MALIGNANT CLASSIFIER

Submitted by:

Harshitha K. S.

ACKNOWLEDGMENT

1. “Machine learning methods for toxic comment classification : a systematic review”. By Darko Androcec.

Nowadays users leave numerous comments on different social
Networks, news portals, and forums. Some of the comments are toxic or abusive. Due to numbers of comments, it is unfeasible to manually moderate them, so most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, They performed a systematic review of the state-of-the-art in toxic comment classification using machine learning methods. They extracted data from 31 selected primary relevant studies. First, They investigated when and where the papers were published and their maturity level. In the analysis of every primary study they investigated: data set used, evaluation metric, used machine learning methods, classes of toxicity, and comment language. They finished thier work with comprehensive list of gaps in current research and suggestions for future research themes related to online toxic comment classification problem.

2. “Identification and Classification of Toxic Comment Using Machine Learning Methods”. by Sathya Narayan Padhy and others.

This section focuses on different algorithms and the various stagesthat are involved for the proposed Toxic comment classification system such as ‘logistic regression’, ‘BR Method with Multinomial Naive Bayes classifiers’ and ‘BR Method with SVM classifier’ which helps us in classifying the comments and results in a decisive outcome
Thus, from the results we can conclude that taking hamming loss as a measure of identifying the optimal algorithm to classify toxic comments we can say that Binary Relevance method with Multinomial Naive Bayes is an efficient algorithm that serves our purpose and has a hamming loss of 3.6 as compared to the hamming loss of SVM with a score of 4.36.

3. “Multi Label Toxic Comment Classification using Machine Learning Algorithms”. by Abhishek Agarwal and Atul Tiwari

This paper has discussed three approaches to implement various machine learning algorithms like naïve bayes, ridge classifier, KNN and random forestand compared their Log-Loss, accuracy, and Hamming-Loss. After proper review, they concluded that there is no single best approach to solve the problem. Instead, each algorithm has got its own best approach for

optimal results. However, if they look at the time complexity of the algorithms, Random forest is not suitable for this data set as other algorithms give almost the same results in lesser time.

4. “Toxic comment classification using machine learning” by Promodya.

Comment classification models are available today for “flagging” the comments. However, determining whether or not a comment should be “flagged” is difficult and time-consuming. Another major problem is the lack of sufficient data for training the model, and there are some issues with the available datasets because those are annotated by the human raters and those annotations are dependent on their personal beliefs. Lack of multi-label comment classification model causes for issues of abusive behavior. This paper presents models for multi-label text classification for identifying the different level of toxicity within a comment. In this paper, we use Wikipedia comments which have been labeled by human raters for toxic behavior provided by Kaggle. Comments have been categorized into six categories as toxic, severe-toxic, obscene, threat, insult, and identityhate. The dataset contains 159572 comments. For data analyzing we use python seaborn library and python matploitlib library. It is understood that the dataset is highly skewed. Most of the comments do not belong to any of the six categories. Researchers used undersampling for majority class to correct the bias in the original dataset. We tested three models: a feed-forward neural network with Keras and word embedding, a Naive Bayes model with Scikit-Learn, and a LightGBM with 4-fold cross-validation. For the neural network, it took 3.5 hours to be trained on Nvidia GeForce 840M which is having 384 CUDA cores, Naive Bayes model with Scikit-Learn took 3 hours where LightGBM with k-fold took 4 hours. Researchersran 100 epochs from each model. At the end of 100 epoch, the neural network gave 0.9930 of validation accuracy and loss was just 0.2714, Naive Bayes model with Scikit-Learn gave 0.9556 validation accuracy and loss was 0.4121 where LightGBM with k-fold accuracy was 0.9000 and validation loss was 0.4263. The neural network gave the best accuracy at the end of the 100th epoch.

INTRODUCTION

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

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

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

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

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

Dataset

- The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples.
- All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.
- The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES.
- There are various comments which have multiple labels.
- The first attribute is a unique ID associated with each comment.

