



MALALIGNANT COMMENTS CLASSIFICATION.

Submitted by: Pratik Kumar

INTRODUCTION

Business Problem Framing

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

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

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

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

Our goal is to build a prototype of online hate and abuse comment classifier .

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

I start analysis on this project in importing the data set and simple play around with the data and identifying the characteristics of each column.

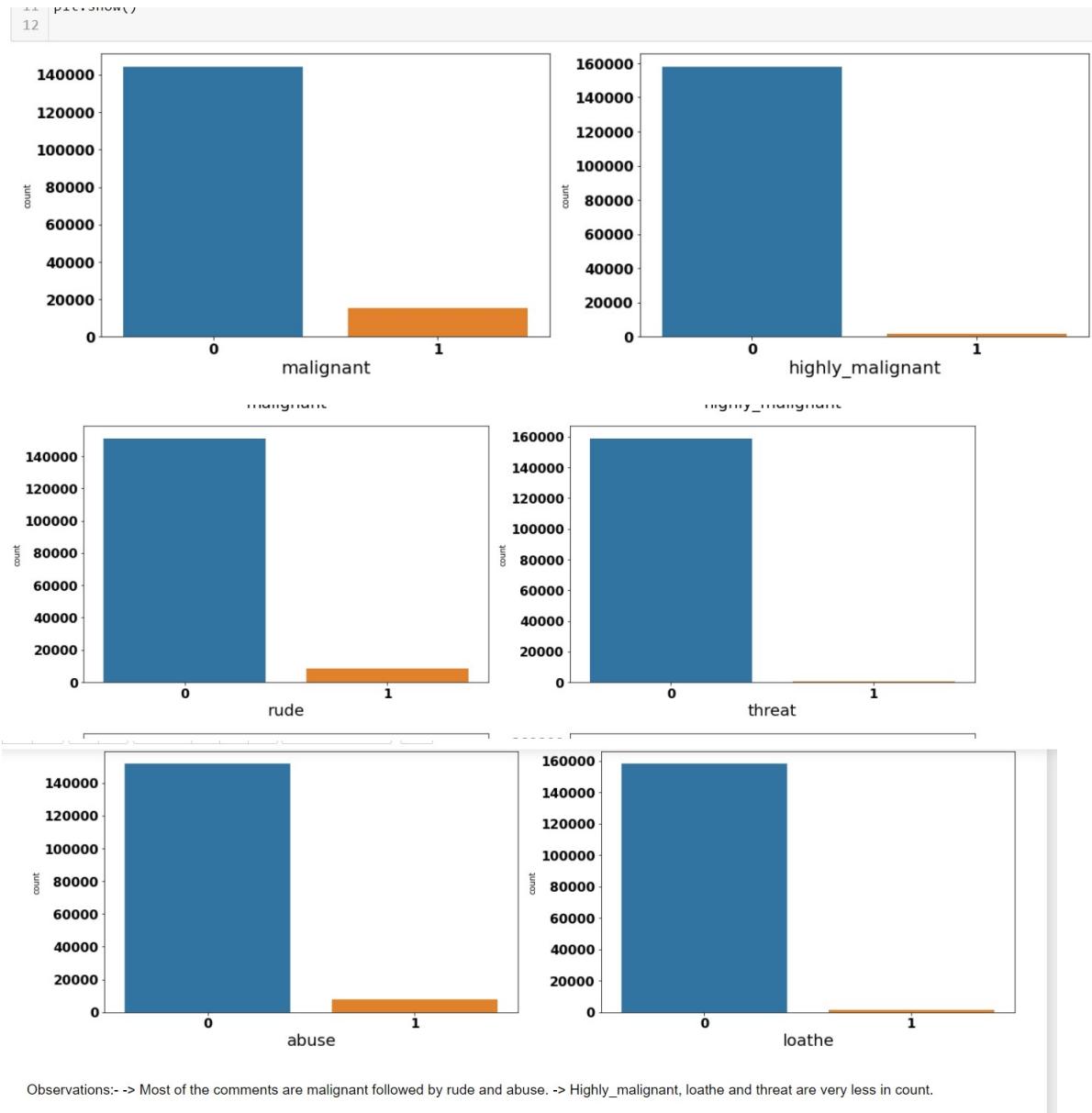
```
In [6]: 1 ## Data type:  
2 df.dtypes  
  
Out[6]: id          object  
comment_text    object  
malignant      int64  
highly_malignant int64  
rude           int64  
threat          int64  
abuse           int64  
loathe          int64  
dtype: object
```

Apart from 'id' and 'comment_text', rest columns datatype is int.

In the first stroke of analysis I understood that there are 8 columns in total in which 6 are numerical column with binary data 0's and 1's and 'id' data which has all unique values "comment text" have string values.

Since 'id' have all unique values, it won't be helpful in analysis so I have dropped 'id' column.

