# Malignant Comments Classifier

***- Comment 1 Example***

★★★★★

I was able to post the above list so quickly because I already had it in a text file in my hard drive  I've been meaning to get around to updating the sound list for some time now.

***- Comment 2 Example***

★★★★★

In other words, you're too lazy to actually point anything out. Until you change that approach, the tag goes.

***- Comment 3 Example***

★★★★★

I went there around the same time he did, and that certainly was not the case at the time. Later on they stopped taking children from such a young age.

# 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 behavior.

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.

# INTRODUCTION

## Business Problem Framing

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 their thoughts freely without any partiality and overcome all the races 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.

# INTRODUCTION

## Conceptual Background of the Domain Problem

In this huge online platform or an online community, there are some people or some motivated mob wilfully bully others to make them not share their thought in a rightful way. They bully others with foul language which among the civilized society is seen as ignominy. And when innocent individuals are being bullied by these mobs 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 stop these mobs from using the foul language in an online community or even block them or block them from using this foul language.

## Review of Literature

The purpose of the literature review is to identify

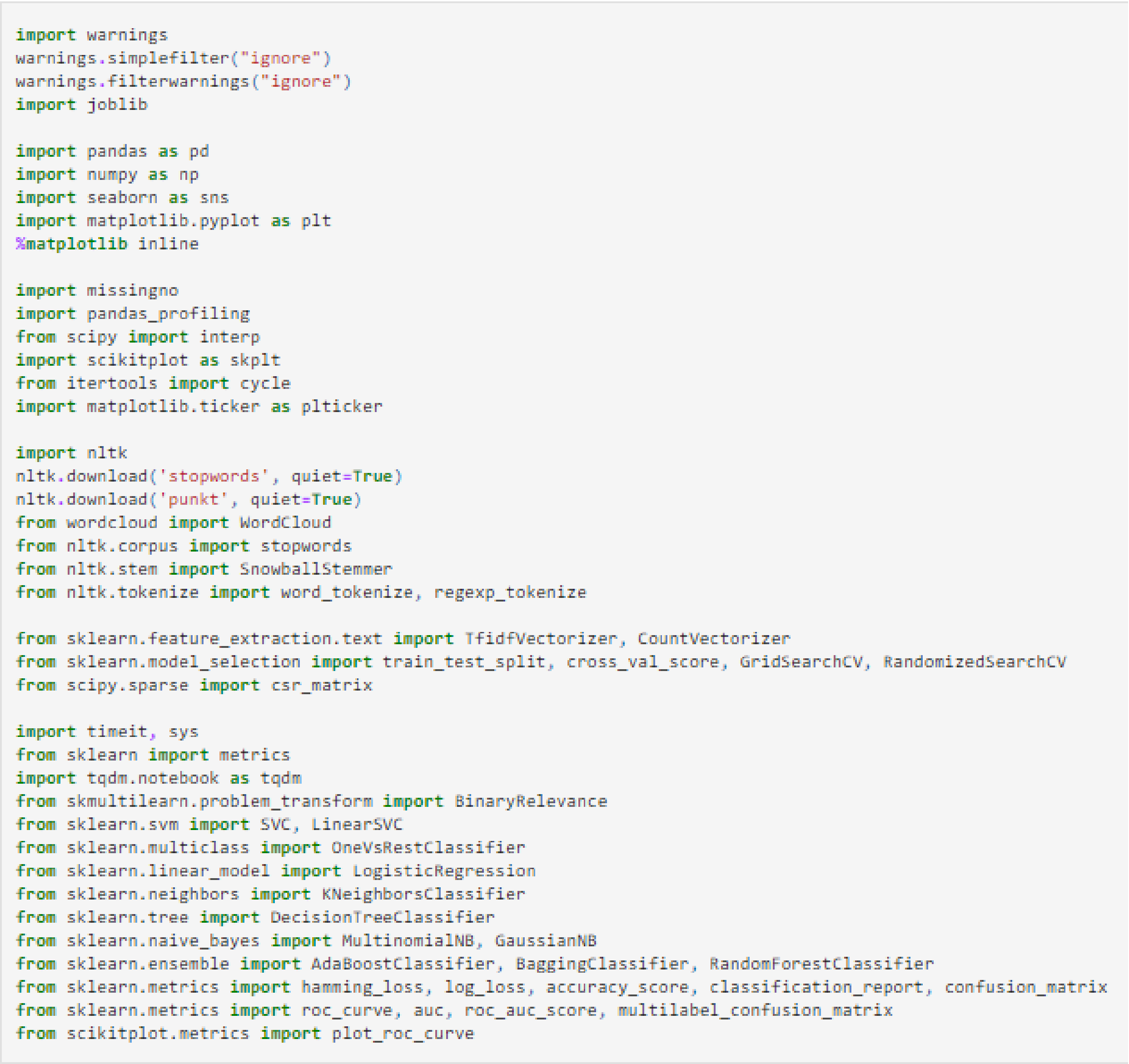
1. The foul words or foul statements are being used.
2. Stop the people from using these foul languages in an online public forums.

# CLASSIFIER

## 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
```

# CLASSIFIER

## Analytical Problem Framing

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

Here in this project, we have been provided with two datasets namely train and test CSV files. I will build a machine learning model by using NLP using a 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 hyperparameters then will select the best model 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. The findings of the comparison indicate the significance of the proposed features in cyberbullying detection.

# CLASSIFIER

## 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

1. id
2. comment_text
3. malignant
4. highly_malignant
5. rude
6. threat
7. abuse
8. 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.

# CLASSIFIER

## Data Sources and their formats

The data set includes

1. Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
2. Highly Malignant: It denotes comments that are highly malignant and hurtful.
3. Rude: It denotes comments that are very rude and offensive.
4. Threat: It contains an indication of the comments that are giving any threat to someone.
5. Abuse: It is for comments that are abusive in nature.
6. Loathe: It describes the comments which are hateful and loathing in nature.
7. ID: It includes unique Ids associated with each comment text given.
8. Comment text: This column contains the comments extracted from various social media platforms.

This project is more about exploration, feature engineering, and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do a good amount of data exploration and derive some interesting features using the comments text column available. Need to build a model that can differentiate between comments and their categories.

# CLASSIFIER

## Data Preprocessing

The following preprocessing 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 a 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

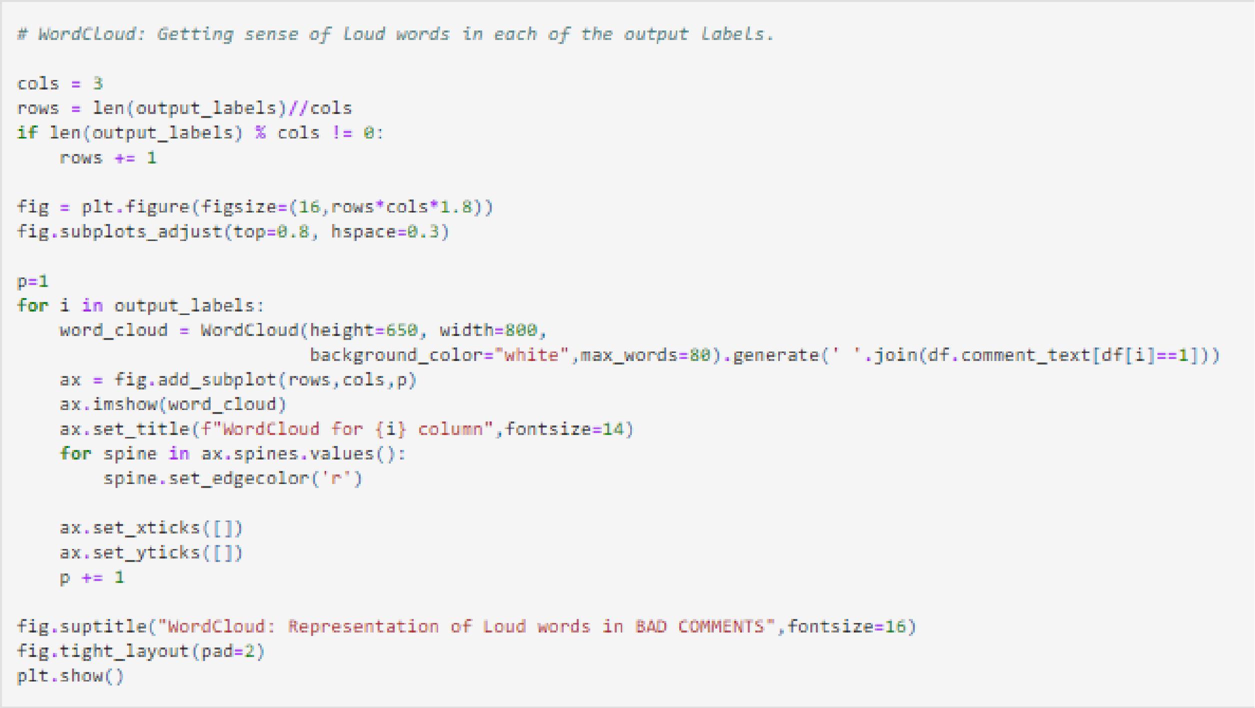
**Data Inputs- Logic- Output Relationships:**

Have analyzed the input-output logic with a word cloud and I have word clouded the sentence that is 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.

# CLASSIFIER

## Data Preprocessing

### Data Inputs- Logic- Output Relationships:



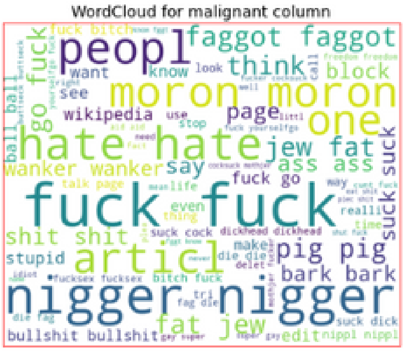
```
# WordCloud: Getting sense of loud words in each of the output labels.

