“MALIGNANT-COMMENTS-CLASSIFIER”

Submitted by:

Sarika Thorat

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Khushboo Garg as well as Flip Robo Technologies who gave me the opportunity to do this project on Malignant Comments Classification, which also helped me in doing lots of research wherein I came to know about so many new things, especially the Natural Language Processing and Natural Language Toolkit parts.

Also, I have utilized a few external resources that helped me to complete this project. I ensured that I learn from the samples and modify things according tomy project requirement. All the external resources that were used in creating this project are listed below:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://scikit-learn.org/stable/user_guide.html
4) https://github.com/
5) https://www.kaggle.com/
6) https://medium.com/
7) https://towardsdatascience.com/
8) https://www.analyticsvidhya.com/

INTRODUCTION

Business Problem Framing

Online forums and social media platforms have provided individuals with the means to put forward their thoughts and freely express their opinion on various issues and incidents. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. **These online comments contain explicit language which may hurt the readers.** The threat of abuse and harassment means that many people stop expressing themselves and give up on seeking different opinions.

To protect users from being exposed to offensive language on online forums or social media sites, companies have started flagging comments and blocking users who are found guilty of using unpleasant language. Several Machine Learning models have been developed and deployed to filter out unruly language and protect internet users from becoming victims of online harassment and cyberbullying. 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 behavior.

There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facingbacklash from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading todepression, 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 “uare an idiot” is clearly offensive.

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

Conceptual Background of the Domain Problem

Online platforms and social media become the place where people share the thoughts freely without any partiality and overcoming all the race people sharetheir thoughts and ideas among the crowd.

Social media is a computer-based technology that facilitates the sharing of ideas, thoughts, and information through the building of virtual networks and communities. By design, social media is Internet-based and gives users quick electronic communication of content. Content includes personal information, documents, videos, and photos. Users engage with social media via a computer, tablet, or smartphone via web-based software or applications.

While social media is ubiquitous in America and Europe, Asian countries likeIndia lead the list of social media usage. More than 3.8 billion people use socialmedia.

In this huge online platform or an online community there are some people orsome motivated mob willfully bully others to make them not to share their thought in rightful way.

They bully others in a foul language which among the civilized society is seen asignominy. And when innocent individuals arebeing bullied by these mob these individuals are going silent without speaking anything. So, ideally the motive of this disgraceful mob is achieved.

To solve this problem, we are now building a model that identifies all the foullanguage and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community oreven block them or block them from using this foul language.

Review of Literature

The purpose of the literature review is to:

1. Identify the foul words or foul statements that are being used.

2. Stop the people from using these foul languages in online public forum.

To solve this problem, we are now building a model using our machine language technique that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

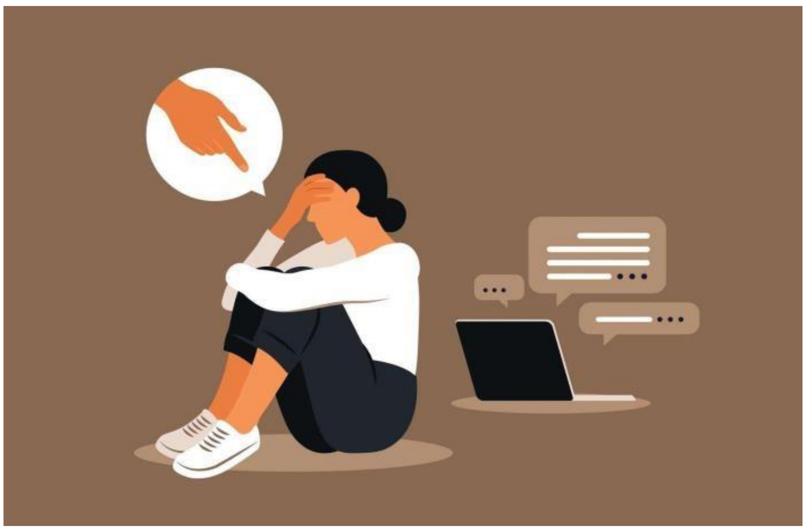
I have used 9 different Classification algorithms and shortlisted the best on basis of the metrics of performance and I have chosen one algorithm and builda model in that algorithm.

Internet comments are bastions of hatred and vitriol. While online anonymityhas 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.

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.

Motivation for the Problem Undertaken

One of the first lessons we learn as children is that the louder you scream andthe bigger of a tantrum you throw, you more you get your way. Part of growingup and maturing into an adult and functioning member of society is learning how to use language and reasoning skills to communicate our beliefs and respectfully disagree with others, using evidence and persuasiveness to try andbring them over to our way of thinking.



Social media is reverting us back to those animalistic tantrums, schoolyard taunts and unfettered bullying that define youth, creating a dystopia where even renowned academics and dispassionate journalists transform from Dr. Jekyll into raving Mr. Hydes, raising the critical question of whether social media should simply enact a blanket ban on profanity and name calling? Actually, ban should be implemented on these profanities and taking that as a motivationI have started this project to identify the malignant comments in social media or in online public forums.

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

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

The libraries/dependencies imported for this project are shown below:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re

import nltk
nltk.download('stopwords', quiet=True)
nltk.download('punkt', quiet=True)

from wordcloud import WordCloud
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from nltk.tokenize import word_tokenize, regexp_tokenize

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, RandomizedSearchCV
from scipy.sparse import csr_matrix

import timeit, sys
from sklearn import metrics
import tqdm.notebook as tqdm

from sklearn.svm import SVC, LinearSVC
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier, RandomForestClassifier
from sklearn.metrics import hamming_loss, log_loss, accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_curve, auc, roc_auc_score, multilabel_confusion_matrix


import warnings
warnings.filterwarnings('ignore')
```

Here in this project, we have been provided with two datasets namely train andtest CSV files. I will build a machine learning model by using NLP using traindataset. And using this model we will make predictions for our test dataset.

I need to build multiple classification machine learning models. Before modelbuilding will need to perform all data pre-processing steps involving NLP. After trying different classification models with different hyper parameters then will select the best model out of it. Will need to follow the complete life cycle of data science that includes steps like -

- Data Cleaning
- Exploratory Data Analysis
- Data Pre-processing
- Model Building
- Model Evaluation
- Selecting the best model

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

Data Sources and their formats

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'. The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels.

The 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 tosomeone.

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 varioussocial media platforms.

Data Pre-processing Done

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

- Loading the dataset
- Remove null values
- Drop column id
- Convert comment text to lower case and replace '\n' with single space.
- Keep only text data ie. a-z' and remove other data from comment text.
- Remove stop words and punctuations
- Apply Stemming using SnowballStemmer

- ➢ Convert text to vectors using TfidfVectorizer
- ➢ Load saved or serialized model
- ➢ Predict values for multi class label

Loading the dataset:

Here I am loading the training dataset into the variable df_train and test dataset as df_test