Post dropping 'id' I tried to understand data from 'malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe' columns by plotting them in count plot.



Key observation:

we can see that there only minimum number of columns in 'malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe' and remaining all in 0.

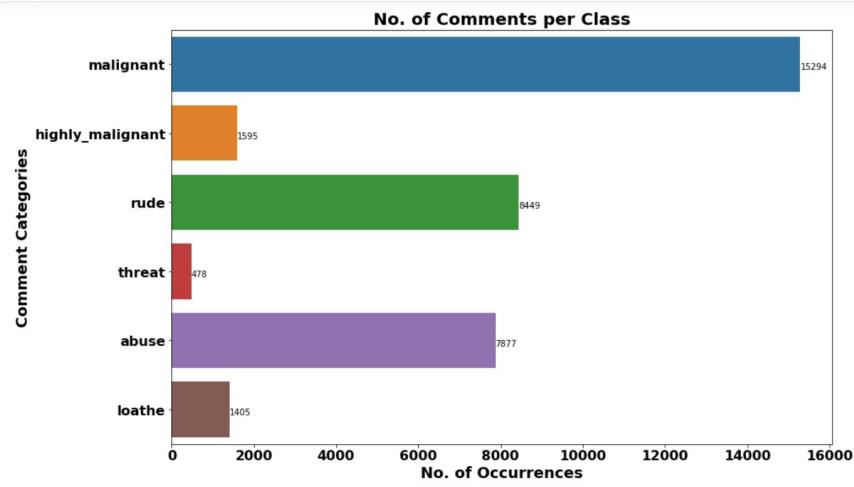
```
In [12]: 1 #Checking the percentage of the comments
2 non_malignant = df[(df['malignant']!=1) & (df['highly_malignant']!=1) & (df['rude']!=1) &
3                               (df['threat']!=1) & (df['abuse']!=1) & (df['loathe']!=1)]
4 percent=len(non_malignant)/len(df)*100
5 print('Percentage of good/neutral comments = ',percent)
6 print('Percentage of negative comments = ', (100-percent))
```

Percentage of good/neutral comments = 89.83211235124176
 Percentage of negative comments = 10.167887648758239

Around 90% of comments are Good/Neutral in nature. Around 10% of comments are negative comments

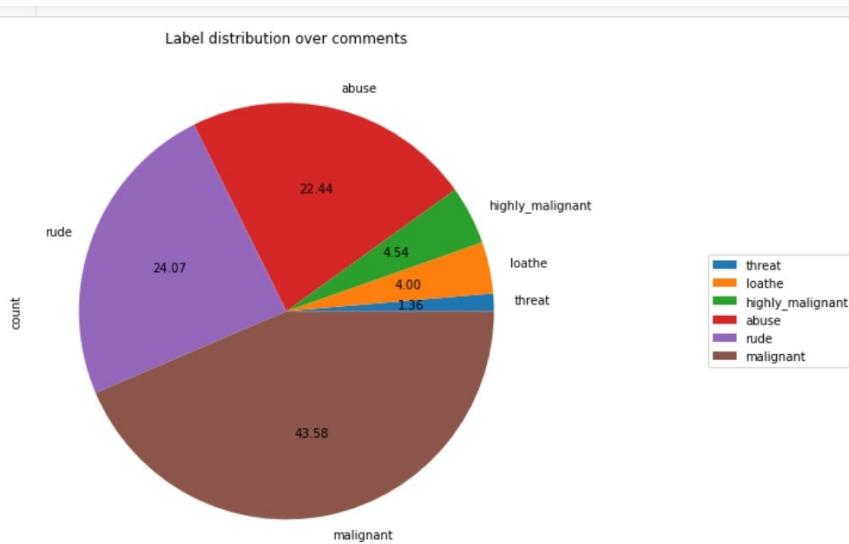
```
In [13]: 1 #Create a new subset of the data by only taking the 2nd column onwards (Comments and categories)
```

So, only 10% of the data is getting classified as malignant comments
data is unbalanced.



Key observation:

