FLIP ROBO

MALIGNANT COMMENTS CLASSIFICATION MODEL

Submitted by:

RIFATH NAZUM SHAIK

ACKNOWLEDGEMENT

The internship opportunity I have with Flip Robo Technologies is a great chance for learning and professional development. I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills acknowledge in the best possible way.

I would like to extend my appreciation and thanks for the mentors from DataTrained and professionals from FlipRoboTechnologies who had extended their help and support.

References:

https://sklearn.org/supervised_learning.html#supervised-learning

https://www.datacamp.com/community

https://github.com/mxc19912008/Andrew-Ng-Machine-Learning-Notes

https://www.analyticsvidhya.com/blog/category/machine-learning/

INTRODUCTION

Business Problem:

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.

Background of domain:

With the proliferation of smart devices and mobile and social network environments, the social side effects of these technologies, including cyberbullying through malicious comments and rumors, have become more serious. Malicious online comments have emerged as an unwelcome social issue worldwide.

Both cyberbullying and malicious comments are increasingly viewed as a social problem due to their role in suicides and other real-world crimes. However, the online environment generally lacks a system of barriers to prevent privacy invasion, personal attacks, and cyberbullying, and the barriers that do exist are weak. Social violence as an online phenomenon is increasingly pervasive, a phenomenon manifesting itself through social divisiveness.

Motivations for malicious comments identified in the study involved targeting people's mistakes. Conversely, most benevolent comments involved encouragement and compliments to help people in difficult or risky situations, showing malicious comments is a primary reason for degradation of online social networks. Moreover, the abolition of anonymity and intensification of punishment in social media can be effective in reducing malicious comments and rumors. However, potential violation of freedom of expression also risks trivializing the online social network itself. Because anonymous forms of freedom of expression have always been controversial in theoretical and normative spheres of social research. Careful consideration of any limiting of comments is necessary before a ban might be contemplated.

Requiring true identities would cause them to be more careful and responsible. Our results also suggest providers of social media services can apply text filters to their systems. Because certain texts are used repeatedly in cyberbullying or malicious comments, providers should be persuaded to develop a system to detect certain texts and alert them as to when to possibly take action against the people posting them. Such a filtering function could reduce the number of all kinds of malicious comments. Conversely, social media service providers should consider posting lists that rank users most active in posting benevolent comments on their sites. Because people generally enjoy self-expression, these rankings could motivate more people to post positive comments as a way to develop a new social norm in which malicious comments are unwelcome and the people posting them are scorned.

.

MOTIVATION FOR PROBLEM UNDER TAKEN:

Based on data provided , a comment is assessed based on different factors. By building the model, we can assess which comments are highly likely to be hateful in varying degrees of hate thereby it will be useful for those people who are target of online hate comments by deleting those comments.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL MODELLING OF PROBLEM:

Mathematical modeling is simply the method of implementing statistical analysis to a dataset where a Statistical Model is a mathematical representation of observed data.

While analyzing the data, there are an array of statistical models we can choose to utilize.

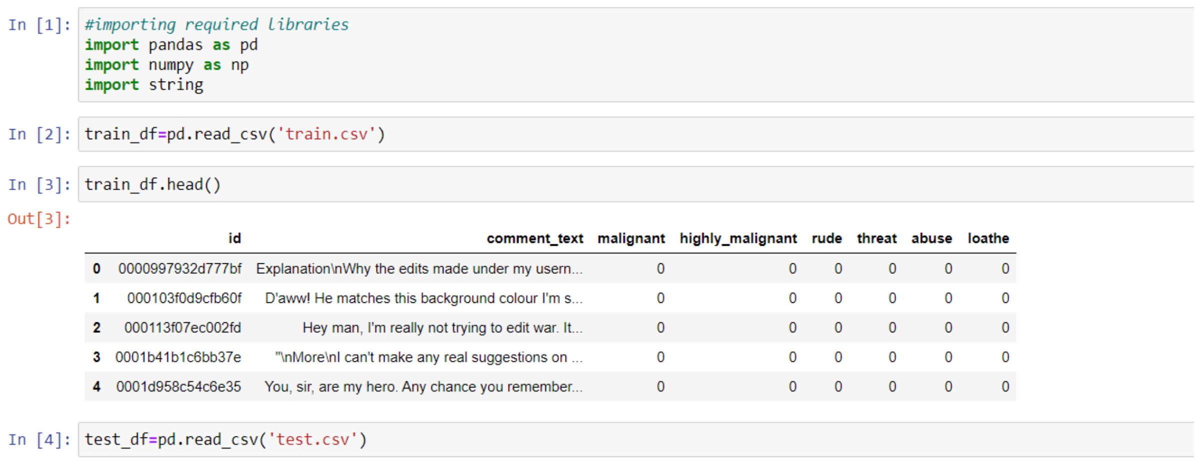
For the given project, we need to predict whether the given comment falls into the any category of hate.

