FLIP ROBO

MALIGNANT COMMENTS CLASSIFIER PROJECT REPORT

Submitted by:

Chris Chhotai

ACKNOWLEDGMENT

I would like to express my heartfelt gratitude to Ms. Swati Mahaseth, my SME (Subject Matter Expert), as well as Flip Robo Technologies, for allowing me to work on this project on Malignant Comments Classification and for assisting me in conducting extensive research that allowed me to learn a lot of new things, particularly about the Natural Language Processing and Natural Language Toolkit parts.

In addition, I used a few outside resources to help me finish this job. I made sure to learn from the samples and adjust things to fit my project's needs. The following is a list of all the external resources that were used to create this project:

1) https://www.google.com/

2) https://www.youtube.com/

3) https://scikit-learn.org/stable/user_guide.html

4) https://github.com/

5) https://www.kaggle.com/

6) https://medium.com/

7) https://towardsdatascience.com/

8) https://www.analyticsvidhya.com/

INTRODUCTION

- **Business Problem Framing:**

  People can now express themselves widely online because to the advent of social media. However, this has led in the growth of violence and hatred, making online environments unappealing to users. Despite the fact that academics have discovered that hate is a problem across numerous platforms, there are no models for detecting online hate.

  Online hatred has been identified as a big problem on online social media platforms, and has been described as abusive language, hostility, cyberbullying, hatefulness, and many other things. The most common venues for such toxic behaviour are social media platforms.

  On numerous social media sites, there has been a significant increase in incidences of cyberbullying and trolls. Many celebrities and influencers face blowback from the public and are subjected to nasty and disrespectful remarks. This can have a negative impact on anyone, resulting in sadness, mental disease, self-hatred, and suicide thoughts.

  Comments on the internet are hotbeds of hatred and venom. Machine learning can be used to combat online anonymity, which has created a new outlet for hostility and hate speech. The issue we were attempting to address was the tagging of internet remarks that were hostile to other users. This means that insults directed towards third parties, such as celebrities, will be classified as non-offensive, whereas "u are an idiot" will be plainly offensive.

Our goal is to create a prototype of an online hate and abuse comment classifier that can be used to classify hate and offensive comments in order to prevent the spread of 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 wilfully 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:**

The purpose of the literature review is to:

1. Determine which foul words or expressions are being utilised.

2. Stop people from using these derogatory terms in public forums on the internet.

To address this issue, we are now developing a model based on our machine language technique that identifies all foul language and foul phrases, with the goal of preventing these mobs from using foul language in online communities or even blocking them from using foul language altogether.

I tested nine different classification algorithms and selected the best based on performance indicators. I then chose one algorithm and created a model using that algorithm.

Comments on the internet are hotbeds of hatred and venom. Machine learning can be used to combat online anonymity, which has created a new outlet for hostility and hate speech. The issue we were attempting to address was the tagging of internet remarks that were hostile to other users.

Our goal is to create a prototype of an online hate and abuse comment classifier that can be used to classify hate and offensive comments in order to prevent the spread of hatred and cyberbullying.

- **Motivation for the Problem Undertaken:**

One of the first lessons we learn as kids is that the louder we scream and the larger the tantrum, the more we get our way. Learning to utilise language and reasoning abilities to communicate our opinions and respectfully disagree with others, using evidence and persuasiveness to try to persuade others to our way of thinking, is an important part of maturing into an adult and productive member of society.

Social media is reverting us to the animalistic tantrums, schoolyard taunts, and unrestrained bullying that define youth, creating a dystopia in which even renowned academics and dispassionate journalists transform from Dr. Jekyll to raving Mr. Hydes, raising the

critical question of whether social media should simply enact a blanket ban on profanity and name calling. Actually, a ban on these profanities should be implemented, and with that as inspiration, I began my project to identify malicious remarks on social media and in online public forums.

