FLIP ROBO

PROJECT REPORT ON :

-"MALIGNANT COMMENT CLASSFIER"

SUBMITTED BY / AUTHOR : -

AMAN KUMAR PATEL

SME :

- MD. KASHIF

BATCH: -

INTERNSHIP 25

ACKNOWLADGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with this project and also to make Defaulter model, it has helped me a lot to get improvisation my analyzation skills and scrapping skills also . And specially I want to express my huge gratitude to Mr. mhd. kashif (SME Flip Robo), he is the person that who has helped me to get out of all the difficulties I faced while doing the project. And also I would like to thanks for unconditional support I have ended up with a beautiful Project. A very 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

Business Problem Framing

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.

There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users.

Conceptual Background of the Domain Problem

In the past few years it's seen that the cases related to social media hatred have increased exponentially. Social media is turning into a dark venomous pit for people nowadays. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc. In social media the people spreading or involved in such kind of activities use filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.

Review of Literature

Sentiment classification regarding toxicity has been intensively researched in the past few years, largely in the context of social media data where researchers have applied various machine learning systems to try and tackle the problem of toxicity as well as the related, more well-known task of sentiment analysis. Comment abuse classification research begins with combining of TF-IDF with sentiment/contextual features. The motivation for our project is to build a model that can detect toxic comments and find the bias with respect to the mention of select identities.

Motivation for the Problem Undertaken

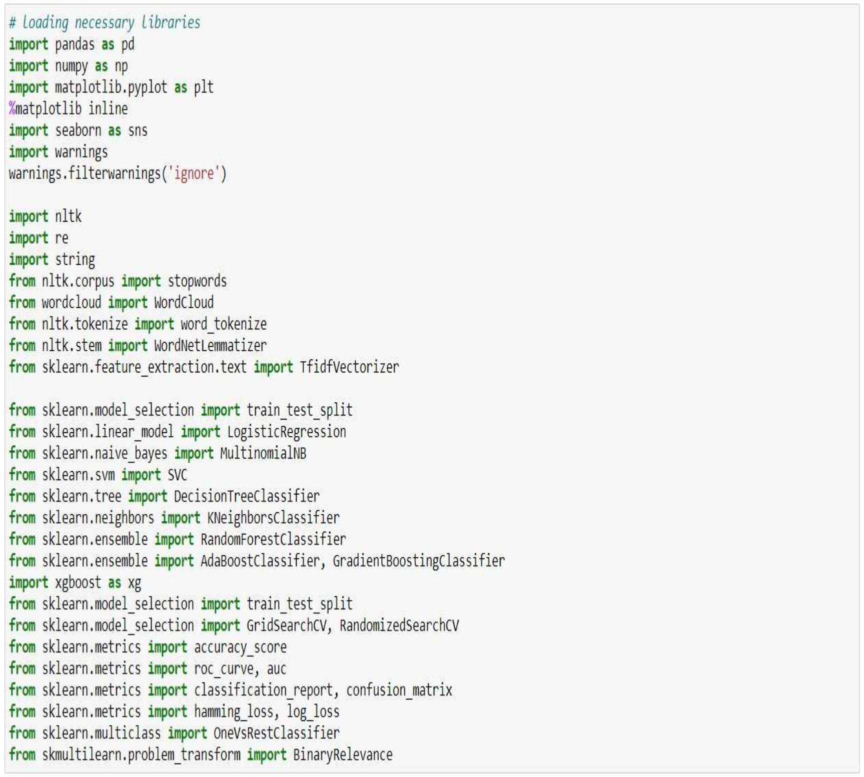
The goal is to build a prototype of online hate and abuse comment classifier which can be used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

- ➔Explore the dataset to get a better picture of how the labels are distributed, how they correlate with each other, and what defines toxic or clean comments.
- ➔To apply data preprocessing and preparation techniques in order to obtain clean data.
- ➔Explore the effectiveness of multiple machine learning approaches and select the best for this problem.

ANALYTICAL PROBLEM FRAMING

The code in this project is written in Python 3.6.6-Anaconda3. To begin the project, importing essential libraries are extremely initial and the very base step as it provides fast, expressive, and flexible data structures to easily work with.

The libraries includes :