```
df_train=df_train = pd.read_csv('Malignant_comment_classifier/train.csv')
df_train
```

	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
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

```
df_test = pd.read_csv('Malignant_comment_classifier/test.csv')
df_test
```

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

153164 rows × 2 columns

Since the data set is huge and includes many categories of comments, we can do good amount of data exploration and derive some interesting features using the comments text column available. We need to build a model on train data that can differentiate between comments and their categories and find the categories of comments in the test dataset using that model.

Identification of possible problem-solving approaches (methods)

I checked through the entire training dataset for any kind of missing values information and all these preprocessing steps were repeated on the testing dataset as well.

```
1  df_train.isna().sum()

id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

```
1  df_test.shape

(153164, 2)

1  df_test.isnull().sum()

id              0
comment_text    0
dtype: int64
```

Using the isna and sum options together we can confirm that there are no missing values in any of the columns present in our training dataset.

Then we went ahead and took a look at the dataset information. Using the info method, we are able to confirm the non-null count details as well as the datatype information. We have a total of 8 columns out of which 2 columns have object datatype while the remaining 6 columns are of integer datatype.

```
df_train.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
```

Then we went ahead and performed multiple data cleaning and data transformation steps. I have added an additional column to store the originallength of our comment_text column.

```
# checking the length of comments and storing it into another column 'original_length'
# copying df_train into another object df
df = df_train.copy()
df['original_length'] = df.comment_text.str.len()
```

```
# checking the first five and last five rows here
df
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67
...	...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0	295
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0	99
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0	81
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0	116
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0	189

159571 rows × 9 columns

Since there was no use of the "id" column I have dropped it and converted allthe text data in our comment text column into lowercase format for easier interpretation.

```python
# as the feature 'id' has no relevance w.r.t. model training I am dropping this column
df.drop(columns=['id'],inplace=True)
```

```python
# converting comment text to lowercase format
df['comment_text'] = df.comment_text.str.lower()
df.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
0	explanation\nwhy the edits made under my usern...	0	0	0	0	0	0	264
1	d'aww! he matches this background colour i'm s...	0	0	0	0	0	0	112
2	hey man, i'm really not trying to edit war. it...	0	0	0	0	0	0	233
3	"\nmore\ni can't make any real suggestions on ...	0	0	0	0	0	0	622
4	you, sir, are my hero. any chance you remember...	0	0	0	0	0	0	67

Text Preprocessing:

In natural language processing, text preprocessing is **the practice of cleaning and preparing text data.** NLTK and re are common Python libraries used to handle many text preprocessing tasks.

Removing and Replacing unwanted characters in the comment_text column

```python
# Replacing '\n' with ' '
df.comment_text = df.comment_text.str.replace('\n',' ')

# Keeping only text with letters a to z, 0 to 9 and words like can't, don't, couldn't etc
df.comment_text = df.comment_text.apply(lambda x: ' '.join(regexp_tokenize(x,"[a-z']+")))

# Removing Stop Words and Punctuations

# Getting the list of stop words of english language as set
stop_words = set(stopwords.words('english'))

# Updating the stop_words set by adding letters from a to z
for ch in range(ord('a'),ord('z')+1):
    stop_words.update(chr(ch))

# Updating stop_words further by adding some custom words
custom_words = ("d'aww","mr","hmm","umm","also","maybe","that's","he's","she's","i'll","he'll","she'll","us",
                "ok","there's","hey","heh","hi","oh","bbq","i'm","i've","nt","can't","could","ur","re","ve",
                "rofl","lol","stfu","lmk","ily","yolo","smh","lmfao","nvm","ikr","ofc","omg","ilu")
stop_words.update(custom_words)

# Checking the new list of stop words
print("New list of custom stop words are as follows:\n\n")
print(stop_words)
```

Removing stop words and Punctuations

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

# Removing punctuations
df.comment_text = df.comment_text.str.replace("[^\w\d\s]","")

# Checking any 10 random rows to see the applied changes
df.sample(10)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
134036	suggestion images interested images agneta fri...	0	0	0	0	0	0	349
130166	talk bangladesh census case anyone wondering t...	0	0	0	0	0	0	292
40834	harassment helenonline neiln warned harassing ...	0	0	0	0	0	0	202
137339	yeah gotten know posts done accidentally witho...	0	0	0	0	0	0	261
146928	ditzynizzy british music tag placed ditzynizzy...	0	0	0	0	0	0	870
54827	section political thought use clarification li...	0	0	0	0	0	0	458
125975	might try reviewing section personal vendettas...	0	0	0	0	0	0	147
8339	merger proposal note article see man would kin...	0	0	0	0	0	0	190
143406	thank information read discussion review block...	0	0	0	0	0	0	118
114440	times monday june th foreign office list order...	0	0	0	0	0	0	312

Here we have removed all the unwanted data from our comment column.

