MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

Jayashri Dandare

ACKNOWLEDGMENT

- I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Miss Khushboo Garg for her constant guidance and support.

- I have primarily referred to various articles scattered across various websites for the purpose of getting an idea on “Micro-credit defaulters” project.

- I would like to thank the technical support team also for helping me out and reaching out to me on clearing all my doubts as early as possible.

- I would like to draw my gratitude to Flip Robo and Data Trained for providing me a suitable environment and guidance to complete my work. Last but not least thanks to the brilliant authors from where I have got the idea to carry out the project.

- References were taken from various articles from Medium, Towards Data Science, Machine Learning Mastery, Analytics Vidya, American Statistical Association, Research Gate and documentations of Python and Sklearn.

TABLE OF CONTENTS

INTRODUCTION

- ## Business Problem Framing:

Social media has given a lot of things to people which beyond imagination. In this era of technology, it has become the hub of information. The numbers of contents on social media are vast and rich and everything has found a place on social media that may be anything. It has given wings to its users to fly high and express their feelings. It has become a boon for the mankind but we all know that if there is good there must be some bad. Likewise, social media has also got the dark side.

I would like to quote Tarana Burke who once told that “Social media is not a safe space.” It is absolutely true even though it has given a lot of things to the mankind it has also taken it toll. Now a days it is becoming a weapon to create disturbance in the society and personal life of people. Everyday the count of incidents of Online hate is increasing. So to face this problem effectively a machine learning model will be created. 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:

In the past few years its seen that the cases related to social media hatred have increased exponentially. The social media is turning into a dark venomous pit for people now a days. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc. In social media the people spreading or involved in such kind of activities uses filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.

The result of such activities can be dangerous. It gives mental trauma to the victims making their lives miserable. People who are not well aware of mental health online hate or cyber bullying become life threatening for them. Such cases are also at rise. It is also taking its toll on religions. Each and every day we can see an incident of fighting between people of different communities or religions due to offensive social media posts. Online hate, described as abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity has been identified as a major threat on online social media platforms. These kinds of activities must be checked for a better future.

- ## Motivation for the Problem Undertaken:

The project was the first provided to me by FlipRobo as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation was to classify the news in order to bring awareness and reduce unwanted chaos and make a good model which will help us to know such kind of miscreants.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem:

Anyone can be a victim of online hate or cyberbully. The social media has become a dangerous place to dwell in. The use of abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity have significantly high negative impact on individual. We can use Machine Learning and NLP technologies to deal with such toxic comments. We were provided with two different datasets. One for training and another to test the efficiency of the model created using the training dataset. The training dataset provided here has a shape of 159571 rows and 8 columns. As it is a multiclass problem it has 6 dependent / target column. Here the target or the dependent variables named “malignant, highly_malignant, rude, threat, abuse, loathe” have two distinct values 0 and 1. Where 1 represents yes and 0 represents no for each class. As the target columns are giving binary outputs and all the independent variables has text so it is clear that it is a supervised machine learning problem where we can use the techniques of NLP and classification-based algorithms of Machine learning.

Here we will use NLP techniques like word tokenization, lemmatization, stemming and tfidf vectorizer then those processed data will be used to create best model using various classification based supervised machine learning algorithms like Logistic Regression, Passive Aggressive Classifier, Multinomial NB, Complement NB with the help of OneVsRestClassifier which is helpful to deal with multilabel classification problems. The passive Aggressive Classifier belongs to the family of online machine learning algorithms and it is very much helpful in processing large scale data. It remains passive for a correct classification and turns aggressive in case of a misclassification. Its aim is to make updates that corrects the loss causing very little change in the weight vector.

• Data Sources and their formats:

The data was provided by FlipRobo in CSV format. After loading the training dataset into Jupyter Notebook using Pandas and using df.head() [Fig. 1] it can be seen that there are eight columns named as " id, comment_text, "malignant, highly_malignant, rude, threat, abuse, loathe". Similarly, the test file can be load using pandas and the first five rows of the dataset can be seen using df.head() method.

```
In [11]: train
```

Out[11]:

	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
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

```
In [12]: test
```

Out[12]:

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.
...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...

153164 rows × 2 columns

The metadata is provided below for better understanding of the data given.

- **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.

As mentioned earlier the shape of the training dataset is (159571, 8) and the shape of test dataset is (153164,2). The shape of the datasets in form of a tuple can be accessed using df.shape(). The column names of the datasets in form of a list can be seen using df.columns.values()

```
In [13]: print("In the training dataset\nNumber of columns=",train.shape[1],'\nNumber of Rows=',train.shape[0],
               '\nName of columns=\n',train.columns.values)

In the training dataset
Number of columns= 8
Number of Rows= 159571
Name of columns=
 ['id' 'comment_text' 'malignant' 'highly_malignant' 'rude' 'threat'
 'abuse' 'loathe']
```