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()
```

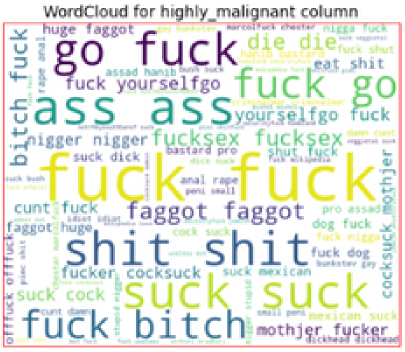
WordCloud: Representation of Loud words in BAD COMMENTS

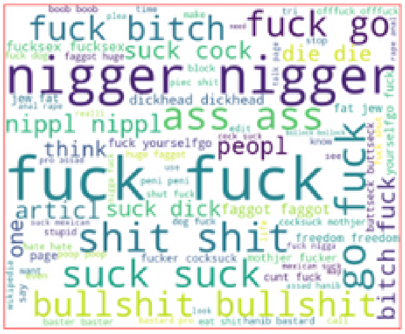
# CLASSIFIER

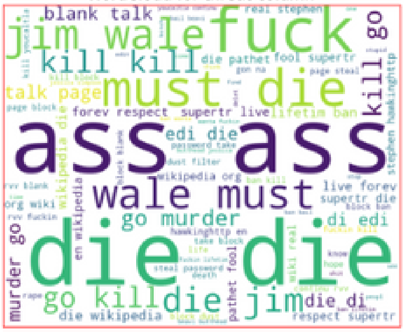
## Data Preprocessing

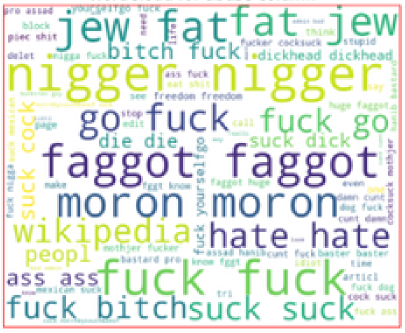
**Data Inputs- Logic- Output Relationships:**

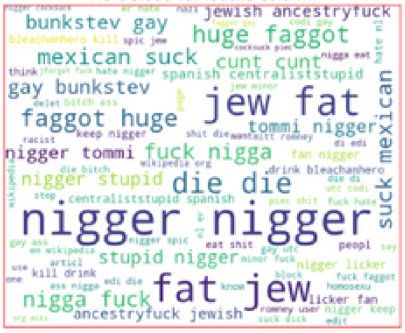

WordCloud for malignant column


WordCloud for highly_malignant column


WordCloud for rude column


WordCloud for threat column


WordCloud for abuse column


WordCloud for loathe column

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.

# CLASSIFIER

## Data Preprocessing

**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.

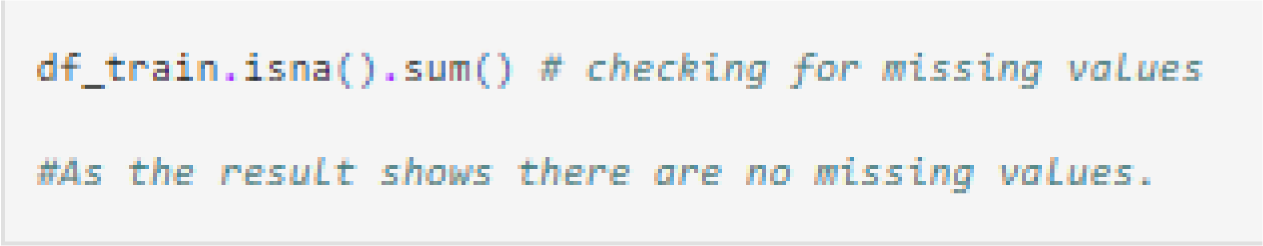
Cyberbullying is a very serious issue affecting not just the young victims, but also the victims' families, the bully, and those who witness instances of cyberbullying. However, the effect of cyberbullying can be most detrimental to the victim, of course, as they may experience a number of emotional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

# CLASSIFIER

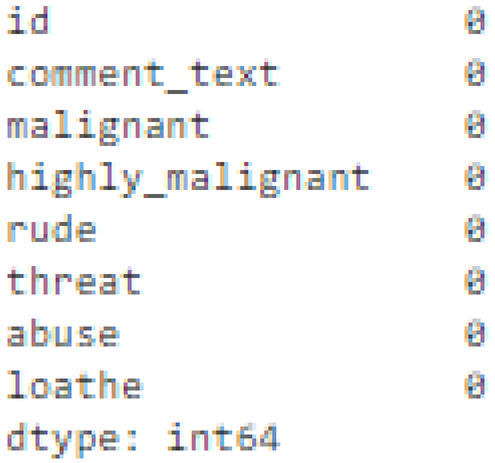
## Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

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.



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

#As the result shows there are no missing values.
```

