0.49      6173
weighted avg       0.49      0.83      0.61      6173
 samples avg       0.04      0.05      0.04      6173


Confusion Matrix:
 [[[42853   371]
  [ 2282  2366]]

 [[47202   154]
  [  388   128]]

 [[45122   179]
  [ 1026  1545]]

 [[47683    39]
  [  110    40]]

 [[45263   218]
  [ 1477   914]]

 [[47339   101]
  [  314   118]]]
```


Note: By observing various outputs of the difference machine learning algorithm, Logistic Regression is giving best result, In Logistic Regression training and testing accuracy is also very close to each other , therefore Logistic algorithm selected as final machine learning algorithm to train the dataset for final model

➢ Ensemble Technique of Logistic Regression:

```
parameters ={"estimator__penalty":["l2","none"],
         "estimator__fit_intercept":[True,False],
         "estimator__solver":["newton-cg","lbfgs","liblinear","sag","saga"]}
```

```
model_to_set = OneVsRestClassifier(LogisticRegression())
model_to_set.get_params().keys()
```

```
dict_keys(['estimator__C', 'estimator__class_weight', 'estimator__dual', 'estimator__fit_intercept', 'estimator__intercept_scaling', 'estimator__l1_
```

```
model_tunning = GridSearchCV(estimator = model_to_set, param_grid=parameters, cv = 3)
model_tunning.fit(x_train, y_train)
model_tunning.best_params_
```

```
{'estimator__fit_intercept': True,
 'estimator__penalty': 'l2',
 'estimator__solver': 'liblinear'}
```

Note: I have applied ensemble technique by using various parameters of Logistic Regression, and got best parameter here.

➢ Final Model (RandomForestClassifer)

```
model = LogisticRegression(fit_intercept = 'True', penalty = 'l2', solver = 'liblinear')
final_model = OneVsRestClassifier(model).fit(x_train, y_train)
prediction = final_model.predict(x_test)
prediction2 = final_model.predict(x_train)
print('Accuracy of Testing ',accuracy_score(prediction, y_test))
print('Accuracy of Training ',accuracy_score(prediction2, y_train))
print('Classification Report: \n', classification_report(prediction, y_test) )
print('Confusion Matrix: \n', multilabel_confusion_matrix(y_test,prediction) )
```

```
Accuracy of Testing  0.9159634024064172
Accuracy of Training  0.9242965469699819
Classification Report:
               precision    recall  f1-score   support

           0       0.56      0.93      0.70      2797
           1       0.21      0.58      0.31       187
           2       0.60      0.92      0.73      1663
           3       0.09      0.78      0.17        18
           4       0.47      0.84      0.60      1343
           5       0.15      0.72      0.24        88

   micro avg       0.51      0.89      0.65      6096
   macro avg       0.35      0.79      0.46      6096
weighted avg       0.53      0.89      0.67      6096
 samples avg       0.04      0.05      0.05      6096
```

```
Confusion Matrix:
 [[[43032   192]
  [ 2043  2605]]

 [[47277    79]
  [  408   108]]

 [[45176   125]
  [ 1033  1538]]

 [[47718     4]
  [  136    14]]

 [[45263   218]
  [ 1266  1125]]

 [[47415    25]
  [  369    63]]]
```

Deploy the model

```
[ ] import pickle
    filename = 'comment_project.pkl'                # model name
    pickle.dump(final_model, open(filename, 'wb'))        # operation to deploy model
```

```
[ ]
```

Loading model

```
[ ] load_model =  pickle.load(open('comment_project.pkl', 'rb'))    # loading deployed model
    result = load_model.score(x_test, y_test)
    print(result)

    0.9159634024064172
```

Conclusion

```
[ ] original = np.array(y_test)
    predicted = np.array(load_model.predict(x_test))
    # convert columns in to np.array
```

```
[ ] print(predicted.shape)
    print(original.shape)
    print(x_test.shape)
    print(y_test.shape)
```

```
(47872, 6)
(47872, 6)
(47872, 180221)
(47872, 6)
```

```
x.shape
```

```
(159571, 180221)
```

```
test_x.shape
```

```
(153164, 180221)
```

Testing Data

```
pred_for_x_test = np.array(load_model.predict(test_x))
```

```
pred_for_x_test
```

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

➢ Prediction for Testing Dataset

```
df_test.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	00001cee341fdb12	yo bitch ja rule succesful ever whats hating s...	1	0	1	0	1	0
1	0000247867823ef7	rfc title fine imo	0	0	0	0	0	0
2	00013b17ad220c46	sources zawe ashton lapland	0	0	0	0	0	0
3	00017563c3f7919a	look back source information updated correct f...	0	0	0	0	0	0
4	00017695ad8997eb	anonymously edit articles	0	0	0	0	0	0

Note: As dataset was in two set one was for testing and second one was for testing, So accordingly machine is trained by train dataset and then predicted output for testing dataset as shown above image.

➢ Hardware and Software Requirements and Tools Used

All used libraries:


```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from scipy.sparse import csr_matrix
from sklearn.metrics import multilabel_confusion_matrix, f1_score
from sklearn.multiclass import OneVsRestClassifier
naive = MultinomialNB()
```

```
# imp libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Pandas: This library used for dataframe operations

Numpy: This library gives statistical computation for smooth functioning

Matplotlib: Used for visualization

Seaborn: This library is also used for visualization

Sklearn: This library having so many machine learning module and we can import them from this library

Pickle: This is used for deploying the model

Imblearn: This library is import to get SMOTE technique for balance the data

Scipy: It is import to perform outlier removing technique using zscore

Warning: To avoid unwanted warning shows in the output

I am giving this requirement and tool used, based on my laptop configuration.

Operating System:	**Window 11**
RAM:	**8 GB**
Processor:	**i5 10th Generation**
Software:	**Jupyter Notebook,**

Observations from the whole problem.

i) As in this problem we are having multiple label or target columns So here I have used OneVsRestClassifier model.

ii) As Logistic regression is giving best accuracy , So we tells that sometimes basic algorithm also perform well.

iii) As data set is very large due which every execution had taken so much time to execute.

iv) TfidVectorizer is used to covert Cleaned comment column into vector, after that one can perform machine learning on it.

Learning Outcomes of the Study in respect of Data Science

My learnings: - the power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say,

Various algorithms I have used in this dataset and to get out best result and save that model. The best algorithm is Logistic Regression.

Ensemble operation was giving biggest challenge which I have faced while working and as this dataset is very large which have leads to take lot of time for machine learning.

Limitations of this work and Scope for Future Work

No as such limitation found for this model. But yes if one wants more accuracy then we can train our model with more data.