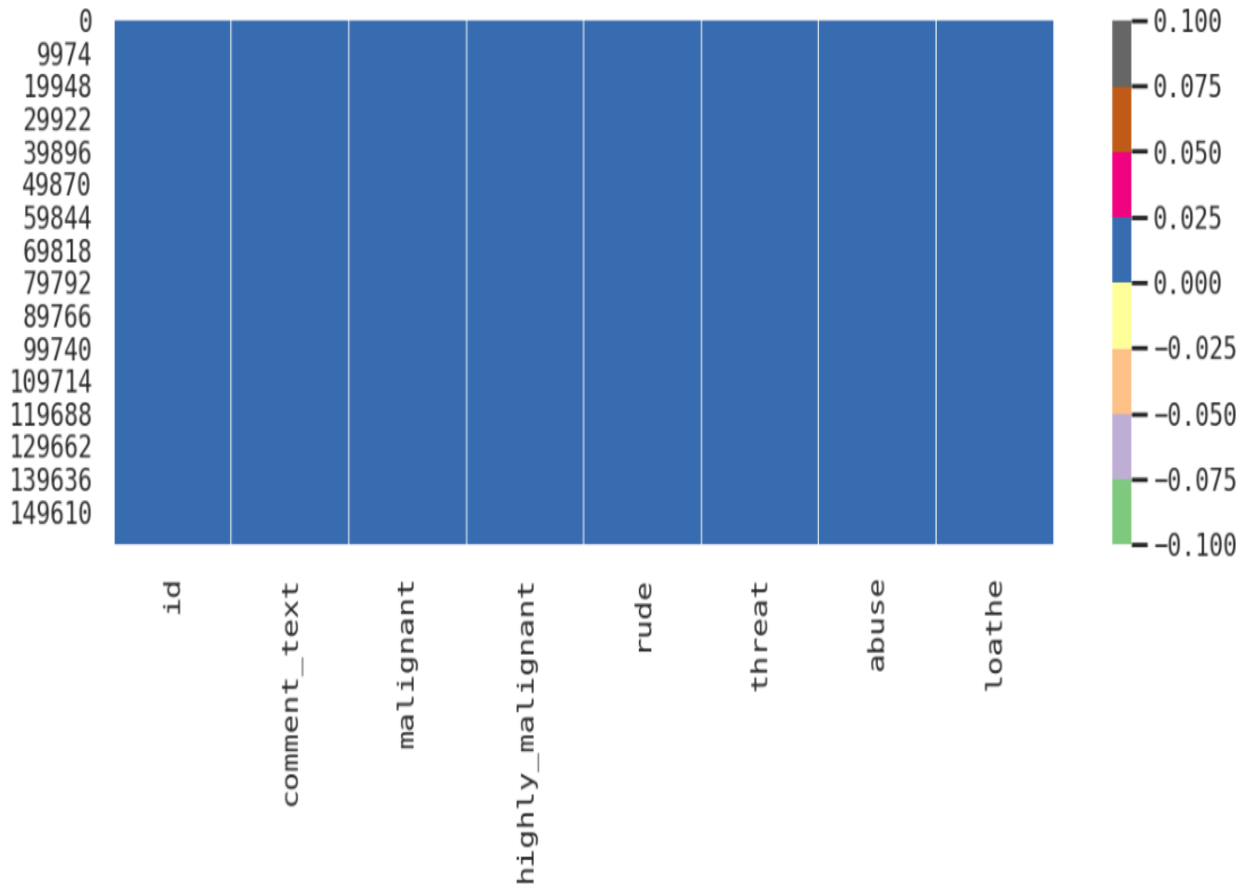
```
In [14]: print("In the test dataset\nNumber of columns=",test.shape[1],'\nNumber of Rows=',test.shape[0],
               '\nName of columns=\n',test.columns.values)

In the test dataset
Number of columns= 2
Number of Rows= 153164
Name of columns=
 ['id' 'comment_text']
```

The datasets have no duplicated values or null values. Both the datset have no trace of any null or duplicated values. The number of duplicated values of a dataset can be seen using df.duplicated().sum() and the null values can be seen using df.isnull().sum() as showing below.

```
In [15]: #checking if there is any duplicated values in training dataset
         print('Number of duplicated values:-',train.duplicated().sum())

         Number of duplicated values:- 0

In [16]: #checking if there is any duplicated values in test dataset
         print('Number of duplicated values:-',test.duplicated().sum())

         Number of duplicated values:- 0

In [17]: # Checking any null value present in dataset

         train.isnull().sum()

Out[17]: id                  0
         comment_text        0
         malignant           0
         highly_malignant    0
         rude                0
         threat              0
         abuse               0
         loathe              0
         dtype: int64

In [18]: # Checking any null value present in dataset

         test.isnull().sum()

Out[18]: id              0
         comment_text    0
         dtype: int64
```

The null values can also be visualized with the help of seaborn heatmap and matplotlib library. Visualization gives a better idea.

```
In [19]: #heatmap of null values
sns.set(context='talk',style='whitegrid',palette='dark',font='monospace',font_scale=0.8)
plt.figure(figsize=(12,4),dpi=120)
sns.heatmap(train.isnull(),cmap='Accent')
plt.show()
```



- ## Data Pre-processing Done:

After the dataset is loaded and the shape, null values and duplicated values were checked then the data- set is further treated where the unwanted column “id” is removed from the training dataset as we will work on the columns like “comment_text, “malignant, highly_malignant, rude, threat, abuse, loathe”. So a copy of the training dataset was made using df.copy() and the column was dropped from the new dataset using df.drop(). Similarly, the ‘id’ column is also dropped from the test dataset.

```
In [20]: #dropping unwanted columns as we'll be working on the comment_text and their categories_
         df=train.copy()
         df.drop(['id'],axis=1,inplace=True)
```

```
In [21]: test.drop(['id'],axis=1,inplace=True)
```

After removing the unwanted column, a new column named ‘normal’ was created in the training dataset which represents the statements not falling under malignant, highly_malignant, rude, threat, abuse, loathe category or statements where values of malignant, highly_malignant, rude, threat, abuse, loathe are 0.

```
In [22]: #adding a new column which represent a normal statement

         labels= ['malignant','highly_malignant','rude','threat','abuse','loathe']
         df['normal']=1-df[labels].max(axis=1)
```

```
In [23]: df
```

Out[23]:

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	normal
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	1
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	1
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	1
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	1
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...
159566	":::::And for the second time of asking, when ...	0	0	0	0	0	0	1
159567	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0	1
159568	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0	1
159569	And it looks like it was actually you who put ...	0	0	0	0	0	0	1
159570	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0	1

