FLIP ROBO

"A PROJECT REPORT ON MALIGNANT COMMENT CLASSIFIER"

SUBMITTED BY

HIMAJA IJJADA

ACKNOWLEDGMENT

I express my sincere gratitude to FlipRobo Technologies for giving me the opportunity to work on “**A PROJECT REPORT ON MALIGNANT COMMENT CLASSIFIER”** using machine learning algorithms. I would also like to thank FlipRobo Technologies for providing me with the requisite datasets to work with. And I would like to express my gratitude to Mr. Mohd Kashif (SME FlipRobo) and Ms. Sapna Verma (SME FlipRobo) for being of a great help in completion of the project.

Most of the concepts used to predict the Prices of flight tickets project are learned from Data Trained Institute and below documentations.

- https://scikit-learn.org/stable/
- https://seaborn.pydata.org/
- https://www.scipy.org/

CONTENTS

Introduction

Business Problem Framing

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

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

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

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to address it. The problem we sought to solve was the tagging of internet comments that are offensive towards other users, which means that insults to third parties such as celebrities will be tagged as inoffensive, but they may be 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 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 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 as ignominy. And when innocent individuals are being 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 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

Nowadays users leave numerous comments on different social networks, news portals, and forums. Some of the comments are toxic or abusive. Due to numbers of comments, it is unfeasible to manually moderate them, so most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, we performed a systematic review of the state-of-the-art in toxic comment classification using machine learning methods. First, we have investigated when and where the papers were published and their maturity level. In our analysis of every primary study we investigated: data set used, evaluation metric, used machine learning methods, classes of toxicity, and comment language.

Motivation for the Problem Undertaken

The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation 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.

Analytical problem framing

Mathematical/ Analytical Modeling of the Problem

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 will 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, we compared the results of proposed and baseline features with other machine learning algorithms. Findings of the comparison indicate the significance of the proposed features in cyberbullying detection.

Data Sources and their formats

```
# Importing the train dataset provided and viewing the top 5 columns in it
df_train = pd.read_csv('MCC_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

```
# Importing the test dataset provided and viewing the top 5 columns in it
df_test = pd.read_csv('MCC_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

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.

This project is more about exploration, feature engineering and classification that can be done on this data. 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 that can differentiate between comments and its categories.

Data Preprocessing Done

Importing all necessary libraries and packages

```
# Importing all necessary libraries and packages
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
import joblib

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import missingno
import pandas_profiling
from scipy import interp
import scikitplot as skplt
from itertools import cycle
import matplotlib.ticker as plticker

import nltk
nltk.download('stopwords', quiet=True)
nltk.download('punkt', quiet=True)
import wordcloud
from PIL import Image
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, RandomizedSearchC\
from scipy.sparse import csr_matrix

import timeit, sys
from sklearn import metrics
import tqdm.notebook as tqdm
from skmultilearn.problem_transform import BinaryRelevance
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_ma
from sklearn.metrics import roc_curve, auc, roc_auc_score, multilabel_confusion_matrix
from scikitplot.metrics import plot_roc_curve
```

We have imported all the necessary libraries/packages.

Checking the shape of the train dataset

```
#Checking the shape of the traindataset
print("We have {} Rows and {} Columns in our dataframe".format(df_train.shape[0], df_train.shape[1]))
```

```
We have 159571 Rows and 8 Columns in our dataframe
```

Checking for missing values

```
# Checking for missing values
df_train.isna().sum()
```

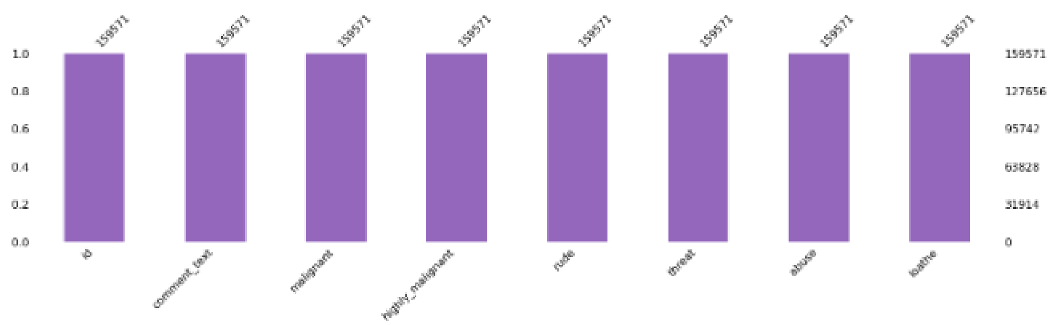
```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

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.

Visualizing for null values

```
# Visualizing for null values
missingno.bar(df_train, figsize = (25,5), color="tab:purple")
```

```
<AxesSubplot:>
```



This is to ensure that there is no missing data in the dataset using missingno.

Checking the info of the train dataset

```
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
```

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.