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

import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
import xgboost as xg
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import hamming_loss, log_loss
from sklearn.multiclass import OneVsRestClassifier
from skmultilearn.problem_transform import BinaryRelevance
```

The Dataset

The data set contains the training set, which has approximately 1, 59,000 samples and the test set which contains nearly 1, 53,000 samples.

All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels.

The data set includes:

1. **ID** : It includes unique Ids associated with each comment text given.
2. **Comment text** : This column contains the comments extracted from various social media platforms.
3. **Malignant** : It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
4. **Highly Malignant** : It denotes comments that are highly malignant and hurtful.
5. **Rude** : It denotes comments that are very rude and offensive.
6. **Threat** : It contains indications of the comments that are giving any threat to someone.
7. **Abuse** : It is for comments that are abusive in nature.
8. **Loathe** : It describes the comments which are hateful and loathing in nature.

Both train and test csv(s) are loaded respectively, where, in training dataset, the **independent variable is Comment text** which is of ‘object’ type and rest **6 categories or labels are the dependent features** whose values needs to be predicted, are of boolean in nature being ‘int64’ type.

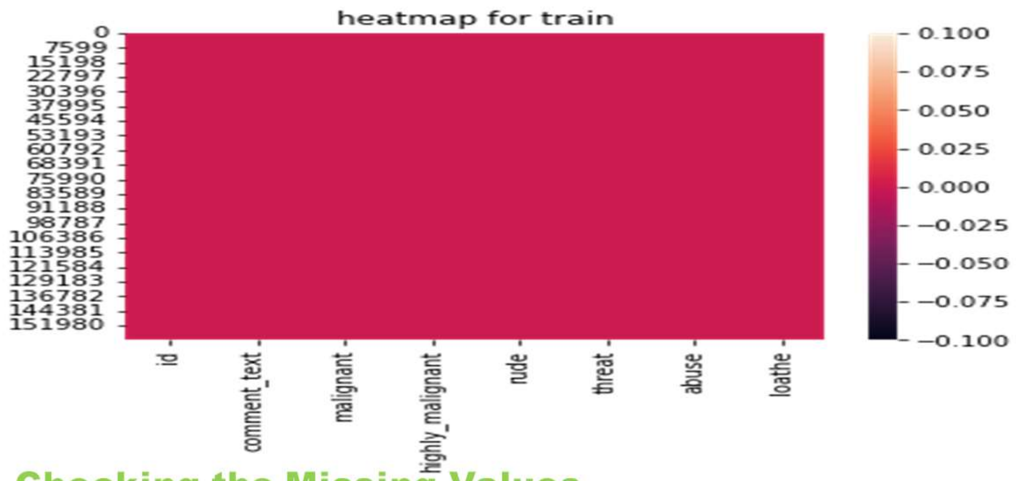
Dropping the Column

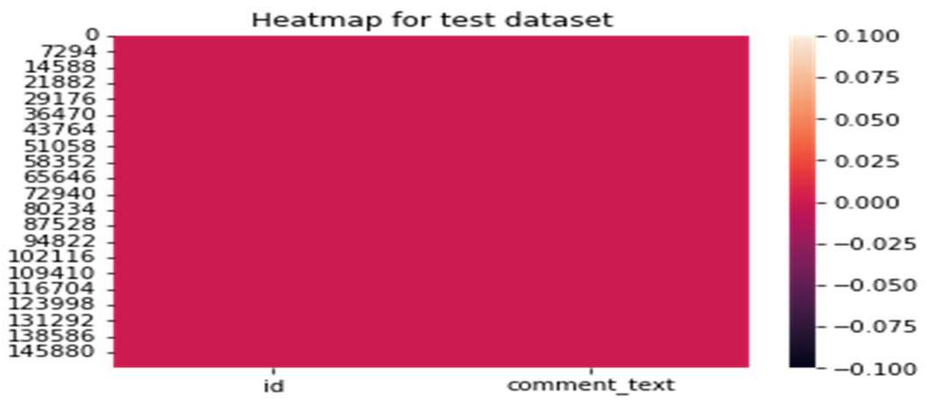
Due to the wide range of given data, it is extremely fruitful to clean, shape and set the data in the most suitable form. Dropping unnecessary columns declines the chances of producing errors.

Thus, column 'ID' was dropped from the train dataset as every comment has its own unique id. After dropping the same, the dataset is now having 7 attributes in total including the target variables.

Checking for the Null values

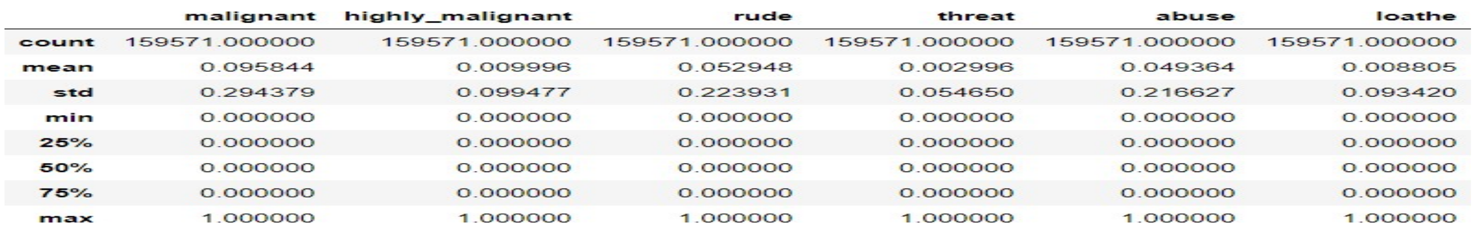
We noticed that we don't have any null values also the null values are checked using isnull().sum(), which results in getting no features having null values in actual. The same has been visualized using heatmap.





Checking the Missing Values

Moving ahead, cleaning the dataset will remove errors which in turn will increase productivity and render highest quality information in decision making.



	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

