FLIP ROBO

MALAIGNANT COMMENTS CLASSIFICATION

Submitted by

Girija Chandra Mohan

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject MatterExpert) Khushboo Garg as well as Flip Robo Technologies who gave me the opportunity to do this project on Malignant Comments Classification, which also helped me in doing lots of research wherein I came to know about so many new things especially the Natural Language Processing and Natural Language Toolkit parts.

Also, I have utilized a few external resources that helped me to completethis project. I ensured that I learn from the samples and modify thingsaccording to my project requirement. All the external resources that wereused in creating this project are listed below:

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

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

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

4) https://github.com/

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

6) https://medium.com/

7) https://towardsdatascience.com/

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

INTRODUCTION

Business Problem Framing

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

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

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

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

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

Conceptual Background of the Domain Problem

Online platforms and social media 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 individualsare being bullied by these mob these individuals are going silent without speaking anything. So, ideally the motive of this disgraceful mob is achieved.

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

Review of Literature

The purpose of the literature review is to:

1. 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 5 different Classification algorithms and shortlisted the best on basis on the metrics of performance and I have chosen one algorithm and build a model in that algorithm.

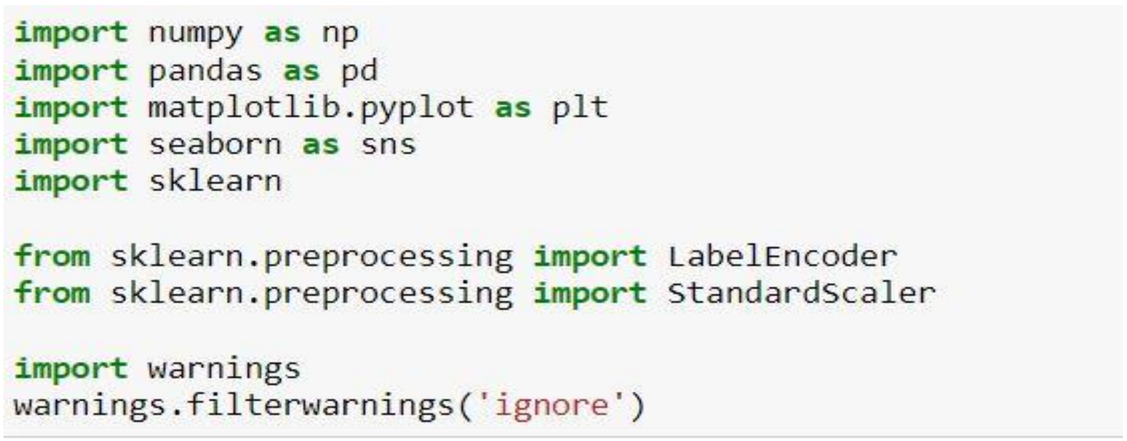
Motivation for the Problem Undertaken

One of the first lessons we learn as children is that the louder you scream and the bigger of a tantrum you throw, you 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 forms.

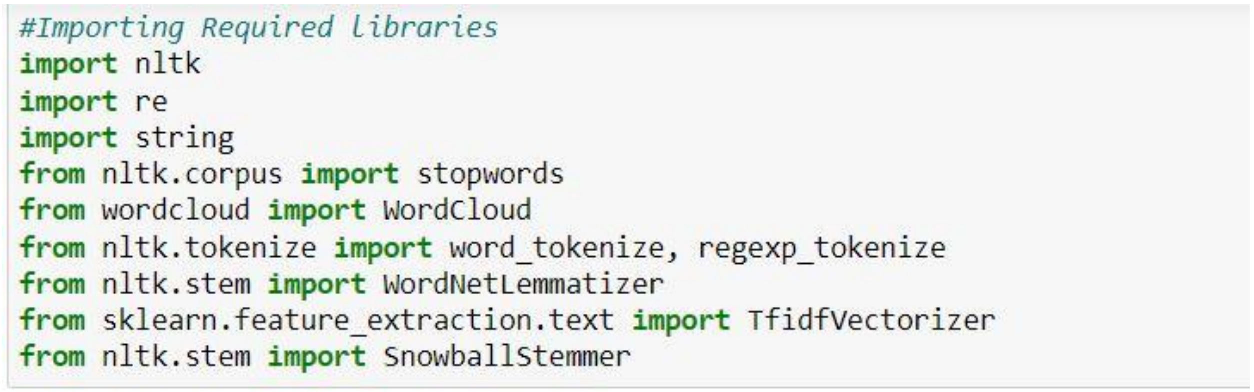
Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

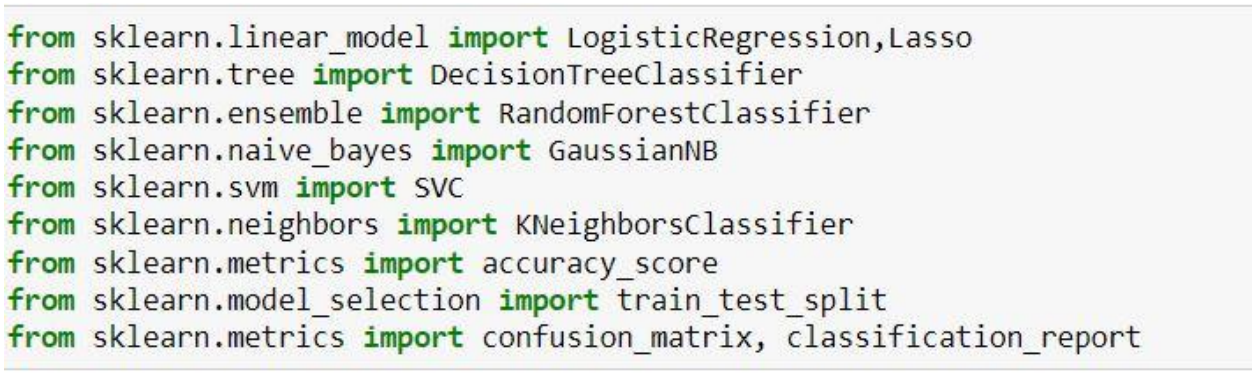
  The libraries/dependencies imported for this project are shown below:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')
```



```
#Importing Required libraries
import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize, regexp_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem import SnowballStemmer
```



