MALIGNANT COMMENTS CLASSIFICATION

ACKNOWLEDGEMENT

First of all, I extol the Almighty for pouring his blessings on me and giving me potentiality and opportunity to carry the work to its end with a success.

“It’s impossible to prepare a project report without the help and fillip of some people and certainly this project report is of no exception.”

At the commencement of this project report I would like to evince my deepest sense of gratitude to Mr. Shubham yadav my honoured mentor. Without her guidance, insightful decision, valuable comments and correction it would not have possible to reach up to this mark.

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, KDnuggets, Towards Data Science, Machine Learning Mastery, Analytics Vidya, American Statistical Association, Research Gate and documentations of Python and Sklearn.

Vijay

CHAPTER-1

INTRODUCTION

1.1 BUSINESS PROBLEM

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.

1.2 BACKGROUND

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.

1.3 MOTIVATION

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.

CHAPTER-2

ANALYTICAL PROBLEM FRAMING

2.1 ANALYTICAL MODELING OF 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.

2.2 DATA SOURECE AND 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 [**Fig. 1**].The metadata table is provided below for better understanding of the data given [**Table 1**].

INDEPENDENT	ID	DENOTES THE UNIQUE IDS ASSOCIATED WITH EACH COMMENT GIVEN.
	COMMENT_TEXT	DENOTES COMMENTS EXTRACTED FROM VARIOUS SOCIAL MEDIA PLATFORMS.
DEPENDENT	MALIGNANT	IT DENOTES IF THE COMMENT ARE MALIGNANT OR NOT.
	HIGHLY_MALIGNANT	IT DENOTES COMMENTS THAT ARE HIGHLY MALIGNANT.
	RUDE	IT DENOTES COMMENTS THAT ARE VERY RUDE AND OFFENSIVE.
	THREAT	IT CONTAINS THE COMMENTS THAT POSE A TO SOMEONE.
	ABUSE	IT IS FOR COMMENTS THAT ARE ABUSIVE IN NATURE.
	LOATHE	IT DESCRIBES THE COMMENTS WHICH ARE HATEFUL AND LOATHING IN NATURE.

(Table 1: METADATA)

```
df.head(7)
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0

```
test.head(7)
```

	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.
5	0001ea8717f6de06	Thank you for understanding. I think very high...
6	00024115d4cbde0f	Please do not add nonsense to Wikipedia. Such ...

(Fig 1. TRAINING DATASET)

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() [**Fig 2**]. 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() **[Fig 3]** and the null values can be seen using df.isnull().sum() **[Fig 4]**. The null values can also be visualized with help of seaborn and matplotlib library **[Fig 5].** Visulization gives a better idea .

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

```
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']
```

(Fig 2. SHAPE AND COLU)MNS

```
#checking if there is any duplicated values in training dataset
print('Number of duplicated values:-',df.duplicated().sum())
```

```
Number of duplicated values:- 0
```

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