The statistical summary of the train dataset depicts :

- The minimum value and the maximum value of the attributes is the same i.e. 0 and 1 respectively.
- The mean and standard deviation is nearly 0-1 of all the attributes in the training dataset.
- Here, with this statistical analysis, it is interpreted that there are no outliers as well as skewness present in this training dataset.
- The count of each field is equal which shows that there are no missing values present.

Finding Relations Among the Analysis

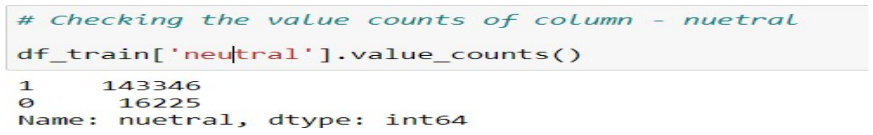
The most commonly used techniques for investigating the relationship between two quantitative variables is correlation. The correlation coefficient has values between -1 to 1.

The correlation is checked and thus visualized using a heat-map which determines the relationship and statistics more distinctly among the variables.

Creating New Feature

A new feature is created in the train dataset in order to explore the data more accurately. A new column/label named - 'neutral' is created which depicts the comments having no toxicity i.e. the comments are not in any category present in the dataset.

The neutral column represents those comments which do not contain or involve any kind of toxicity and as per dataset, which is not included in any of the 6 categories of toxicity of comments present in the dataset.



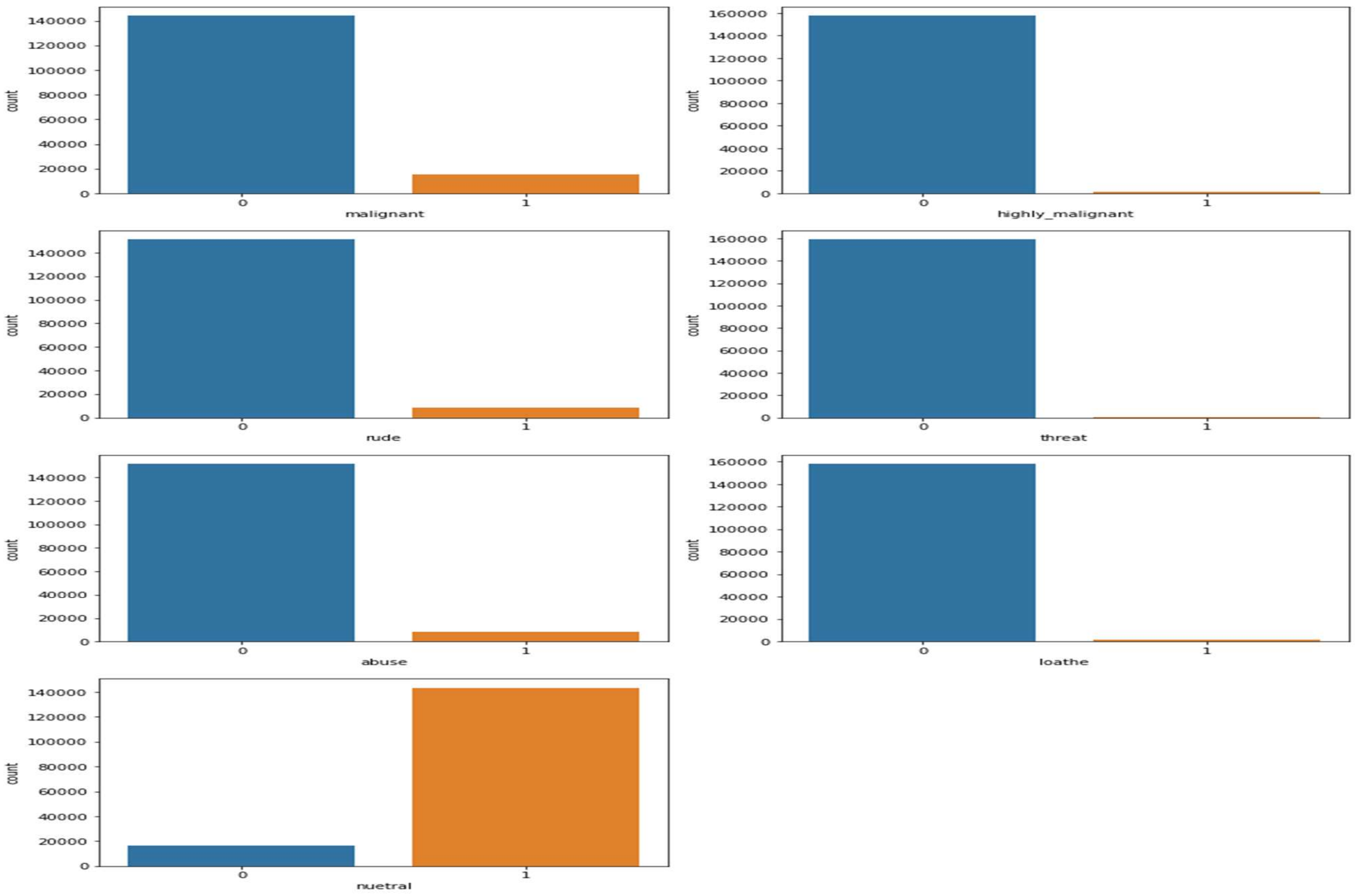
```
# Checking the value counts of column - nuetral
df_train['neutral'].value_counts()
```

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

Analyzing the Data Distinctly through Visualization

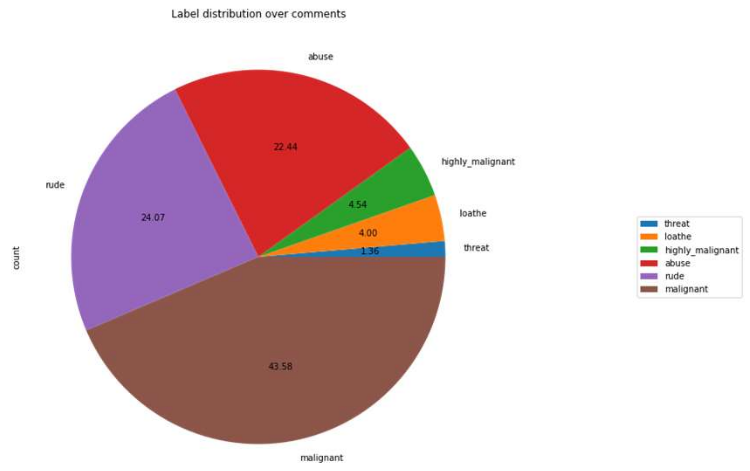
The use of tables, graphs, and charts play a vital role in presenting the data being used to draw these conclusions. Thus, data visualization is the best way to explore the data as it allows in-depth analysis.

➔For better understanding of the dataset, The categories of comments were separated in a list and checked the values counts of each label using value_counts() and the same has been visualized using the **Count Plot**.

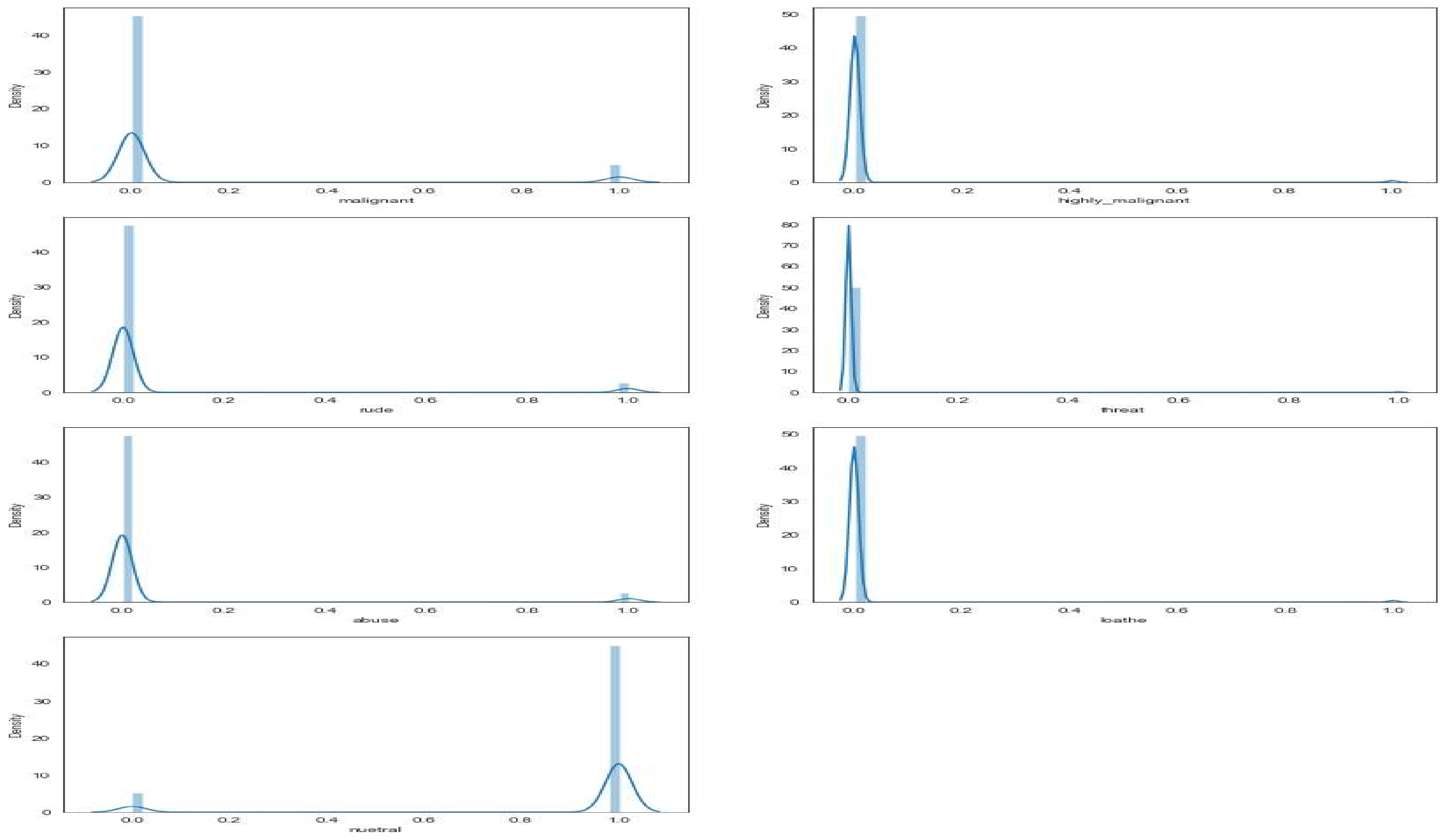

count
malignant
highly_malignant
rude
threat
abuse
loathe
nuetral
0
1

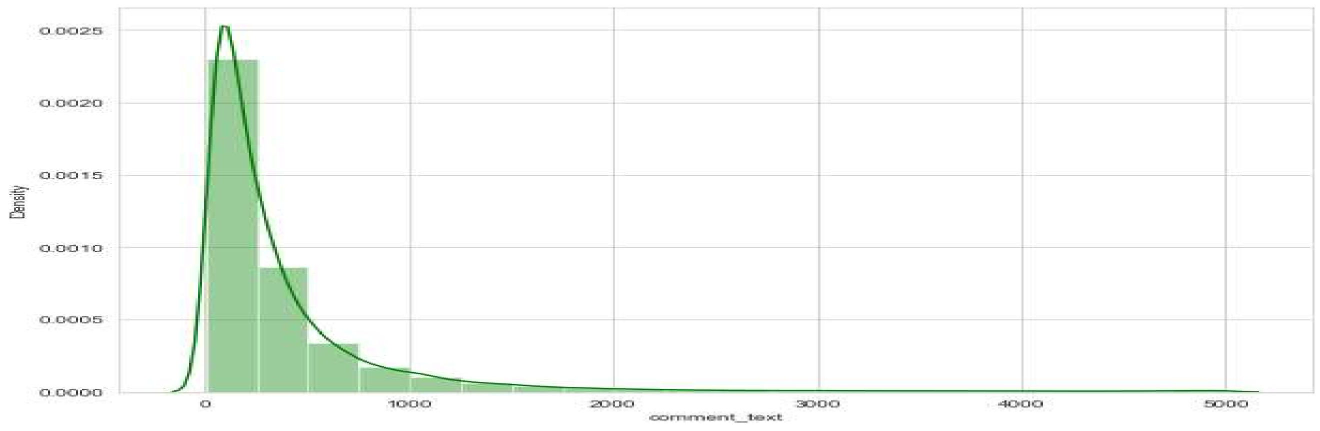
➔The categories of comments are visualized using **Pie Charts**, with original labels and with the new feature added to the train dataset.

Pie chart representing the percentage of each label originally included in the dataset.

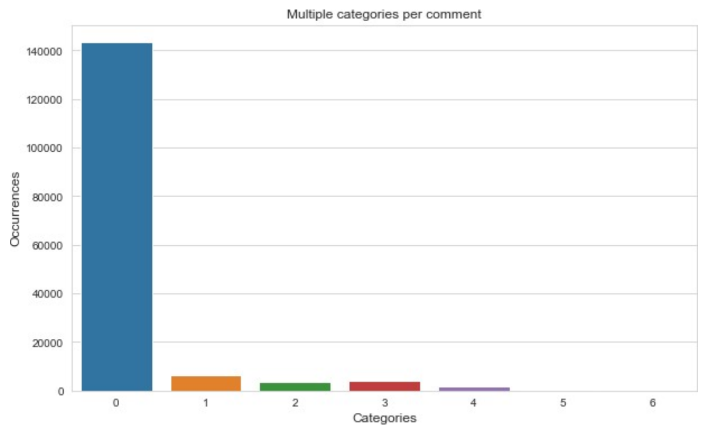


