FLIP ROBO

MALIGNANT COMMENTS DETECTION

Submitted by:

NITIN SINGH TATRARI

BATCH: 1825

ACKNOWLEDGMENT

I have studied articles and watch videos on multi-label classification problems. I have also studies similar model on kaggle. I have read articles online on cyber bullying and its negative impact and why we should try to stop it.

INTRODUCTION

- ## Business Problem Framing

Online hate, described as abusive language, aggression, cyber bullying, hatefulness and many others has been identified as a major threat on online social media platforms. Research has demonstrated a number of serious consequences of cyber bullying victimization. Victims may have lower self-esteem, increased suicidal ideation, and a variety of emotional responses, including being scared, frustrated, angry, and depressed. Expressing at online platform many times result in hate and conflict, which will lead to many other problems. Thus filtering and removing these hate comments are very important for maintaining the user population.

- ## Conceptual Background of the Domain Problem

If you run a business and put your heart and soul into it, it might be challenging for you to deal with negativity. But, you have to handle it strategically. Otherwise, angry customers will write a bunch of new bad comments to harm your brand.

Young people are connected more than ever before and while this can be a huge benefit in linking them with friends, communities, loved ones and knowledge, it can, of course, be problematic in that they are exposed to an almost constant stream of information which they may not have the critical skills to filter and navigate. Thus, exposure of hate comments, it can create tension, misguide people and can effect psychological of people.

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.

- ## Review of Literature

Hurting sentiments of people through social media platform can leads to adverse effect. As social media spread information faster than in person socialization, the impact is also very large compare in real world. People can go through depression or even develop suicidal thoughts if they can face cyber bullying. Cyber bullying is comparatively faster and easier than bullying in real world.

This comments can turn a person life upside down if it is not controlled, as it is spread like fire in the forest. Anybody can lie on comment and if there is no proper control on it, it will be too late to control this fire.

- ## Motivation for the Problem Undertaken

Social media have become a part of our life and this virtual world impacts our mind and life the same way like in the real world. Thus we have to be alert in this virtual world too just as we are in real world. The cyber bullying like trolling and stalking impact the mental health of the person same like a normal bullying does. Thus any hate or negative comments on social media platform can impact the psychologically of a person and even lead to depression or developing suicidal tendencies in him/her.
Even in terms of businesses also, these hate comments can impact the brand name and influence the judgement of other peoples. Thus, it is very import to detect these hates comments by classifying it so that it does not able to spread hatred.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

1) We have added two columns describing length of comments before and after cleaning.

```
df['length'] = df.comment_text.str.len()
df.head(5)
```

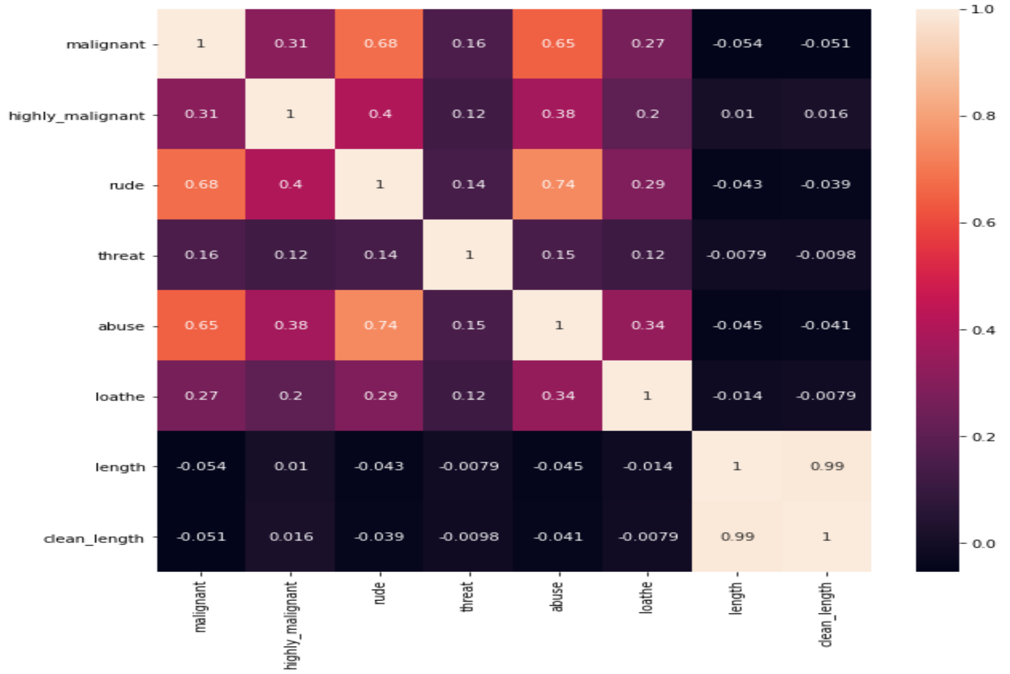
	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67

```
# New column (clean_length) after puncuations,stopwords removal
df['clean_length'] = df.comment_text.str.len()
df.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	158
1	aww matches background colour seemingly stuck ...	0	0	0	0	0	0	112	69
2	hey man really trying edit war guy constantly ...	0	0	0	0	0	0	233	141
3	make real suggestions improvement wondered sec...	0	0	0	0	0	0	622	374

2) Checking correlation