Stemming:

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP).

```
# Stemming words
snb_stem = SnowballStemmer('english')
df.comment_text = df.comment_text.apply(lambda x: ' '.join(snb_stem.stem(word) for word in word_tokenize(x)))

# Checking any 10 random rows to see the applied changes
df.sample(10)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
3741	pleas make chang explan page recent updat	0	0	0	0	0	0	86
106813	wp newt udi want inform unwit part experi newb...	0	0	0	0	0	0	656
9758	know sure juli constitut allow	0	0	0	0	0	0	65
113598	realli sorri use non free imag show live perso...	0	0	0	0	0	0	199
53811	amend expand talk see respons comment	0	0	0	0	0	0	95
72782	star planetbox type box sure might idea make t...	0	0	0	0	0	0	600
83270	tv marti start articl state began broadcast ma...	0	0	0	0	0	0	200
18988	criteria user benlisquar state medic articl wi...	0	0	0	0	0	0	146
22345	thank thank catch addit vandal brown board edu...	0	0	0	0	0	0	153
60958	fuck jdelanoy german cock sucker fucker mother...	1	0	1	0	1	1	109

```python
# Checking the length of comment_text after cleaning and storing it in cleaned_length variable
df["cleaned_length"] = df.comment_text.str.len()

# Taking a loot at first 10 rows of data
df.head(10)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length	cleaned_length
0	explan edit made usernam hardcor metallica fan...	0	0	0	0	0	0	264	135
1	match background colour seem stuck thank talk ...	0	0	0	0	0	0	112	57
2	man realli tri edit war guy constant remov rel...	0	0	0	0	0	0	233	112
3	make real suggest improv wonder section statis...	0	0	0	0	0	0	622	310
4	sir hero chanc rememb page	0	0	0	0	0	0	67	26
5	congratul well use tool well talk	0	0	0	0	0	0	65	33
6	cocksuck piss around work	1	1	1	0	1	0	44	25
7	vandal matt shirvington articl revert pleas ban	0	0	0	0	0	0	115	47
8	sorri word nonsens offens anyway intend write ...	0	0	0	0	0	0	472	235
9	align subject contrari dulithgow	0	0	0	0	0	0	70	32

```python
# Now checking the percentage of length cleaned
print(f"Total Original Length        : {df.original_length.sum()}")
print(f"Total Cleaned Length         : {df.cleaned_length.sum()}")
print(f"Percentage of Length Cleaned : {(df.original_length.sum()-df.cleaned_length.sum())*100/df.original_length.sum()}%")
```

```
Total Original Length        : 62893130
Total Cleaned Length         : 34297506
Percentage of Length Cleaned : 45.46700728680541%
```

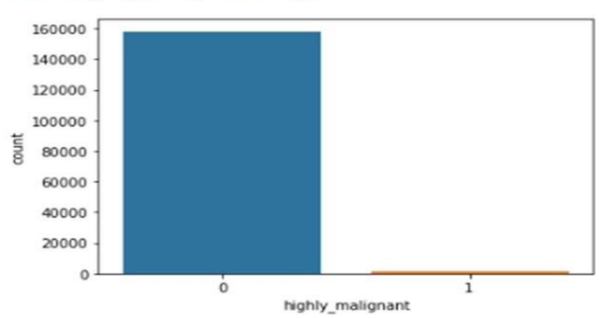

Visualization:

For the Visualization we have used Matplotlib and Seaborn library to plot thenumerical data into graphs.

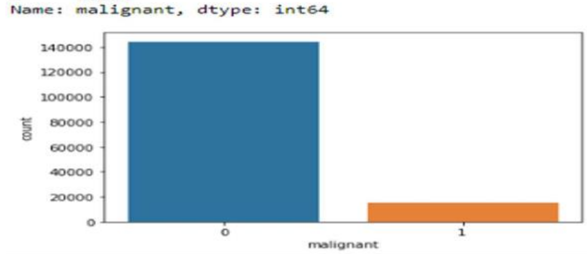
Univariate Analysis

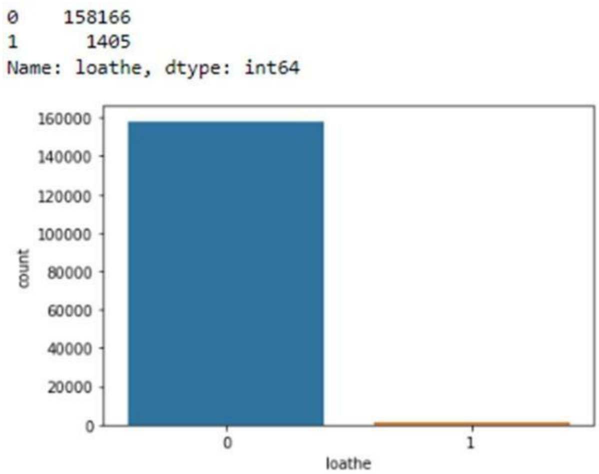
<u>Value counts of different label of comments</u>

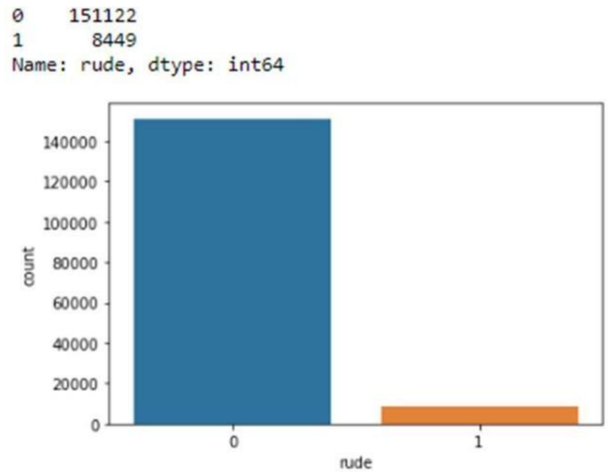
```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```

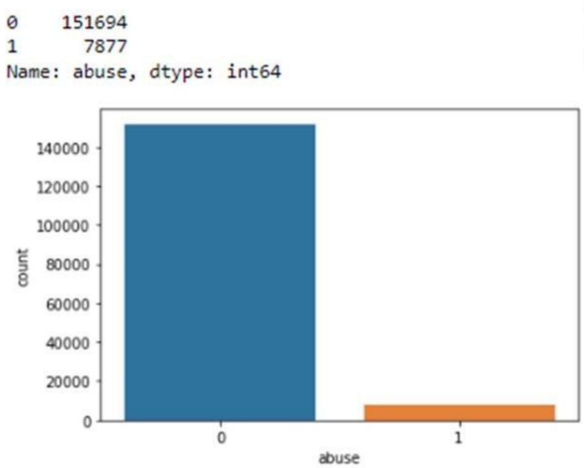


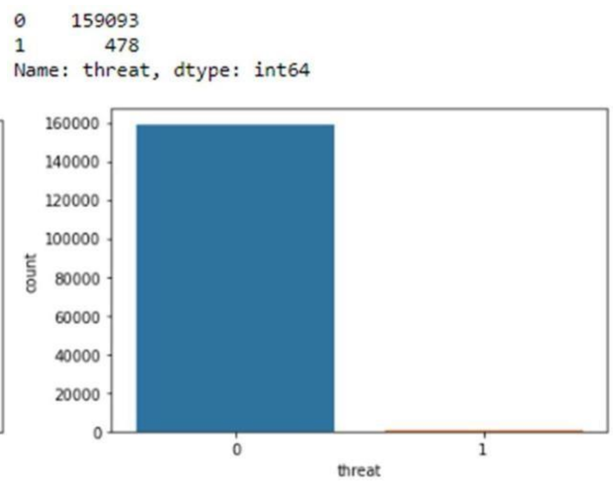

