MALIGNANT COMMENT CLASSIFIER

Submitted by:

VIKAS MISHRA`

ACKNOWLEDGMENT

1. FlipRobo Project Documentation.

2. Project Data file.

3. Project Dataset Description.

4. Data trained Training Knowledge.

5. Sample Project references from Google.

INTRODUCTION

- Business Problem Framing

Problem Statement :

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.

The objective of the problem is to pass target column ‘comment text’ into various ML Algo to predict the comment falls into which Multi- classified columns (['malignant', 'highly_malignant', 'rude', 'threat','abuse', 'loathe']).

- Conceptual Background of the Domain Problem

As per my understanding following knowledge is important to understand and to solve problem.

1. Understanding Classification and Multiclassification Multilabel problems and how to solve Multiclassification Machine learning Problems.
2. As the target column is text So One must have NLP knowledge to process and clean textual data.
3. Exploratory Data Analysis on Textual Data knowledge is also important.
4. This problem can be solved through deep learning as well with LSTM etc,

- Review of Literature :

I have reviewed various literatures on internet to understand toxic comment classification problems

- Motivation for the Problem Undertaken

Describe your objective behind to make this project, this domain and what is the motivation behind.

My interest to solve this problem as the problem was new for me to solve MultiClass and Multilabel problems, Second was processing textual data and applying NLP knowledge to solve the problem also to perform EDA on textual data.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

1.Initial Analysis : The train & test dataset has 8 column where target column is 'comment_text' and other columns are classified column for the type of comment present in target column, the target column has offensive, threat, abusing, vulgar type of comments which are classified as malicious, highly malicious, abuse, rude, threat and loathe accordingly those columns are present in a dataset. The main objective of problem to classify the comments as per those classified columns. further each toxic classified column is further binary classifed as '0' and '1', where the presence of '1' indicate the severity of toxic comment and the presence of '0' means there is no toxic comment present in a target column. The ID

and Target Column is Object Type and remaining classification column is Int data type.

Above information is analysed by understanding the shape of data, data type of train and test set.

2.Univarate Plots : The following below plots helps us to know about variables information

A) Histplot : helps us to know about size of length of comments .

B) Countplot : With the help of count plot it is observed that all the toxic multiclass columns is highly imbalanced and because of this we cannot trust only on accuracy and AUC score metric for selecting the best model, Hence humming and Log Loss metric is also included for analysis which will correctly help to select the best model based on analysis of all the metrics.

C) WordCloud : With the help of wordcloud it is Observed below said observation:

Trainset : From partitioned train set we can see following(article, Wikipedia, known, One etc) words is most frequently occurred.

Testset : And from partitioned testset following words is most frequently occurred (article, think, source, Thank, page etc.)

3.BiVariate Analysis : The correlation plot is plotted for all the multiclass columns which has binary label '0' & '1' the

correlation plot show some positive correlation as mentioned below.

- malignant shows positive correlation with abuse and rude column.

- rude shows positive correlation with abuse and malignant.

- abuse shows positive correlation with rude and malignant.

4.Partioning of Train and Testset :The train and test dataset size is reduced and partitioned to smaller size in order to process the smaller size data for further analysis and model building, As the system was not able to process such large size data and was throwing memory error.

5.Cleaning of Target Column for Text Processing : The target column comment text needs to be cleaned for the words which are not relevant and must be removed for further analysis and model building, hence below steps are taken to clean the target column for further analysis and model building.

A)Stopwords : Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive/negative impact on our statement and also do not convey any useful meaning and hence can be directly removed.

B)Stemming : Stemming is the process of reducing a word to its stem or root format. For example three words, “branched”, “branching” and “branches”. They all can be reduced to the same word “branch”. After all, all the three convey the same idea of something separating into multiple paths or branches. Again, this helps reduce complexity while retaining the essence of meaning carried by these three words. This helps in achieving the training process with a better accuracy. The snowball stemmer is applied to the target column ‘comment.text’ through df.apply method.

C) Word Vectorizer : TfidfVectorizer - As target column 'comment_text' contains text which needs to be converted into numbers to process and build the model, So TfidfVectorizer is used as one of the method to convert text into numbers. tf-idf is used to classify documents, ranking in search engine. tf: term frequency(count of the words present in document from its own vocabulary), idf: inverse document frequency(importance of the word to each document).The method addresses the fact that all words should not be weighted equally, using the weights to indicate the words that are most unique to the document, and best used to characterize it.TfidfVectorizer Transforms text to feature vectors that can be used as input to estimator. Each sentence is a vector. In each vector the numbers (weights) represent features tf-idf score, So tf-idf creates a set of its own vocabulary from the entire set of documents which is seen in first line of output. (For better understanding I have sorted it) where vocabulary_ Is a dictionary that converts each token (word) to feature index in the matrix, each unique token gets a feature index.

6.Separating train set ino target x & independet y dataset: The X_raw is target column'comment_text' and all multiclass column is acting as Y dataframe.

7.Model building : The following Classification ML Algorithm used to build the model (logistic, Decision tree, RandomForest,K-NN,XGBoost, Binaryrelevance-SVM, Binaryrelevance-MultinomialNB).K-fold Cross Validation and halvingGridsearchCV used to tune hyper parameter on above selected Algorithm(To Prevent Over-fitting and Under-fitting of the model).

8.Metrics Calculation : As the multiclass column is highly class imbalanced where we cannot depend only on accuracy/AUC Score, hence humming & log loss is also used to select best Model.

9.Model Selection: The best model which will perform well is logistic regression as it shows less humming loss and high accuracy, AUC score on y_test_malignant as well on y-test multiclass labels.