```python
# checking ratio of data which contains malignant comments and normal or unoffensive comments.
output_labels = df_train.columns[2:]

# counting non-zero rows i.e. Malignant Comments
malignant_comments = len(df_train[df_train[output_labels].any(axis=1)])

# counting rows containing zero i.e. Normal Comments
normal_comments = len(df_train)-malignant_comments

print(f"Total Malignant Comments: {malignant_comments} ({round(malignant_comments*100/len(df_train),2)}
print(f"Total Normal Comments: {normal_comments} ({round(normal_comments*100/len(df_train),2)}%)")
```

```
Total Malignant Comments: 16225 (10.17%)
Total Normal Comments: 143346 (89.83%)
```

Above ratio shows that our dataframe consists 10.17% of Malignant Comments and 89.83% of Normal Comments. Hence, it is clear that the dataset is imbalanced and needs to be treated accordingly during train test split of model training.

```python
# 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

Data Cleaning

```
# 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

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

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, 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"
                "ok","there's","hey","heh","hi","oh","bbq","i'm","i've","nt","can't","could","ur","re",
                "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)
```

```
New list of custom stop words are as follows:


{'that', 'ain', 'but', 't', "she'll", 'being', "hasn't", 'h', 'z', "he's", 'omg', "there's", 'too',
're', 'i', "you'd", 'its', 'is', 'such', 'for', 'some', 'no', 'you', 'with', 'won', "wasn't", 'ok',
'between', 'and', 'to', 'then', "should've", "you'll", "you're", 'mightn', 'were', 'only', 'into', 't
his', 'until', 'each', 'both', 'most', 'yolo', "won't", 'during', 'at', 'any', "mustn't", 'very',
'u', 'can', 'below', 'those', 'not', 'b', 'needn', 'ourselves', 'should', 'themselves', "doesn't", 'b
e', 'us', 'he', 'itself', 'your', 'hmm', "can't", 'them', 'have', 'don', 'been', 'once', 'maybe', 's
o', 'why', 'an', "couldn't", 'yours', 'mustn', 'couldn', 'or', 'weren', 'what', 'will', 'q', 'agains
t', 'which', 'nor', 'when', 'stfu', 'am', 'f', 'my', 'had', 'did', 'lmfao', 'c', 'they', 'hers', 'o
n', 'o', 'lmk', 'd', 'all', 'who', 'm', 'wasn', "didn't", 'ikr', 'how', 'a', "shouldn't", 'in', 'who
m', 'hey', 'does', "shan't", 'ma', 'above', 'hasn', 'p', 'll', 'bbq', "hadn't", 'over', 'has', 'thei
r', 'having', 'under', 'smh', 'yourself', 'also', 'yourselves', 'him', 'do', "isn't", "aren't", 'il
u', "i'll", 'same', "that's", "weren't", 'hadn', 'k', 'these', 'could', "i'm", "haven't", 'j', 'nvm',
'are', 'myself', 'theirs', 'doesn', 'herself', 'ours', 'there', 'here', 'shouldn', 'rofl', "needn't",
'e', 'because', 'our', 'it', 'if', 'was', 'down', "you've", 'haven', 'after', 'her', 'didn', 'doing',
"she's", "he'll", 'off', 'l', 'about', 'while', 'from', 'up', 'we', 'out', 'himself', 'again', 'y',
'lol', "mightn't", "wouldn't", 'nt', 'heh', 'few', 'where', 've', 'oh', "don't", 'ofc', 'she', 'now',
'his', 'g', "d'aww", "i've", 'as', 'ily', 'mr', 'other', 'by', 'isn', 'hi', 'before', 'more', 'than',
'just', 'v', "it's", 'w', 'umm', 'x', 's', 'shan', 'n', 'through', 'wouldn', 'r', 'further', 'me', "t
hat'll", 'ur', 'own', 'aren', 'the', 'of'}
```

Checking any 10 random rows to see the applied changes

```
# Removing stop words
df.comment_text = df.comment_text.apply(lambda x: ' '.join(word for word in x.split() if word not in st

# 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
122501	part article several articles quotes article q...	0	0	0	0	0	0	108
113291	move proposal partements france hello tariq th...	0	0	0	0	0	0	118
69481	leave article alone	0	0	0	0	0	0	28
76760	exactly right director creator casting remove ...	0	0	0	0	0	0	149
123873	dubious assertions mis citations miss citation...	0	0	0	0	0	0	381
84779	months solid	0	0	0	0	0	0	32
46514	problem looking tune weavers internet connecti...	0	0	0	0	0	0	207
90212	right speedied nonsense pages last night night...	0	0	0	0	0	0	112
20512	preceding unsigned comment added talk contribs	0	0	0	0	0	0	65
37937	needs someone write	0	0	0	0	0	0	47

Checking any 10 random rows to see the applied changes

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

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

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
156417	yes problem past make better editor lower hell...	1	0	0	0	1	0	523
157925	agre complet content review process need monit...	0	0	0	0	0	0	351
92627	hog bait involv dog submit harm kill hog pig b...	0	0	0	0	0	0	271
138425	problem forc mean second command falsifi rabbi...	1	0	0	0	0	0	754
66374	polish boy far import get polish boy sandwich ...	0	0	0	0	0	0	309
47193	check shankbon admit mix alcohol anti depress ...	0	0	0	0	0	0	686
147743	contain singl refer polit lace hearsay someon ...	0	0	0	0	0	0	99
35820	accept adopt offer id love new home	0	0	0	0	0	0	60
51456	wow delet whole thing american never understan...	0	0	0	0	0	0	151
49077	one guy seem sort administr delet contribut fi...	1	0	0	0	1	0	797

Checking the length of comment_text after cleaning and storing it in cleaned_length variable