```
from sklearn.linear_model import LogisticRegression,Lasso
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
```

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.

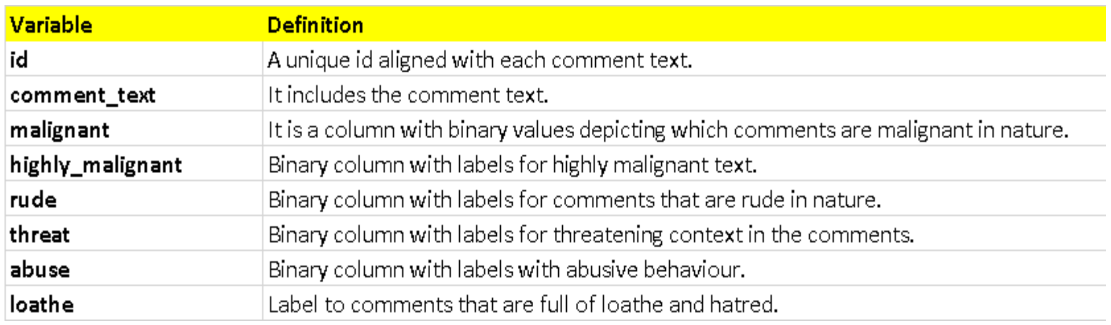
Threat: It contains indication of the comments that are giving any threat to someone.

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

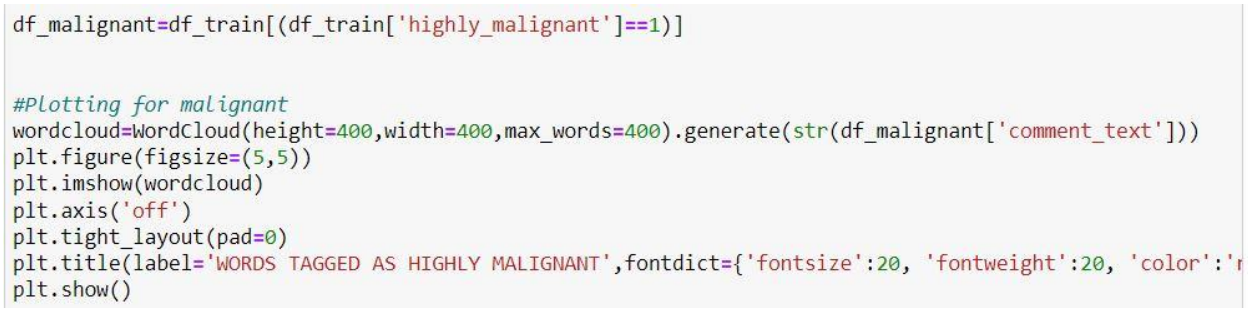
8. Convert text to vectors using TfidfVectorizer

9. Load saved or serialized model

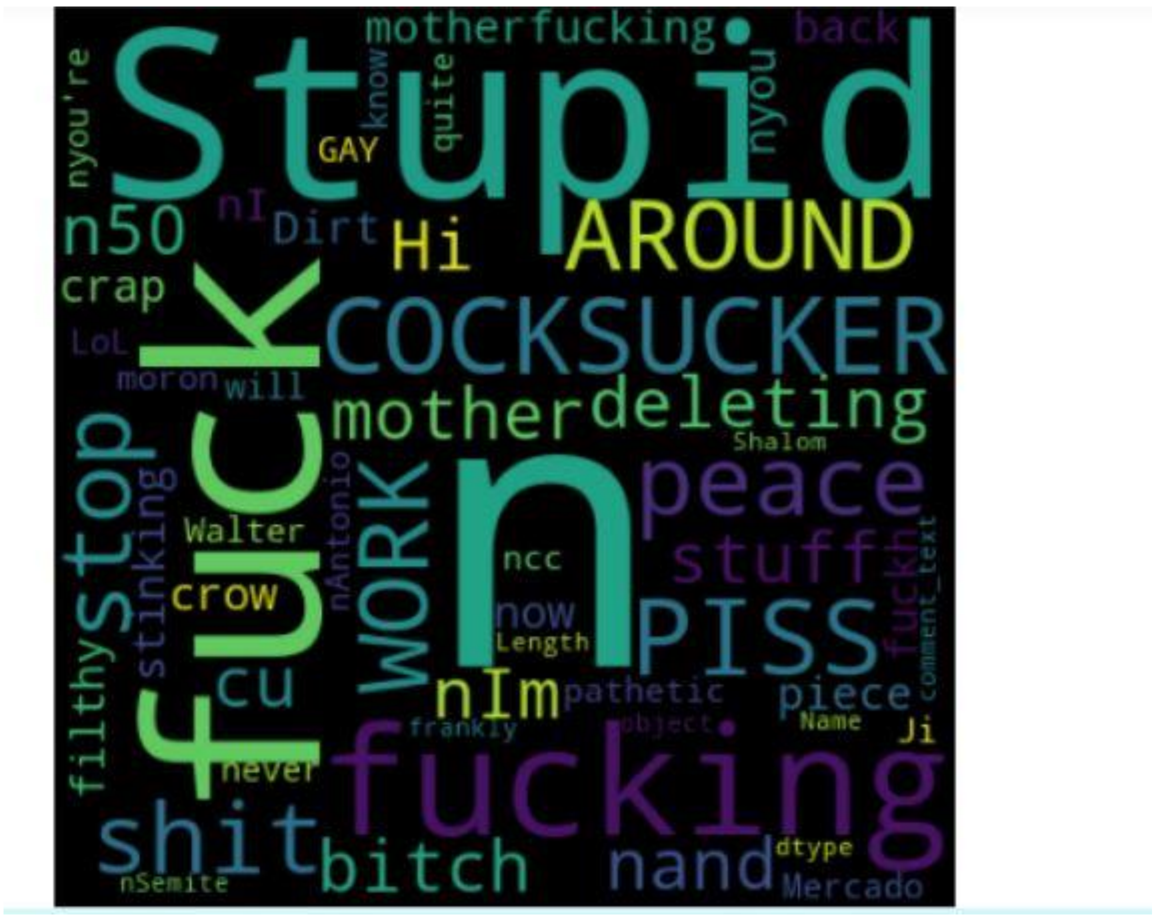
10. Predict values for multi class label.

- 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.



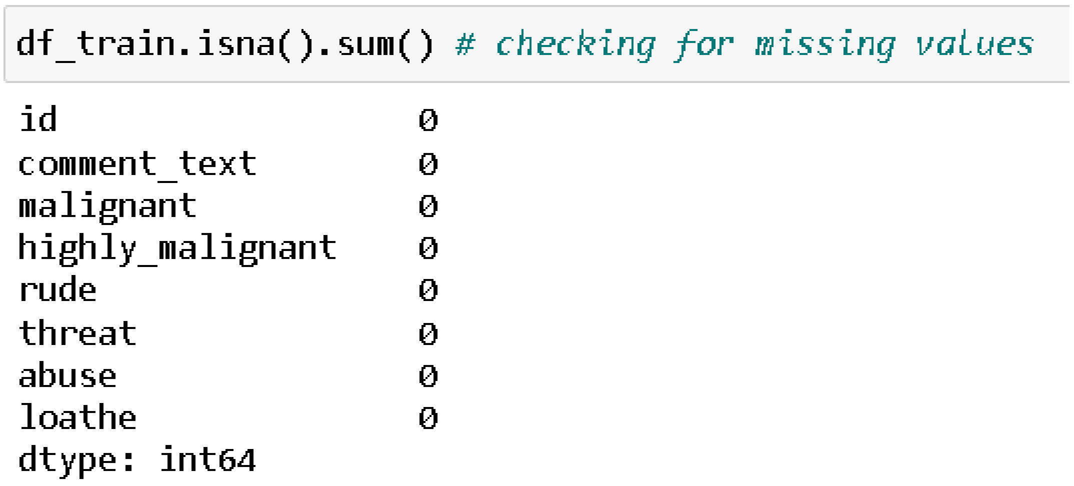
```
df_malignant=df_train[(df_train['highly_malignant']==1)]