```
#Checking correlation
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



a) There is no relation of length of comments with type of comments

b) Malignant comment is highly correlated with rude comments and abuse comments

c) Threat comments and loathe comments have low correlation with other type of comments.

- Data Sources and their formats

1) We have uploaded the train data set

```
data=pd.read_csv('malignant_train.csv')
data
```

2) We have uploaded the data in data frame df.

```
df=pd.DataFrame(data)
df.head(10)
```

	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
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

There are 8 columns and 159571 rows in the data frame.

3) There are no null values in the data.

```
#Checking null values
df.isnull().sum()
```

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

4) Data type

```
#Checking data type
df.dtypes
```

```
comment_text          object
malignant              int64
highly_malignant       int64
rude                   int64
threat                 int64
abuse                  int64
loathe                 int64
dtype: object
```

Comment_text is string type and rest are integer type variable.

5) Checking ratio

```
#Ratio
print ('malignant ratio = ', (len(df[df['malignant']==1]) / len(df.malignant))*100,'%')
print ('highly_malignant ratio = ', (len(df[df['highly_malignant']==1]) / len(df.highly_malignant))*100,'%')
print ('rude ratio = ', (len(df[df['rude']==1]) / len(df.rude))*100,'%')
print ('threat ratio = ', (len(df[df['threat']==1]) / len(df.threat))*100,'%')
print ('abuse ratio = ', (len(df[df['abuse']==1]) / len(df.abuse))*100,'%')
print ('loathe ratio = ', (len(df[df['loathe']==1]) / len(df.loathe))*100,'%')
```

```
malignant ratio =  9.584448302009765 %
highly_malignant ratio =  0.9995550569965721 %
rude ratio =  5.2948217407925 %
threat ratio =  0.2995531769557125 %
abuse ratio =  4.936360616904074 %
loathe ratio =  0.8804858025581089 %
```

We can see data is imbalance.

- Data Pre-processing Done

1) Converting comments to lower case

```
# Convert all comments to lower case
df['comment_text'] = df['comment_text'].str.lower()
```

2) Removing punctuation

```
# Remove punctuation
df['comment_text'] = df['comment_text'].str.replace(r'[^\w\d\s]', ' ')
```

3) Replacing white spaces between space

```
# Replace whitespace between terms with a single space
df['comment_text'] = df['comment_text'].str.replace(r'\s+', ' ')
```

4) Removing leading and trailing whitespace

```
#Remove leading and trailing whitespace
df['comment_text'] = df['comment_text'].str.replace(r'^\s+|\s+?$', '')
```

5) Replacing email address, URLs, numbers, money symbols & 10 digits phone room with a blank space.

```
# Replace email addresses with ' '
df['comment_text'] = df['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', ' ')
```

```
# Replace URLs with ' '
df['comment_text'] = df['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',' ')
```

```
# Replace numbers with ' '
df['comment_text'] = df['comment_text'].str.replace(r'\d+(\.\d+)?', ' ')
```

```
# Replace money symbols with ' '
df['comment_text'] = df['comment_text'].str.replace(r'£|\$', ' ')
```

```
# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with ' '
df['comment_text'] = df['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',' ')
```

6) Removing stopwords

```
# Remove stopwords
import string
import nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))

df['comment_text'] = df['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))
```