```
0    144277
1     15294
Name: malignant, dtype: int64
```

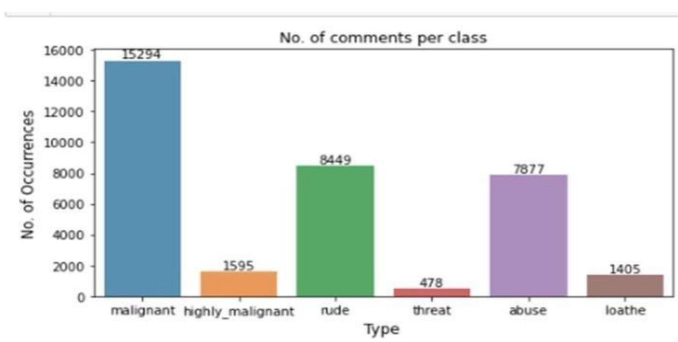


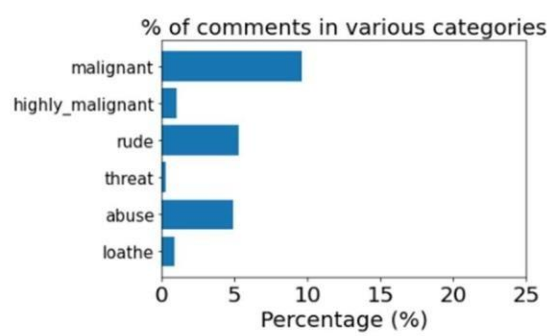












OBSERVATIONS:

Based on the above graphs we can say that there is less percentage of negative comments which are in form of malignant, abusive, loathe, threat and highly_malignant in nature.

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.

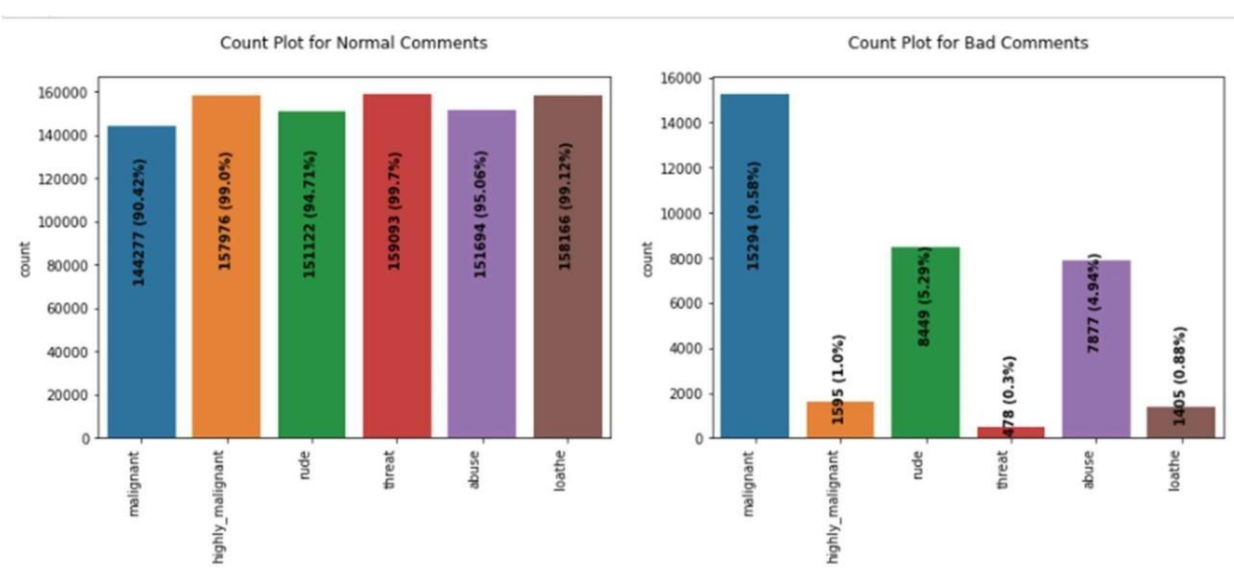
Threat: It contains indication of the comments that are giving any threat to someone.

Abuse: It is for comments that are abusive in nature.

Loathe: It describes the comments which are hateful and loathing in nature.

Comment text: This column contains the comments extracted from various social media platforms.

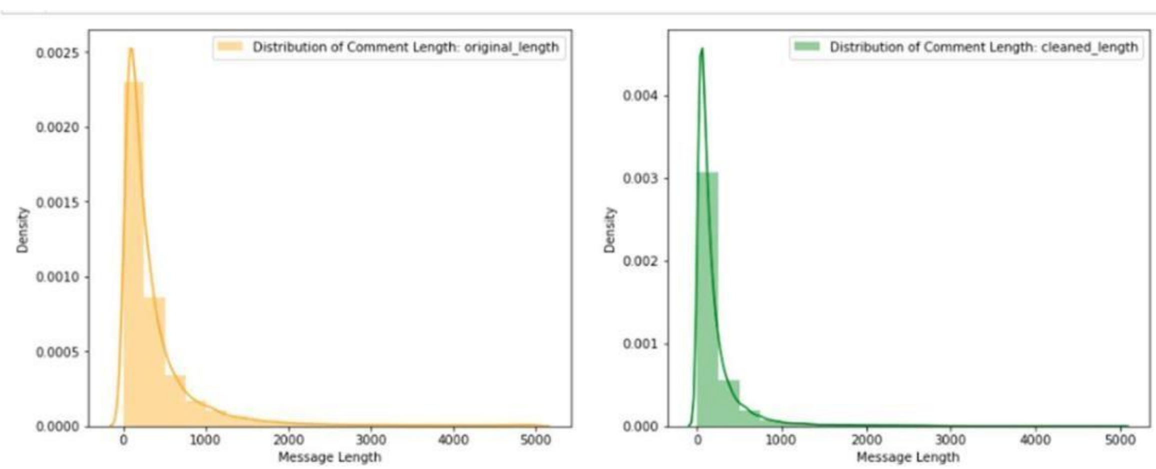
Comparing normal comments and bad comments using count plot



Observation:

- Dataset consists of higher number of Normal Comments than Bad or Malignant Comments. Therefore, it is clear that dataset is imbalanced and needs to be handle accordingly.
- Most of the bad comments are of type malignant while least number of type threat is present in dataset.
- Majority of bad comments are of type malignant, rude and abuse.

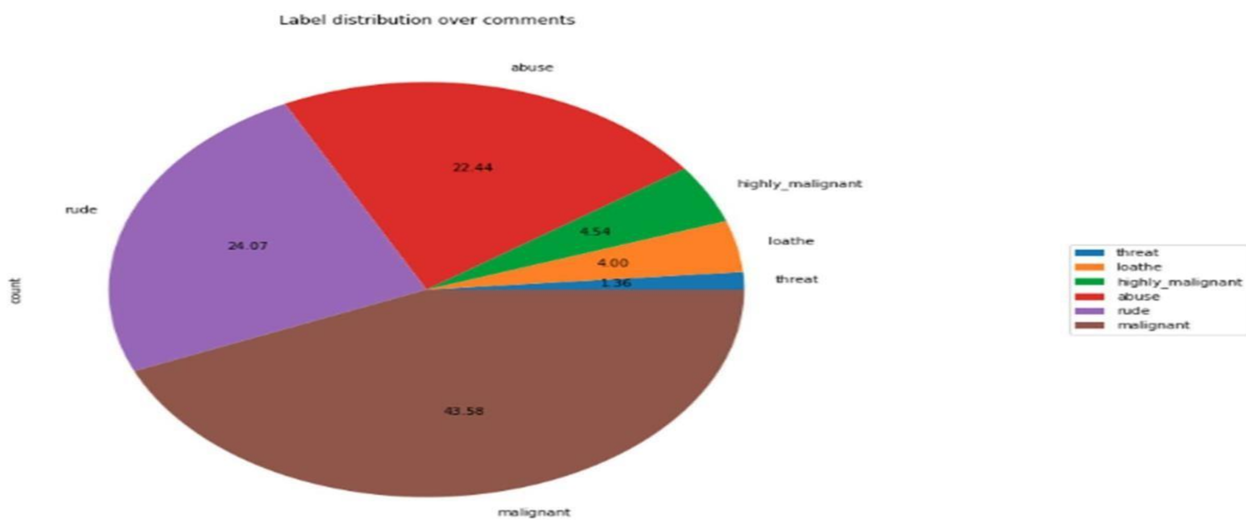
Comparing the comment text length distribution before cleaning and after cleaning



Observation:

Before cleaning comment_text column most of the comment's length lies between 0 to 1100 while after cleaning it has been reduced between 0 to 900.

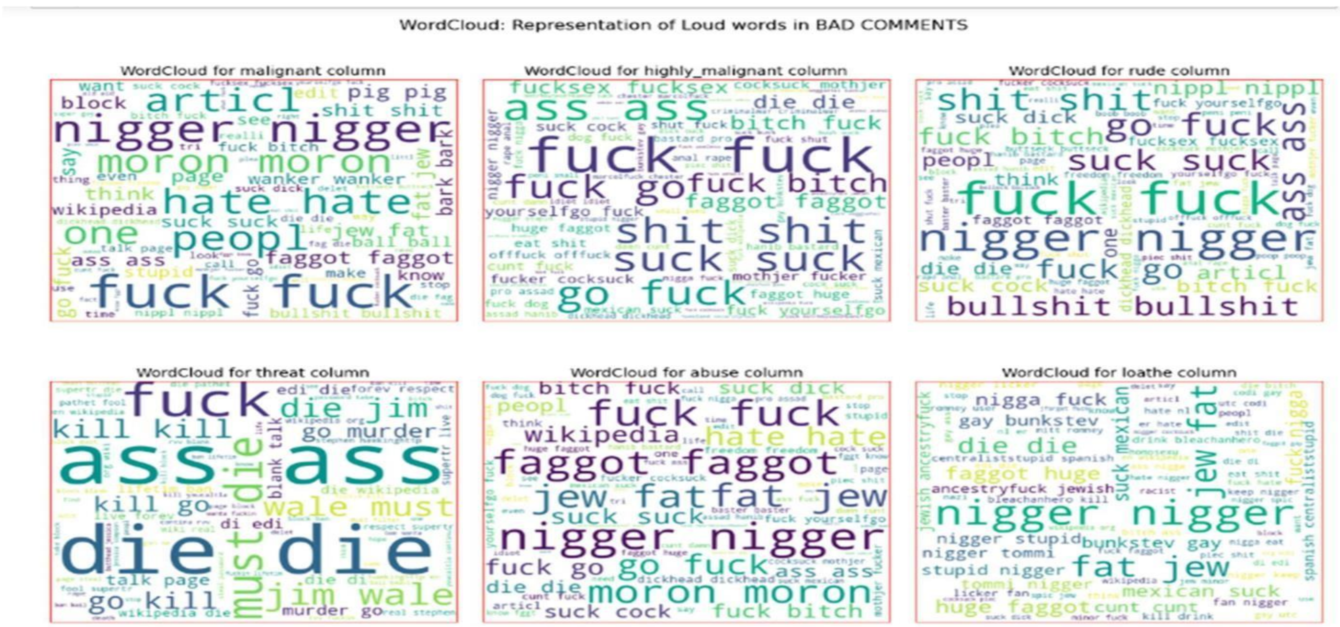
Visualizing the label distribution of comments using pie chart



Plotting heatmap for visualizing the correlation:

Word Cloud: Getting sense of loud words in each of the output labels

I have analyzed the input output logic with word cloud and I have word clouded the sentenced that as classified as foul language in every category.



Observation:

- From word cloud of malignant comments, it is clear that it mostly consists of words like fuck, nigger, moron, hate, suck ect.
- From word cloud of highly_malignant comments, it is clear that it mostly consists of words like ass, fuck, bitch, shit, die, suck, faggot etc.
- From word cloud of rude comments, it is clear that it mostly consists of words like nigger, ass, fuck, suck, bullshit, bitch etc.
- From word cloud of threat comments, it is clear that it mostly consists of words like die, must die, kill, murder etc.

- From the word cloud of abuse comments, it is clear that it mostly consists of words like a moron, nigger, fat, Jew, bitch etc.
- From word cloud of loathe comments, it is clear that it mostly consists of words like nigga, stupid, nigger, die, gay cunt etc.

Data Preparation for Model Training and Testing

1. Convert text to Vectors