#Plotting for malignant
wordcloud=WordCloud(height=400,width=400,max_words=400).generate(str(df_malignant['comment_text']))
plt.figure(figsize=(5,5))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title(label='WORDS TAGGED AS HIGHLY MALIGNANT',fontdict={'fontsize':20, 'fontweight':20, 'color':'r
plt.show()
```



Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

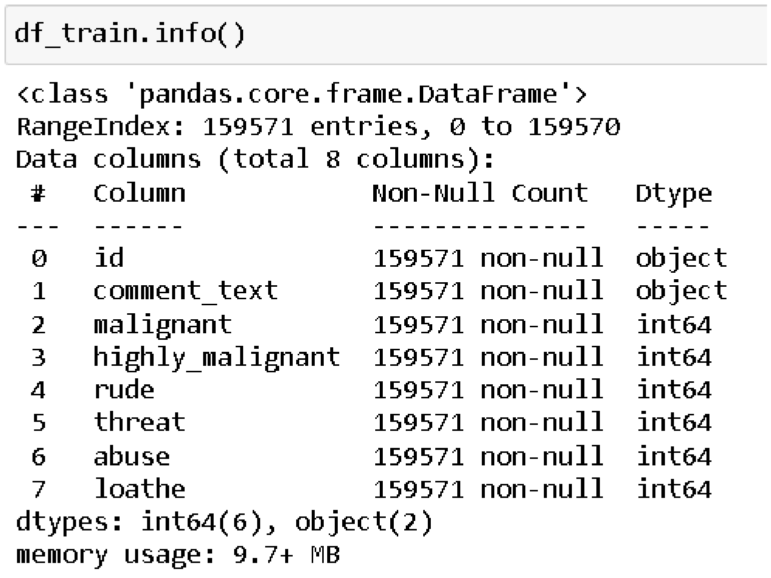
  I checked through the entire training dataset for any kind of missing values information and all these pre processing steps were repeated on the testing dataset as well.



```
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
```

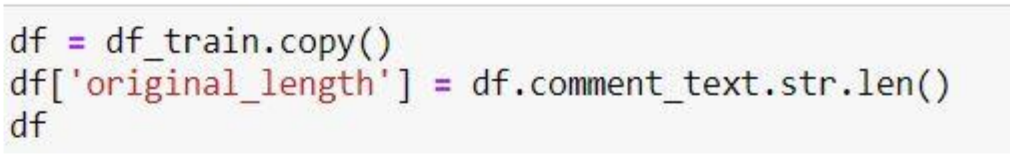
Then we went ahead and took a look at the dataset information. Using the info method, we are able to confirm the non-null count details as well as the datatype information. We have a total of 8 columns out of which 2 columns have object datatype while the remaining 6 columns are of integer datatype.



```
df_train.info()
```

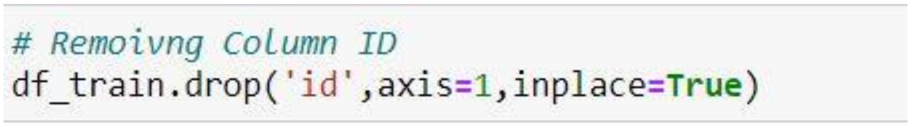
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 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.



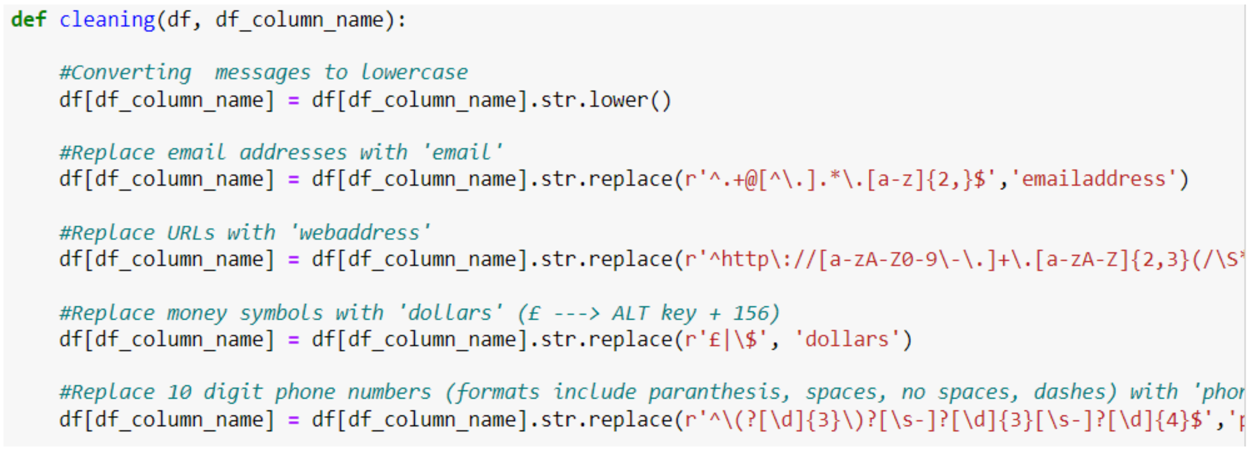
```
df = df_train.copy()
df['original_length'] = df.comment_text.str.len()
df
```

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.

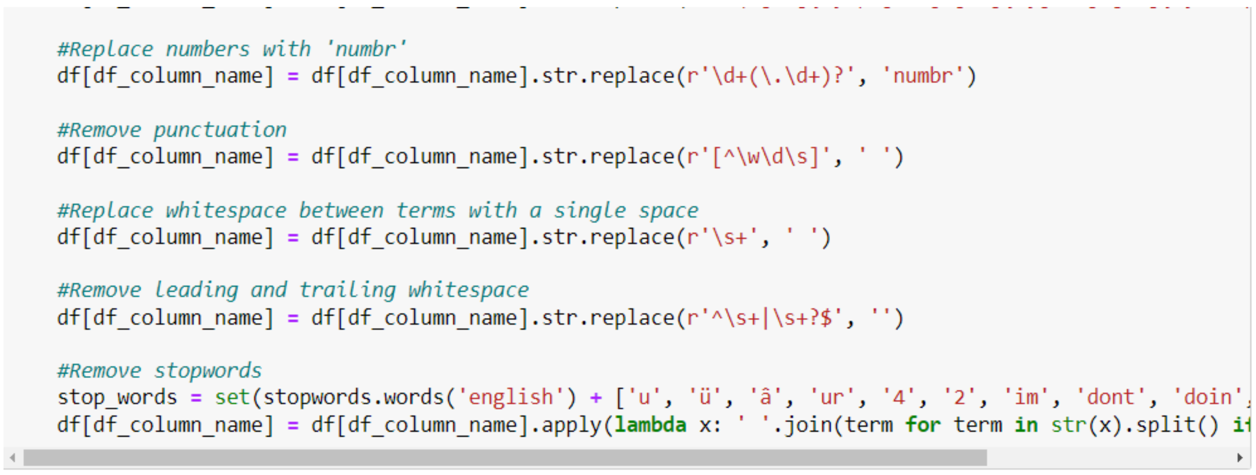