```
Number of duplicated values:- 0
```

(Fig 3. DUPLICATED VALUES)

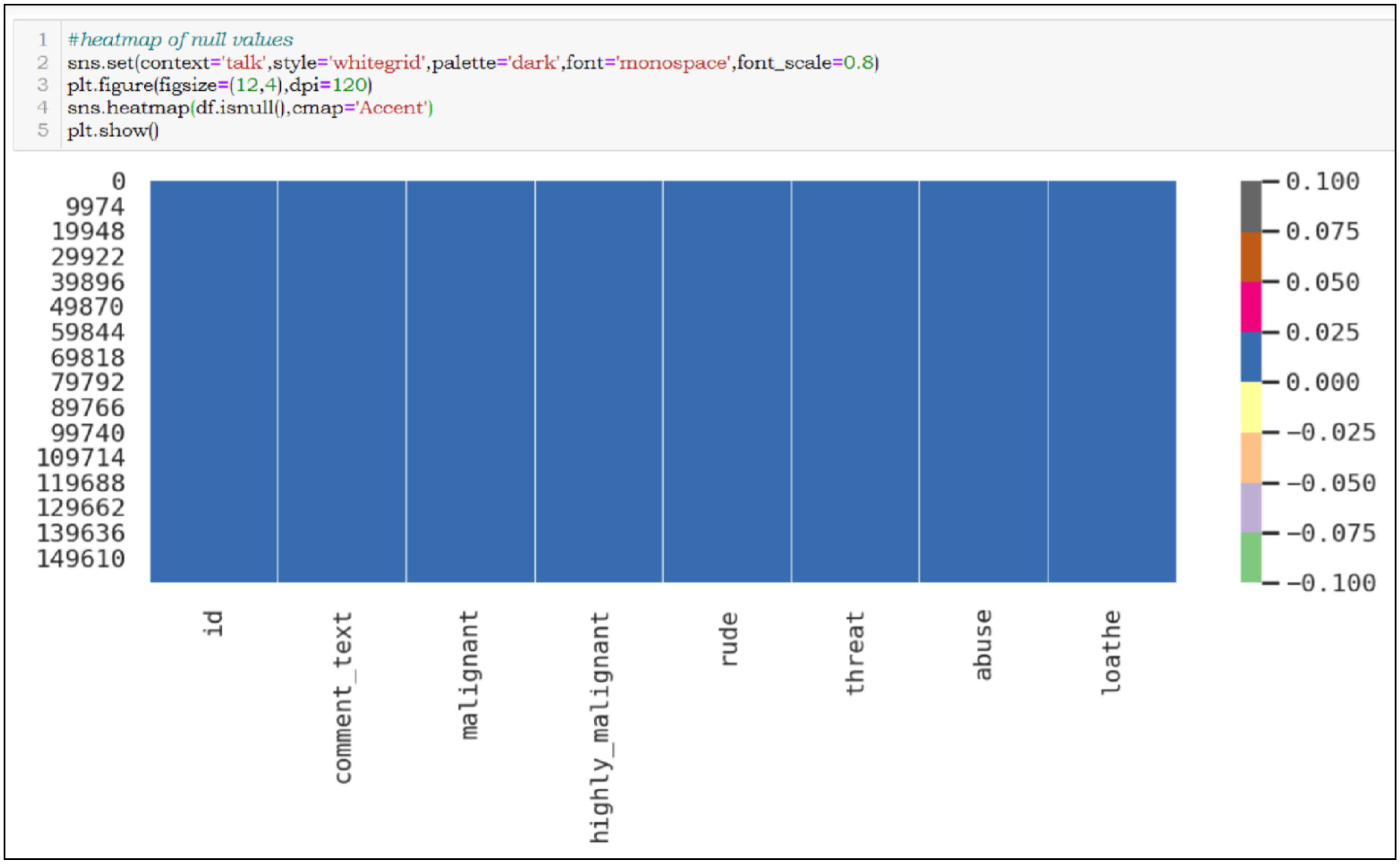
```
#checking for null values in training dataset
for i in df.columns:
    null=df[i].isnull().sum()
    if null>0:
        print('Number of Null Values At  '+"'"+ i+"'" ,'==  ',null )
    else:
        print('There are no null values in  '+"'"+ i+"'")
```

```
There are no null values in  'id'
There are no null values in  'comment_text'
There are no null values in  'malignant'
There are no null values in  'highly_malignant'
There are no null values in  'rude'
There are no null values in  'threat'
There are no null values in  'abuse'
There are no null values in  'loathe'
```

```
#checking for null values in test dataset
for i in test.columns:
    null=test[i].isnull().sum()
    if null>0:
        print('Number of Null Values At  '+"'"+ i+"'" ,'==  ',null )
    else:
        print('There are no null values in  '+"'"+ i+"'")
```

```
There are no null values in  'id'
There are no null values in  'comment_text'
```

(Fig 4. NULL VALUES)



(Fig 5. HEATMAP OF NULL VALUES)

2.3 DATA PREPROCESSING

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. [**Fig 6]**.

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 [**Fig 7**].

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 preprocessing and later a new column will be created to compare the length of strings before and after preprocessing

```
#adding a new column which represent a normal statement

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

```
cmt.head(7)
```

	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
5	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0	1
6	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0	0

