FLIP ROBO

MALIGNANT COMMENTS CLASSIFICATION PROJECT

Submitted by:

G. PREMSAGAR

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to my SME (Subject Matter Expert) Khushboo Garg, 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 components.

In addition, I used a few other resources to assist me finish my job. I made sure to learn from the samples and adjust things to fit my project's needs. The following are all of the external resources that were utilised 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

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

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

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

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but “u are an idiot” is clearly offensive.

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

- ## Conceptual Background of the Domain Problem

Online platforms and social media have evolved into places where individuals may freely discuss their beliefs without regard for race, and where people can share their thoughts and ideas with a large group of people.

Through the creation of virtual networks and communities, social media is a computer-based technology that allows the exchange of ideas, opinions, and information. Social media is Internet-based by design, allowing people to share material quickly via electronic means. Personal information, documents, movies, and images are all included in the content. Users interact with social media using web-based software or applications on a computer, tablet, or smartphone.

While social media is widely used in the United States and Europe, Asian nations such as India are at the top of the list. Social media is used by about 3.8 billion people.

Some people or a motivated mob on this massive internet platform or online community wilfully abuse others to prevent them from sharing their thoughts in a proper manner. They use filthy language to intimidate others, which is considered a form of ignominy in civilised society. When innocent people are intimidated by these mobs, they remain mute without saying anything. As a result, the disgusting mob's goal is excellently realised.

To address this issue, we are now developing a model that recognises all foul language and foul terms, with the goal of preventing these mobs from using foul language in online communities or perhaps blocking them from using foul language altogether.

Review of Literature

The goal of the literature study is to:

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

2. Discourage individuals from using derogatory language in public forums on the internet.

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

I tested nine different classification algorithms and selected the best based on performance indicators. I then selected one approach and built a model in that algorithm.

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

Our objective is to create a prototype of an online hate and abuse comment classifier that can be used to categorise and manage hate and offensive remarks in order to prevent the spread of hatred and cyberbullying.

Motivation for the Problem Undertaken

One of the first things 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 convey our opinions and politely 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.

Even respected academics and impartial journalists are transforming from Dr. Jekyll into raging Mr. Hydes, raising the vital question of whether social media should simply adopt a blanket ban on profanity and name calling. Actually, a ban on these profanities should be enacted, and with that as inspiration, I began my project to detect the malicious remarks on social media and in online public forums. Social networking platforms have given us more options than ever before, and their advantages are evident, thanks to widespread use and popularity of online social networks.

People may be embarrassed, ridiculed, bullied, and tormented by anonymous users, strangers, or peers, notwithstanding the advantages. In this paper, we use a pointwise mutual information method to offer a cyberbullying detection system that generates features from online material. We built a supervised machine learning approach for cyberbullying detection and multi-class severity categorization based on these characteristics. Experiments in a multi-class context using our suggested framework have yielded encouraging results in terms of classifier accuracy and fmeasure metrics. These findings suggest that our suggested framework is a viable option for detecting cyberbullying in online social networks and its severity.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

The following are the libraries and dependencies that were imported for this project:

```
# Importing libraries and necessary libraries

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)
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
from scikitplot.metrics import plot_roc_curve

import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")

import joblib
```

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.

Multiple classification machine learning models will be required.

All data pre-processing processes using NLP must be completed before model development. After experimenting with several classification models and hyper parameters, the optimal model will be chosen.

Will need to follow the entire data science life cycle, which includes procedures such as -

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 outcomes of suggested and baseline characteristics. The results of the comparison show that the recommended characteristics are important in detecting cyberbullying.

• Data Sources and their formats

The data set includes a training set with roughly 1,59,000 samples and a test set with nearly 1,53,000 samples. 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse', and 'Loathe' are all fields found in all data samples. The label might be 0 or 1, with 0 indicating a NO and 1 indicating a YES. There are a number of comments with several labels. Each comment has a unique ID, which is the first property.

The following items are included in the data set:

- Malignant: This is the Label column, which has values 0 and 1, indicating whether or not the comment is malignant.
- Highly Malignant: Comments that are exceedingly malignant and damaging are referred to be "particularly malignant."
- Rude: It refers to remarks that are obnoxious and rude.
- Threat: It includes indications of words that are threatening to someone.
- Abuse: It refers to statements that are abusive.