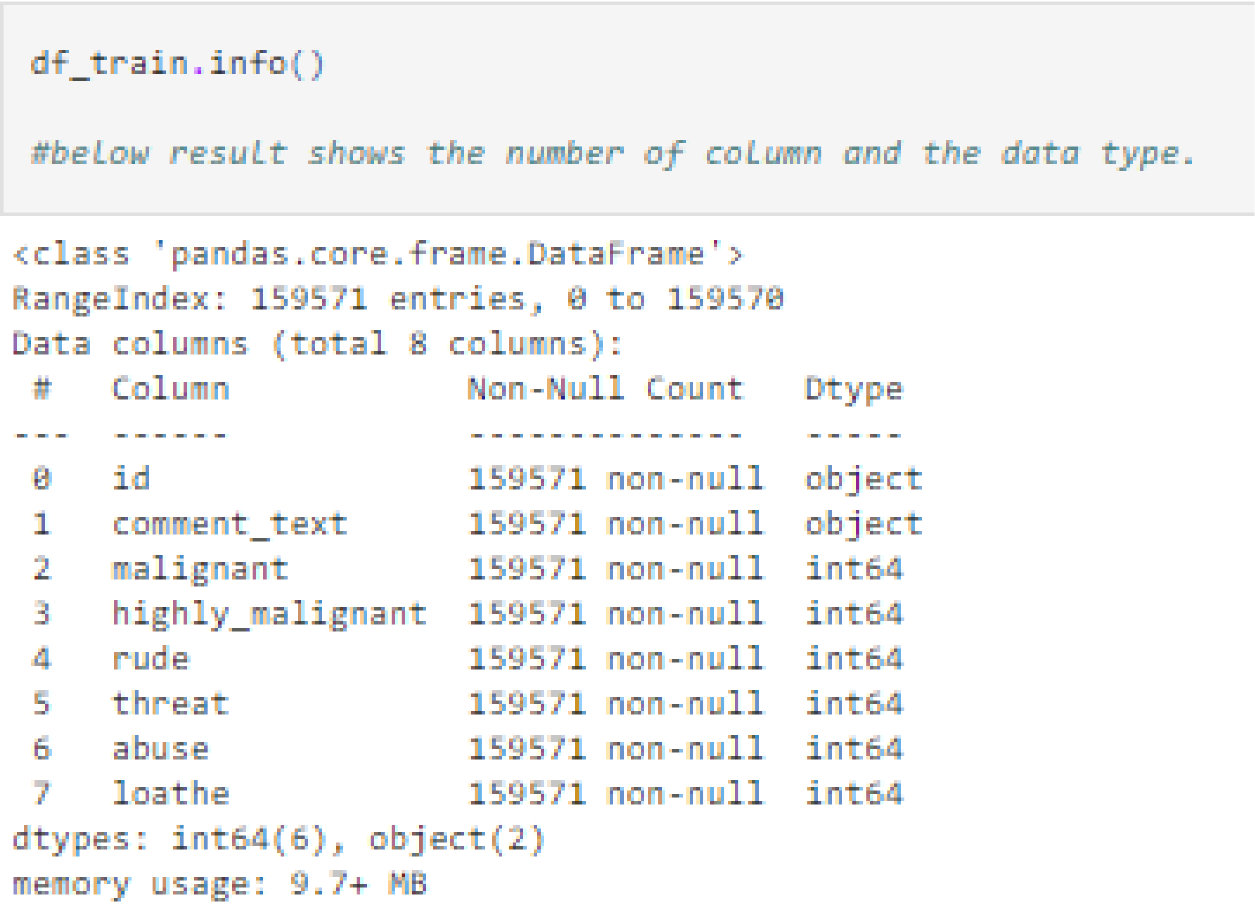


```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

# CLASSIFIER

## Model/s Development and Evaluation

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.



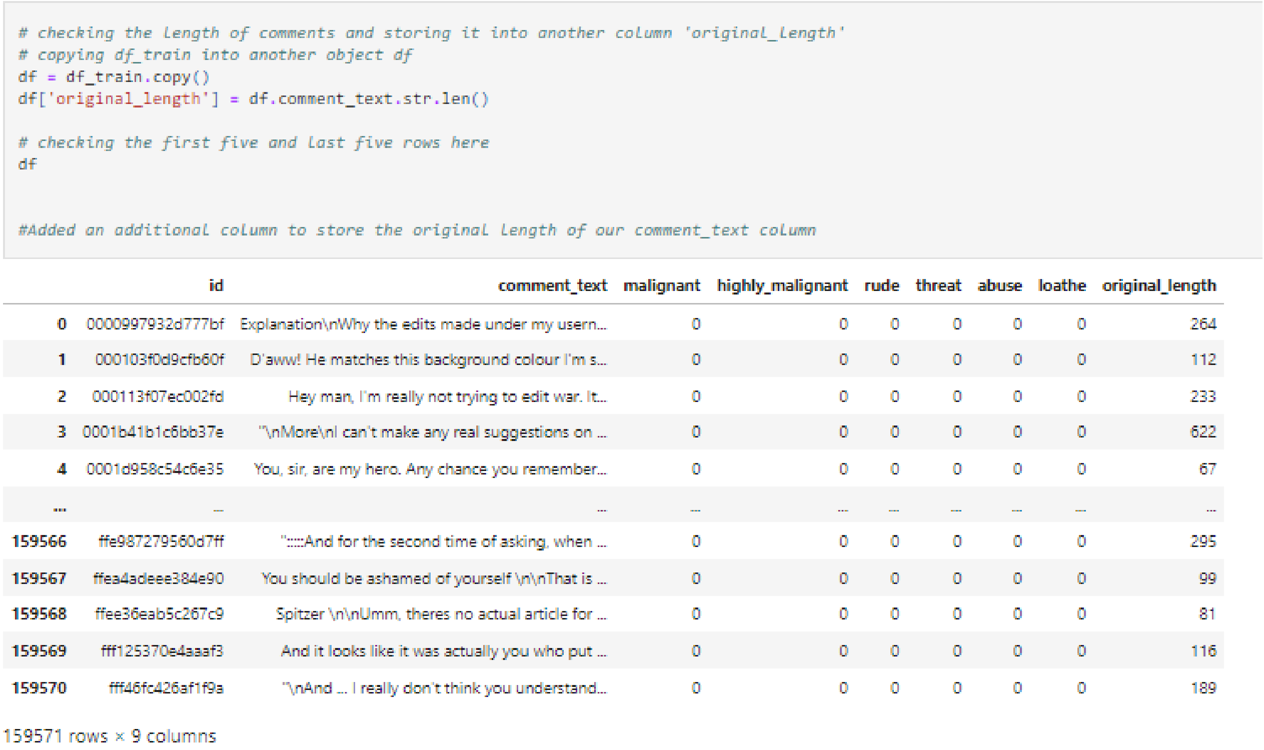
```
df_train.info()

#below result shows the number of column and the data type.
```

```
<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
```

# CLASSIFIER

## Model/s Development and Evaluation

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


#Added an additional column to store the original length of our comment_text column
```

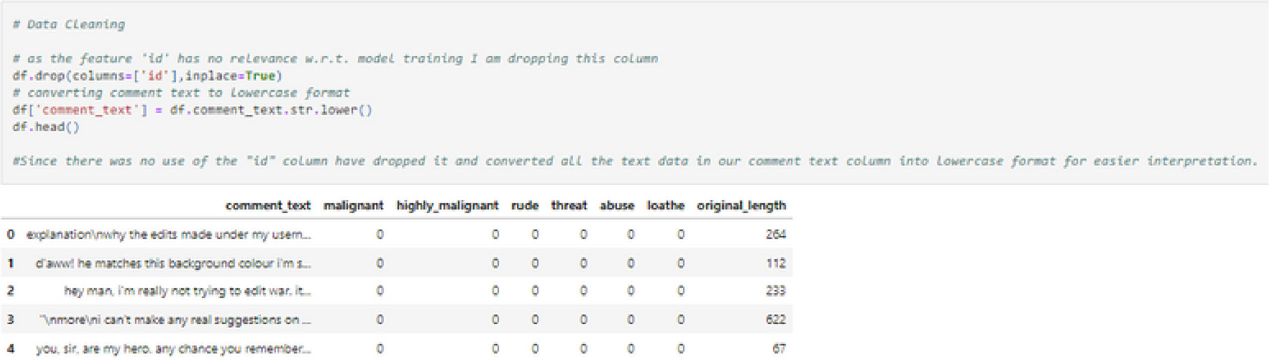
| | id | comment_text | malignant | highly_malignant | rude | threat | abuse | loathe | original_length |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0000997932d777bf | Explanation\nWhy the edits made under my usern... | 0 | 0 | 0 | 0 | 0 | 0 | 264 |
| **1** | 000103f0d9cfb60f | D'aww! He matches this background colour I'm s... | 0 | 0 | 0 | 0 | 0 | 0 | 112 |
| **2** | 000113f07ec002fd | Hey man, I'm really not trying to edit war. It... | 0 | 0 | 0 | 0 | 0 | 0 | 233 |
| **3** | 0001b41b1c6bb37e | "\nMore\nI can't make any real suggestions on ... | 0 | 0 | 0 | 0 | 0 | 0 | 622 |
| **4** | 0001d958c54c6e35 | You, sir, are my hero. Any chance you remember... | 0 | 0 | 0 | 0 | 0 | 0 | 67 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **159566** | ffe987279560d7ff | ":::::And for the second time of asking, when ... | 0 | 0 | 0 | 0 | 0 | 0 | 295 |
| **159567** | ffea4adeee384e90 | You should be ashamed of yourself \n\nThat is ... | 0 | 0 | 0 | 0 | 0 | 0 | 99 |
| **159568** | ffee36eab5c267c9 | Spitzer \n\nUmm, theres no actual article for ... | 0 | 0 | 0 | 0 | 0 | 0 | 81 |
| **159569** | fff125370e4aaaf3 | And it looks like it was actually you who put ... | 0 | 0 | 0 | 0 | 0 | 0 | 116 |
| **159570** | fff46fc426af1f9a | "\nAnd ... I really don't think you understand... | 0 | 0 | 0 | 0 | 0 | 0 | 189 |