The column named 'normal' represents the columns which are not falling under 'malignant','highly_malignant','rude','threat','abuse','loathe' categories

(Fig 7. CREATING NEW LABEL)

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

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

(Fig 6. DROPPING COLUMN)

```
#adding a column 'raw length' to the dataset which will show the length of characters in column 'comment_text'
cmt['raw length']= cmt.comment_text.str.len().astype('int64')
cmt.head(7)
```

	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
5	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0	1	65
6	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0	0	44

(Fig 8. RAW STRING LENGTH)

After that we can check the which label carries how many comments using df.value_count() method [**Fig 9].** It'll briefly show the count of numbers of 0 and 1 of all dependent columns / target columns.

```
#value counts of  label columns
values=['malignant','highly_malignant','rude','threat','abuse','loathe']
for i in values:
    vc=cmt[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
```

(Fig 9. VALUE COUNT)

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 **[Fig 10].** 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 **[Fig 11]**.

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

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

(Fig 10. COUNT OF LABELS WITH 1)

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

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

(Fig 11. COUNT OF COMMENTS FALLING UNDER MORE THE CATEGORIES)

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 **[Fig 12].**

```
1 cmt.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
```

```
1 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
```

(Fig 12. INFO OF TEST & TRAIN DATASET)

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 **[Fig 13].** 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, those tokenized words were joined to form a string. All these operations were compiled inside a function [**Fig 14**]. After the function was created a test run was done on a sample text to check the effectiveness of the function so created [**Fig 15-16**]. 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 [**Fig 17**].

```
stopwords=['i','me','my','myself','we','our','ours','ourselves','you',"you're","you've","you'll","you'd",'yo','nothin','from','bein','u','ok','yup','youve',
          'your','yours','yourself','yourselves','he','him','his','himself','she',"she's",'her','hers','ur','mlm','nbfc',"he's",'ip','ja',"there's",'tyme','yep',\
          'herself','it',"it's",'its','itself','they','them','their','theirs','themselves','what','which','lol','lool','fwiw','argh',"dont","i'll",'utc','too','y','u','r',\
          'doesnt','who','whom','this','that',"that'll",'these','how','these','those','am','is','are','was','were','oh','hay','thanks','ty','wc','ha','hi','d','re',\
          'll','there','someone','say','be','been','being','have','has','had','having','do','does','did','done','doing','a','an','the','even','aww','bye!','bye','e',\
          'f','and','but','if','or','because','as','until','while','of','at','by','for','with','about','against','between','into','through','during','before','after',\
          'above','below','to','from','up','down','in','out','on','off','over','under','again','further','then','once','here','there','when','where','why','how',\
          'all','any','both','each','few','more','most','other','some','such','no','nor','not','only','own','same','so','than','too','very','s','t','can','will','just','
          don',"don't",'should',"should've",'now','d','ll','m','o','re','ve','y','ain','aren',"aren't",'couldn',"couldn't",'didn',"didn't",'doesn',"doesn't",\
          'hadn', "hadn't",'hasn',"hasn't",'haven',"haven't",'isn',"isn't",'ma','mightn',"mightn't",'mustn',"mustn't",'needn',"needn't",'shan',"shan't",
          'shouldn',"shouldn't",'wasn',"wasn't",'weren',"weren't",'won',"won't",'wouldn',"wouldn't", 'looked', "what's",'although','upright','bit',\
          'right','state',"i've",'much','more',"there's","You've",'got',"i'd",'everything','true','yes','moreover','would','could','like','mr.','but',"i'm",'able','
          'back','get','still','ought','perhaps','without','away','onto','ive','let','must','see','went','saw','many','whats','id','let','day','never','yet','im','go',
          'thatll','theyre','came','youll','come','word','noone',"mrs.","now!","then?",'mr','ve','Â Â','january','days','february','march','april','may',\
          'june','july','august','september','october','november','december','everyone','hey','ok','okay','cant','bbq','let','thats','also','time','name',\
          'oh', 'said','asked','anyone','however','wow','daww']

print(len(stopwords))
```

```
322
```

(Fig 13. STOPWORDS)

