MALIGNANT COMMENTS CLASSIFIER PROJECT REPORT

Submitted by:

Shikha Chaudhary

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to Shubham Yadav, 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 in the Natural Language Processing and Natural Language Toolkit sections.

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

  People may now express themselves broadly 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 hatred is

an issue across numerous platforms, there are no models for detecting online hate.

Online hatred has been highlighted as a big problem on online social media platforms, and has been defined 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 may have a negative impact on anyone, resulting in sadness, mental disease, self-hatred, and suicide thoughts.

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. This implies 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 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.

- ## 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 purpose of the literature review is to:

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

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

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

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

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

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

```
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)
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
```

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

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.

Variable	Definition
id	A unique id aligned with each comment text.
comment_text	It includes the comment text.
malignant	It is a column with binary values depicting which comments are malignant in nature.
highly_malignant	Binary column with labels for highly malignant text.
rude	Binary column with labels for comments that are rude in nature.
threat	Binary column with labels for threatening context in the comments.
abuse	Binary column with labels with abusive behaviour.
loathe	Label to comments that are full of loathe and hatred.

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. You need to build a model that can differentiate between comments and its 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

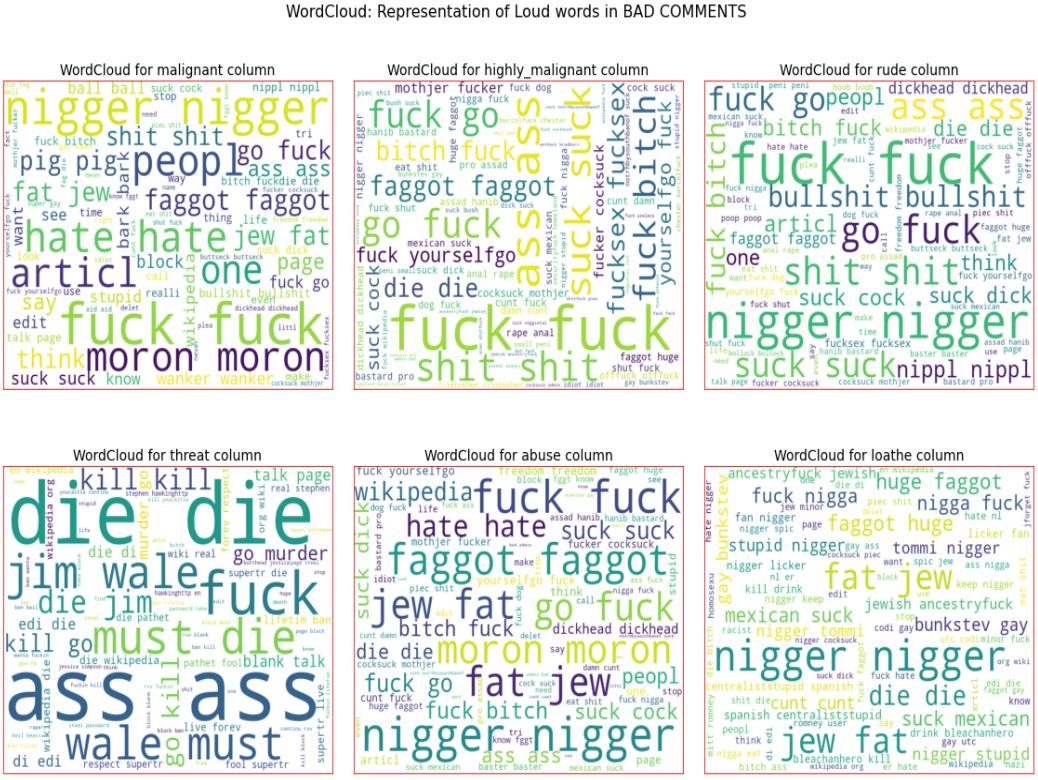
- **Data Inputs- Logic- Output Relationships**