159571 rows × 9 columns

# CLASSIFIER

## Model/s Development and Evaluation

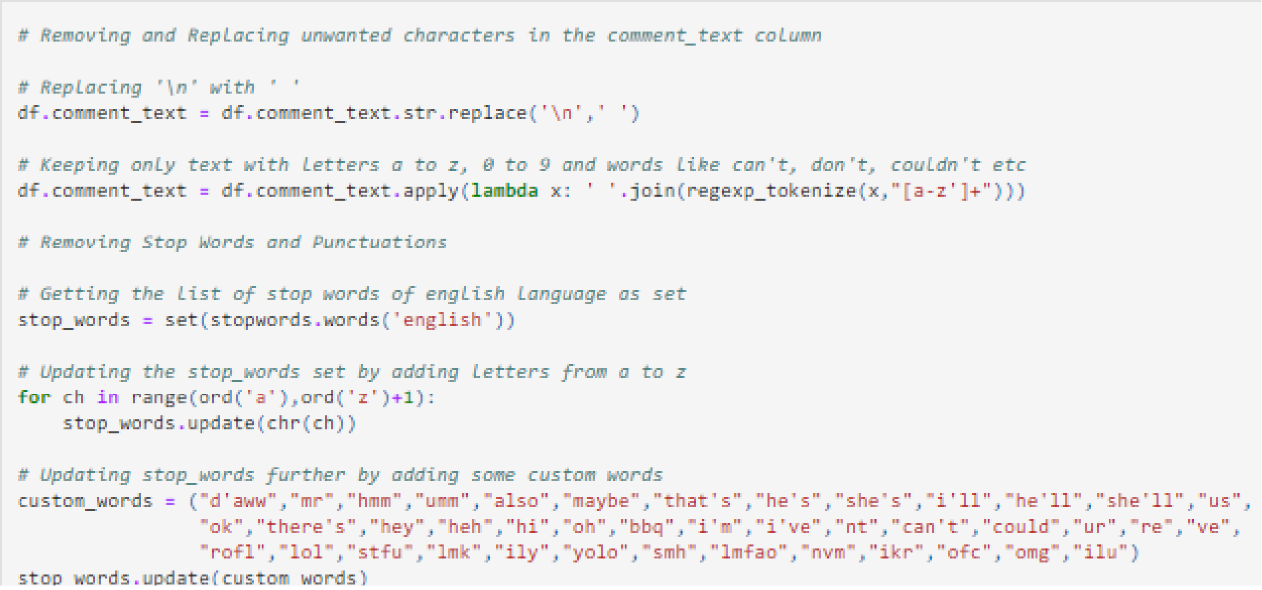
Since there was no use of the "id" column I have dropped it and converted all the text data in our comment text column into lowercase format for easier interpretation.



```
# Data Cleaning

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

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

| | 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 |

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).

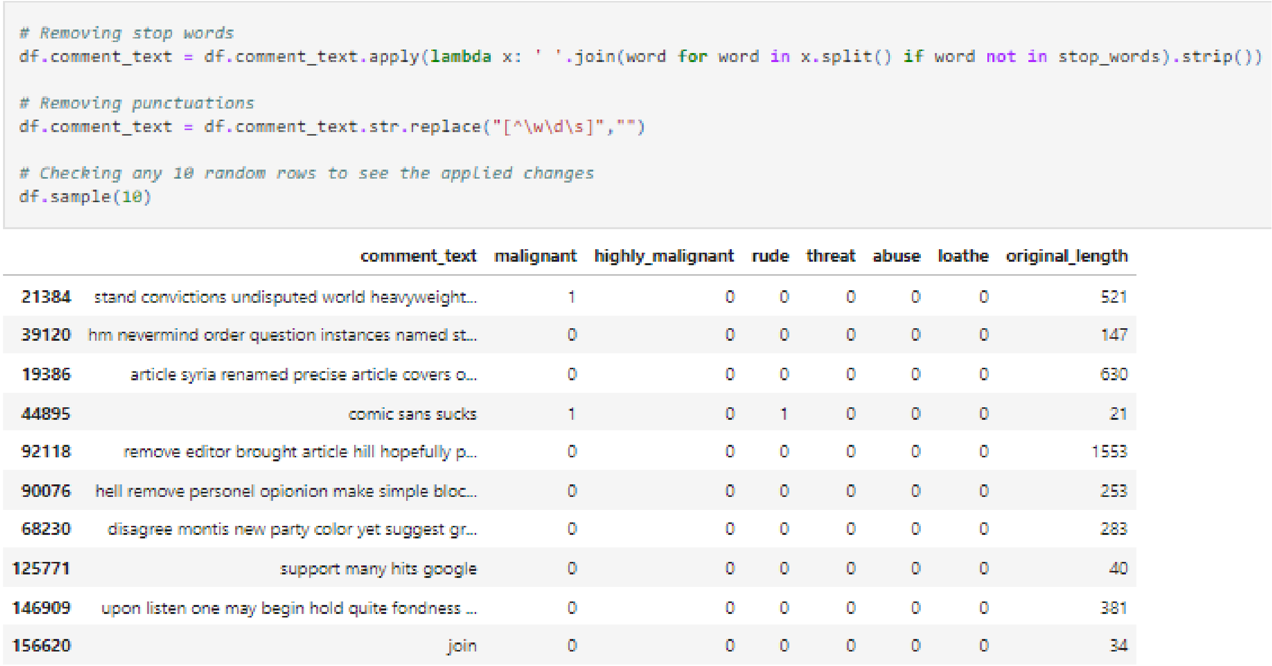


```
# Removing and Replacing unwanted characters in the comment_text column

# 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)
```

# CLASSIFIER

## Model/s Development and Evaluation

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



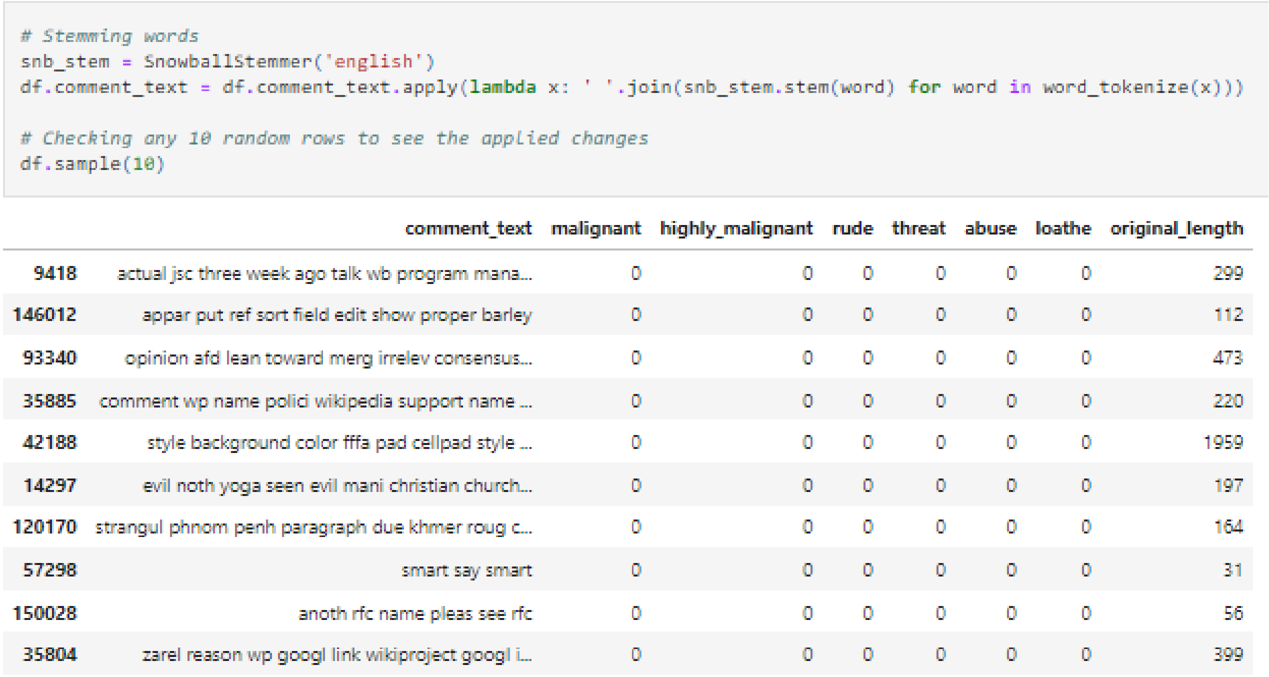
```
# Removing stop words
df.comment_text = df.comment_text.apply(lambda x: ' '.join(word for word in x.split() if word not in stop_words).strip())