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

def preprocess(text):
    processed=[]
    lower=text.lower().replace(r'\n',' ').replace(r'^.+@[^\.].*\.[a-z]{2,}$',' ').replace(r'^http://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',' ')
    #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'–',' ').replace('«', ' ').replace('»',' ')
    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()
```

(Fig 14. FUNCTION FOR PROCESSING DATA)

```
#TESTING THE FUNCTION CREATED ABOVE
sample=" As much as human rights and ethnic rights should be respected, spray painting every possible detail of unverifiable information\
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 the \
Arakanese people and state that have no concern of the Rohingya but include them for the sake of brotherly respect - this is pushing the\
line 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 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 desired\
on the main article on Rakhine state - albeit short and concise, not dump an entire list of names copied directly from some publication.\nW
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))
```

(Fig 15. SAMPLE TESTING)

```
Original Document:
 As much as human rights and ethnic rights should be respected, spray painting every possible detail of unverifiable informationon the Rohing
ya, 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 is pushing theline a bit far. Rohingyas should be treated fairly - I do not c
ontest that. But articles like this one - are pure self-pitying and clutters Wikipedia with absolutely useless information. I wonder when will som
ebody change the name of the article 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 concis
e, 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', 'every', 'possible', 'detail', 'of', 'unverifia
ble', 'informationon', 'the', 'Rohingya,', 'and', 'getting', 'around', 'the', 'verification', 'by', 'claiming', 'that', 'the', 'information', 'was', 'destroyed', 'b
y', 'an', 'interested', 'party', '-', 'are', 'not', 'valid', 'reasons', 'for', 'having', 'a', 'list', 'of', 'villages', 'where', 'a', 'certain', 'group', 'of', 'people', 'live.', 'T
here', 'is', 'already', 'a', 'lot', 'of', 'articles', 'on', 'the', 'Arakanese', 'people', 'and', 'state', 'that', 'have', 'no', 'concern', 'of', 'the', 'Rohingya', 'but', 'inc
lude', 'them', 'for', 'the', 'sake', 'of', 'brotherly', 'respect', '-', 'this', 'is', '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', 'us
eless', 'information.', 'I', 'wonder', 'when', 'will', 'somebody', 'change', 'the', 'name', 'of', 'the', 'article', 'on', 'Burma/Myanmar', 'on', 'wiki', 'to', 'Cou
ntry', 'where', 'the', 'Rohingya', 'are', 'Persecuted.\nRather,', 'a', 'brief', 'mention', 'of', 'where', 'the', 'Rohingyas', 'reside', 'should', 'be', 'placed', 'i
f', 'desiredon', 'the', 'main', 'article', 'on', 'Rakhine', 'state', '-', 'albeit', 'short', 'and', 'concise,', 'not', 'dump', 'an', 'entire', 'list', 'of', 'names', 'copie
d', 'directly', 'from', 'some', 'publication.\nWithall', 'due', 'respect,', 'this', 'article', 'should', '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 c
ertain group peopl live alreadi lot articl arakanes peopl concern rohingya includ sake brotherli respect push thelin far rohingya treat fairli conte
st articl one pure selfpiti clutter wikipedia absolut useless inform wonder somebodi chang articl wiki countri rohingya persecut rather brief me
ntion rohingya resid place desiredon main articl rakhin albeit short concis dump entir list name copi directli public withal due respect articl del
et
```

(Fig 16. TEST RESULTS)

```
%%time
clean = []

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

```
Wall time: 3min 5s
```

(Fig 17. FUNCTION IMPLEMENTATION)

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 [Fig 18]. 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 [Fig 19].

```
#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 })
cmt['comment']= processed

cmt['len of cleaned comment']=cmt['comment'].str.len().astype('int64')
cmt.head(5)
```

	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

(Fig 18. DATASET AFTER FUNCTI)ON

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

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

(Fig 19. REDUCTION DUE TO FUNCTI)ON

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’ **[Fig 20].**

```
%%time
comments = []

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

Wall time: 2min 45s

```
#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' : comments })
test['comment']= processed
test.head(5)
```

	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

(Fig 20. FUNCTION ON TEST DATASET)

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' [**Fig 21**]. After the splitting the tfidf vectorizer was initialized and X is fitted into it and converted into an array [**Fig 22**].