7) Applying Lemmatization

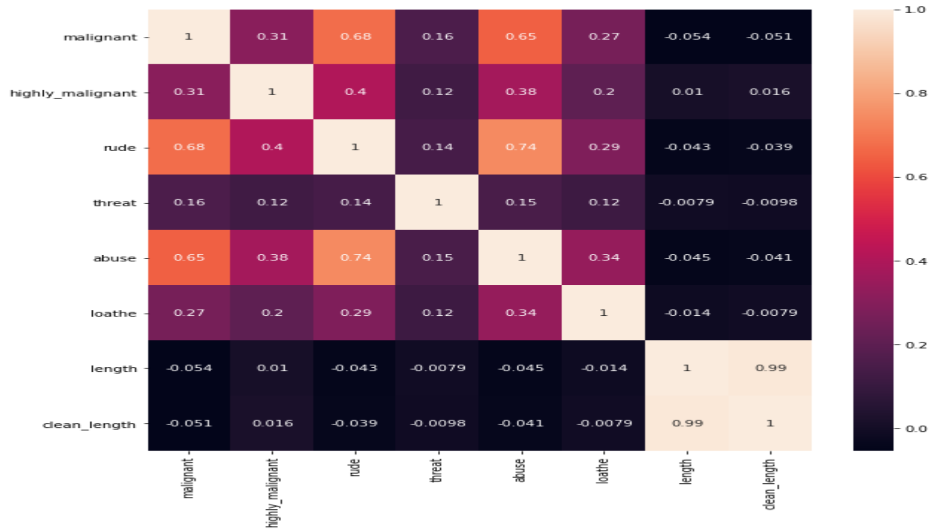
```
from nltk.stem import WordNetLemmatizer
```

```
lem=WordNetLemmatizer()
df['comment_text'] = df['comment_text'].apply(lambda x: ' '.join(lem.lemmatize(t) for t in x.split()))
```

- Data Inputs- Logic- Output Relationships

1) Correlation

```
#Checking correlation
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



a) There is no relation of length of comments with type of comments
b) Malignant comment is highly correlated with rude comments and abuse comments
c) Threat comments and loathe comments have low correlation with other type of comments.

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

No assumptions were considered.

- Hardware and Software Requirements and Tools Used

The libraries used are: pandas, numpy, matplotlib.pyplot, seaborn, scikit_multilearn and scikit_learn. The laptop used is with Intel I5 10th generation, 4GB RAM, 4GB GPU.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

1) I have cleaned the data by removing punctuation, whitespace, numbers, emails, URLs, phone number and stop words.
2) Then I have changed the comments into vector form using TF-DIF vectorizer.
3) I have transformed the target variables from multi-label to multi-class target.

- Testing of Identified Approaches (Algorithms)

The algorithms used for testing are as follows:-

1) Term Frequency Inverse Document Frequency Vectorizer(TF-IDF)
2) Multinomial Naïve Bayes
3) Gaussian Naïve Bayes
4) Decision Tree classifier
5) Random Forest classifier
6) Ada Boost Classifier
7) Binary Relevance
8) Classifier Chain

- Run and Evaluate selected models

1) We convert the text in comment_text into vectors using TF-IDF vectorizer.

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

```
tf_vec = TfidfVectorizer(max_features=1000)
features = tf_vec.fit_transform(df.comment_text).toarray()
x = features
y = df[['malignant', 'highly_malignant', 'rude', 'threat','abuse', 'loathe']]
```

```
tf_vec
```

```
TfidfVectorizer(max_features=1000)
```

2) We split the data into train and test.

```
# Split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=42)
```

3) We build a function to convert multi-label target variable to multi-class variable. Then apply different algorithm on it.

```
def build_model(model,estimator,x_train,y_train,x_test,y_test):
    clf=estimator(model)
    clf.fit(x_train,y_train)
    clf_pred=clf.predict(x_test)
    acc=accuracy_score(y_test,clf_pred)
    ll=log_loss(y_test,clf_pred.toarray())
    result= {'accuracy':acc,'log_loss':ll}
    print(classification_report(y_test,clf_pred))
    return result
```

Then we apply the above with different algorithm.

a) MultinomialNB with Binary Relevance

```
B_r_mnb=build_model(MultinomialNB(),BinaryRelevance,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

           0       0.95      0.46      0.62      4582
           1       0.52      0.29      0.37       486
           2       0.92      0.52      0.67      2556
           3       0.40      0.01      0.03       136
           4       0.81      0.43      0.56      2389
           5       0.50      0.12      0.19       432

   micro avg       0.88      0.44      0.59     10581
   macro avg       0.69      0.31      0.41     10581
weighted avg       0.87      0.44      0.58     10581
 samples avg       0.04      0.03      0.04     10581
```

```
B_r_mnb
```

```
{'accuracy': 0.9131016042780749, 'log_loss': 1.3429786840782607}
```

b) GaussianNB with Binary Relevance

```
B_r_gnb=build_model(GaussianNB(),BinaryRelevance,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

           0       0.27      0.86      0.41      4582
           1       0.03      0.84      0.07       486
           2       0.17      0.89      0.29      2556
           3       0.01      0.75      0.02       136
           4       0.15      0.85      0.25      2389
           5       0.02      0.80      0.05       432

   micro avg       0.12      0.86      0.21     10581
   macro avg       0.11      0.83      0.18     10581
weighted avg       0.20      0.86      0.31     10581
 samples avg       0.04      0.08      0.05     10581
```

```
B_r_gnb
```

```
{'accuracy': 0.5522852606951871, 'log_loss': 0.8899959038056307}
```

c) GaussianNB with Classifier Chain