# 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)
```

| | comment_text | malignant | highly_malignant | rude | threat | abuse | loathe | original_length |
|---|---|---|---|---|---|---|---|---|
| **21384** | stand convictions undisputed world heavyweight... | 1 | 0 | 0 | 0 | 0 | 0 | 521 |
| **39120** | hm nevermind order question instances named st... | 0 | 0 | 0 | 0 | 0 | 0 | 147 |
| **19386** | article syria renamed precise article covers o... | 0 | 0 | 0 | 0 | 0 | 0 | 630 |
| **44895** | comic sans sucks | 1 | 0 | 1 | 0 | 0 | 0 | 21 |
| **92118** | remove editor brought article hill hopefully p... | 0 | 0 | 0 | 0 | 0 | 0 | 1553 |
| **90076** | hell remove personel opionion make simple bloc... | 0 | 0 | 0 | 0 | 0 | 0 | 253 |
| **68230** | disagree montis new party color yet suggest gr... | 0 | 0 | 0 | 0 | 0 | 0 | 283 |
| **125771** | support many hits google | 0 | 0 | 0 | 0 | 0 | 0 | 40 |
| **146909** | upon listen one may begin hold quite fondness ... | 0 | 0 | 0 | 0 | 0 | 0 | 381 |
| **156620** | join | 0 | 0 | 0 | 0 | 0 | 0 | 34 |

# CLASSIFIER

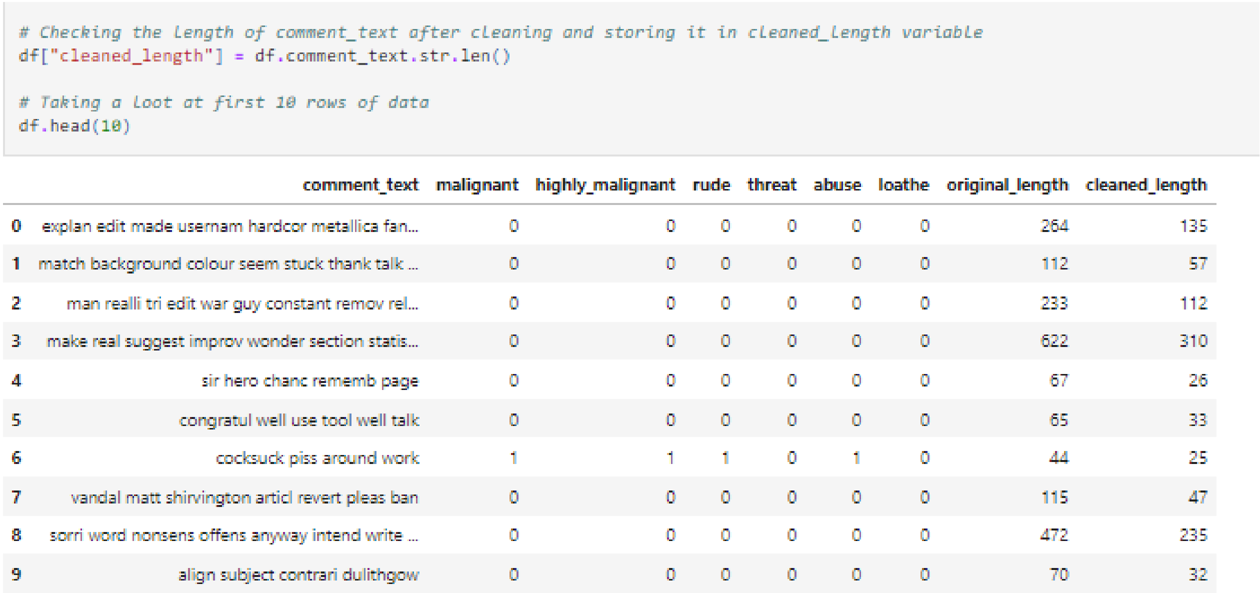
## Model/s Development and Evaluation

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



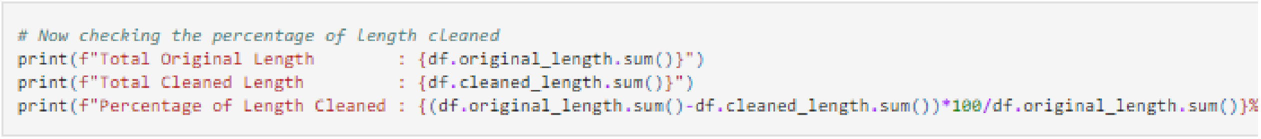
```
# Stemming words
snb_stem = SnowballStemmer('english')
df.comment_text = df.comment_text.apply(lambda x: ' '.join(snb_stem.stem(word) for word in word_tokenize(x)))

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

| | comment_text | malignant | highly_malignant | rude | threat | abuse | loathe | original_length |
|---|---|---|---|---|---|---|---|---|
| 9418 | actual jsc three week ago talk wb program mana... | 0 | 0 | 0 | 0 | 0 | 0 | 299 |
| 146012 | appar put ref sort field edit show proper barley | 0 | 0 | 0 | 0 | 0 | 0 | 112 |
| 93340 | opinion afd lean toward merg irrelev consensus... | 0 | 0 | 0 | 0 | 0 | 0 | 473 |
| 35885 | comment wp name polici wikipedia support name ... | 0 | 0 | 0 | 0 | 0 | 0 | 220 |
| 42188 | style background color fffa pad cellpad style ... | 0 | 0 | 0 | 0 | 0 | 0 | 1959 |
| 14297 | evil noth yoga seen evil mani christian church... | 0 | 0 | 0 | 0 | 0 | 0 | 197 |
| 120170 | strangul phnom penh paragraph due khmer roug c... | 0 | 0 | 0 | 0 | 0 | 0 | 164 |
| 57298 | smart say smart | 0 | 0 | 0 | 0 | 0 | 0 | 31 |
| 150028 | anoth rfc name pleas see rfc | 0 | 0 | 0 | 0 | 0 | 0 | 56 |
| 35804 | zarel reason wp googl link wikiproject googl i... | 0 | 0 | 0 | 0 | 0 | 0 | 399 |



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

# Taking a loot at first 10 rows of data
df.head(10)
```

| | comment_text | malignant | highly_malignant | rude | threat | abuse | loathe | original_length | cleaned_length |
|---|---|---|---|---|---|---|---|---|---|
| 0 | explan edit made usernam hardcor metallica fan... | 0 | 0 | 0 | 0 | 0 | 0 | 264 | 135 |
| 1 | match background colour seem stuck thank talk ... | 0 | 0 | 0 | 0 | 0 | 0 | 112 | 57 |
| 2 | man realli tri edit war guy constant remov rel... | 0 | 0 | 0 | 0 | 0 | 0 | 233 | 112 |
| 3 | make real suggest improv wonder section statis... | 0 | 0 | 0 | 0 | 0 | 0 | 622 | 310 |
| 4 | sir hero chanc rememb page | 0 | 0 | 0 | 0 | 0 | 0 | 67 | 26 |
| 5 | congratul well use tool well talk | 0 | 0 | 0 | 0 | 0 | 0 | 65 | 33 |
| 6 | cocksuck piss around work | 1 | 1 | 1 | 0 | 1 | 0 | 44 | 25 |
| 7 | vandal matt shirvington articl revert pleas ban | 0 | 0 | 0 | 0 | 0 | 0 | 115 | 47 |
| 8 | sorri word nonsens offens anyway intend write ... | 0 | 0 | 0 | 0 | 0 | 0 | 472 | 235 |
| 9 | align subject contrari dulithgow | 0 | 0 | 0 | 0 | 0 | 0 | 70 | 32 |