- Loathe: Hateful and loathing comments are described as hateful and loathing comments.
- ID: It contains unique identifiers for each comment content.
- Comment text: This column comprises the extracted remarks from various social media networks.

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

Data Preprocessing Done

Before developing the classification model prediction, the following pre-processing pipeline must be completed:

1. Load the dataset
2. Remove any values that are null.
3. Remove the column id.
4. Lowercase the comment text and replace the 'n' with a single space.
5. Remove all other data from the comment text except text data (a-z).
6. Eliminate all stop words and punctuation marks.
7. Use the SnowballStemmer to apply stemming.
8. Use TfidfVectorizer to convert text to vectors.
9. Load a model that has been stored or serialised.
10. Predict multi-class label values

- ## 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 information on websites or to view free-form text. It's a picture made from of words from a certain book or topic, with the size of each word indicating 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:

These are comments of various types, and with the assistance of a word cloud, we can determine if there is an abuse comment, what sort of words it includes, and how it compares to other comments.

- ## State the set of assumptions (if any) related to the problem under consideration

Cyberbullying is becoming more prevalent in countries all around the world. In essence, cyberbullying is quite similar to the sort of bullying that many youngsters 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, since they may experience a variety of emotional disorders that influence their social and academic performance, as well as their general mental health.

- ## Hardware and Software Requirements and Tools Used

  - **Hardware technology being used:**

    CPU : MacBook Pro, M1 Chipset

    RAM :8GB

    GPU : 8GB

  - **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, Pandas-profiling, Missingno, NLTK.

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

All of these pre-processing procedures were done on the testing dataset as well, and I verified the full training dataset for any kind of missing values information.

Code:

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

Output:

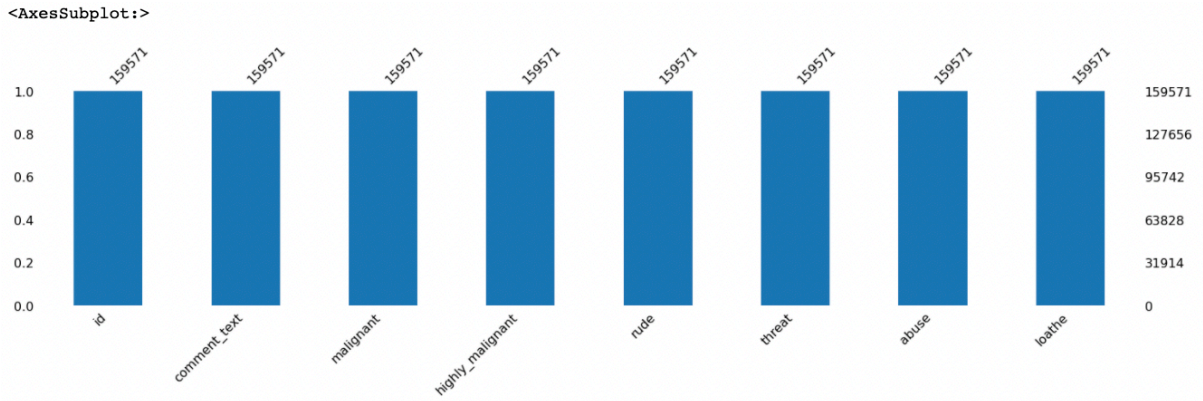
```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

Visual Representation:

Code:

```
# checking the missing value in visual graph using missingno.

missingno.bar(df_train, figsize = (25,5), color="tab:blue")
```

Output:



We next proceeded to examine the dataset details.

We may validate the non-null count details as well as the datatype information using the info method. We have a total of 8 columns, two of which are of object datatype and the other six are of integer datatype.

Code:

```
# checking info of train dataset

df_train.info()
```

Output:

```
<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 initial 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()
```

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.

