FLIP ROBO

Malignant Comments Classification Project

Submitted by:

Khushboo Khatri

ACKNOWLEDGMENT

To start with I would like to show my gratitude DataTrained team for their continuous support, and provide us with the knowledge required to work in the data science field.

I would like to extend my sincere appreciation to FlipRobo Technologies, to provide us the opportunity to work on such exciting project, so that we can put our theoretical knowledge in practical use and refine our skills in the field of Data Science.

I would also like to thank our SME Shwetank Mishra, to provide us with necessary guidance whenever needed.

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 which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

- ## Conceptual Background of the Domain Problem

  Online platforms and social media become the place where people share the thoughts freely without any partiality and overcoming all the race people share their thoughts and ideas among the crowd.

  Social media is a computer-based technology that facilitates the sharing of ideas, thoughts, and information through the building of virtual networks and communities. By design, social media is Internet-based and gives users quick electronic communication of content. Content includes personal information, documents, videos, and photos. Users engage with social media via a computer, tablet, or smartphone via web-based software or applications.

  While social media is ubiquitous in America and Europe, Asian countries like India lead the list of social media usage. More than 3.8 billion people use social media.

  In this huge online platform or an online community there are some people or some motivated mob wilfully bully others to make them not to share their thought in rightful way. They bully others in a foul language which among the civilized society is seen as ignominy. And when innocent individuals are being bullied by these mob these individuals are going silent without speaking anything. So, ideally the motive of this disgraceful mob is achieved.

  To solve this problem, we are now building a model that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

- ## Review of Literature

The purpose of the literature review is to:

1. Identify the foul words or foul statements that are being used.

2. Stop the people from using these foul languages in online public forum.

To solve this problem, we are now building a model using our machine language technique that identifies all the foul language and foul words, using which the online platforms like social media principally stops these mob using the foul language in an online community or even block them or block them from using this foul language.

I have used 6 different Classification algorithms and shortlisted the best on basis on the metrics of performance and I have chosen one algorithm and build a model in that algorithm.

- Motivation for the Problem Undertaken

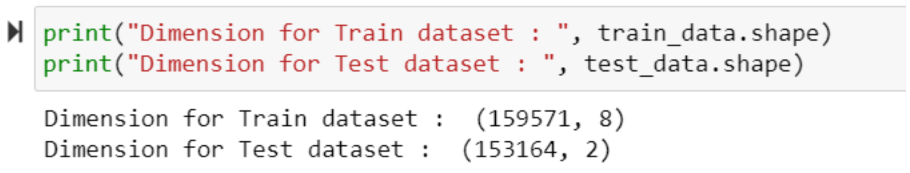
  One of the first lessons we learn as children is that the louder you scream and the bigger of a tantrum you throw, you more you get your way. Part of growing up and maturing into an adult and functioning member of society is learning how to use language and reasoning skills to communicate our beliefs and respectfully disagree with others, using evidence and persuasiveness to try and bring them over to our way of thinking. Social media is reverting us back to those animalistic tantrums, schoolyard taunts and unfettered bullying that define youth, creating a dystopia where even renowned academics and dispassionate journalists transform from Dr. Jekyll into raving Mr. Hydes, raising the critical question of whether social media should simply enact a blanket ban on profanity and name calling? Actually, ban should be implemented on these profanities and taking that as a motivation I have started this project to identify the malignant comments in social media or in online public forms.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  To start with our Problem, as we two dataset one is train dataset and other one is test dataset.

We start with getting the dataset, checking their dimensions using .shape method.

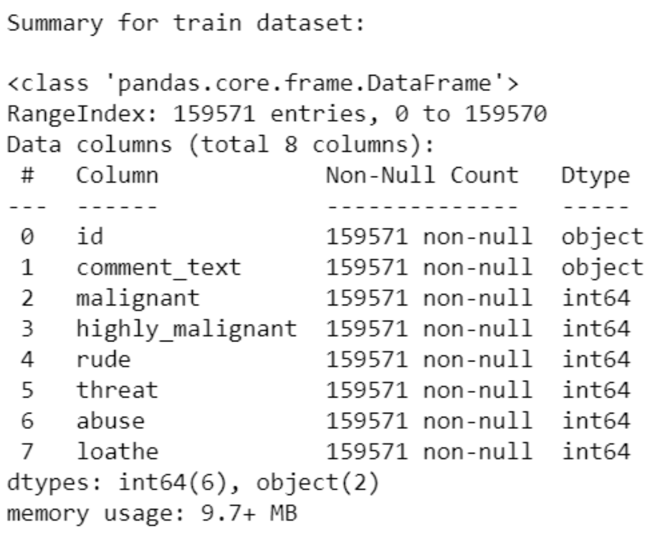


```
print("Dimension for Train dataset : ", train_data.shape)
print("Dimension for Test dataset : ", test_data.shape)
```

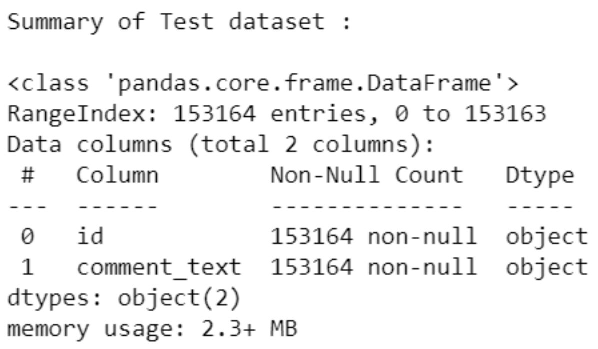
```
Dimension for Train dataset :  (159571, 8)
Dimension for Test dataset :  (153164, 2)
```

We can see that we have 8 columns in train dataset and 159571 rows, while test dataset has 2 columns and 153164 rows.

Then we find the general summary of the dataset using .info()



```
Summary for train dataset:

<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
```