Social networking platforms have given us more options than ever before, and their benefits are evident, thanks to widespread use and popularity of online social networks. People may be embarrassed, insulted, bullied, and harassed by anonymous users, strangers, or peers, notwithstanding the benefits. In this paper, we use a pointwise mutual information technique to offer a cyberbullying detection system that generates features from online content. We built a supervised machine learning approach for cyberbullying detection and multi-class severity categorization based on these features. 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 behaviour and its severity in online social networks.

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem:**

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

In [2]:
```
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

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

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 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_matrix
from sklearn.metrics import roc_curve, auc, roc_auc_score, multilabel_confusion_matrix
```

We were given two datasets to work with in this project: train and test CSV files. Using NLP and the train dataset, I will create a machine learning model. We'll generate predictions for our test dataset using this model.

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 used different machine learning techniques to compare the results of proposed and baseline features. The results of the comparison show that the proposed features are important in detecting cyberbullying.

- **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.

This project focuses on the data's exploration, feature engineering, and classification capabilities. We can do a lot of data exploration and derive some interesting characteristics from the comments text column because the data set is large and includes numerous categories of comments. You must create a model that can distinguish between comments and their categories.

- **Data Preprocessing Done:**

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

1. Load 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

- **Data Inputs- Logic- Output Relationships:**

I used a word cloud to analyse the input output logic, and I word clouded the sentences that were identified as foul language in each category. A tag/word cloud is a unique visual representation of text data that is frequently used to display keyword metadata on websites or to view free-form text. It's a picture made from of words from a specific book or topic, with the size of each word indicating its frequency or importance.

Code:

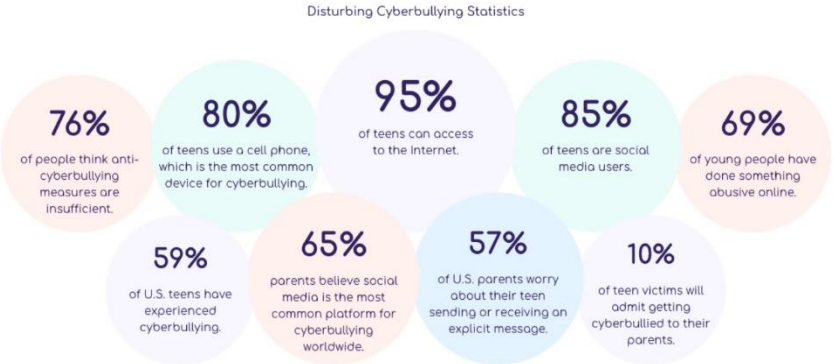
In [20]:
```
# WordCloud: Getting sense of loud words in each of the output labels.

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:

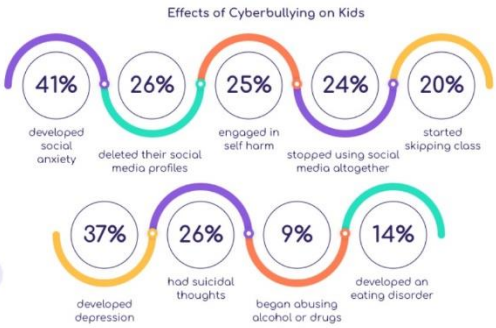
These are the comments that belongs to different type so which the help of word cloud we can see if there is abuse comment which type of words it contains and similar to other comments as well.

- **<u>State the set of assumptions (if any) related to the problem under consideration:</u>**

  Cyberbullying is becoming more prevalent in countries all around the world. In essence, cyberbullying is very similar to the type of bullying that many children have become accustomed to at school. The only difference is that it is conducted entirely online.



Cyberbullying is a severe problem that affects not only the young victims, but also their families, the bully, and anyone who witness cyberbullying. Cyberbullying, on the other hand, can have the greatest negative impact on the victim, as they may experience a variety of emotional disorders that influence their social and academic performance, as well as their overall mental health.



- **<u>Hardware and Software Requirements and Tools Used:</u>**

Hardware technology being used.

RAM : 16 GB

CPU : 11th Gen Intel(R) Core (TM) i7-11390H @ 3.40GHz 2.92 GHz

Software technology being used.

Programming language : Python

Distribution : Anaconda Navigator

Browser based language shell : Jupyter Notebook

Libraries/Packages specifically being used.

Pandas, NumPy, matplotlib, seaborn, scikit-learn, NLTK

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods):**

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

  Code:

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

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

  Code:

```
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 proceeded to undertake a number of data cleaning and transformation operations. I've created a new column to keep track of 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
```

I removed the "id" column because it was no longer needed, and I transformed all of the text data in our comment text column to lowercase for easy reading.

```
# 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()
```

The process of reducing a word to its word stem, which affixes to suffixes and prefixes or to the roots of words known as a lemma, is known as stemming. Natural language understanding (NLU) and natural language processing (NLP) both benefit from stemming (NLP.

```
# 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","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)
```

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

```
# Removing stop words
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)
```

```
# 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)
```

```
# 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)
```

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

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

- **Testing of Identified Approaches (Algorithms):**

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) K Nearest Neighbors Classifier
8) Decision Tree Classifier
9) Bagging Classifier

- **Run and Evaluate selected models:**

For the development of our Classification Machine Learning models, I constructed a classification function that incorporated the assessment metrics details.

```
# 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=========================================================================================\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
```

Code:

```
# Preparing the list of models for classification purpose
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()},
          "K Nearest Neighbors Classifier": {"name": KNeighborsClassifier()},
          "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:


```
================================================================================
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 [27.41599629999999 sec.]
================================================================================
```

Observation:

With an Accuracy Score of 91.35586783137106 percent and a Hamming Loss of 1.9977212305355107 percent, it is obvious that Linear Support Vector Classifier outperforms the other classification models. As a result, for the next step in the Hyperparameter tuning process, I'm going to employ a Linear Support Vector Classifier. I'll do my best to improve the accuracy score of our final categorization machine learning model using the hyperparameter tuning method.