EDA

1.Understanding of the data

2.Checking the missing values

3.Checking for numerical columns

4.Checking for the distribution of numerical variables

5.Checking for Categorical variables

6. Types of categorical variables

7.Detecting outliers

Understanding of the data

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

Importing libraries and the dataset

```
import numpy as np
import pandas as pd
```

- Imported necessary libraries which is required for the project.

```
df=pd.read_csv("train.csv")
df.head()
```

	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

- Imported training dataset required for analysis.

Knowing Info. of the dataset.

```
# Shape of the dataset
df.shape
```

```
(159571, 8)
```

```
# info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 2   malignant         159571 non-null  int64 
 3   highly_malignant  159571 non-null  int64 
 4   rude              159571 non-null  int64 
 5   threat            159571 non-null  int64 
 6   abuse             159571 non-null  int64 
 7   loathe            159571 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

- df.shape() describes the total rows and columns available in the dataset.
- There are 159571 rows and 8 columns present in the dataset.
- df.info() provides information about the data types.
- The data info says that there are 6 integer columns and 2 categorical columns.

Checking the missing values

```
df.isnull().sum()
# There are no null values in the dataset
```

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

- ✓ df.isnull().sum() Shows the presence of missing values in the data set.
- ✓ There were no missing values present in the dataset.

Checking for numerical columns and their distribution.

```
df['malignant'].value_counts()
```

```
0    144277
1     15294
Name: malignant, dtype: int64
```

```
df['highly_malignant'].value_counts()
```

```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```

```
df['rude'].value_counts()
```

```
0    151122
1      8449
Name: rude, dtype: int64
```

```
df['threat'].value_counts()
```

```
0    159093
1       478
Name: threat, dtype: int64
```

```
df['loathe'].value_counts()
```

```
0    158166
1      1405
Name: loathe, dtype: int64
```

```
df['abuse'].value_counts()
```

```
0    151694
1      7877
Name: abuse, dtype: int64
```

- ❖ There are only 2 unique values in all the columns.
- ❖ Hence the columns are not the continuous data.
- ❖ The data is ordinal type of categorical data.
- ❖ Hence no outliers can be found in the dataset.

Data cleaning and Importing nlp libraries.

```
import string
```

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
ps=PorterStemmer()
```

```
dfcopy=df['comment_text'].iloc[:100]
dfcopy
```

```
0      Explanation\nWhy the edits made under my usern...
1      D'aww! He matches this background colour I'm s...
2      Hey man, I'm really not trying to edit war. It...
3      "\nMore\nI can't make any real suggestions on ...
4      You, sir, are my hero. Any chance you remember...
                             ...
95     "\n\nThanks. I can see that violating clearly ...
96     "\nHi\nThanks for our kind words. See you arou...
97     Collusion in poker \n\nThis is regarded as mos...
98     Thanks much - however, if it's been resolved, ...
99     You can do all you're doing right now but if y...
Name: comment_text, Length: 100, dtype: object
```

- The above are the necessary libraries which is required for text cleaning.

```
# We need to clean the dataset
cleaned_dataset=[]
for i in df['comment_text']:
    cleaned_text=i.split()
    cleaned_text=[i.lower() for i in cleaned_text]
    cleaned_text=[re.sub(r'[\n]','',i)for i in cleaned_text]
    cleaned_text=[re.sub(r'[^a-zA-Z]','',i)for i in cleaned_text]
    cleaned_text=[j for j in cleaned_text if j not in stopwords.words('english')]
    cleaned_text=[ps.stem(k) for k in cleaned_text]
    cleaned_text=(' ').join(cleaned_text)
    cleaned_dataset.append(cleaned_text)
```

- The above is the code to extract only text from the dataset by eluminating all the extra special characters, numerical. By creating the empty list and appending the cleaned data into it.

The column which is resulted after cleaning.