```
chain_model_Gau= build_model(GaussianNB(),ClassifierChain,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

           0       0.27      0.86      0.41      4582
           1       0.04      0.84      0.08       486
           2       0.17      0.89      0.28      2556
           3       0.01      0.75      0.02       136
           4       0.14      0.87      0.24      2389
           5       0.02      0.80      0.05       432

   micro avg       0.12      0.86      0.21     10581
   macro avg       0.11      0.83      0.18     10581
weighted avg       0.19      0.86      0.31     10581
 samples avg       0.04      0.08      0.05     10581
```

```
chain_model_Gau
```

```
{'accuracy': 0.5873997326203209, 'log_loss': 0.8212934895676468}
```

d) MultinomialNB with Classifier Chain

```
chain_model_Multi= build_model(MultinomialNB(),ClassifierChain,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

           0       0.95      0.46      0.62      4582
           1       0.35      0.63      0.45       486
           2       0.79      0.67      0.72      2556
           3       0.07      0.45      0.12       136
           4       0.69      0.62      0.65      2389
           5       0.14      0.67      0.23       432

   micro avg       0.57      0.56      0.57     10581
   macro avg       0.50      0.58      0.47     10581
weighted avg       0.78      0.56      0.62     10581
 samples avg       0.03      0.04      0.03     10581
```

```
chain_model_Multi
```

```
{'accuracy': 0.8999623997326203, 'log_loss': 0.496213520208631}
```

e) Decision Tree Classifier with Classifier Chain

```
chain_model_DTC= build_model(DecisionTreeClassifier(),ClassifierChain,x_train,y_train,x_test,y_test)
```

```
              precision    recall  f1-score   support

           0       0.66      0.58      0.62      4582
           1       0.26      0.16      0.20       486
           2       0.74      0.67      0.70      2556
           3       0.24      0.22      0.23       136
           4       0.59      0.52      0.55      2389
           5       0.41      0.31      0.35       432

   micro avg       0.64      0.55      0.59     10581
   macro avg       0.48      0.41      0.44     10581
weighted avg       0.63      0.55      0.59     10581
 samples avg       0.05      0.05      0.05     10581
```

```
chain_model_DTC
```

```
{'accuracy': 0.8891836564171123, 'log_loss': 1.4221770046756532}
```

f) Logistic Regression with classifier Chain

```
1 clf_rf=ClassifierChain(RandomForestClassifier())
```

```
1 chain_model_RF= build_model(RandomForestClassifier(),ClassifierChain,x_train,y_train,x_test,y_test)
```

	precision	recall	f1-score	support
0	0.86	0.60	0.71	4582
1	0.47	0.09	0.15	486
2	0.84	0.70	0.77	2556
3	0.58	0.08	0.14	136
4	0.70	0.59	0.64	2389
5	0.72	0.18	0.29	432
micro avg	0.80	0.57	0.67	10581
macro avg	0.69	0.37	0.45	10581
weighted avg	0.79	0.57	0.66	10581
samples avg	0.05	0.05	0.05	10581

```
1 chain_model_RF
```

{'accuracy': 0.9150651737967914, 'log_loss': 1.3149543448086285}

g) Random Forest classifier with Classifier Chain

```
1 chain_model_RF= build_model(RandomForestClassifier(),ClassifierChain,x_train,y_train,x_test,y_test)
```

	precision	recall	f1-score	support
0	0.86	0.60	0.71	4582
1	0.47	0.09	0.15	486
2	0.84	0.70	0.77	2556
3	0.58	0.08	0.14	136
4	0.70	0.59	0.64	2389
5	0.72	0.18	0.29	432
micro avg	0.80	0.57	0.67	10581
macro avg	0.69	0.37	0.45	10581
weighted avg	0.79	0.57	0.66	10581
samples avg	0.05	0.05	0.05	10581

```
1 chain_model_RF
```

{'accuracy': 0.9150651737967914, 'log_loss': 1.3149543448086285}

h) Ada Boost classifier with classifier chain

```
1 chain_model_AB= build_model(AdaBoostClassifier(),ClassifierChain,x_train,y_train,x_test,y_test)
```

	precision	recall	f1-score	support
0	0.87	0.53	0.66	4582
1	0.53	0.22	0.31	486
2	0.85	0.68	0.76	2556
3	0.42	0.24	0.30	136
4	0.68	0.61	0.64	2389
5	0.53	0.24	0.33	432
micro avg	0.79	0.55	0.65	10581
macro avg	0.65	0.42	0.50	10581
weighted avg	0.79	0.55	0.64	10581
samples avg	0.04	0.05	0.04	10581