- **Key Metrics for success in solving problem under consideration:**

Hyperparameter Tuning:

```
# Choosing Linear Support Vector Classifier model

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

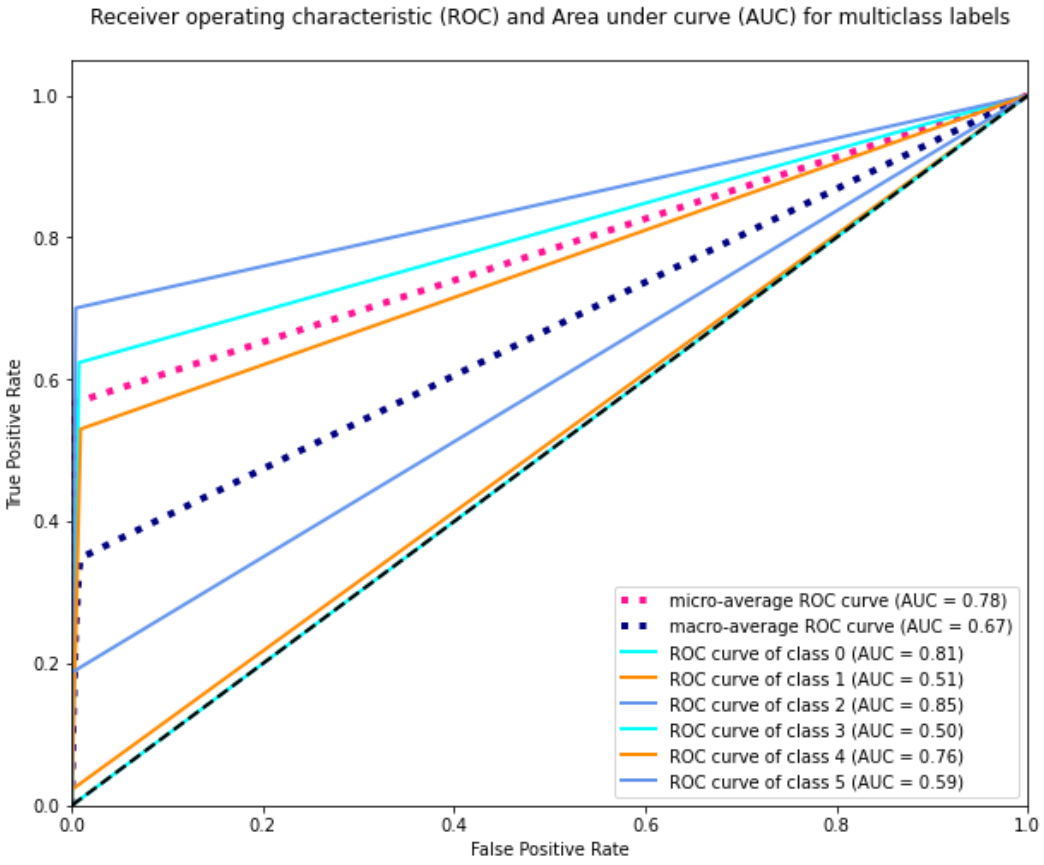
```
{'estimator__loss': 'hinge',
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2',
 'estimator__random_state': 42}
```

- **Final Classification Model details:**

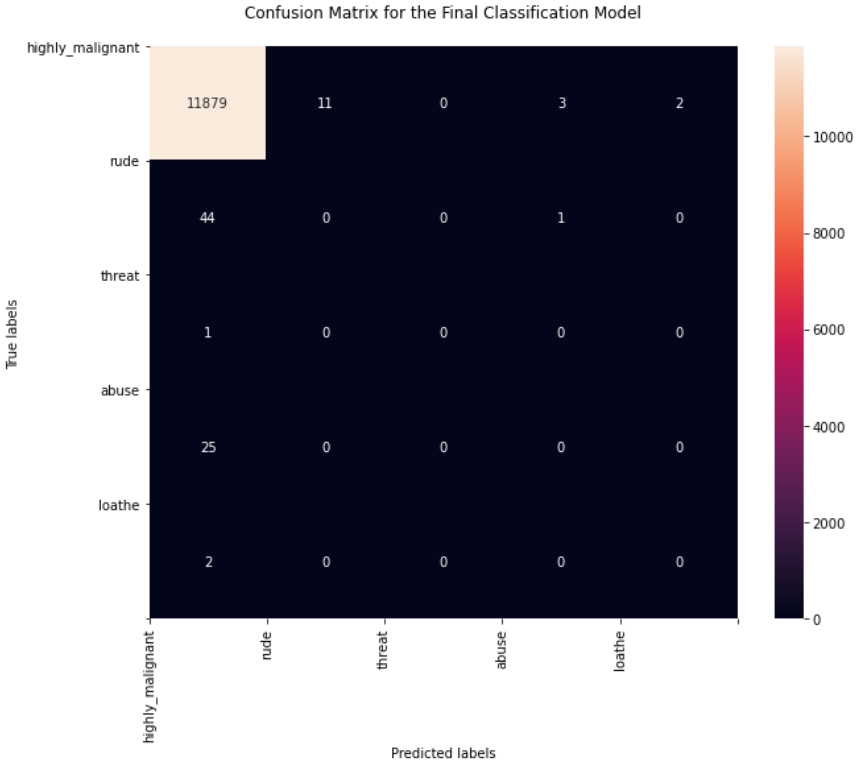
```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge', multi_class='ovr', penalty='l2', random_state=42))
Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)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