```
# Remoivng Column ID
df_train.drop('id',axis=1,inplace=True)
```

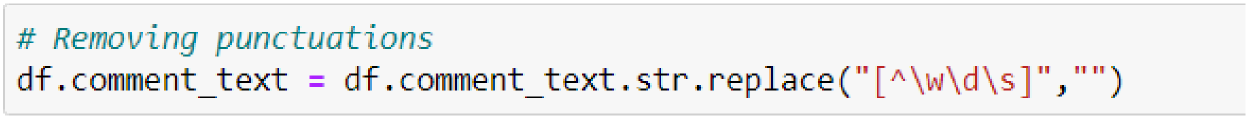
Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP).



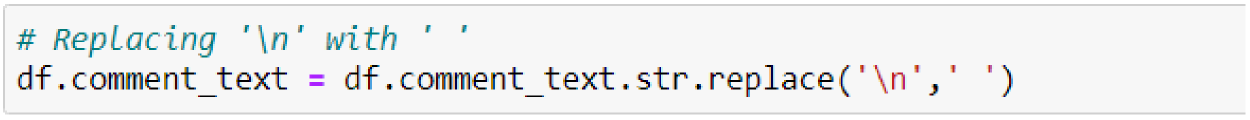
```
def cleaning(df, df_column_name):

    #Converting  messages to lowercase
    df[df_column_name] = df[df_column_name].str.lower()

    #Replace email addresses with 'email'
    df[df_column_name] = df[df_column_name].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

    #Replace URLs with 'webaddress'
    df[df_column_name] = df[df_column_name].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S

    #Replace money symbols with 'dollars' (£ ---> ALT key + 156)
    df[df_column_name] = df[df_column_name].str.replace(r'£|\$', 'dollars')

    #Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phor
    df[df_column_name] = df[df_column_name].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','
```



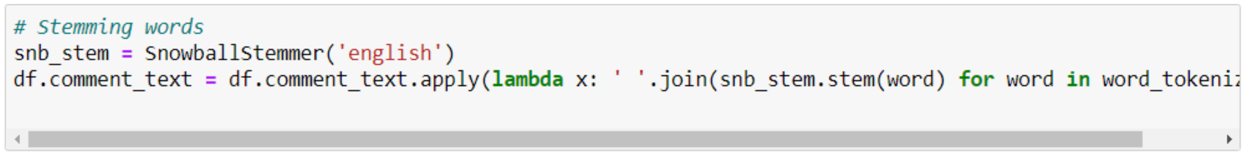
```
    #Replace numbers with 'numbr'
    df[df_column_name] = df[df_column_name].str.replace(r'\d+(\.\d+)?', 'numbr')

    #Remove punctuation
    df[df_column_name] = df[df_column_name].str.replace(r'[^\w\d\s]', ' ')

    #Replace whitespace between terms with a single space
    df[df_column_name] = df[df_column_name].str.replace(r'\s+', ' ')

    #Remove leading and trailing whitespace
    df[df_column_name] = df[df_column_name].str.replace(r'^\s+|\s+?$', '')

    #Remove stopwords
    stop_words = set(stopwords.words('english') + ['u', 'ü', 'â', 'ur', '4', '2', 'im', 'dont', 'doin',
    df[df_column_name] = df[df_column_name].apply(lambda x: ' '.join(term for term in str(x).split() if
```



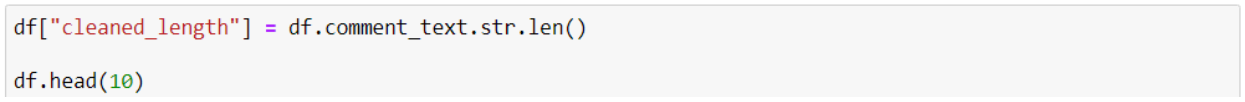
```
# Removing punctuations
df.comment_text = df.comment_text.str.replace("[^\w\d\s]","")
```



```
# Replacing '\n' with ' '
df.comment_text = df.comment_text.str.replace('\n',' ')
```



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



```
df["cleaned_length"] = df.comment_text.str.len()

df.head(10)
```

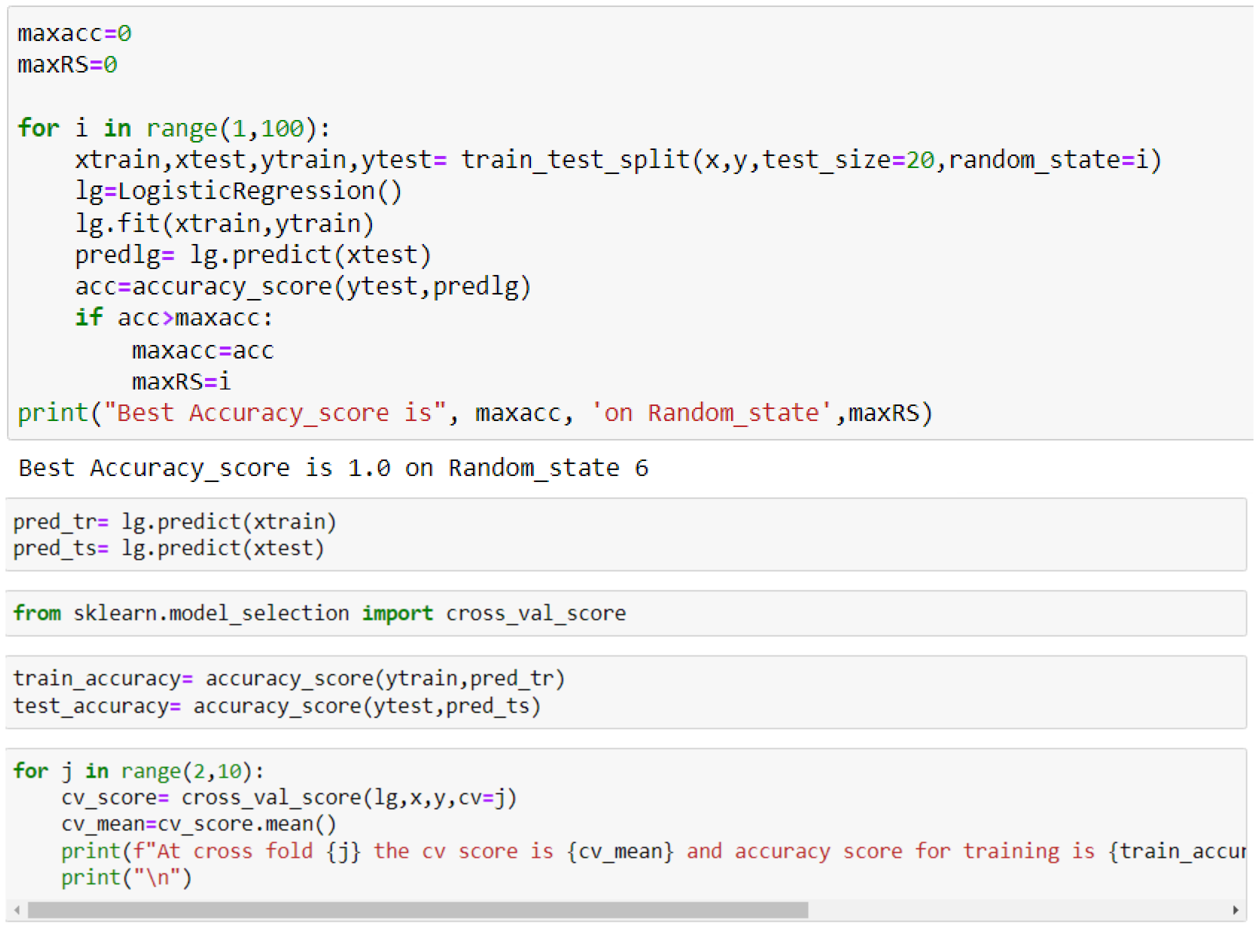
- Testing of Identified Approaches (Algorithms)

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

1) Logistic Regression
2) Random Forest Classifier
3) SVC
4) K Nearest Neighbors Classifier
5) Decision Tree 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.