- **Data Sources and their formats:**

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 which will be treated as target column or target variable.

	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
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0
10	0005300084f90edc	"\nFair use rationale for Image:Wonju.jpg\n\nT...	0	0	0	0	0	0
11	00054a5e18b50dd4	bbq \n\nbe a man and lets discuss it-maybe ove...	0	0	0	0	0	0

```
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
```

```
print(df.shape)
```

```
(159571, 8)
```

The ID and Target Column is Object Type and remaining classification column is Int data type.

- **Data Preprocessing Done:**

  The target column comment text needs to be cleaned for the words which are not relevant and must be removed for further analysis and model building, hence below steps are taken to clean the target column for further analysis and model building.

  1.Missing Value : The train and test set do no have missing values.

  2. Stopwords : Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive/negative impact on our statement and also do not convey any useful meaning and hence can be directly removed.

  3. Stemming : Stemming is the process of reducing a word to its stem or root format. for example three words, “branched”, “branching” and “branches”. They all can be reduced to the same word “branch”. After all, all the three convey the same idea of something separating into multiple paths or branches. Again, this helps reduce complexity while retaining the essence of meaning carried by these three words.This helps in achieving the training process with a better accuracy.The snowball stemmer is applied to the target column comment.text through df.apply method.

4. Word Vectorizer : TfidfVectorizer - As target column commnet_text contains text which needs to be converted into numbers to process and build the model, So TfidfVectorizer is used as one of the method to convert text into numbers. tf-idf is used to classify documents, ranking in search engine. tf: term frequency(count of the words present in document from its own vocabulary), idf: inverse document frequency(importance of the word to each document).The method addresses the fact that all words should not be weighted equally, using the weights to indicate the words that are most unique to the document, and best used to characterize it.TfidfVectorizer Transforms text to feature vectors that can be used as input to estimator. Each sentence is a vector. In each vector the numbers (weights) represent features tf-idf score, So tf-idf creates a set of its own vocabulary from the entire set of documents. Which is seen in first line of output. (For better understanding I have sorted it) where vocabulary_ Is a dictionary that converts each token (word) to feature index in the matrix, each unique token gets a feature index.

- Data Inputs- Logic- Output Relationships

The dataset has target column 'comment_text' which contains audience comments, So based on audience comments multiclass toxic comment category or column is created namely ('malignant', 'highly_malignant', 'rude', 'threat','abuse', 'loathe') in the dataset to classify the severity of toxicity of comments.

Each multiclass toxic column has binary class as zero and one where presence of one indicate toxicity of comment.

Hence, The objective of the problem is to pass target column 'comment text' into various ML Algo to predict the comment

falls into which Multi- classified columns (['malignant', 'highly_malignant', 'rude', 'threat','abuse', 'loathe']).

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

Partitioning Trainset

```
import pandas as pd
df = pd.read_csv("train.csv")

# no of csv files with row size
k = 1
size = 6383

for i in range(k):
    df = df[size*i:size*(i+1)]
    df.to_csv(f'train_{i+1}.csv', index=False)


df_1 = pd.read_csv("train_1.csv")
print(df_1)
```

	malignant	highly_malignant	rude	threat	abuse	loathe
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
...	...	...	...	...	...	...
6378	0	0	0	0	0	0
6379	0	0	0	0	0	0
6380	0	0	0	0	0	0
6381	0	0	0	0	0	0
6382	0	0	0	0	0	0

[6383 rows x 8 columns]

The train & test dataset has 8 column where target column is 'comment_text' and other columns are classified column for the type of comment present in target column, the target column has offensive, threat, abusing, vulgar type of

comments which are classifed as malicious, highly malicious, abuse, rude, threat and loathe accordingly those columns are present in a dataset. The main objective of problem to classify the comments as per those classified columns. further each toxic classified column is further binary classifed as '0' and '1', where the presence of '1' indicate the severity of toxic comment and the presence of '0' means there is no toxic comment present in a target column.The train and test dataset size is reduced and partioned to smaller size inorder to process the smaller size data for further analysis and model building as system was not able to process such large size data and was throwing memory error.

- Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

Hardware :

1. Laptop.
2. Software Tools : Anaconda, Spider & Jupyter, Python Module and Libraries.

Note : Detailed Description is mentioned in the Jupyter file.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

1.Initial Analysis : The train & test dataset has 8 column where target column is 'comment_text' and other columns are classified column for the type of comment present in target column, the target column has offensive, threat, abusing, vulgar type of comments which are classified as malicious, highly malicious, abuse, rude, threat and loathe accordingly those columns are present in a dataset. The main objective of problem to classify the comments as per those classified columns. further each toxic classified column is further binary classifed as '0' and '1', where the presence of '1' indicate the severity of toxic comment and the presence of '0' means there is no toxic comment present in a target column. The ID and Target Column is Object Type and remaining classification column is Int data type.
Above information is analysed by understanding the shape of data, data type of train and test set.

2.Univarate Plots : The following below plots helps us to know about variables information
A) Histplot : helps us to know about size of length of comments .
B) Countplot : With the help of count plot it is observed that all the toxic multiclass columns is highly imbalanced and because of this we cannot trust only on accuracy and AUC score metric for selecting the best model, Hence humming and Log Loss metric is also included for analysis which will correctly help to select the best model based on analysis of all the metrics.
C) WordCloud : With the help of wordcloud it is Observed below said observation:

Trainset : From partitioned train set we can see following(article, Wikipedia, known, One etc) words is most frequently occurred.
Testset : And from partitioned testset following words is most frequently occurred (article, think, source, Thank, page etc.)

