



---

***MALIGNANT COMMENT CLASSIFIER  
PROJECT REPORT***

---

**Submitted by:**  
**Akanksha Kaushik**

## **ACKNOWLEDGMENT**

I would like to express my deepest gratitude to my SME - Mohd Kashif as well as FlipRobo Technologies who gave me the opportunity to do this project on Malignant Comments Classification, which also helped me in doing lots of research which is a great learning for me as while doing the project I was able to learn so many new things like Natural Language Processing and Natural Language Toolkit parts.

I also have utilized a few external resources that helped me in completing the project. All the external resources that were used are listed below:

- 1.) <https://www.google.com/>
- 2.) <https://www.youtube.com/>
- 3.) [https://scikit-learn.org/stable/user\\_guide.html](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. In some cases, these online comments contain explicit language which may hurt the readers. Comments containing explicit language can be classified into myriad categories such as Toxic, Severe Toxic, Obscene, Threat, Insult, and Identity Hate. 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 the unruly language and protect internet users from becoming victims of online harassment and cyberbullying.

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.

## • **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 share their 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 like India lead the list of social media usage. More than 3.8 billion people use social media.

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

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

mob



To solve this problem, we are now building a model 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.

- **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.

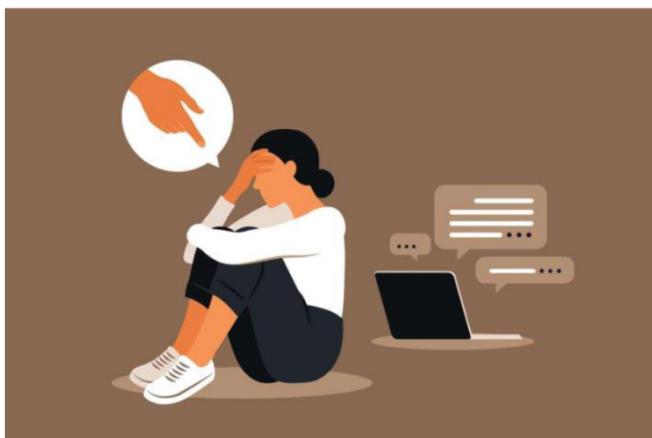
I have used 9 different Classification algorithms and shortlisted the best on basis of the metrics of performance and I have chosen one algorithm and build a model in that algorithm.

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.

Our 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.

- **Motivation for the Problem Undertaken**

One of the first lessons we learn as children is that the louder you scream and the bigger of a tantrum you throw, you more you get your way. Part of growing up 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 and bring 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 motivation I 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 Modeling of the Problem**

The libraries 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 joblib

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
from scipy import interp
from itertools import cycle
import matplotlib.ticker as plticker

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

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

I need to build multiple classification machine learning models. Before model building 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 -

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. 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 first attribute is a unique ID associated with each comment.

The data set includes:

Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.

Highly Malignant: It denotes comments that are highly malignant and hurtful.

Rude: It denotes comments that are very rude and offensive.

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

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

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

ID: It includes unique Ids associated with each comment text given.

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

- **Data Pre-processing Done**

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

1. Loading the dataset
2. Remove null values
3. Drop column id
4. Convert comment text to lower case and replace '\n' with single space.
5. Keep only text data ie. a-z' and remove other data from comment text.
6. Remove stop words and punctuations
7. Apply Stemming using SnowballStemmer
8. Convert text to vectors using TfIdfVectorizer
9. Load saved or serialized model
10. Predict values for multi class label

- **Loading the Dataset:**

## Loading the DataSets

```
df_train = pd.read_csv(r"C:\Users\Gautam Sarasvat\Downloads\Malignant Comments Classifier Project\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	ffa4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	fee36eab5c267c9	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(r"C:\Users\Gautam Sarasvat\Downloads\Malignant Comments Classifier Project\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	fffd0960ee309b5	. \n i totally agree, this stuff is nothing bu...
153160	ffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...
153161	ffda9e8d6faf9e	" \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 original length 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 all the text data in our comment text column into lowercase format for easier interpretation.

## Data Cleansing

