FLIP ROBO

Malignant Comments Classifier Project

Submitted by:

ROHIT KATTEWAR

ACKNOWLEDGEMENT

I would like to thank and express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project named ‘Malignant Comments Classifier Project’ using Natural Language Processing (NLP) algorithms and toolkit.

I will thank my mentors, under whose guidance I learned a lot about Machine Learning, Natural Language Processing and much more.

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 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 the Literature

The purpose of the literature review is to:

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

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

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

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

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users.

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

❖ Motivation for the Problem Undertaken

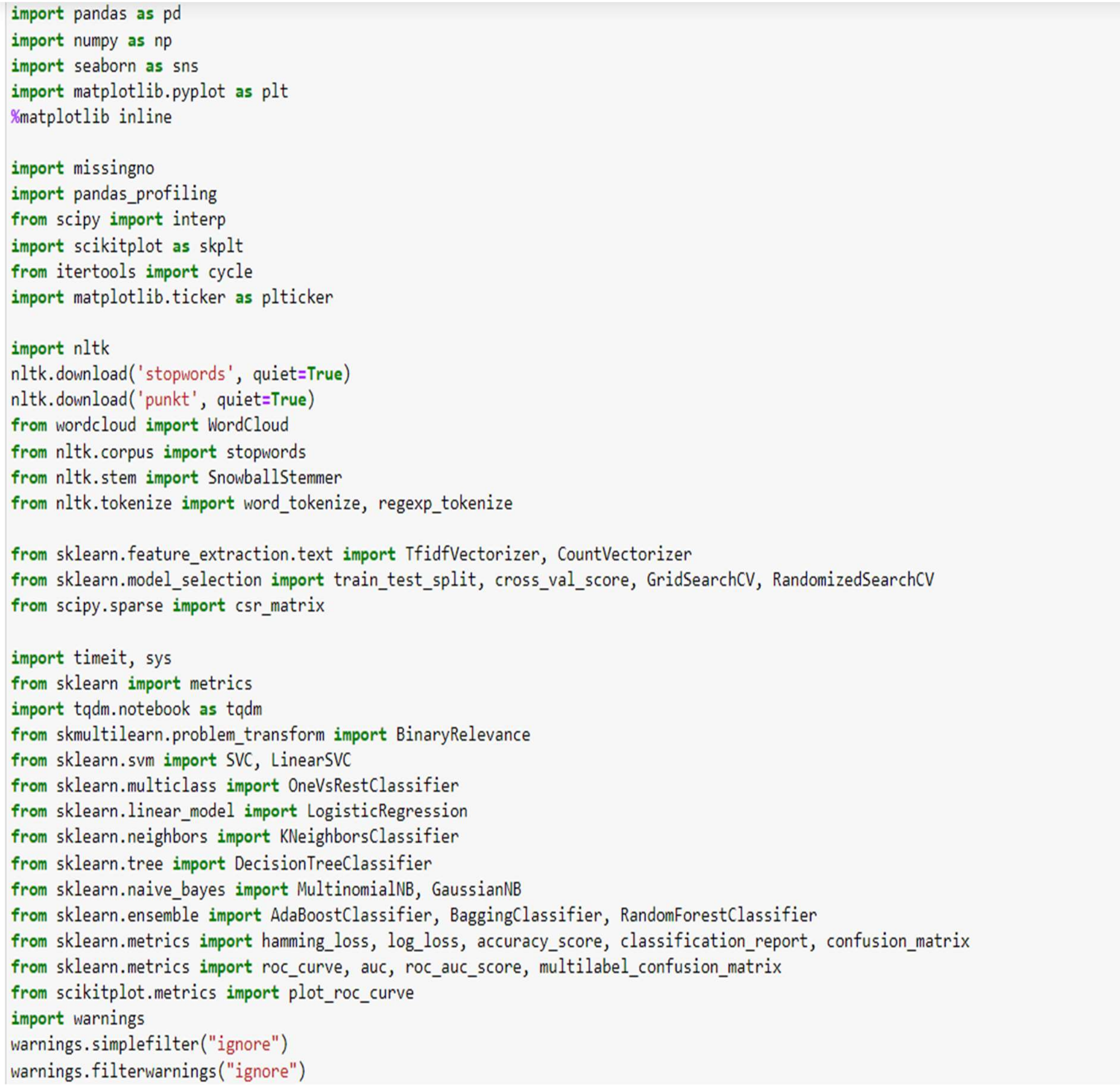
One of the first lessons we learn as children is that the louder you scream and the bigger of a tantrum you throw, you more you get your way. Part of growing up and maturing into an adult and functioning member of society is learning how to use language and reasoning skills to communicate our beliefs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

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

With widespread usage of online social networks and its popularity, social networking platforms have given us incalculable opportunities 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:



```
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")
```

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.

❖ 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.

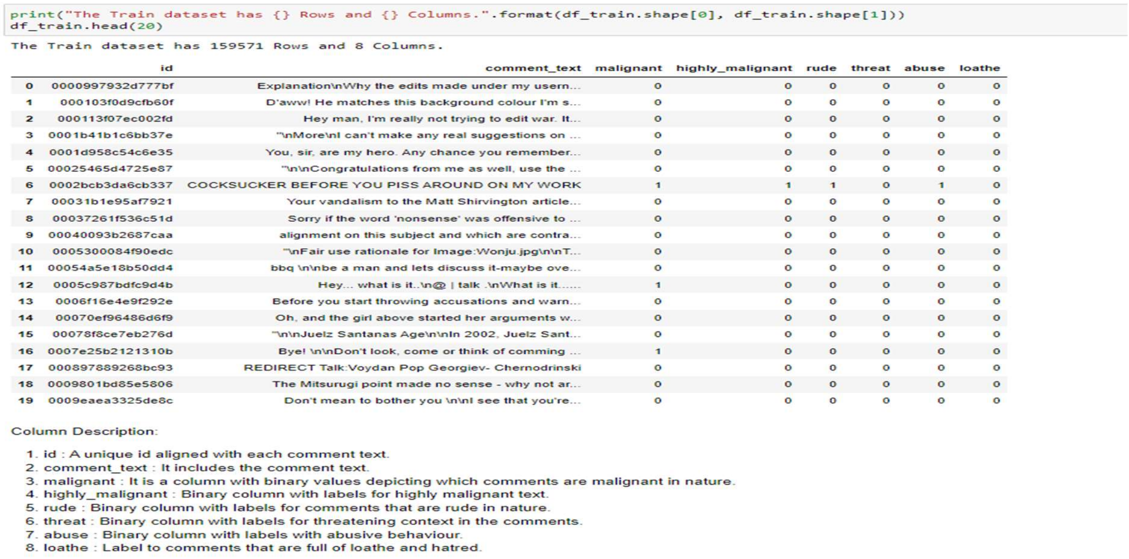
Threat: It contains indication of the comments that are giving any threat to someone.