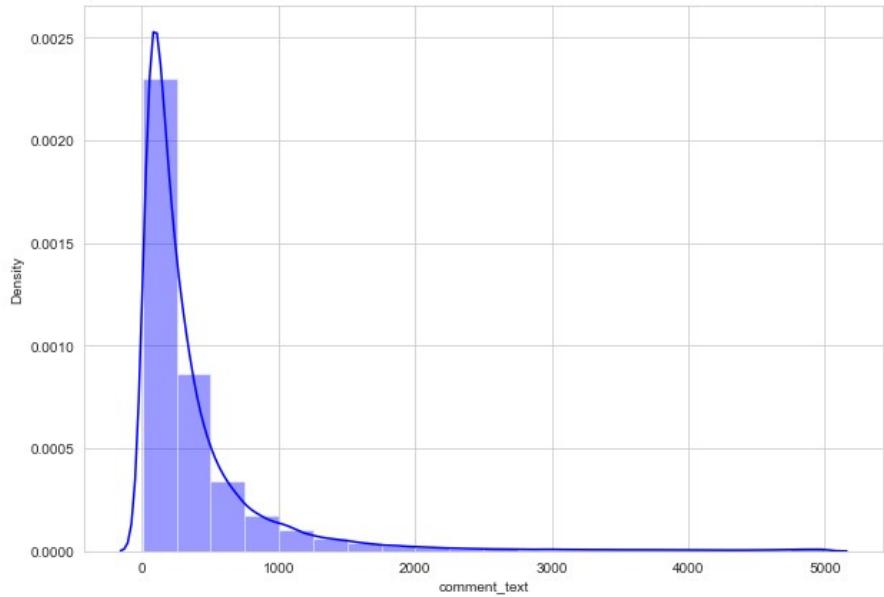
1. We can see malignant and rude are high categorised sentences in the data.



Observation : Out of total negative comments around 43.58% are malignant in nature followed by 24.07% are rude comments.

Key observation:

1. As we see above malignant, and rude sentence are high classified and threat, loathe are least classified.



Key observation:

1. We can see that few sentences are really long but most of the sentence are small.



Key Observations:

1. We can see more corelations in the variables, Abuse have more corelation with malignant and rude.
2. Rude has more positive corelation with malignant 3. we don't have any negative corelations in the data.

Data Sources and their formats

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

1. Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
2. Highly Malignant: It denotes comments that are highly malignant and hurtful.
3. Rude: It denotes comments that are very rude and offensive.
4. Threat: It contains indication of the comments that are giving any threat to someone.
5. Abuse: It is for comments that are abusive in nature.
6. Loathe: It describes the comments which are hateful and loathing in nature.
7. ID: It includes unique Ids associated with each comment text given.
8. 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.

Data Pre-processing Done

```
1 #Importing Required libraries
2 import nltk
3 import re
4 import string
5 from nltk.corpus import stopwords
6 from wordcloud import WordCloud
7 from nltk.tokenize import word_tokenize
8 from nltk.stem import WordNetLemmatizer
9 from sklearn.feature_extraction.text import TfidfVectorizer
10
```

I imported all the required libraries for cleansing the data.

After importing all the required libraries, I have defined Stopwords and lemmatize to a variable.

```
1 #Defining the stop words
2 stop_words = stopwords.words('english')
3 |
4 #Defining the lemmatizer
5 lemmatizer = WordNetLemmatizer()
6
```

Post which I have defined a function for cleaning the data.

```
def format_comments(text):
    #convert to lower case
    lowered_text = text.lower()

    #Replacing email addresses with 'emailaddress'
    text = re.sub(r'^[a-z0-9]+[\._]?[ a-z0-9]+@[ ]\w{2,3}\$', 'emailaddress', lowered_text)

    #Replace URLs with 'webaddress'
    text = re.sub(r'http\S+', 'webaddress', text)

    #Removing numbers
    text = re.sub(r'\d+', " ", text)

    #Removing the HTML tags
    text = re.sub(r"<.*?>", " ", text)

    #Removing Punctuations
    text = re.sub(r'[^w\s]', ' ', text)
    text = re.sub(r'\_',' ',text)

    #Removing all the non-ascii characters
    clean_words = re.sub(r'[^\x00-\x7f]',r'', text)

    #Removing the unwanted white spaces
    text = " ".join(text.split())

    #Splitting data into words
    tokenized_text = word_tokenize(text)

    #Removing remaining tokens that are not alphabetic, Removing stop words and Lemmatizing the text
    removed_stop_text = [lemmatizer.lemmatize(word) for word in tokenized_text if word not in stop_words if word.isalpha()]
```

Post on creating a function I have passed my data into the same to clean it.

```

1 # Apply above function for the column comment_text in training dataset to replace original with cleaned text
2 df['comment_text'] = df['comment_text'].apply(format_comments)
3 df['comment_text'].head()
4

0 explanation edits made username hardcore metal...
1 aww match background colour seemingly stuck th...
2 hey man really trying edit war guy constantly ...
3 make real suggestion improvement wondered sect...
4 sir hero chance remember page
Name: comment_text, dtype: object

```

```

In [27]: 1 print("Total Words Removed:", (df.length_before_cleaning.sum() - (df.length_after_cleaning.sum())))
2

```

Total Words Removed: 24416722

The total amount of data that is cleansed from the original data is 24416722. Now the data is cleansed and ready for training but before which I converted the data into vectors for the machine learning models to understand the data, so I imported TFIDF vectorizer and I have made the max feature as 2000.