```

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

```

```

# 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

```
# 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 could't etc
df.comment_text = df.comment_text.apply(lambda x: ' '.join(regexp_tokenize(x, "[a-zA-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
120278	please vandalize pages edit drum continue bloc...	0	0	0	0	0	0	145
104134	work npia vandalised anything fact written npi...	0	0	0	0	0	0	381
127010	note made comment sure protocol notifications ...	0	0	0	0	0	0	99
108180	fact stupid limited understanding plays acedem...	1	0	0	0	0	0	206
146310	thank know pliat duuts frisian plattdeutsch go ...	0	0	0	0	0	0	114
22562	tya allen talk ctrb mail	0	0	0	0	0	0	82
117174	cnn article merits consideration imo	0	0	0	0	0	0	43
64914	list countries military expenditures need help...	0	0	0	0	0	0	1183
90158	russian gave several sources say millions peop...	0	0	0	0	0	0	261
156803	important make finland scandinavia difference ...	0	0	0	0	0	0	455

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
37898	quot materi paraphras	0		0	0	0	0	44
12140	android sideload alway process get line busi a...	0		0	0	0	0	205
145280	insert suggest tribe dan simpli said historian...	0		0	0	0	0	282
42434	film nehram hope studi go well vacat first eve...	0		0	0	0	0	1455
19706	question ed wonder feel share common ancestor ...	1		0	0	0	0	274
34020	best wish bet wish warmest greet land smile le...	0		0	0	0	0	162
79779	pleas stop continu vandal page block edit wiki...	0		0	0	0	0	98
68057	state user troll assum good faith threaten use...	0		0	0	0	0	219
158988	yes understand yes yes yes	0		0	0	0	0	33
44466	wonder chang color raw signatur	0		0	0	0	0	58

```

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

```

#Now checking the percentage of Length cleaned

```

```

print(f"Total Original Length : {df.original_length.sum()}")
print(f"Percentage of Length Cleaned : {(df.original_length.sum()*100/df.original_length.sum())}%")

```

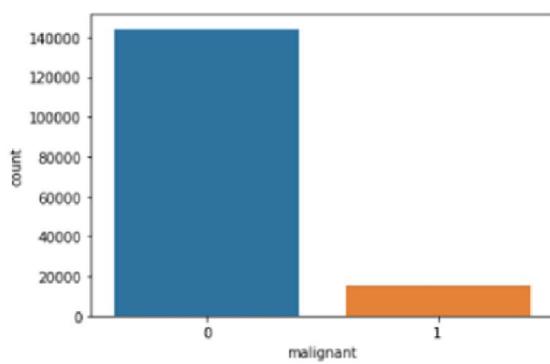
```

Total Original Length      : 62893130
Percentage of Length Cleaned : 100.0%

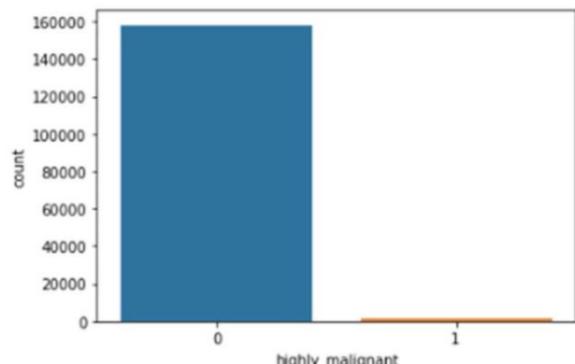
```

## Visualization:

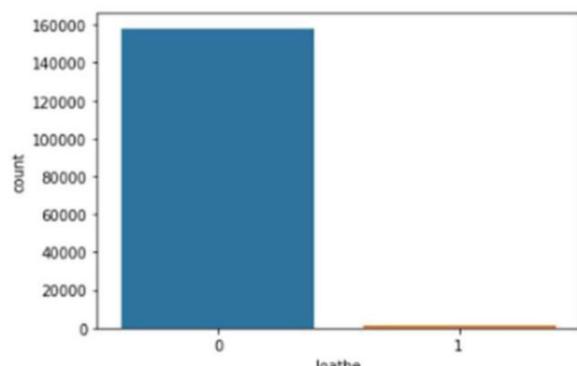
0 144277  
1 15294  
Name: malignant, dtype: int64



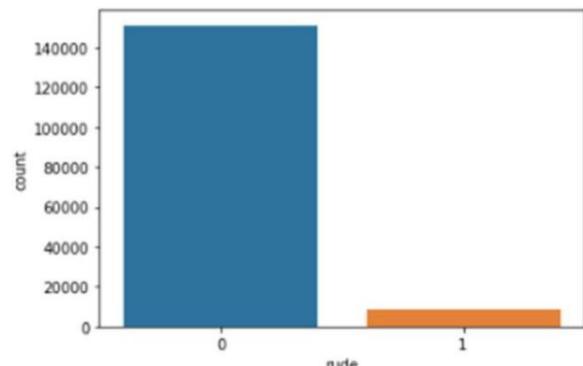
0 157976  
1 1595  
Name: highly\_malignant, dtype: int64



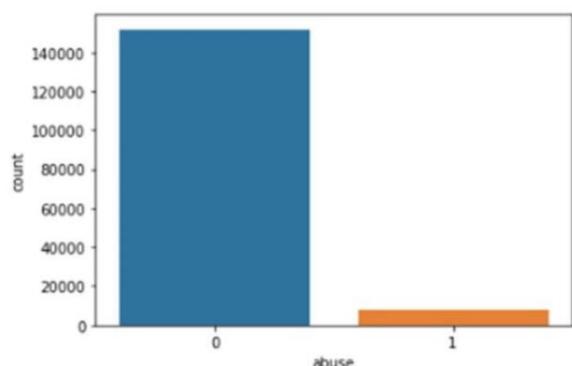
0 158166  
1 1405  
Name: loathe, dtype: int64