- In the above chart, it is clear that 43.58% of comments belong to the malignant category.
- There is just around 2% difference between ‘rude’ and ‘abuse’ comments being 24.07% and 22.44% of the total comments.
- Least are the ‘threat’ comments.

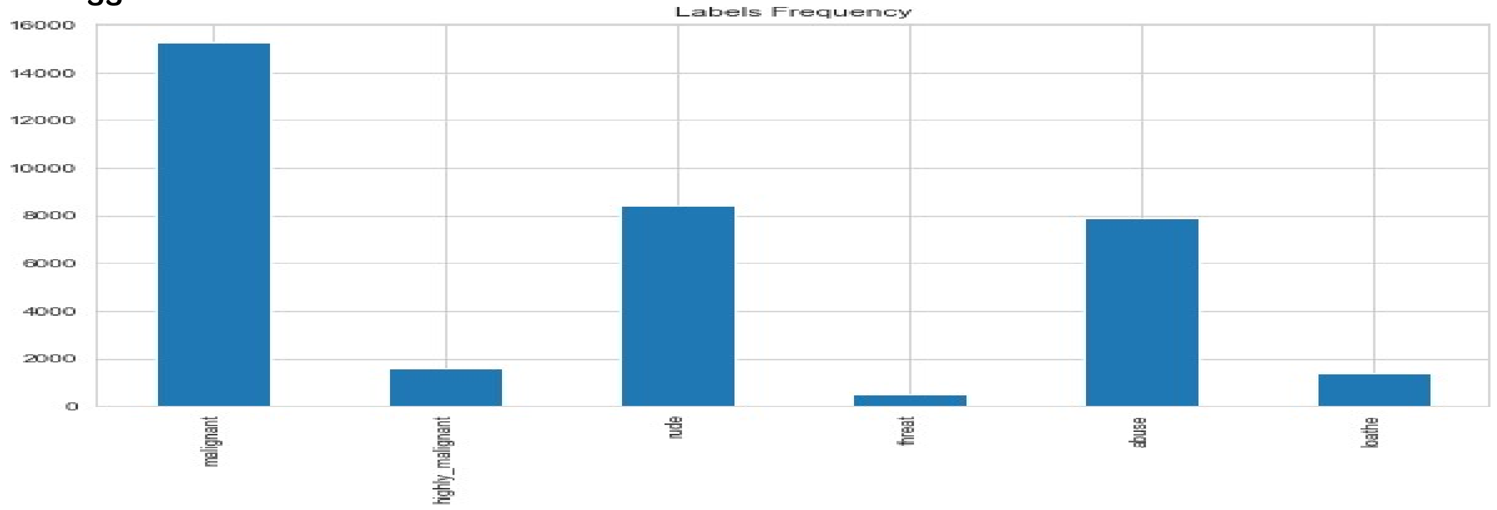

Density
malignant
highly_malignant
rude
threat
abuse
loathe
nuetral
0.0
0.2
0.4
0.6
0.8
1.0



- Above, is a plot showing the comment length frequency.
- As noticed, most of the comments are short with only a few comments longer than 1000 words.
- Majority of the comments are of length 500, where maximum length is 5000 and minimum length is 5.
- Median length being 250.

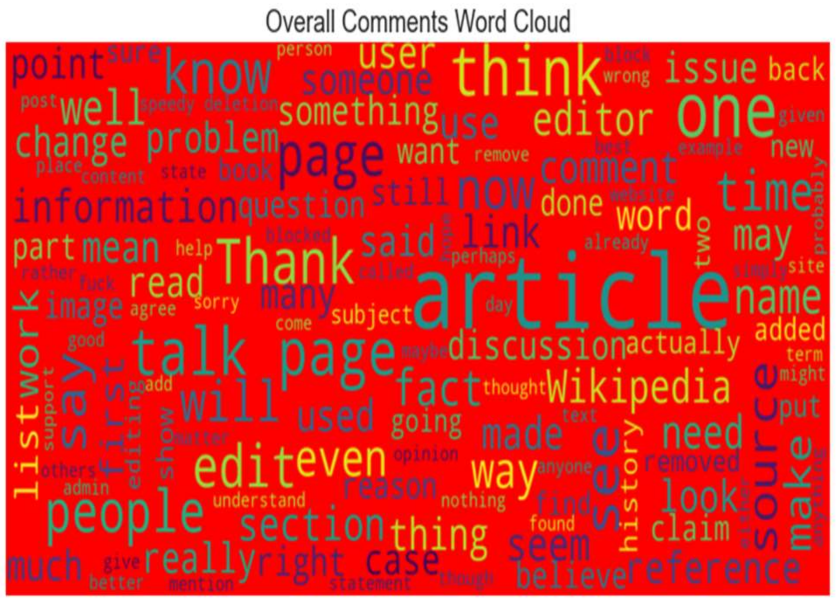


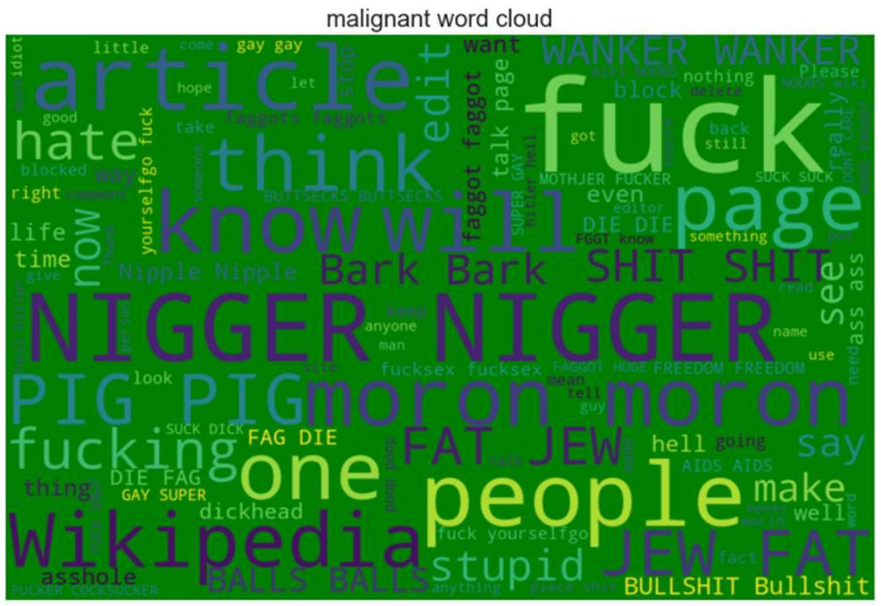
Vast majority of the comment text are not labeled which is around 89%. This essentially shows that a large amount of the entire dataset is tagged to none of the six labels.



From the above plot, it is observed :

- **"malignant" comment has the highest frequency of occurrence followed by "rude" and "abuse", respectively.**
- **On the other hand, a few comments belong to the 'threat' category.**
- **With the count of nearly 8000 the comments are categorised as 'rude' and 'abuse'.**





Here is the word cloud of 'comment_text'showing the frequency of words in the comments present in the dataset.

Cleaning the Data

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. This data is usually not necessary or helpful when it comes to analyzing data because it may hinder the process or provide inaccurate results.

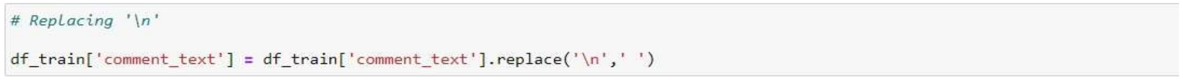
Before cleaning the data a new column is created named ‘length_before_cleaning’ which shows the total length of the comments respectively before cleaning the text.

The following steps were taken in order to clean the text:

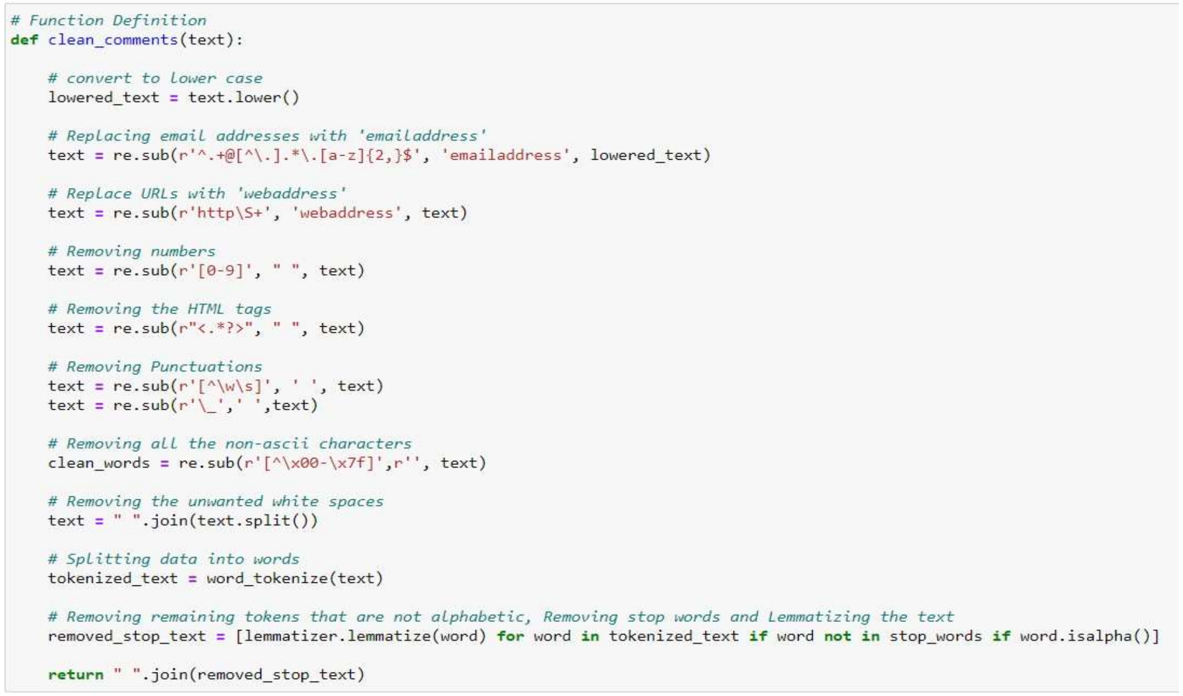
- Replaced the extra lines or ‘\n’ from the text.
- Transform the text into lower case.
- Replaced the email addresses with the text 'emailaddress'
- Replaced the URLs with the text 'webaddress'
- Removed the numbers
- Removed the HTML tags
- Removed the punctuations
- Removed all the non-ascii characters
- Removed the unwanted white spaces
- Removed the remaining tokens that are not alphabetic
- Removed the stop words

All the text cleaning or the above steps are performed by **defining a function and applying the same using apply() to the comment_text column of the train dataset.**

Below is the code shown :



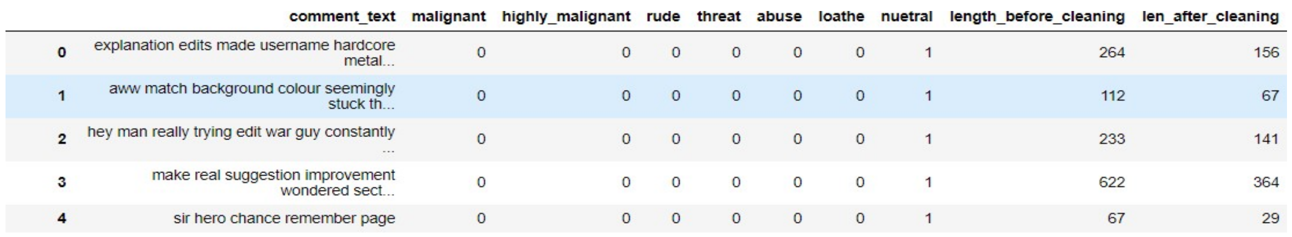
```
# Replacing '\n'