```
# 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

checking the percentage of length cleaned

```
# 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.origi
```

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

pandas_profiling

```
pandas_profiling.ProfileReport(df)
```

```
Summarize dataset:   0%|          | 0/23 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

pandas-profiling is an open source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It generates interactive reports in web format that can be presented to any person, even if they don't know programming. It also offers report generation for the dataset with lots of features and customizations for the report generated. In short, what pandas-profiling does is save us all the work of visualizing and understanding the distribution of each variable. It generates a report with all the information easily available.

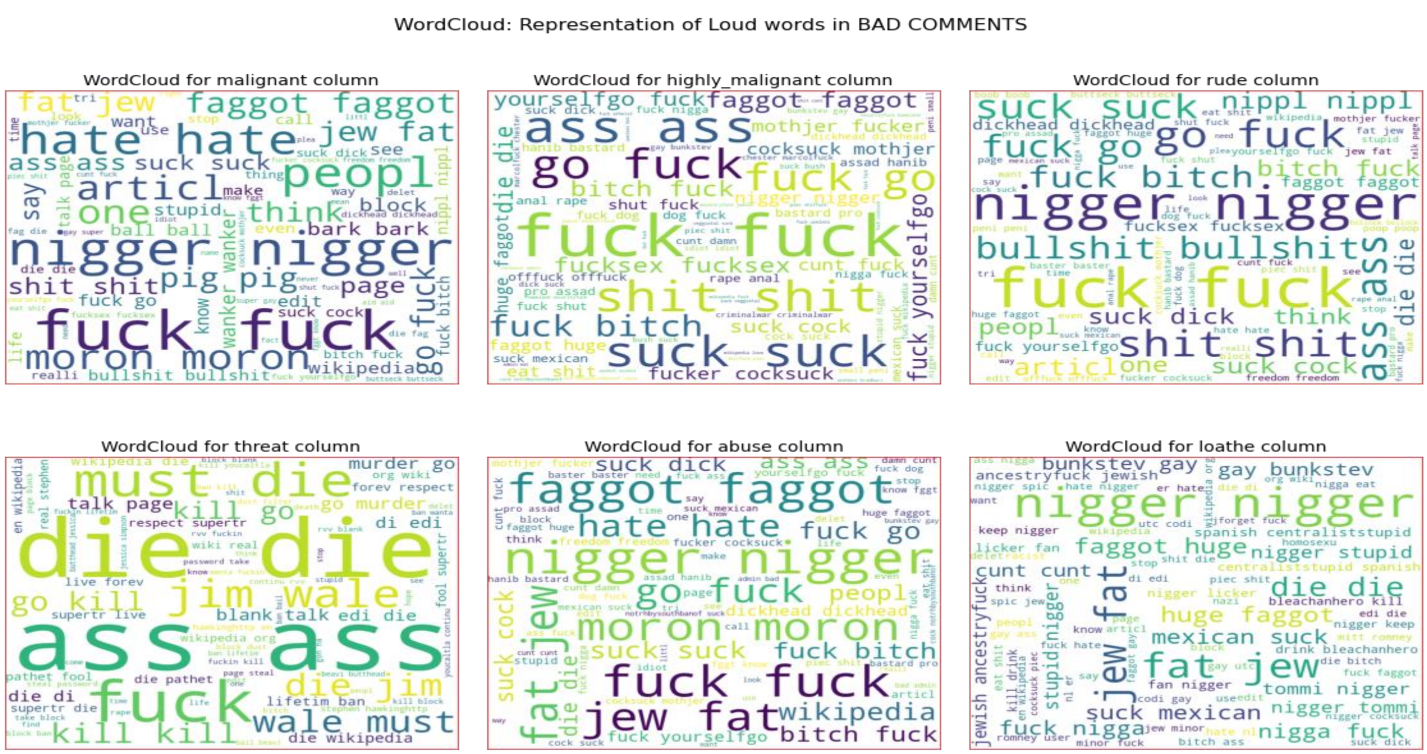
Data Inputs- Logic- Output Relationships

I have analysed the input output logic with word cloud and I have word clouded the sentenced that as 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.

Code:

```
# WordCloud: Getting sense of loud words in each of the output labels.