```
1 chain_model_AB
```

{'accuracy': 0.913519385026738, 'log_loss': 1.0129685036135692}

From above we can see except GaussianNB , all other algorithm are giving accuracy above 88%.

But the lowest Log loss is given by MultinomialNB with classifer chain , that is, 0.49 and give F1 score of 57%.

4) HYPERPARAMETER TUNNING

```
from sklearn.model_selection import GridSearchCV
parameters = [{'classifier':[MultinomialNB()],'classifier__alpha': [1, 0.1, 0.01, 0.001, 0.0001, 0.00001]}]
Grid=GridSearchCV(ClassifierChain(),parameters,scoring='accuracy')
Grid.fit(x_train,y_train)
```

```
GridSearchCV(estimator=ClassifierChain(require_dense=[True, True]),
             param_grid=[{'classifier': [MultinomialNB(alpha=1e-05)],
                          'classifier__alpha': [1, 0.1, 0.01, 0.001, 0.0001,
                                                1e-05]}],
             scoring='accuracy')
```

```
print (Grid.best_params_, Grid.best_score_)
```

```
{'classifier': MultinomialNB(alpha=1e-05), 'classifier__alpha': 1e-05} 0.9002408159802423
```

Thus, at alpha=0.00001, algorithm will give the best accuracy. Thus applying it

```
clf=ClassifierChain(MultinomialNB(alpha=0.00001))
clf.fit(x_train,y_train)
clf_pred=clf.predict(x_test)
acc=accuracy_score(y_test,clf_pred)
ll=log_loss(y_test,clf_pred.toarray())
print({'accuracy':acc,'log_loss':ll})
```

```
{'accuracy': 0.9009024064171123, 'log_loss': 0.5027187428663634}
```

```
print(classification_report(y_test,clf_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.46      0.62      4582
           1       0.31      0.64      0.42       486
           2       0.79      0.67      0.73      2556
           3       0.07      0.52      0.12       136
           4       0.68      0.62      0.65      2389
           5       0.14      0.65      0.23       432

   micro avg       0.57      0.57      0.57     10581
   macro avg       0.49      0.59      0.46     10581
weighted avg       0.78      0.57      0.62     10581
 samples avg       0.03      0.04      0.03     10581
```

- Key Metrics for success in solving problem under consideration

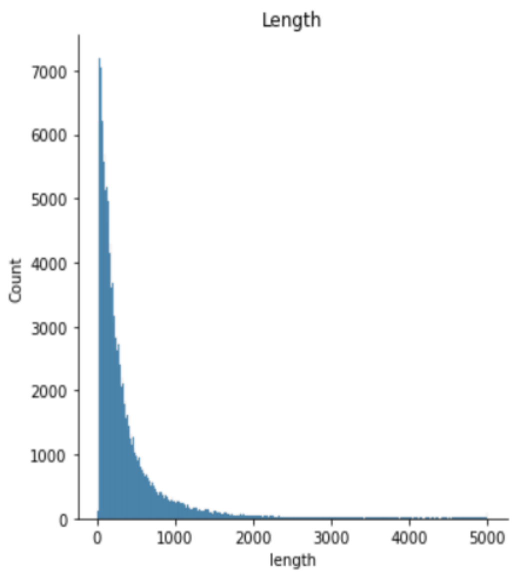
The metrics used are accuracy_score, classification_report and Log_loss.
Above it is shown.

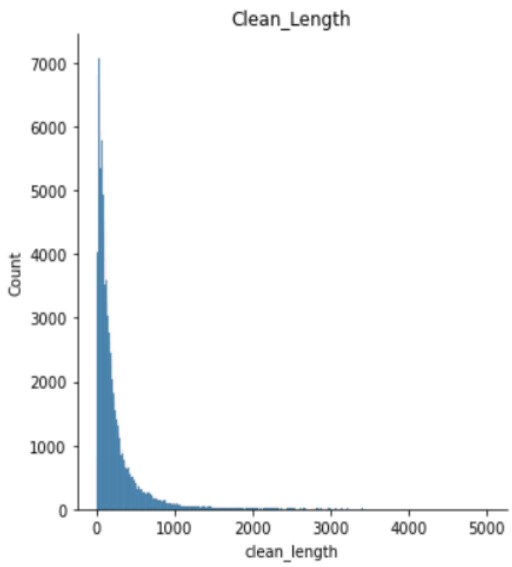
- Visualizations

A) Plotting graphs