This is a classification problem. There are wide varieties of classification models like decision trees, random forests, nearest neighbor, Logistic Regression.

DATA SOURCE AND FORMAT:

The data has been provided in different train and test datasets in a comma separated values(.csv) format.

1. The data will be loaded into pandas dataframe.

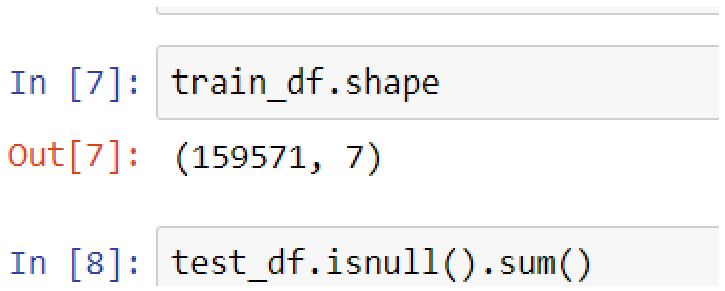


```
In [1]: #importing required libraries
        import pandas as pd
        import numpy as np
        import string

In [2]: train_df=pd.read_csv('train.csv')

In [3]: train_df.head()

Out[3]:
```

	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

```
In [4]: test_df=pd.read_csv('test.csv')
```

2. Checking no. of rows and columns of the data frame and the data type of columns.



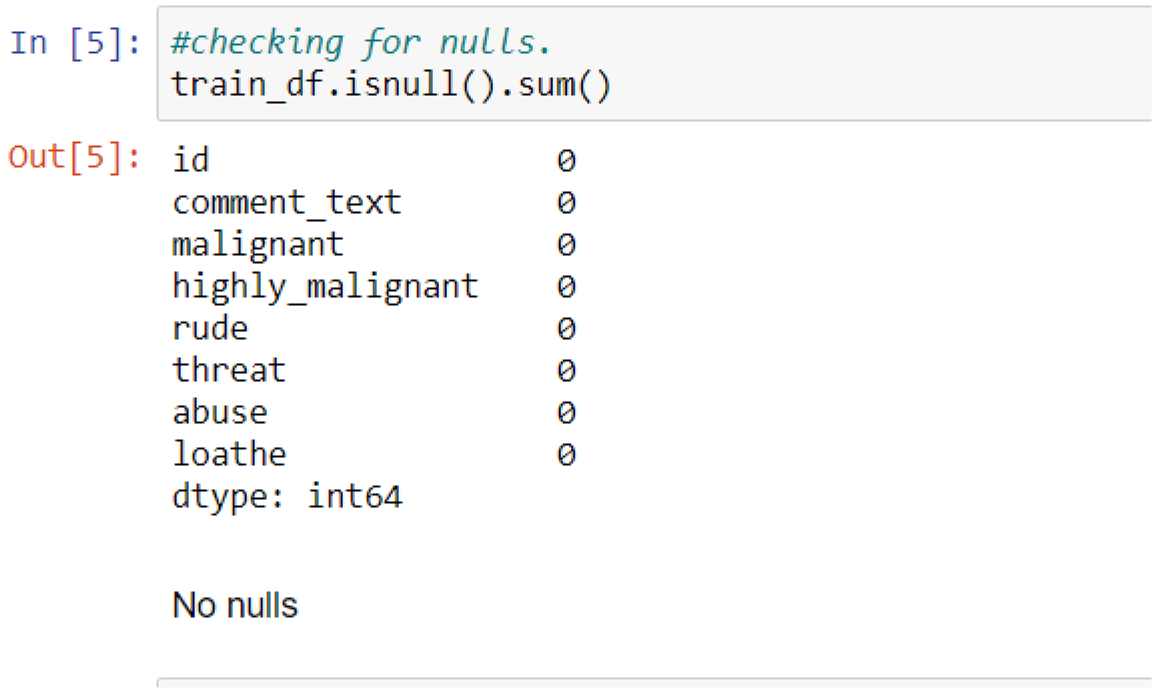
```
In [7]: train_df.shape

Out[7]: (159571, 7)

In [8]: test_df.isnull().sum()
```