3.BiVariate Analysis : The correlation plot is plotted for all the multiclass columns which has binary label '0' & '1' the correlation plot show some positive correlation as mentioned below.
- malignant shows positive correlation with abuse and rude column.
- rude shows positive correlation with abuse and malignant.
- abuse shows positive correlation with rude and malignant.

4.Partioning of Train and Testset :The train and test dataset size is reduced and partitioned to smaller size in order to process the smaller size data for further analysis and model building, As the system was not able to process such large size data and was throwing memory error.

5.Cleaning of Target Column for Text Processing : The target column comment text needs to be cleaned for the words which are not relevant and must be removed for further analysis and model building, hence below steps are taken to clean the target column for further analysis and model building.

A)Stopwords : Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive/negative impact on

our statement and also do not convey any useful meaning and hence can be directly removed.

B)Stemming : Stemming is the process of reducing a word to its stem or root format. For example three words, “branched”, “branching” and “branches”. They all can be reduced to the same word “branch”. After all, all the three convey the same idea of something separating into multiple paths or branches. Again, this helps reduce complexity while retaining the essence of meaning carried by these three words. This helps in achieving the training process with a better accuracy. The snowball stemmer is applied to the target column ‘comment.text’ through df.apply method.

C) Word Vectorizer : TfidfVectorizer - As target column ‘comment_text’ contains text which needs to be converted into numbers to process and build the model, So TfidfVectorizer is used as one of the method to convert text into numbers. tf-idf is used to classify documents, ranking in search engine. tf: term frequency(count of the words present in document from its own vocabulary), idf: inverse document frequency(importance of the word to each document).The method addresses the fact that all words should not be weighted equally, using the weights to indicate the words that are most unique to the document, and best used to characterize it.TfidfVectorizer Transforms text to feature vectors that can be used as input to estimator. Each sentence is a vector. In each vector the numbers (weights) represent features tf-idf score, So tf-idf creates a set of its own vocabulary from the entire set of documents which is seen in first line of output. (For better understanding I have sorted it) where vocabulary_ Is a dictionary that converts each token

(word) to feature index in the matrix, each unique token gets a feature index.

6.Separating train set ino target x & independet y dataset: The X_raw is target column'comment_text' and all multiclass column is acting as Y dataframe.

7.Model building : The following Classification ML Algorithm used to build the model (logistic, Decision tree, RandomForest,K-NN,XGBoost, Binaryrelevance-SVM, Binaryrelevance-MultinomialNB).K-fold Cross Validation and halvingGridsearchCV used to tune hyper parameter on above selected Algorithm(To Prevent Over-fitting and Under-fitting of the model).

8.Metrics Calculation : As the multiclass column is highly class imbalanced where we cannot depend only on accuracy/AUC Score, hence humming & log loss is also used to select best Model.

9.Model Selection: The best model which will perform well is logistic regression as it shows less humming loss and high accuracy, AUC score on y_test_malignant as well on y-test multiclass labels

- Testing of Identified Approaches (Algorithms)
    1. Logistic Regression-Best Model
    2. Decisiontree Classifier
    3. Random Forest Classifier
    4. K-Nearest Neigbors Classifier
    5. XGBoost Classifier
    6. BinaryRelevance- SVM
    7. BinaryRelevance-MultinomialNB
- Run and Evaluate selected models

Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

Hamming & Log Loss Common Python Function for ML Algo

```
from sklearn.metrics import hamming_loss
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss
def evaluate_score(Y_test,predict):
    loss = hamming_loss(y_test,predict)
    print("Hamming_loss : {}".format(loss*100))
    accuracy = accuracy_score(y_test,predict)
    print("Accuracy : {}".format(accuracy*100))
    try :
        loss = log_loss(y_test,predict)
    except :
        loss = log_loss(y_test,predict.toarray())
    print("Log_loss : {}".format(loss))
```

BibaryRelevance-SVM Classifier

```
#create and fit classifier
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.svm import SVC
classifier = BinaryRelevance(classifier = SVC(), require_dense = [False, True])
classifier.fit(X_train_transformed, y_train)
predictions = classifier.predict(X_test_transformed)
evaluate_score(y_test,predictions)
```

```
Hamming_loss : 2.936570086139389
Accuracy : 89.50665622552859
Log_loss : 1.2757319646626866
```

BinaryRelevance-MultinomialNB Classifier

```
from sklearn.naive_bayes import MultinomialNB
classifier = BinaryRelevance(classifier = MultinomialNB(), require_dense = [False, True])
classifier.fit(X_train_transformed, y_train)
predictionsNB = classifier.predict(X_test_transformed)
evaluate_score(y_test,predictionsNB)
```

```
Hamming_loss : 3.9415296267293134
Accuracy : 88.72357086922474
Log_loss : 0.55195492900179
```

Logistic Regression

```
parameter = {'penalty': ['l2'],
             'C': [0.5,0.75,1],
             'class_weight': ['balanced',None],
             'solver': ['liblinear']}
```

```
cross_validation=KFold(5, shuffle =False)
```

```
GCV=HalvingGridSearchCV(LogisticRegression(),parameter,cv=cross_validation)
```

```
GCV.fit(X_train_transformed,y_train.malignant)
```

```
HalvingGridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
                    estimator=LogisticRegression(),
                    param_grid={'C': [0.5, 0.75, 1],
                                'class_weight': ['balanced', None],
                                'penalty': ['l2'], 'solver': ['liblinear']},
                    refit=<function _refit_callable at 0x000000001DB3C620>)
```

```
GCV.best_params_
# Best Parameter is selected from grid search CV which is
#'criterion': 'gini','max_depth': 9,'max_features': 'sqrt','splitter': 'best'
```

```
{'C': 0.5, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'liblinear'}
```

```
LR = LogisticRegression(C = 0.5, penalty='l2', solver = 'liblinear',class_weight = 'balanced')
LR.fit(X_train_transformed,y_train.malignant)
y_pred_LR = LR.predict(X_test_transformed)
print(f'Roc_auc at test:  = {roc_auc_score(y_test.malignant,y_pred_LR)}')
acc=accuracy_score(y_test.malignant,y_pred_LR)
print(acc*100)
```

```
Roc_auc at test:  = 0.835973584761614
94.59671104150353
```