```
sns.displot(x='length',data=df)
plt.title('Length')
plt.show()
```

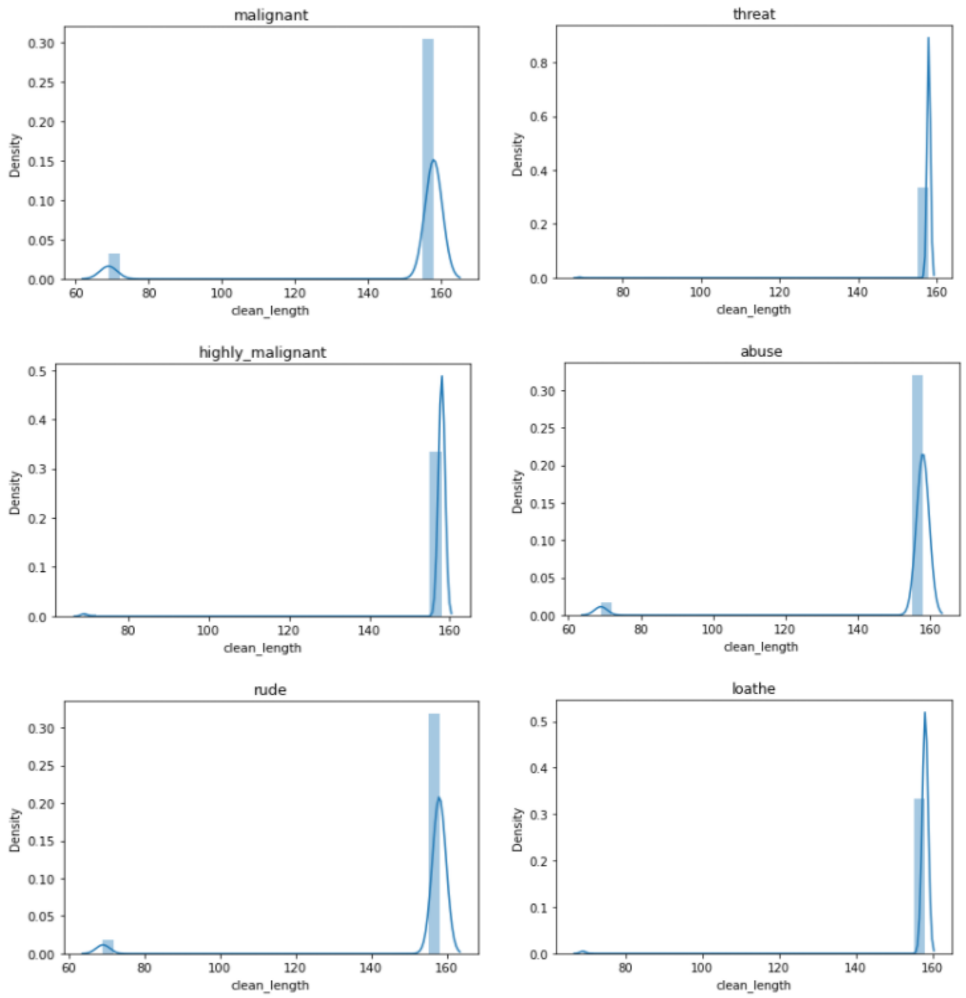
```
sns.displot(x='clean_length',data=df)
plt.title('Clean_Length')
plt.show()
```





Thus, we can see length of comment_text reduces after cleaning.

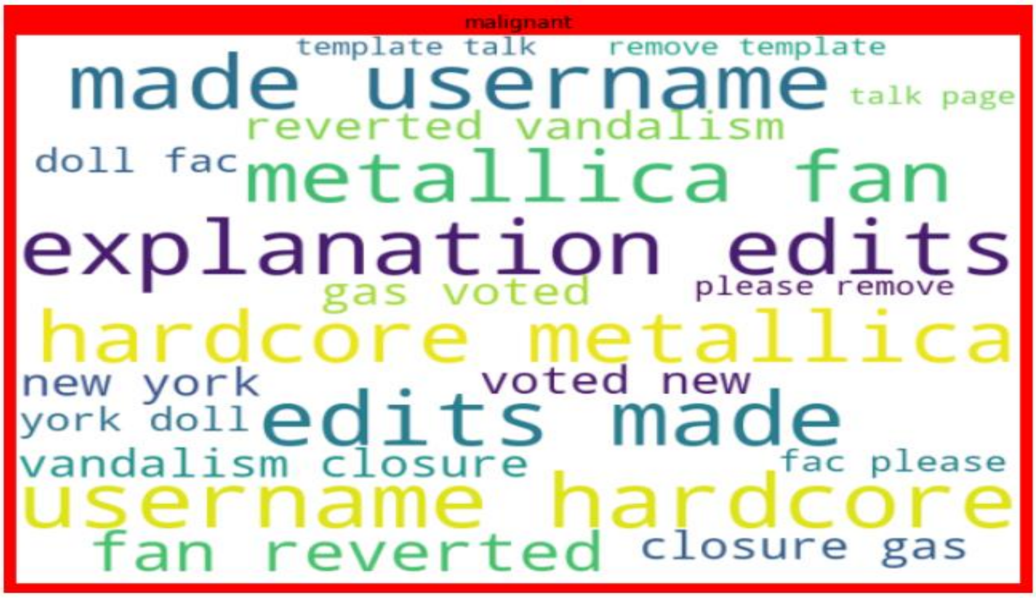
```
for i in ['malignant','highly_malignant','rude','threat','abuse','loathe']:
    sns.distplot(df['clean_length'][df[i]],bins=30)
    plt.title(i)
    plt.show()
```



Mostly have shows a high density peak around 160.

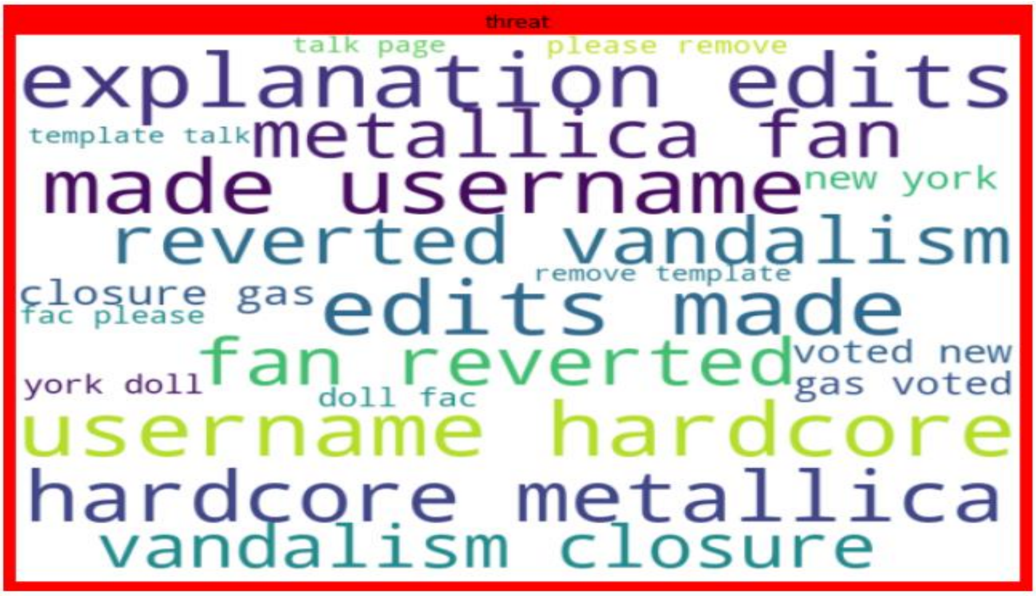
```
from wordcloud import WordCloud
```

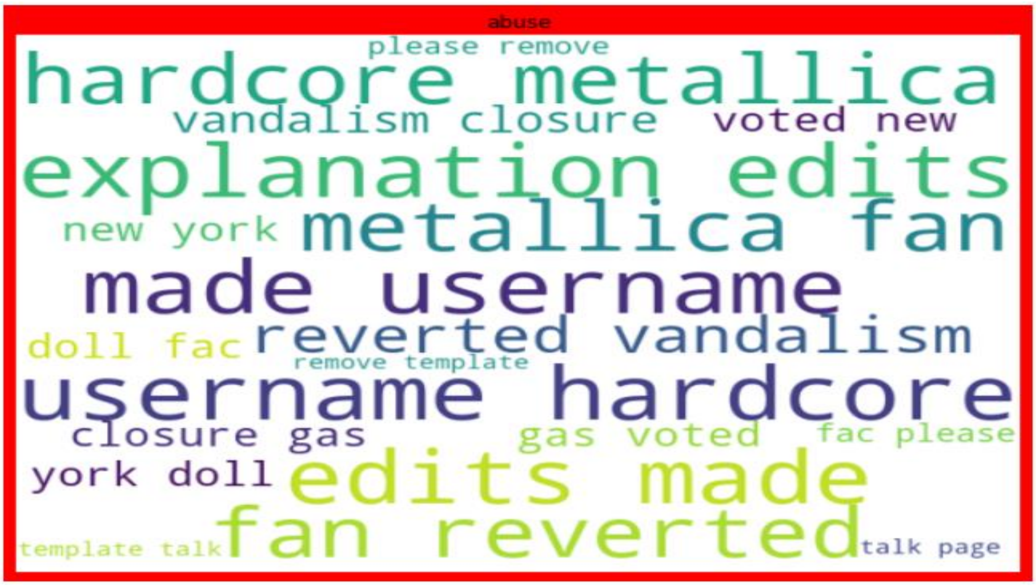
```
for i in ['malignant','highly_malignant','rude','threat','abuse','loathe']:
    rev = df['comment_text'][df[i]]