This data set has around 1lakh,59thousand rows and 7 columns. Comment_text is object column where as malignant,highly malignant,rude ,threat,abuse and loathe are numeric columns.

DATA PRE PROCESSING:

Data preprocessing is a technique of converting raw data into useful format. Data cleaning is a part of preprocessing technique which involves filling missing values.

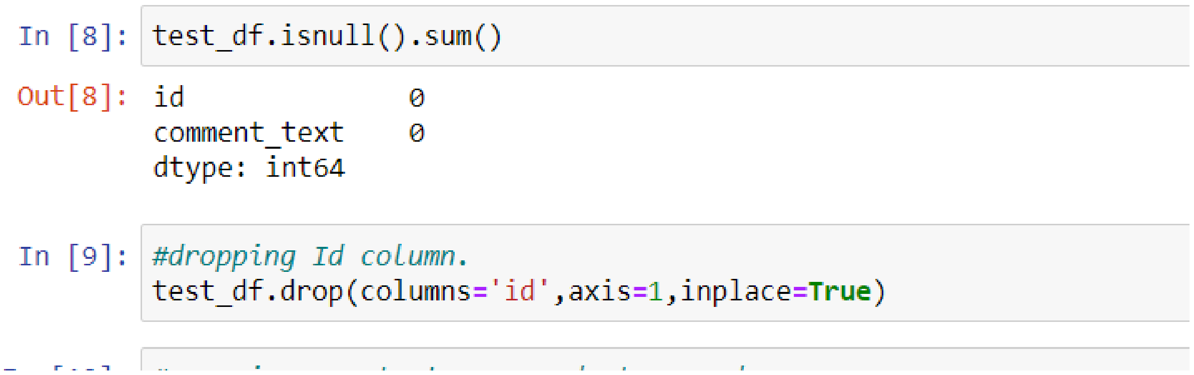
Firstly, I checked for presence of null values



```
In [5]: #checking for nulls.
        train_df.isnull().sum()

Out[5]: id                  0
        comment_text        0
        malignant           0
        highly_malignant    0
        rude                0
        threat              0
        abuse               0
        loathe              0
        dtype: int64
```

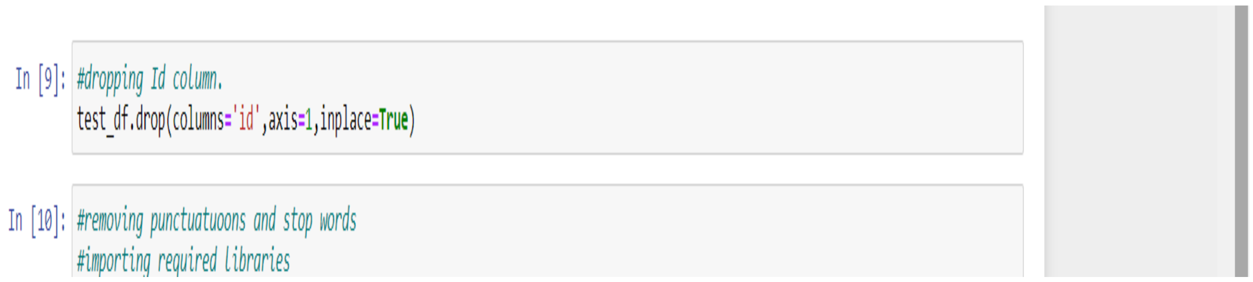
No nulls

There are two different data sets provided for test and train.So performing the same cleaning operation on both.



```
In [8]: test_df.isnull().sum()

Out[8]: id              0
        comment_text    0
        dtype: int64

In [9]: #dropping Id column.
        test_df.drop(columns='id',axis=1,inplace=True)
```