```
maxacc=0
maxRS=0

for i in range(1,100):
    xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=20,random_state=i)
    lg=LogisticRegression()
    lg.fit(xtrain,ytrain)
    predlg= lg.predict(xtest)
    acc=accuracy_score(ytest,predlg)
    if acc>maxacc:
        maxacc=acc
        maxRS=i
print("Best Accuracy_score is", maxacc, 'on Random_state',maxRS)
```

```
Best Accuracy_score is 1.0 on Random_state 6
```

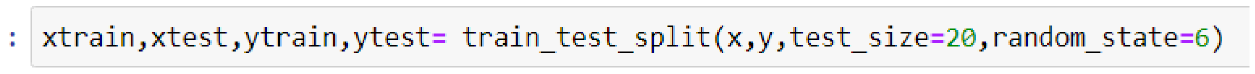
```
pred_tr= lg.predict(xtrain)
pred_ts= lg.predict(xtest)
```

```
from sklearn.model_selection import cross_val_score
```

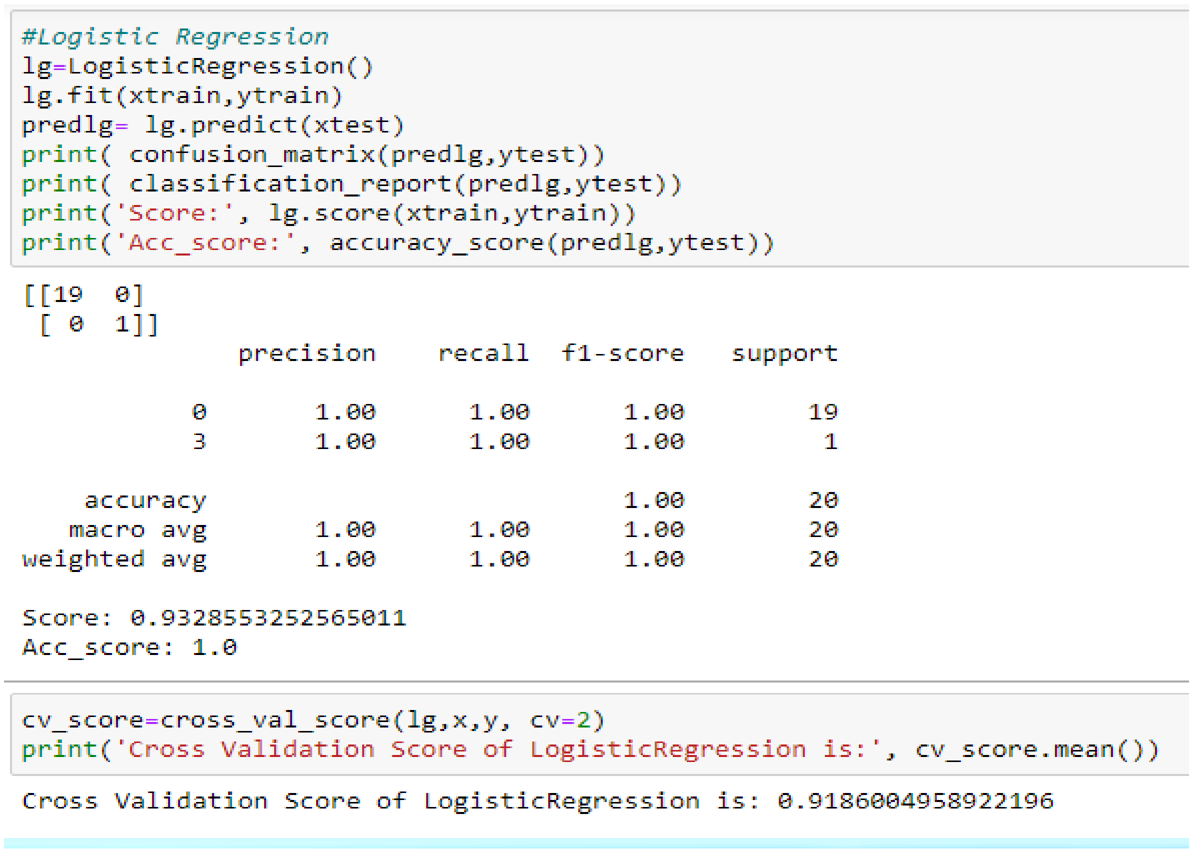
```
train_accuracy= accuracy_score(ytrain,pred_tr)
test_accuracy= accuracy_score(ytest,pred_ts)
```

```
for j in range(2,10):
    cv_score= cross_val_score(lg,x,y,cv=j)
    cv_mean=cv_score.mean()
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score for training is {train_accu
    print("\n")
```

Check The Accuracy and Error