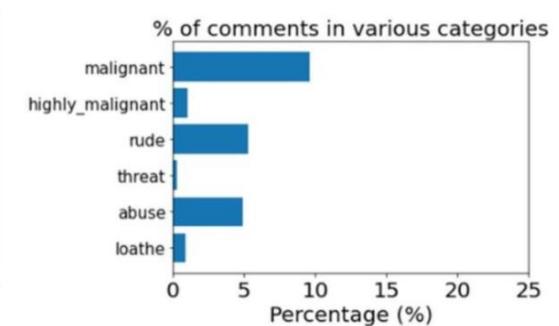
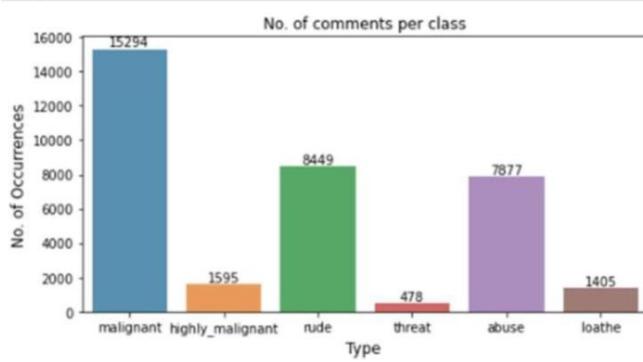
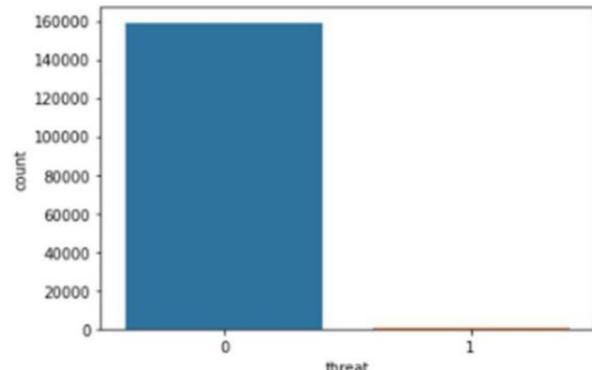
0 151122  
1 8449  
Name: rude, dtype: int64



0 151694  
1 7877  
Name: abuse, dtype: int64



0 159093  
1 478  
Name: threat, 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.

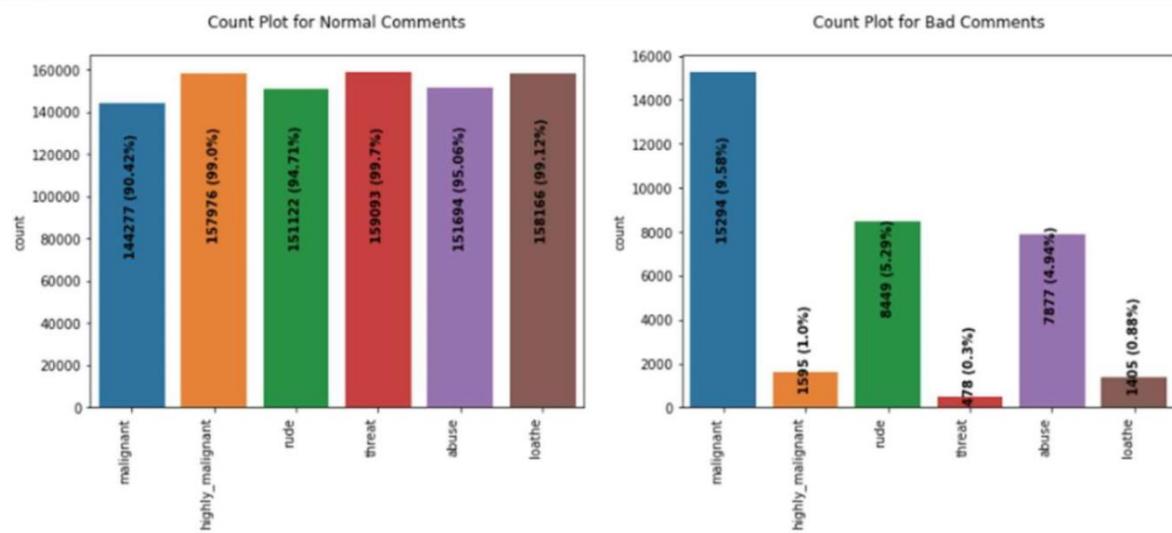
Threat: It contains indication of the comments that are giving any threat to someone.

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.