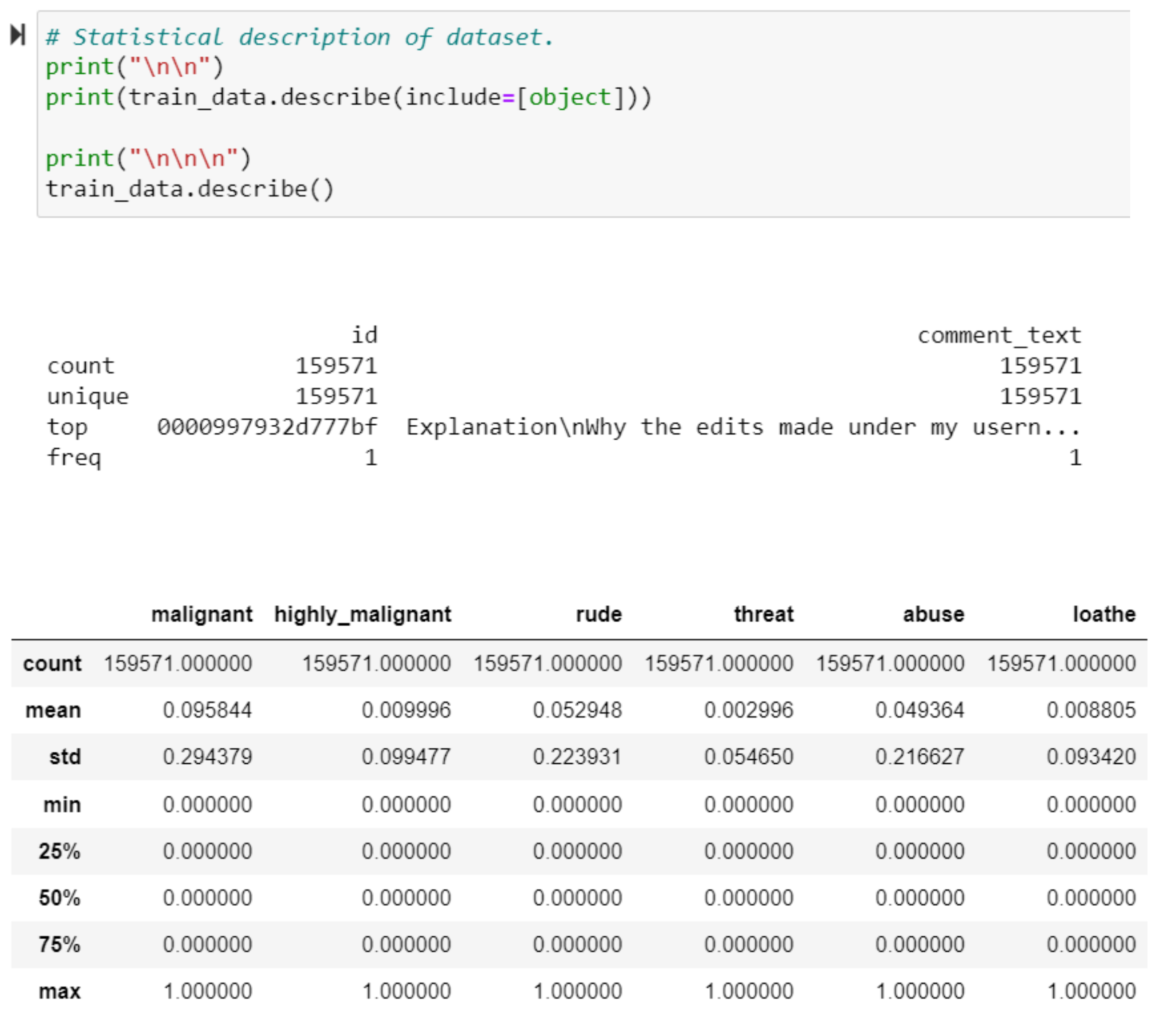


```
Summary of Test dataset :

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153164 entries, 0 to 153163
Data columns (total 2 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   id            153164 non-null  object
 1   comment_text  153164 non-null  object
dtypes: object(2)
memory usage: 2.3+ MB
```

From above observation we can say that we don't have any null values in either of the dataset. We can also observe that id and comment_text are object type while other columns are numerical type data.

We need to pre-process these object type of columns, to get some useful information out of it.

For Statistical analysis of the dataset, we use .describe() for it.



```
# Statistical description of dataset.
print("\n\n")
print(train_data.describe(include=[object]))

print("\n\n\n")
train_data.describe()
```

```
                      id                                       comment_text
count             159571                                             159571
unique            159571                                             159571
top     0000997932d777bf  Explanation\nWhy the edits made under my usern...
freq                   1                                                  1
```

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

From here we can observe that Both id and comment_text has total identical value throughout. From comment_text we need to do some NLP to get some meaningful information out of it.

Id seems to be the column which has unique value identifying particular user. We can eliminate this column, in order to make dataset suitable for model building.

We did perform similar function on test dataset as well.

- ## 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:

- **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.

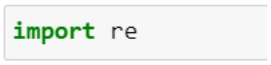
This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do good amount of data exploration and derive some interesting features using the comments text column available.

- ## Data Preprocessing Done

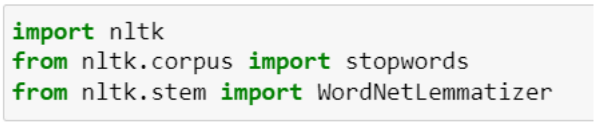
  We did preprocessing of our comment_text column using NLP (Natural Language Processing), in which we perform sentence deconstruction (data cleaning), stemming, lemmatization in order to clean and make text column more understandable to machine.

  To start with we import or download all the necessary libraries and modules required.

  Then we define a function which deconstruct the data on the comment_text column, followed by downloading stop_words, punctuation(punkt), wordnet, omw-1.4.

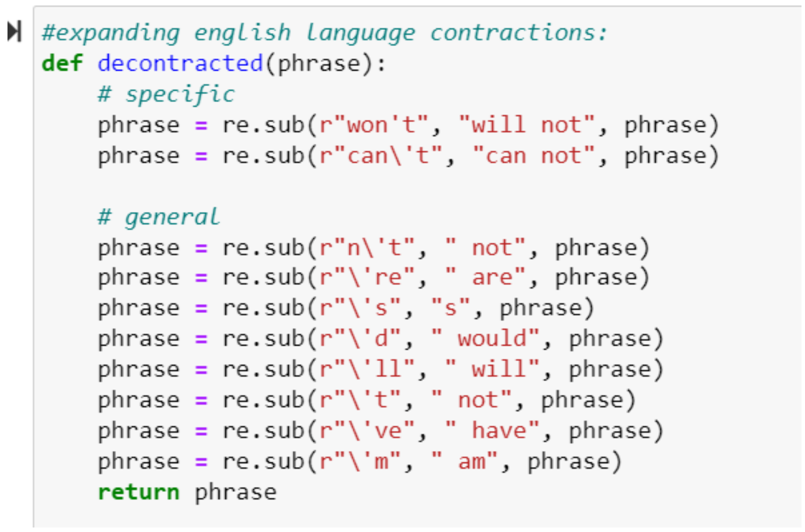


```python
import re
```



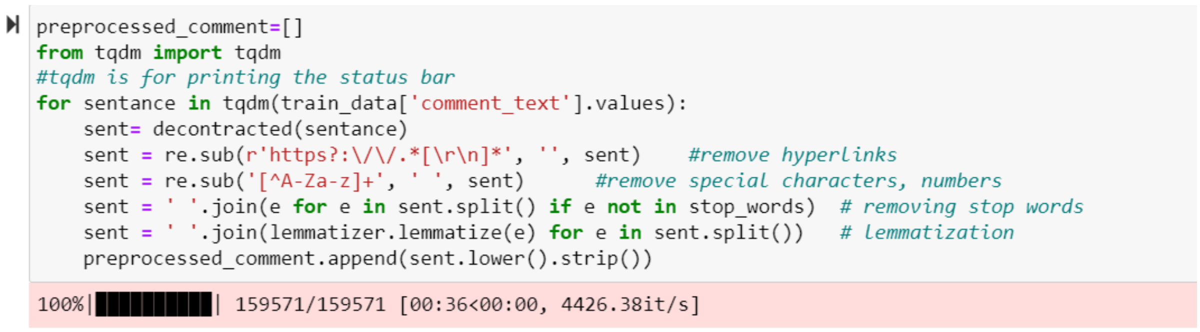
```python
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

#Deconstruction Function



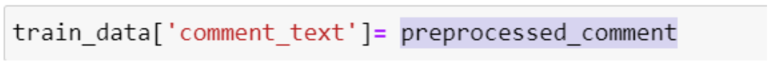
```python
#expanding english language contractions:
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", "s", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