df_train['comment_text'] = df_train['comment_text'].replace('\n',' ')
```



```
# Function Definition
def clean_comments(text):

    # convert to lower case
    lowered_text = text.lower()

    # Replacing email addresses with 'emailaddress'
    text = re.sub(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress', lowered_text)

    # Replace URLs with 'webaddress'
    text = re.sub(r'http\S+', 'webaddress', text)

    # Removing numbers
    text = re.sub(r'[0-9]', " ", text)

    # Removing the HTML tags
    text = re.sub(r"<.*?>", " ", text)

    # Removing Punctuations
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'\_',' ',text)

    # Removing all the non-ascii characters
    clean_words = re.sub(r'[^\x00-\x7f]',r'', text)

    # Removing the unwanted white spaces
    text = " ".join(text.split())

    # Splitting data into words
    tokenized_text = word_tokenize(text)

    # Removing remaining tokens that are not alphabetic, Removing stop words and Lemmatizing the text
    removed_stop_text = [lemmatizer.lemmatize(word) for word in tokenized_text if word not in stop_words if word.isalpha()]

    return " ".join(removed_stop_text)
```

After cleaning the text, a new column is created named 'len_after_cleaning' representing the length of each comment respectively in a column 'comment_text' after doing the required cleaning of the text. With this, it's come to know about how much data is cleaned.

Similar step was used on test data. The dataset was processed using the function created



	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	nuetral	length_before_cleaning	len_after_cleaning
0	explanation edits made username hardcore metal...	0	0	0	0	0	0	1	264	156
1	aww match background colour seemingly stuck th...	0	0	0	0	0	0	1	112	67
2	hey man really trying edit war guy constantly ...	0	0	0	0	0	0	1	233	141
3	make real suggestion improvement wondered sect...	0	0	0	0	0	0	1	622	364
4	sir hero chance remember page	0	0	0	0	0	0	1	67	29

MODEL DEVELOPMENT AND EVALUATION

Separating the input and output variables

In order to get a more accurate list of variables, the input and output variables are separated. Thus, the data is separated in train and test data.

'X' represents the input variable, which is the 'comment_text' carrying the cleaned text.

'y' represents the output variable which is the target variable. As this problem is a multi-label classification problem. The target variables are the categories of comments i.e. there are 6 target variables - malignant, highly_malignant, rude, threat, abuse and loathe.

'test_X' represents the test data for which the predicted values will be stored.

On checking the shape of above defined variables, it is determined that 'X' contains 159571 rows and a column whereas 'y' contains 159571 rows and 6 columns. The 'test_X' contains 153164 rows.

Splitting the train and test data

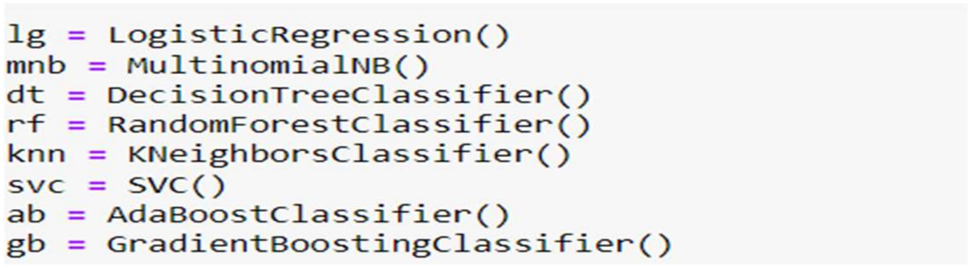
Training and Testing the models minimizes the effects of data discrepancies and better understand the characteristics of the model.

The **training** data is used to make the machine recognize patterns in the data and the **test** data is used to see how well the machine can predict new answers based on its training.

'X' and 'y' were split for training and testing using train_test_split in a ratio of 80:20 respectively

Algorithms

Using various algorithms, the models were created and in different ways

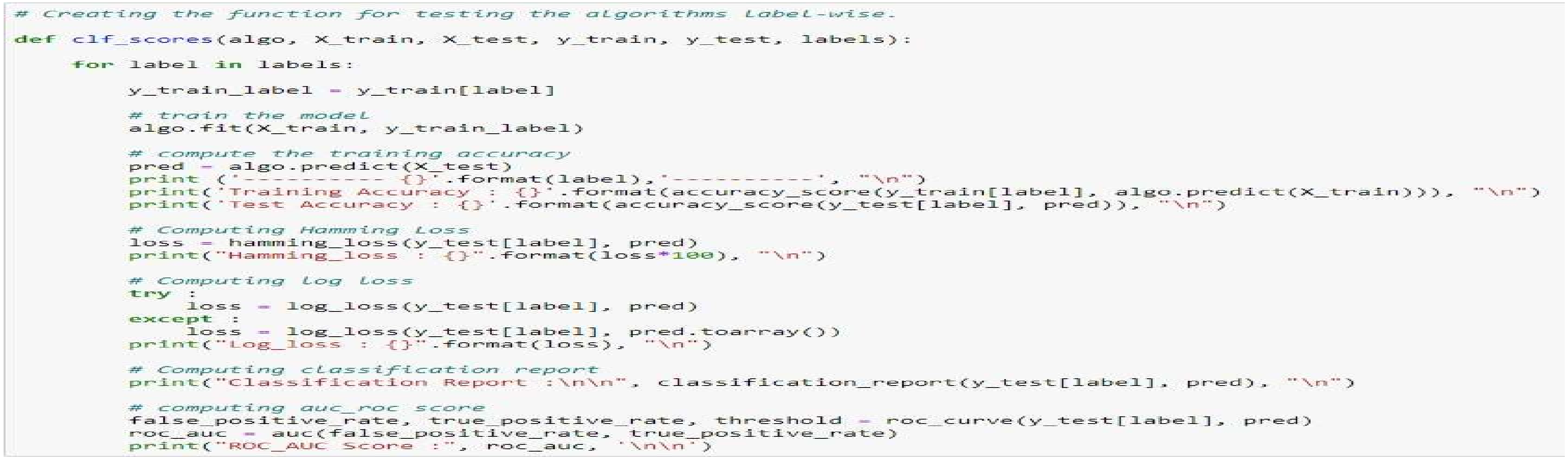