- **Confusion Matrix for Final Model:**



- **Saving the best 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'))
```

- **Final predicted dataframe:**

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

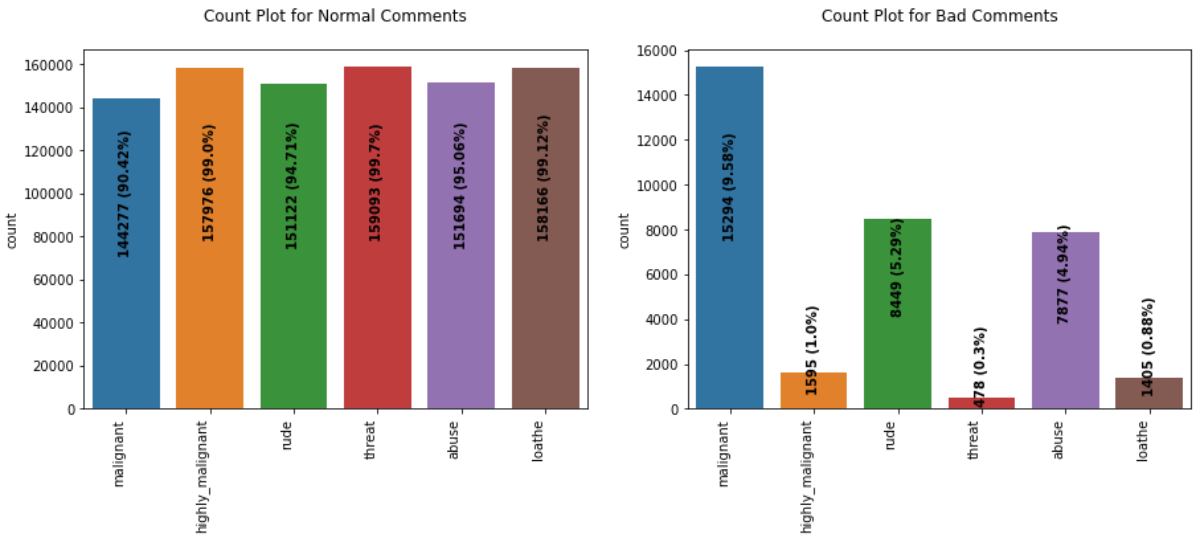
153164 rows × 7 columns

- **Visualizations:**

Code:

```
# 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()
```

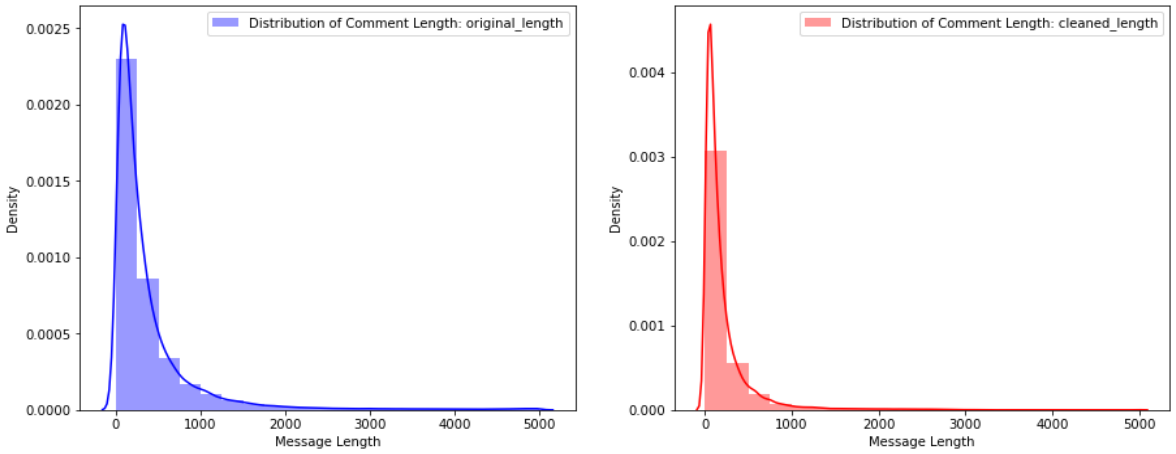
Output:



Code:

```
# 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()
```

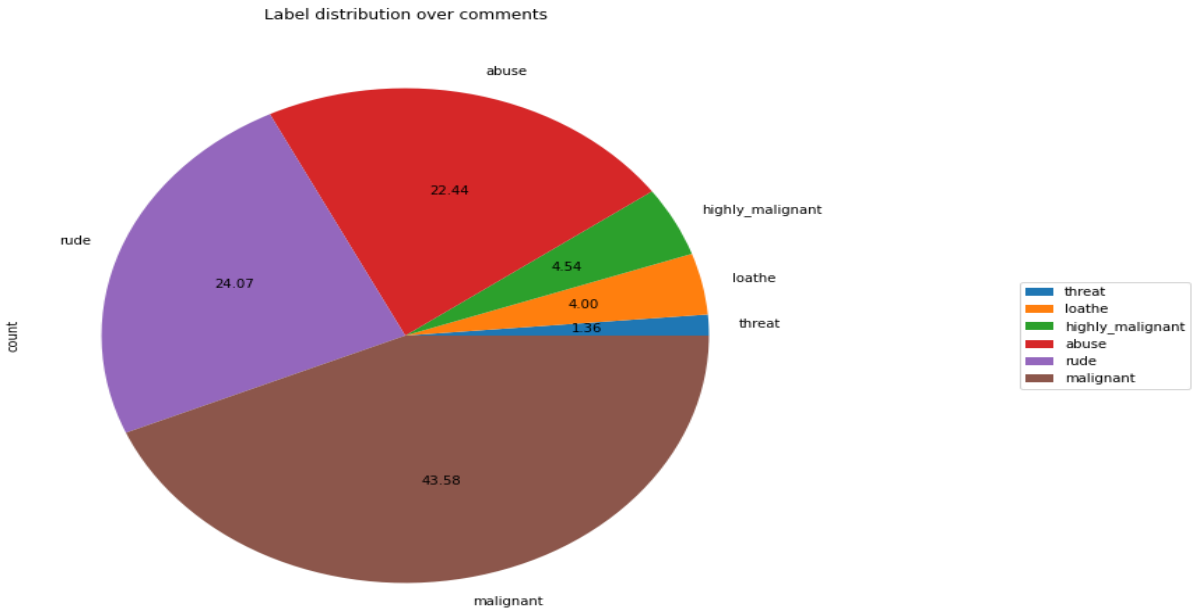
Output:



Code:

```
# 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', figsize = (15, 10))\
                            .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