OneVsRestClassifier(Log Regn) : To predict all multiclass labels on y_test

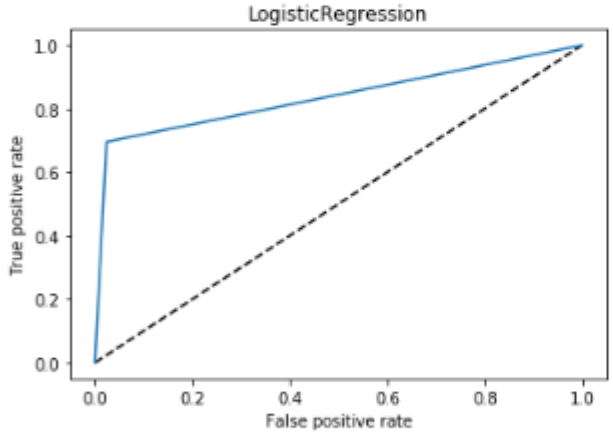
```
LROne=  LogisticRegression(C = 0.5, penalty='l2', solver = 'liblinear',class_weight = 'balanced')
classifier_ovr_LROne = OneVsRestClassifier(LROne)
classifier_ovr_LROne.fit(X_train_transformed, y_train)
y_LROne_classifer_over = classifier_ovr_LROne.predict(X_test_transformed)
print("ROC AUC Score Test:", roc_auc_score(y_test, y_LROne_classifer_over))
print(classification_report(y_test, y_LROne_classifer_over))
```

```
ROC AUC Score Test: 0.8026535100900768
              precision    recall  f1-score   support

           0       0.78      0.70      0.74       138
           1       0.28      0.64      0.39        11
           2       0.92      0.74      0.82        74
           3       1.00      0.50      0.67         2
           4       0.66      0.62      0.64        69
           5       0.38      0.50      0.43        10

   micro avg       0.72      0.68      0.70       304
   macro avg       0.67      0.62      0.61       304
weighted avg       0.76      0.68      0.71       304
 samples avg       0.06      0.07      0.06       304
```

AUC-ROC Score & Curve for Logistic Regression:

```
y_pred_prob= LR.predict_proba(X_test_transformed)[:,1]
fpr,tpr,thresholds=roc_curve(y_test.malignant, y_pred_LR,pos_label= 1)
```

```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='LogisticRegression')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('LogisticRegression')
plt.show()
auc_score=roc_auc_score(y_test.malignant,LR.predict(X_test_transformed))
auc_score
```




```
0.835973584761614
```

```
auc_score=roc_auc_score(y_test, classifier_ovr_LROne.predict(X_test_transformed))
auc_score
```

```
0.994811480788453
```

Prediction on Test Data Part 1 for Logistic Regression

```
LR = LogisticRegression(C = 0.5, penalty='l2', solver = 'liblinear',class_weight = 'balanced')
classifier_ovr_LROne = OneVsRestClassifier(LROne)
classifier_ovr_LROne.fit(X_raw_transformed, y)
pred_on_testdataLR1 = classifier_ovr_LROne.predict(X_raw_test_transformed)
print(pred_on_testdataLR1)
```

```
[[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]
 [0 0 0 0 0 0]]
```

Prediction on Test Data Part 2 for Logistic Regression

```
LR = LogisticRegression(C = 0.5, penalty='l2', solver = 'liblinear',class_weight = 'balanced')
classifier_ovr_LROne = OneVsRestClassifier(LROne)
classifier_ovr_LROne.fit(X_raw_transformed, y)
pred_on_testdata2LR = classifier_ovr_LROne.predict(X_raw_test_transformed2)
print(pred_on_testdata2LR)
```

```
[[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]
 [0 0 0 0 0 0]]
```

XGBoost Classifier

```
import numpy as np
parameter = {'max_depth' : [8,10],
             'learning_rate':[0.1,0.01],
             'gamma' : [0],
             'n_estimators' : [12]}
```

```
cross_validation=KFold(5, shuffle =False)
```

```
GCV=HalvingGridSearchCV(XGBClassifier(),parameter,cv=cross_validation)
```

```
GCV.fit(X_train_transformed,y_train.malignant)
# Grid search CV applied to the Training Dataset
```

```
HalvingGridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
                    estimator=XGBClassifier(base_score=None, booster=None,
                                            colsample_bylevel=None,
                                            colsample_bynode=None,
                                            colsample_bytree=None, gamma=None,
                                            gpu_id=None, importance_type='gain',
                                            interaction_constraints=None,
                                            learning_rate=None,
                                            max_delta_step=None, max_depth=None,
                                            min_child_weight=None, missin...
                                            monotone_constraints=None,
                                            n_estimators=100, n_jobs=None,
                                            num_parallel_tree=None,
                                            random_state=None, reg_alpha=None,
                                            reg_lambda=None,
                                            scale_pos_weight=None,
                                            subsample=None, tree_method=None,
                                            validate_parameters=None,
                                            verbosity=None),
                    param_grid {'gamma': [0], 'learning_rate': [0.1, 0.01]
```

```
GCV.best_params_
```

```
{'gamma': 0, 'learning_rate': 0.01, 'max_depth': 10, 'n_estimators': 12}
```