```
cleaned_dataset
```

```
['explan edit made usernam hardcor metallica fan revert werent vandal closur ga vote new york doll fac pleas dont remov templ
at talk page sinc im retir',
 'daww match background colour im seemingli stuck thank talk  januari  utc',
 'hey man im realli tri edit war guy constantli remov relev inform talk edit instead talk page seem care format actual info',
 ' cant make real suggest improv  wonder section statist later subsect type accid think refer may need tidi exact format ie d
ate format etc later noon els first  prefer format style refer want pleas let know appear backlog articl review guess may del
ay review turn list relev form eg wikipediagoodarticlenominationstransport ',
 'sir hero chanc rememb page that',
 ' congratul well use tool well  talk ',
 'cocksuck piss around work',
 'vandal matt shirvington articl revert pleas dont ban',
 'sorri word nonsens offens anyway im intend write anyth articlewow would jump vandal im mere request encycloped one use scho
ol refer select breed page almost stub point anim breed short messi articl give info must someon around expertis eugen ',
 'align subject contrari dulithgow',
 ' fair use rational imagewonjujpg thank upload imagewonjujpg notic imag page specifi imag use fair use explan rational use w
ikipedia articl constitut fair use addit boilerpl fair use templat must also write imag descript page specif explan rational
use imag articl consist fair use pleas go imag descript page edit includ fair use rational upload fair use media consid check
specifi fair use rational page find list imag page edit click contribut link locat top wikipedia page log select imag dropdow
n box note fair use imag upload  may  lack explan delet one week upload describ criteria speedi delet question pleas ask medi
a copyright question page thank talk  contrib  unspecifi sourc imagewonjujpg thank upload imagewonjujpg notic file descript
```

```
len(cleaned_dataset)
```

```
159571
```

The dataset view

```
df.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	explan edit made usernam hardcor metallica fan...
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	daww match background colour im seemingli stuc...
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	hey man im realli tri edit war guy constantli ...
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	cant make real suggest improv wonder section...
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	sir hero chanc rememb page that

Extracting dataset with new column for further analysis.

```
dfnew=df.iloc[:,2:]
dfnew.head()
```

	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
0	0	0	0	0	0	0	explan edit made usernam hardcor metallica fan...
1	0	0	0	0	0	0	daww match background colour im seemingli stuc...
2	0	0	0	0	0	0	hey man im realli tri edit war guy constantli ...
3	0	0	0	0	0	0	cant make real suggest improv wonder section...
4	0	0	0	0	0	0	sir hero chanc rememb page that

Columns with multi-labels.

```
dfnew[(dfnew['malignant']==1) & (dfnew['highly_malignant']==1)]
```

	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
6	1	1	1	0	1	0	cocksuck piss around work
55	1	1	1	0	1	0	stupid peac shit stop delet stuff asshol go di...
181	1	1	1	0	1	0	stupid fuck mother cunt stink
442	1	1	1	0	1	0	hi im fuck bitch
579	1	1	1	0	1	0	motherfuck piec crap fuckhead block us
...	...	...	...	...	...	...	...
159096	1	1	1	0	1	0	filthi stink crow back dirti crow better delet...

- Text which is malignant and highly malignant.
- 1595 rows has both tags.

```
dfnew[(dfnew['malignant']==1) & (dfnew['highly_malignant']==1) & (dfnew['rude']==1) ]
```

	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
6	1	1	1	0	1	0	cocksuck piss around work
55	1	1	1	0	1	0	stupid peac shit stop delet stuff asshol go di...
181	1	1	1	0	1	0	stupid fuck mother cunt stink
442	1	1	1	0	1	0	hi im fuck bitch
579	1	1	1	0	1	0	motherfuck piec crap fuckhead block us
...	...	...	...	...	...	...	...
159096	1	1	1	0	1	0	filthi stink crow back dirti crow better delet...
159099	1	1	1	0	1	0	fuck pathet moron cc jimbo wale rootmyass
159281	1	1	1	0	1	1	lol your gay never know good feel fuck woman ass

- Text which is malignant , highly malignant and rude.
- 1517 rows has all the tags.

```
dfnew[(dfnew['malignant']==1) & (dfnew['highly_malignant']==1) & (dfnew['rude']==1) & (dfnew['abuse']==1)]
```

	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
6	1	1	1	0	1	0	cocksuck piss around work
55	1	1	1	0	1	0	stupid peac shit stop delet stuff asshol go di...
181	1	1	1	0	1	0	stupid fuck mother cunt stink
442	1	1	1	0	1	0	hi im fuck bitch
579	1	1	1	0	1	0	motherfuck piec crap fuckhead block us
...	...	...	...	...	...	...	...
159096	1	1	1	0	1	0	filthi stink crow back dirti crow better delet...