1. Vectorizer & Splitting Train dataset

```

1 # Converting the features into number vectors
2 tf_vec = TfidfVectorizer(max_features = 2000, stop_words='english')
3

1 # Let's Separate the input and output variables represented by X and y respectively in train data and convert them
2 X = tf_vec.fit_transform(df['comment_text']).toarray()
3

1 df.head()

comment_text malignant highly_malignant rude threat abuse loathe label length_before_cleaning length_after_cleaning
0 explanation edits made username hardcore metal... 0 0 0 0 0 0 0 264 156
1 aww match background colour seemingly stuck th... 0 0 0 0 0 0 0 112 67
2 hey man really trying edit war guy constantly ... 0 0 0 0 0 0 0 233 141
3 make real suggestion improvement wondered sect... 0 0 0 0 0 0 0 622 364
4 sir hero chance remember page 0 0 0 0 0 0 0 67 29

1 output_labels= df.columns[1:7]

1 # output variables
2 from scipy.sparse import csr_matrix
3 Y = csr_matrix(df[output_labels]).toarray()

```

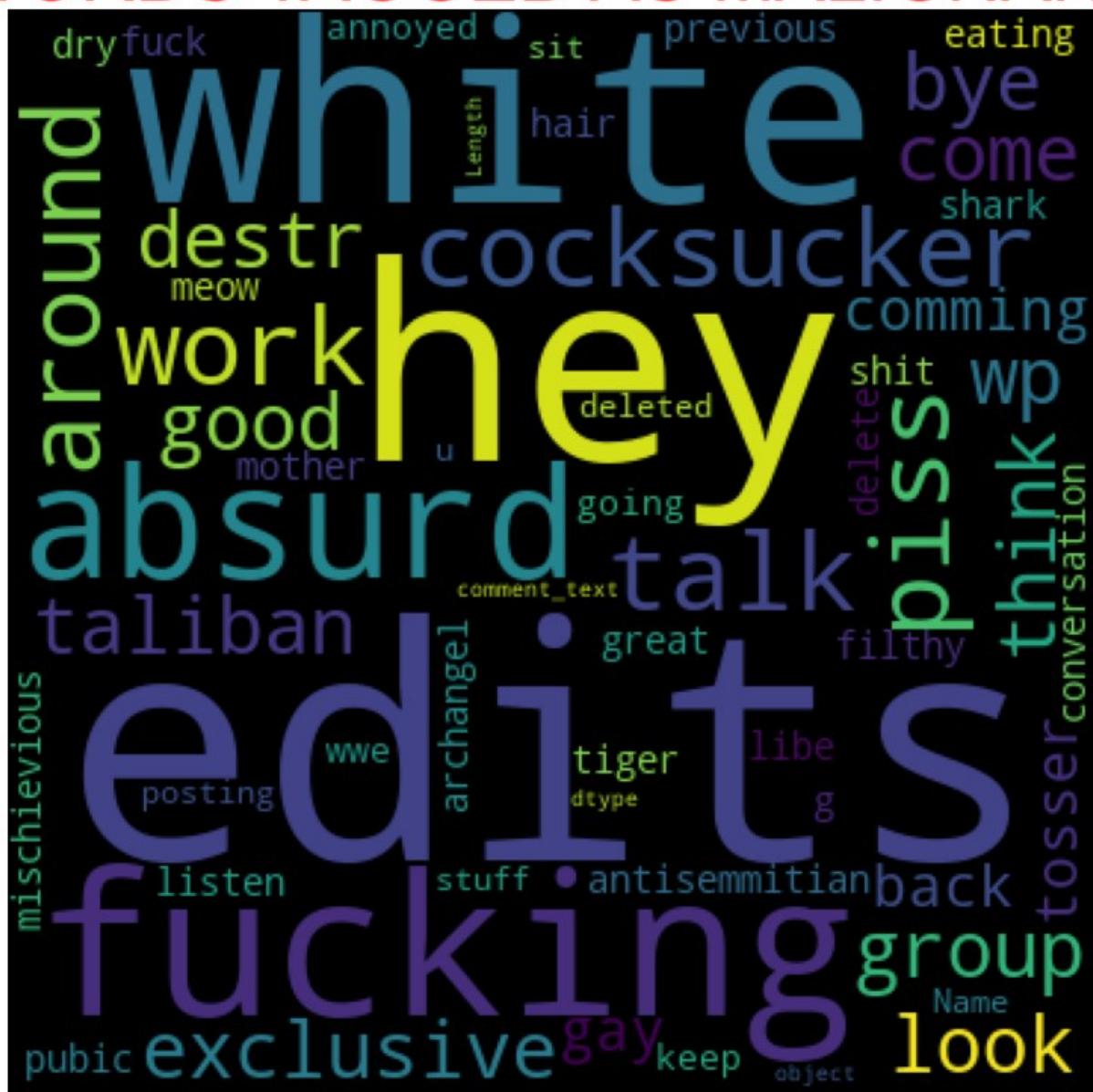
And I have split the data into two parts X and y and made them ready for training. I have created a new feature name label and summed all

the other numerical column and changed the output as binary i.e., if the sentence is categorised as malignant it will be 1 or else it will be 0.