cols = 3
rows = len(output_labels)//cols
if len(output_labels) % cols != 0:
    rows += 1

fig = plt.figure(figsize=(16,rows*cols*1.8))
fig.subplots_adjust(top=0.8, hspace=0.3)

p=1
for i in output_labels:
    word_cloud = WordCloud(height=650, width=800,
                           background_color="white",max_words=80).generate(' '.join(df.comment_text[df[i]==1]))
    ax = fig.add_subplot(rows,cols,p)
    ax.imshow(word_cloud)
    ax.set_title(f"WordCloud for {i} column",fontsize=14)
    for spine in ax.spines.values():
        spine.set_edgecolor('r')

    ax.set_xticks([])
    ax.set_yticks([])
    p += 1

fig.suptitle("WordCloud: Representation of Loud words in BAD COMMENTS",fontsize=16)
fig.tight_layout(pad=2)
plt.show()
```

Output:



Assumptions - These are the comments that belongs to different type so which the help of word cloud we can see if there is abuse comment which type of words it contains and similar to other comments as well.

- From wordcloud of malignant comments, it is clear that it mostly consists of words like fuck, nigger, moron, hate, suck ect.

- From wordcloud of highly_malignant comments, it is clear that it mostly consists of words like ass, fuck, bitch, shit, die, suck, faggot ect.

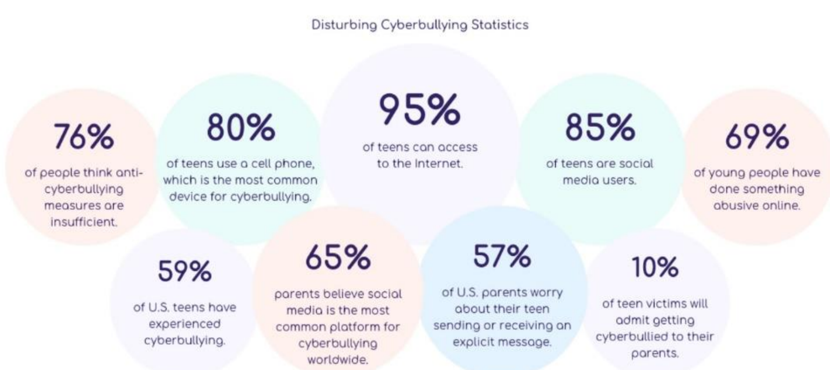
- From wordcloud of rude comments, it is clear that it mostly consists of words like nigger, ass, fuck, suck, bullshit, bitch etc.

- 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.

Cyberbullying has become a growing problem in countries around the world. Essentially, cyberbullying doesn't differ much from the type of bullying that many children have unfortunately grown accustomed to in school. The only difference is that it takes place online.



Cyberbullying is a very serious issue affecting not just the young victims, but also the victims' families, the bully, and those who witness instances of cyberbullying. However, the effect of cyberbullying can be most detrimental to the victim, of course, as they may experience a number of emotional issues that affect their social and academic performance as well as their overall mental health.

Hardware and Software Requirements and Tools Used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required:

- Processor: core i5 or above
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

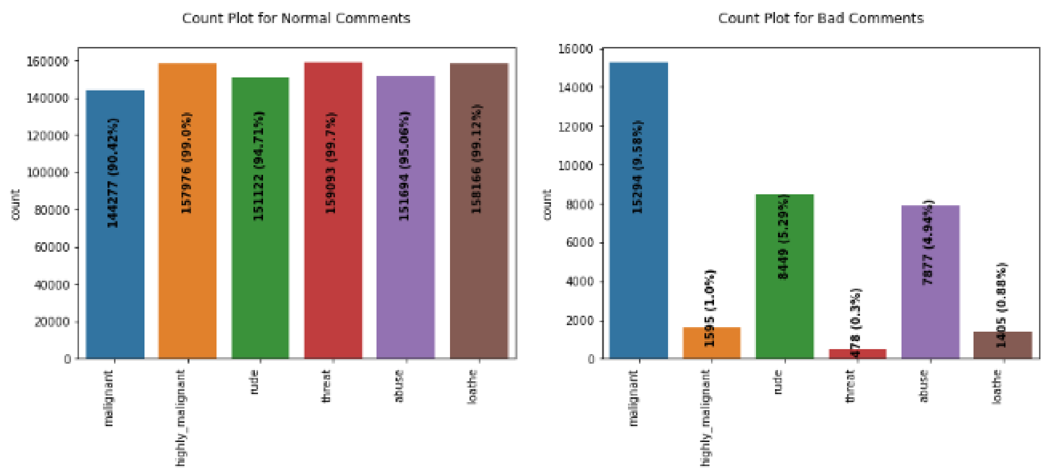
Software required:

- Distribution: Anaconda Navigator
- Programming language: Python
- Browser based language shell: Jupyter Notebook
- Word cloud: For visual display of text data
- Libraries/Packages specifically being used - Pandas, NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling, missingno, NLTK

Model/s Development and evaluation

Visualizations

```python
# comparing normal comments and bad comments using count plot

fig, ax = plt.subplots(1,2,figsize=(15,5))

for i in range(2):
    sns.countplot(data=df[output_labels][df[output_labels]==i], ax=ax[i])
    if i == 0:
        ax[i].set_title("Count Plot for Normal Comments\n")
    else:
        ax[i].set_title("Count Plot for Bad Comments\n")

    ax[i].set_xticklabels(output_labels, rotation=90, ha="right")
    p=0
    for prop in ax[i].patches:
        count = prop.get_height()
        s = f"{count} ({round(count*100/len(df),2)}%)"
        ax[i].text(p,count/2,s,rotation=90, ha="center", fontweight="bold")
        p += 1