Dropping the id column as it has no use in predictions.Dropping it for both test and train sets.



```
In [9]: #dropping Id column.
        test_df.drop(columns='id',axis=1,inplace=True)

In [10]: #removing punctuatuoons and stop words
         #importing required libraries
```

Importing required libraries to clean the coment_text column.

```
[10]: #removing punctuatuoons and stop words
      #importing required libraries
      import nltk
      import string
      from nltk.corpus import stopwords
```

```
[11]: nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\rifath
[nltk_data]     nazum\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[11]: True
```

```
[12]: Stop_Words=stopwords.words('english')
      punct=string.punctuation
```

```
[13]: from nltk.tokenize import word_tokenize
```

```
[14]: nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to C:\Users\rifath
[nltk_data]     nazum\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
[14]: True
```

```
In [18]: train_df['comment_text'].head(10)
```

```
Out[18]: 0    Explanation\nWhy the edits made under my usern...
         1    D'aww! He matches this background colour I'm s...
         2    Hey man, I'm really not trying to edit war. It...
         3    "\nMore\nI can't make any real suggestions on ...
         4    You, sir, are my hero. Any chance you remember...
         5    "\n\nCongratulations from me as well, use the ...
         6         COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK
         7    Your vandalism to the Matt Shirvington article...
         8    Sorry if the word 'nonsense' was offensive to ...
         9    alignment on this subject and which are contra...
         Name: comment_text, dtype: object
```

```
In [19]: test_df['comment_text'].head(10)
```

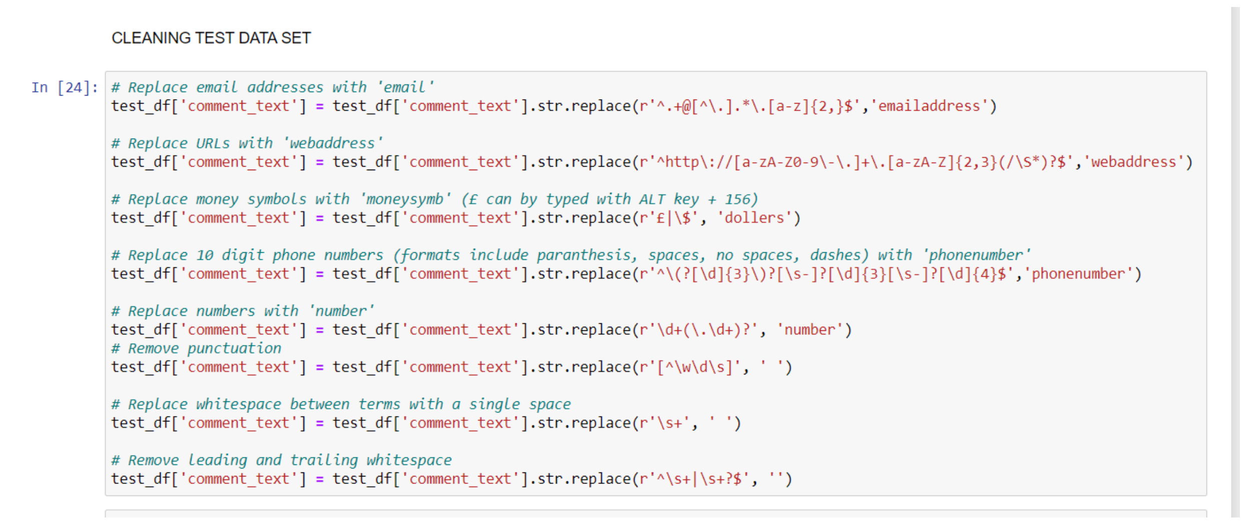
```
Out[19]: 0    Yo bitch Ja Rule is more succesful then you'll...
         1    == From RfC == \n\n The title is fine as it is...
         2    " \n\n == Sources == \n\n * Zawe Ashton on Lap...
         3    :If you have a look back at the source, the in...
         4            I don't anonymously edit articles at all.
         5    Thank you for understanding. I think very high...
         6    Please do not add nonsense to Wikipedia. Such ...
         7                         :Dear god this site is horrible.
         8    " \n Only a fool can believe in such numbers. ...
         9    == Double Redirects == \n\n When fixing double...
         Name: comment_text, dtype: object
```

```
In [20]: print ('Origian Length of training set before cleaning:',train_df['comment_text'].str.len().sum())

         Origian Length of training set before cleaning: 62893130
```

```
In [21]: print ('Origian Length of Test set before cleaning:',test_df['comment_text'].str.len().sum())

         Origian Length of Test set before cleaning: 55885733
```