Data Inputs- Logic- Output Relationships

I have analysed the input output logic with word cloud and I have word clouded the sentences that are classified as foul language in every category.

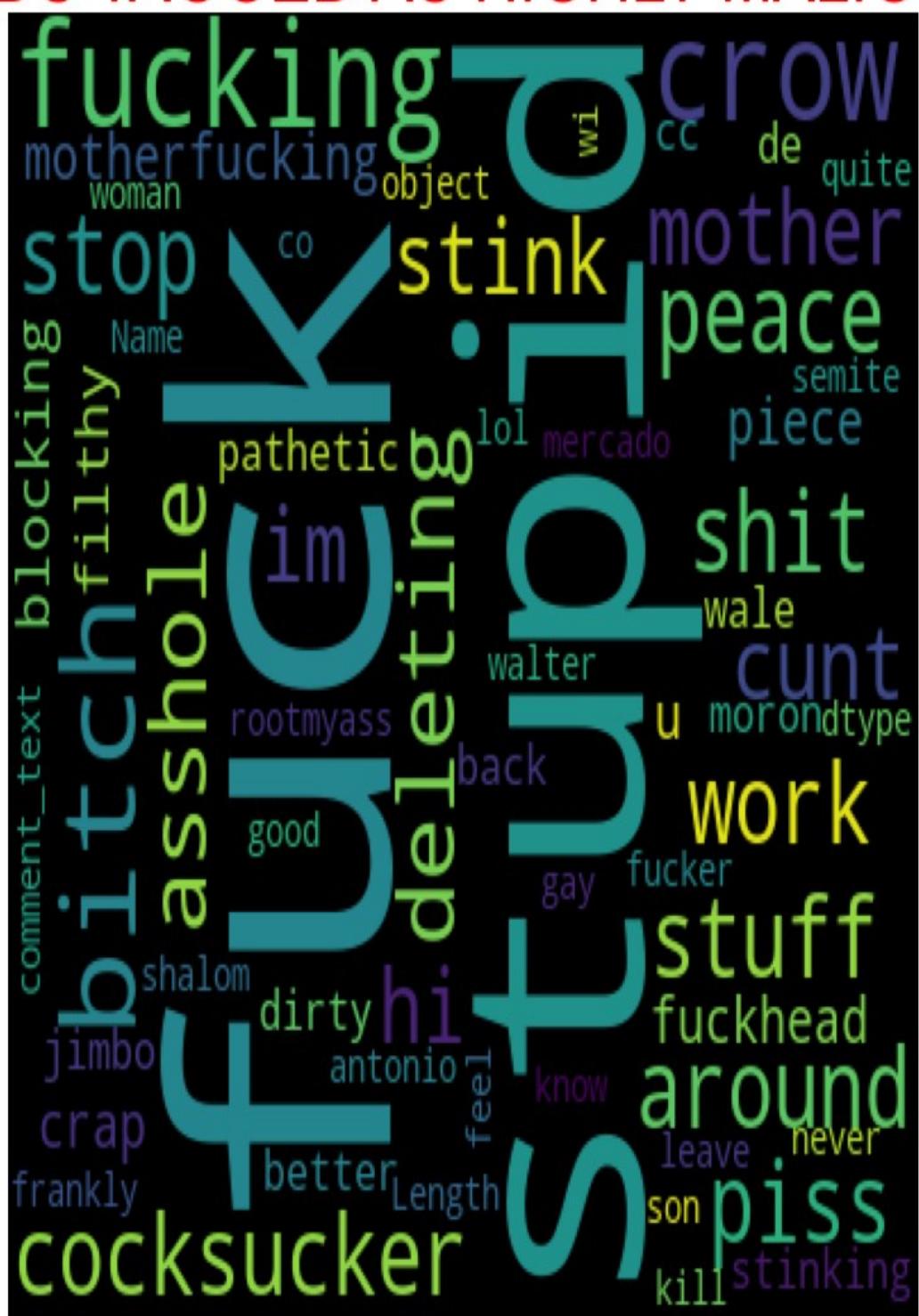
WORDS TAGGED AS MALIGNANT



Key observation.

We can see the foul words that are mostly used in malignant classified sentences we are seeing top 300 words the words which are bigger in size are mostly used.