```
: xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=20,random_state=6)
```

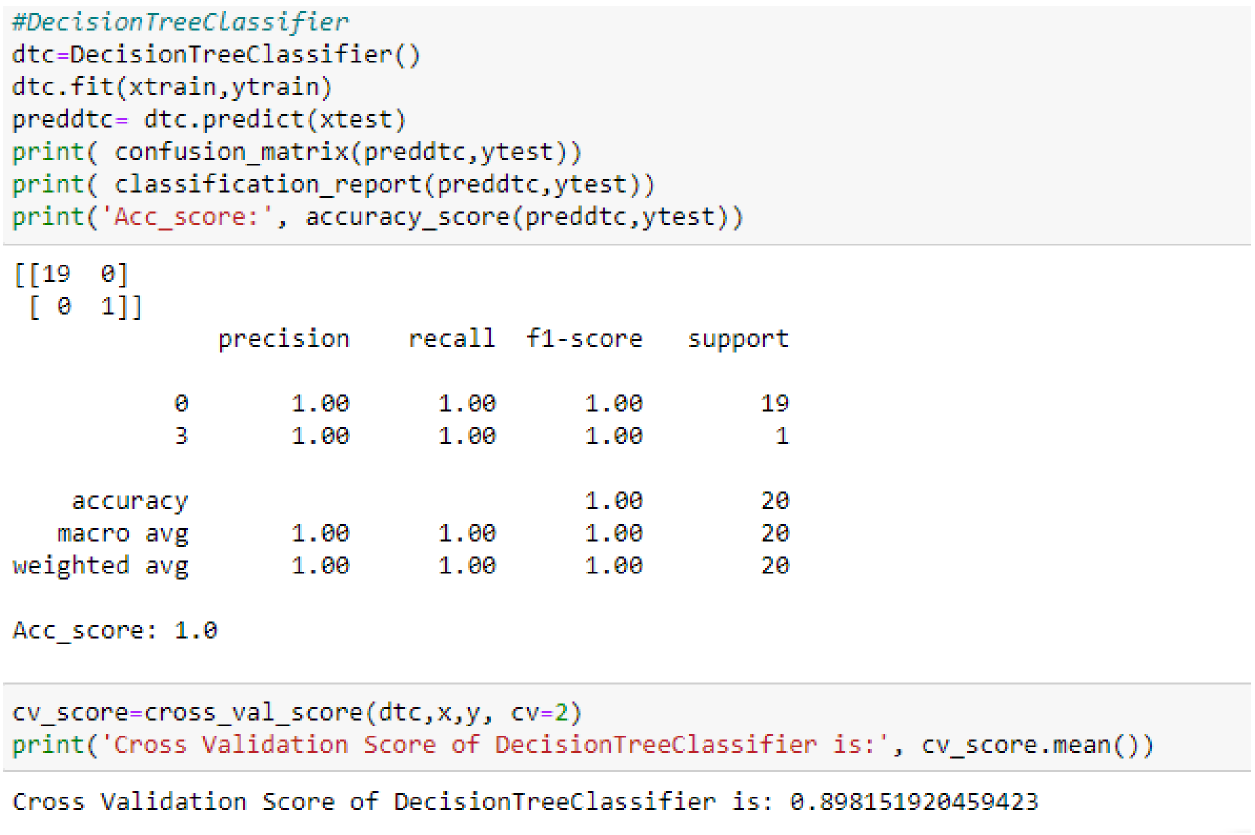


```
#Logistic Regression
lg=LogisticRegression()
lg.fit(xtrain,ytrain)
predlg= lg.predict(xtest)
print( confusion_matrix(predlg,ytest))
print( classification_report(predlg,ytest))
print('Score:', lg.score(xtrain,ytrain))
print('Acc_score:', accuracy_score(predlg,ytest))
```

```
[[19  0]
 [ 0  1]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           3       1.00      1.00      1.00         1

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Score: 0.9328553252565011
Acc_score: 1.0
```

```
cv_score=cross_val_score(lg,x,y, cv=2)
print('Cross Validation Score of LogisticRegression is:', cv_score.mean())
```

```
Cross Validation Score of LogisticRegression is: 0.9186004958922196
```



```
#DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(xtrain,ytrain)
preddtc= dtc.predict(xtest)
print( confusion_matrix(preddtc,ytest))
print( classification_report(preddtc,ytest))
print('Acc_score:', accuracy_score(preddtc,ytest))
```

```
[[19  0]
 [ 0  1]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           3       1.00      1.00      1.00         1

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Acc_score: 1.0
```

```
cv_score=cross_val_score(dtc,x,y, cv=2)
print('Cross Validation Score of DecisionTreeClassifier is:', cv_score.mean())
```

```
Cross Validation Score of DecisionTreeClassifier is: 0.898151920459423
```

```
#RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(xtrain,ytrain)
predrf= rf.predict(xtest)
print( confusion_matrix(predrf,ytest))
print( classification_report(predrf,ytest))
print('Acc_score:', accuracy_score(predrf,ytest))
```

```
[[19  0]
 [ 0  1]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           3       1.00      1.00      1.00         1

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Acc_score: 1.0
```

```
cv_score=cross_val_score(rf,x,y, cv=2)
print('Cross Validation Score of RandomForestClassifier is:', cv_score.mean())
```

```
Cross Validation Score of RandomForestClassifier is: 0.9174223364036995
```

```
#knn
knn=KNeighborsClassifier()
knn.fit(xtrain,ytrain)
predknn= knn.predict(xtest)
print( confusion_matrix(predknn,ytest))
print( classification_report(predknn,ytest))
print('Acc_score:', accuracy_score(predknn,ytest))
```

```
[[19  1]
 [ 0  0]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        20
           3       0.00      0.00      0.00         0

    accuracy                           0.95        20
   macro avg       0.50      0.47      0.49        20
weighted avg       1.00      0.95      0.97        20

Acc_score: 0.95
```

```
#SVC
svc=SVC()
svc.fit(xtrain,ytrain)
predsvc= svc.predict(xtest)
print( confusion_matrix(predsvc,ytest))
print( classification_report(predsvc,ytest))
print('Acc_score:', accuracy_score(predsvc,ytest))
```

```
[[19  0]
 [ 0  1]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           3       1.00      1.00      1.00         1

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20

Acc_score: 1.0
```

```
cv_score=cross_val_score(svc,x,y, cv=2)
print('Cross Validation Score of SVC is:', cv_score.mean())
```

```
Cross Validation Score of SVC is: 0.9186443682642885
```

- Key Metrics for success in solving problem under consideration

  Hyperparameter Tuning:

```
parameters={'penalty': ['l1', 'l2', 'elasticnet', 'none'],
            'C': [0,1,2],
            'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
            'multi_class':['auto', 'ovr', 'multinomial'],
            'max_iter': [50,100,150,200]}
```

```
GCV=GridSearchCV(LogisticRegression(), parameters, cv=2, scoring='accuracy')
GCV.fit(xtrain,ytrain)
GCV.best_params_
```

```
{'C': 2}
```

```
GCV.best_estimator_
```

```
LogisticRegression(C=2)
```

```
GCV_pred=GCV.best_estimator_.predict(xtest)
accuracy_score(ytest,GCV_pred)
```