Accuracy and Roc Score on y_test_malignant

```
XGBC = XGBClassifier(gamma = 0,learning_rate = 0.01,max_depth = 10, n_estimators=12)
XGBC.fit(X_train_transformed,y_train.malignant)
y_predXGBC = XGBC.predict(X_test_transformed)
print(f'Roc_auc at test:  = {roc_auc_score(y_test.malignant,y_predXGBC)}')
acc=accuracy_score(y_test.malignant,y_predXGBC)
print(acc*100)
```

```
Roc_auc at test:  = 0.728811187031594
93.65700861393891
```

OneVsRestClassifier(XGBoost Classifier) : To predict all multiclass labels on y_test

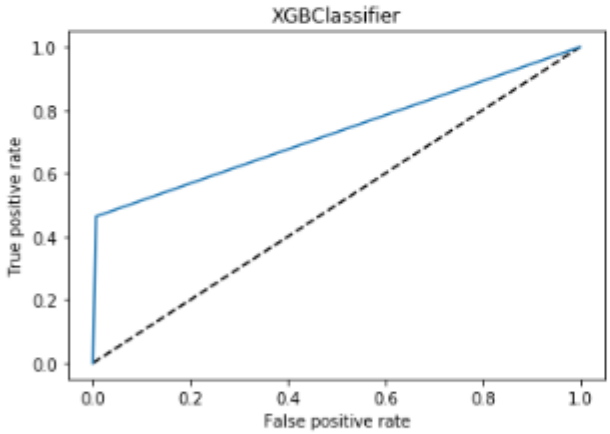
```
XGBC_One =  XGBClassifier(gamma = 0,learning_rate = 0.01,max_depth = 10, n_estimators=12)
classifier_ovr_XGBC_One = OneVsRestClassifier(XGBC_One)
classifier_ovr_XGBC_One.fit(X_train_transformed, y_train)
y_XGBC_One_pred = classifier_ovr_XGBC_One.predict(X_test_transformed)
print("ROC AUC Score Test:", roc_auc_score(y_test, y_XGBC_One_pred))
print(classification_report(y_test,y_XGBC_One_pred))
```

```
ROC AUC Score Test: 0.6732198509612464
              precision    recall  f1-score   support

           0       0.90      0.46      0.61       138
           1       0.30      0.27      0.29        11
           2       0.90      0.70      0.79        74
           3       0.00      0.00      0.00         2
           4       0.60      0.48      0.53        69
           5       0.29      0.20      0.24        10

   micro avg       0.77      0.51      0.61       304
   macro avg       0.50      0.35      0.41       304
weighted avg       0.78      0.51      0.61       304
 samples avg       0.05      0.05      0.04       304
```

AUC ROC Curve & Score for XGBoost Classifier

```
y_pred_prob= RFC.predict_proba(X_test_transformed)[:,1]
fpr,tpr,thresholds=roc_curve(y_test.malignant, y_predXGBC,pos_label= 1)
```

```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='XGBClassifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('XGBClassifier')
plt.show()
auc_score=roc_auc_score(y_test.malignant,XGBC.predict(X_test_transformed))
auc_score
```




```
0.728811187031594
```

```
auc_score=roc_auc_score(y_test, classifier_ovr_XGBC_One.predict(X_test_transformed))
auc_score
```

```
0.6732198509612464
```

Prediction on Testset Partition 1 for XGBC Classifiers

```
XGBC_One = XGBClassifier(gamma = 0,learning_rate = 0.01,max_depth = 10, n_estimators=12)
classifier_ovr_XGBC_One = OneVsRestClassifier(XGBC_One)
classifier_ovr_XGBC_One.fit(X_raw_transformed, y)
pred_on_testdataXGBCOne = classifier_ovr_XGBC_One.predict(X_raw_test_transformed)
print(pred_on_testdataXGBCOne)
```

```
[[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 0 0 0 0 0]
 [0 0 0 0 0 0]]
```

Prediction on Testset Partition 2 for XGBC Classifiers

```
XGBC_One =  XGBClassifier(gamma = 0,learning_rate = 0.01,max_depth = 10, n_estimators=12)
classifier_ovr_XGBC_One = OneVsRestClassifier(XGBC_One)
classifier_ovr_XGBC_One.fit(X_raw_transformed, y)
pred_on_testdataXGBCOne2 = classifier_ovr_XGBC_One.predict(X_raw_test_transformed2)
print(pred_on_testdataXGBCOne2)
```

```
[[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 0 0 0 0 0]
 [0 0 0 0 0 0]]
```

Model Summary and Performance:

Machine Learning Algorithm	Losses/Metrics					
	Hamming Loss	Log Loss	Accuracy-on malignant	F1-Score=y_test	roc_auc_score of y_test.malignant	roc_auc_score - y_test(all Classes)
Logistic Regression-Best Model	2.31	1.23	94.59	0.7	0.83	0.8
Decisiontree Classifier	3.96	0.42	89.19	0	0.5	0.5
Random Forest Classifier	3.96	0.42	89.19	0	0.5	0.5
K-Nearest Neigbors Classifier	3.79	0.47	89.58	0.09	0.518	0.519
XGBoost Classifier	2.57	1.32	93.65	0.61	0.72	0.67
BinaryRelevance- SVM	2.93	1.27				
BinaryRelevance-MultinomialNB	3.94	0.55				

- The following Classification Ml Algorithm used to build the model (logistic, Decisiontree, RandomForest,K-NN,XGBoost, Bianryrelevance-SVM, Binaryrelevance-MultinomialNB.)
- K-fold Cross Validation and halvingGridsearchCV used to tune hyper parameter on to top above selected Algorithm(To Prevent Over-fitting and Under-fitting of the model).