- Text which is malignant , highly malignant , rude and abuse.

- 1349 rows has all the tags.

```
dfnew[(dfnew['malignant']==1) & (dfnew['highly_malignant']==1) & (dfnew['rude']==1) & (dfnew['abuse']==1)& (dfnew['threat']==1)]
```

	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
1017	1	1	1	1	1	1	wouldnt first time bitch fuck ill find live so...
1312	1	1	1	1	1	1	shut fat poop kick ass
2894	1	1	1	1	1	0	go die son bich
4712	1	1	1	1	1	0	go fuck bastard yyou life go fuck bastard yank...
5569	1	1	1	1	1	0	dare send abus fuck cockmast dare warn mr jeff...
...	...	...	...	...	...	...	...
155722	1	1	1	1	1	0	come ducttak rape till die fuckhead
157428	1	1	1	1	1	1	ok bitch swear god fuckin beat face itz n popp...

- Text which is malignant , highly malignant , rude , abuse and threat.
- 95 rows has all the tags.

```
dfnew[(dfnew['malignant']==1) & (dfnew['highly_malignant']==1) & (dfnew['rude']==1) & (dfnew['abuse']==1) &
      (dfnew['threat']==1)& (dfnew['loathe']==1)]
```

	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
1017	1	1	1	1	1	1	wouldnt first time bitch fuck ill find live so...
1312	1	1	1	1	1	1	shut fat poop kick ass
7299	1	1	1	1	1	1	your stupid cunt fuck dumb ars mum hairi cunt ...
13648	1	1	1	1	1	1	bitch littl bitch fuckin spent hour big sam r...
13964	1	1	1	1	1	1	go murder zimzalabim st evil homosexu jew
22158	1	1	1	1	1	1	fuck fuck nigger bag shit hope die horribl fir...
29968	1	1	1	1	1	1	u motherfukkin bitch want rape smelli whore st...
32098	1	1	1	1	1	1	fuck asyriac nation qamishli belong arminian f...
33951	1	1	1	1	1	1	go fuck bitch hate sould mother fucker hell th...

- Text which is malignant , highly malignant , rude , abuse , threat and loathe.
- 31 rows has all the tags.

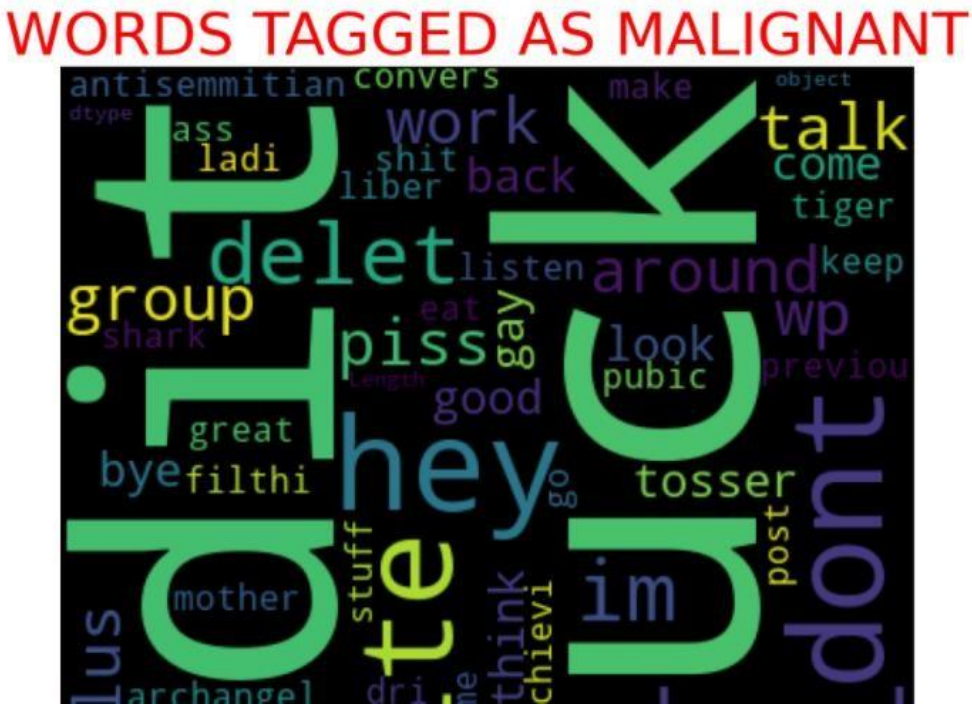
```
dfnewnormal=dfnew[(dfnew['abuse']==0) & (dfnew['highly_malignant']==0) & (dfnew['loathe']==0) & (dfnew['malignant']==0)
                  & (dfnew['rude']==0) & (dfnew['threat']==0)]
dfnewnormal
```