```
lg = LogisticRegression()
mnb = MultinomialNB()
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
knn = KNeighborsClassifier()
svc = SVC()
ab = AdaBoostClassifier()
gb = GradientBoostingClassifier()
```

Testing of Identified Algorithms

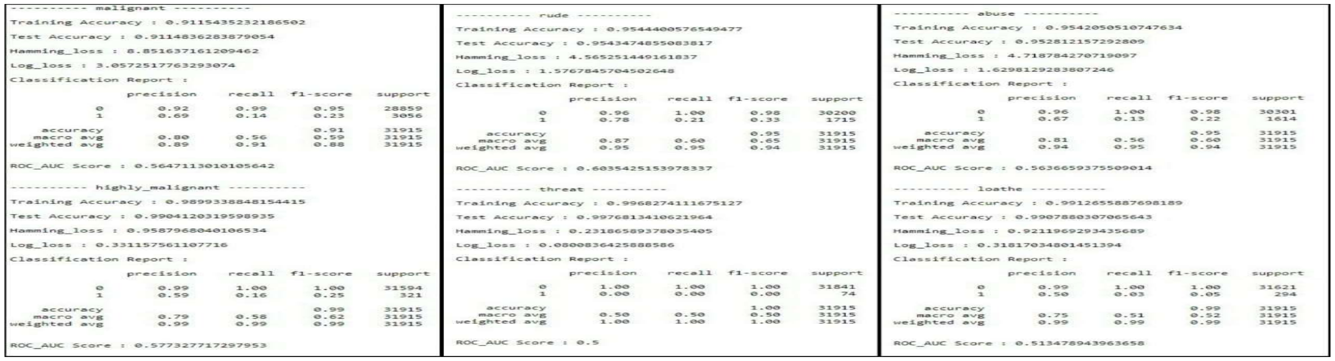
The Models are trained and tested in various ways:

1) Firstly, the algorithms are trained and tested label-wise, for which a function is defined, which results in all the metrics used in order to find the best model, for each label.



```
# Creating the function for testing the algorithms Label-wise.

def clf_scores(algo, X_train, X_test, y_train, y_test, labels):

    for label in labels:

        y_train_label = y_train[label]

        # train the model
        algo.fit(X_train, y_train_label)

        # compute the training accuracy
        pred = algo.predict(X_test)
        print ('----------- {}'.format(label),'-----------', "\n")
        print('Training Accuracy : {}'.format(accuracy_score(y_train[label], algo.predict(X_train))), "\n")
        print('Test Accuracy : {}'.format(accuracy_score(y_test[label], pred)), "\n")

        # Computing Hamming Loss
        loss = hamming_loss(y_test[label], pred)
        print("Hamming_loss : {}".format(loss*100), "\n")

        # Computing Log Loss
        try :
            loss = log_loss(y_test[label], pred)
        except :
            loss = log_loss(y_test[label], pred.toarray())
        print("Log_loss : {}".format(loss), "\n")

        # Computing classification report
        print("Classification Report :\n\n", classification_report(y_test[label], pred), "\n")

        # computing auc_roc score
        false_positive_rate, true_positive_rate, threshold = roc_curve(y_test[label], pred)
        roc_auc = auc(false_positive_rate, true_positive_rate)
        print("ROC_AUC Score :", roc_auc, '\n\n')
```

Here as an example, the results of each label using logistic Regression is shown below :

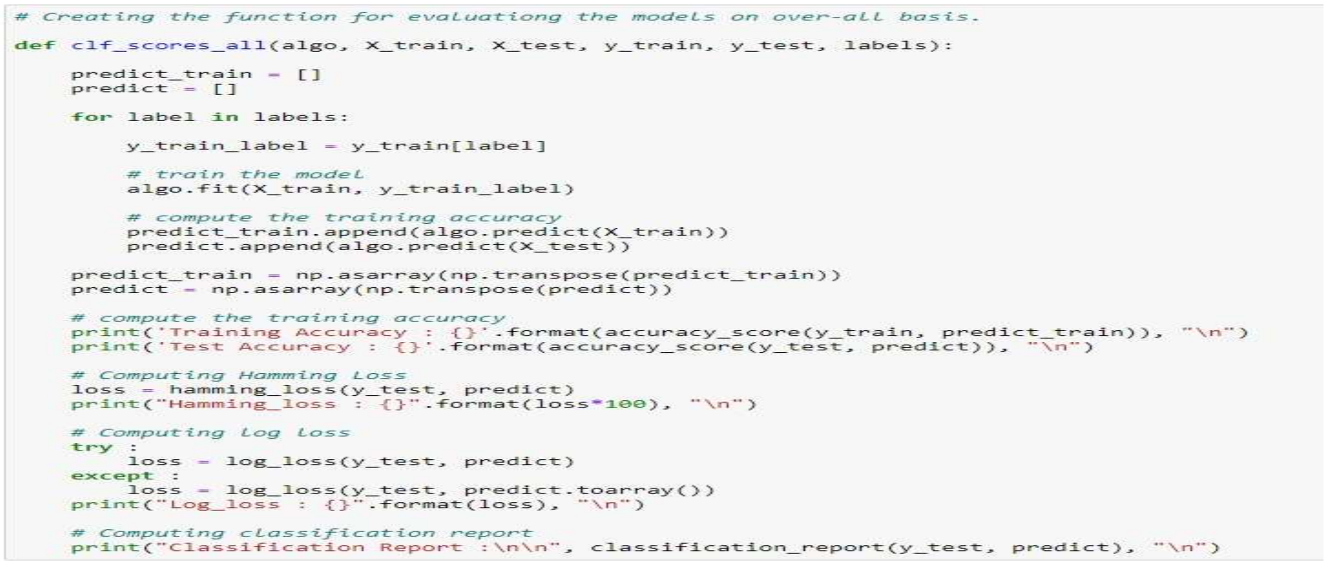


```
----------- malignant -----------
Training Accuracy : 0.9115435232186502
Test Accuracy : 0.9114836283879054
Hamming_loss : 8.851637161209462
Log_loss : 3.0572517763293074
Classification Report :
              precision    recall  f1-score   support

           0       0.92      0.99      0.95     28859
           1       0.69      0.14      0.23      3056

    accuracy                           0.91     31915
   macro avg       0.80      0.56      0.59     31915
weighted avg       0.89      0.91      0.88     31915

ROC_AUC Score : 0.5647113010105642

----------- highly_malignant -----------
Training Accuracy : 0.9899338848154415
Test Accuracy : 0.9904120319598935
Hamming_loss : 0.9587968040106534
Log_loss : 0.331157561107716
Classification Report :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     31594
           1       0.59      0.16      0.25       321

    accuracy                           0.99     31915
   macro avg       0.79      0.58      0.62     31915
weighted avg       0.99      0.99      0.99     31915

ROC_AUC Score : 0.577327717297953
```

```
----------- rude -----------
Training Accuracy : 0.9544400576549477
Test Accuracy : 0.9543474855083817
Hamming_loss : 4.565251449161837
Log_loss : 1.5767845704502648
Classification Report :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     30200
           1       0.78      0.21      0.33      1715

    accuracy                           0.95     31915
   macro avg       0.87      0.60      0.65     31915
weighted avg       0.95      0.95      0.94     31915