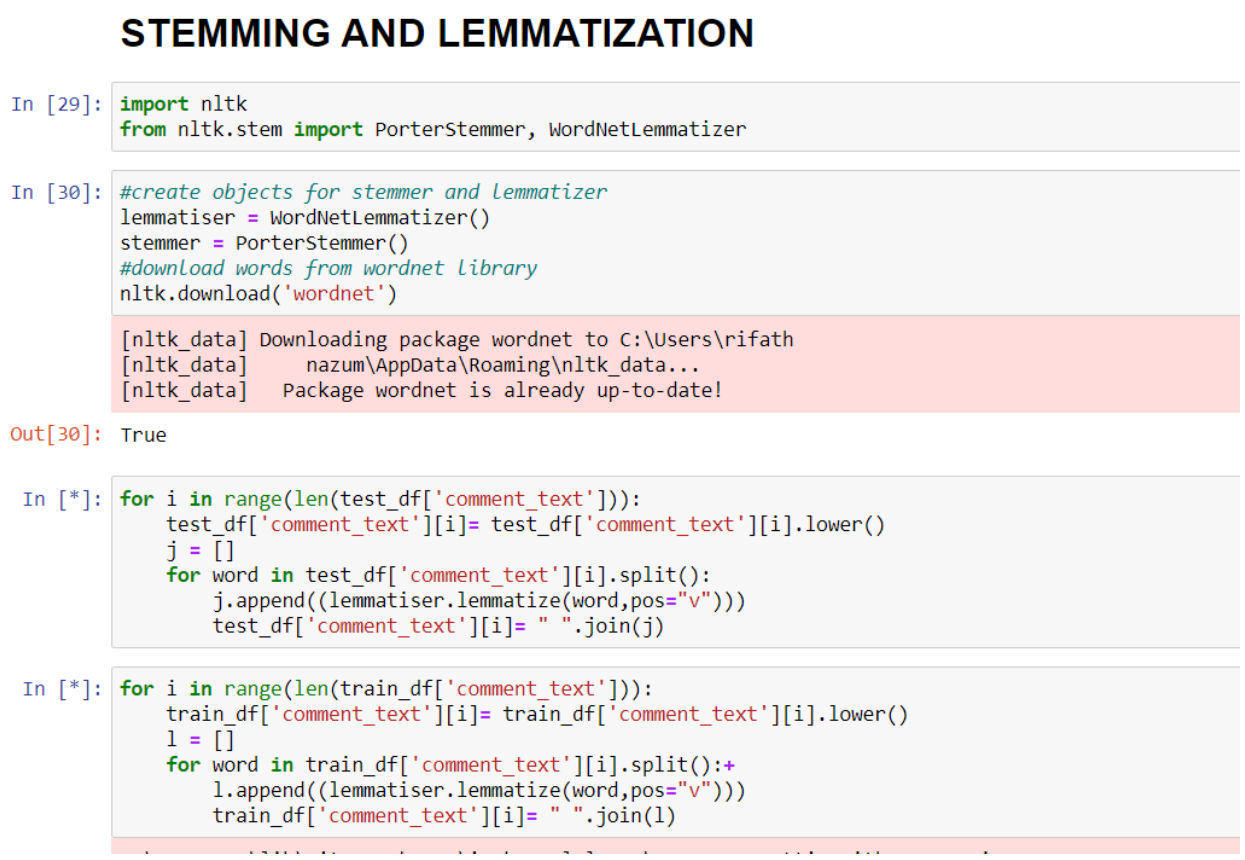
DATA PRE PROCESSING

```
In [23]: # Replace email addresses with 'email'
         train_df['comment_text'] = train_df['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddres

         # Replace URLs with 'webaddress'
         train_df['comment_text'] = train_df['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2

         # Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
         train_df['comment_text'] = train_df['comment_text'].str.replace(r'£|\$', 'dollers')

         # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenu
         train_df['comment_text'] = train_df['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]

         # Replace numbers with 'number'
         train_df['comment_text'] = train_df['comment_text'].str.replace(r'\d+(\.\d+)?', 'number')
         # Remove punctuation
         train_df['comment_text'] = train_df['comment_text'].str.replace(r'[^\w\d\s]', ' ')

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

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



CLEANING TEST DATA SET