	malignant	highly_malignant	rude	threat	abuse	loathe	cleaned_text
0	0	0	0	0	0	0	explan edit made usernam hardcor metallica fan...
1	0	0	0	0	0	0	daww match background colour im seemingli stuc...
2	0	0	0	0	0	0	hey man im realli tri edit war guy constantli ...
3	0	0	0	0	0	0	cant make real suggest improv wonder section...
4	0	0	0	0	0	0	sir hero chanc rememb page that
...	...	...	...	...	...	...	...
159566	0	0	0	0	0	0	second time ask view complet contradict covera...

- Normal rows with negative comments.
- Totally 143346 rows.

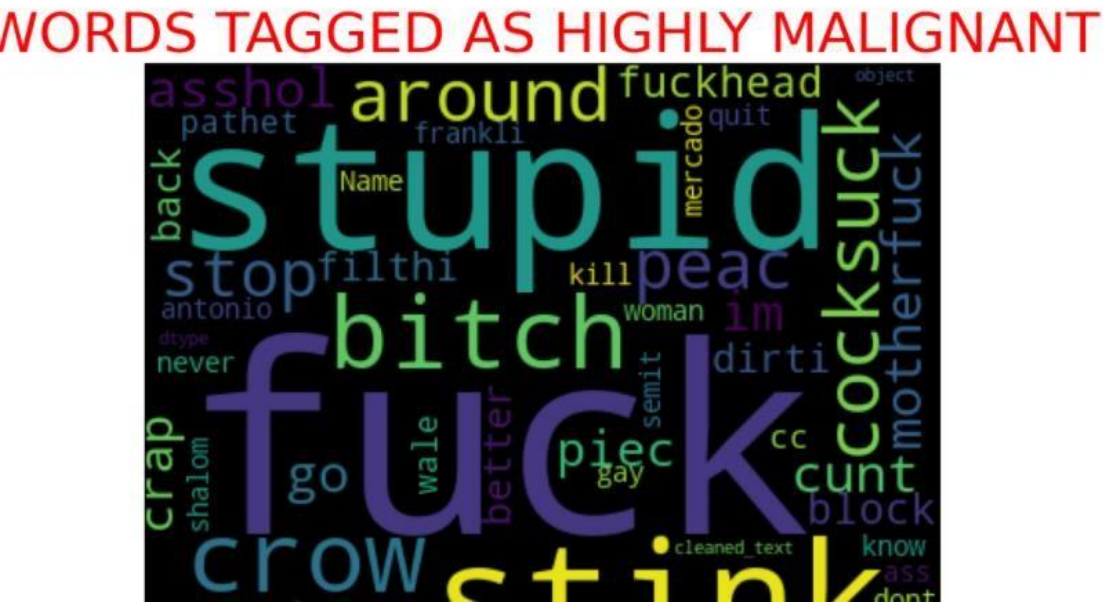
Wordcloud for each label separately

```
wordcloud=WordCloud(height=400,width=400,max_words=400).generate(str(df_malignant['cleaned_text']))
plt.figure(figsize=(10,10))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title(label='WORDS TAGGED AS MALIGNANT',fontdict={'fontsize':50, 'fontweight':50, 'color':'red'})
plt.show()
```