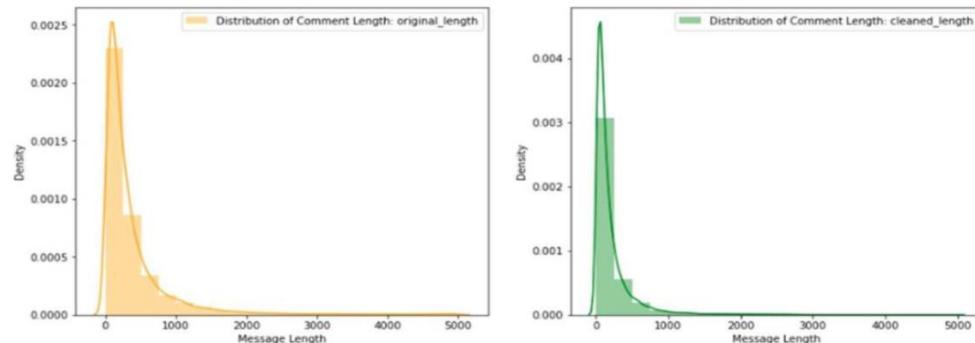
## Comparing normal comments and bad comments using count plot



Observation:

1. 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.
2. Most of the bad comments are of type malignant while least number of type threat is present in dataset.
3. Majority of bad comments are of type malignant, rude and abuse.

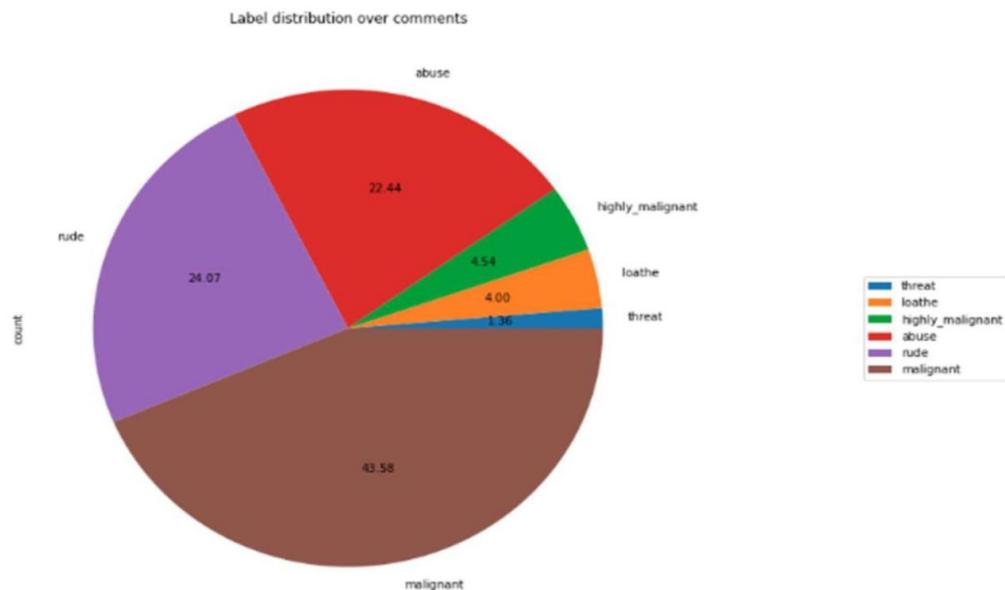
## Comparing the comment text length distribution before cleaning and after cleaning