```
#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)
```

In this case, we've deleted all of the unnecessary 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)
```

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

```
#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         : 39125213
Percentage of Length Cleaned : 37.790959044334414%
```

- ## Testing of Identified Approaches (Algorithms)

The following is a comprehensive list of all the algorithms used in the training and testing of the classification model:

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

- ## 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.

```
#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  : 0.18968223825800734
        Accuracy Score: 0.5210786175465248
              precision    recall  f1-score   support

           0       0.18      0.80      0.30      1281
           1       0.08      0.47      0.14       150
           2       0.12      0.72      0.21       724
           3       0.02      0.30      0.04        44
           4       0.12      0.67      0.20       650
           5       0.04      0.44      0.08       109

   micro avg       0.13      0.71      0.22      2958
   macro avg       0.09      0.57      0.16      2958
weighted avg       0.14      0.71      0.23      2958
 samples avg       0.05      0.07      0.06      2958
Completed in [17.697587334001582 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 tweaking method.

- ## Key Metrics for success in solving problem under consideration

Hyperparameter Tunning:

Input:

```
#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_
```

Output:

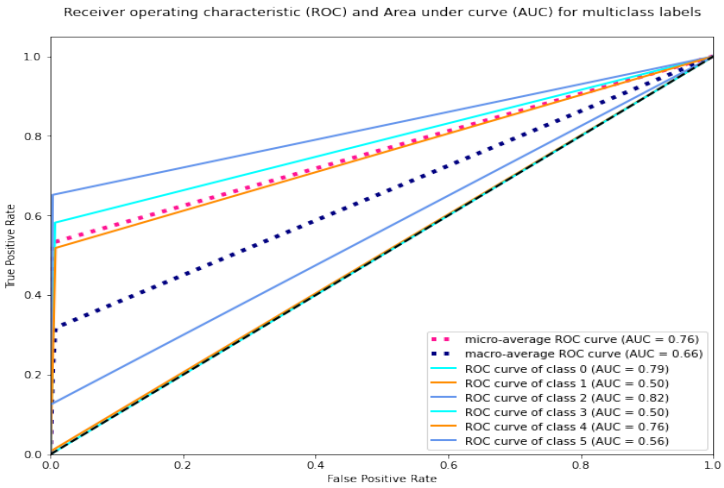
```
{'estimator__loss': 'hinge',
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2',
 'estimator__random_state': 42}
```

Final Classification Model Details:

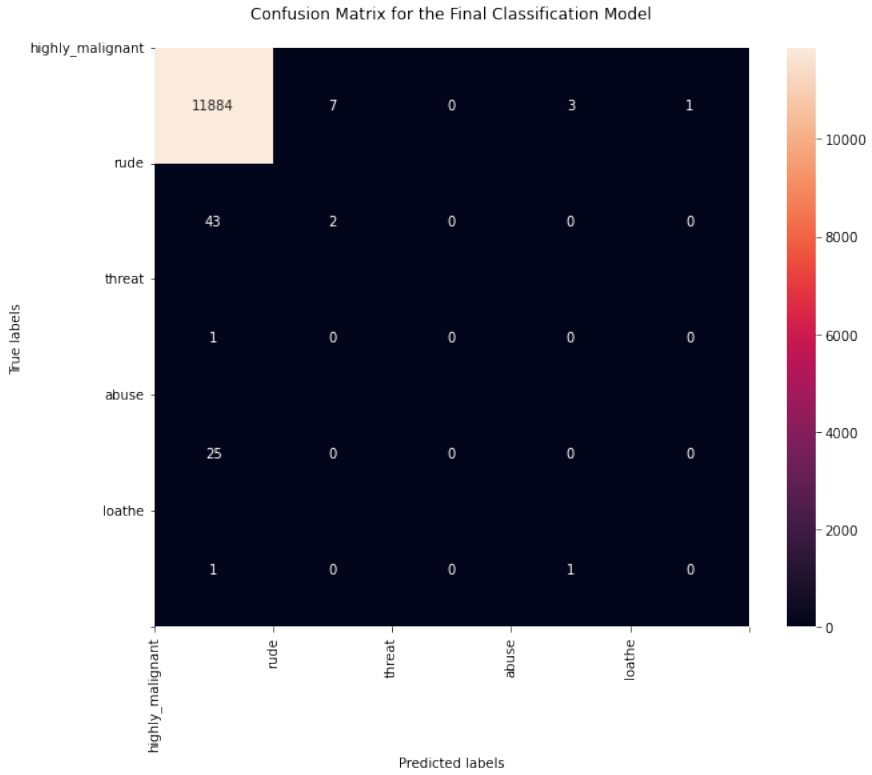
```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge', multi_class='ovr', penalty='l2', random_state=42))
Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100