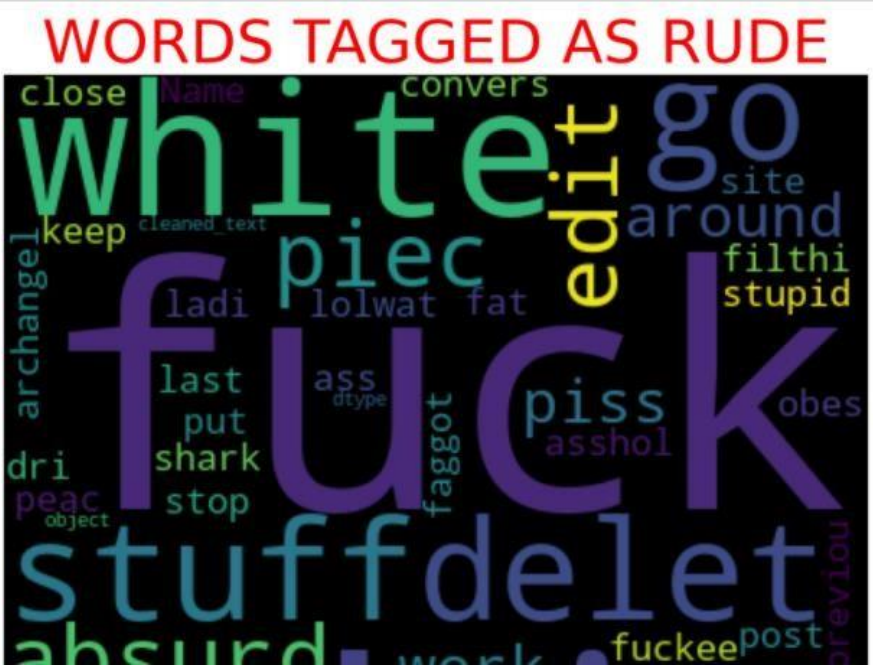
Wordcloud for Malignant.

```
wordcloud=WordCloud(height=400,width=400,max_words=400).generate(str(df_highlymalignant['cleaned_text']))
plt.figure(figsize=(10,10))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title(label='WORDS TAGGED AS HIGHLY MALIGNANT',fontdict={'fontsize':50, 'fontweight':50, 'color':'red'})
plt.show()
```



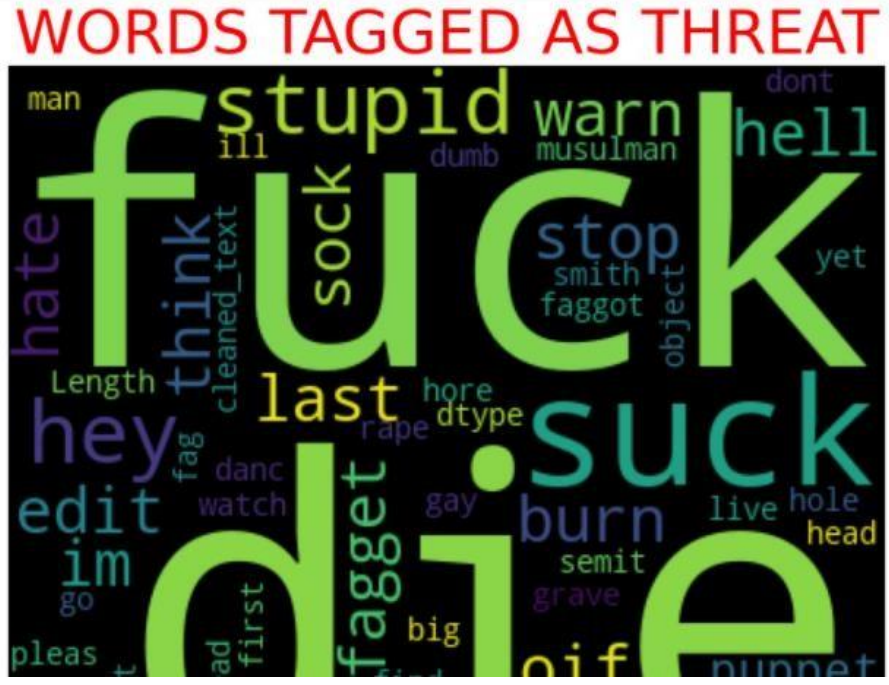
Wordcloud for Highly Malignant.

```
wordcloud=WordCloud(height=400,width=400,max_words=400).generate(str(df_rude['cleaned_text']))
plt.figure(figsize=(10,10))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title(label='WORDS TAGGED AS RUDE',fontdict={'fontsize':50, 'fontweight':50, 'color':'red'})
plt.show()
```