```
# 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()}%
```

# CLASSIFIER

## Testing of Identified Approaches (Algorithms)

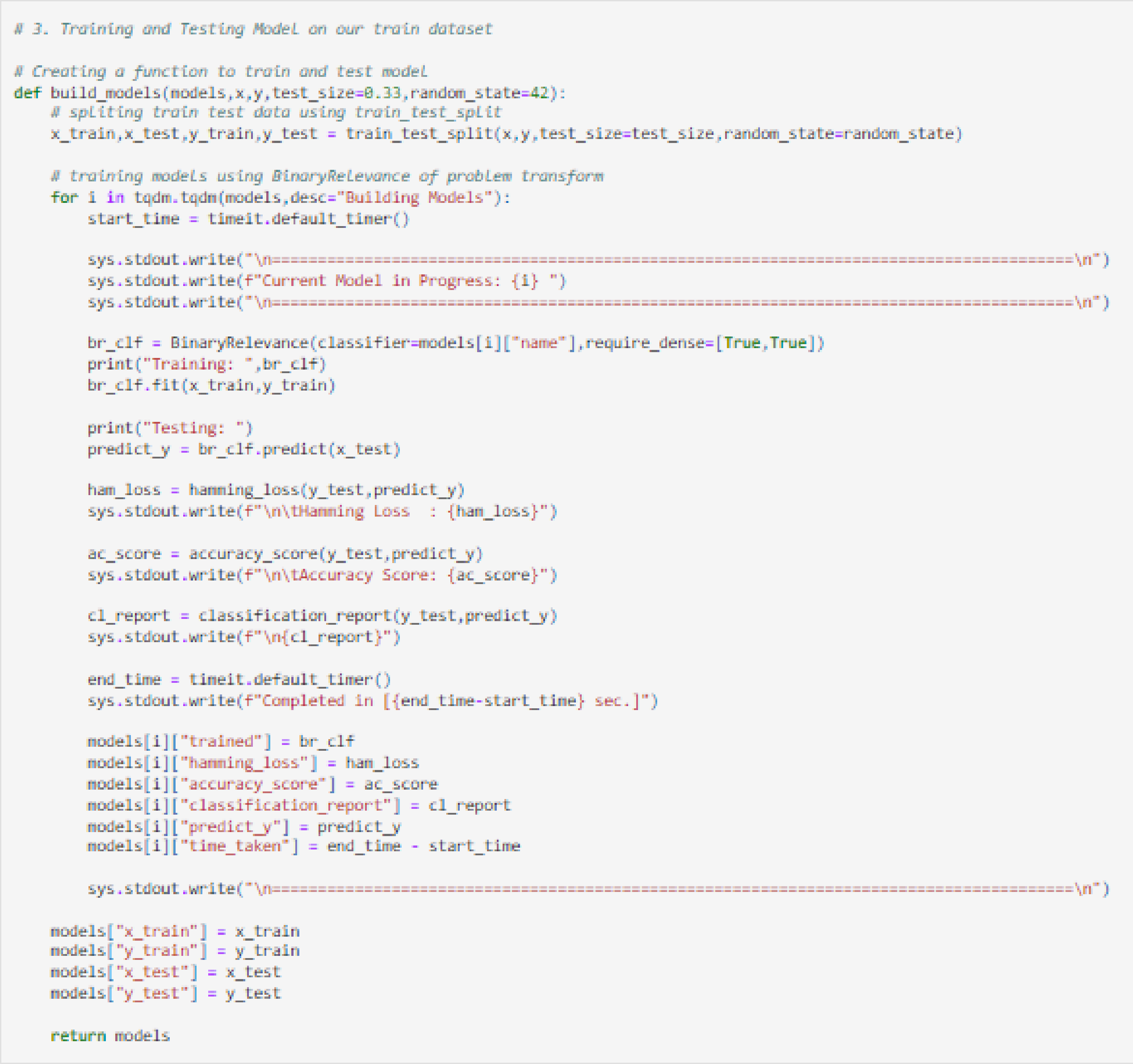
The complete list of all the algorithms used for the training and testing classification model are listed below.

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

# CLASSIFIER

## Run and Evaluate selected models

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

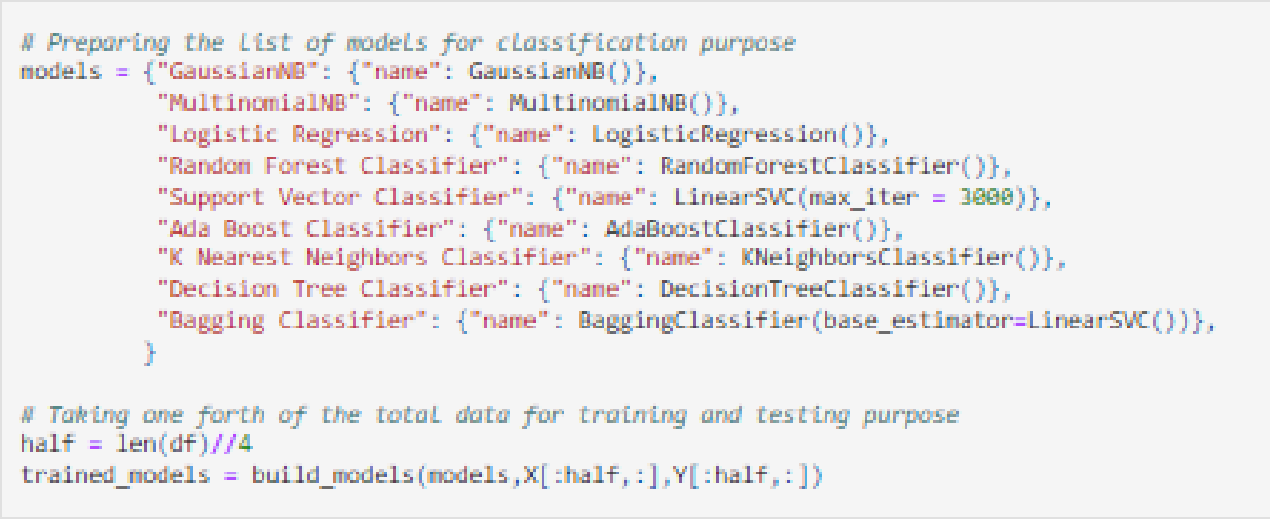


```
# 3. Training and Testing Model on our train dataset

# 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
```

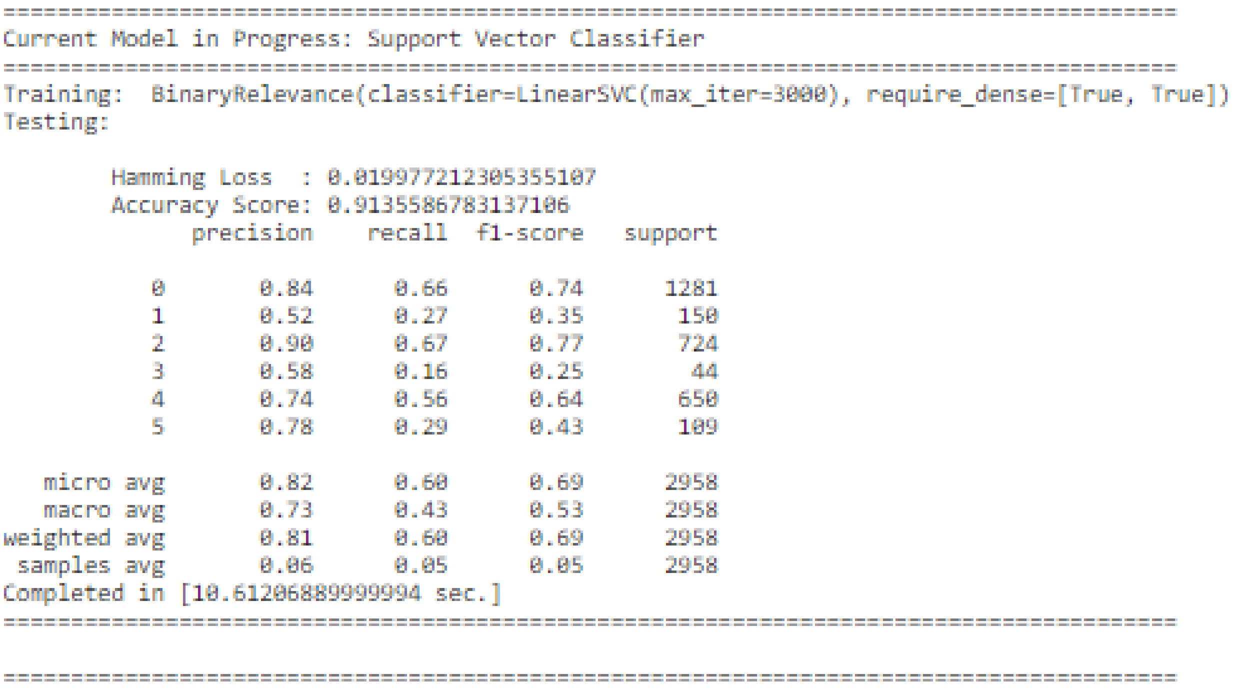
# CLASSIFIER

## Run and Evaluate selected models

Created a classification function that included the evaluation metrics details for the generation of our Classification Machine Learning 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,:])
```