159571 rows × 8 columns

After the new column 'normal' was added and unwanted column 'id' was dropped a new column named "raw length" representing the string length of the 'comment_text' column is added to the dataset [Fig 8]. It'll help to know the length of the strings in 'comment_text' columns before pre-processing and later a new column will be created to compare the length of strings before and after pre-processing.

```
In [24]: #adding a column 'raw length' to the dataset which will show the length of characters in column 'comment_text'
df['raw length']= df.comment_text.str.len().astype('int64')
df
```

Out[24]:

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

159571 rows × 9 columns

After that we can check the which label carries how many comments using df.value_count() method. It'll briefly show the count of numbers of 0 and 1 of all dependent columns / target columns.

```
In [25]: #value counts of  label columns
         values=['malignant','highly_malignant','rude','threat','abuse','loathe']
         for i in values:
             vc=df[i].value_counts()
             print('VALUE COUNT OF UNIQUE VALUES IN ' +"'"+ i+"' :\n ",vc,'\n')

VALUE COUNT OF UNIQUE VALUES IN 'malignant' :
  0    144277
1     15294
Name: malignant, dtype: int64

VALUE COUNT OF UNIQUE VALUES IN 'highly_malignant' :
  0    157976
1      1595
Name: highly_malignant, dtype: int64

VALUE COUNT OF UNIQUE VALUES IN 'rude' :
  0    151122
1      8449
Name: rude, dtype: int64

VALUE COUNT OF UNIQUE VALUES IN 'threat' :
  0    159093
1       478
Name: threat, dtype: int64

VALUE COUNT OF UNIQUE VALUES IN 'abuse' :
  0    151694
1      7877
Name: abuse, dtype: int64

VALUE COUNT OF UNIQUE VALUES IN 'loathe' :
  0    158166
1      1405
Name: loathe, dtype: int64
```

We can also check the count of 1 (yes case) for each label which will show the number of malignant, highly_malignant, rude, threat, abuse, loathe, normal comments.

```
In [27]: #COUNT OF DIFFERENT LABELS
         x=df.iloc[:,2:-1].sum()
         x

Out[27]: highly_malignant      1595
         rude                  8449
         threat                 478
         abuse                 7877
         loathe                1405
         normal              143346
         dtype: int64
```

It can be seen that there are comments which represents more than 1 category of labels this can also be checked and it'll be helpful for more understanding.

```
In [29]: #CHECKING THE COUNT OF COMMENTS WITH 1 OR MORE THAN 1 LABELS
         summation=df.iloc[:,2:-1].sum(axis=1) #not including comment_text and raw length column
         vc=summation.value_counts()
         vc

Out[29]: 1    147303
         0      5666
         2      4406
         3      1780
         4       385
         5        31
         dtype: int64
```

After the columns to show anew category and to show the raw length of strings were created and unnecessary column 'id' was dropped the df.info() was used to get the detailed summary of the training and test dataset.

In [31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   comment_text      159571 non-null  object
 1   malignant         159571 non-null  int64 
 2   highly_malignant  159571 non-null  int64 
 3   rude              159571 non-null  int64 
 4   threat            159571 non-null  int64 
 5   abuse             159571 non-null  int64 
 6   loathe            159571 non-null  int64 
 7   normal            159571 non-null  int64 
 8   raw length        159571 non-null  int64 
dtypes: int64(8), object(1)
memory usage: 11.0+ MB
```

In [32]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153164 entries, 0 to 153163
Data columns (total 1 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   comment_text  153164 non-null  object
dtypes: object(1)
memory usage: 1.2+ MB
```

After the basic EDA is done the NLP techniques were implemented for processing the texts in the ‘comment_text’ columns. For this process a list of stopwords were created manually.

In [34]:
```
stopwords=['i','me','my','myself','we','our','ours','ourselves','you',"you're","you've","you'll","you'd",'yo','nothin','from','be
           'your','yours','yourself','yourselves','he','him','his','himself','she',"she's",'her','hers','ur','mlm','nbfc',"he's",
           'herself','it',"it's",'its','itself','they','them','their','theirs','themselves','what','which','lol','lool','fwiw','a
           'who','whom','this','that',"that'll",'these','how','these','those','am','is','are','was','were','oh','hay','thanks','t
           'be','been','being','have','has','had','having','do','does','did','done','doing','a','an','the','even','aww','bye!','b
           'between','into','through','during','before','after','above','below','to','from','up','down','in','out','on','off','ov
           'all','any','both','each','few','more','most','other','some','such','no','nor','not','only','own','same','so','than',
           'o','re','ve','y','ain','aren',"aren't",'couldn',"couldn't",'didn',"didn't",'doesn',"doesn't",'hadn',"hadn't",'hasn','
           "needn't",'shan',"shan't",'shouldn',"shouldn't",'wasn',"wasn't",'weren',"weren't",'won',"won't",'wouldn',"wouldn't", '
           "there's","You've",'got',"i'd",'everything','true','yes','moreover','would','could','like','mr.','but',"i'm",'able','b
          'must','see','went','saw','many','whats','id','let','day','never','yet','im','go','thatll','theyre','came','youll','con
           'october','november','december','everyone','hey','ok','okay','cant','bbq','let','thats','also','time','name','oh','sai
```