```
In [24]: # Replace email addresses with 'email'
test_df['comment_text'] = test_df['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

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

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
test_df['comment_text'] = test_df['comment_text'].str.replace(r'£|\$', 'dollers')

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

# Replace numbers with 'number'
test_df['comment_text'] = test_df['comment_text'].str.replace(r'\d+(\.\d+)?', 'number')
# Remove punctuation
test_df['comment_text'] = test_df['comment_text'].str.replace(r'[^\w\d\s]', ' ')

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

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



STEMMING AND LEMMATIZATION

```
In [29]: import nltk
         from nltk.stem import PorterStemmer, WordNetLemmatizer

In [30]: #create objects for stemmer and lemmatizer
         lemmatiser = WordNetLemmatizer()
         stemmer = PorterStemmer()
         #download words from wordnet library
         nltk.download('wordnet')

         [nltk_data] Downloading package wordnet to C:\Users\rifath
         [nltk_data]     nazum\AppData\Roaming\nltk_data...
         [nltk_data]   Package wordnet is already up-to-date!

Out[30]: True

In [*]:  for i in range(len(test_df['comment_text'])):
             test_df['comment_text'][i]= test_df['comment_text'][i].lower()
             j = []
             for word in test_df['comment_text'][i].split():
                 j.append((lemmatiser.lemmatize(word,pos="v")))
                 test_df['comment_text'][i]= " ".join(j)

In [*]:  for i in range(len(train_df['comment_text'])):
             train_df['comment_text'][i]= train_df['comment_text'][i].lower()
             l = []
             for word in train_df['comment_text'][i].split():+
                 l.append((lemmatiser.lemmatize(word,pos="v")))
                 train_df['comment_text'][i]= " ".join(l)
```

Hardware and Softwares Used:

Software requirement: Anaconda, Jupyter notebook

Libraries and packages used: Numpy, Pandas, Sklearn,seaborn,Matplotlib,nltk.

Model/s Development and Evaluation

Problem-solving approach:

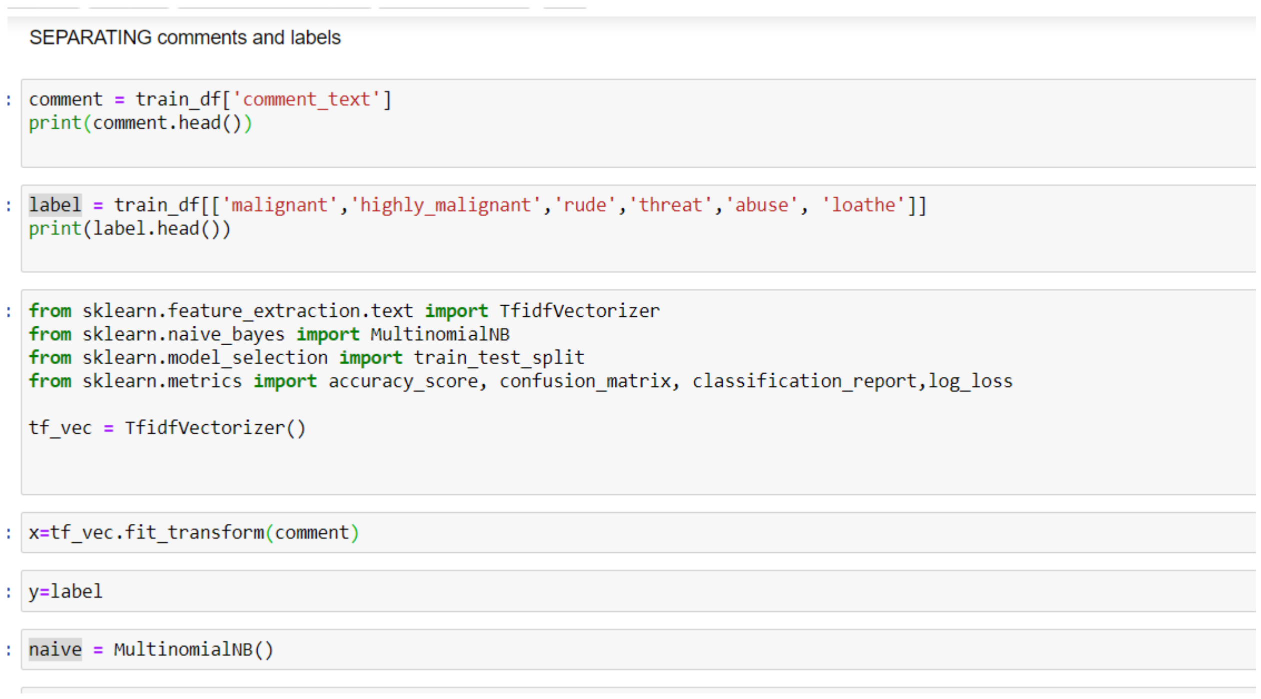
The given problem, based on the comment we have to predict whether the comment falls into one or more categories.so I used multinomialNB algorithm.