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: 91.40207219251337
Hamming loss for the Best Model is: 2.0401626559714794
```

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 succesful ever whats hating s...	0	0	0	0	0	0
1	rfc title fine imo	0	0	0	0	0	0
2	sources zawe ashton lapland	0	0	0	0	0	0
3	look back source information updated correct f...	0	0	0	0	0	0
4	anonymously edit articles	0	0	0	0	0	0
...	...	...	...	...	...	...	...
153159	totally agree stuff nothing long crap	0	0	0	0	0	0
153160	throw field home plate get faster throwing cut...	0	0	0	0	0	0
153161	okinotorishima categories see changes agree co...	0	0	0	0	0	0
153162	one founding nations eu germany law return qui...	0	0	0	0	0	0
153163	stop already bullshit welcome fool think kind ...	0	0	0	0	0	0

153164 rows × 7 columns

- ## Visualizations

I generated an initial thorough report on my dataframe values using the pandas profiling tool. It provides us with information like as correlations, missing values, duplicate rows, variable types, memory capacity, and so on for the produced dataset. This aids us in a more thorough visualisation by isolating each portion one by one and comparing and researching the effects of all the accessible feature columns on the prediction of our target label.

Code:

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

```
Summarize dataset: 100% 26/26 [00:26<00:00, 3.14it/s, Completed]
Generate report structure: 100% 1/1 [00:01<00:00, 1.28s/it]
Render HTML: 100% 1/1 [00:00<00:00, 3.83it/s]
```

Output:

Overview | Alerts 28 | Reproduction

Dataset statistics

Number of variables	9
Number of observations	159571
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	286
Duplicate rows (%)	0.2%
Total size in memory	11.0 MiB
Average record size in memory	72.0 B

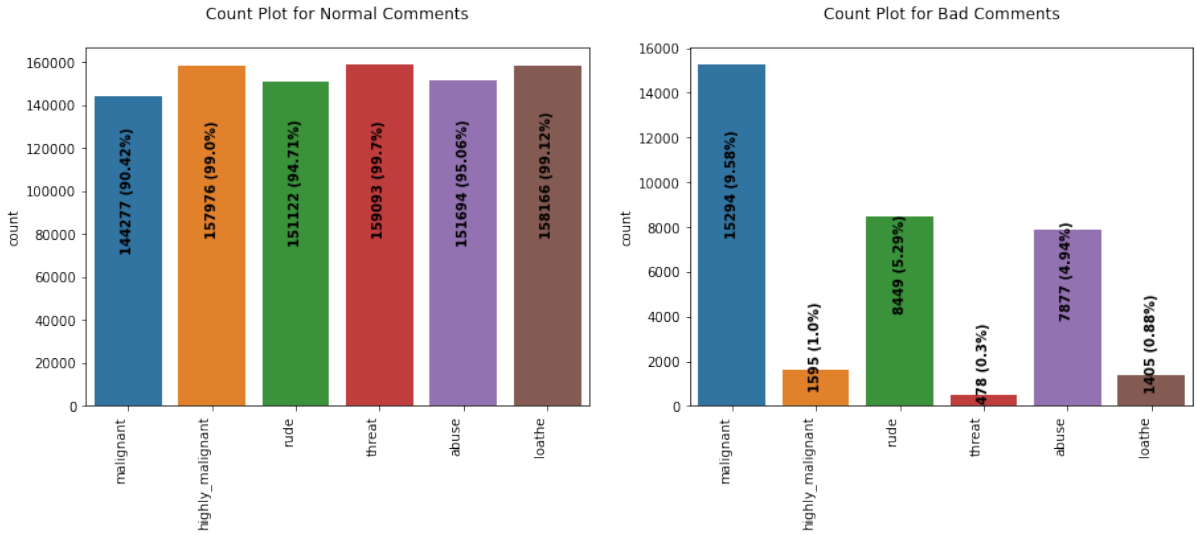
Variable types

Categorical	7
Numeric	2

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()
```