```
0.95
```

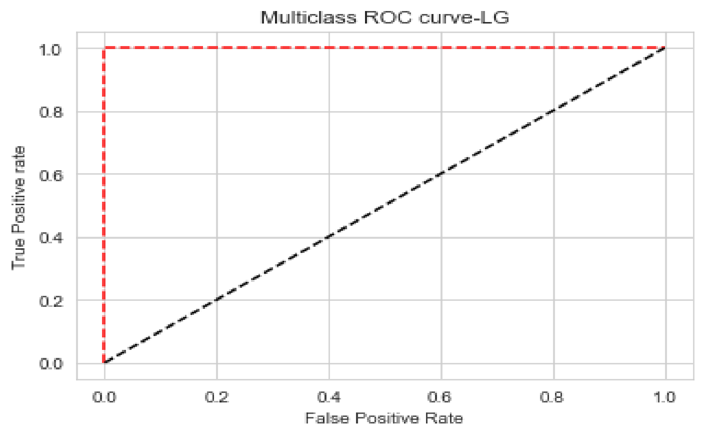
Final Classification Model details:

```
#Logistic Regression
fnlg=LogisticRegression(penalty='l2', C=2, max_iter=100)
fnlg.fit(xtrain,ytrain)
predfnlg= fnlg.predict(xtest)
print( confusion_matrix(predfnlg,ytest))
print( classification_report(predfnlg,ytest))
print('Score:', fnlg.score(xtrain,ytrain))
print('Acc_score:', accuracy_score(predfnlg,ytest))
```

```
[[19  0  0]
 [ 0  0  1]
 [ 0  0  0]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           2       0.00      0.00      0.00         1
           3       0.00      0.00      0.00         0

    accuracy                           0.95        20
   macro avg       0.33      0.33      0.33        20
weighted avg       0.95      0.95      0.95        20

Score: 0.9379007339346039
Acc_score: 0.95
```

AUC ROC Curve for Final Model:

```
plt.plot(fpr[0], tpr[0], linestyle='--',color='red', label='Class 0 vs Rest')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green', label='Class 1 vs Rest')

plt.title('Multiclass ROC curve-LG')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.plot([0,1],[0,1],'k--')
plt.show()
```



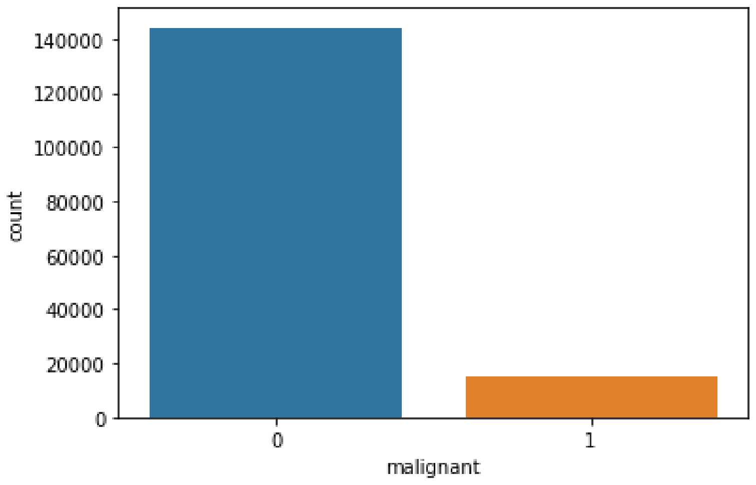
Saving the best model:

```
import pickle
filename= 'MCC.pkl'
pickle.dump(fnlg,open(filename, 'wb'))
```

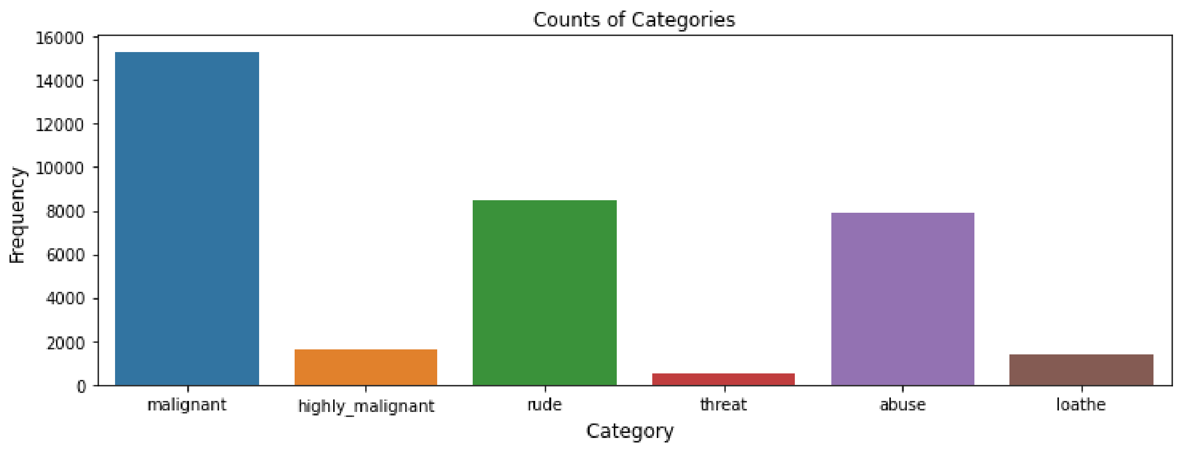
- Visualizations

  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.

```
columns=['malignant', 'highly_malignant', 'rude', 'threat',
         'abuse', 'loathe']
for i in columns:
    sns.countplot(df_train[i])
    plt.show()
```

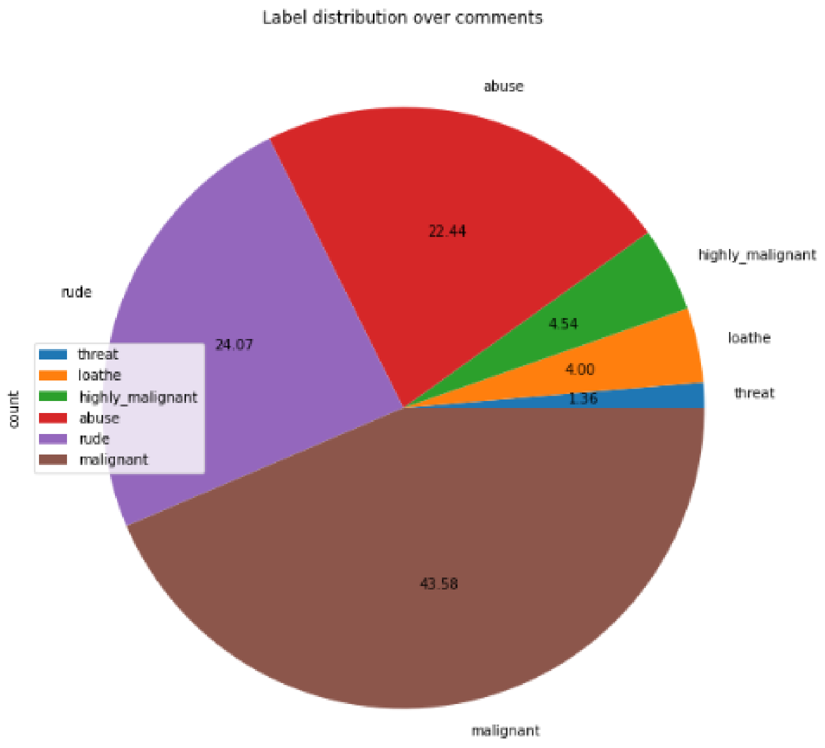