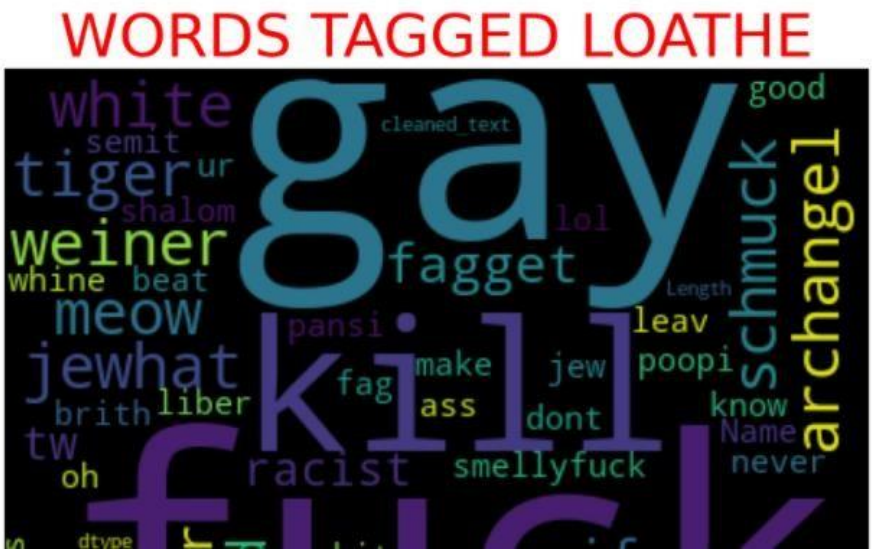
Wordcloud for Rude.

```
wordcloud=WordCloud(height=400,width=400,max_words=400).generate(str(df_threat['cleaned_text']))
plt.figure(figsize=(10,10))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title(label='WORDS TAGGED AS THREAT',fontdict={'fontsize':50, 'fontweight':50, 'color':'red'})
plt.show()
```



Wordcloud for Threat.

```
wordcloud=WordCloud(height=400,width=400,max_words=400).generate(str(df_loathe['cleaned_text']))
plt.figure(figsize=(10,10))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title(label='WORDS TAGGED LOATHE',fontdict={'fontsize':50, 'fontweight':50, 'color':'red'})
plt.show()
```



wordcloud for loathe.

MODEL DEVELOPMENT AND EVALUATION.

Vectorization:

```
# Lets apply tfidf to the text data for converting it into number vectors
from sklearn.feature_extraction.text import TfidfVectorizer
vect=TfidfVectorizer()
```

```
X_dfnew=vect.fit_transform(dfnew['cleaned_text'])
```

```
Y=dfnew.iloc[:,0:6]
Y.head()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0

- After cleaning the text data, the text data was converted to numerical vector for the further analysis.
- The dependent and independent data was separately labelled.

```
from sklearn.model_selection import train_test_split
train_x,test_x,train_y,test_y=train_test_split(X_dfnew,Y,random_state=0,test_size=0.20)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.multioutput import MultiOutputClassifier
```

```
forest=RandomForestClassifier(random_state=1)
multi_target_forest=MultiOutputClassifier(forest)
```

```
multi_target_forest.fit(train_x,train_y)
pred=multi_target_forest.predict(test_x)
```

```
pred=multi_target_forest.predict(test_x)
```

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
pred=OneVsRestClassifier(LinearSVC(random_state=0)).fit(train_x,train_y).predict(test_x)
```

```
classifier=OneVsRestClassifier(LinearSVC(random_state=0))
classifier.fit(train_x,train_y)
classifier.predict(test_x)
```

```
array([[0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       ...,
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0]])
```