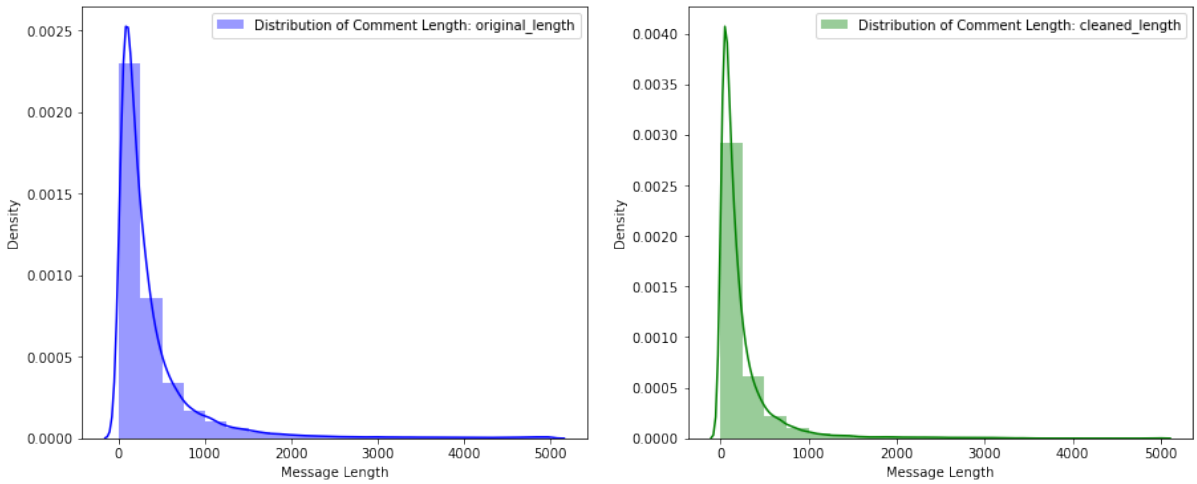
Output:



Code:

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

fig, ax = plt.subplots(1,2,figsize=(15,6))
j=0
colors = ['blue','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()
```

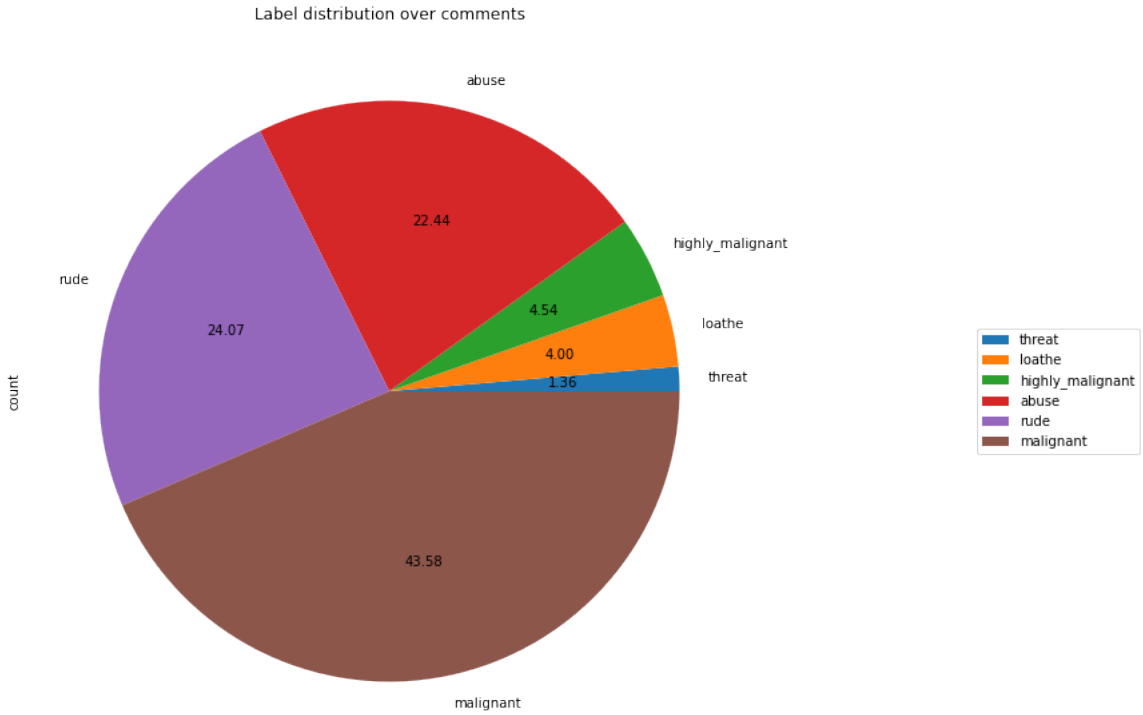
Output:



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))
```