In the preprocessing the string converted to lower case as it is easier to understand for the machine then from the strings the stopwords, special characters, digits were dropped using proper techniques. After those unnecessary characters were removed the string is tokenized using word_tokenization() function of NLTK library then those tokenized words were checked for stopwords and token length of 3. If both the condition were satisfied the tokenized words were lemmatized and stemmed using wordnetLemmatizer() and PorterStemmer() which brings back all words to their root form. Then again, 1 8 MALIGNANT COMMENTS CLASSIFICATION those tokenized words were joined to form a string. All these operations were compiled inside a function.

```
#CREATING A FUNCTION TO PERFORM ASERIES OF OPERATIONS

def preprocess(text):
    processed=[]
    lower=text.lower().replace(r'\n'," ").replace(r'^.+@[^\.].*\.[a-z]{2,}$',' ').replace(r'^http://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,
    #converting to lower case and replacing mail id,links by white space

    text=lower.replace(r'\s+', ' ').replace(r'\d+(\.\d+)?', ' ')
    #removing \n,large white space and leading_trailing white spaces, numbers by white space

    text=lower.replace(r"[^a-zA-Z]+", " ").replace(r"—"," ").replace(r'“', ' ').replace(r'”',' ').replace('’',' ').replace(r'-','
    text=text.replace('®',' ').replace('´',' ').replace('·',' ').replace('½',' ').replace('¾',' ').replace('¤',' ')
    #removing special characters by single white space



    punct=text.translate(str.maketrans('', '', p))  #remove punctuation
    digit=punct.translate(str.maketrans('', '', d))       #remove digits if any
    word= wt(digit, "english")

    for i in word:
        if i not in stopwords and len(i)>=3 and len(i)<12:
            lemma=porter().stem(wl().lemmatize(i))
           # lemma=wl().lemmatize(i)
            #stem=porter.stem(lemma)
            processed.append(lemma)
    return (" ".join([x for x in processed])).strip()
```

After the function was created a test run was done on a sample text to check the effectiveness of the function. After successful testing the entire ‘comment_text’ column was processed using the function created to get a clear and pure form of data for further operations.

```
#TESTING THE FUNCTION CREATED ABOVE
sample=" As much as human rights and ethnic rights should be respected, spray painting every possible detail of unverifiable info
on the Rohingya, and getting around the verification by claiming that the information was destroyed by an interested party - \
are not valid reasons for having a list of villages where a certain group of people live. There is already a lot of articles on t
Arakanese people and state that have no concern of the Rohingya but include them for the sake of brotherly respect - this is push
line a bit far. Rohingyas should be treated fairly - I do not contest that. But articles like this one - are pure self-pitying ar
Wikipedia with absolutely useless information. I wonder when will somebody change the name of the article on Burma/Myanmar on \
wiki to ""Country where the Rohingya are Persecuted"".\nRather, a brief mention of where the Rohingyas reside should be placed if
on the main article on Rakhine state - albeit short and concise, not dump an entire list of names copied directly from some publi
all due respect, this article should be deleted."

print("Original Document: \n",sample)

processed=[]
for word in sample.split(' '):
    processed.append(word)
print('\n',processed)
print("\n\nTokenized and lemmatized document: \n")
print(preprocess(sample))
```

Test Results

```
Original Document:
  As much as human rights and ethnic rights should be respected, spray painting every possible detail of unverifiable informati
onon the Rohingya, and getting around the verification by claiming that the information was destroyed by an interested party -
are not valid reasons for having a list of villages where a certain group of people live. There is already a lot of articles on
the Arakanese people and state that have no concern of the Rohingya but include them for the sake of brotherly respect - this i
s pushing theline a bit far. Rohingyas should be treated fairly - I do not contest that. But articles like this one - are pure
self-pitying and clutters Wikipedia with absolutely useless information. I wonder when will somebody change the name of the art
icle on Burma/Myanmar on wiki to Country where the Rohingya are Persecuted.
Rather, a brief mention of where the Rohingyas reside should be placed if desiredon the main article on Rakhine state - albeit
short and concise, not dump an entire list of names copied directly from some publication.
Withall due respect, this article should be deleted.

 ['', 'As', 'much', 'as', 'human', 'rights', 'and', 'ethnic', 'rights', 'should', 'be', 'respected,', 'spray', 'painting', 'eve
ry', 'possible', 'detail', 'of', 'unverifiable', 'informationon', 'the', 'Rohingya,', 'and', 'getting', 'around', 'the', 'verif
ication', 'by', 'claiming', 'that', 'the', 'information', 'was', 'destroyed', 'by', 'an', 'interested', 'party', '-', 'are', 'n
ot', 'valid', 'reasons', 'for', 'having', 'a', 'list', 'of', 'villages', 'where', 'a', 'certain', 'group', 'of', 'people', 'liv
e.', 'There', 'is', 'already', 'a', 'lot', 'of', 'articles', 'on', 'the', 'Arakanese', 'people', 'and', 'state', 'that', 'hav
e', 'no', 'concern', 'of', 'the', 'Rohingya', 'but', 'include', 'them', 'for', 'the', 'sake', 'of', 'brotherly', 'respect', '-
', 'this', 'is', 'pushing', 'theline', 'a', 'bit', 'far.', 'Rohingyas', 'should', 'be', 'treated', 'fairly', '-', 'I', 'do', 'n
ot', 'contest', 'that.', 'But', 'articles', 'like', 'this', 'one', '-', 'are', 'pure', 'self-pitying', 'and', 'clutters', 'Wiki
pedia', 'with', 'absolutely', 'useless', 'information.', 'I', 'wonder', 'when', 'will', 'somebody', 'change', 'the', 'name', 'o
f', 'the', 'article', 'on', 'Burma/Myanmar', 'on', 'wiki', 'to', 'Country', 'where', 'the', 'Rohingya', 'are', 'Persecuted.\nRa
ther,', 'a', 'brief', 'mention', 'of', 'where', 'the', 'Rohingyas', 'reside', 'should', 'be', 'placed', 'if', 'desiredon', 'th
e', 'main', 'article', 'on', 'Rakhine', 'state', '-', 'albeit', 'short', 'and', 'concise,', 'not', 'dump', 'an', 'entire', 'lis
t', 'of', 'names', 'copied', 'directly', 'from', 'some', 'publication.\nWithall', 'due', 'respect,', 'this', 'article', 'shoul
d', 'be', 'deleted.']


Tokenized and lemmatized document:

human right ethnic right respect spray paint everi possibl detail rohingya get around claim inform destroy interest parti valid
reason list villag certain group peopl live alreadi lot articl arakanes peopl concern rohingya includ sake brotherli respect pu
sh thelin far rohingya treat fairli contest articl one pure selfpiti clutter wikipedia absolut useless inform wonder somebodi c
hang articl wiki countri rohingya persecut rather brief mention rohingya resid place desiredon main articl rakhin albeit short
concis dump entir list name copi directli public withal due respect articl delet
```

```
%%time
clean = []

for i in df.comment_text:
    clean.append(preprocess(i))
```

```
Wall time: 7min 2s
```