```
# Converting text to vectors using TfidfVectorizer
tfidf = TfidfVectorizer(max_features=4000)
features = tfidf.fit_transform(df.comment_text).toarray()

# Checking the shape of features
features.shape
```

```
(159571, 4000)
```

2. Seperating Input and Output Variables

```
# input variables
X = features

# output variables
Y = csr_matrix(df[output_labels]).toarray()

# checking shapes of input and output variables to take care of data imbalance issue
print("Input Variable Shape:", X.shape)
print("Output Variable Shape:", Y.shape)
```

```
Input Variable Shape: (159571, 4000)
Output Variable Shape: (159571, 6)
```

As our target or depedent data is labeled either 0 or 1 which indicates that we are dealing with Classification type of problem. Let's builid a model on Classification now.

Training and Testing Model on our train dataset

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

- Gaussian Naïve Bayes
- Multinomial Naïve Bayes
- Logistic Regression
- Random Forest Classifier
- Linear Support Vector Classifier
- Ada Boost Classifier
- Decision Tree Classifier
- Bagging Classifier

Run and Evaluate selected models

I created a classification function that included the evaluation metrics details for the generation of our Classification Machine Learning models.

```python
# Creating a function to train and test model
def build_models(models,x,y,test_size=0.33,random_state=42):
    # spliting train test data using train_test_split
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=test_size,random_state=random_state)

    # training models using BinaryRelevance of problem transform
    for i in tqdm.tqdm(models,desc="Building Models"):
        start_time = timeit.default_timer()

        sys.stdout.write("\n==========================================================================================\n")
        sys.stdout.write(f"Current Model in Progress: {i} ")
        sys.stdout.write("\n==========================================================================================\n")

        br_clf = BinaryRelevance(classifier=models[i]["name"],require_dense=[True,True])
        print("Training: ",br_clf)
        br_clf.fit(x_train,y_train)

        print("Testing: ")
        predict_y = br_clf.predict(x_test)

        ham_loss = hamming_loss(y_test,predict_y)
        sys.stdout.write(f"\n\tHamming Loss  : {ham_loss}")

        ac_score = accuracy_score(y_test,predict_y)
        sys.stdout.write(f"\n\tAccuracy Score: {ac_score}")

        cl_report = classification_report(y_test,predict_y)
        sys.stdout.write(f"\n{cl_report}")

        end_time = timeit.default_timer()
        sys.stdout.write(f"Completed in [{end_time-start_time} sec.]")

        models[i]["trained"] = br_clf
        models[i]["hamming_loss"] = ham_loss
        models[i]["accuracy_score"] = ac_score
        models[i]["classification_report"] = cl_report
        models[i]["predict_y"] = predict_y
        models[i]["time_taken"] = end_time - start_time

        sys.stdout.write("\n==========================================================================================\n")

    models["x_train"] = x_train
    models["y_train"] = y_train
    models["x_test"] = x_test
    models["y_test"] = y_test

    return models
```

```python
        }

# Taking one forth of the total data for training and testing purpose
half = len(df)//4
trained_models = build_models(models,X[:half,:],Y[:half,:])
```

OUTPUT:

```
Current Model in Progress: GaussianNB
============================================================
==
Training:  BinaryRelevance(classifier=GaussianNB(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.21560957083175086
        Accuracy Score: 0.4729965818458033
              precision    recall  f1-score   support

           0       0.16      0.79      0.26      1281
           1       0.08      0.46      0.13       150
           2       0.11      0.71      0.19       724
           3       0.02      0.25      0.03        44
           4       0.10      0.65      0.17       650
           5       0.04      0.46      0.07       109

   micro avg       0.11      0.70      0.20      2958
   macro avg       0.08      0.55      0.14      2958
weighted avg       0.12      0.70      0.21      2958
 samples avg       0.05      0.07      0.05      2958
Completed in [21.589762399999927 sec.]
```

```
Current Model in Progress: MultinomialNB
============================================================
==
Training:  BinaryRelevance(classifier=MultinomialNB(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.024091657171793898
        Accuracy Score: 0.9074060007595898
              precision    recall  f1-score   support

           0       0.94      0.48      0.63      1281
           1       1.00      0.01      0.01       150
           2       0.93      0.45      0.60       724
           3       0.00      0.00      0.00        44
           4       0.84      0.35      0.49       650
           5       0.00      0.00      0.00       109

   micro avg       0.91      0.39      0.55      2958
   macro avg       0.62      0.21      0.29      2958
weighted avg       0.87      0.39      0.53      2958
 samples avg       0.04      0.03      0.04      2958
Completed in [5.853549199999975 sec.]
```

```
Current Model in Progress: Logistic Regression
============================================================
==
Training:  BinaryRelevance(classifier=LogisticRegression(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.021939486010887455
        Accuracy Score: 0.9128750474743639
              precision    recall  f1-score   support

           0       0.94      0.53      0.67      1281
           1       0.60      0.18      0.28       150
           2       0.96      0.54      0.69       724
           3       0.00      0.00      0.00        44
           4       0.80      0.42      0.56       650
           5       0.91      0.09      0.17       109

   micro avg       0.90      0.46      0.61      2958
   macro avg       0.70      0.29      0.39      2958
weighted avg       0.88      0.46      0.60      2958
 samples avg       0.05      0.04      0.04      2958
Completed in [42.32174509999993 sec.]
```