- As the multiclass column is highly class imbalanced where we cannot depend only on accuracy/AUC Score, hence humming & log loss is also used to select best Model.
- The best model which will perform well is logistic regression as it shows low humming loss and high accuracy, AUC score on y_test_malignant as well on y-test multiclass labels.

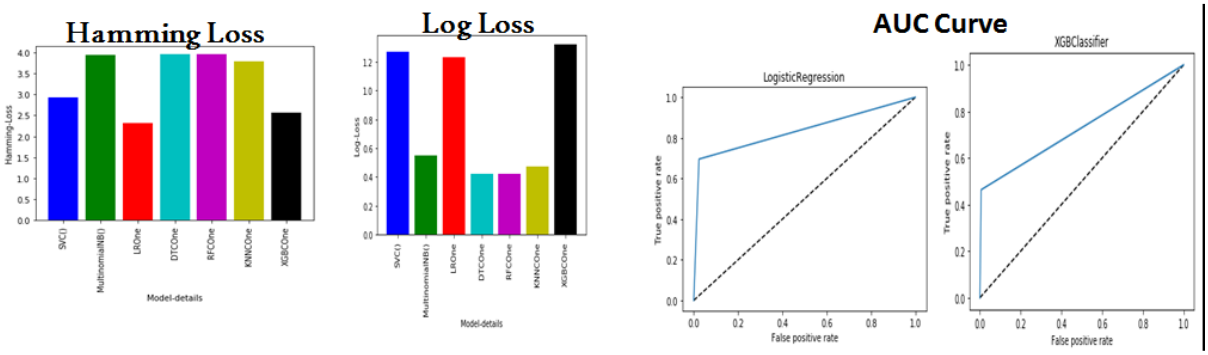
- Key Metrics for success in solving problem under consideration

  1.Hamming Loss.
  2.Log Loss.
  3.Accuracy-on y_test.malignant
  4.F1-Score of y_test(All Classes)
  5.roc_auc_score of y_test.malignant
  6.roc_auc_score of y_test(all Classes).

- **Humming Loss** : Is the fraction of the wrong labels to the total number of labels, Since hamming loss is designed for multi class while Precision, Recall, F1-Measure are designed for the binary class, it is better to compare the last one to Accuracy Also specifically for imbalanced problems, accuracy is a problematic metric. The Hamming loss lower being better.
- **Log-Loss:** is indicative of how close the prediction probability is to the corresponding actual/true value (0 or 1 in case of binary classification). The more the predicted probability diverges from the actual value, the higher is the log-loss value.but log-loss is still a good metric for comparing models. For any given problem, a lower log-loss value means better predictions.

  So, if we compare models based only on Log Loss than DecsionTree and RandomForest Classification model shows low Log Loss Values.

- The only logistic regression shows best AUC Curve & AUC Score as well as high accuracy among all followed by XGBoost Classifier.
- From Further analysis we can see Logistic regression shows low humming loss and high accuracy as well AUC Score followed by XGBoost Classifier.
- From Above Analysis Logistic regression will perform best among rest other ML Algorithm.



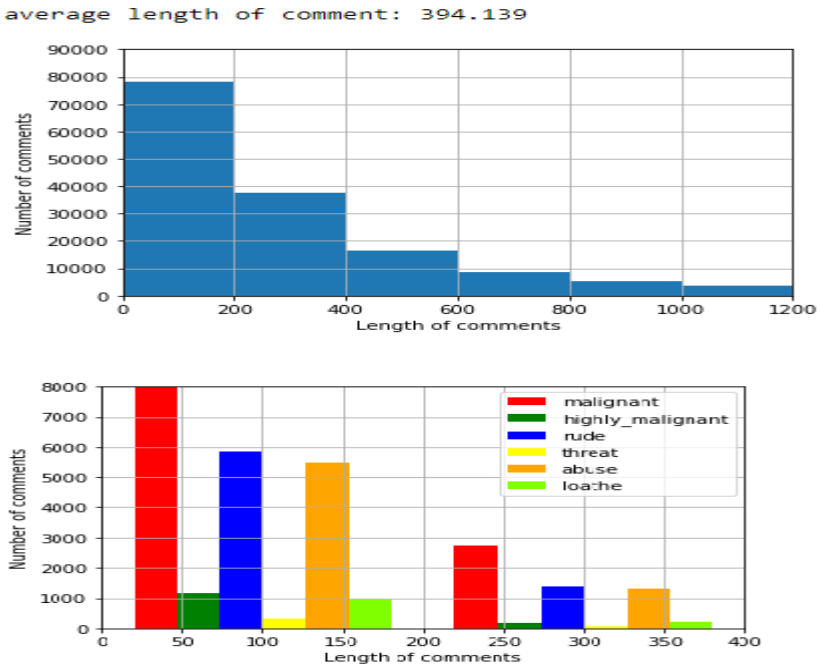
- **Visualizations**

A) **HistPlot :**

- From the first visualization we can see length of comment is large for the value 0-200 and thereafter it started to decrease from 200-400 and so on. the length of comment decrease for higher value, hence it is advisable to have small size of comments for processing.
- From the second figure we can observe that:

-Malignant : length of comments is large for the till 50
-Highly Malignant: very less from 50-25
-rude: high from value 25 to 100, but less then malignat
-threat: very less
-abuse: high from value 125 to 150 but less then rude and malignant.
-loathe : not so large, but greater than the value of threat.