After the procedure was completed a list of cleaned data were obtained which was added to the dataset by column name 'comment' and another column name 'len of clean comment' was added showing the length of words in the 'comment' column. Further calculation revelled that there were total of 62893130 words were present in the raw 'comment_text' column and after processing it became 29977753 the pre-processing led to a reduction of 32915377 strings.

```
#USING THE EXTRACTED FEATURE AS ''comment" also adding an extra column to represent the length of string of the cleaned comments
processed = pd.DataFrame({'comment' : clean })
df['comment']= processed

df['len of cleaned comment']=df['comment'].str.len().astype('int64')
df
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	normal	raw length	comment	len of cleaned comment
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	1	264	explan edit made usernam hardcor metallica fan...	141
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	1	112	match background colour seemingli stuck talk	44
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	1	233	man realli tri edit war guy constantli remov r...	114
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	1	622	make real suggest improv wonder section statis...	250
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	1	67	sir hero chanc rememb page	26
...	...	...	...	...	...	...	...	...	...	...	...
159566	":::::And for the second time of asking, when ...	0	0	0	0	0	0	1	295	second ask view complet contradict coverag rel...	137
159567	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0	1	99	asham horribl thing put talk page	33
159568	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0	1	81	spitzer umm there actual articl ring crunch ca...	51
159569	And it looks like it was actually you who put ...	0	0	0	0	0	0	1	116	look actual who put speedi first version delet...	51
159570	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0	1	189	realli think understand idea bad kind commun g...	76

159571 rows × 11 columns

In [42]:
```
print('Original Length = ',df['raw length'].sum())
print('Clean Length =  ', df['len of cleaned comment'].sum())
print('Total Reduction = ',df['raw length'].sum()-df['len of cleaned comment'].sum())
```

```
Original Length =  62893130
Clean Length =   29977753
Total Reduction =  32915377
```

Similar step was used on test data. The dataset was processed using the function created and after the processing is done the processed list added to the test dataset as a new column named 'comment'.

```
%%time
comments = []

for i in test.comment_text:
    comments.append(preprocess(i))
```

```
Wall time: 6min 29s
```

```
#USING THE EXTRACTED FEATURE AS ''comment" also adding an extra column to represe
processed = pd.DataFrame({'comment' : comments })
test['comment']= processed
test
```

	comment_text	comment
0	Yo bitch Ja Rule is more succesful then you'll...	bitch rule succes ever hate sad mofuckasi bitc...
1	== From RfC == \n\n The title is fine as it is...	rfc titl fine imo
2	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	sourc zaw ashton lapland
3	:If you have a look back at the source, the in...	look sourc inform updat correct form guess sou...
4	I don't anonymously edit articles at all.	anonym edit articl
...	...	...
153159	. \n i totally agree, this stuff is nothing bu...	total agre stuff noth toolongcrap
153160	== Throw from out field to home plate. == \n\n...	throw field home plate faster throw cut man di...
153161	" \n\n == Okinotorishima categories == \n\n I ...	categori chang agre correct gotten confus foun...
153162	" \n\n == ""One of the founding nations of the...	one found nation germani law return quit simil...
153163	" \n :::Stop already. Your bullshit is not wel...	stop alreadi bullshit welcom fool think kind e...

153164 rows × 2 columns

After getting a cleaned data TF-IDF vectorizer will be used. It'll help to transform the text data to feature vector which can be used as input in our modelling. The TFIDF stands for Term Frequency Inverse Document Frequency. It is a common algorithm to transform text into vectors or numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in. Mathematically, TF-IDF =TF(t*d) * IDF (t,d) So here the dataset is divided into two parts X and Y. X represents the column 'comment' which carries the cleaned text and Y represents the labels like 'malignant, highly_malignant, rude, threat, abuse, loathe, normal'. After the splitting the tfidf vectorizer was initialized and X is fitted into it and converted into an array.

```
In [50]: X=df.comment
         y=df.iloc[:,1:-3]
```

```
In [51]: X.head(4)
```

```
Out[51]: 0    explan edit made usernam hardcor metallica fan...
         1         match background colour seemingli stuck talk
         2    man realli tri edit war guy constantli remov r...
         3    make real suggest improv wonder section statis...
         Name: comment, dtype: object
```

```
In [52]: y.head(4)
```

Out[52]:

	malignant	highly_malignant	rude	threat	abuse	loathe	normal
0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
3	0	0	0	0	0	0	1