```
Current Model in Progress: Random Forest Classifier
============================================================
==
Training:  BinaryRelevance(classifier=RandomForestClassifier(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.020407646537536395
        Accuracy Score: 0.9114318268135208
              precision    recall  f1-score   support

           0       0.86      0.63      0.73      1281
           1       0.50      0.06      0.11       150
           2       0.89      0.72      0.79       724
           3       0.00      0.00      0.00        44
           4       0.70      0.56      0.62       650
           5       0.83      0.14      0.24       109

   micro avg       0.82      0.58      0.68      2958
   macro avg       0.63      0.35      0.41      2958
weighted avg       0.80      0.58      0.66      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [993.5433835 sec.]
```

```
Current Model in Progress: Support Vector Classifier
================================================================
==
Training:  BinaryRelevance(classifier=LinearSVC(max_iter=3000), require_dense=[True, True])
Testing:

       Hamming Loss  : 0.019977212305355107
       Accuracy Score: 0.9135586783137106
            precision    recall  f1-score   support

           0       0.84      0.66      0.74      1281
           1       0.52      0.27      0.35       150
           2       0.90      0.67      0.77       724
           3       0.58      0.16      0.25        44
           4       0.74      0.56      0.64       650
           5       0.78      0.29      0.43       109

   micro avg       0.82      0.60      0.69      2958
   macro avg       0.73      0.43      0.53      2958
weighted avg       0.81      0.60      0.69      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [9.800140100000135 sec.]
```

```
Current Model in Progress: Ada Boost Classifier
================================================================
==
Training:  BinaryRelevance(classifier=AdaBoostClassifier(), require_dense=[True, True])
Testing:

       Hamming Loss  : 0.023281428028864414
       Accuracy Score: 0.9044436004557539
            precision    recall  f1-score   support

           0       0.83      0.55      0.66      1281
           1       0.48      0.24      0.32       150
           2       0.88      0.62      0.73       724
           3       0.50      0.18      0.27        44
           4       0.74      0.38      0.50       650
           5       0.63      0.29      0.40       109

   micro avg       0.81      0.50      0.62      2958
   macro avg       0.68      0.38      0.48      2958
weighted avg       0.79      0.50      0.61      2958
 samples avg       0.05      0.04      0.05      2958
Completed in [658.2901809 sec.]
```

```
Current Model in Progress: Decision Tree Classifier
================================================================
==
Training:  BinaryRelevance(classifier=DecisionTreeClassifier(), require_dense=[True, True])
Testing:

       Hamming Loss  : 0.026522344600582 35
       Accuracy Score: 0.8848461830611469
            precision    recall  f1-score   support

           0       0.69      0.69      0.69      1281
           1       0.28      0.20      0.23       150
           2       0.77      0.74      0.75       724
           3       0.21      0.14      0.17        44
           4       0.56      0.59      0.58       650
           5       0.41      0.32      0.36       109

   micro avg       0.65      0.63      0.64      2958
   macro avg       0.49      0.45      0.46      2958
weighted avg       0.64      0.63      0.64      2958
 samples avg       0.06      0.06      0.06      2958
Completed in [1209.0093106999998 sec.]
```

```
Current Model in Progress: Bagging Classifier
================================================================
==
Training:  BinaryRelevance(classifier=BaggingClassifier(base_estimator=LinearSVC()),
           require_dense=[True, True])
Testing:

       Hamming Loss  : 0.020027851626788 2
       Accuracy Score: 0.9138625142423091
            precision    recall  f1-score   support

           0       0.85      0.65      0.74      1281
           1       0.48      0.21      0.30       150
           2       0.91      0.66      0.76       724
           3       0.67      0.18      0.29        44
           4       0.77      0.51      0.61       650
           5       0.79      0.24      0.37       109

   micro avg       0.84      0.58      0.68      2958
   macro avg       0.75      0.41      0.51      2958
weighted avg       0.83      0.58      0.68      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [218.38481049999973 sec.]
```

Observation:

From the above model comparison, it is clear that Linear Support Vector Classifier performs better with Accuracy Score: 91.35586783137106% and Hamming Loss: 1.9977212305355107% than the other classification models. Therefore, I am now going to use Linear Support Vector Classifier for further Hyperparameter tuning process. With the help of hyperparameter tuning process I will be trying my best to increase the accuracy score of our final classification machine learning model.

Hyperparameter Tuning:

After comparing all the classification models, I have selected Linear Support Vector Classifier as my best model and have listed down it's parameters above referring the sklearn webpage. I am using the Grid Search CV method for hyper parameter tuning my best model. I have trained the Grid Search CV with the list of parameters I feel it should check for best possible outcomes. So the Grid Search CV has provided me with the best parameters list out of all the combinations it used to train the model that I can use on my final model.

```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge', multi_class='ovr', penalty='l2', random_state=42))
Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)
h_loss = hamming_loss(y_test,fmod_pred)*100
print("Hamming loss for the Best Model is:", h_loss)
```

```
Accuracy score for the Best Model is: 91.51069518716578
Hamming loss for the Best Model is: 1.9593917112299464
```

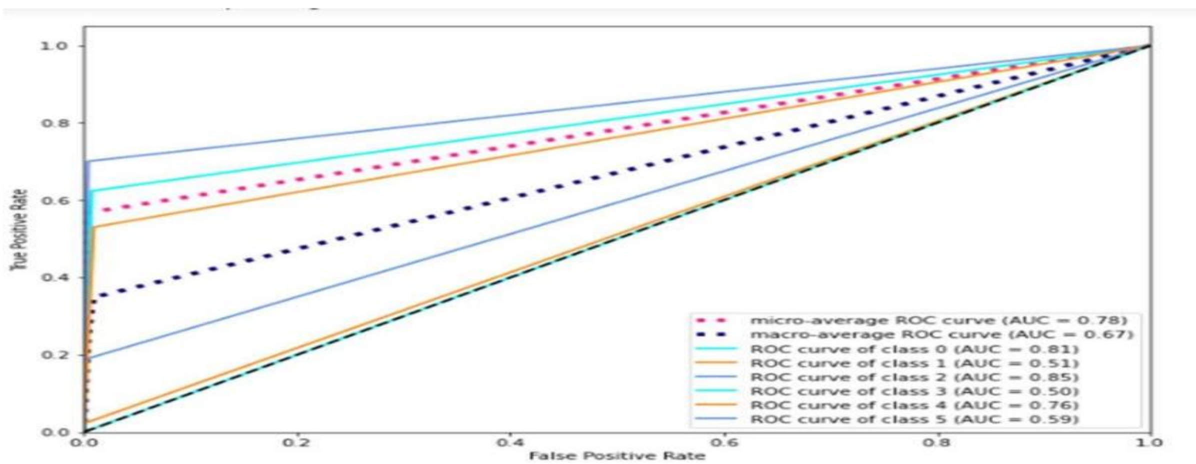
```
# Choosing Linear Support Vector Classifier model