Testing of Identified Approaches (Algorithms):

List of algorithms used:

- MultinomialNB

Run and Evaluate selected models:



SEPARATING comments and labels

```
comment = train_df['comment_text']
print(comment.head())
```

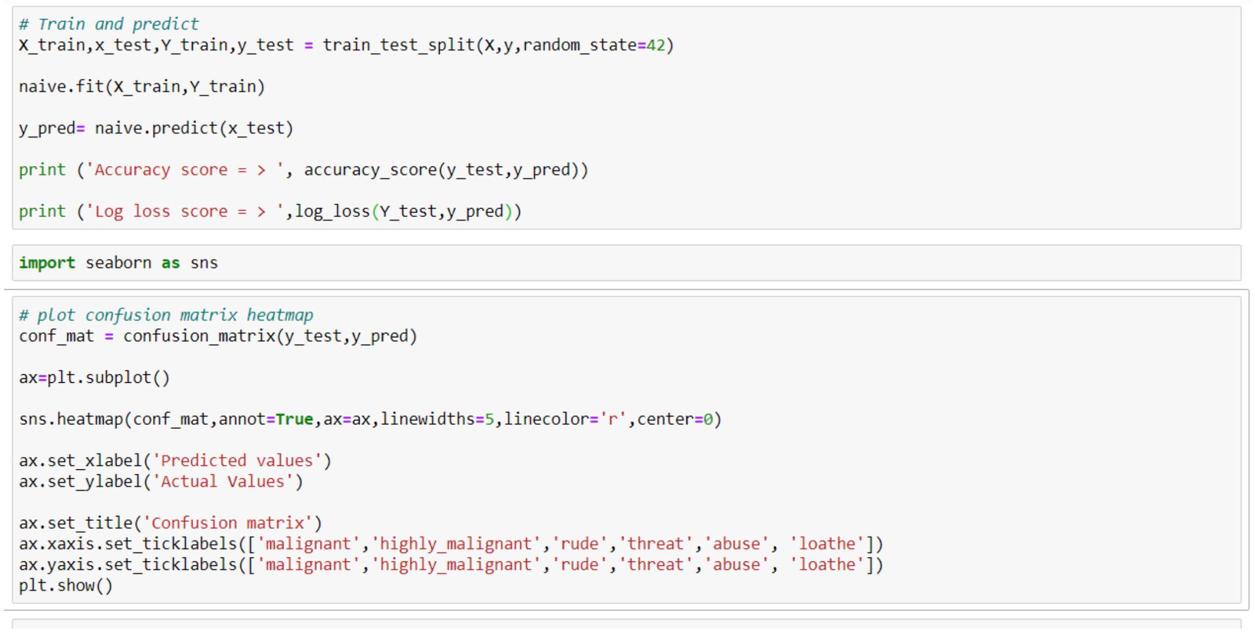
```
label = train_df[['malignant','highly_malignant','rude','threat','abuse', 'loathe']]
print(label.head())
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,log_loss

tf_vec = TfidfVectorizer()
```

```
x=tf_vec.fit_transform(comment)
```

```
y=label
```

```
naive = MultinomialNB()
```



```
# Train and predict
X_train,x_test,Y_train,y_test = train_test_split(X,y,random_state=42)

naive.fit(X_train,Y_train)

y_pred= naive.predict(x_test)

print ('Accuracy score = > ', accuracy_score(y_test,y_pred))

print ('Log loss score = > ',log_loss(Y_test,y_pred))
```

```
import seaborn as sns
```

```
# plot confusion matrix heatmap
conf_mat = confusion_matrix(y_test,y_pred)

ax=plt.subplot()

sns.heatmap(conf_mat,annot=True,ax=ax,linewidths=5,linecolor='r',center=0)

ax.set_xlabel('Predicted values')
ax.set_ylabel('Actual Values')

ax.set_title('Confusion matrix')
ax.xaxis.set_ticklabels(['malignant','highly_malignant','rude','threat','abuse', 'loathe'])
ax.yaxis.set_ticklabels(['malignant','highly_malignant','rude','threat','abuse', 'loathe'])
plt.show()
```