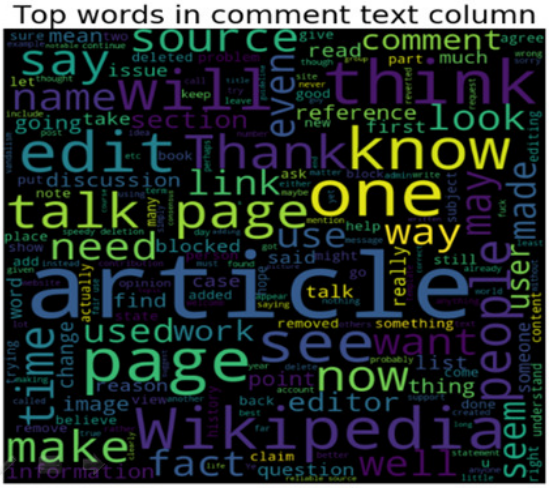
All above length of comments is started decreasing from the value 200 till 400 as the length of comments value increases.



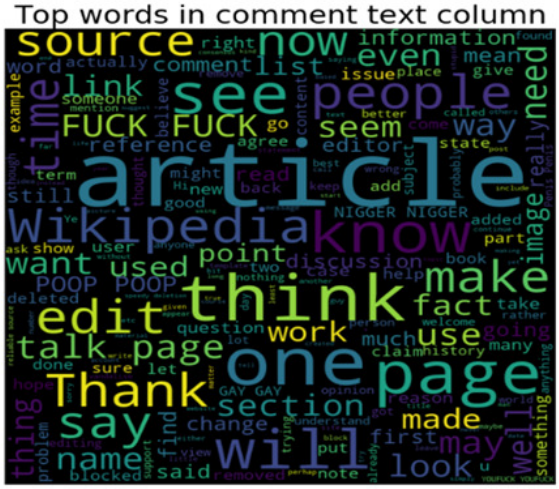
B) **Wordcloud :** Is a powerful way to visualise what the audience think about the Topic, which is a collection of a cluster of words occurring more frequently. The bigger and bolder the words the words appears, the more often it is voted by the audience. **Trainset :** From partitioned train set we can see following(article, Wikipedia, known, One etc) words is most frequently occurred.

Testset : And from partitioned testset following words is most frequently occurred (article, think, source, Thank, page etc.)

TRAINING SET WORDCLOUD



TESTSET WORDCLOUD

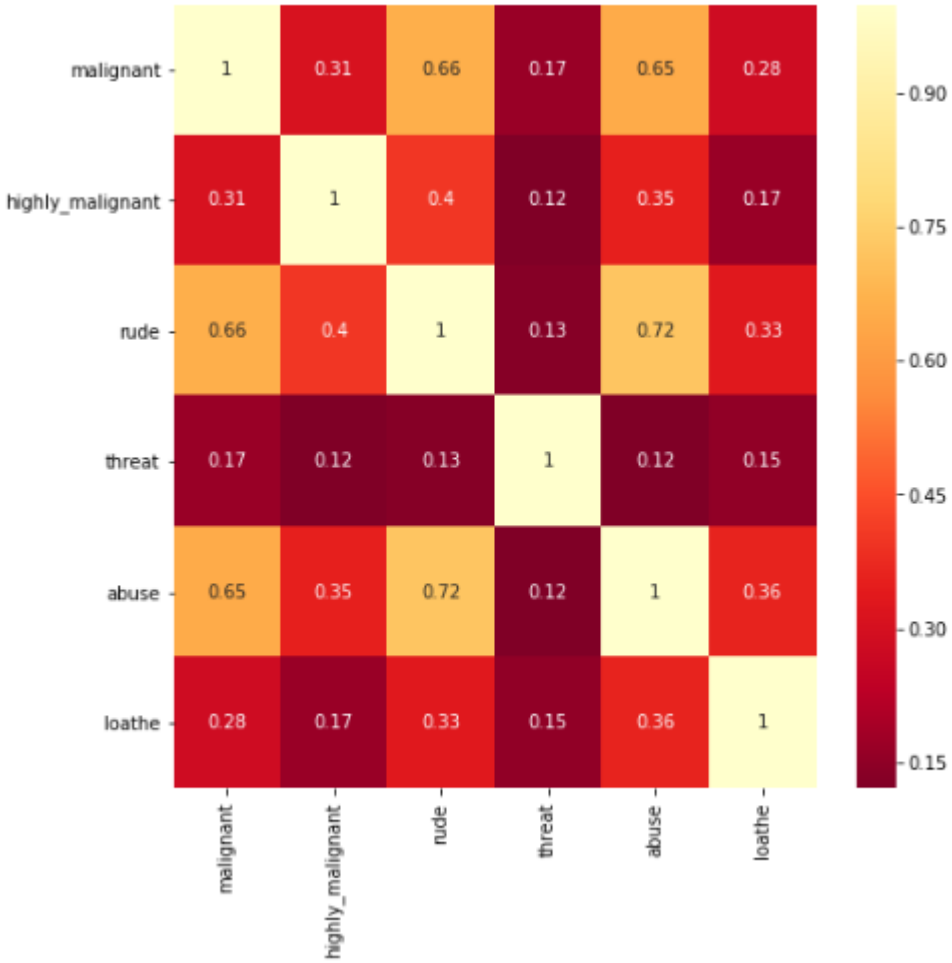


C) **Correlation Plot :**

The trainset is separated into X & y, where X is a target column 'comment_text' and y dataset contains toxic multiclass columns as shown below in corr plot.