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

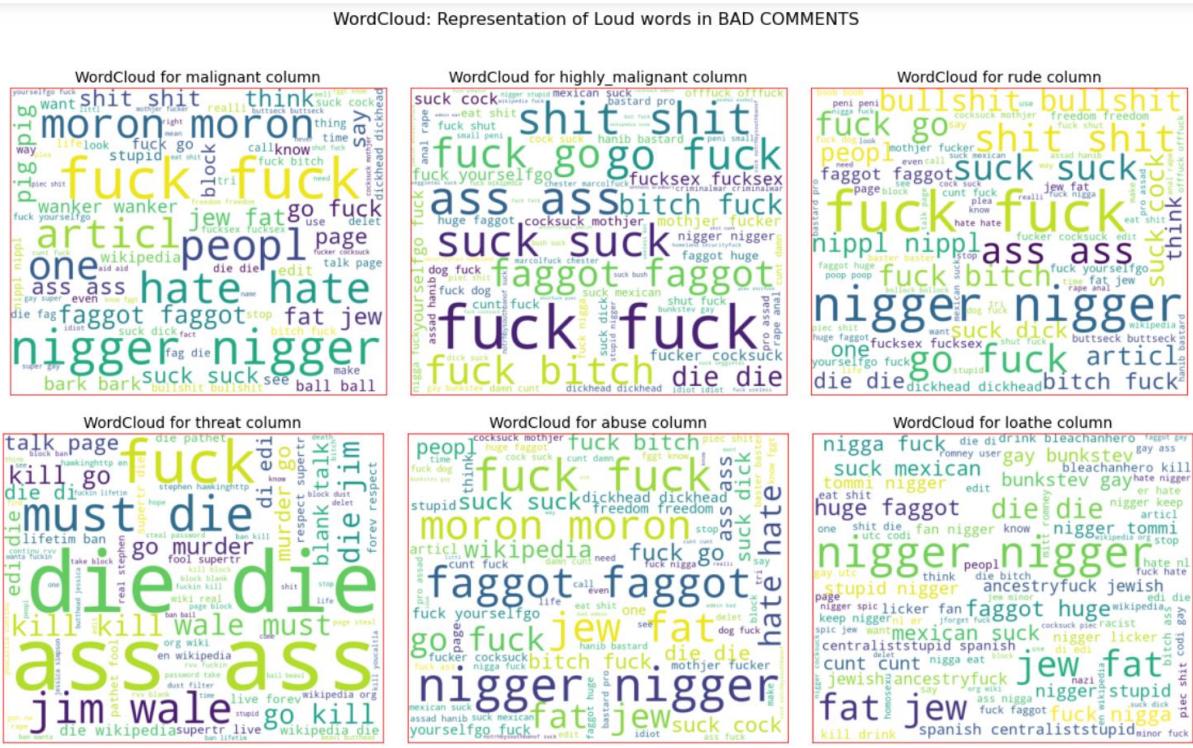


## Plotting heatmap for visualizing the correlation:



## 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 sentences that are classified as foul language in every category. A *tag/word cloud is a novelty visual representation of text data, typically used to depict keyword metadata on websites or to visualize free-form text. It's an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance.*



### Observation:

- From wordcloud of malignant comments, it is clear that it mostly consists of words like fuck, nigger, moron, hate, suck etc.
- From wordcloud of highly\_malignant comments, it is clear that it mostly consists of words like ass, fuck, bitch, shit, die, suck, faggot etc.
- From wordcloud of rude comments, it is clear that it mostly consists of words like nigger, ass, fuck, suck, bullshit, bitch etc.
- From wordcloud of threat comments, it is clear that it mostly consists of words like die, must die, kill, murder etc.
- From wordcloud of abuse comments, it is clear that it mostly consists of words like moron, nigger, fat, jew, bitch etc.
- From wordcloud 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. Separating 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 dependent data is labeled either 0 or 1 which indicates that we are dealing with Classification type of problem. Let's build 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:

- 1) Gaussian Naïve Bayes
- 2) Multinomial Naïve Bayes
- 3) Logistic Regression
- 4) Random Forest Classifier
- 5) Linear Support Vector Classifier
- 6) Ada Boost Classifier
  
- 7) Decision Tree Classifier
- 8) 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.

# Classification Machine Learning Model building

## 3. Training and Testing Model on our train dataset

```
# Creating a function to train and test model
def build_models(models,x,y,test_size=0.33,random_state=42):
    # splitting 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

# Preparing the List of models for classification purpose
from skmultilearn.problem_transform import BinaryRelevance
models = {"GaussianNB": {"name": GaussianNB()},
          "MultinomialNB": {"name": MultinomialNB()},
          "Logistic Regression": {"name": LogisticRegression()},
          "Random Forest Classifier": {"name": RandomForestClassifier()},
          "Support Vector Classifier": {"name": LinearSVC(max_iter = 3000)},
          "Ada Boost Classifier": {"name": AdaBoostClassifier()},
          "Decision Tree Classifier": {"name": DecisionTreeClassifier()},
          "Bagging Classifier": {"name": BaggingClassifier(base_estimator=LinearSVC())},
         }

# 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:

```
Building Models:  0% | 0/8 [00:00<?, ?it/s]

=====
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 [15.07817609999995 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
      -----
      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 [4.109494100000006 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 [21.999491199999966 sec.]
=====

=====

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

Hamming Loss : 0.02049626535004431
Accuracy Score: 0.9112799088492214
      precision    recall   f1-score   support
0         0.86     0.63     0.73     1281
1         0.42     0.05     0.09     150
2         0.88     0.71     0.79     724
3         0.00     0.00     0.00      44
4         0.72     0.53     0.61     650
5         0.87     0.12     0.21     109
micro avg    0.83     0.57     0.68     2958
macro avg    0.62     0.34     0.41     2958
weighted avg  0.80     0.57     0.66     2958
samples avg   0.06     0.05     0.05     2958
Completed in [832.8201614000001 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 [5.7365087999999105 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 [504.22326969999995 sec.]