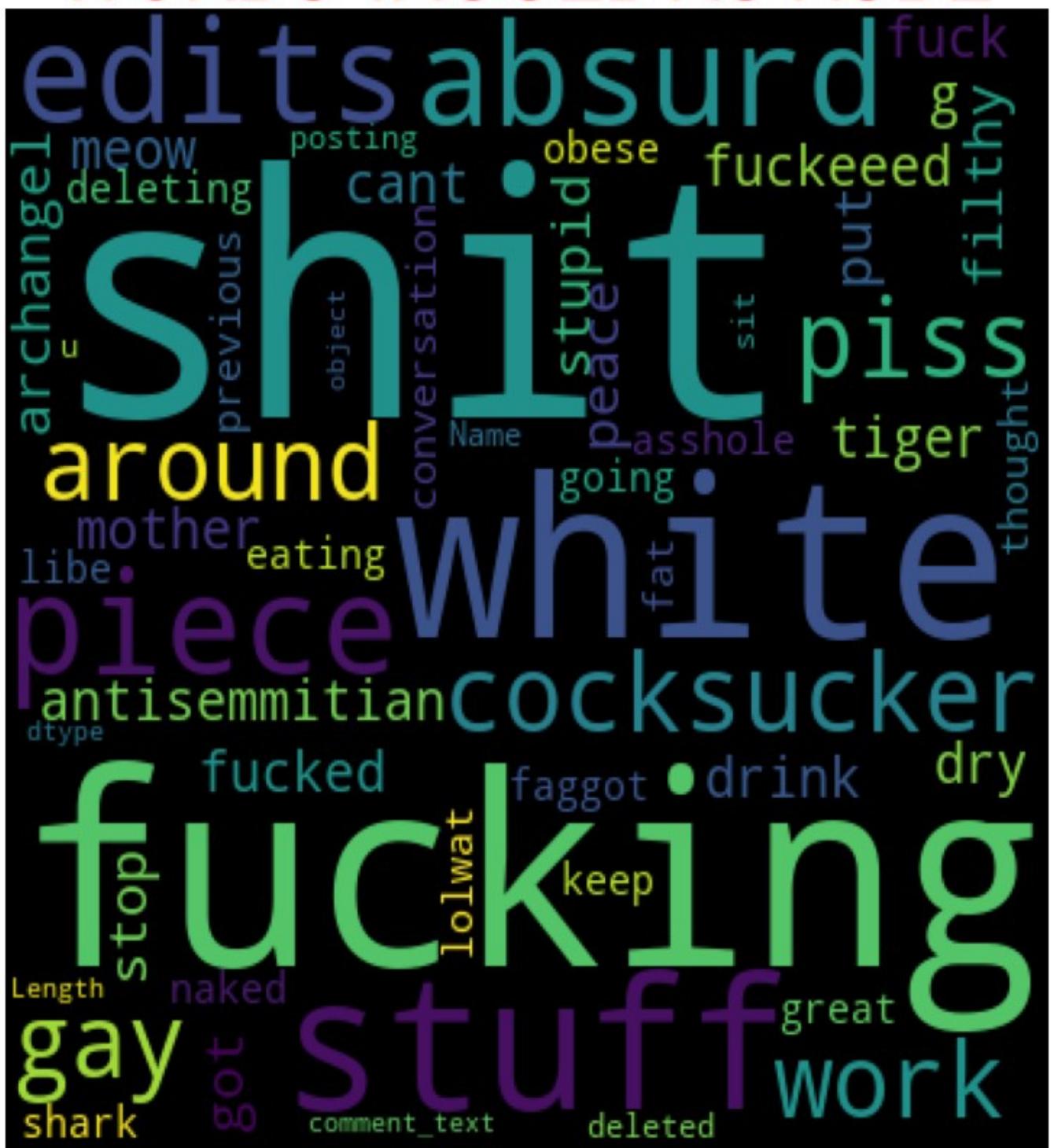
WORDS TAGGED AS HIGHLY MALIGNANT



Key observation.

We can see the foul words that are mostly used in highly_malignant classified sentences we are seeing top 300 words the words which are bigger in size are mostly used.