# Preprocessing the comment_text in train dataset



```python
preprocessed_comment=[]
from tqdm import tqdm
#tqdm is for printing the status bar
for sentance in tqdm(train_data['comment_text'].values):
    sent= decontracted(sentance)
    sent = re.sub(r'https?:\/\/.*[\r\n]*', '', sent)     #remove hyperlinks
    sent = re.sub('[^A-Za-z]+', ' ', sent)       #remove special characters, numbers
    sent = ' '.join(e for e in sent.split() if e not in stop_words)  # removing stop words
    sent = ' '.join(lemmatizer.lemmatize(e) for e in sent.split())   # lemmatization
    preprocessed_comment.append(sent.lower().strip())
```

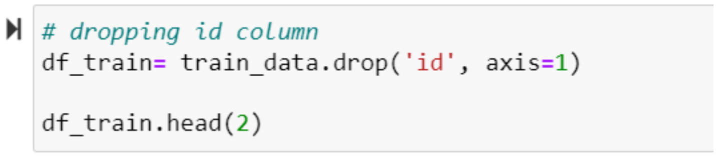
```
100%|██████████| 159571/159571 [00:36<00:00, 4426.38it/s]
```



```python
train_data['comment_text']= preprocessed_comment
```

We did perform similar function on test dataset as well.

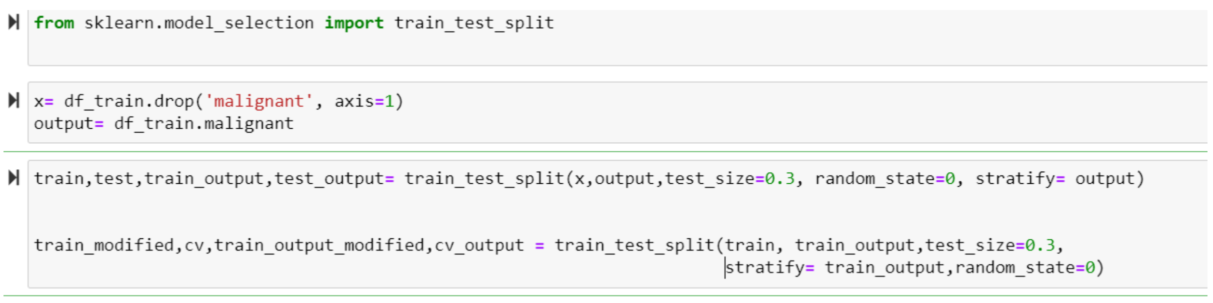
Also during preprocessing, and Data cleaning, we did drop Id column from both train dataset and test dataset, as it simply indicates all unique values which represent particular column, it is not of much importance during model building.



```python
# dropping id column
df_train= train_data.drop('id', axis=1)

df_train.head(2)
```

Our comment_text column is prepared for encoding, however before that, we will do some data splitting. We split our data into three sets, train, cv, test. We will use cv dataset in cross validation, as it helps to increase the efficiency of the model. And also helps us to check how well our model is working.

We perform train test split, at this stage and taking the test size of 30%.



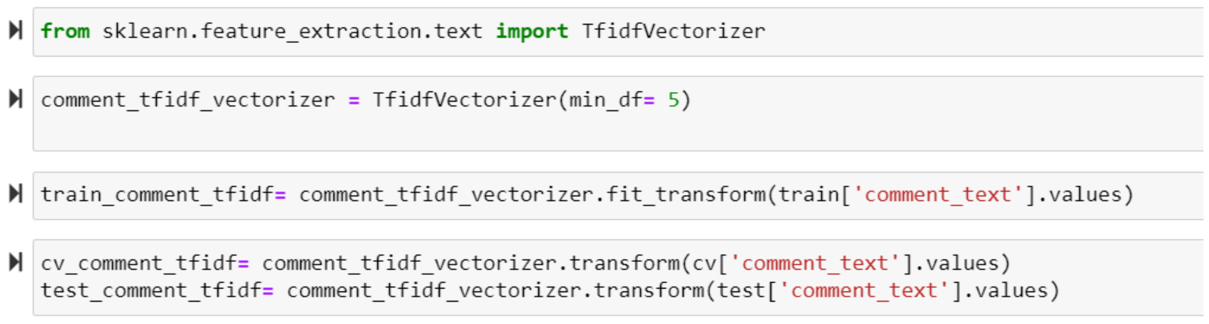
```python
from sklearn.model_selection import train_test_split
```

```python
x= df_train.drop('malignant', axis=1)
output= df_train.malignant
```

```python
train,test,train_output,test_output= train_test_split(x,output,test_size=0.3, random_state=0, stratify= output)


train_modified,cv,train_output_modified,cv_output = train_test_split(train, train_output,test_size=0.3,
                                                            stratify= train_output,random_state=0)
```

Here we use stratify as we want the same ratio of 1 & 0 in our training and testing set and cv set as it is present in the main dataset.

After this we did Data Encoding, using Vectorization.



```python
from sklearn.feature_extraction.text import TfidfVectorizer
```

```python
comment_tfidf_vectorizer = TfidfVectorizer(min_df= 5)
```

```python
train_comment_tfidf= comment_tfidf_vectorizer.fit_transform(train['comment_text'].values)
```

```python
cv_comment_tfidf= comment_tfidf_vectorizer.transform(cv['comment_text'].values)
test_comment_tfidf= comment_tfidf_vectorizer.transform(test['comment_text'].values)
```

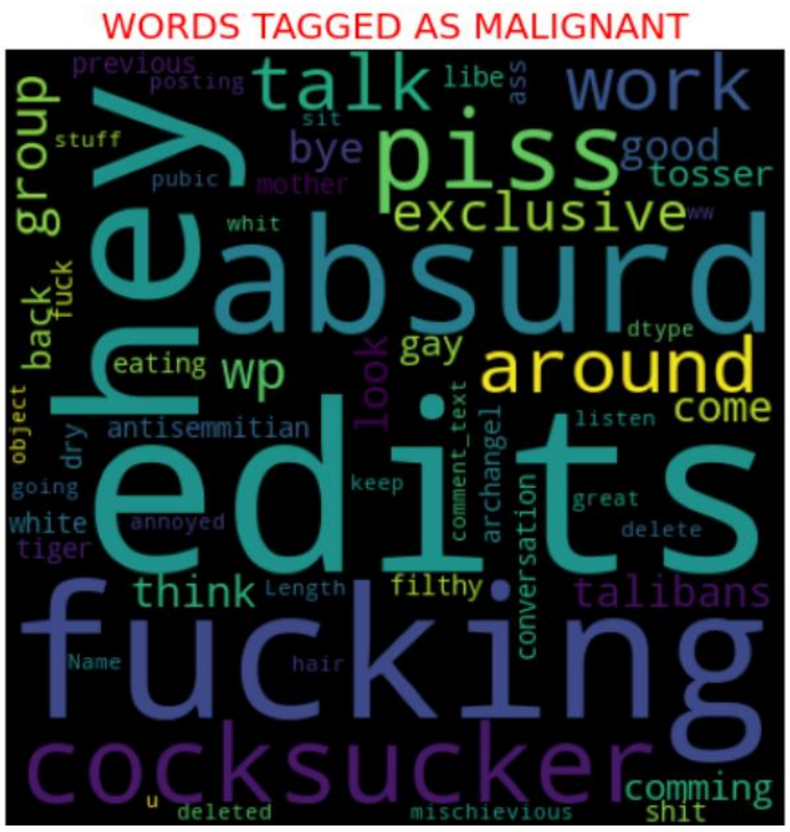
We encode the comment_text column using TfidfVectorizer. Now our data is model ready, we will use this For Model building Purpose.