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.



```
print("The Train dataset has {} Rows and {} Columns.".format(df_train.shape[0], df_train.shape[1]))
df_train.head(20)
```

```
The Train dataset has 159571 Rows and 8 Columns.
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0
10	0005300084f90edc	"\nFair use rationale for Image:Wonju.jpg\n\nT...	0	0	0	0	0	0
11	00054a5e18b50dd4	bbq \n\nbe a man and lets discuss it-maybe ove...	0	0	0	0	0	0
12	0005c987bdfc9d4b	Hey... what is it..\n@ | talk .\nWhat is it......	1	0	0	0	0	0
13	0006f16e4e9f292e	Before you start throwing accusations and warn...	0	0	0	0	0	0
14	00070ef96486d6f9	Oh, and the girl above started her arguments w...	0	0	0	0	0	0
15	00078f8ce7eb276d	"\n\nJuelz Santanas Age\n\nIn 2002, Juelz Sant...	0	0	0	0	0	0
16	0007e25b2121310b	Bye! \n\nDon't look, come or think of comming ...	1	0	0	0	0	0
17	000897889268bc93	REDIRECT Talk:Voydan Pop Georgiev- Chernodrinski	0	0	0	0	0	0
18	0009801bd85e5806	The Mitsurugi point made no sense - why not ar...	0	0	0	0	0	0
19	0009eaea3325de8c	Don't mean to bother you \n\nI see that you're...	0	0	0	0	0	0

Column Description:

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

❖ Data Pre-processing 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 i.e. a-z' and remove other data from comment text.

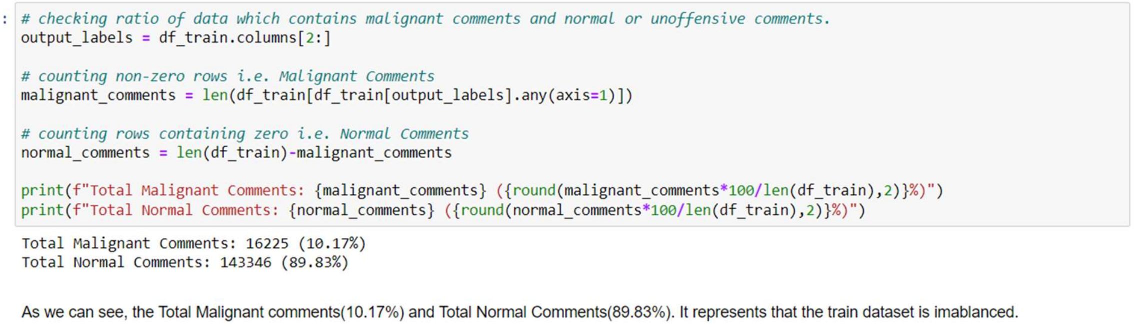
6. Remove stop words and punctuations

7. Apply Stemming using Snowball Stemmer

8. Convert text to vectors using TfidfVectorizer

9. Load saved or serialized model

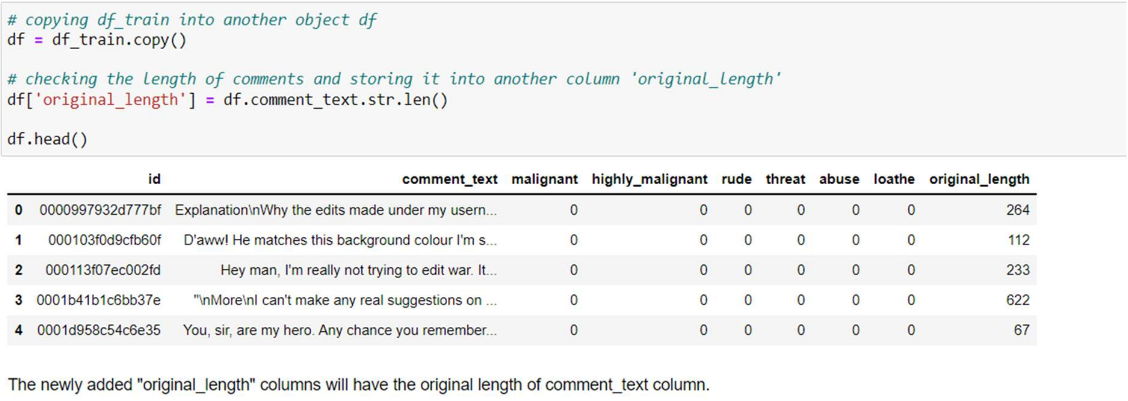
10. Predict values for multi class label



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

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

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

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

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

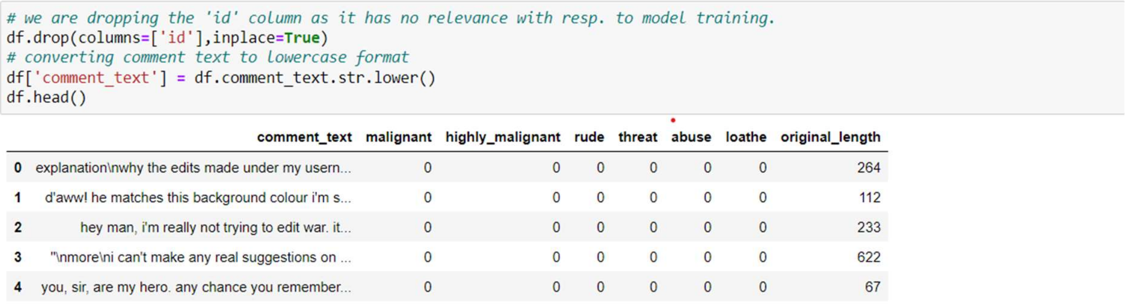
As we can see, the Total Malignant comments(10.17%) and Total Normal Comments(89.83%). It represents that the train dataset is imablanced.



```
# copying df_train into another object df
df = df_train.copy()

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

df.head()
```

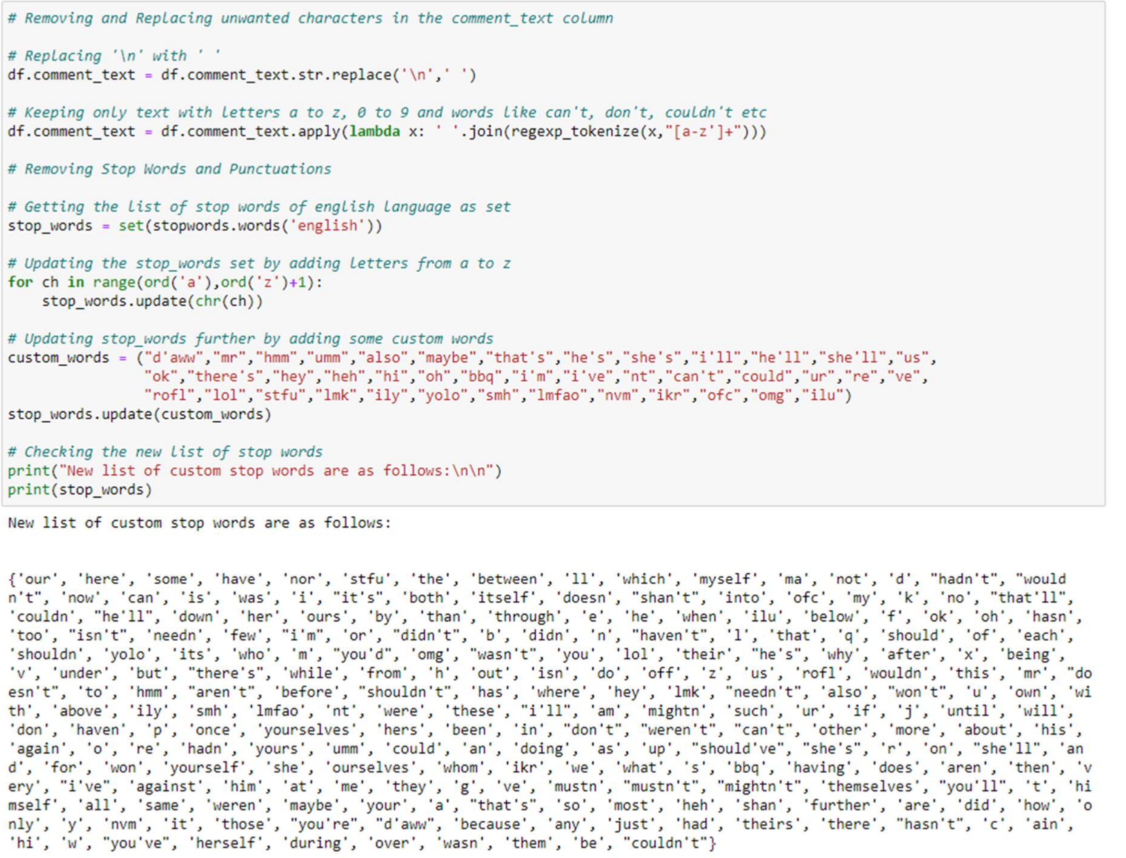
	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67

The newly added "original_length" columns will have the original length of comment_text column.



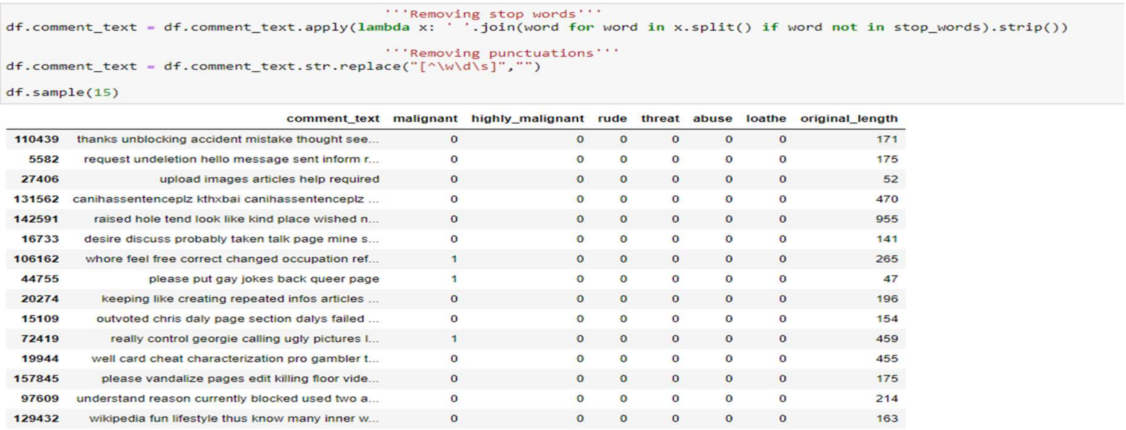
```
# we are dropping the 'id' column as it has no relevance with resp. to model training.
df.drop(columns=['id'],inplace=True)
# converting comment text to lowercase format
df['comment_text'] = df.comment_text.str.lower()
df.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
0	explanation\nwhy the edits made under my usern...	0	0	0	0	0	0	264
1	d'aww! he matches this background colour i'm s...	0	0	0	0	0	0	112
2	hey man, i'm really not trying to edit war. it...	0	0	0	0	0	0	233
3	"\nmore\ni can't make any real suggestions on ...	0	0	0	0	0	0	622
4	you, sir, are my hero. any chance you remember...	0	0	0	0	0	0	67