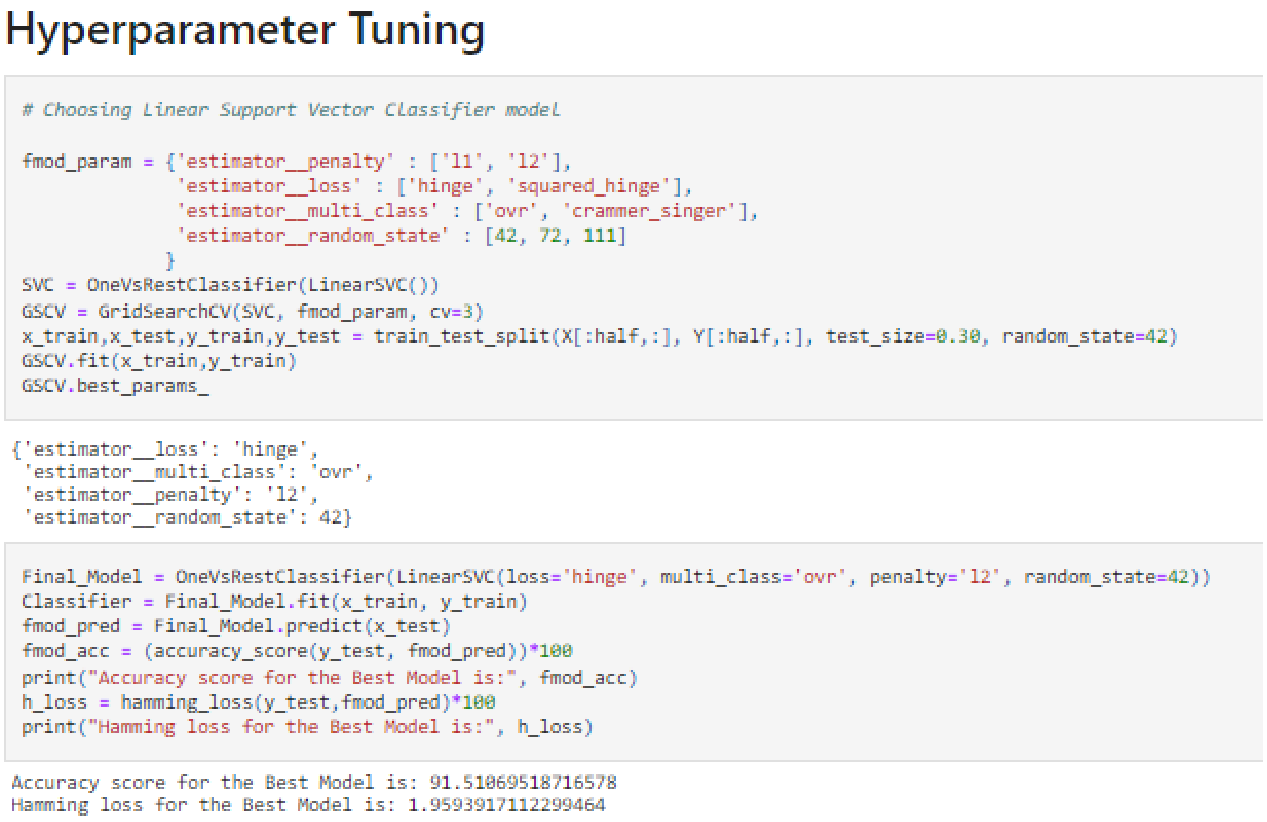


```
=======================================================================================
Current Model in Progress: Support Vector Classifier
=======================================================================================
Training:  BinaryRelevance(classifier=LinearSVC(max_iter=3000), require_dense=[True, 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 [10.612068899999994 sec.]
=======================================================================================

=======================================================================================
```

# CLASSIFIER

## Run and Evaluate selected models

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_
```

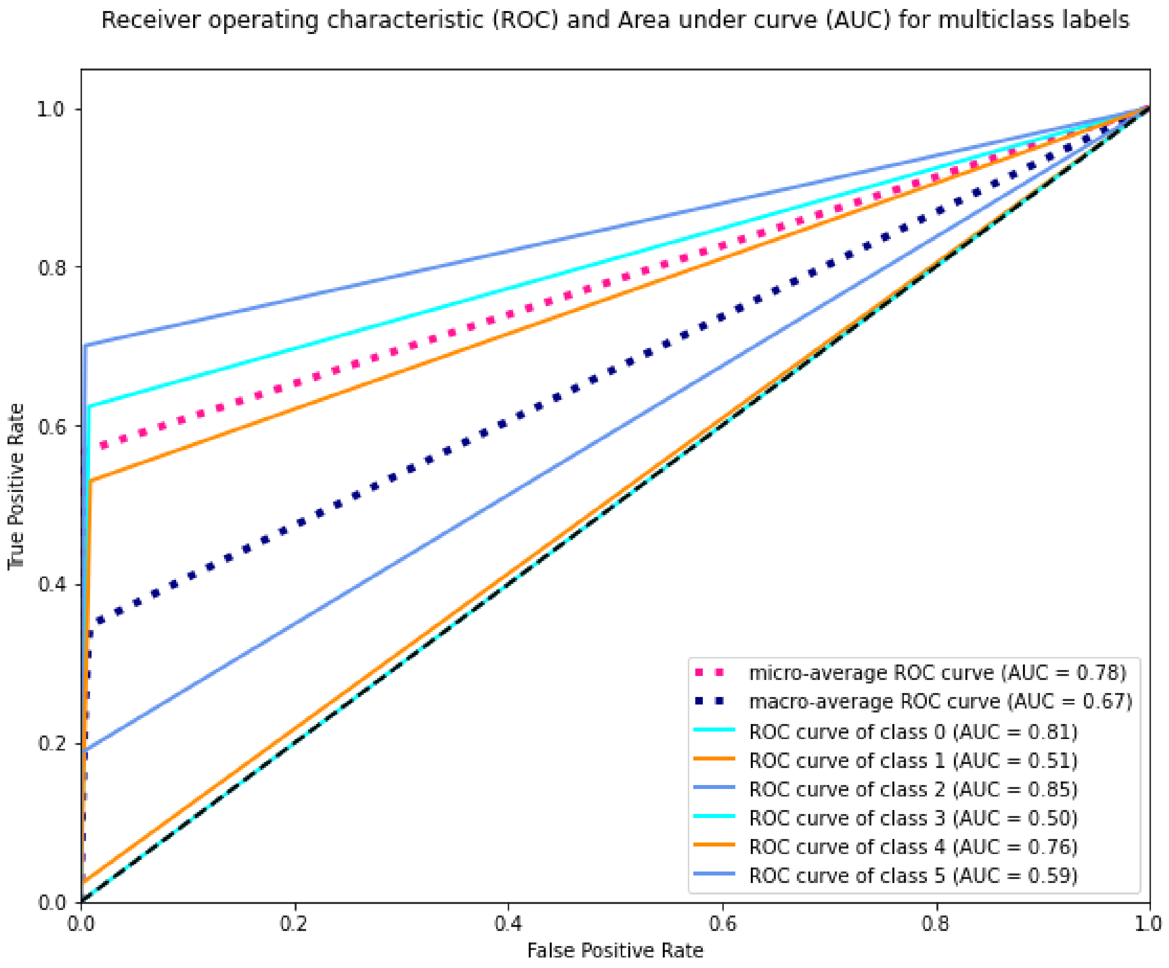
```
{'estimator__loss': 'hinge',
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2',
 'estimator__random_state': 42}
```