- Data Inputs- Logic- Output Relationships

In our dataset we got the input in the form of text, with the help of NLP we built a model which helps us to identify weather a comment is malignant or not, highly malignant or not, rude or not and other categories as well.

We also did visualization to find the red flags words in the comments which our model considered as offensive or text with bad intention using wordcloud.

```
# making a dataframe which consist of malignant comment_text using which we will identify desired text.
df_malignant=df_train[(df_train['malignant']==1)]


#Plotting for malignant
wordcloud=WordCloud(height=400,width=400,max_words=400).generate(str(df_malignant['comment_text']))
plt.figure(figsize=(8,6))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.title(label='WORDS TAGGED AS MALIGNANT',fontdict={'fontsize':20, 'fontweight':20, 'color':'red'})
plt.show()
```



Although our label is malignant , however, we plot similar wordcloud for highly malignant and rude labels.

From here it is very clear that certain set of words if find in the comment_text the model would consider them malignant or any other label which we want to tried on. Also the large font words are mostly used ones.

- State the set of assumptions (if any) related to the problem under consideration

Here, we assume that malignant is our label column, and comment_text is our primary feature column, on the basis of which we need to build the model around.

- Hardware and Software Requirements and Tools Used

1. Python
2. Numpy
3. Pandas
4. Matplotlib
5. Seaborn
6. Warnings

#NLP Libraries

1. IPython
2. NLTK
3. Stopwords
4. WordNetLemmitizer
5. re
6. punkt
7. WordCloud
8. tqdm
9. TfidfVectorizer

Model/s Development and Evaluation

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

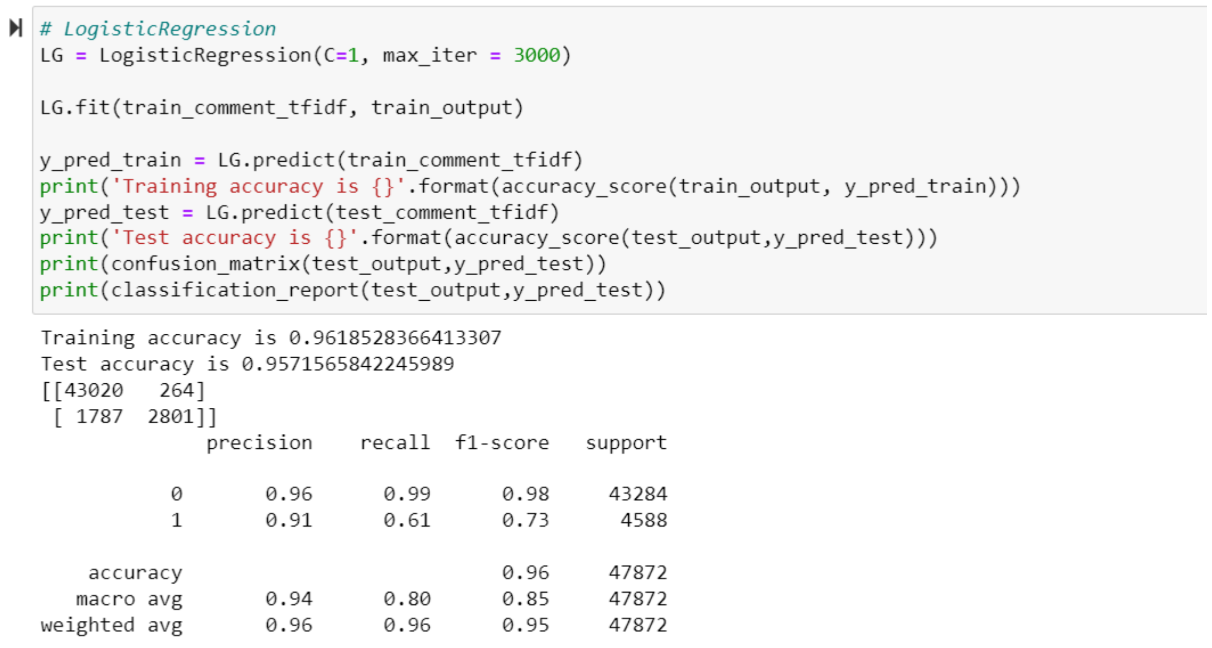
We analyse the data and also do some statistical summarization, then do some data cleaning, data pre processing , encoding, splitting data into train test and cv using train_test_split().

With this pre processed encoded data we starts to train our model using different algorithms we also did some cross validation of each model to cross check the approach and also efficiency.

- Testing of Identified Approaches (Algorithms):
  1. Logistic Regression
  2. Decision Tree Classifier
  3. Random Forest Classifier
  4. XGBoost Classifier
  5. AdaBoostClassifier
  6. KNeighborsClassifier

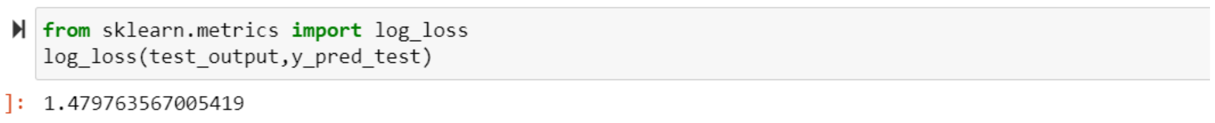
- Run and Evaluate selected models:

  Our selected model for this dataset is Logistic Regression. Here is how we build the model.



```
# LogisticRegression
LG = LogisticRegression(C=1, max_iter = 3000)

LG.fit(train_comment_tfidf, train_output)

y_pred_train = LG.predict(train_comment_tfidf)
print('Training accuracy is {}'.format(accuracy_score(train_output, y_pred_train)))
y_pred_test = LG.predict(test_comment_tfidf)
print('Test accuracy is {}'.format(accuracy_score(test_output,y_pred_test)))
print(confusion_matrix(test_output,y_pred_test))
print(classification_report(test_output,y_pred_test))
```

```
Training accuracy is 0.9618528366413307
Test accuracy is 0.9571565842245989
[[43020   264]
 [ 1787  2801]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     43284
           1       0.91      0.61      0.73      4588

    accuracy                           0.96     47872
   macro avg       0.94      0.80      0.85     47872
weighted avg       0.96      0.96      0.95     47872
```



```
from sklearn.metrics import log_loss
log_loss(test_output,y_pred_test)
```

```
]:  1.479763567005419
```

```
# Cross validation
from sklearn.model_selection import cross_val_score

# validating accuracy

scr= cross_val_score(LG,cv_comment_tfidf,cv_output,scoring='precision' ,cv=10)
LG_scr=scr.mean()
print("Cross Validation Score For Logistic Regression model : ", LG_scr)
```

```
Cross Validation Score For Logistic Regression model :  0.9473584321001176
```

We did the scoring in cross validation using precision score, as we already see the number of 1 is low as compared to 0, so this seems to be a good fit to check the precision of the model.