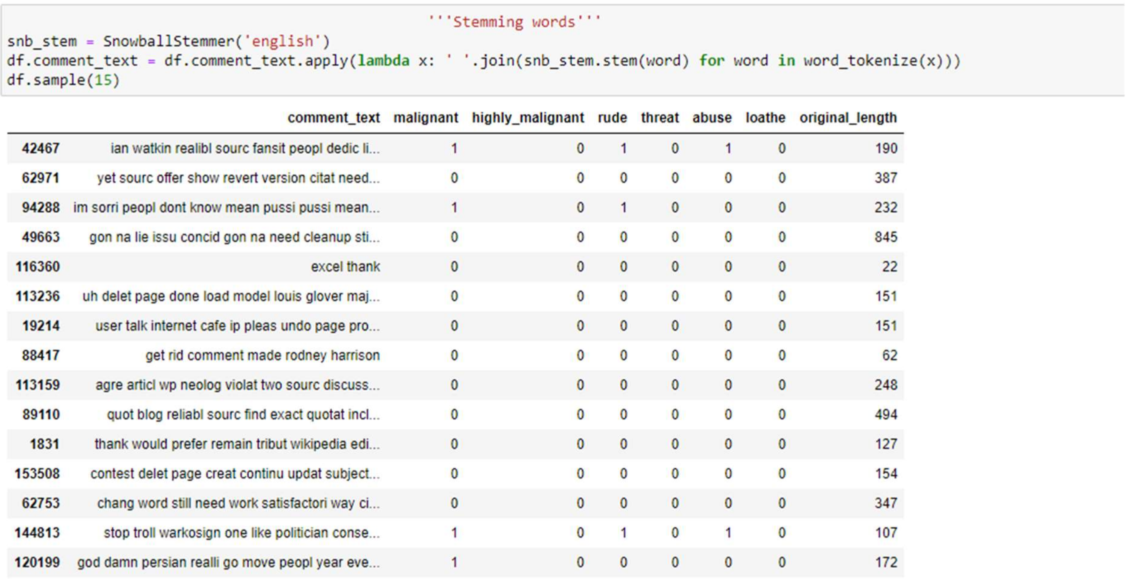
```
# Removing and Replacing unwanted characters in the comment_text column

# 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)
```

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


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



```
                                  '''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]","")

df.sample(15)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
110439	thanks unblocking accident mistake thought see...	0	0	0	0	0	0	171
5582	request undeletion hello message sent inform r...	0	0	0	0	0	0	175
27406	upload images articles help required	0	0	0	0	0	0	52
131562	canihassentenceplz kthxbai canihassentenceplz ...	0	0	0	0	0	0	470
142591	raised hole tend look like kind place wished n...	0	0	0	0	0	0	955
16733	desire discuss probably taken talk page mine s...	0	0	0	0	0	0	141
106162	whore feel free correct changed occupation ref...	1	0	0	0	0	0	265
44755	please put gay jokes back queer page	1	0	0	0	0	0	47
20274	keeping like creating repeated infos articles ...	0	0	0	0	0	0	196
15109	outvoted chris daly page section dalys failed ...	0	0	0	0	0	0	154
72419	really control georgie calling ugly pictures l...	1	0	0	0	0	0	459
19944	well card cheat characterization pro gambler t...	0	0	0	0	0	0	455
157845	please vandalize pages edit killing floor vide...	0	0	0	0	0	0	175
97609	understand reason currently blocked used two a...	0	0	0	0	0	0	214
129432	wikipedia fun lifestyle thus know many inner w...	0	0	0	0	0	0	163

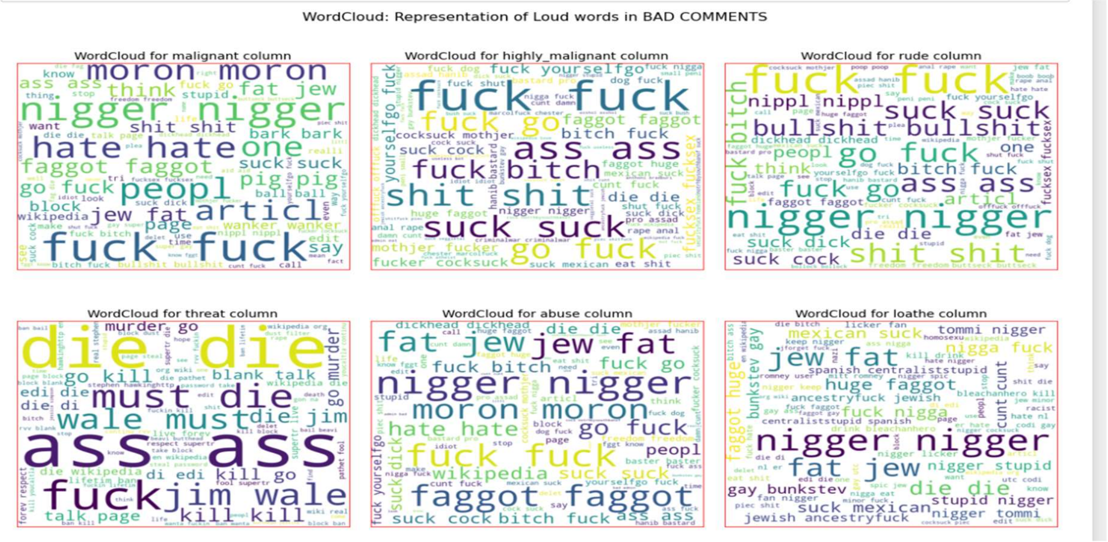