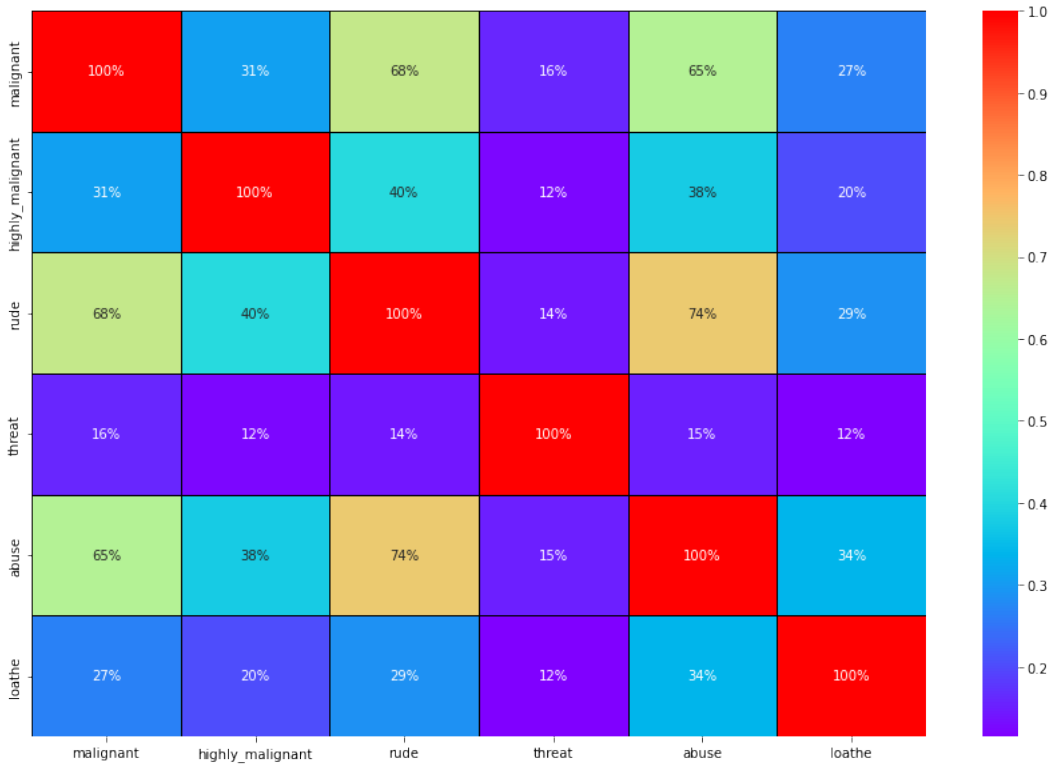
Output:



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='rainbow', 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, it was discovered that the dataset is unbalanced, with more records for normal comments than for negative remarks (including malignant, very malignant, impolite, threat, abuse, and detest). Furthermore, using a distribution plot for comment length, it was discovered that after cleaning, the majority of comment length reduces from 0-1100 to 0-900. Moving on to the word cloud, it was discovered that malignant comments contain terms such as fuck, nigger, moron, hatred, suck, and so on. Words like ass, fuck, bitch, shit, die, suck, faggot, and others appear often in highly malignant remarks. Words like nigger, ass, fuck, suck, crap, bitch, and others are used in nasty statements. Words like death, must die, kill, murder, and others are used in threat statements. Words like stupid, nigger, obese, jew, bitch, and others are used in abusive comments. and despise remarks include phrases such as nigga, dumb, nigger, die, gay, cunt, and so on.

CONCLUSION

- ## Key Findings and Conclusions of the Study

The survey discovered that just a small percentage of online users use unparliamentary language. And the majority of these phrases contain a lot of stop words and are pretty lengthy. As previously said, a few motivated rude mobs use harsh language in internet forums to harass individuals and prevent them from doing what they are not permitted 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 assessment criteria in this research, including log loss, hamming loss, and accuracy.

My project's conclusion is that we should use good, courteous language on 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 be courteous, cool, and collected; one of the greatest solutions is to ignore it and overcome it in a constructive way.

Limitations of this work and Scope for Future Work

Problems faced while working in this project:

- Because it took more than 2 hours, extra computer power was necessary.
- The dataset is unbalanced, and the comment messages are poorly written.
- Because time was used more, good parameters could not be acquired through hyper-parameter tweaking.

Areas of improvement:

- Could be given a solid dataset that doesn't take too long.
- Time complexity is reduced.
- Creating a well-balanced dataset with fewer mistakes.

THANK YOU!