- The data was splitted into train and test for training and valiadation.
- The Random forest and SVC were the two models used for modelling.
- The model which was performed better was later used for prediction.

```
from sklearn.metrics import hamming_loss, accuracy_score
print("Accuracy Score", accuracy_score(pred,test_y))
print("Hamming Loss", hamming_loss(pred,test_y))
```

```
Accuracy Score 0.9132069559768135
Hamming Loss 0.021677372186537158
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(pred,test_y))
```

```
              precision    recall  f1-score   support

           0       0.52      0.93      0.67      1748
           1       0.05      0.50      0.10        36
           2       0.60      0.90      0.72      1123
           3       0.04      0.57      0.08         7
           4       0.44      0.81      0.57       870
           5       0.09      0.90      0.16        30

   micro avg       0.48      0.89      0.62      3814
   macro avg       0.29      0.77      0.38      3814
weighted avg       0.52      0.89      0.65      3814
 samples avg       0.04      0.05      0.04      3814
```

Model evaluation


```
from sklearn.metrics import hamming_loss, accuracy_score
print("Accuracy Score", accuracy_score(pred,test_y))
print("Hamming Loss", hamming_loss(pred,test_y))
```

```
Accuracy Score 0.9132069559768135
Hamming Loss 0.021677372186537158
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(pred,test_y))
```

```
              precision    recall  f1-score   support

           0       0.52      0.93      0.67      1748
           1       0.05      0.50      0.10        36
           2       0.60      0.90      0.72      1123
           3       0.04      0.57      0.08         7
           4       0.44      0.81      0.57       870
           5       0.09      0.90      0.16        30

   micro avg       0.48      0.89      0.62      3814
   macro avg       0.29      0.77      0.38      3814
weighted avg       0.52      0.89      0.65      3814
 samples avg       0.04      0.05      0.04      3814
```


- The accuracy score for Random forest model was found to be 91.3 per cent and the loss was 2 per cent.

```
print("Accuracy Score", accuracy_score(pred,test_y))
print("Hamming Loss", hamming_loss(pred,test_y))
```

```
Accuracy Score 0.9176876077079743
Hamming Loss 0.018789492923912476
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(pred,test_y))
```

```
              precision    recall  f1-score   support

           0       0.68      0.88      0.77      2388
           1       0.23      0.51      0.32       152
           2       0.70      0.88      0.78      1348
           3       0.14      0.65      0.23        20
           4       0.55      0.77      0.64      1137
           5       0.28      0.65      0.39       126

   micro avg       0.61      0.84      0.71      5171
   macro avg       0.43      0.72      0.52      5171
weighted avg       0.63      0.84      0.72      5171
 samples avg       0.06      0.06      0.06      5171
```

- The accuracy score for Random forest model was found to be 91.7 per cent and the loss was 1.8 per cent.
- From both the model we can conclude that the SVC model is performing better when compare to Random forest model. Hence can be used for predicting the unseen data.

Test data for prediction.

```
dftest=pd.read_csv('test.csv')
```

```
cleaned_testdataset=[]
for i in dftest['comment_text']:
    cleaned_text=i.split()
    cleaned_text=[i.lower() for i in cleaned_text]
    cleaned_text=[re.sub(r'[\n]','',i)for i in cleaned_text]
    cleaned_text=[re.sub(r'[^a-zA-Z]','',i)for i in cleaned_text]
    cleaned_text=[j for j in cleaned_text if j not in stopwords.words('english')]
    cleaned_text=[ps.stem(k) for k in cleaned_text]
    cleaned_text=(' ').join(cleaned_text)
    cleaned_testdataset.append(cleaned_text)
```

```
dftest['cleaned_comment']=cleaned_testdataset
```

```
X_test=vect.transform(dftest['cleaned_comment'])
```

- Imported test data for prediction.
- The same EDA was carried out for the test data which was followed for train data.

```
predictions=classifier.predict(X_test)
predictiondf=pd.DataFrame(predictions)
```

```
predictiondf.to_csv("Malignant_Comments_predictions", index=False,header=False)
```

Predictions was converted into data frame and was stored in CSV file.

CONCLUSION

- From both the model we can conclude that the SVC model is performing better when compare to Random forest model. Hence can be used for predicting the unseen data.
- In further research, we can use algorithm adaptation methods that transform the algorithms to perform multi-label classification directly. Furthermore, we can also experiment with more complex deep learning algorithms like CNN (convolutional neural network), MLP (multilayer perceptron), and RNN (Recurrent neural networks) in the near future as we believe our approach could reach the top performance when combined with deep learning models.