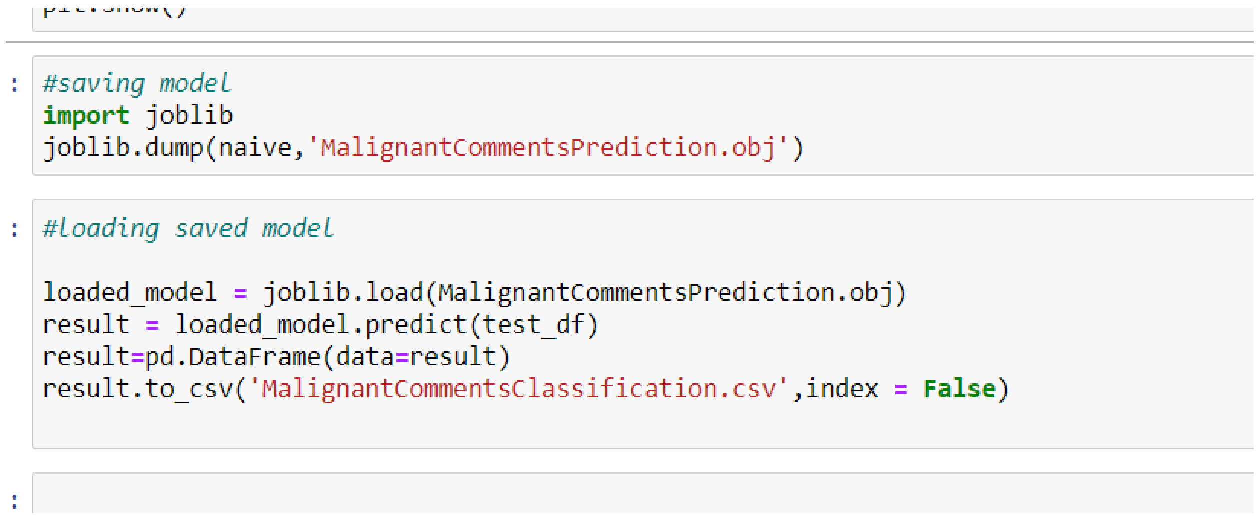
Key Metrics for success in solving problem under consideration:

A confusion matrix helps us gain an insight into how correct our predictions were and how they hold up against the actual values.

The following metrics are used :

1)Accuracy : Accuracy is the ratio of the total number of correct predictions and the total number of predictions.

2)Precision: Precision is the ratio between the True Positives and all the Positives



```
: #saving model
  import joblib
  joblib.dump(naive,'MalignantCommentsPrediction.obj')
```

```
: #loading saved model

  loaded_model = joblib.load(MalignantCommentsPrediction.obj)
  result = loaded_model.predict(test_df)
  result=pd.DataFrame(data=result)
  result.to_csv('MalignantCommentsClassification.csv',index = False)
```

```
:
```

CONCLUSION

- **Learning Outcomes of the Study in respect of Data Science**

One of the challenge i faced while data cleaning is cleaning the comments