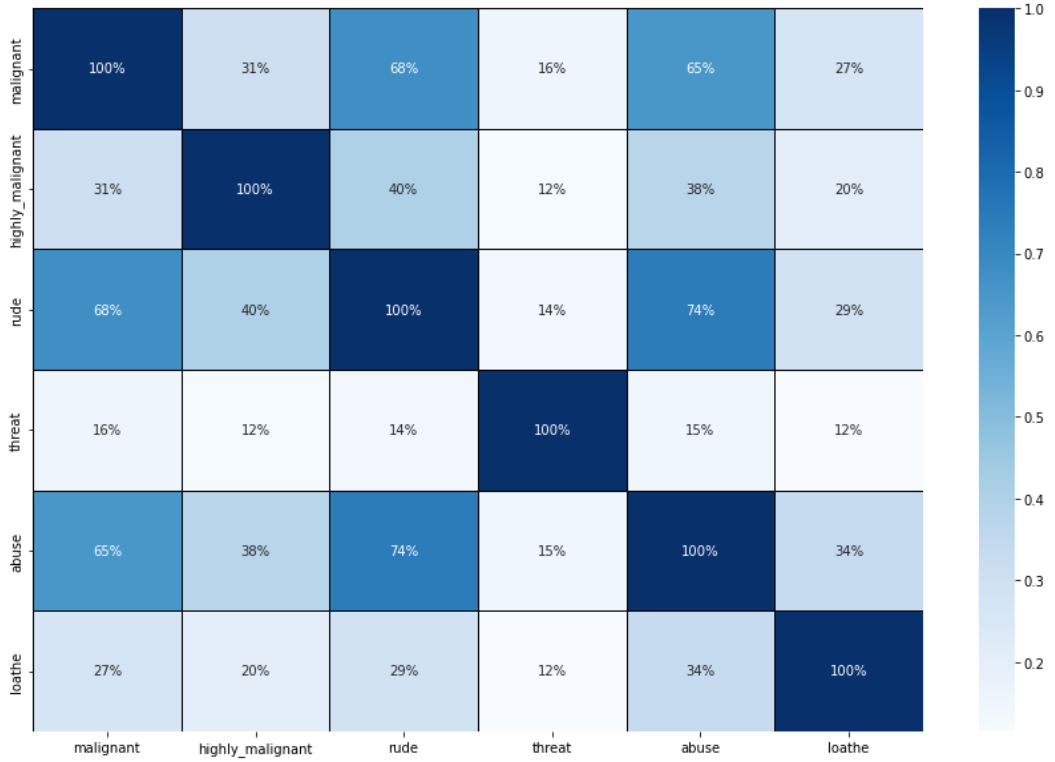
Output:



Code:

```
# 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='YlGn_r', annot=True)
plt.show()
```

Output:



Data Preparation steps:

```
# 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)
```

- **Interpretation of the Results:**

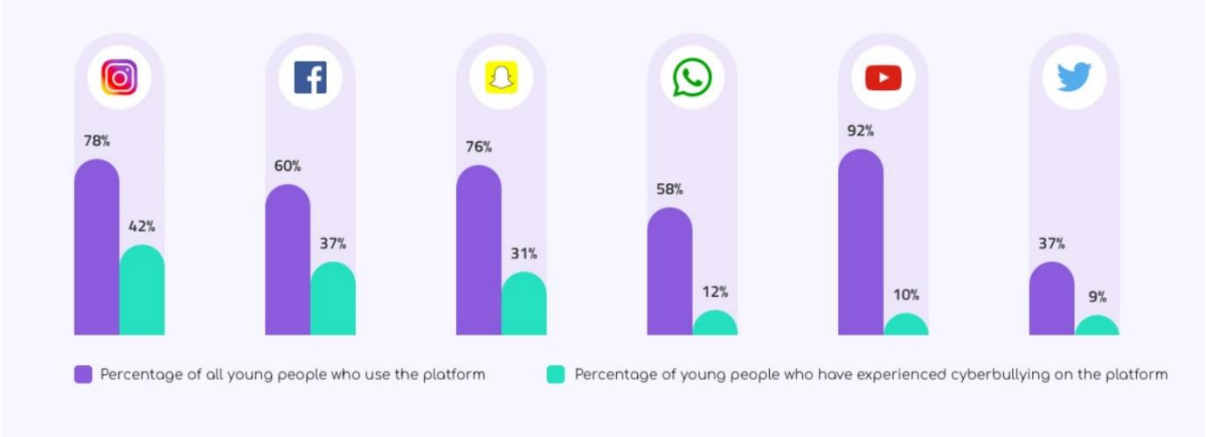
Starting with univariate analysis and using a count plot, it was discovered that the dataset is unbalanced, with more entries for normal comments than for poor remarks (including malignant, highly malignant, rude, threat, abuse and loathe). Furthermore, using a distribution plot for comment length, it was discovered that after cleaning, the majority of comments lengths reduce from 0-1100 to 0-900. Moving on to the word cloud, it was discovered that malignant comments contain words such as fuck, nigger, moron, hatred, suck, and so on. Words like ass, fuck, bitch, shit, die, suck, faggot, and others appear frequently in highly malignant comments. Words like nigger, ass, fuck, suck, crap, bitch, and others

are used in nasty statements. 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