---

=====

Current Model in Progress: Decision Tree Classifier

---

Training: BinaryRelevance(classifier=DecisionTreeClassifier(), require\_dense=[True, True])

Testing:

---

Hamming Loss : 0.026180529180908976
Accuracy Score: 0.8853019369540448
precision recall f1-score support
0 0.68 0.69 0.69 1281
1 0.31 0.26 0.28 150
2 0.77 0.75 0.76 724
3 0.24 0.11 0.15 44
4 0.58 0.62 0.60 650
5 0.43 0.33 0.37 109

```

    micro avg      0.65      0.65      0.65      2958
    macro avg      0.50      0.46      0.48      2958
    weighted avg   0.64      0.65      0.64      2958
    samples avg    0.06      0.06      0.06      2958
Completed in [1005.3080897 sec.]
=====

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

Hamming Loss : 0.02006583111786302
Accuracy Score: 0.9133308013672616
precision     recall   f1-score   support
-----
0            0.85      0.64      0.73      1281
1            0.49      0.20      0.28      150
2            0.91      0.66      0.76      724
3            0.50      0.11      0.19       44
4            0.77      0.53      0.63      650
5            0.81      0.28      0.41      109
micro avg    0.83      0.58      0.68      2958
macro avg    0.72      0.40      0.50      2958
weighted avg  0.82      0.58      0.68      2958
samples avg   0.06      0.05      0.05      2958
Completed in [160.66932899999983 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 its 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.

---

```
# 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': 'crammer_singer',
 'estimator__penalty': 'l1',
 'estimator__random_state': 42}

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
```

---

# AUC ROC Curve for Final Model

```
n_classes = y_test.shape[1]

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], fmod_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), fmod_pred.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

plt.rcParams["figure.figsize"] = (10,8) # used to change the output figure size

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(
    fpr["micro"],
    tpr["micro"],
    label="micro-average ROC curve (AUC = {:.2f})".format(roc_auc["micro"]),
    color="deeppink",
    linestyle=":",
    linewidth=4,
)

plt.plot(
    fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (AUC = {:.2f})".format(roc_auc["macro"]),
    color="navy",
    linestyle=":",
    linewidth=4,
)
```

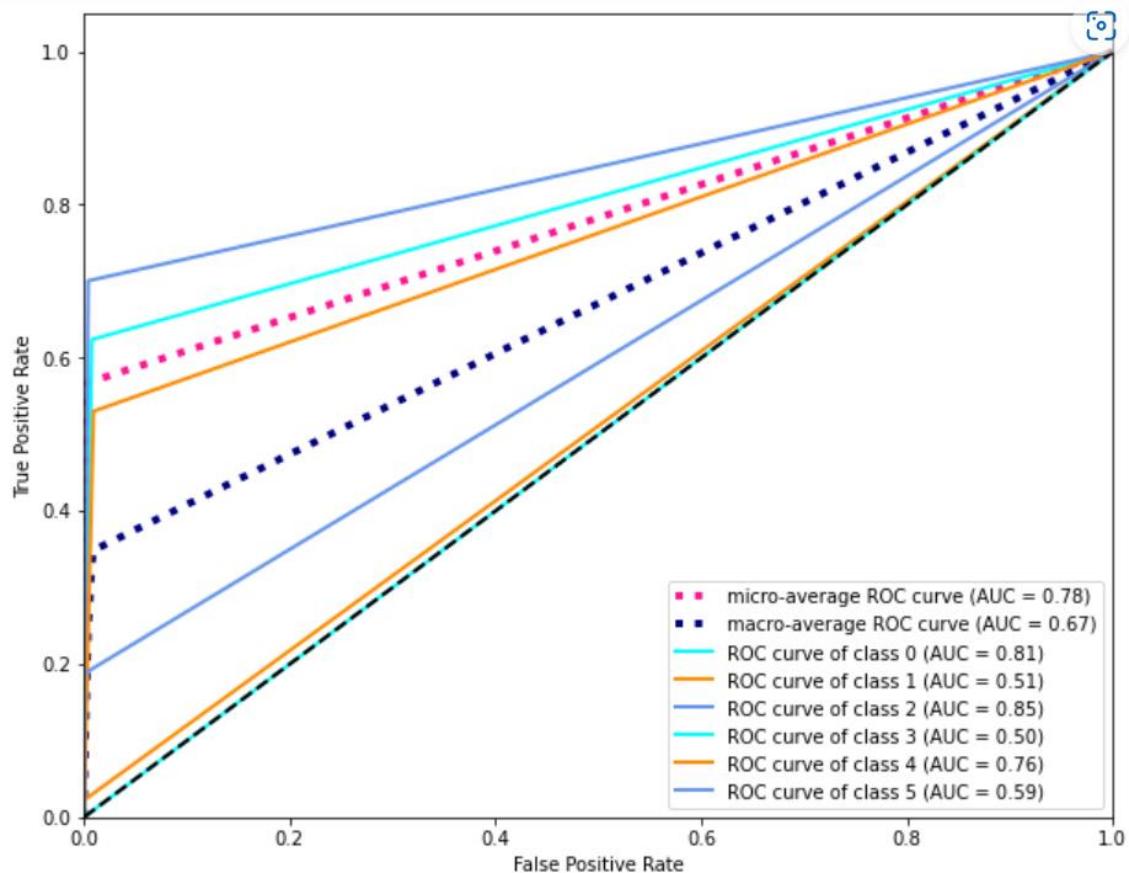
```