ROC_AUC Score : 0.6035425153978337

----------- threat -----------
Training Accuracy : 0.9968274111675127
Test Accuracy : 0.9976813410621964
Hamming_loss : 0.23186589378035405
Log_loss : 0.0800836425888586
Classification Report :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     31841
           1       0.00      0.00      0.00        74

    accuracy                           1.00     31915
   macro avg       0.50      0.50      0.50     31915
weighted avg       1.00      1.00      1.00     31915

ROC_AUC Score : 0.5
```

```
----------- abuse -----------
Training Accuracy : 0.9542050510747634
Test Accuracy : 0.952812157292809
Hamming_loss : 4.718784270719097
Log_loss : 1.6298129283807246
Classification Report :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     30301
           1       0.67      0.13      0.22      1614

    accuracy                           0.95     31915
   macro avg       0.81      0.56      0.60     31915
weighted avg       0.94      0.95      0.94     31915

ROC_AUC Score : 0.5636659375509014

----------- loathe -----------
Training Accuracy : 0.9912655887698189
Test Accuracy : 0.9907880307065643
Hamming_loss : 0.9211969293435689
Log_loss : 0.31817034801451394
Classification Report :
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     31621
           1       0.50      0.03      0.05       294

    accuracy                           0.99     31915
   macro avg       0.75      0.51      0.52     31915
weighted avg       0.99      0.99      0.99     31915

ROC_AUC Score : 0.513478943963658
```

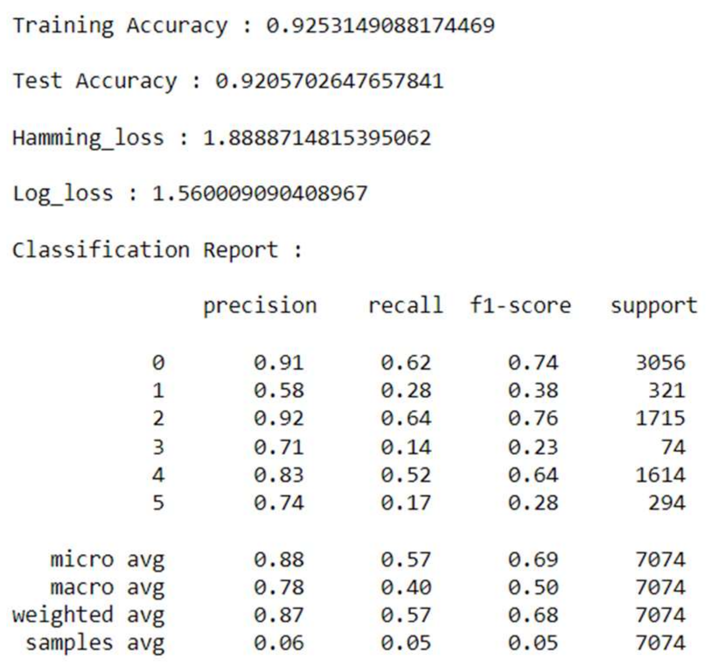
2) Then, the models were trained and tested on an overall basis in order to have an overall results of the algorithms used. To perform such a thing, a function is created which is then passed through the models. This function also involves each matrix to be evaluated in order to select the best performing model.



```
# Creating the function for evaluationg the models on over-all basis.
def clf_scores_all(algo, X_train, X_test, y_train, y_test, labels):

    predict_train = []
    predict = []

    for label in labels:

        y_train_label = y_train[label]

        # train the model
        algo.fit(X_train, y_train_label)

        # compute the training accuracy
        predict_train.append(algo.predict(X_train))
        predict.append(algo.predict(X_test))

    predict_train = np.asarray(np.transpose(predict_train))
    predict = np.asarray(np.transpose(predict))

    # compute the training accuracy
    print('Training Accuracy : {}'.format(accuracy_score(y_train, predict_train)), "\n")
    print('Test Accuracy : {}'.format(accuracy_score(y_test, predict)), "\n")

    # Computing Hamming Loss
    loss = hamming_loss(y_test, predict)
    print("Hamming_loss : {}".format(loss*100), "\n")

    # Computing Log Loss
    try :
        loss = log_loss(y_test, predict)
    except :
        loss = log_loss(y_test, predict.toarray())
    print("Log_loss : {}".format(loss), "\n")

    # Computing classification report
    print("Classification Report :\n\n", classification_report(y_test, predict), "\n")
```

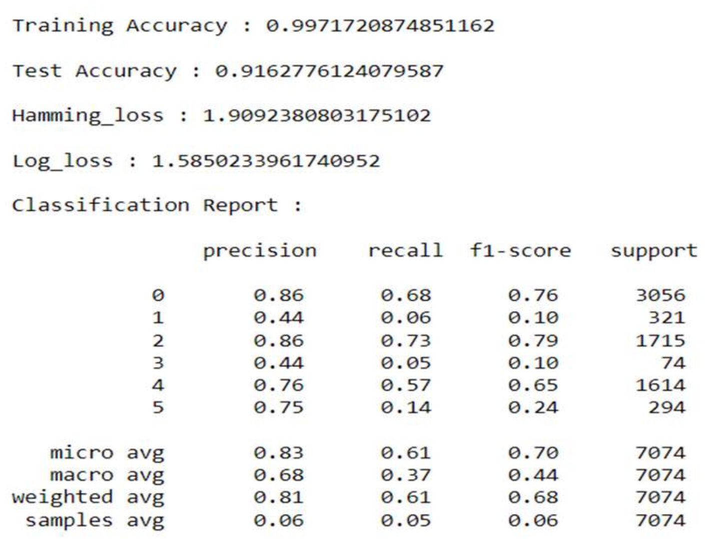
This method seems better than the before one as this method or logic used to get the results of the models shows the overall performance of the model as well as individual scores of each label which are represented as 0-5.

Logistic Regression:-



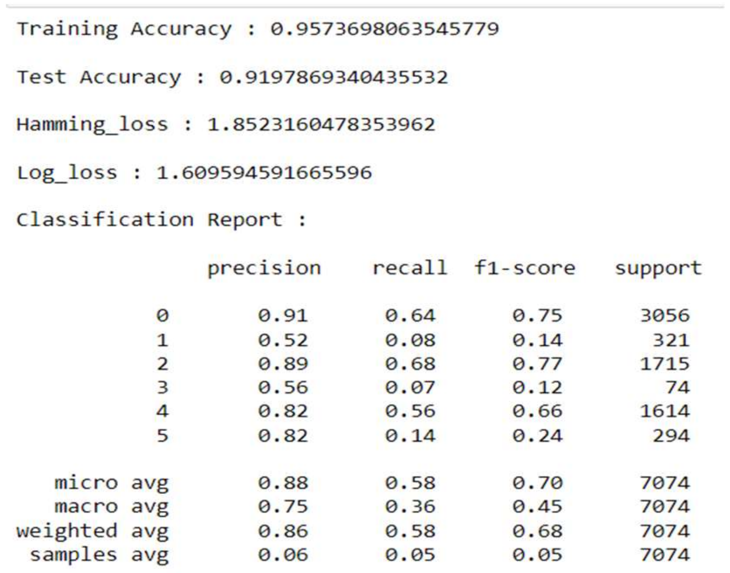
```
Training Accuracy : 0.9253149088174469

Test Accuracy : 0.9205702647657841

Hamming_loss : 1.8888714815395062

Log_loss : 1.560009090408967

Classification Report :

              precision    recall  f1-score   support

           0       0.91      0.62      0.74      3056
           1       0.58      0.28      0.38       321
           2       0.92      0.64      0.76      1715
           3       0.71      0.14      0.23        74
           4       0.83      0.52      0.64      1614
           5       0.74      0.17      0.28       294

   micro avg       0.88      0.57      0.69      7074
   macro avg       0.78      0.40      0.50      7074
weighted avg       0.87      0.57      0.68      7074
 samples avg       0.06      0.05      0.05      7074
```

Random Forest Classifier:-