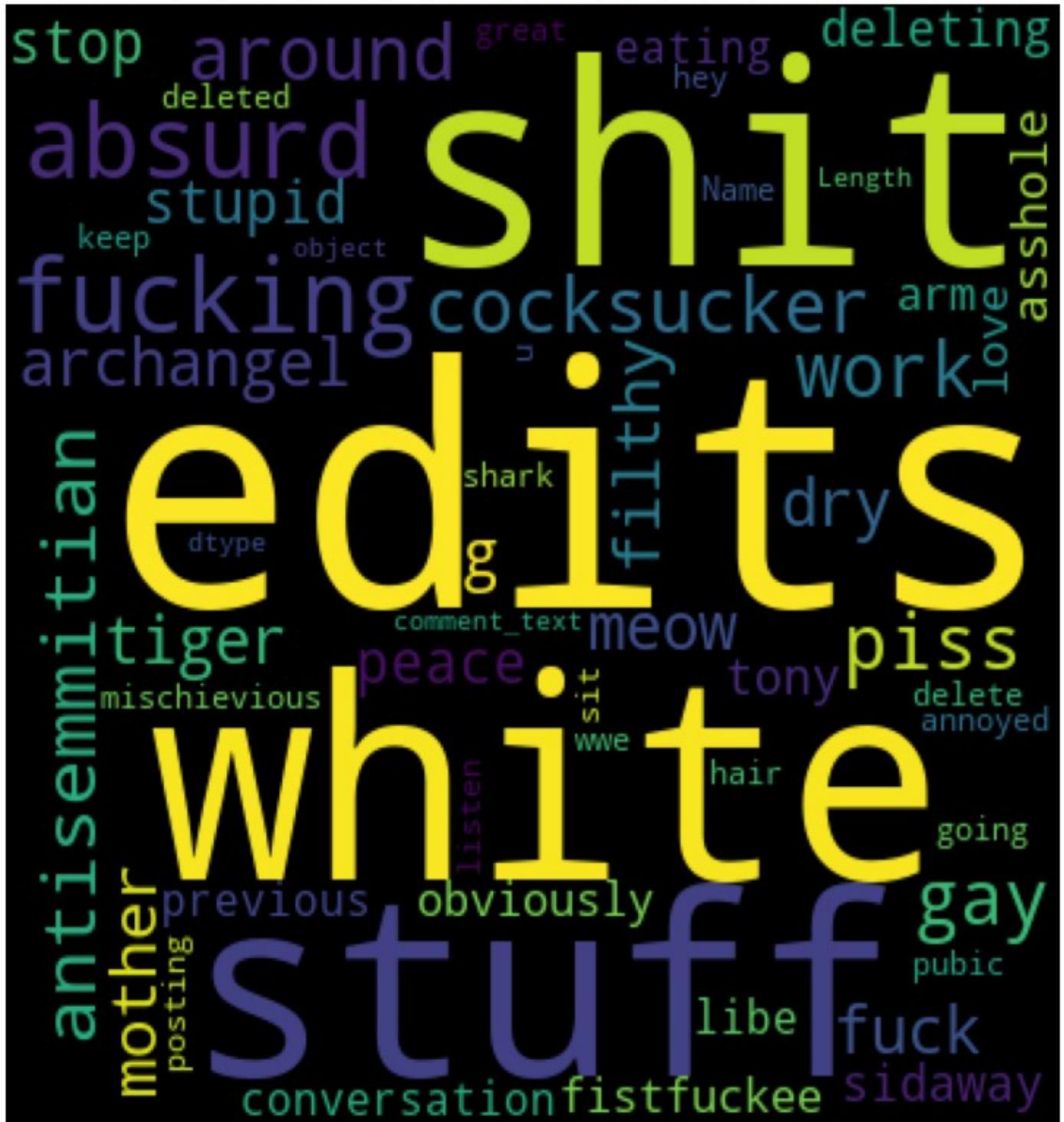
WORDS TAGGED AS RUDE



Key observation.

words that are mostly used in rude

WORDS TAGGED AS ABUSE



Key observation.

words that are mostly used in abuse

200

WORDS TAGGED AS LOATHE



Key observation.

Model/s Development and Evaluation

Testing of Identified Approaches (Algorithms)

I have started the training in selecting the best random state parameter for the model as follows.

```
1 #Importing Machine Learning Model library
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.naive_bayes import MultinomialNB
4 from sklearn.tree import DecisionTreeClassifier
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.ensemble import RandomForestClassifier
7 from sklearn.ensemble import AdaBoostClassifier
8 from sklearn.ensemble import GradientBoostingClassifier
9 from xgboost import XGBClassifier
10 from sklearn.multiclass import OneVsRestClassifier
11 from sklearn.svm import SVC, LinearSVC
12 #from sklearn.multiclass import OneVsRestClassifier
13 from sklearn.model_selection import train_test_split,cross_val_score
14 from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
15 from sklearn.metrics import roc_auc_score, roc_curve, auc
16 from sklearn.metrics import hamming_loss, log_loss
17

1 import timeit, sys
2 import tqdm.notebook as tqdm
3

4 # Creating a function to train and test model
5 def build_models(models,x,y,test_size=0.33,random_state=13):
6     # splitting train test data using train_test_split
7     x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=test_size,random_state=random_state)
8

9     # training models using BinaryRelevance of problem transform
10    for i in tqdm.tqdm(models,desc="Building Models"):
11        start_time = timeit.default_timer()
12
13        br_clf = BinaryRelevance(classifier=models[i]["name"],require_dense=[True,True])
14        print("Training: ",br_clf)
15        br_clf.fit(x_train,y_train)
16
17        print("Testing: ")
18        y_pred = br_clf.predict(x_test)
19
20        ham_loss = hamming_loss(y_test,y_pred)
21        sys.stdout.write(f"\n\tHamming Loss : {ham_loss}")
22
23        ac_score = accuracy_score(y_test,y_pred)
24        sys.stdout.write(f"\n\tAccuracy Score: {ac_score}")
25
26        cl_report = classification_report(y_test,y_pred)
27        sys.stdout.write(f"\n{cl_report}")
28
29        end_time = timeit.default_timer()
30        sys.stdout.write(f"\nCompleted in [{end_time-start_time} sec.]")
```