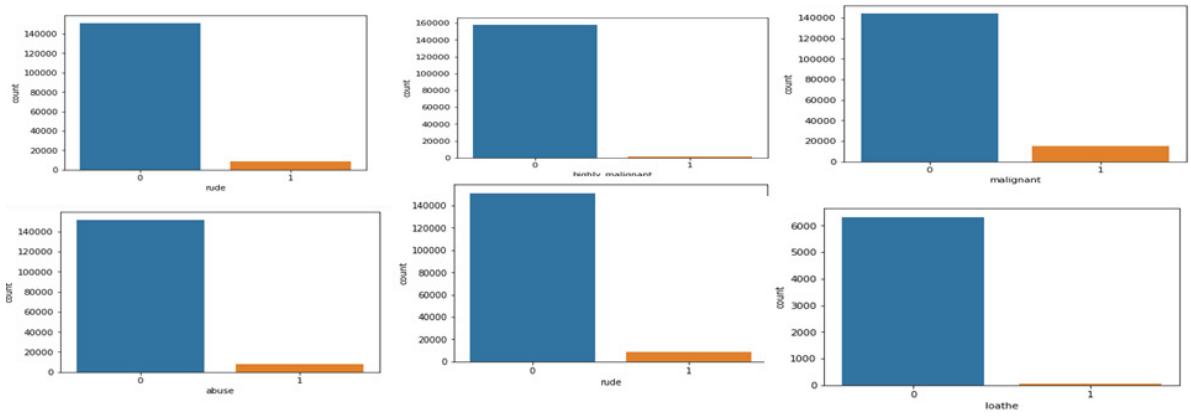
where we can see :

- malignant shows positive correlation with abuse and rude column.
- rude shows positive correlation with abuse and malignant.
- abuse shows positive correlation with rude and malignant.



D) CountPlot :

The dataset has target column and binary labelled multiclass column where the severity of toxic comments is classified as malignant, highly malignant etc as shown below, these toxic columns is further binary classified as 0 and 1 where 1 indicate the comment is toxic and falls into one of the severity of toxic class below.
Further With the help of countplot we can see all the toxic multiclass columns the binary class is highly imbalanced and because of this we cannot trust only on accuracy and AUC score metric for selecting the best model, Hence hamming and loss loss metric is also included for analysis which will correctly help to select the best model based on analysis of all the metrics.



- Interpretation of the Results

Give a summary of what results were interpreted from the

1.Visualizations: With the help of hist plot, wordcloud and Correlation it was possible to find the number of length of comments , the number of frequency of words occurring in train set and testset & also the correlation among multiclass variables.

Further with the help of Countplot it was possible to know class imbalance issue in multiclass variables.

2. Preprocessing: The target column comment text needs to be cleaned for the words which are not relevant and must be removed for further analysis and model building, hence below steps are taken to clean the target column for further analysis and model building.

i) Missing Value, ii) Removing Stopwords, iii) Stemminig & iv) TFID Vectorizer

3. Modelling.

- The following Classification Ml Algorithm used to build the model (logistic, Decisiontree, RandomForest,K-

NN,XGBoost, Bianryrelevance-SVM, Binaryrelevance-MultinomialNB.)

- K-fold Cross Validation and halvingGridsearchCV used to tune hyper parameter on to top above selected Algorithm(To Prevent Over-fitting and Under-fitting of the model).
- As the multiclass column is highly class imbalanced where we cannot depend only on accuracy/AUC Score, hence humming & log loss is also used to select best Model.
- The best model which will perform well is logistic regression as it shows low humming loss and high accuracy, AUC score on y_test_malignant as well on y-test multiclass labels.

CONCLUSION

- Key Findings and Conclusions of the Study

Describe the key findings, inferences, observations from the whole problem.

1) **Partitioning of Dataset:** The train and test Dataset size is large, so the size of both train and test size is reduced for further analysis, processing and to build the model.
2) **Selection of Best Model for multiclass variables :**
   i) As the multiclass column is highly class imbalanced where we cannot depend only on accuracy/AUC Score, hence humming & log loss is also used to select best Model.
   ii)The best model which will perform well is logistic regression as it shows low humming loss and high accuracy, AUC score on y_test_malignant as well on y-test multiclass labels.

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

Challenges Faced :

- Finding out and Performing EDA on Important Variables.
- Processing of large value of training and test dataset.
- Cleaning of text data on comment text column
- Calculation of hamming and log loss and applying on all model.
- Model building & Hype parameter tuning and selection of best model.

Overcoming Challenges:

- As it was a multiclassification problem, So with the help of Countplot it was possible to detect class imbalance issue, Corr plot detect correlated variables among multiclass variables, plotting wordcloud to know most repeated word occurring in train and test set and hist plot helps to know length of comments present in a target variable 'comment_text'.
- As the training and test set has large number of rows which is not easy for the system to process, Hence both train & test size reduced to 6000-7000 rows in order to analyse, process and build the model.
- Stopwords, Stemming and TFID Vecotorizer is appled to target variables in order to process and clean target variables before model building..
- Hamming Loss and Log Loss is also used as the problem is multilabel multiclass problem where it might be difficult to trust only on accuracy , F1 or AUC-ROC Score metrics to select the best model..

- Selection of hyperparameter is done through halving-gridsearchCV along with K-Fold CV to select best param.

- Limitations of this work and Scope for Future Work

  What are the limitations of this solution provided, the future scope? What all steps/techniques can be followed to further extend this study and improve the results.

  1. **Out-of-Vocublary words:** There is possibility occurrence of words present in train dataset might not be present in test dataset.
  2. **Long range Dependencies:** It might be difficult to find the toxicity for longer comments .
  3. **Multiword Phrases:** There is possibility that comment may have multiword's phrases where algorithm will be able to detect their toxicity only if they recognizes as a single toxic phrase.