- ## Key Metrics for success in solving problem under consideration:

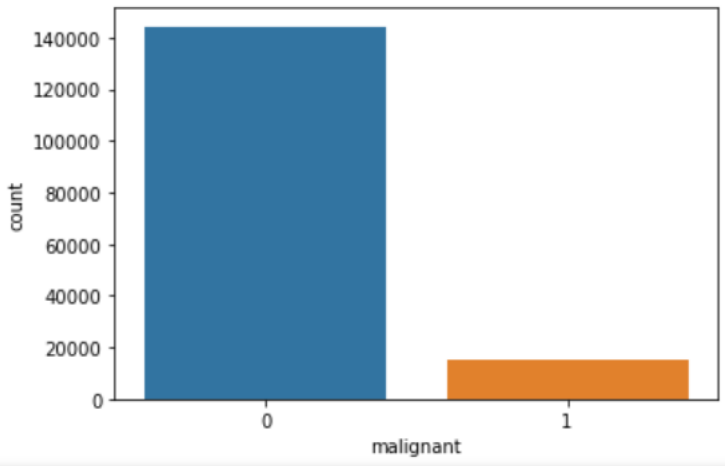
  In our model the primary metrics on the basis of which we are evaluating our model is precision and log_loss.
  While Accuracy_score is our secondary metrics.

  We decide on a model with high precision score and low log_loss.

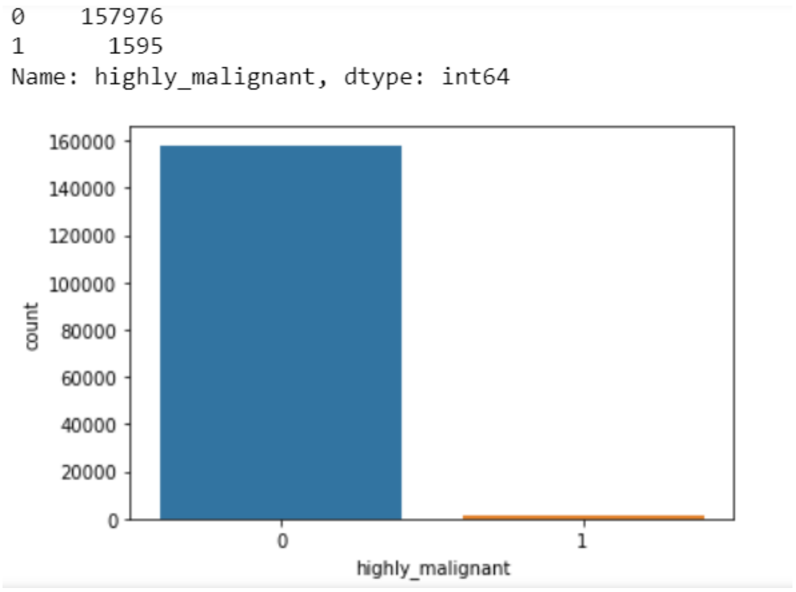
- ## Visualizations:

  We start our EDA with univariate analysis, then we move to Multivariate analysis.

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

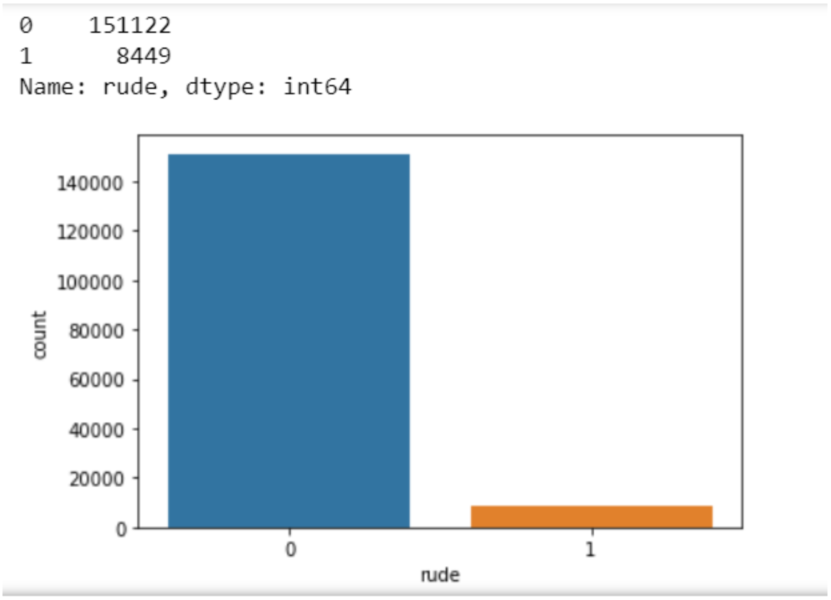


From above observation we can see that malignant (1) is very low as compared to not malignant(0) in the dataset, which simply means our dataset is imbalanced, and we need to deal with it accordingly.