```
In [53]: tfidf=tf(input='content', encoding='utf-8', lowercase=True,stop_words='english',max_features=10000,ngram_range=(1,3))
         x=tfidf.fit_transform(X).toarray()
```

Hardware and Software Requirements and Tools Used

- **Hardware:**

Windows specifications

Edition	Windows 10 Home Single Language
Version	21H2
Installed on	22-11-2021
OS build	19044.1741
Experience	Windows Feature Experience Pack 120.2212.4180.0

Device specifications

Device name	JAYASHRI
Processor	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Installed RAM	20.0 GB (17.7 GB usable)
Device ID	6D461F8E-D860-40CC-8403-520AD0F77092
Product ID	00327-36267-14244-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

- **SOFTWARE:**

- Jupyter Notebook (Anaconda 3) – Python 3.7.6
- Microsoft Excel 2010

- # LIBRARIES:

- The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.preprocessing.

In [8]:
```
#Importing all the required library
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from collections import Counter
from string import digits as d, punctuation as p
from nltk.tokenize import word_tokenize as wt
from nltk.stem import WordNetLemmatizer as wl, PorterStemmer as porter
from gensim import corpora

from sklearn.feature_extraction.text import TfidfVectorizer as tf
from sklearn.model_selection import train_test_split as tts, RandomizedSearchCV as rsv, cross_val_score as cvs
from sklearn.metrics import accuracy_score,classification_report,f1_score,auc,roc_curve,roc_auc_score,confusion_matrix,log_loss,\
precision_score,recall_score,mean_squared_error

from sklearn.linear_model import LogisticRegression,PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB, ComplementNB
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.multiclass import OneVsRestClassifier

from PIL import Image
import requests
from wordcloud import WordCloud

import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings('ignore',message="numpy.dtype size changed")
warnings.filterwarnings('ignore',message="numpy.ufunc size changed")
import joblib
```

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods):

After TF-IDF implementation array conversion we have x and y for modelling. Then x and y were split for training and testing using train_test_split in a ratio of 70:30 respectively. After split the shape of x_train and x_test found to be (111699,10000) and (47872, 10000) and y_train and y_test found to be (111699,7) and (47872,7) respectively.

```
In [55]: x_train,x_test,y_train,y_test=tts(x,y,test_size=0.30,random_state=95)
```

```
In [56]: print('shape of x_train:',x_train.shape,'\nshape of x_test:',x_test.shape)

         print('shape of y_train:',y_train.shape,'\nshape of y_test:',y_test.shape)

shape of x_train: (111699, 10000)
shape of x_test: (47872, 10000)
shape of y_train: (111699, 7)
shape of y_test: (47872, 7)
```

- ## Testing of Identified Approaches (Algorithms):

As it is a multi-label classification problem, we will use OneVsRestClassifier from sklearn with other classification algorithms like;

Logistic Regression()

Passive Aggressive Classifier()

Multinomial NB()

Complement NB()

During modelling various metrices like f1_score, confusion matrix, accuracy score, classification report, roc curve, roc auc score, mean squared error, precision score, recall score, log loss will be used to determine the performance of the model. At each step at the end of a model a data frame will be generated which will show the performance of the model per class. This will be executed with the help of pipe line.

Here for each model I have created a number of list named as F1, ACCURACY, PRECISION, RECALL, RMSE, MSE, AUC, LOG_LOSS to hold the values of matrices like f1 scores, accuracy scores, precision values, recall values, root mean squared error values, mean squared error values, auc scores, tpr values, fpr values, cross validation with f1 values, log loss values respectively. Here a pipeline has been created where the algorithm will run under OneVsRestClassifier. It'll also show the confusion matrix, accuracy score, classification report, roc curve, auc, roc auc score, mean squared error, precision score, recall score, tpr, fpr, f1 score, log loss value along with AUC Curves and Heatmap of confusion matrix for each label. The values obtained will be added to their respective lists. Below are few images of function performed on algorithms. Showing the metrics, heatmap of confusion matrix, AUC ROC curve. Below is the image of the pipeline created to achieve the required metrics and graphs.

```
LogReg_pipeline = Pipeline([('clf', OneVsRestClassifier(LogisticRegression(solver='sag'), n_jobs=1))])
F1=[]
ACCURACY = []
PRECESION = []
RECALL = []
RMSE = []
MSE = []
AUC=[]
TPR=[]
FPR=[]
CV_ACC=[]
LOG_LOSS=[]

for category in labels:
    print('Processing {}'.format(category))
    print('-------------------------------------')
    LogReg_pipeline.fit(x_train, y_train[category])
    pred = LogReg_pipeline.predict(x_test)
    f1=f1_score(pred,y_test[category])
    acc=accuracy_score(pred,y_test[category])
    clr=classification_report(y_test[category],pred)
    pre=precision_score(y_test[category],pred)
    rec=recall_score(y_test[category],pred)
    mse=mean_squared_error(y_test[category],pred)
    rmse=np.sqrt(mse)
    log = log_loss( y_test[category],pred)
    auc_scr=roc_auc_score(y_test[category],pred)
    tpr,fpr,threshold=roc_curve(y_test[category],pred)
    conf=confusion_matrix(y_test[category],pred)
```

```python
print('ACCURACY SCORE:', acc)
print('\nF1 score:',f1)
print('\nCLASSIFICATION REPORT:\n',clr)
print('\nPRECISION:\n',pre)
print('\nRECALL:\n',rec)
print('\nMEAN SQUARED ERROR:\n',mse)
print('\nROOT MEAN SQ. ERROR:\n',rmse)
print('\nLOG_LOSS:',log)
print('\nAUC_ROC Score:\n',auc_scr)
print('\nTPR:',tpr,'\nFPR:',fpr)

print('\n')
#plotting the auc_roc curve
print('\n\n\t________________________________________ROC_AUC CURVE____________________________________________')
print()
sns.set(context='talk',style='whitegrid',palette='dark',font='monospace',font_scale=1)
plt.figure(figsize=(8,3),dpi=120)
plt.plot([0,1],[0,1],color='lime',linestyle=":",lw=3)
plt.plot(tpr,fpr,label="AUC= %0.2f" % auc_scr,color='deepskyblue',lw=3,linestyle='--')
plt.legend(fancybox=True,shadow=True, fontsize='medium')