I 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()
```

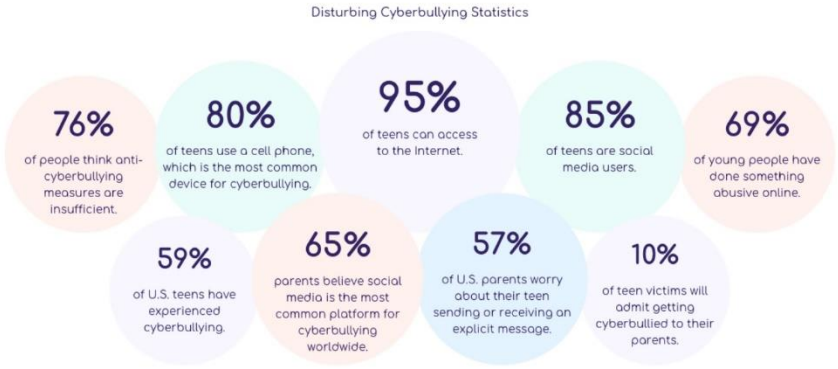
Output:



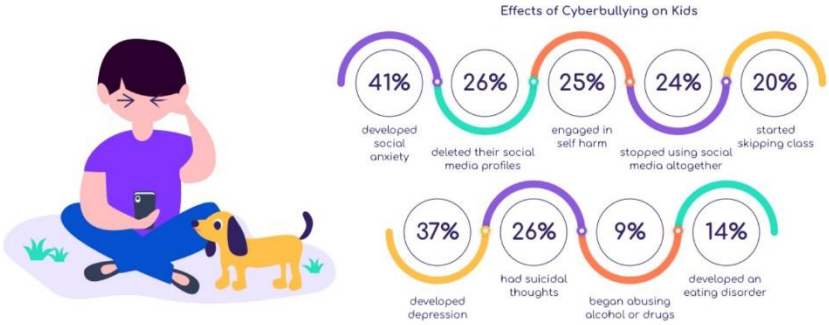
These are the comments that belongs to different type so which the help of word cloud we can see if there is abuse comment which type of words it contains and similar to other comments as well.

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

  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**

Hardware technology being used.

RAM : 8 GB

CPU : Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 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, pandas-profiling, missingno, NLTK

Model/s Development and Evaluation

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

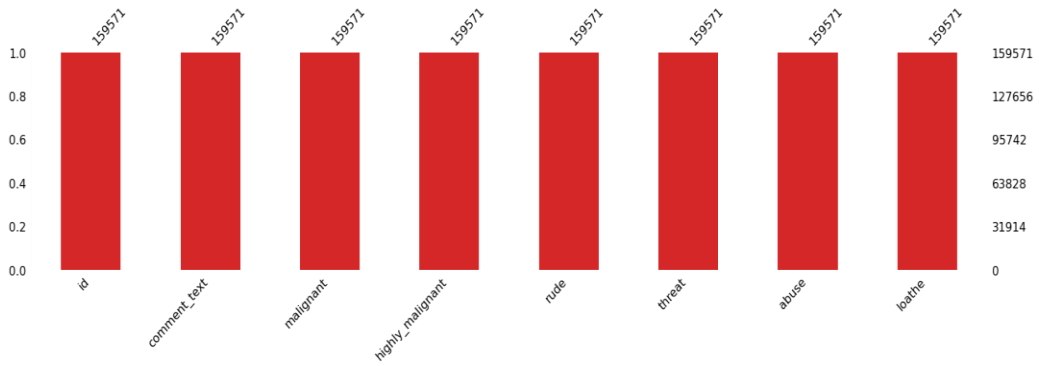
I checked through the entire training dataset for any kind of missing values information and all these pre processing steps were repeated on the testing dataset as well.

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
```

Visual Representation:



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

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

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

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

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

```
# Data Cleansing

# as the feature 'id' has no relevance w.r.t. model training I am dropping this column
df.drop(columns=['id'],inplace=True)
# converting comment text to lowercase format
df['comment_text'] = df.comment_text.str.lower()
df.head()
```