```
Final_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
```

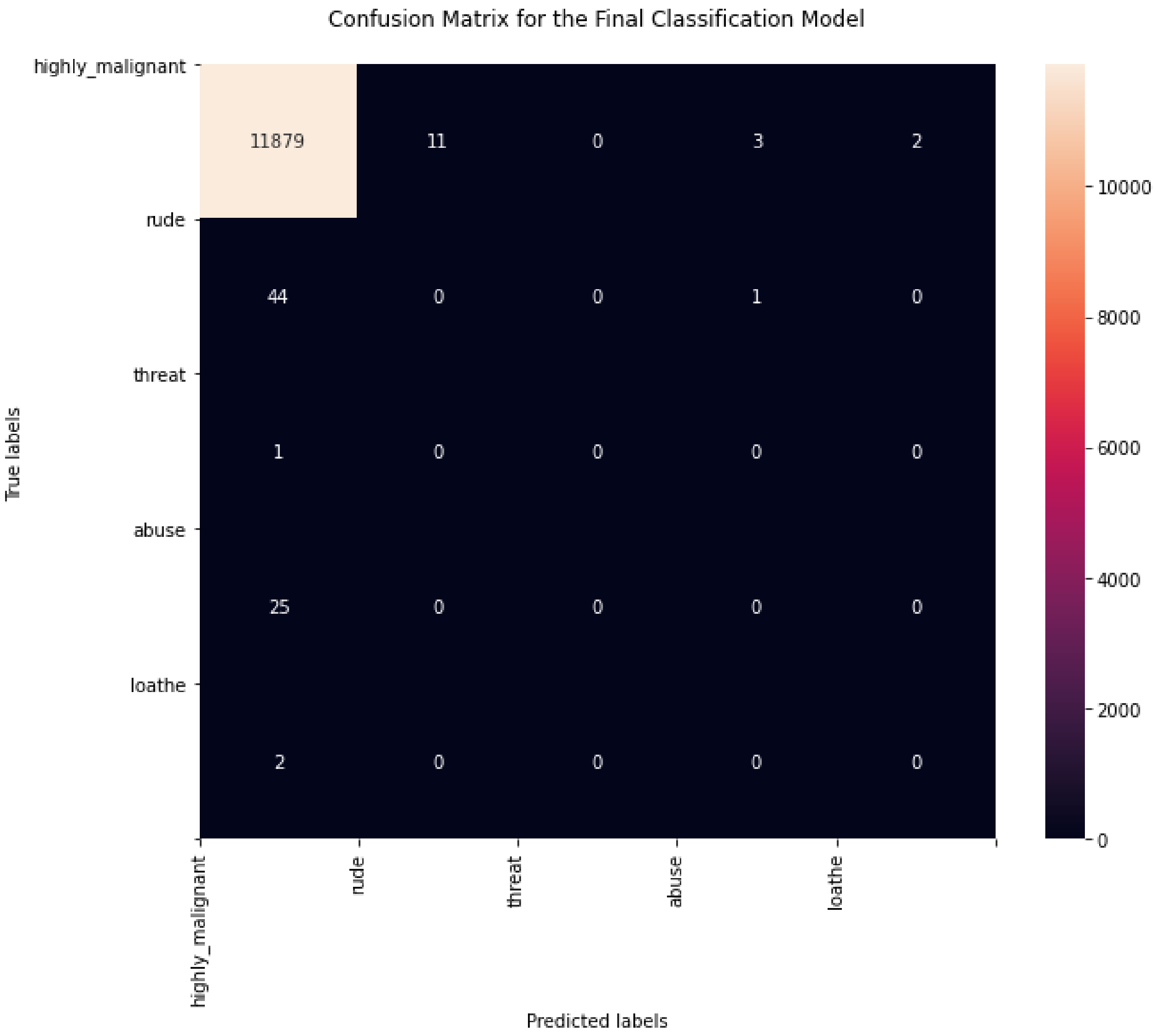
# CLASSIFIER

## AUC ROC Curve for Final Model



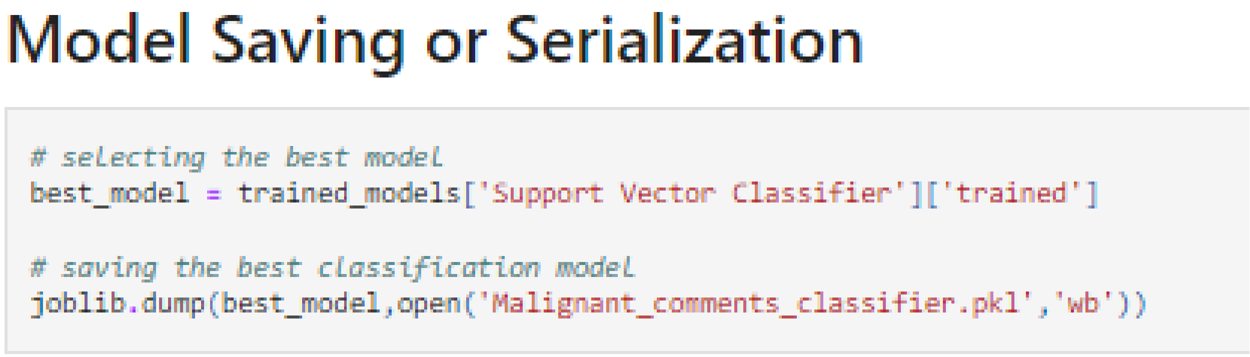
# CLASSIFIER

## Confusion Matrix for Final Model



# CLASSIFIER

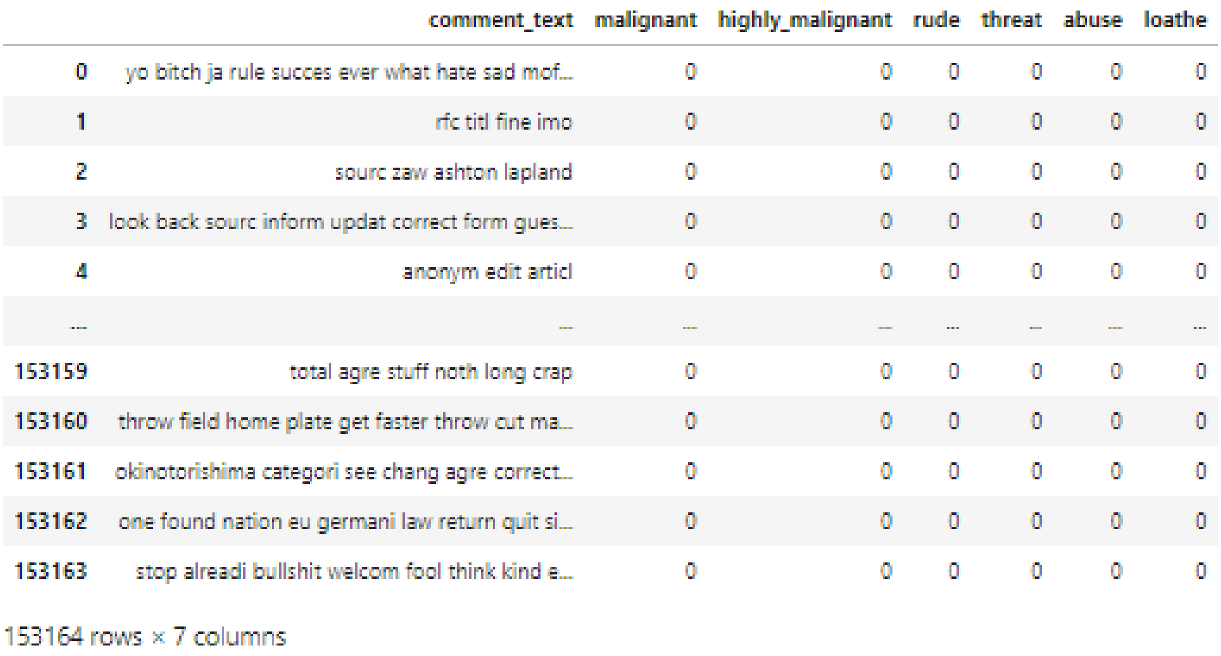
## Saving the best model



### Model Saving or Serialization

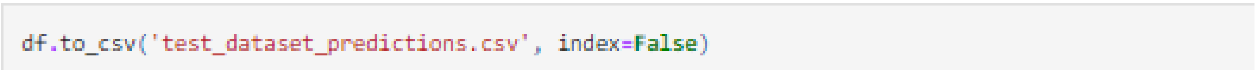
```
# selecting the best model
best_model = trained_models['Support Vector Classifier']['trained']

# 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



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

# THANK YOU!!!

## Acknowledgement

I would like to Acknowledge that I have used various external sources to complete the project and improve myself.