plt.xlabel("TPR",weight='bold',fontsize=10)
plt.ylabel('FPR',weight='bold',fontsize=10)
plt.title('RECEIVER OPERATING CHARACTERISTICS CURVE\n',size=10,weight='bold',loc='center')
plt.show()
print('\n')
#plotting confusion matrix
print('\n\n\t______________________________________CONFUSION MATRIX_____________________________________\n')
sns.set(context='talk',style='whitegrid',palette='dark',font='monospace',font_scale=1.3)
plt.figure(figsize=(7,4),dpi=120)
sns.heatmap(conf,annot=True,fmt='.2f',vmax=1,vmin=0,cmap='nipy_spectral',linewidths=0.8, linecolor='blue')
plt.title('HEATMAP OF CONFUSION MATRIX\n',size=10,weight='bold',loc='center')
plt.show()
print('\n')
```

```python
    ACCURACY.append(acc)
    F1.append(f1)
    PRECESION.append(pre)
    RECALL.append(rec)
    RMSE.append(rmse)
    MSE.append(mse)
    AUC.append(auc_scr)
    TPR.append(tpr)
    FPR.append(fpr)
    LOG_LOSS.append(log)

#creating a dataframe to show the performance of the model

logi_results = pd.DataFrame({"LABELS":labels,"F1":F1,'Acuracy':ACCURACY,'Precision': PRECESION ,'Recall': RECALL,
                             'RMSE':RMSE,'MSE':MSE,'AUC':AUC,'LOG_LOSS':LOG_LOSS})

logi_results.style.set_properties(**{'background-color':'midnightblue','color': 'lime','border-color': 'darkorange'})
```

- # METRICE OF EVALUATION

In the modeling I have chosen metrices like F1 score, Accuracy score, Precision, Recall, Mean Squared Error, Root Mean Square Error as my evaluation criteria. All the values were stored in a list and later they were saved in form of a DataFrame for proper evaluation and visualization of the values. Below are the result obtained using various algorithms with OneVsRestClassifier().

```
RESULTS OBTAINED FROM LOGISTIC REGRESSION
```

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.724075	0.956091	0.905747	0.603105	0.209544	0.043909	0.798238	1.516560
1	highly_malignant	0.266667	0.990349	0.459016	0.187919	0.098238	0.009651	0.592916	0.333326
2	rude	0.746009	0.977398	0.909559	0.632312	0.150339	0.022602	0.814414	0.780646
3	threat	0.120000	0.997243	0.600000	0.066667	0.052511	0.002757	0.533270	0.095236
4	abuse	0.622527	0.970108	0.813793	0.504058	0.172894	0.029892	0.749064	1.032445
5	loathe	0.265655	0.991916	0.679612	0.165094	0.089911	0.008084	0.582199	0.279214
6	normal	0.975653	0.955423	0.958227	0.993725	0.211133	0.044577	0.804460	1.539673

RESULTS OBTAINED FROM PASSIVE AGGRESSIVE CLASSIFIER

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.614474	0.900631	0.488153	0.828996	0.315229	0.099369	0.868596	3.432155
1	highly_malignant	0.389328	0.987091	0.348673	0.440716	0.113620	0.012909	0.716478	0.445882
2	rude	0.751057	0.975393	0.800811	0.707123	0.156867	0.024607	0.848689	0.849913
3	threat	0.334884	0.997013	0.450000	0.266667	0.054655	0.002987	0.632872	0.103173
4	abuse	0.643072	0.965241	0.645842	0.640325	0.186439	0.034759	0.811136	1.200559
5	loathe	0.394062	0.990621	0.460568	0.344340	0.096846	0.009379	0.670368	0.323948
6	normal	0.974413	0.953480	0.963574	0.985498	0.215685	0.046520	0.827286	1.606767

RESULTS OBTAINED FROM MULTINOMIAL NB

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.640663	0.947464	0.924155	0.490269	0.229207	0.052536	0.743010	1.814530
1	highly_malignant	0.237537	0.989138	0.344681	0.181208	0.104222	0.010862	0.588980	0.375173
2	rude	0.639857	0.970442	0.887712	0.500199	0.171924	0.029558	0.748347	1.020899
3	threat	0.027397	0.995551	0.035714	0.022222	0.066704	0.004449	0.510263	0.153677
4	abuse	0.542460	0.966348	0.809322	0.407945	0.183445	0.033652	0.701502	1.162311
5	loathe	0.092035	0.989284	0.184397	0.061321	0.103518	0.010716	0.529449	0.370122
6	normal	0.970960	0.946441	0.946978	0.996189	0.231429	0.053559	0.750365	1.849919

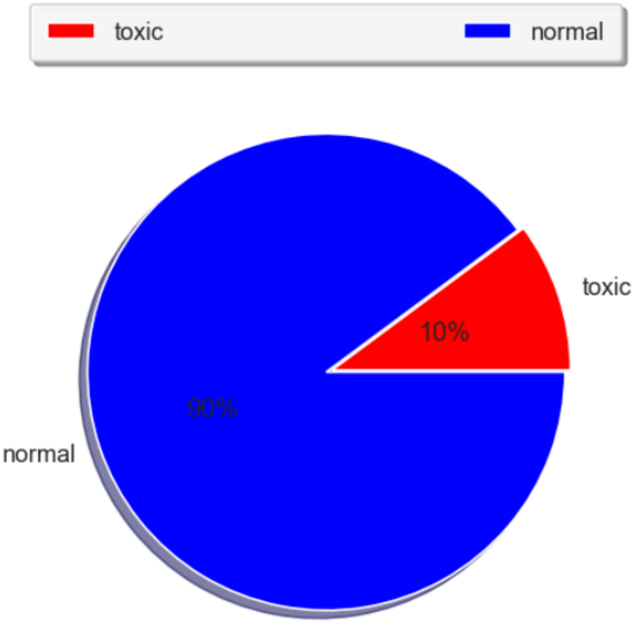
RESULTS OBTAINED FROM COMPLEMENT NB

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.590730	0.884713	0.446925	0.870982	0.339539	0.115287	0.878573	3.981940
1	highly_malignant	0.204190	0.937312	0.115824	0.861298	0.250376	0.062688	0.899663	2.165216
2	rude	0.494992	0.906271	0.345104	0.875050	0.306152	0.093729	0.891525	3.237359
3	threat	0.061586	0.949073	0.032481	0.592593	0.225671	0.050927	0.771337	1.759012
4	abuse	0.473946	0.904892	0.324834	0.876121	0.308396	0.095108	0.891246	3.284978
5	loathe	0.160317	0.929103	0.089552	0.764151	0.266266	0.070897	0.847364	2.448764
6	normal	0.924862	0.872723	0.985182	0.871502	0.356759	0.127277	0.877535	4.395998

RESULTS OBTAINED FROM LINER SVC

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.590730	0.884713	0.446925	0.870982	0.339539	0.115287	0.878573	3.981940
1	highly_malignant	0.204190	0.937312	0.115824	0.861298	0.250376	0.062688	0.899663	2.165216
2	rude	0.494992	0.906271	0.345104	0.875050	0.306152	0.093729	0.891525	3.237359
3	threat	0.061586	0.949073	0.032481	0.592593	0.225671	0.050927	0.771337	1.759012
4	abuse	0.473946	0.904892	0.324834	0.876121	0.308396	0.095108	0.891246	3.284978
5	loathe	0.160317	0.929103	0.089552	0.764151	0.266266	0.070897	0.847364	2.448764
6	normal	0.924862	0.872723	0.985182	0.871502	0.356759	0.127277	0.877535	4.395998

- **Visualizations:**
  - Visualization plays a crucial role in EDA as well as during modelling. It gives a better idea about the things going on beautifully. Below are the few visualizations used during this project to understand the dataset and performance of the algorithms.



- **Interpretation of the**

- ## Results

Basing on the result obtained ‘Logistic Regression’ have performed well and has given better result as compared to other models so it has been selected as final model and it will be saved using joblib library