fmod_param = {'estimator__penalty' : ['l1', 'l2'],
              'estimator__loss' : ['hinge', 'squared_hinge'],
              'estimator__multi_class' : ['ovr', 'crammer_singer'],
              'estimator__random_state' : [42, 72, 111]
             }
SVC = OneVsRestClassifier(LinearSVC())
GSCV = GridSearchCV(SVC, fmod_param, cv=3)
x_train,x_test,y_train,y_test = train_test_split(X[:half,:], Y[:half,:], test_size=0.30, random_state=42)
GSCV.fit(x_train,y_train)
GSCV.best_params_
```

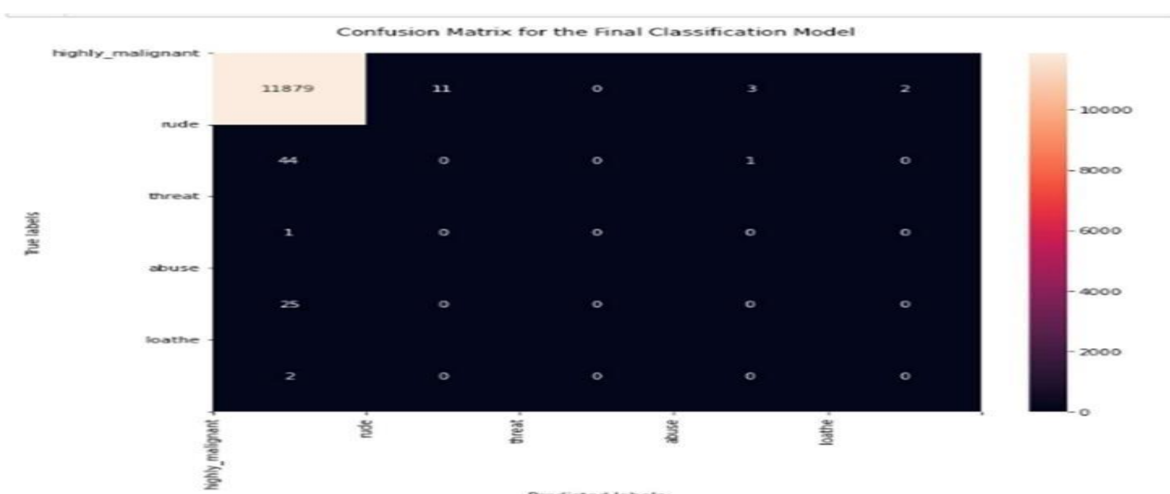
```
{'estimator__loss': 'hinge',
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2',
 'estimator__random_state': 42}
```

I have successfully incorporated the Hyper Parameter Tuning on my Final Model and received the accuracy score for it.

AUC ROC Curve for Final Model



Confusion Matrix for Final Model



Model Saving or Serialization

```
# selecting the best model
best_model = trained_models['Support Vector Classifier']['trained']

# saving the best classification model
joblib.dump(best_model,open('Malignant_comments_classifier.pkl','wb'))
```

I am using the joblib option to save the final classification model but it can be done using pickle too.

Final predicted dataframe:

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	yo bitch ja rule succes ever what hate sad mof...	0	0	0	0	0	0
1	rfc titl fine imo	0	0	0	0	0	0
2	sourc zaw ashton lapland	0	0	0	0	0	0
3	look back sourc inform updat correct form gues...	0	0	0	0	0	0
4	anonym edit articl	0	0	0	0	0	0
...	...	...	...	...	...	...	...
153159	total agre stuff noth long crap	0	0	0	0	0	0
153160	throw field home plate get faster throw cut ma...	0	0	0	0	0	0
153161	okinotorishima categori see chang agre correct...	0	0	0	0	0	0
153162	one found nation eu germani law return quit si...	0	0	0	0	0	0
153163	stop alreadi bullshit welcom fool think kind e...	0	0	0	0	0	0

153164 rows × 7 columns

Interpretation of the Results

Starting with univariate analysis, with the help of count plot it was found that dataset is imbalanced with having higher number of records for normal comments than bad comments (including malignant, highly malignant, rude, threat, abuse and loathe). Also, with the help of distribution plot for comments length it was found that after cleaning most of comments length decreases from range 0-1100 to 0-900. Moving further with word cloud it was found that malignant comments consists of words like fuck, nigger, moron, hate, suck etc. highly_malignant comments consists of words like ass, fuck, bitch, shit, die, suck, faggot etc. rude comments consists of words like nigger, ass, fuck, suck, bullshit, bitch etc. threat comments consists of words like die, must die, kill, murder etc. abuse comments consists of words like moron, nigger, fat, jew, bitch etc. and loathe comments consists of words like nigga, stupid, nigger, die, gay, cunt etc.

CONCLUSION

Key Findings and Conclusions of the Study

The finding of the study is that only few users over online use unparliamentarylanguage. And most of these sentences have more stop words and are being quite long. Few motivated disrespectful crowds use thesefoul languages in the online forum to bully the people around and to stop themfrom doing these things that they are not supposed to do.

Problems faced while working in this project:

- More computational power was required as it took more than 2 hours
- Imbalanced dataset and bad comment texts
- Good parameters could not be obtained using hyperparameter tuning as time was consumed more

Areas of improvement:

- Could be provided with a good dataset which does not take more time.
- Less time complexity
- Providing a proper balanced dataset with less errors.

My point of view from my project is that we need to use proper words which are respectful and also avoid using abusive, vulgar and worst words in social media. It can cause many problems which could affect our lives. Try to be polite, calm and composed while handling stress and negativity and one of the best solutions is to avoid it and overcoming in a positive manner.

THANK YOU