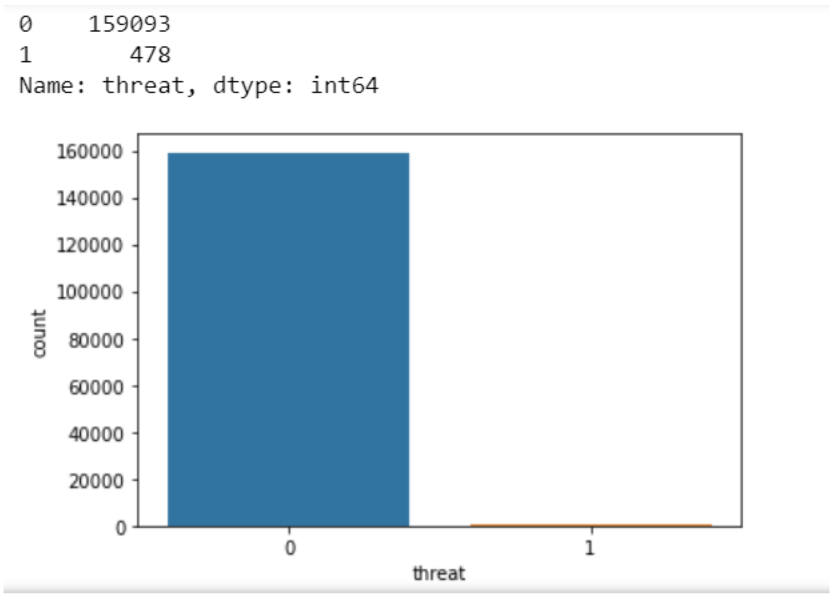
```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```

We can observe that we got very less comments which are highly_malignant as value of 1 here is extremely low compared to 0.



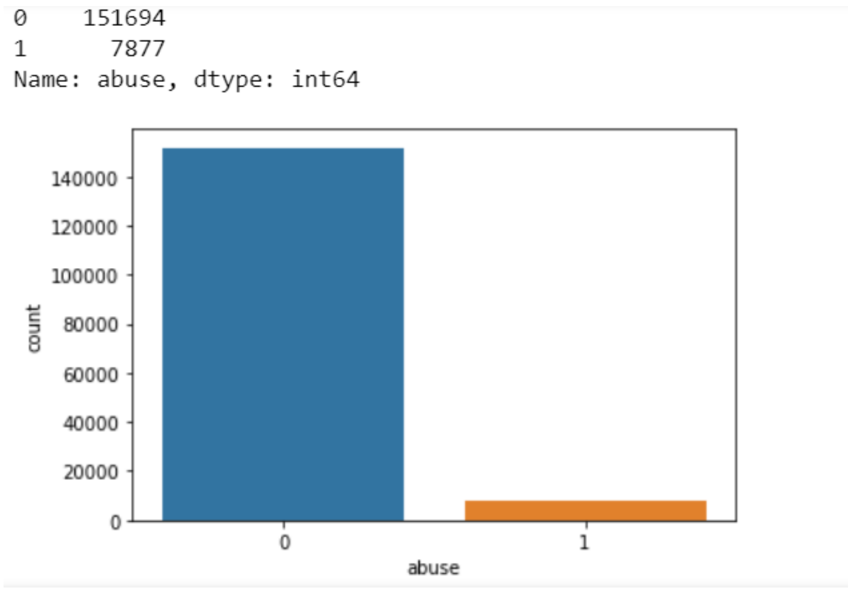
```
0    151122
1      8449
Name: rude, dtype: int64
```

We can observe that we definitely get rude comments more that highly malignant comments. However even here the co unt of 1 is way less than 0.



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

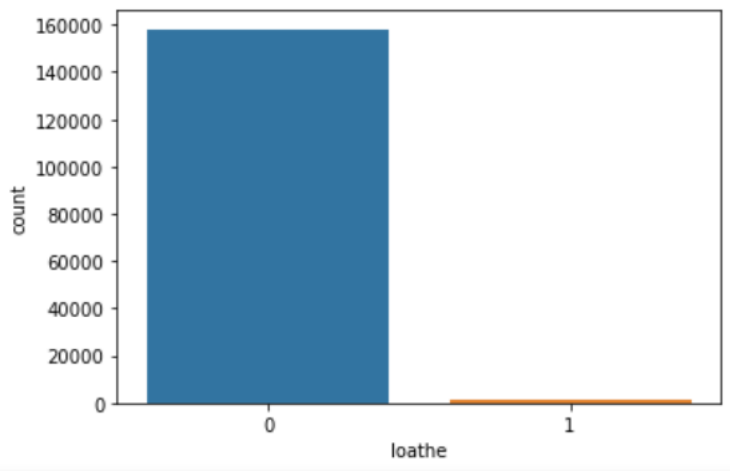
From above observation we can see that there were very few threat comments in the entire dataset.



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

Rude and abuse comments are almost same in number in entire dataset.

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



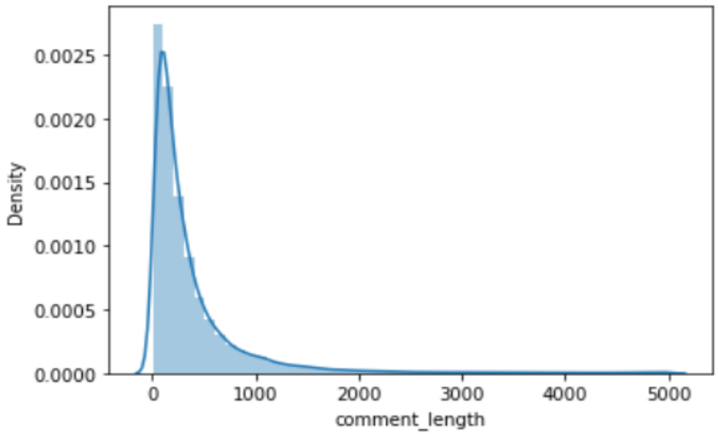
From above observation we can see there were few loathe comments in the dataset. We also analyse that negative comments are very less in number as compared to neutral of positive comments.

```
#As we visualize above the Bad comment is much lower than the Good comments lets see that mathematicaly.
Bad_comment = train_data[(train_data['malignant']!=0) + (train_data['highly_malignant']!=0) + (train_data['rude']!=0) +
                         (train_data['threat']!=0) + (train_data['abuse']!=0) + (train_data['loathe']!=0)]
percent=len(Bad_comment)/len(train_data)*100
print('Percentage of Bad/Negative comments = ',percent)
print('Percentage of Positive/Neutral comments = ', (100-percent))
```

```
Percentage of Bad/Negative comments =  10.167887648758233
Percentage of Positive/Neutral comments =  89.83211235124176
```

From here we can observe that merely 10% of the data, in the entire dataset is consist of negative /bad/offensive comments while remaining 89-90% data is way better, i.e. comments are either positive or neutral in tone. Which on the bright side is good thing to know, however that much negativity can be toxic for some, so its always a good idea to reduce/remove them as much as possible.

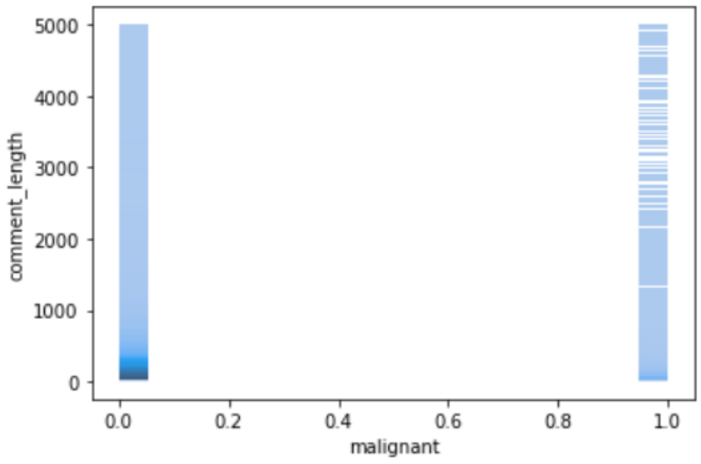
```
#let's see how comment_lenght is distributed.

sns.distplot(train_data['comment_length'])

plt.show()
```



We can see that, most of the comment are with in 650 word limit, there were few very large comment having upto 5000 words.

```
# Multivariate Analysis
sns.histplot(x='malignant', y='comment_length', data= train_data)
```

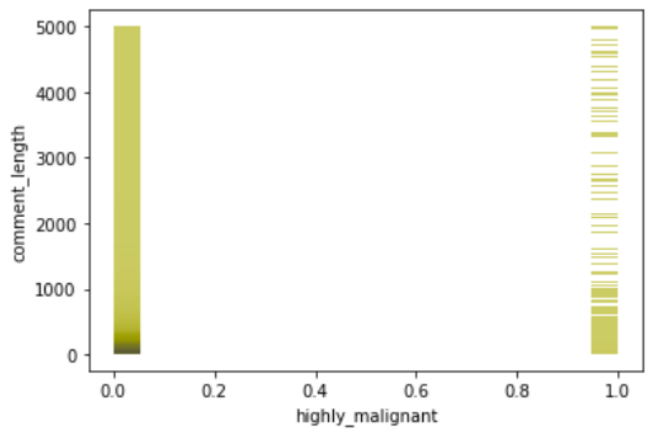
```
]: <AxesSubplot:xlabel='malignant', ylabel='comment_length'>
```



We can observe that malignant not necessary depends in length of comment_text., let's check for other columns as well.