```
                                             '''Stemming words'''
snb_stem = SnowballStemmer('english')
df.comment_text = df.comment_text.apply(lambda x: ' '.join(snb_stem.stem(word) for word in word_tokenize(x)))
df.sample(15)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_length
42467	ian watkin realibl sourc fansit peopl dedic li...	1	0	1	0	1	0	190
62971	yet sourc offer show revert version citat need...	0	0	0	0	0	0	387
94288	im sorri peopl dont know mean pussi pussi mean...	1	0	1	0	0	0	232
49663	gon na lie issu concid gon na need cleanup sti...	0	0	0	0	0	0	845
116360	excel thank	0	0	0	0	0	0	22
113236	uh delet page done load model louis glover maj...	0	0	0	0	0	0	151
19214	user talk internet cafe ip pleas undo page pro...	0	0	0	0	0	0	151
88417	get rid comment made rodney harrison	0	0	0	0	0	0	62
113159	agre articl wp neolog violat two sourc discuss...	0	0	0	0	0	0	248
89110	quot blog reliabl sourc find exact quotat incl...	0	0	0	0	0	0	494
1831	thank would prefer remain tribut wikipedia edi...	0	0	0	0	0	0	127
153508	contest delet page creat continu updat subject...	0	0	0	0	0	0	154
62753	chang word still need work satisfactori way ci...	0	0	0	0	0	0	347
144813	stop troll warkosign one like politician conse...	1	0	1	0	1	0	107
120199	god damn persian realli go move peopl year eve...	1	0	0	0	0	0	172

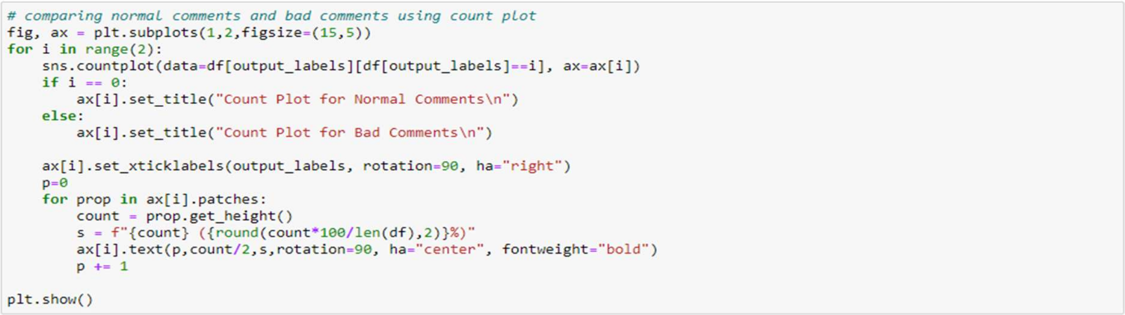
❖ Data Inputs- Logic- Output Relationships:

We 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.

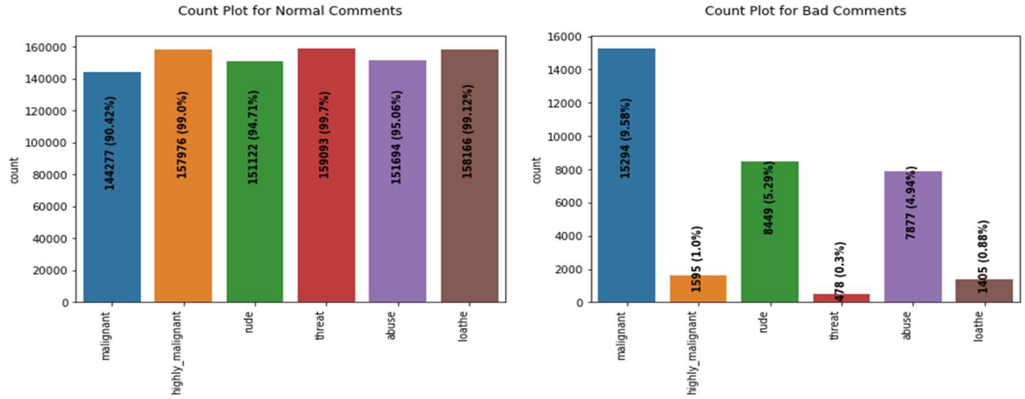


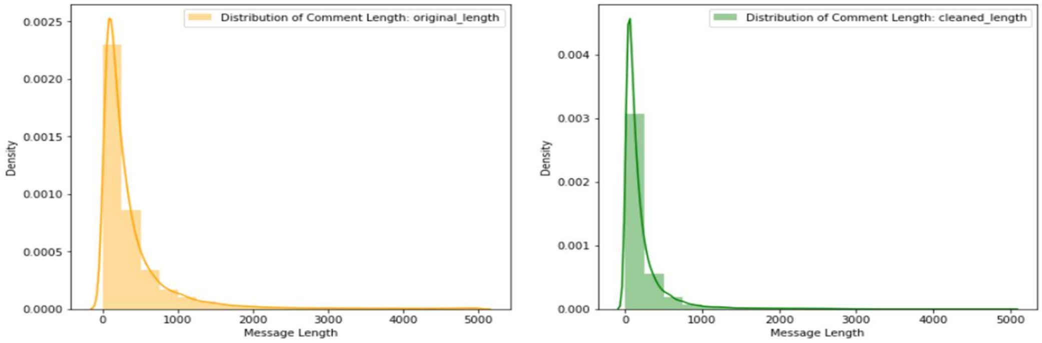
These are the comments that belongs to different type so which the help of word cloud we can see if there is abuse comment which type of words it contains and similar to other comments as well.

Visualization:

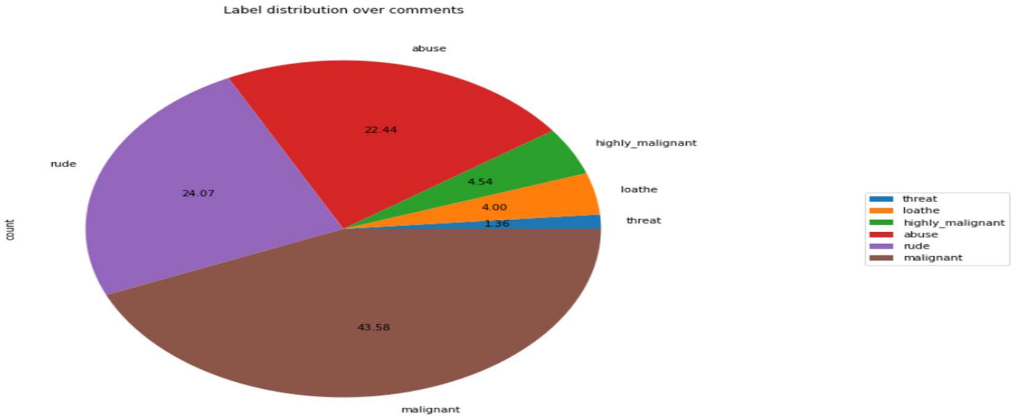


```
# comparing normal comments and bad comments using count plot
fig, ax = plt.subplots(1,2,figsize=(15,5))
for i in range(2):
    sns.countplot(data=df[output_labels][df[output_labels]==i], ax=ax[i])
    if i == 0:
        ax[i].set_title("Count Plot for Normal Comments\n")
    else:
        ax[i].set_title("Count 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()
```