plt.show()
```



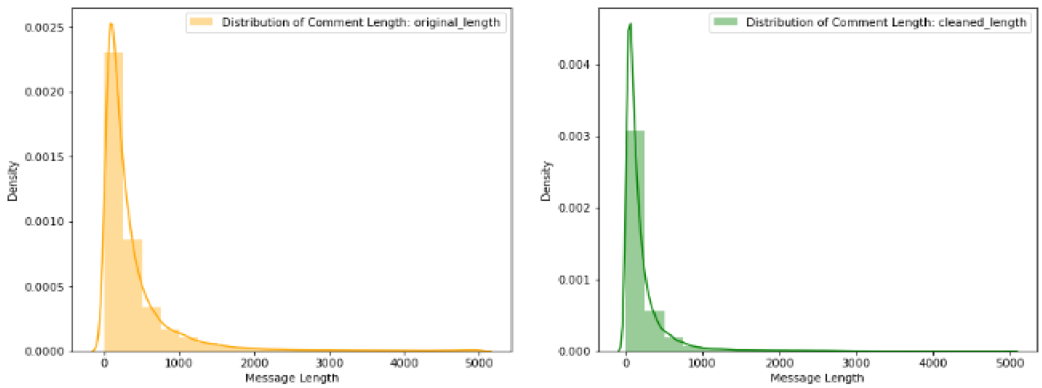
Observation:

- 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.

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

fig, ax = plt.subplots(1,2,figsize=(15,6))
j=0
colors = ['orange','green']
for i in df.columns[-2:]:
    label_text = f"Distribution of Comment Length: {i}"
    sns.distplot(df[i],ax=ax[j],bins=20,color=colors[j],label=label_text)
    ax[j].set_xlabel("Message Length")
    ax[j].legend()
    j += 1

plt.show()
```

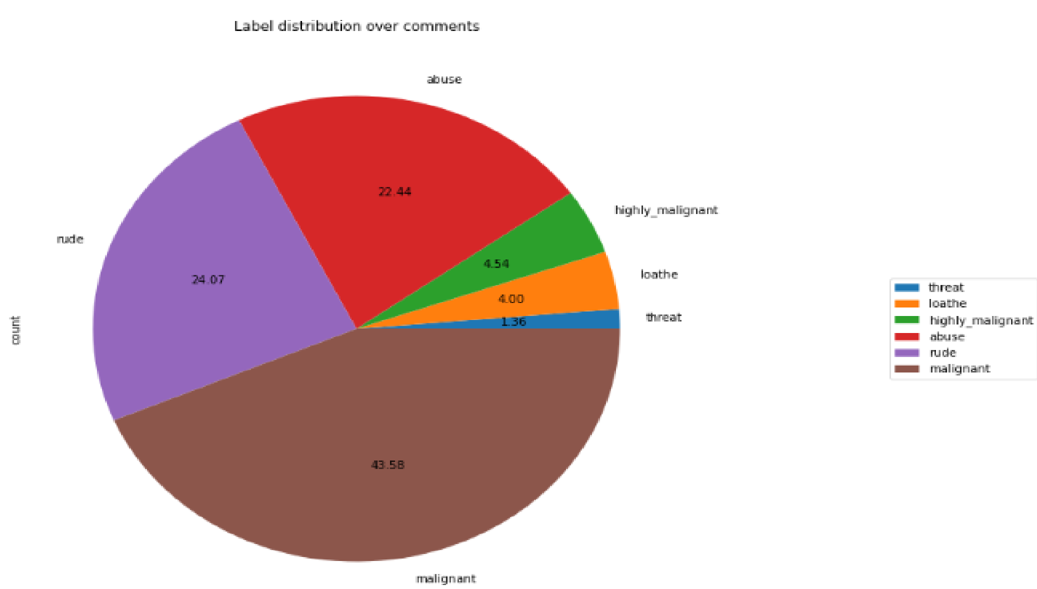


Observation:

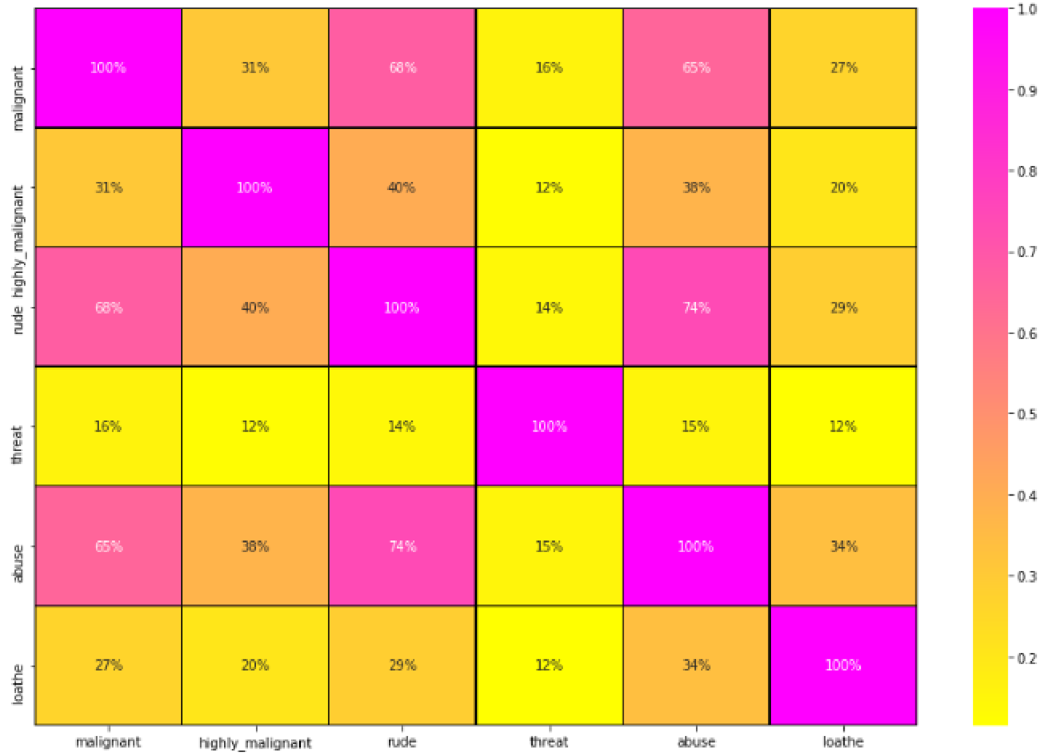
- Before cleaning comment_text column most of the comment's length lies between 0 to 1100 while after cleaning it has been reduced between 0 to 900.

```python
# Visualizing the label distribution of comments using pie chart

comments_labels = ['malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe']
df_distribution = df_train[comments_labels].sum()\
                    .to_frame()\
                    .rename(columns={0: 'count'})\
                    .sort_values('count')

df_distribution.plot.pie(y = 'count', title = 'Label distribution over comments', autopct='%.2f', figsi
                    .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

```
<matplotlib.legend.Legend at 0x1ca631c6580>
```



```
# Plotting heatmap for visualizing the correlation

plt.figure(figsize=(15, 10))
corr = df_train.corr() # corr() function provides the correlation value of each column
sns.heatmap(corr, linewidth=0.5, linecolor='black', fmt='.0%', cmap='spring_r', annot=True)
plt.show()
```



Identification of possible problem-solving approaches (methods)

```
# 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)
```

Testing of Identified Approaches (Algorithms)

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

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

    return models
```

```
==========================================================================================
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 [32.70419479999998 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 [6.7433986000000345 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 [43.60085359999999 sec.]
========================================================================================

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

        Hamming Loss  : 0.02030636789467021
        Accuracy Score: 0.9122673756171668
              precision    recall  f1-score   support

           0       0.86      0.63      0.73      1281
           1       0.48      0.07      0.12       150
           2       0.88      0.72      0.79       724
           3       0.00      0.00      0.00        44
           4       0.73      0.52      0.61       650
           5       0.92      0.11      0.20       109

   micro avg       0.83      0.57      0.68      2958
   macro avg       0.65      0.34      0.41      2958
weighted avg       0.81      0.57      0.66      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [1566.6139196000001 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 [8.274850300000026 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 [985.1354943000001 sec.]
========================================================================================

========================================================================================
Current Model in Progress: K Nearest Neighbors Classifier
========================================================================================
Training:  BinaryRelevance(classifier=KNeighborsClassifier(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.03201671097607292
        Accuracy Score: 0.8950246866691987
              precision    recall  f1-score   support

           0       0.72      0.24      0.36      1281
           1       0.37      0.15      0.21       150
           2       0.83      0.28      0.41       724
           3       0.00      0.00      0.00        44
           4       0.69      0.25      0.36       650
           5       0.65      0.16      0.25       109

   micro avg       0.72      0.24      0.36      2958
```

```
   macro avg       0.54      0.18      0.27      2958
weighted avg       0.71      0.24      0.36      2958
 samples avg       0.02      0.02      0.02      2958
Completed in [427.1659903999998 sec.]
=============================================================================

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

        Hamming Loss  : 0.026307127484491707
        Accuracy Score: 0.8858336498290923
              precision    recall  f1-score   support

           0       0.69      0.69      0.69      1281
           1       0.29      0.25      0.27       150
           2       0.77      0.75      0.76       724
           3       0.23      0.11      0.15        44
           4       0.57      0.60      0.59       650
           5       0.41      0.34      0.37       109

   micro avg       0.65      0.64      0.65      2958
   macro avg       0.49      0.46      0.47      2958
weighted avg       0.64      0.64      0.64      2958
 samples avg       0.06      0.06      0.06      2958
Completed in [1757.5026318999999 sec.]
=============================================================================

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

        Hamming Loss  : 0.020091150778579567
        Accuracy Score: 0.913482719331561
              precision    recall  f1-score   support

           0       0.86      0.64      0.74      1281
           1       0.49      0.22      0.30       150
           2       0.90      0.65      0.75       724
           3       0.44      0.09      0.15        44
           4       0.77      0.53      0.63       650
           5       0.79      0.25      0.38       109
```

```
   micro avg       0.84      0.58      0.68      2958
   macro avg       0.71      0.40      0.49      2958
weighted avg       0.82      0.58      0.67      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [302.3315358 sec.]
==========================================================================================
```

Observation : From the above model comparision 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.

Run and Evaluate selected models

```
# 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=4
GSCV.fit(x_train,y_train)
GSCV.best_params_
```

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

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.