```
sns.histplot(x='highly_malignant', y='comment_length', data= train_data, color='yellow' )
```

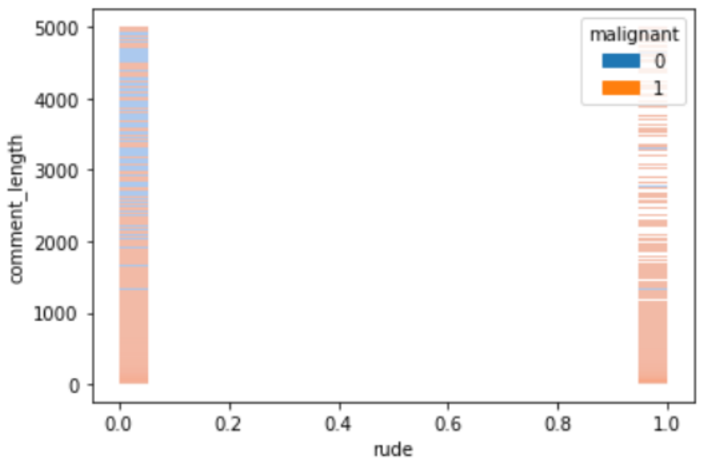
```
]: <AxesSubplot:xlabel='highly_malignant', ylabel='comment_length'>
```



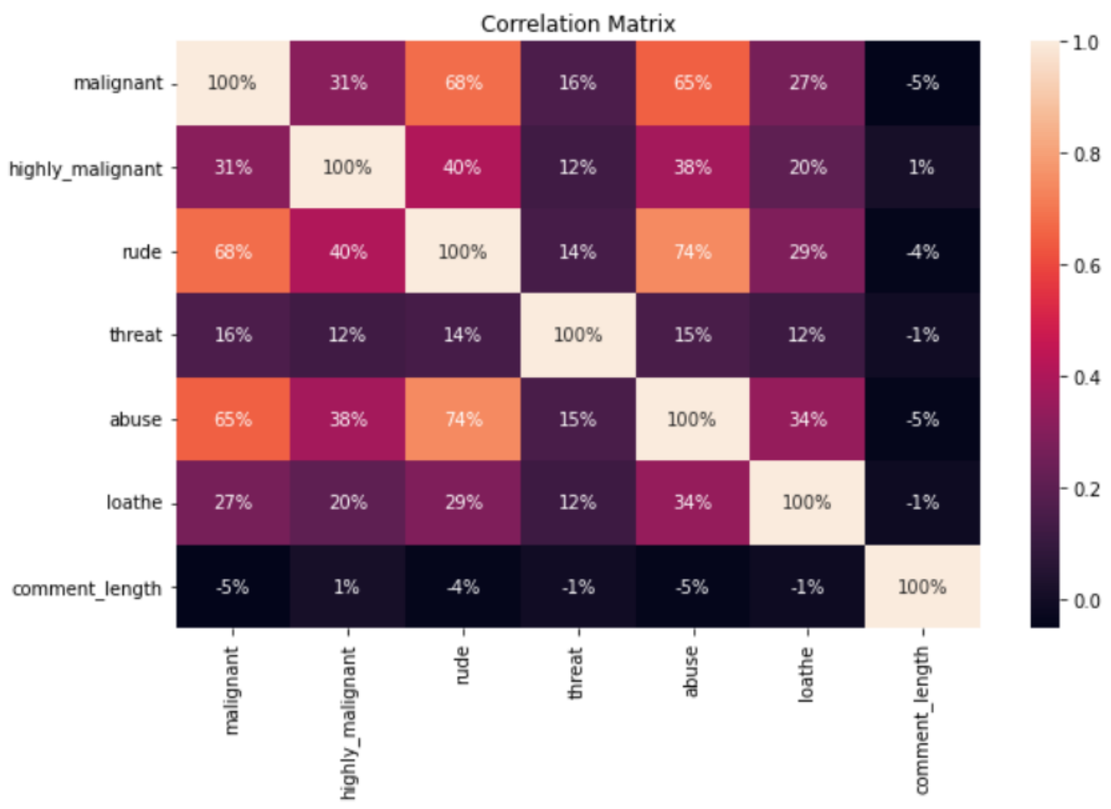
we can how highly malignant data is distributed on comments lenght. It is not necessary that large comment is highly malignant. Lenght of comment does not defines the higly malignant characteristics of comment_text.

```
sns.histplot(x='rude', y='comment_length',hue='malignant', data=train_data)
```

```
]: <AxesSubplot:xlabel='rude', ylabel='comment_length'>
```



We can observe that some of the rude comments can be malignant as well.

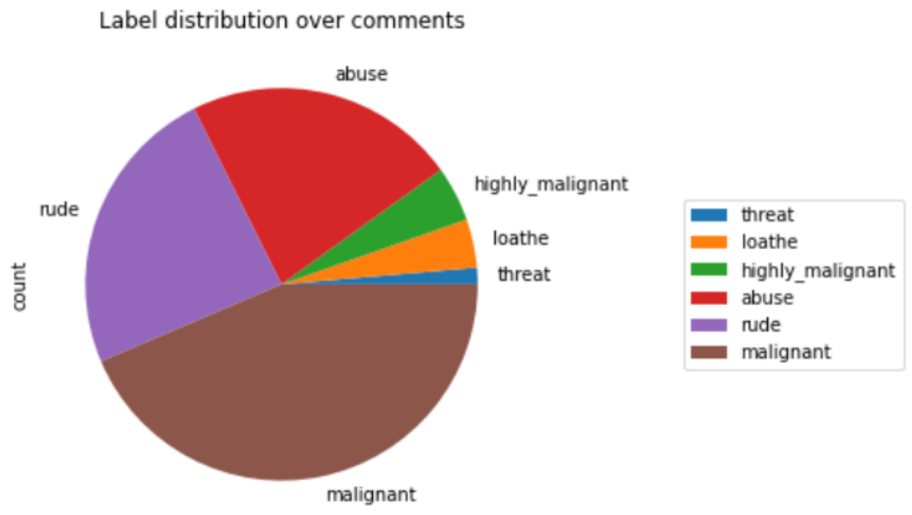


	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length
malignant	100%	31%	68%	16%	65%	27%	-5%
highly_malignant	31%	100%	40%	12%	38%	20%	1%
rude	68%	40%	100%	14%	74%	29%	-4%
threat	16%	12%	14%	100%	15%	12%	-1%
abuse	65%	38%	74%	15%	100%	34%	-5%
loathe	27%	20%	29%	12%	34%	100%	-1%
comment_length	-5%	1%	-4%	-1%	-5%	-1%	100%

From above observation we can establish the fact that comment_lenght is not much of significant importance. Also, some of the comments has multiple labels. Our focus point here is Malignant and we observe that rude and abuse shows strong correlation with it.

In common man’s term we can say that mostly rude and abusive comments are malignant in nature.

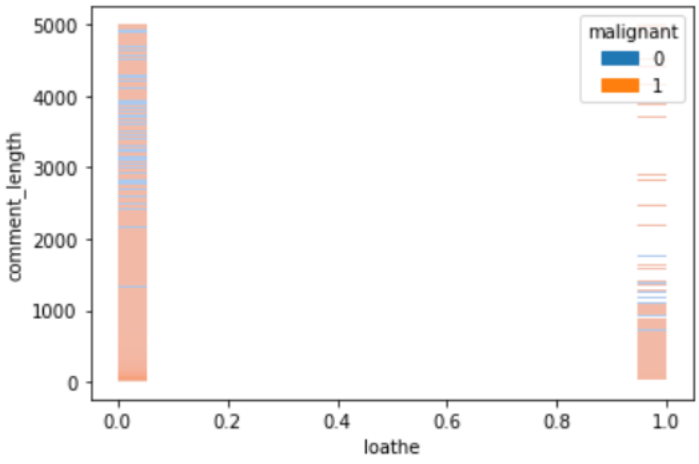
One more thing is seen from here is that threat comments are clearly distinctive. One cannot miss these with other comment easily.



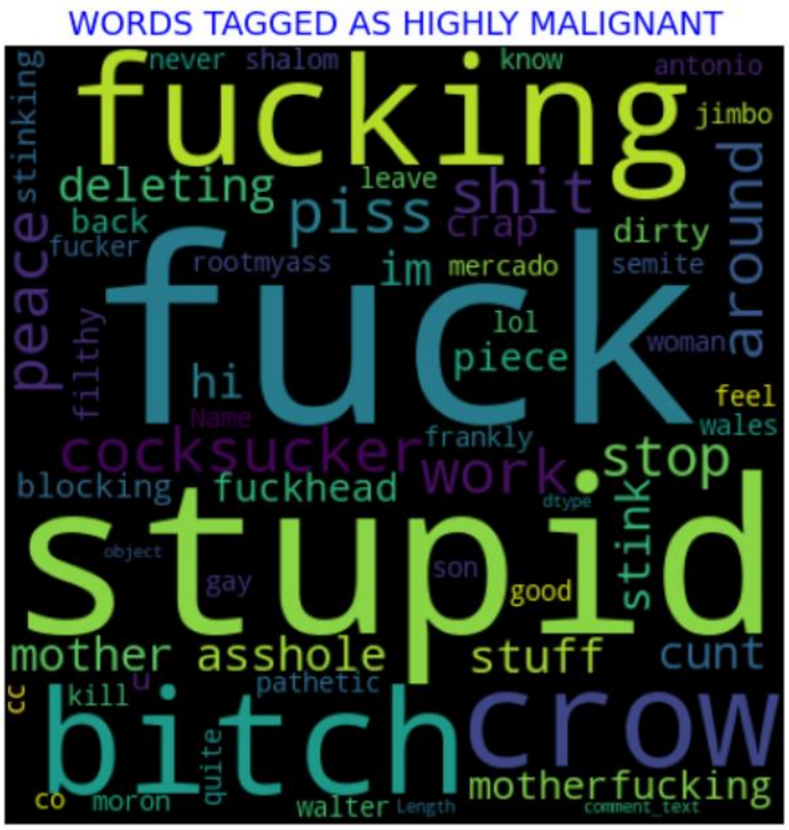
From above observation we can see that most of our comments has malignant as label, and least number of threat comment is present in the dataset. Rude and abuse comments are also good in numbers.