colors = cycle(["aqua", "darkorange", "cornflowerblue"])
for i, color in zip(range(n_classes), colors):
    plt.plot(
        fpr[i],
        tpr[i],
        color=color,
        lw=2,
        label="ROC curve of class {0} (AUC = {1:0.2f})".format(i, roc_auc[i]),
    )

plt.plot([0, 1], [0, 1], "k--", lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic (ROC) and Area under curve (AUC) for multiclass labels\n")
plt.legend(loc="lower right")
plt.show()

```

Receiver operating characteristic (ROC) and Area under curve (AUC) for multiclass labels

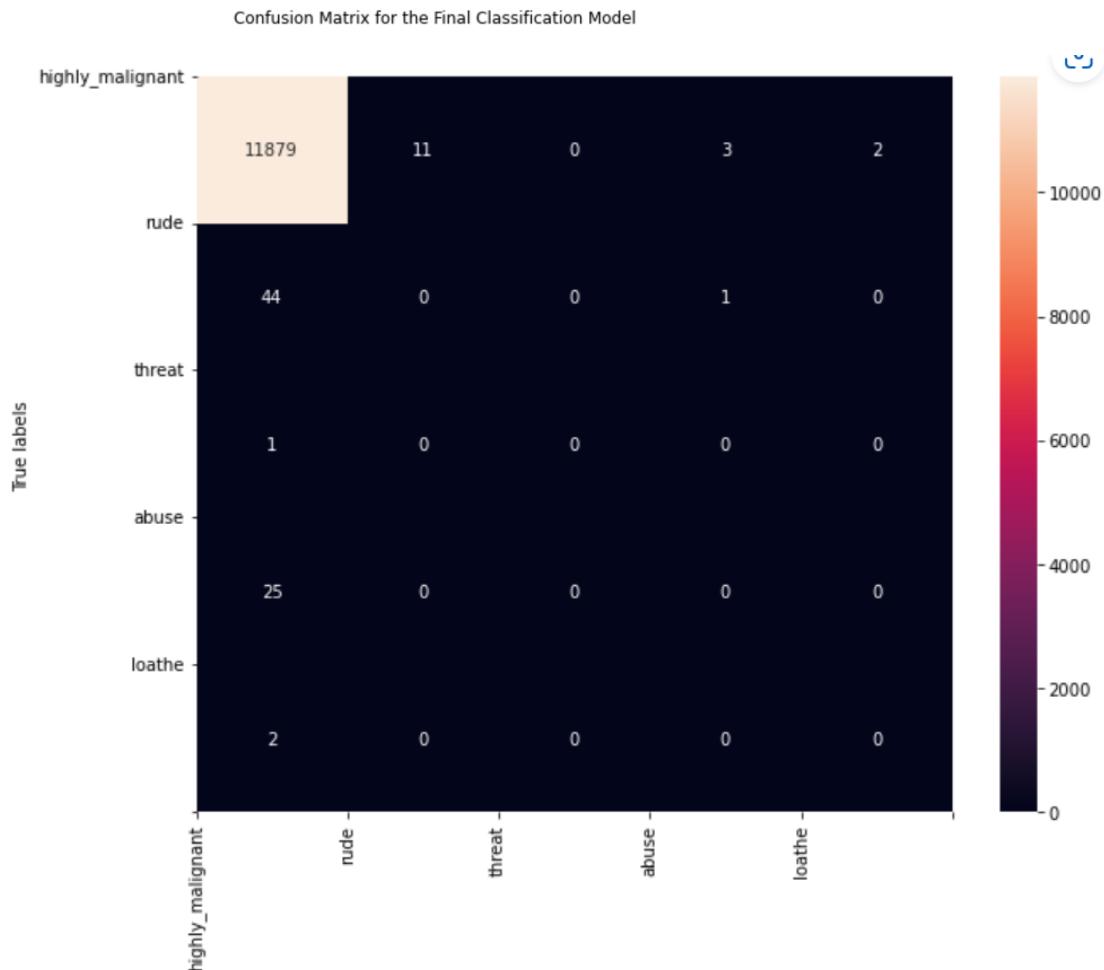


## Confusion Matrix for Final Model

```
print("Confusion matrix:\n\n", multilabel_confusion_matrix(y_test, fmod_pred))

plt.rcParams["figure.figsize"] = (10,8) # used to change the output figure size
ax= plt.subplot()
cm = confusion_matrix(np.asarray(y_test).argmax(axis=1), np.asarray(fmod_pred).argmax(axis=1))
sns.heatmap(cm, annot=True, fmt='g', ax=ax); # annot=True to annotate cells, fmt='g' to disable scientific notation