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

```
# 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**

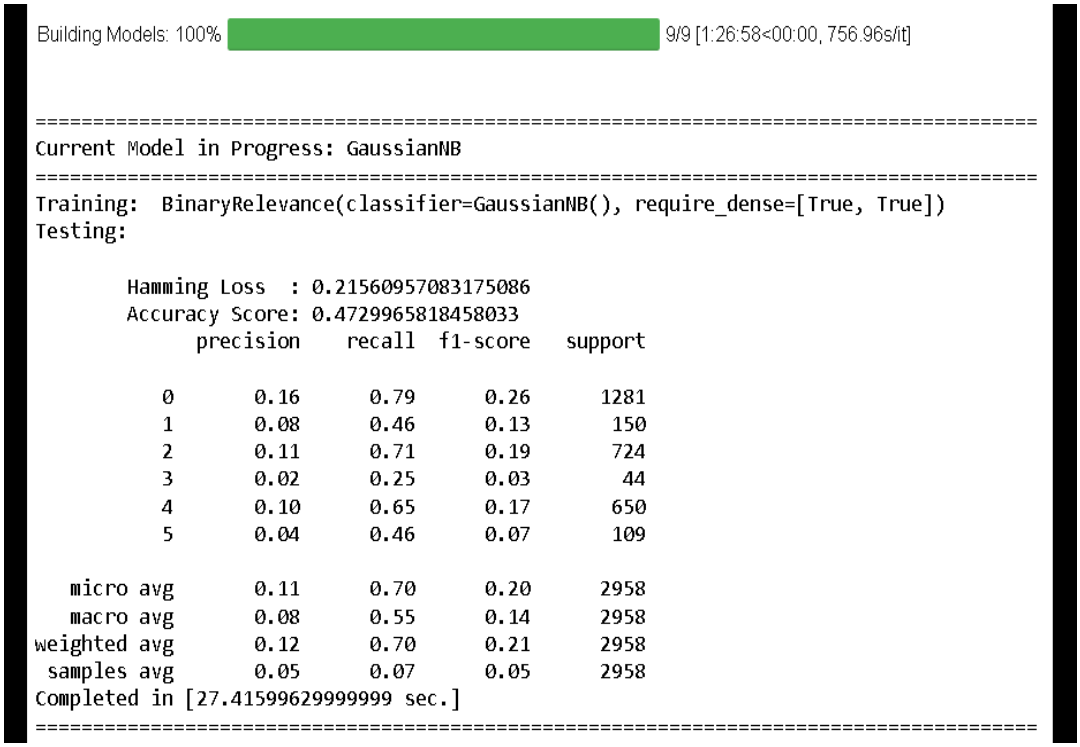
I created a classification function that included the evaluation metrics details for the generation of our Classification Machine Learning models.

```
# 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:



Observation:

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

- **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_
```

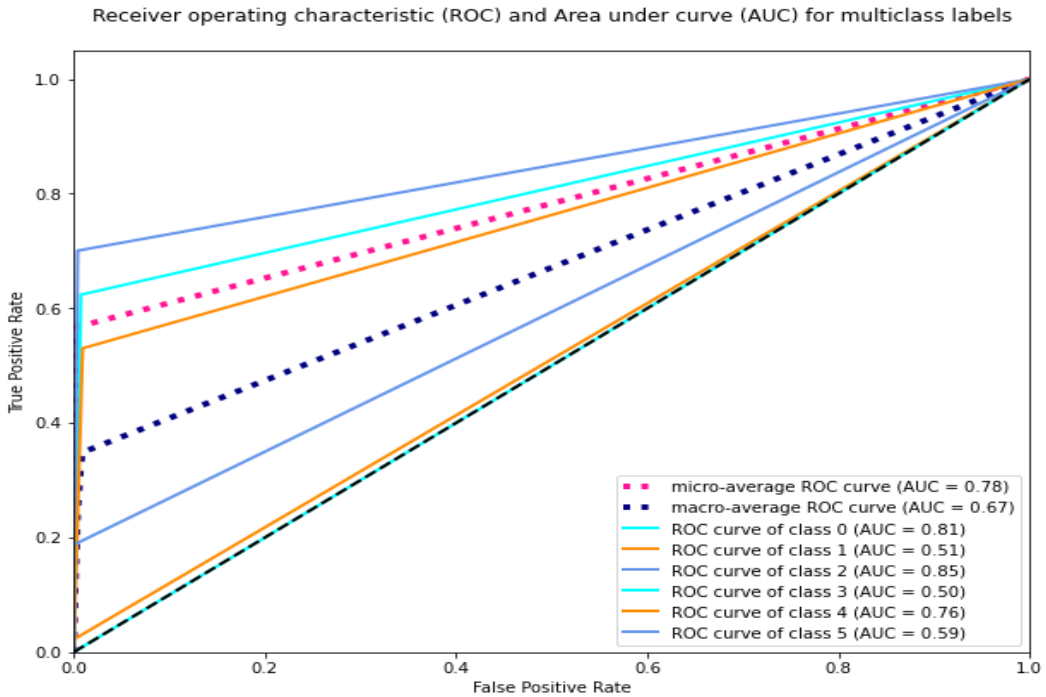
```
{'estimator__loss': 'hinge',
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2',
 'estimator__random_state': 42}
```

Final Classification Model details:

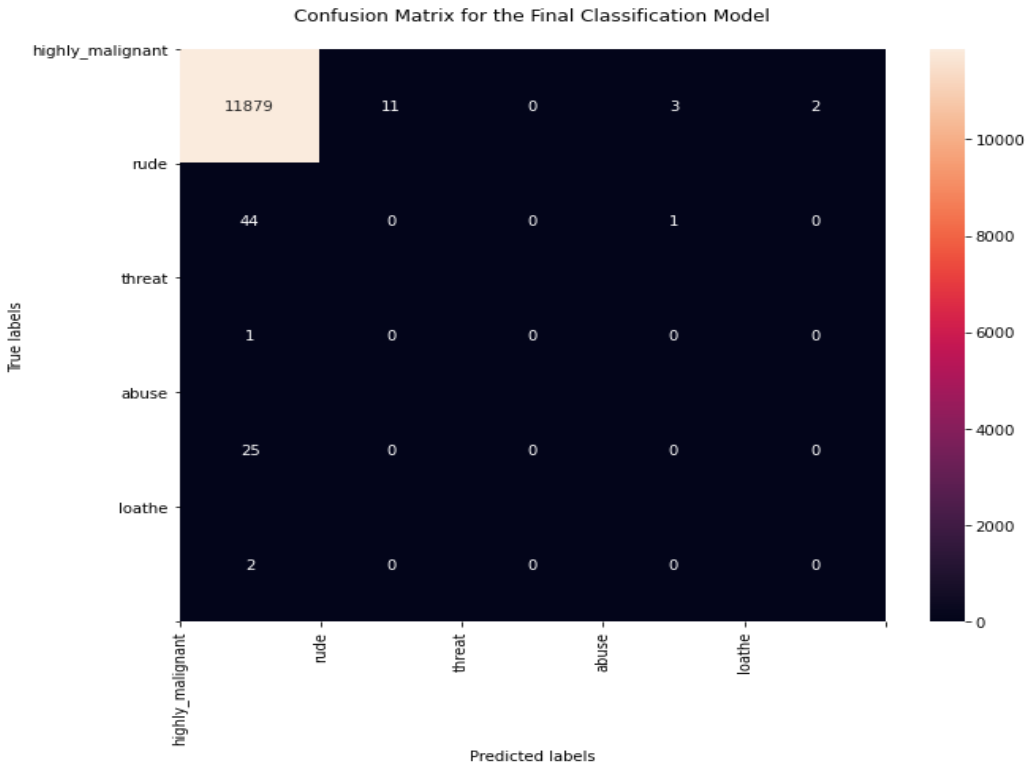
```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge', multi_class='ovr', penalty='l2', random_state=42))
Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)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

I used the pandas profiling feature to generate an initial detailed report on my dataframe values. It gives us various information on the rendered dataset like the correlations, missing values, duplicate rows, variable types, memory size etc. This assists us in further detailed visualization separating each part one by one comparing and research for the impacts on the prediction of our target label from all the available feature columns.

Code:

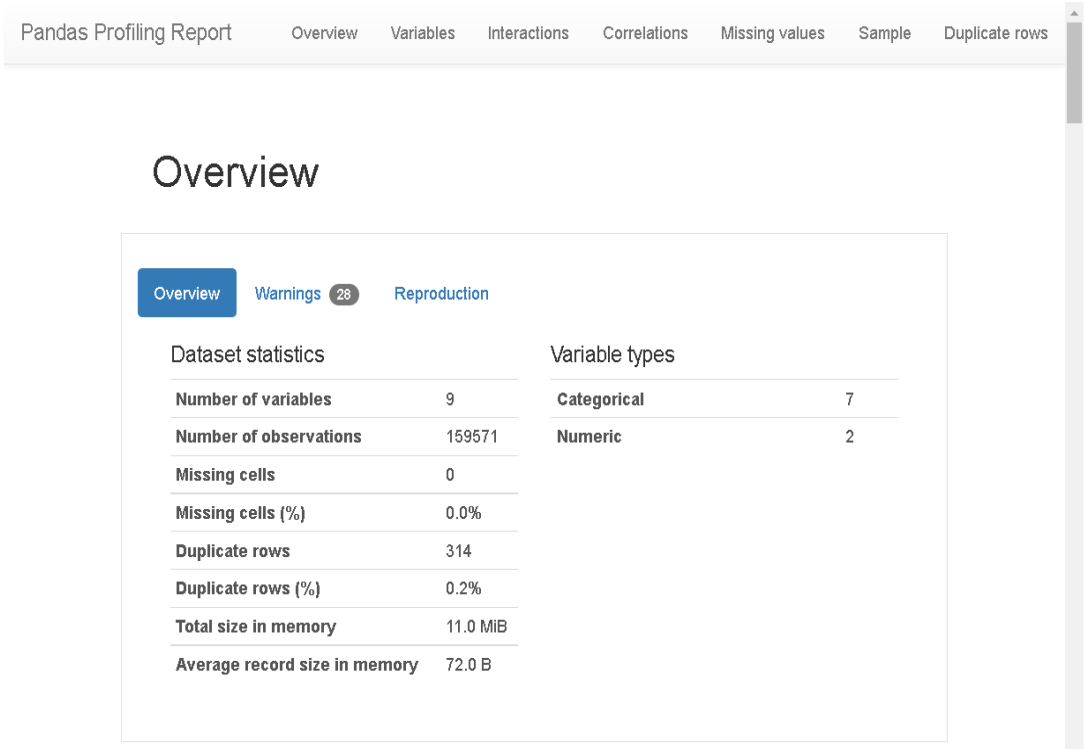
```
pandas_profiling.ProfileReport(df)
```

Summarize dataset: 100% 22/22 [00:23<00:00, 1.56it/s, Completed]

Generate report structure: 100% 1/1 [00:04<00:00, 4.67s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.38it/s]

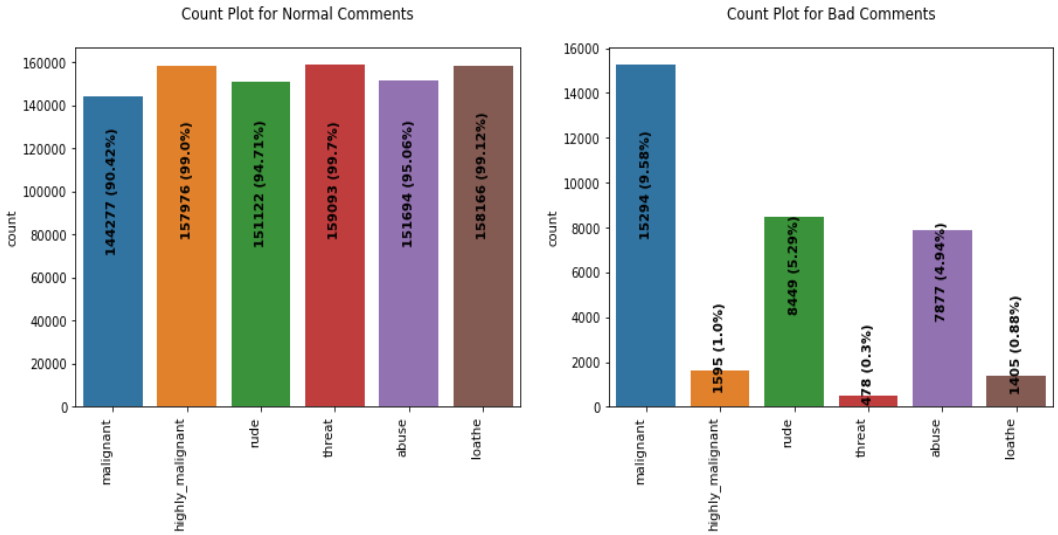
Output:



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()
```

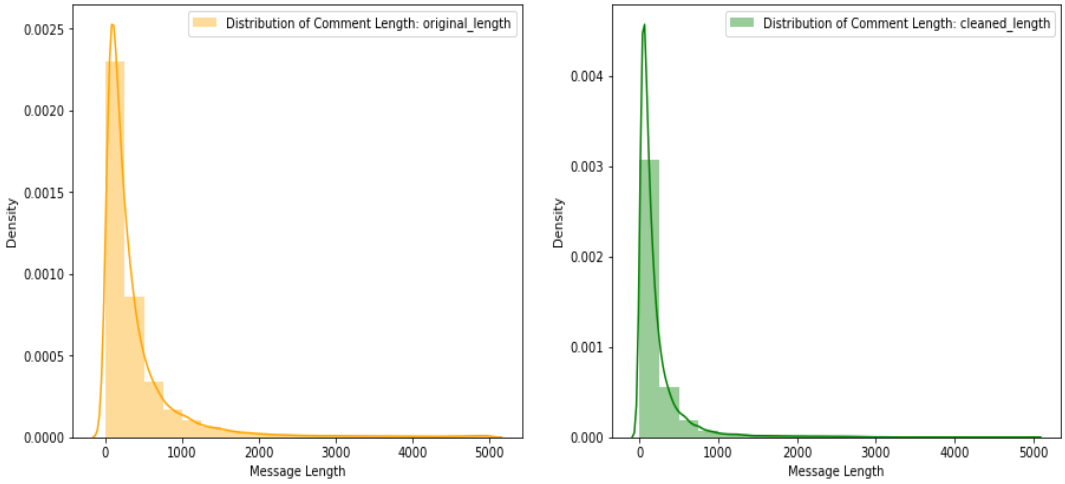
Output:



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()
```

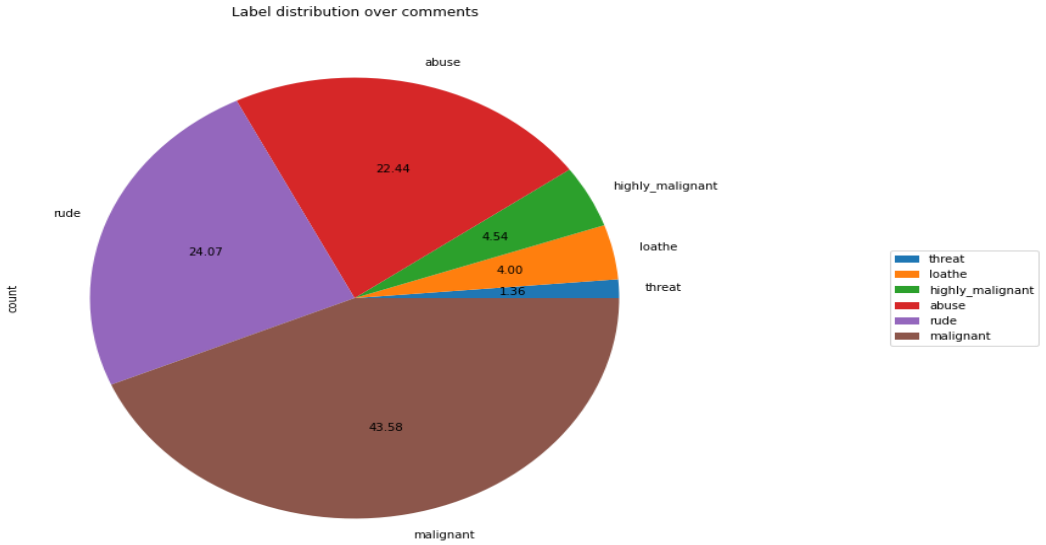
Output:



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))
```

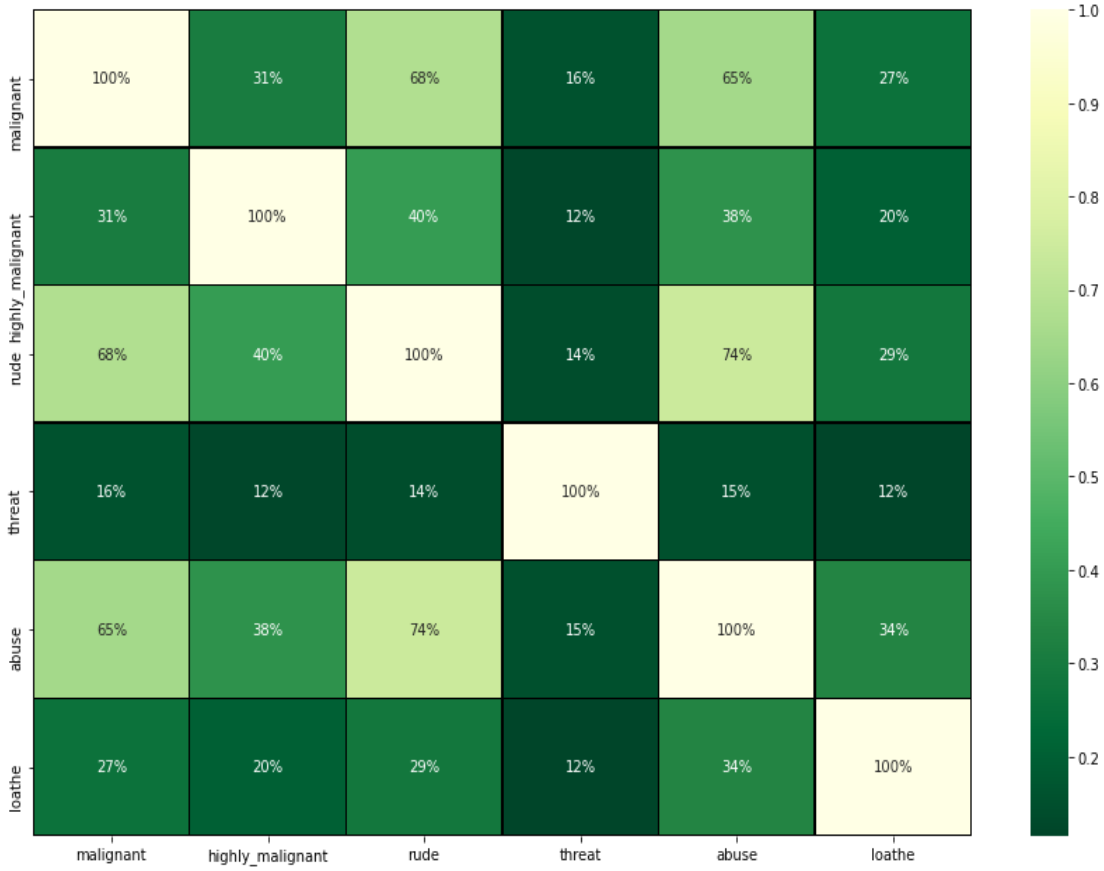
Output:



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**

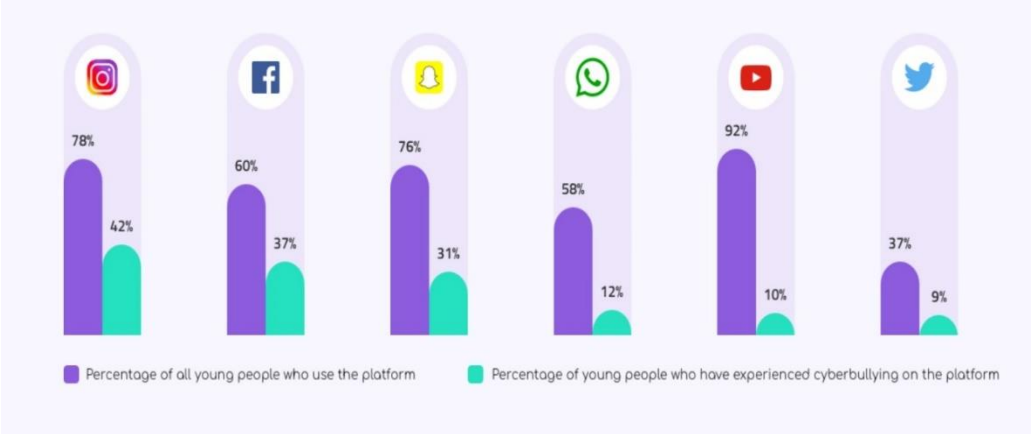
Starting with univariate analysis, with the help of count plot it was found that dataset is imbalanced with having higher number of records for normal comments than bad comments (including malignant, highly malignant, rude, threat, abuse and loathe). Also, with the help of distribution plot for comments 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 unparliamentary 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

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.

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

- **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
- Good parameters could not be obtained using hyperparameter tuning as time was consumed more

Areas of improvement:

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