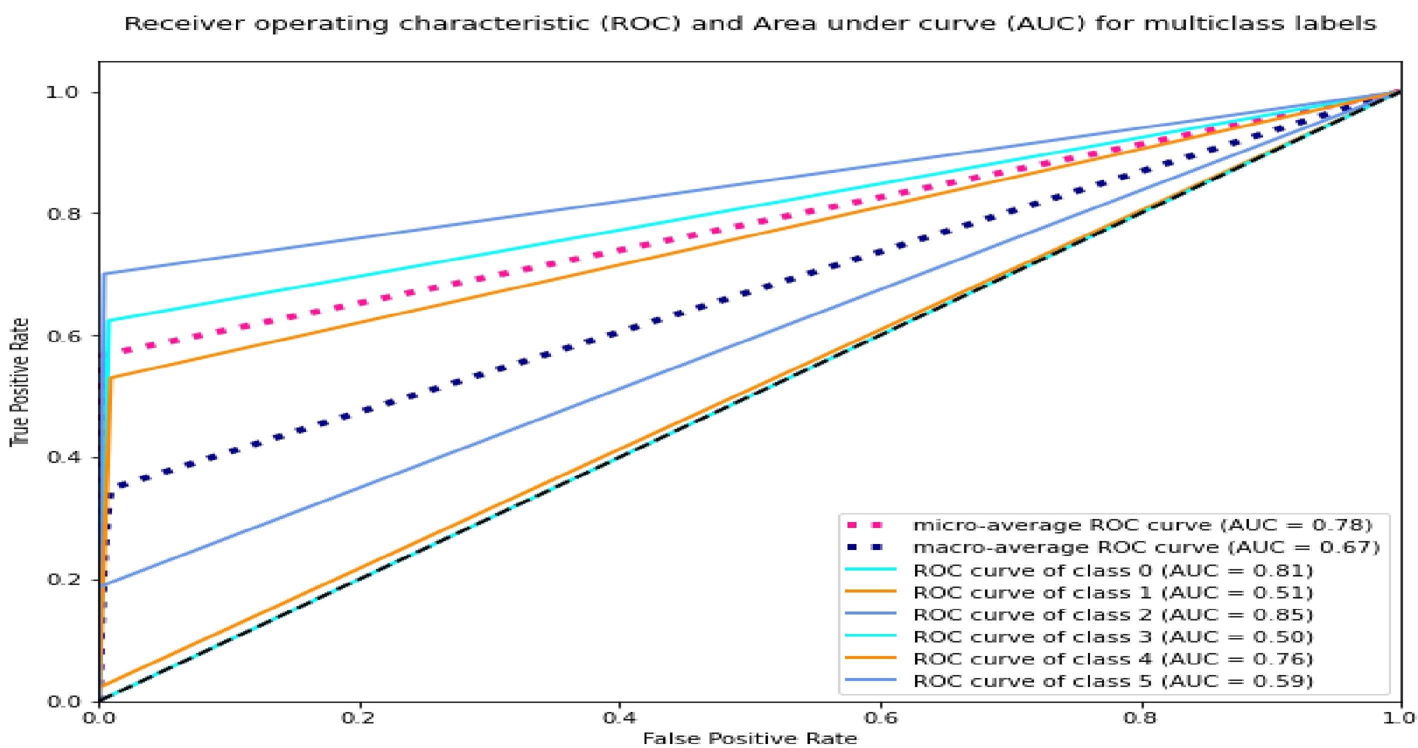
```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge', multi_class='ovr', penalty='l2', random_state
Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)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

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

```
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 = {0:0.2f})".format(roc_auc["micro"]),
    color="deeppink",
    linestyle=":",
    linewidth=4,
)

plt.plot(
    fpr["macro"],
    tpr["macro"],
    label="macro-average ROC curve (AUC = {0:0.2f})".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()
```



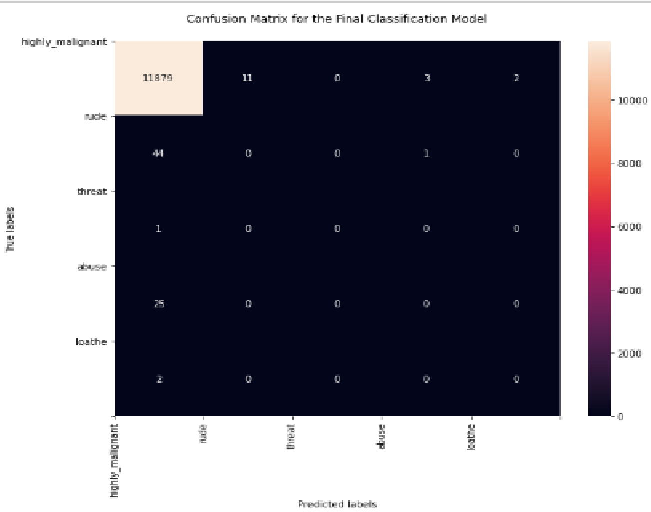
I have generated the ROC Curve for my final model and it shows separate curve for every class present in our multi label target variable along with it's AUC values.

Confusion Matrix for Final Model

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

```
Confusion matrix:

 [[[10710    83]
  [  442   733]]

 [[11832     1]
  [  132     3]]

 [[11263    47]
  [  197   461]]

 [[11930     0]
  [   38     0]]

 [[11266   106]
  [  280   316]]

 [[11869     3]
  [   78    18]]]
```

```
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, ftm='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()
```



With the help of above confusion matrix I am able to understand the number of times I got the correct outputs and the number of times my final model missed to provide the correct prediction (depicting in the black boxes).

Saving Model

```
# 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'))
```

Preprocessing for test dataset

The following preprocessing pipeline is required to perform model prediction:

- Use the test dataset
- Remove null values if any
- 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 best model
- Predict values and create a new CSV file

```python
# 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(regexp_tokenize(x,"[a-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()

# Load saved or serialized model and predict
model_loaded = joblib.load('Malignant_comments_classifier.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
```

Predicted_test_output

```
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

Saving the output into csv

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

Key Metrics for success in solving problem under consideration

1. **Accuracy**

Accuracy can also be defined as the ratio of the number of correctly classified cases to the total of cases under evaluation. The best value of accuracy is 1 and the worst value is 0.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} = \frac{\text{Number of correctly classified cases}}{\text{Total number of cases under evaluation}}$$

In python, the following code calculates the accuracy of the machine learning model.

Accuracy = metrics.accuracy_score(y_test, preds)
Accuracy

2. **Precision**

Precision can be defined with respect to either of the classes. The precision of negative class is intuitively the ability of the classifier not to label as positive a sample that is negative. The precision of positive class is intuitively the ability of the classifier not to label as negative a sample that is positive. The best value of precision is 1 and the worst value is 0.

$$Precision\ (positive\ class) = \frac{TP}{TP+FP} = \frac{True\ Positive}{Number\ of\ cases\ predicted\ as\ positive}$$

$$Precision\ (negative\ class) = \frac{TN}{TN+FN} = \frac{True\ Negative}{Number\ of\ cases\ predicted\ as\ negative}$$

3. **Recall**

Recall can also be defined with respect to either of the classes. Recall of positive class is also termed sensitivity and is defined as the ratio of the True Positive to the number of actual positive cases. It can intuitively be expressed as the ability of the classifier to capture all the positive cases. It is also called the True Positive Rate (TPR).

4. **F1-score**

F1-score is considered one of the best metrics for classification models regardless of class imbalance. F1-score is the weighted average of recall and precision of the respective class. Its best value is 1 and the worst value is 0.

$$F1-score = \frac{TN}{TN+FP} = \frac{2 * Precision * Recall}{Precision + Recall}$$

In python, F1-score can be determined for a classification model using

```
f1_positive = metrics.f1_score(y_test, preds, pos_label=1)

f1_negative = metrics.f1_score(y_test, preds, pos_label=0)

f1_positive, f1_negative
```

It gives an output of (0.937, 0.966)

Accuracy, Precision, Recall, and F1-score can altogether be calculated using the method classification_report in python

5. **ROC and AUC score**

ROC is the short form of Receiver Operating Curve, which helps determine the optimum threshold value for classification. The threshold value is the floating-point value between two classes forming a boundary between those two classes. Here in our model, any predicted output above the threshold is classified as class 1 and below it is classified as class 0.

ROC is realized by visualizing it in a plot. The area under ROC, famously known as AUC is used as a metric to evaluate the classification model. ROC is drawn by taking false positive rate in the x-axis and true positive rate in the y-axis. The best value of AUC is 1 and the worst value is 0. However, AUC of 0.5 is generally considered the bottom reference of a classification model.

6. **Hamming Loss**

Hamming loss is the fraction of targets that are misclassified. The best value of the hamming loss is 0 and the worst value is 1. It can be calculated as

hamming_loss = metrics.hamming_loss(y_test, preds)

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 unparliamentarily language. And most of these sentences have more stop words and are being quite long. As discussed before few motivated disrespectful crowds use these foul languages in the online forum to bully the people around and to stop them from doing these things that they are not supposed to do. Our study helps the online forums and social media to induce a ban to profanity or usage of profanity over these forums.

Learning Outcomes of the Study in respect of Data Science

I found that the dataset was quite interesting to handle. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in property research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove unrealistic values and stopwords.

Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stopwords. We were also able to learn to convert strings into vectors through hash vectorizer. In this project we applied different evaluation metrics like log loss, hamming loss besides accuracy. This study is an exploratory attempt to use four machine learning algorithms in estimating malignant comments, and then compare their results.

To conclude, the application of machine learning in malignant classification is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to crediting institutes, and presenting an alternative approach to the valuation of malignance. We all need to be aware of social sense and use the relatively suitable words which does not demean or degrade the other person or entity and also avoid using abusive, vulgar and mean words in social media. It can cause many problems which could affect the lives of people around us. Try to be polite, calm, empathetic and composed while handling stress and negativity and one of the best solutions is to avoid it and overcoming in a positive manner. Criticism can be given in a constructive way unless it does not hurt other's feelings.

Limitations of this work and Scope for Future Work

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

Areas of improvement:

- Could be provided with a better dataset
- Less time complexity
- Providing a proper balanced dataset with less errors.