    rev_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(rev))

    plt.figure(figsize=(10,8),facecolor='r')
    plt.imshow(rev_cloud)
    plt.axis('off')
    plt.title(i)
    plt.show()
```

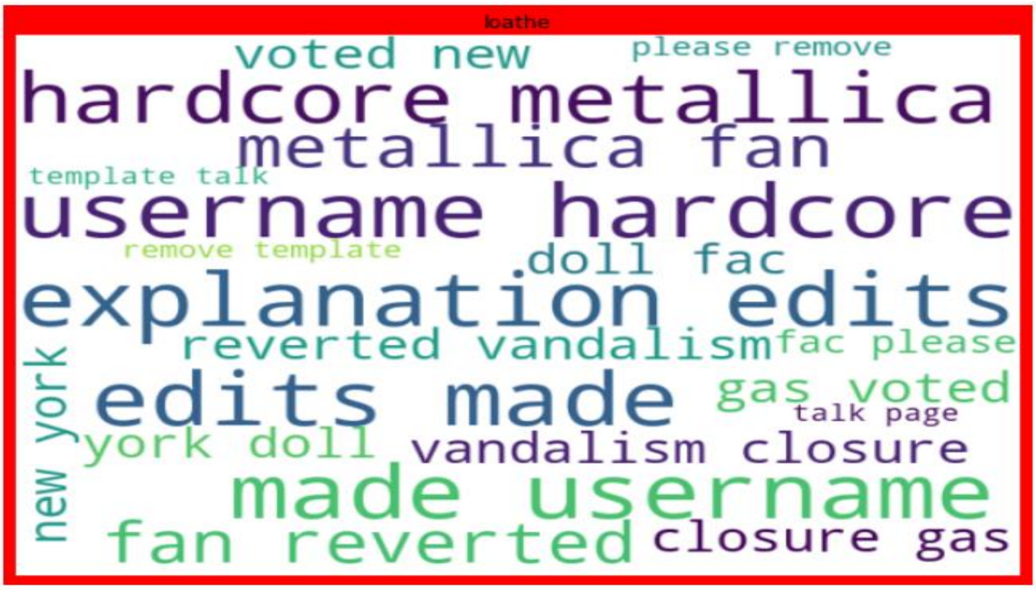

malignant
template talk
remove template
made username
talk page
reverted vandalism
doll fac
metallica fan
explanation edits
gas voted
please remove
hardcore metallica
new york
voted new
york doll
edits made
vandalism closure
fac please
username hardcore
fan reverted
closure gas


highly_malignant
username hardcore
reverted vandalism
voted new
explanation edits
edits made
gas voted
doll fac
closure gas
new york
metallica fan
made username
fan reverted
york doll
talk page
vandalism closure
fac please
hardcore metallica
template talk
please remove
remove template


rude
hardcore metallica
please remove
remove template
made username
doll fac
reverted vandalism
username hardcore
template talk
metallica fan
york doll
gas voted
fac please
new york
edits made
voted new
vandalism closure
talk page
explanation edits
closure gas
fan reverted


threat
talk page
please remove
explanation edits
template talk
metallica fan
made username
new york
reverted vandalism
remove template
closure gas
fac please
edits made
voted new
york doll
fan reverted
gas voted
doll fac
username hardcore
hardcore metallica
vandalism closure


abuse
please remove
hardcore metallica
vandalism closure
voted new
explanation edits
new york
metallica fan
made username
doll fac
reverted vandalism
remove template
username hardcore
closure gas
gas voted
fac please
york doll
edits made
template talk
fan reverted
talk page


loathe
voted new
please remove
hardcore metallica
metallica fan
template talk
username hardcore
remove template
doll fac
explanation edits
reverted vandalism
fac please
new york
edits made
gas voted
talk page
york doll
vandalism closure
made username
fan reverted
closure gas

- Interpretation of the Results

1) The length of text was reduced considerably after applying all pre-processing steps.
2) The target variables have correlation with each other but not with the length of text.

CONCLUSION

- Key Findings and Conclusions of the Study

1) There were punctuations, white spaces, numbers, URLs, phone numbers, email ids which were removed to decrease the length.
2) Malignant comment is highly correlated with rude comments and abuse comments
3) Threat comments and loathe comments have low correlation with other type of comments.
4) Length of comments has no relationship with it classification.

- Learning Outcomes of the Study in respect of Data Science

First of all cleaning the data was very important to removes all punctuation, whitespaces, stop words, URls, email ids, etc which have no impact on classification. Then I have lemmatized the text.

Then I have converted the text into vector form for machine learning. I have to convert mutli-label variable into multi-class variable to be prepared for model making.

I have train the model with algorithm and find the best models with the least log loss.

- Limitations of this work and Scope for Future Work

The data was imbalanced, so better result can be contains.

People also upload some pictures with their comments, thus by neural network we can classify on these images too.