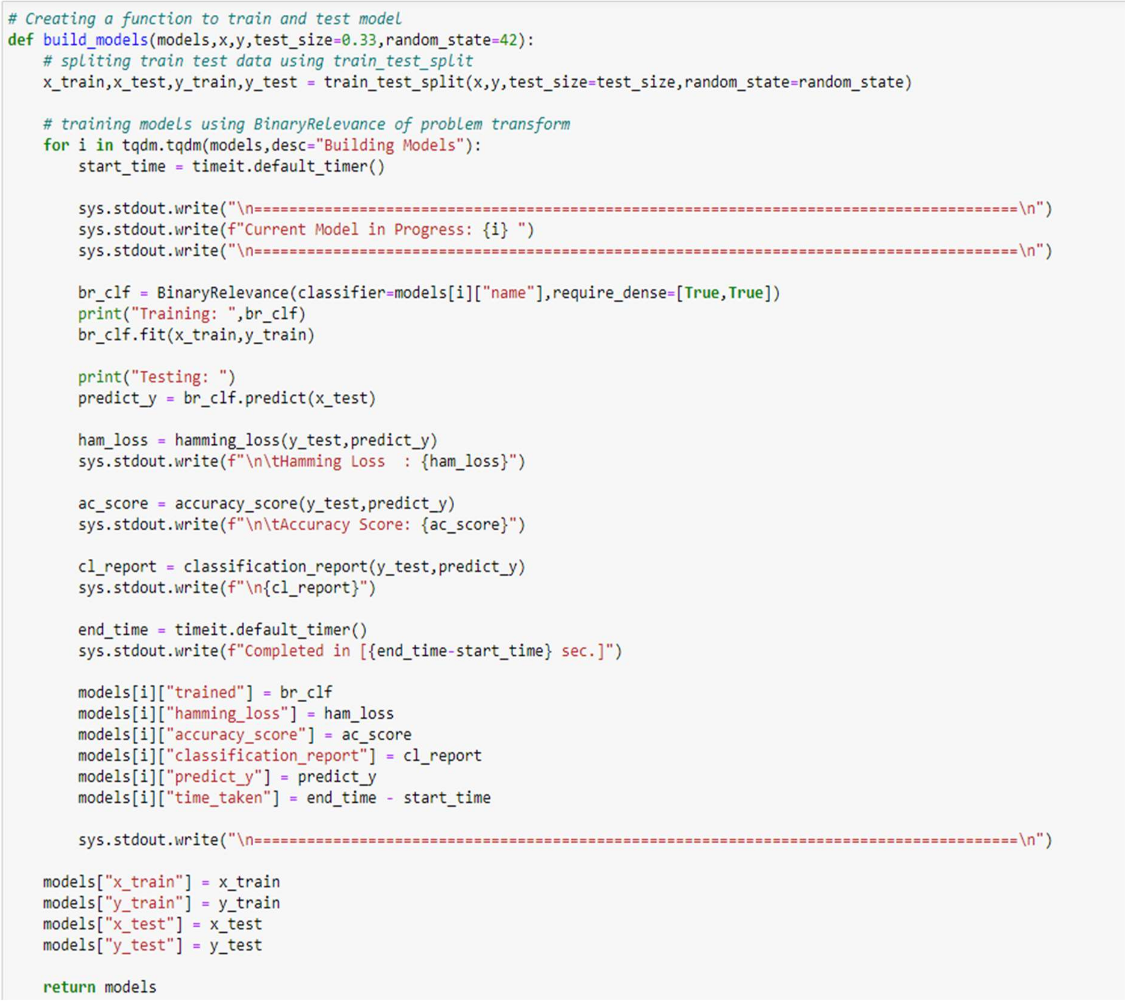
Before cleaning comment_text column most of the comment's length lies between 0 to 1100 while after cleaning it has been reduced between 0 to 900.



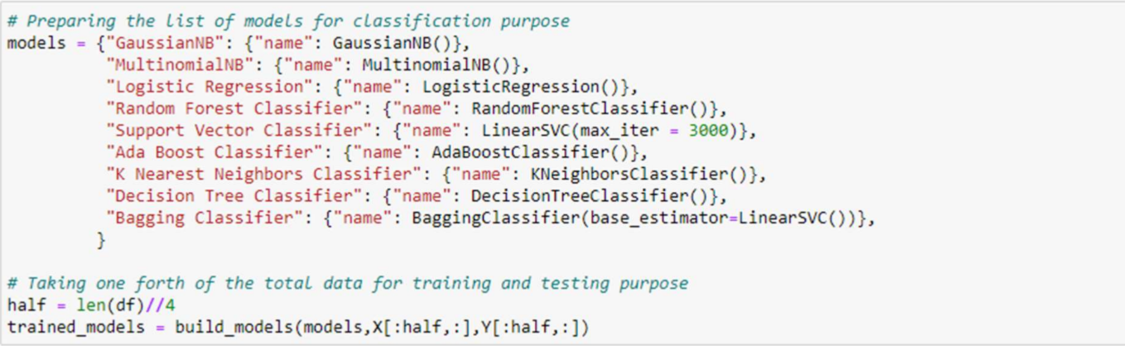
We have used following algorithms for training and testing our model:

- Gaussian Naïve Bayes
- Multinomial Naïve Bayes
- Logistic Regression
- Random Forest Classifier
- Linear Support Vector Classifier
- Ada Boost Classifier
- K Nearest Neighbors Classifier
- Decision Tree Classifier
- Bagging Classifier

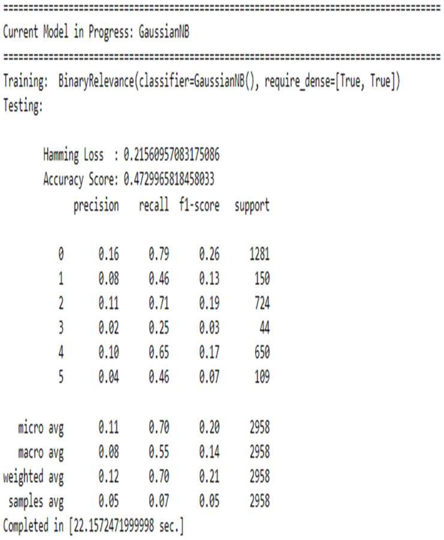
❖ Model Building:



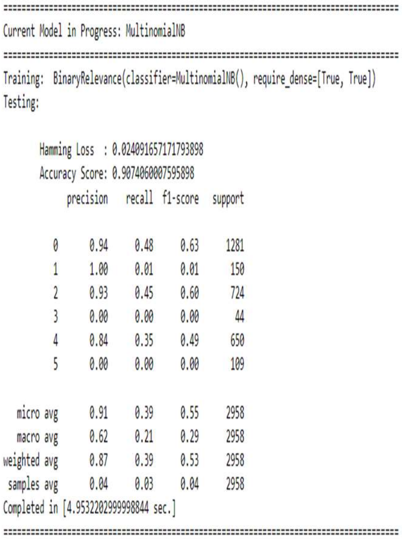
```
# Creating a function to train and test model
def build_models(models,x,y,test_size=0.33,random_state=42):
    # spliting train test data using train_test_split
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=test_size,random_state=random_state)

    # 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
```



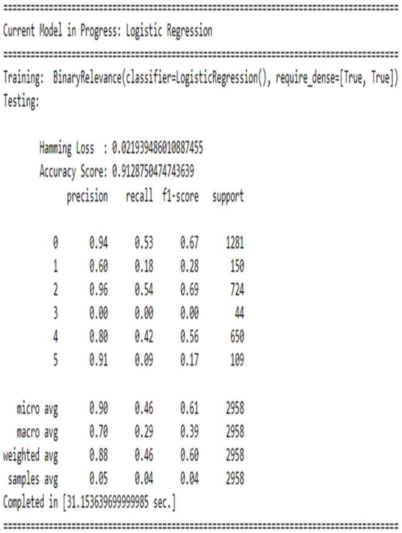
```python
# Preparing the list of models for classification purpose
models = {"GaussianNB": {"name": GaussianNB()},
          "MultinomialNB": {"name": MultinomialNB()},
          "Logistic Regression": {"name": LogisticRegression()},
          "Random Forest Classifier": {"name": RandomForestClassifier()},
          "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,:])
```