```
plt.figure(figsize=(12,4))
ax = sns.barplot(counts.index, counts.values)
plt.title("Counts of Categories")
plt.ylabel('Frequency', fontsize=12)
plt.xlabel('Category ', fontsize=12)

plt.show()
```



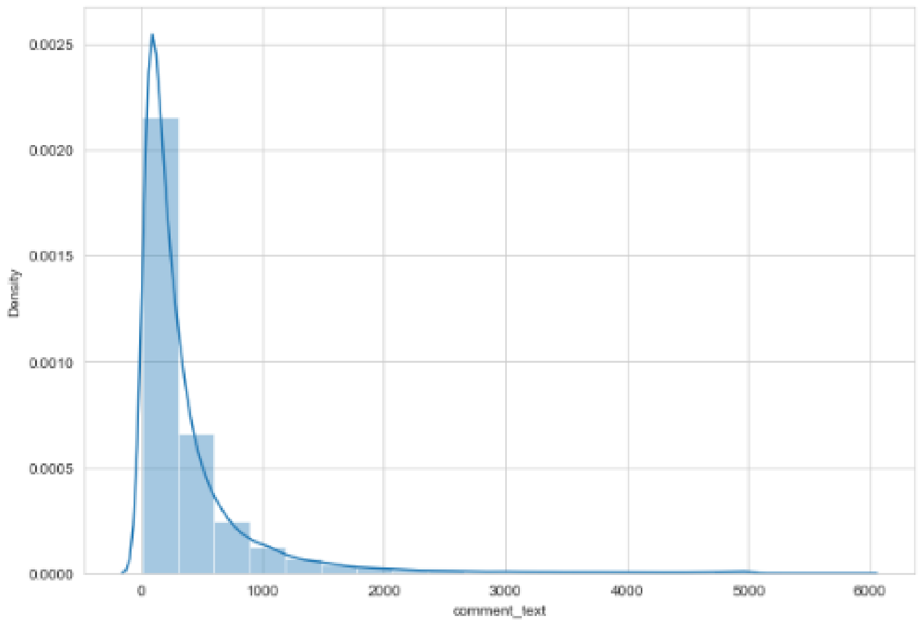
```
df_dist.plot.pie(y = 'count', title = 'Label distribution over comments', autopct='%.2f', figsize = (20, 10))\
                            .legend(loc='center left')
```

```
<matplotlib.legend.Legend at 0x196a6550>
```



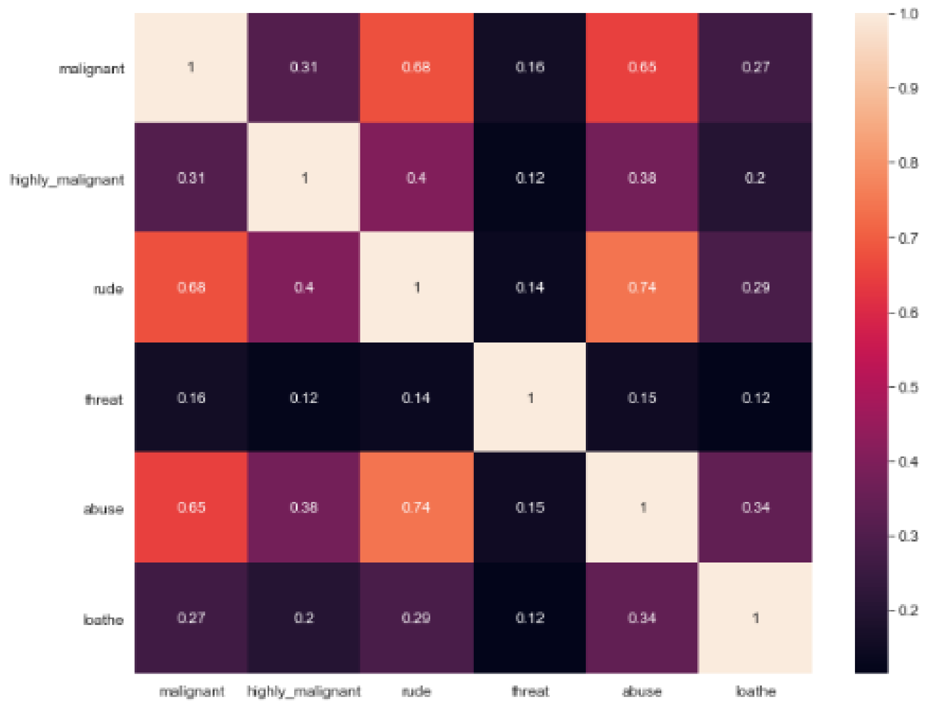
```
#Distribution of comments length
sns.set_style('whitegrid')
plt.figure(figsize=(10,7))
comment_len = df_train.comment_text.str.len()
sns.distplot(comment_len, bins=20)
```

```
<AxesSubplot:xlabel='comment_text', ylabel='Density'>
```



we can see that most of the senterence are small.

```
plt.figure(figsize=(10,8))
sns.heatmap(df_train.corr(), annot = True)
plt.show()
```



	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1	0.31	0.68	0.16	0.65	0.27
highly_malignant	0.31	1	0.4	0.12	0.38	0.2
rude	0.68	0.4	1	0.14	0.74	0.29
threat	0.16	0.12	0.14	1	0.15	0.12
abuse	0.65	0.38	0.74	0.15	1	0.34
loathe	0.27	0.2	0.29	0.12	0.34	1

We can see more corelations in the variables, Abuse have more corelation with malignant and rude.

Rude has more postive corelation with malignant

- 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 wordcloud 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 foul 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

The use of social media is the most common trend among the activities of today's people. Social networking sites offer today's teenagers a platform for communication and entertainment. They use social media to collect more information from their friends and followers. The vastness of social media sites ensures that not all of them provide a decent environment for children. In such cases, the impact of the negative influences of social media on teenage users increases with an increase in the use of **offensive language** in social conversations. This increase could lead to **frustration**, **depression** and a large change in their behaviour. Hence, I propose a novel approach to classify bad language usage in text conversations. I have considered the English medium for textual conversation. I have developed our system based on a foul language classification approach; it is based on an improved version of a Logistic Regression Algorithm that detects offensive language usage in a conversation. As per our evaluation, we found that lesser number of users conversation is not decent all the time. We trained 159571 observations for eight context categories using a Logistic Regression algorithm for context detection. Then, the system classifies the use of foul language in one of the trained contexts in the text conversation. In our test data, we observed 10% of participants used foul language during their text conversation. Hence, our proposed approach can identify the impact of foul language in text conversations using a classification technique and emotion detection to identify the foul language usage

Limitations of this work and Scope for Future Work

The limitation of the study is that we have a imbalanced data so our model learnt more about the non-abusive sentence more than the abusive sentence. Which makes our model act like a overfit model when tested with live data. And also, model tend to not identify a foul or a sarcastically foul language.