```
In [65]: joblib.dump(LogReg_pipeline,'MALIGNANT_COMMENT.pkl')

Out[65]: ['MALIGNANT_COMMENT.pkl']

In [66]: #loading the model
         model=joblib.load('MALIGNANT_COMMENT.pkl')
```

- ## Testing

  After the model is saved it was again load into the system by joblib.load() method along with a variable name. From the test dataset the processed column "comment" was then transformed into vectors using tfidf vectorizer and then the result was predicted for possible classes using the model load.

```
In [67]: test
```

Out[67]:

	comment_text	comment
0	Yo bitch Ja Rule is more succesful then you'll...	bitch rule succes ever hate sad mofuckasi bitc...
1	== From RfC == \n\n The title is fine as it is...	rfc titl fine imo
2	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	sourc zaw ashton lapland
3	:If you have a look back at the source, the in...	look sourc inform updat correct form guess sou...
4	I don't anonymously edit articles at all.	anonym edit articl
...	...	...
153159	. \n i totally agree, this stuff is nothing bu...	total agre stuff noth toolongcrap
153160	== Throw from out field to home plate. == \n\n...	throw field home plate faster throw cut man di...
153161	" \n\n == Okinotorishima categories == \n\n I ...	categori chang agre correct gotten confus foun...
153162	" \n\n == ""One of the founding nations of the...	one found nation germani law return quit simil...
153163	" \n :::Stop already. Your bullshit is not wel...	stop alreadi bullshit welcom fool think kind e...

153164 rows × 2 columns

```
In [68]: X=test['comment']
         X
```

```
Out[68]: 0          bitch rule succes ever hate sad mofuckasi bitc...
         1                                       rfc titl fine imo
         2                                sourc zaw ashton lapland
         3          look sourc inform updat correct form guess sou...
         4                                      anonym edit articl
                                          ...
         153159                    total agre stuff noth toolongcrap
         153160     throw field home plate faster throw cut man di...
         153161     categori chang agre correct gotten confus foun...
         153162     one found nation germani law return quit simil...
         153163     stop alreadi bullshit welcom fool think kind e...
         Name: comment, Length: 153164, dtype: object
```

```
In [69]: tfidf=tf(input='content', encoding='utf-8', lowercase=True,stop_words='english',max_features=10000,ngram_range=(1,3))
         test_x=tfidf.fit_transform(X)

In [70]: test_x.shape

Out[70]: (153164, 10000)

In [71]: result=model.predict(test_x)
```

CONCLUSION

- ## Key Findings and Conclusions of the Study:

From the above analysis the below mentioned results were achieved which depicts the chances and conditions of a comment being a hateful comment or a normal comment; o With the increasing popularity of social media, more and more people consume feeds from social media and due differences they spread hate comments to instead of love and harmony. It has strong negative impacts on individual users and broader society.

- ## Learning Outcomes of the Study in respect of Data Science:

It is possible to classify the comments content into the required categories of authentic and however, using this kind of project an awareness can be created to know what is fake and authentic.

- ## Limitations of this work and Scope for Future Work

- Every effort has been put on it for perfection but nothing is perfect and this project is of no exception. There are certain areas which can be enhanced. Comment 3 7 MALIGNANT COMMENTS CLASSIFICATION detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field.