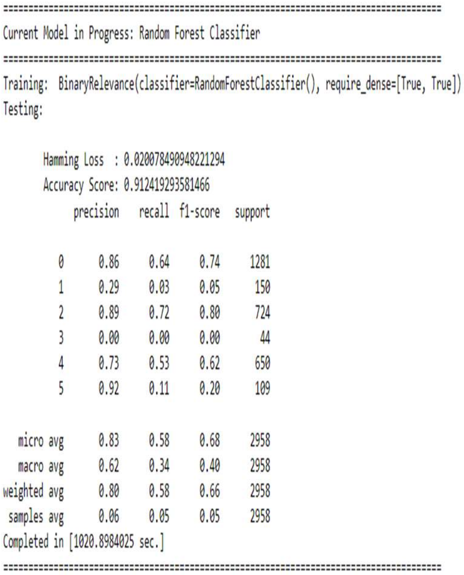
```
================================================================================
Current Model in Progress: GaussianNB
================================================================================
Training:  BinaryRelevance(classifier=GaussianNB(), require_dense=[True, True])
Testing:

        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 [22.1572471999998 sec.]
```



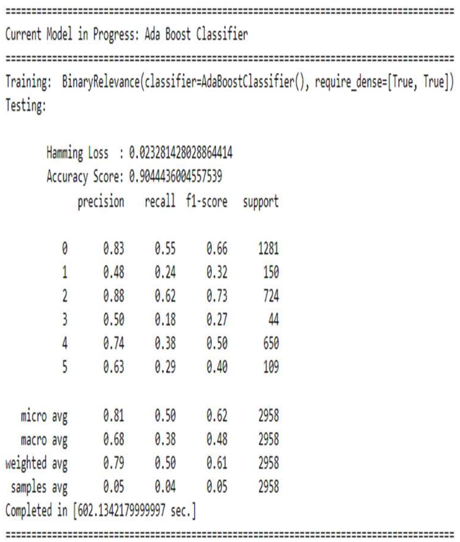
```
================================================================================
Current Model in Progress: MultinomialNB
================================================================================
Training:  BinaryRelevance(classifier=MultinomialNB(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.024091657171793898
        Accuracy Score: 0.9074060007595898
              precision    recall  f1-score   support

           0       0.94      0.48      0.63      1281
           1       1.00      0.01      0.01       150
           2       0.93      0.45      0.60       724
           3       0.00      0.00      0.00        44
           4       0.84      0.35      0.49       650
           5       0.00      0.00      0.00       109

   micro avg       0.91      0.39      0.55      2958
   macro avg       0.62      0.21      0.29      2958
weighted avg       0.87      0.39      0.53      2958
 samples avg       0.04      0.03      0.04      2958
Completed in [4.9532202999998844 sec.]
================================================================================
```



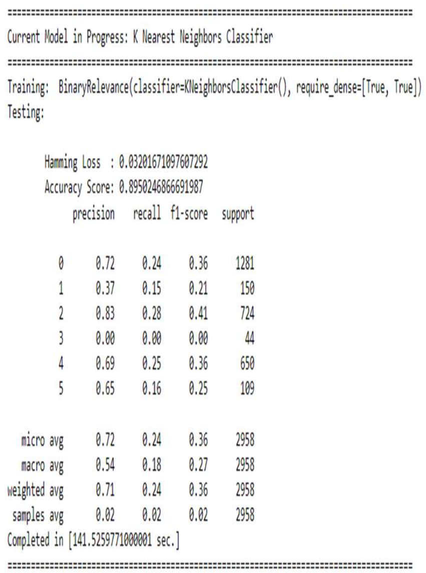
```
================================================================================
Current Model in Progress: Logistic Regression
================================================================================
Training:  BinaryRelevance(classifier=LogisticRegression(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.021939486010887455
        Accuracy Score: 0.9128750474743639
              precision    recall  f1-score   support

           0       0.94      0.53      0.67      1281
           1       0.60      0.18      0.28       150
           2       0.96      0.54      0.69       724
           3       0.00      0.00      0.00        44
           4       0.80      0.42      0.56       650
           5       0.91      0.09      0.17       109

   micro avg       0.90      0.46      0.61      2958
   macro avg       0.70      0.29      0.39      2958
weighted avg       0.88      0.46      0.60      2958
 samples avg       0.05      0.04      0.04      2958
Completed in [31.153639699999985 sec.]
================================================================================
```



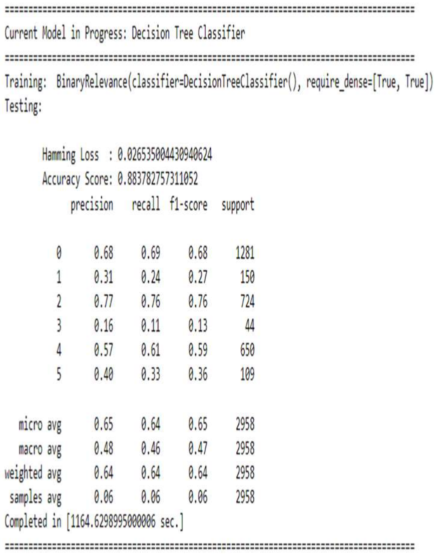
```
================================================================================
Current Model in Progress: Random Forest Classifier
================================================================================
Training:  BinaryRelevance(classifier=RandomForestClassifier(), require_dense=[True,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

        Hamming Loss  : 0.020078490948221294
        Accuracy Score: 0.912419293581466
              precision    recall  f1-score   support

           0       0.86      0.64      0.74      1281
           1       0.29      0.03      0.05       150
           2       0.89      0.72      0.80       724
           3       0.00      0.00      0.00        44
           4       0.73      0.53      0.62       650
           5       0.92      0.11      0.20       109

   micro avg       0.83      0.58      0.68      2958
   macro avg       0.62      0.34      0.40      2958
weighted avg       0.80      0.58      0.66      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [1020.8984025 sec.]
================================================================================
```



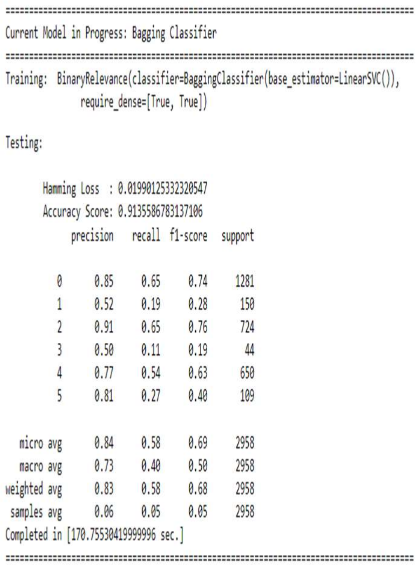
```
==========================================================================================
Current Model in Progress: Ada Boost Classifier
==========================================================================================
Training:  BinaryRelevance(classifier=AdaBoostClassifier(), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.023281428028864414
        Accuracy Score: 0.9044436004557539
              precision    recall  f1-score   support

           0       0.83      0.55      0.66      1281
           1       0.48      0.24      0.32       150
           2       0.88      0.62      0.73       724
           3       0.50      0.18      0.27        44
           4       0.74      0.38      0.50       650
           5       0.63      0.29      0.40       109

   micro avg       0.81      0.50      0.62      2958
   macro avg       0.68      0.38      0.48      2958
weighted avg       0.79      0.50      0.61      2958
 samples avg       0.05      0.04      0.05      2958
Completed in [602.1342179999997 sec.]
==========================================================================================
```