```
Training Accuracy : 0.9971720874851162

Test Accuracy : 0.9162776124079587

Hamming_loss : 1.9092380803175102

Log_loss : 1.5850233961740952

Classification Report :

              precision    recall  f1-score   support

           0       0.86      0.68      0.76      3056
           1       0.44      0.06      0.10       321
           2       0.86      0.73      0.79      1715
           3       0.44      0.05      0.10        74
           4       0.76      0.57      0.65      1614
           5       0.75      0.14      0.24       294

   micro avg       0.83      0.61      0.70      7074
   macro avg       0.68      0.37      0.44      7074
weighted avg       0.81      0.61      0.68      7074
 samples avg       0.06      0.05      0.06      7074
```

SVC:-

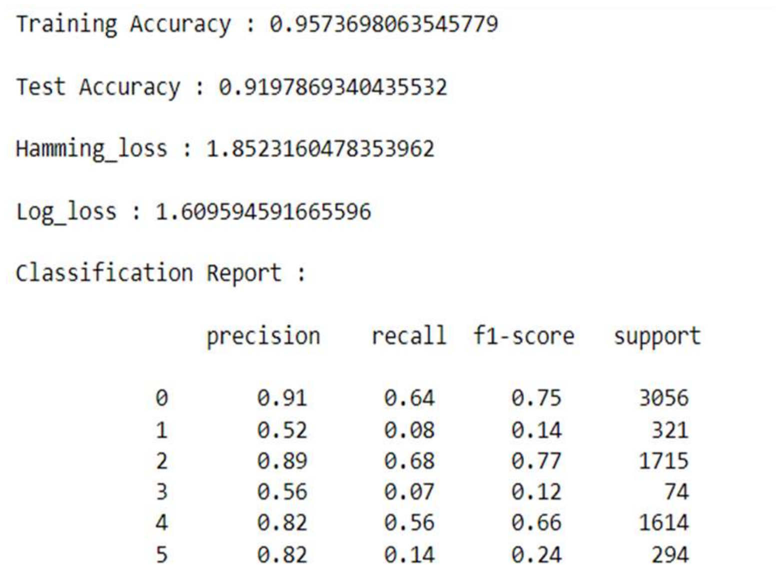


```
Training Accuracy : 0.9573698063545779

Test Accuracy : 0.9197869340435532

Hamming_loss : 1.8523160478353962

Log_loss : 1.609594591665596

Classification Report :

              precision    recall  f1-score   support

           0       0.91      0.64      0.75      3056
           1       0.52      0.08      0.14       321
           2       0.89      0.68      0.77      1715
           3       0.56      0.07      0.12        74
           4       0.82      0.56      0.66      1614
           5       0.82      0.14      0.24       294

   micro avg       0.88      0.58      0.70      7074
   macro avg       0.75      0.36      0.45      7074
weighted avg       0.86      0.58      0.68      7074
 samples avg       0.06      0.05      0.05      7074
```

The methods used are:

a)Binary relevance
b)OneVsRestClassifier

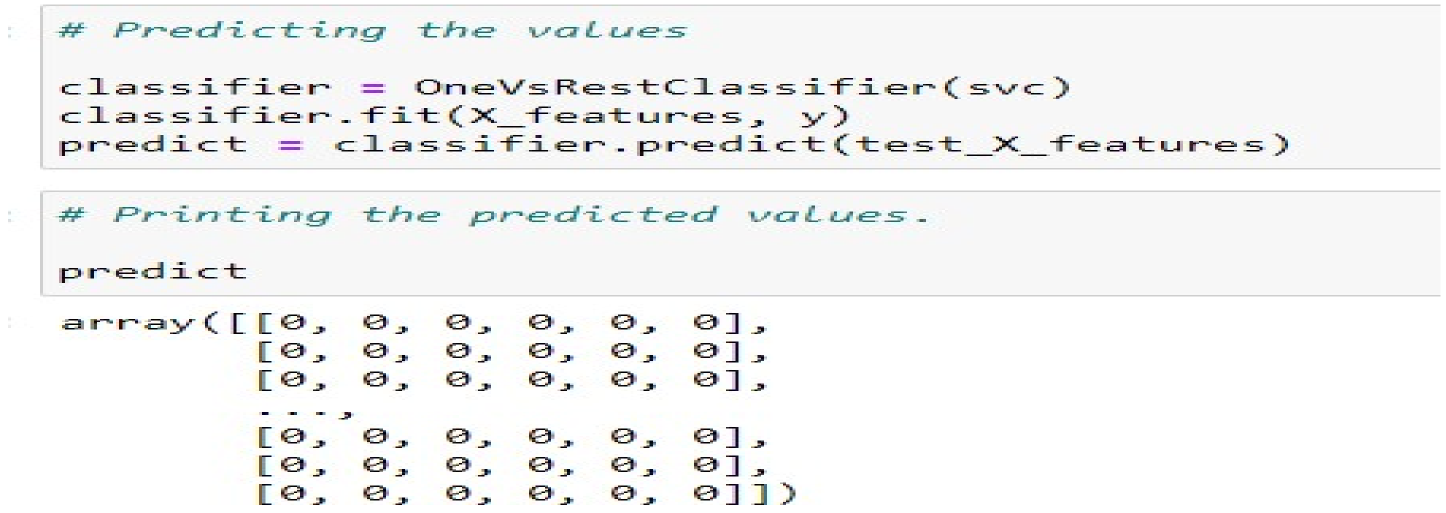
categories, **SVC** is selected as a best model. The SVC using oneVsRestClassifier is giving test accuracy of 95% and thus, selected as a best model for this data.



```
Training Accuracy : 0.9573698063545779

Test Accuracy : 0.9197869340435532

Hamming_loss : 1.8523160478353962

Log_loss : 1.609594591665596

Classification Report :

              precision    recall  f1-score   support

           0       0.91      0.64      0.75      3056
           1       0.52      0.08      0.14       321
           2       0.89      0.68      0.77      1715
           3       0.56      0.07      0.12        74
           4       0.82      0.56      0.66      1614
           5       0.82      0.14      0.24       294
```

Predicting the values using the selected model - SVC using oneVsRestClassifier

As the SVC is performing the best out of all the models tested the values are predicted of all the target variables or labels, which has given a matrix.

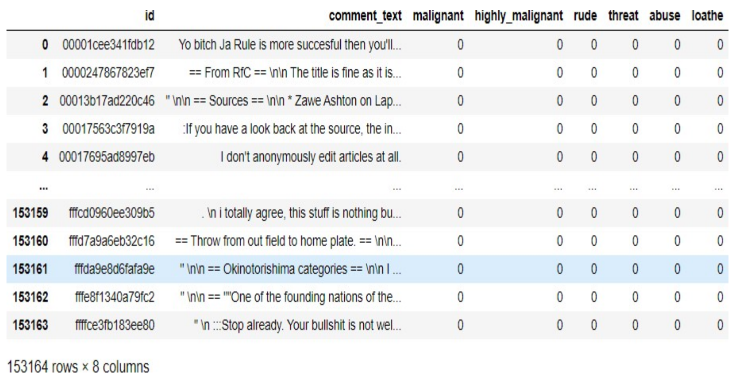


```
# Predicting the values

classifier = OneVsRestClassifier(svc)
classifier.fit(X_features, y)
predict = classifier.predict(test_X_features)
```

```
# Printing the predicted values.

predict
```

```
array([[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]])
```

Then, converted the predicted values into a dataframe and as a final step added these predicted values to the test data and saved the same to a csv file.



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

153164 rows × 8 columns

CONCLUSION

In this report or project, several Classification models are built to predict the labels of the category of toxic comments. The models are evaluated and compared to determine the one with highest performance. The features according to their importance, ranked on the basis of model, are also looked at. In this project, the data science process starting with getting the data, then cleaning and preprocessing the data, followed by exploring the data and building models, then evaluating the results and communicating them with visualizations is followed. Thus, this is a Machine Learning Approach combined with Natural Language Processing (NLP) for toxicity detection and its type identification in user comments.

Finally, the Mean Validation Accuracy, so obtained, is 90.08% which is by far the highest ever numeric accuracy reached by any Comment Toxicity Detection Model. The research done in this project is intended to enhance fair online talk and views sharing in social media.

THANK

FLIP ROBO

YOU