```
sns.histplot(x='loathe',y='comment_length', hue='malignant', data= train_data)
```

```
<AxesSubplot:xlabel='loathe', ylabel='comment_length'>
```

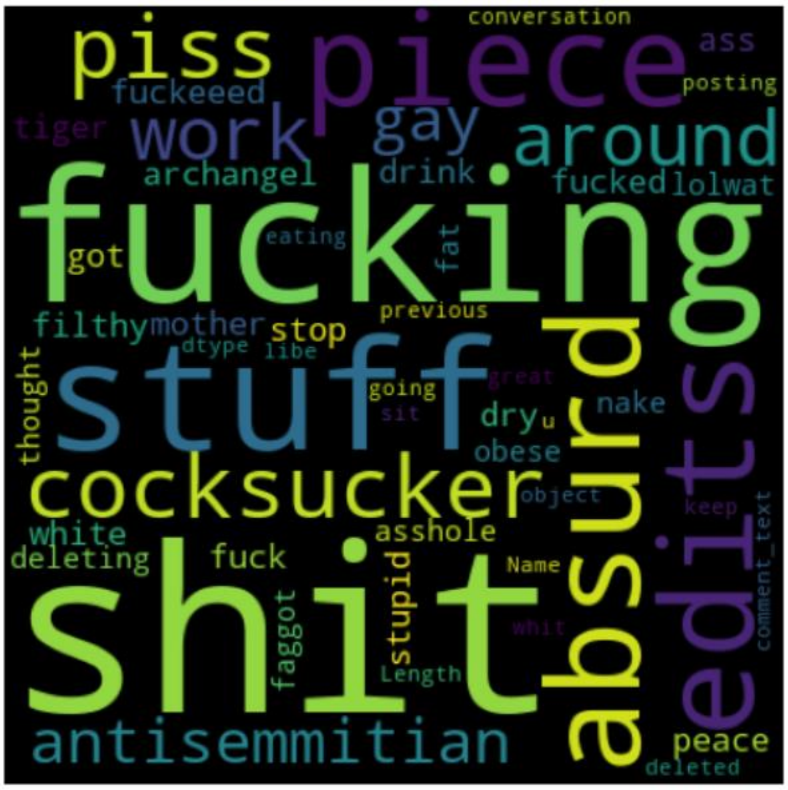


From above observation we can say that some of the loathe and malignant comment text are same.



This is wordcloud representation of all the tag which represents as highly malignant.

WORDS TAGGED AS RUDE



From here it is very clear that certain set of words if find in the comment_text the model would consider them malignant or any other label which we want to tried on. Also, the large font words are mostly used ones.

- Interpretation of the Results

From above visualization it is clear that there are certain tag/words which makes the comment malignant.

It is very necessary to identify them using machine learning model so that we spam them in real time, and avoid spreading negativity even in the virtual world.

CONCLUSION

- ## Key Findings and Conclusions of the Study

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

  NLP is a great tool which helps us to work with text data, and make the maximum use of it to convert it into machine understandable form and hence helps in model building.

  Our selected model on the basis of which we predicted for test dataset is Logistic Regression. And also, we store the result label in csv form.

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

  The use of social media is the most common trend among the activities of today's people. Social networking sites offer today's teenagers a platform for communication and entertainment. They use social media to collect more information from their friends and followers. The vastness of social media sites ensures that not all of them provide a decent environment for children. In such cases, the impact of the negative influences of social media on teenage users increases with an increase in the use of **offensive language** in social conversations. This increase could lead to **frustration**, **depression** and a large change in their behaviour. Hence, I propose a novel approach to classify bad language usage in text conversations. I have considered the English medium for textual conversation. I have developed our system based on a foul language classification approach; it is based on an improved version of a Random Forest Classification Algorithm that detects offensive language usage in a conversation. As per our evaluation, we found that lesser number of users conversation is not decent all the time. We trained 159571 observations for eight context categories using a Random Forest algorithm for context detection. Then, the system classifies the use of foul language in one of the trained contexts in the text conversation. In our testbed, we observed 10% of participants used foul language during their text conversation. Hence, our proposed approach can identify the impact of foul language in text conversations using a classification technique and emotion detection to identify the foul language usage.

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

The limitation of the study is that we have an imbalanced data so our model learnt more about the non-abusive sentence more than the abusive sentence. Which makes our model act like an overfit model when tested with live data. And also, model tend to not identify a foul or a sarcastically foul language. Also, here our label was malignant, we can even build a model for all the other categories individually.