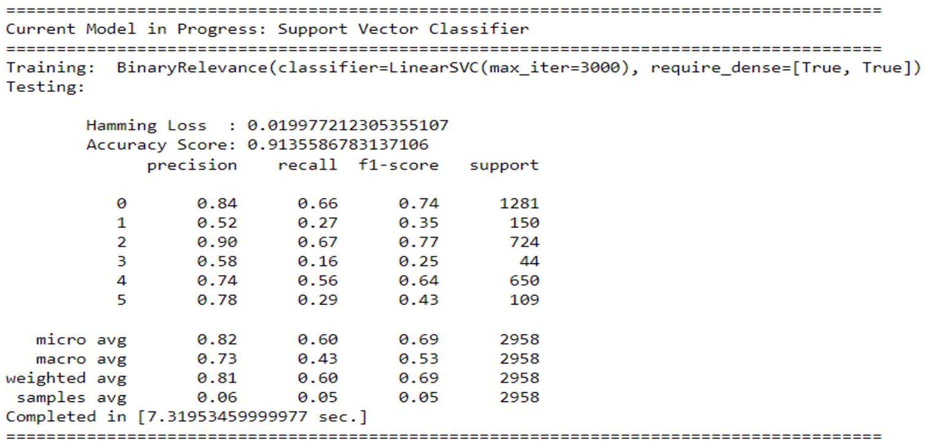
```
==========================================================================================
Current Model in Progress: K Nearest Neighbors Classifier
==========================================================================================
Training:  BinaryRelevance(classifier=KNeighborsClassifier(),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

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

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

   micro avg       0.72      0.24      0.36      2958
   macro avg       0.54      0.18      0.27      2958
weighted avg       0.71      0.24      0.36      2958
 samples avg       0.02      0.02      0.02      2958
Completed in [141.5259771000001 sec.]
==========================================================================================
```



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

        Hamming Loss  : 0.026535004430940624
        Accuracy Score: 0.883782757311052
              precision    recall  f1-score   support

           0       0.68      0.69      0.68      1281
           1       0.31      0.24      0.27       150
           2       0.77      0.76      0.76       724
           3       0.16      0.11      0.13        44
           4       0.57      0.61      0.59       650
           5       0.40      0.33      0.36       109

   micro avg       0.65      0.64      0.65      2958
   macro avg       0.48      0.46      0.47      2958
weighted avg       0.64      0.64      0.64      2958
 samples avg       0.06      0.06      0.06      2958
Completed in [1164.6298995000006 sec.]
==========================================================================================
```



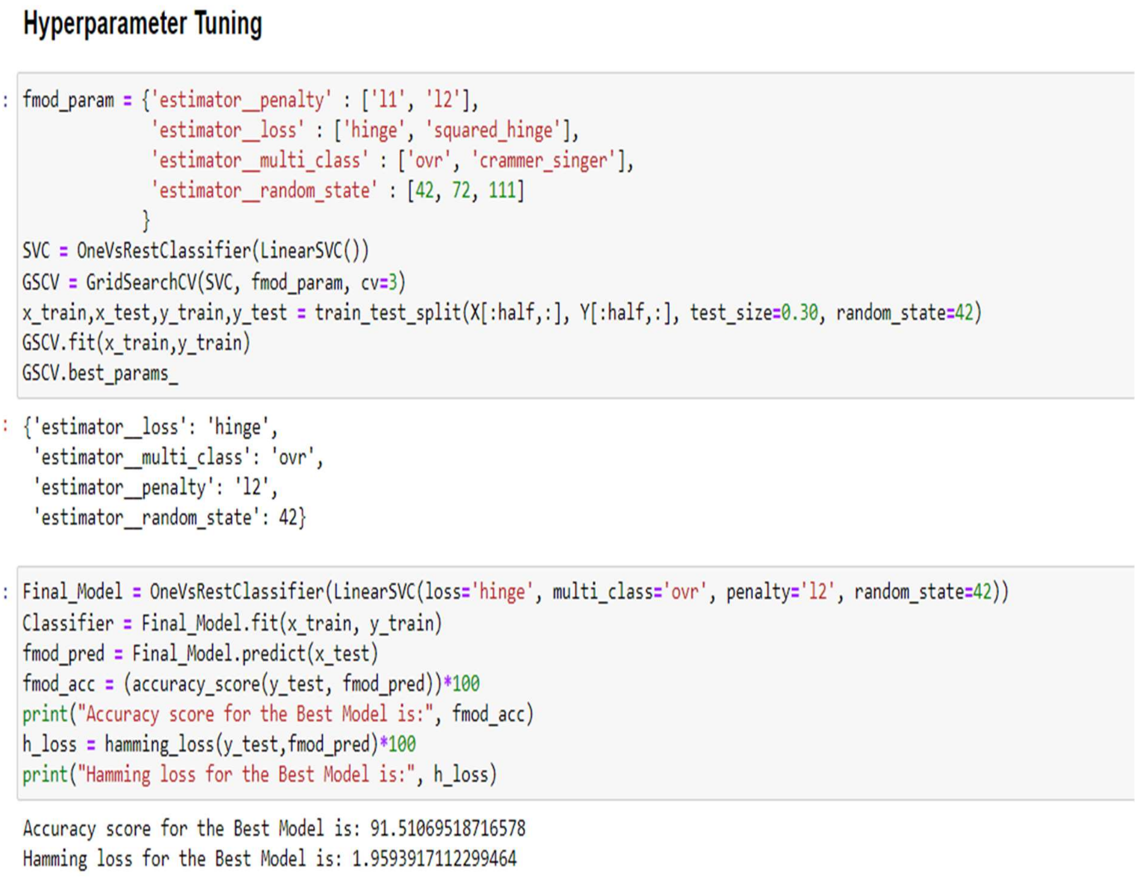
```
==========================================================================================
Current Model in Progress: Bagging Classifier
==========================================================================================
Training:  BinaryRelevance(classifier=BaggingClassifier(base_estimator=LinearSVC()),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

Testing:

        Hamming Loss  : 0.01990125332320547
        Accuracy Score: 0.9135586783137106
              precision    recall  f1-score   support

           0       0.85      0.65      0.74      1281
           1       0.52      0.19      0.28       150
           2       0.91      0.65      0.76       724
           3       0.50      0.11      0.19        44
           4       0.77      0.54      0.63       650
           5       0.81      0.27      0.40       109

   micro avg       0.84      0.58      0.69      2958
   macro avg       0.73      0.40      0.50      2958
weighted avg       0.83      0.58      0.68      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [170.75530419999996 sec.]
==========================================================================================
```



```
==========================================================================================
Current Model in Progress: Support Vector Classifier
==========================================================================================
Training:  BinaryRelevance(classifier=LinearSVC(max_iter=3000), require_dense=[True, True])
Testing:

        Hamming Loss  : 0.019977212305355107
        Accuracy Score: 0.9135586783137106
              precision    recall  f1-score   support

           0       0.84      0.66      0.74      1281
           1       0.52      0.27      0.35       150
           2       0.90      0.67      0.77       724
           3       0.58      0.16      0.25        44
           4       0.74      0.56      0.64       650
           5       0.78      0.29      0.43       109