After selecting the best random state parameter, I have splitted the data into test and train with test size as 33 %. Again, I have imported the required libraries to import my ML algorithms.

Run and evaluate selected models

```
In [53]: 1 # Preparing the List of models for classification purpose
2 models = {
3     "Logistic Regression": {"name": LogisticRegression()},
4     "Random Forest Classifier": {"name": RandomForestClassifier()},
5     "Support Vector Classifier": {"name": LinearSVC(max_iter = 3000)},
6     "Ada Boost Classifier": {"name": AdaBoostClassifier()},
7     "Decision Tree Classifier": {"name": DecisionTreeClassifier()},
8     "KNeighbors Classifier": {"name": KNeighborsClassifier()},
9     "Random Forest Classifier": {"name": RandomForestClassifier()},
10    "Gradient Boosting Classifier": {"name": GradientBoostingClassifier()},
11    "XGBClassifier": {"name": XGBClassifier()},
12    "MultinomialNB": {"name": MultinomialNB()},
13 }
14
15 # Taking one forth of the total data for training and testing purpose
16 half = len(df)//4
17 trained_models = build_models(models,X[:half,:],Y[:half,:])
```

As we can see above, I have imported 10 classification algorithms and I am going to shortlist the best amongst these in basis of accuracy precision recall and F1 scores.

So, like above I have run all the algorithms with the data.

Key-Metrics for success in solving problem under consideration.

I have taken key metrics as Accuracy, Precision, Recall and F1 scores to analysis the best model.

```
=====
Training: BinaryRelevance(classifier=RandomForestClassifier(), require_dense=[True, True])
Testing:
Hamming Loss : 0.022015444993037092
Accuracy Score: 0.906950246866992
      precision    recall   f1-score   support
          0       0.83     0.61     0.70     1314
          1       0.55     0.07     0.13      151
          2       0.84     0.71     0.77      710
          3       0.00     0.00     0.00      41
          4       0.72     0.55     0.62      680
          5       0.83     0.15     0.25     128
   micro avg       0.80     0.57     0.66     3024
   macro avg       0.63     0.35     0.41     3024
weighted avg       0.78     0.57     0.64     3024
samples avg       0.06     0.05     0.05     3024
Completed in [592.8934651 sec.]
```

As we can see above Support Vector Classifier tops the chart, I have selected Support Vector Classifier as my final model and I have saved the same for the further usage. Further I have imported the test data and have done prediction on the same.

CONCLUSION

Key Findings and Conclusions of the Study

The finding of the study is that only few users over online use

```
# Make predictions and view the results
predict_test = best_model.predict(test_vec.toarray())

# Saving predicted values into a CSV file
pd.DataFrame(predict_test).to_csv('Predicted_test_output.csv')

:

df1 = pd.read_csv('Predicted_test_output.csv')
df1.drop('Unnamed: 0', axis=1, inplace=True)
df1.rename({'0':'malignant', '1':'highly_malignant', '2':'rude', '3':'threat', '4':'abuse', '5':'loathe'}, axis='columns', inplace=True)
df2=df_test.copy()
df = pd.concat([df2, df1], axis=1)
df

id      comment_text  length_before_cleaning  length_after_cleaning  malignant
0      yo bitch ja rule succesful ever whats hating s...          367              235       NaN
1      rfc title fine imo                                     50                18       NaN
2      source zawe ashton lapland                           54                26       NaN
3      look back source information updated correct f...          205              109       NaN
4      anonymously edit article                           41                24       NaN
...
...
153159   totally agree stuff nothing long crap           60                37       NaN
153160   throw field home plate get faster throwing cut...          198              107       NaN
153161   okinotorishima category see change agree corre...          423              238       NaN
153162   one founding nation eu germany law return quit...          502              319       NaN
153163   stop already bullshit welcome fool think kind ...          141                74       NaN

153164 rows × 5 columns

df.to_csv('test_dataset_predictions2.csv', index=False)
```

unparliamentary language. And most of these sentences have more stop words, and are being long. As discussed before few motivated disrespectful crowds uses these foul languages in the online forum to bully the people around and to stop them from doing the things that they are supposed to do. Our Study helps the online forms and social media to induce a ban to profanity or usage of profanity over these forms.