```
X=cmt.comment
y=cmt.iloc[:,1:-3]
```

```
X.head(4)
```

```
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
```

```
y.head(4)
```

	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

(Fig 21. X Y SPLITING)

```
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()
```

(Fig 22. TFIDF VECTORIZER)

2.4 HARDWARE & TOOL USED

In this project the below mentioned Hardware, IDE, Language, Packages were used.

HARDWARE	LAPTOP: ASUS TUF A17 OS: WIN 10 HOME BASIC PROCESSOR: AMD RYZEN 7 4800H RAM: 16GB VRAM: 6GB NVIDIA GTX 1660Ti

LANGUAGE	Python 3.8
IDE	JUPYTER NOTEBOOK 6.0.3
PACKAGES	PANDAS, NLTK, SKLEARN, MATPLOTLIB, SEABORN

(Table 2: HARDWARE & TOOLS)

CHAPTER-3

DEVELOPMENT AND EVALUTION

3.1 IDENTIFACTION OF POSSIBLE PROBLEM-SOLVING APPROACHES

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 [**Fig 23**].

```
x_train,x_test,y_train,y_test=tts(x,y,test_size=0.30,random_state=95)
```

```
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)
```

(Fig 23. TRAIN TEST SPLIT)

3.2 TESTING OF IDENTIFIED 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.

3.3 TESTING OF IDENTIFIED ALGORITHMS

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.

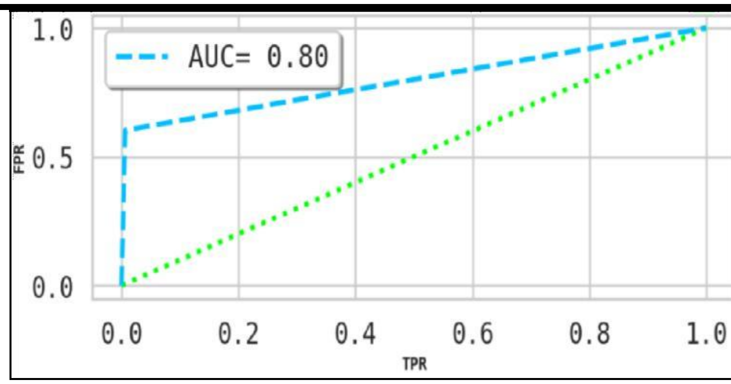
```python
LogReg_pipeline = Pipeline([('clf', OneVsRestClassifier(LogisticRegression(solver='sag'), n_jobs=1))])
F1=[]
ACCURACY = []
PRECESION = []
RECALL = []
RMSE = []
MSE = []
AUC=[]
TPR=[]
FPR=[]
CV_ACC=[]
LOG_LOSS=[]

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)


    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)
```

```python
    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')

    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'})
```

(Fig 24. LOGISTIC REGRESSION PIPELINE)

RESULT AT MALIGNANT LABEL

```
Processing malignant
--------------------------------------
ACCURACY SCORE: 0.9560703542780749

F1 score: 0.7239070500196929

CLASSIFICATION REPORT:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     43299
           1       0.91      0.60      0.72      4573

    accuracy                           0.96     47872
   macro avg       0.93      0.80      0.85     47872
weighted avg       0.95      0.96      0.95     47872