   micro avg       0.82      0.60      0.69      2958
   macro avg       0.73      0.43      0.53      2958
weighted avg       0.81      0.60      0.69      2958
 samples avg       0.06      0.05      0.05      2958
Completed in [7.31953459999977 sec.]
==========================================================================================
```

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

Hyperparameter Tuning:



Hyperparameter Tuning

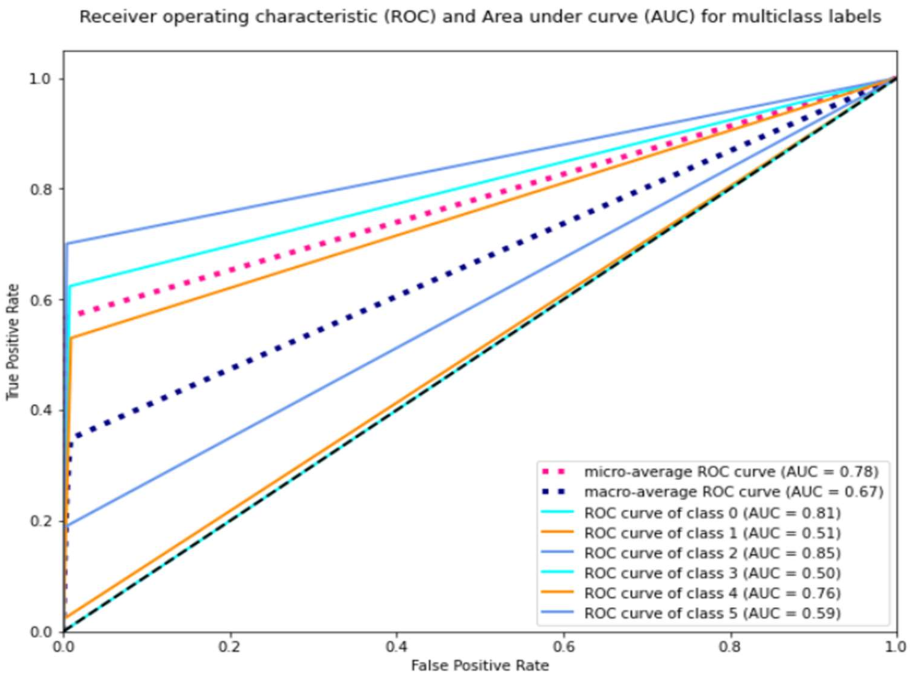
```
fmod_param = {'estimator__penalty' : ['l1', 'l2'],
              'estimator__loss' : ['hinge', 'squared_hinge'],
              'estimator__multi_class' : ['ovr', 'crammer_singer'],
              'estimator__random_state' : [42, 72, 111]
             }
SVC = OneVsRestClassifier(LinearSVC())
GSCV = 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}
```

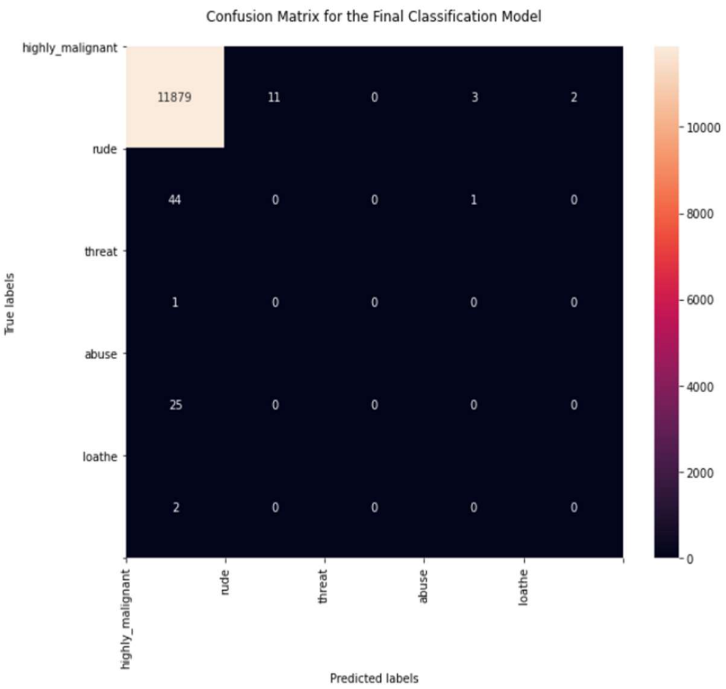
```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge', multi_class='ovr', penalty='l2', random_state=42))
Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)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



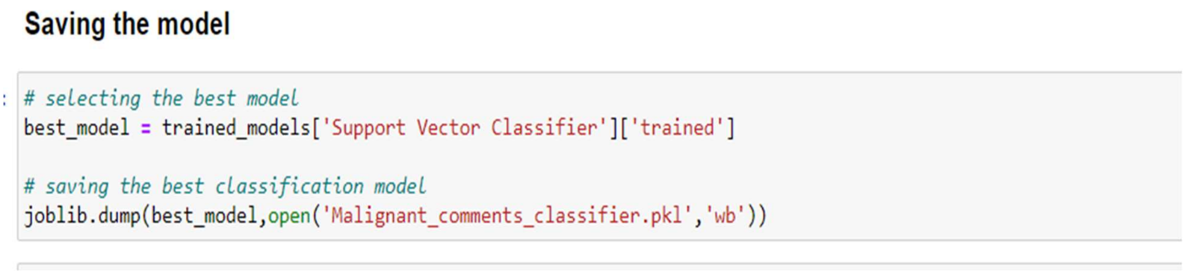
Confusion Matrix:



	highly_malignant	rude	threat	abuse	loathe
highly_malignant	11879	11	0	3	2
rude	44	0	0	1	0
threat	1	0	0	0	0
abuse	25	0	0	0	0
loathe	2	0	0	0	0

From the above, we came to know that the number of times we get correct outputs and the number of times our final model missed to provide the correct prediction (depicting in the black boxes).

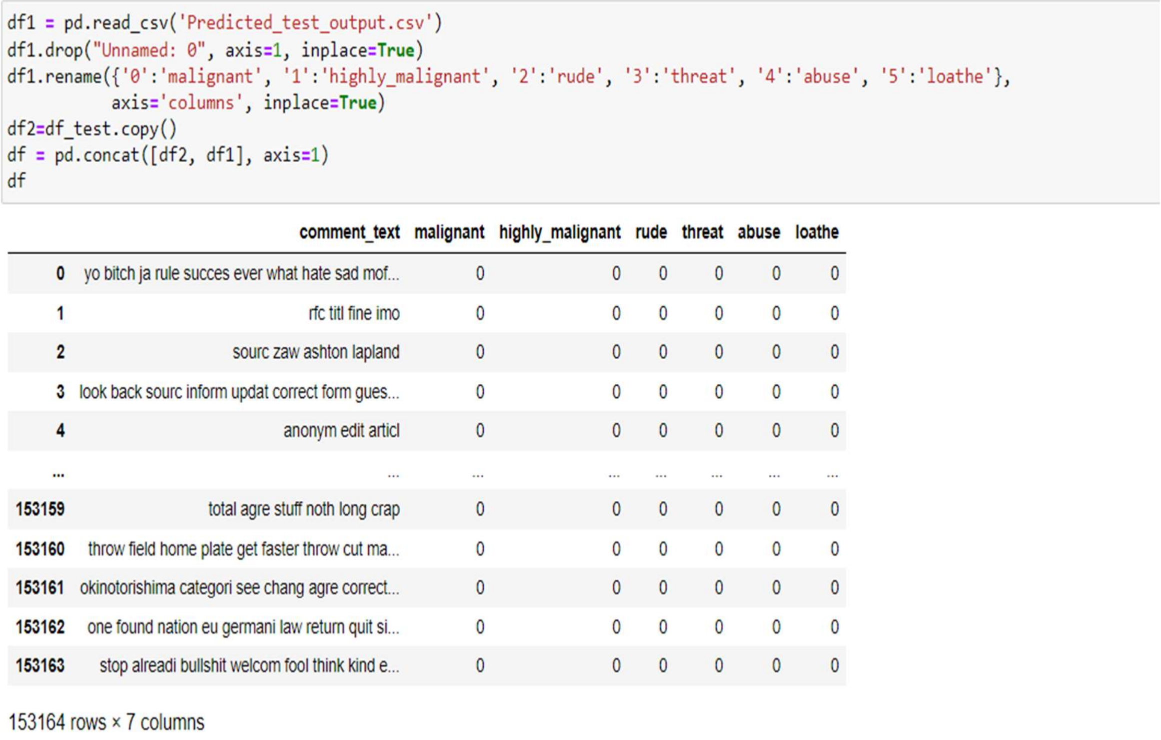
Saving the Model:



Saving the 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'))
```

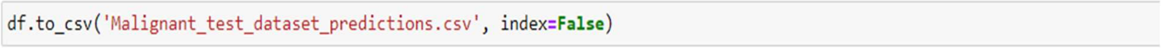
Predicted Values:



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

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

153164 rows × 7 columns



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

----- --:-- -----

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:

  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. Our study helps the online forums and social media to induce a ban to profanity or usage of profanity over these forums.

- Learning Outcomes:
  Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stop words. 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.

- Limitations:
  1. Imbalanced dataset and bad comment texts.
  2. Good parameters could not be obtained using hyperparameter tuning as time was consumed more.

- Areas of Improvement:
  1. Could be provided with a good dataset which does not take more time.
  2. Less time complexity.
  3. Providing a proper balanced dataset with less errors.

----- --:-- ----- ----- --:-- ----- ❄ ----- --:-- ---------- --:-- -----