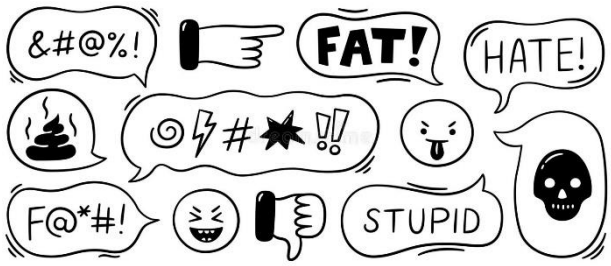
- **Key Findings and Conclusions of the Study:**

The survey discovered that just a small percentage of online users use unparliamentary language. And majority of these phrases contain a lot of stop words and are long. As previously said, a few motivated rude mobs use harsh language in internet forums to bully individuals and prevent them from doing what they are not supposed to do. Our research aids online forums and social media in enforcing a prohibition on swearing or the use of profanity on these platforms.



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

We learned several natural language processing techniques through this research, such as lemmatization, stemming, and stopword elimination. Through the hash vectorizer, we were also able to learn how to turn strings into vectors. We used a variety of evaluation methods and metrics like log loss, hamming loss besides accuracy.



My project's conclusion is that we should use good, courteous language in social media and avoid using abusive, vulgar, and derogatory terms. It has the potential to produce a slew of issues that will have an impact on our life. When dealing with tension and negativity, try to remain polite, cool, and composed, and one of the finest methods is to ignore it and overcoming obstacles in a positive way.

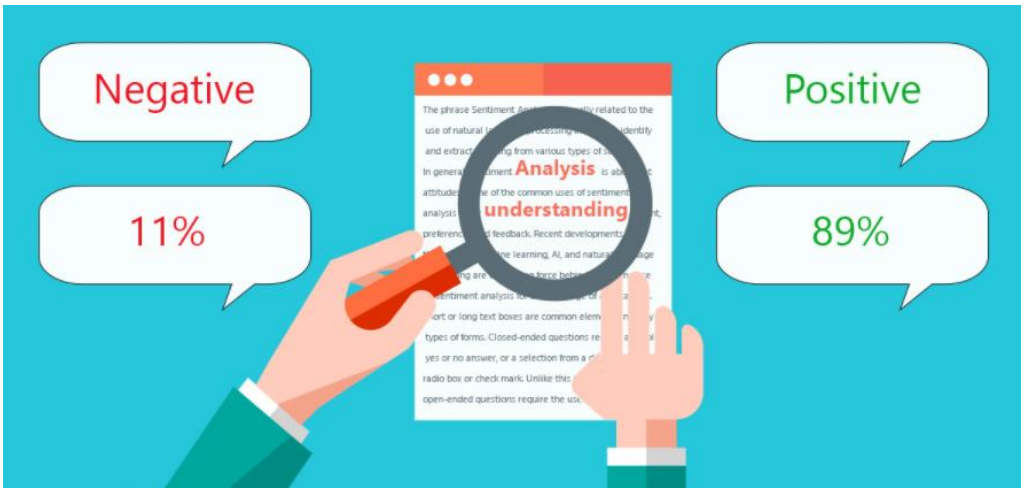
- **Limitations of this work and Scope for Future Work**

Problems faced while working in this project:

- Since it took more than 2 hours, extra computational power was required.
- The dataset is unbalanced, and the comment texts are poor.
- Because time was consumed more, good parameters could not be acquired through hyperparameter tuning.

Areas of improvement:

- Could be given a good dataset that doesn't take a lot of time.
- Less time complexity
- Delivering a well-balanced dataset with fewer errors.



Thank You.