# title, labels and ticks
ax.set_title('Confusion Matrix for the Final Classification Model\n');
ax.set_xlabel('Predicted labels'); ax.set_ylabel('True labels');
loc = plticker.MultipleLocator()
ax.xaxis.set_major_locator(loc); ax.yaxis.set_major_locator(loc);
ax.set_xticklabels(comments_labels); ax.set_yticklabels(comments_labels);
plt.xticks(rotation=90); plt.yticks(rotation=0);
plt.show()
```



## Model Saving or Serialization

```
1 # selecting the best model
2 best_model = trained_models['Support Vector Classifier']['trained']
3
4 # saving the best classification model
5 joblib.dump(best_model,open('Malignant_comments_classifier_Project.pkl','wb'))
6
```

## Preprocessing Pipeline for test dataframe

The following preprocessing pipeline is required to perform model prediction:

1. Use the test dataset
2. Remove null values if any
3. Drop column id
4. Convert comment text to lower case and replace '\n' with single space
5. Keep only text data ie. a-z' and remove other data from comment text
6. Remove stop words and punctuations
7. Apply Stemming using SnowballStemmer
8. Convert text to vectors using TfidfVectorizer
9. Load saved or serialized best model
10. Predict values and create a new CSV file"

```
# Remove null values
if df_test.isnull().sum()[1] != 0:
    df_test.dropna(inplace=True)

# Drop coulmn id
df_test.drop(columns=['id'],inplace=True)

# Convert comment text to lower case and replace '\n' with single space
df_test["comment_text"] = df_test.comment_text.str.lower()
df_test["comment_text"] = df_test.comment_text.str.replace('\n', ' ')

# Keep only text data i.e., a-z' and remove other data from comment text.
df_test.comment_text = df_test.comment_text.apply(lambda x: ' '.join(reexp_tokenize(x, "[a-zA-Z]+")))

# Remove stopwords
df_test.comment_text = df_test.comment_text.apply(lambda x: ' '.join(word for word in x.split() if word not in stop_words).strip())

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

# Apply Stemming using SnowballStemmer
df_test.comment_text = df_test.comment_text.apply(lambda x: ' '.join(snb_stem.stem(word) for word in word_tokenize(x)))

print(df_test.info(memory_usage="deep"))

# Convert text to vectors using TfidfVectorizer
tfidf = TfidfVectorizer(analyzer = 'word', max_features=4000)
test_features = tfidf.fit_transform(df_test.comment_text).toarray()
```

```

# Convert text to vectors using TfidfVectorizer
tfidf = TfidfVectorizer(analyzer = 'word', max_features=4000)
test_features = tfidf.fit_transform(df_test.comment_text).toarray()

# Load saved or serialized model and predict
model_loaded = joblib.load('Malignant_comments_classifier_Project.pkl')

# Make predictions and view the results
predict_test = model_loaded.predict(test_features)

# Saving predicted values into a CSV file
pd.DataFrame(predict_test.toarray()).to_csv('Predicted_test_output.csv')

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153164 entries, 0 to 153163
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype  
---  --  
 0   comment_text  153164 non-null   object 
dtypes: object(1)
memory usage: 37.2 MB
None

```

```

df1 = pd.read_csv('Predicted_test_output.csv')
df1.drop("Unnamed: 0", axis=1, inplace=True)
df1.rename({'0':'malignant', '1':'highly_malignant', '2':'rude', '3':'threat', '4':'abuse', '5':'loathe'},
           axis='columns', inplace=True)
df2=df_test.copy()
df = pd.concat([df2, df1], axis=1)
df

```

	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

```
df.to_csv('test_dataset_predictions.csv', index=False)
```

## Inference

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 wordcloud 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.

## Problems faced while working in this project:

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

## Areas of improvement:

1. Could be provided with a good dataset which doesnot take more time.
2. Less time complexity
3. 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 solution is to avoid it and overcoming in a positive manner.*

---

# Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

Describe the approaches you followed, both statistical and analytical, for solving of this problem.

- Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

- 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.

- Key Metrics for success in solving problem under consideration

What were the key metrics used along with justification for using it?  
You may also include statistical metrics used if any.

- Visualizations

Mention all the plots made along with their pictures and what were the inferences and observations obtained from those. Describe them in detail.

If different platforms were used, mention that as well.

- Interpretation of the Results

Give a summary of what results were interpreted from the visualizations, preprocessing and modelling.

## CONCLUSION

- Key Findings and Conclusions of the Study

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

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

List down your learnings obtained about the power of visualization, data cleaning and various algorithms used. You can describe which algorithm works best in which situation and what challenges you faced while working on this project and how did you overcome that.

- **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.