PRECISION:
 0.9057161629434954

RECALL:
 0.6028865077629565

MEAN SQUARED ERROR:
 0.04392964572192513 4

ROOT MEAN SQ. ERROR:
 0.20959400211343152

LOG_LOSS: 1.5172810044067442

AUC_ROC Score:
 0.7981290895820717

TPR: [0.         0.00662833 1.        ]
FPR: [0.         0.60288651 1.        ]
```



AUC CURVE AT MALIGNANT LABEL

HEATMAP OF CONFUSION MARIX AT MALGNANT

(Fig 25. LOGISTIC REGRESSION RESULT WITH MALIGNANT LABEL)

3.4 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().

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.723907	0.956070	0.905716	0.602887	0.209594	0.043930	0.798129	1.517281
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	0.622691	0.970129	0.814355	0.504058	0.172833	0.029871	0.749075	1.031723
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 USING LOGISTIC REGRESSION

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.718617	0.946294	0.719325	0.717909	0.231745	0.053706	0.844162	1.854951
1	highly_malignant	0.303371	0.989639	0.407547	0.241611	0.101789	0.010361	0.619150	0.357858
2	rude	0.745846	0.976354	0.855744	0.660963	0.153774	0.023646	0.827395	0.816722
3	threat	0.193939	0.997222	0.533333	0.118519	0.052709	0.002778	0.559113	0.095957
4	abuse	0.626212	0.968604	0.749405	0.537804	0.177190	0.031396	0.764279	1.084394
5	loathe	0.353896	0.991686	0.567708	0.257075	0.091180	0.008314	0.627663	0.287151
6	normal	0.971356	0.948362	0.968617	0.974110	0.227239	0.051638	0.846882	1.783526

RESULTS USING PASSIVE AGGRESSIVE CLASSIFIER

	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 USING MULTINOMIAL NB

	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.590774	0.884734	0.446976	0.870982	0.339508	0.115266	0.878584	3.981219
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 USING COMPLEMENT NB

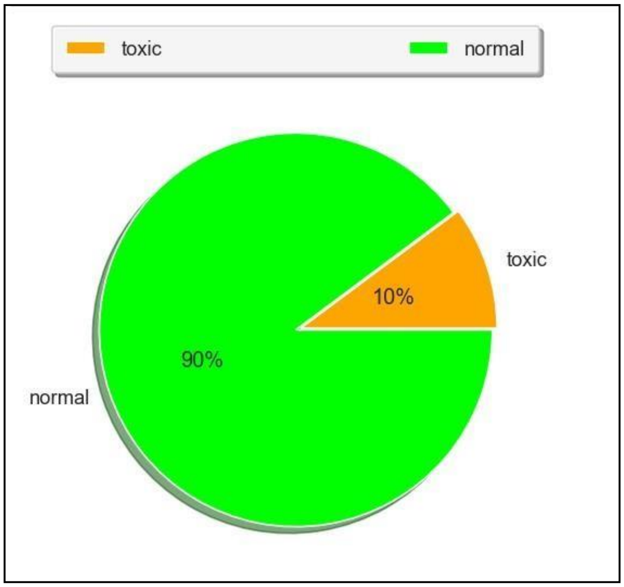
	LABELS	F1	Acuracy	Precision	Recall	RMSE	MSE	AUC	LOG_LOSS
0	malignant	0.590774	0.884734	0.446976	0.870982	0.339508	0.115266	0.878584	3.981219
1	highly_malignant	0.204190	0.937312	0.115824	0.861298	0.250376	0.062688	0.899663	2.165216
2	rude	0.494992	0.906271	0.345104	0.875050	0.306152	0.093729	0.891525	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 USING LINER SVC

(Fig 26. RESULTS)

3.5 VISUALIZATION

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.



(Fig 27. PIE PLOT OF NORMAL & HATE COMMENTS)

(Fig 28. WORD CLOUDS)

3.6 INTERPRETATION

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.

```
joblib.dump(LogReg_pipeline,'COMMENT_LOGI.pkl')
['COMMENT_LOGI.pkl']
```

(Fig 29. SAVING MODEL)

3.7 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.

```
test.head(7)
```

	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
5	Thank you for understanding. I think very high...	thank think highli revert discuss
6	Please do not add nonsense to Wikipedia. Such ...	pleas add nonsens wikipedia edit consid vandal...

```
X=test['comment']
X
```

```
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
```

(Fig 30. CLEANED TEST DAT)ASET

```
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)
```

```
test_x.shape
```

```
(153164, 10000)
```

```
result=model.predict(test_x)
```

(Fig 31. MODELLING OF TEST DAT)ASET

CHAPTER-4

CONCLUSION

4.1 KEY FINDINGS

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; ○ 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.

4.2 LEARNING OUTCOMES

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.

4.3 LIMITATION AND SCOPE OF 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 detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field.