FLIP ROBO

Malignant Comments Classification

Submitted by:

Shama Tanweer

Internship-12

ACKNOWLEDGMENT

I would like to express my sincere gratitude to FlipRobo Technologies for supporting me throughout the internship and giving me the opportunity to explore the depth of Data Science by providing multiple projects like this.I have taken help from following sites whenever stuck

Towarsdatascience.com

Datacamp.com

INTRODUCTION

- ### Business Problem Framing

  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 objective 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

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. 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.

Researchers have applied various machine learning systems to try to tackle the problem of toxicit., There are various factors affecting the comments and can be related to the emotion, figure of speeches, and sometimes sarcasms which is related to indirect taunting is also rude and as a human it'ssometimes hard for us to distinguish between sarcasms and real appreciation.

- Motivation for the Problem Undertaken

  This project was highly motivated as it includes the real time problem of analysing toxic behaviour and providing us opportunity to build a model which can classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

• Identification of possible problem-solving approaches (methods)

Used the following process for problem solving

1. Data Preprocessing
2. Building word dictionary
3. Feature extraction
4. Training classifiers
5. Testing
6. Performance evaluation using multiple metrics

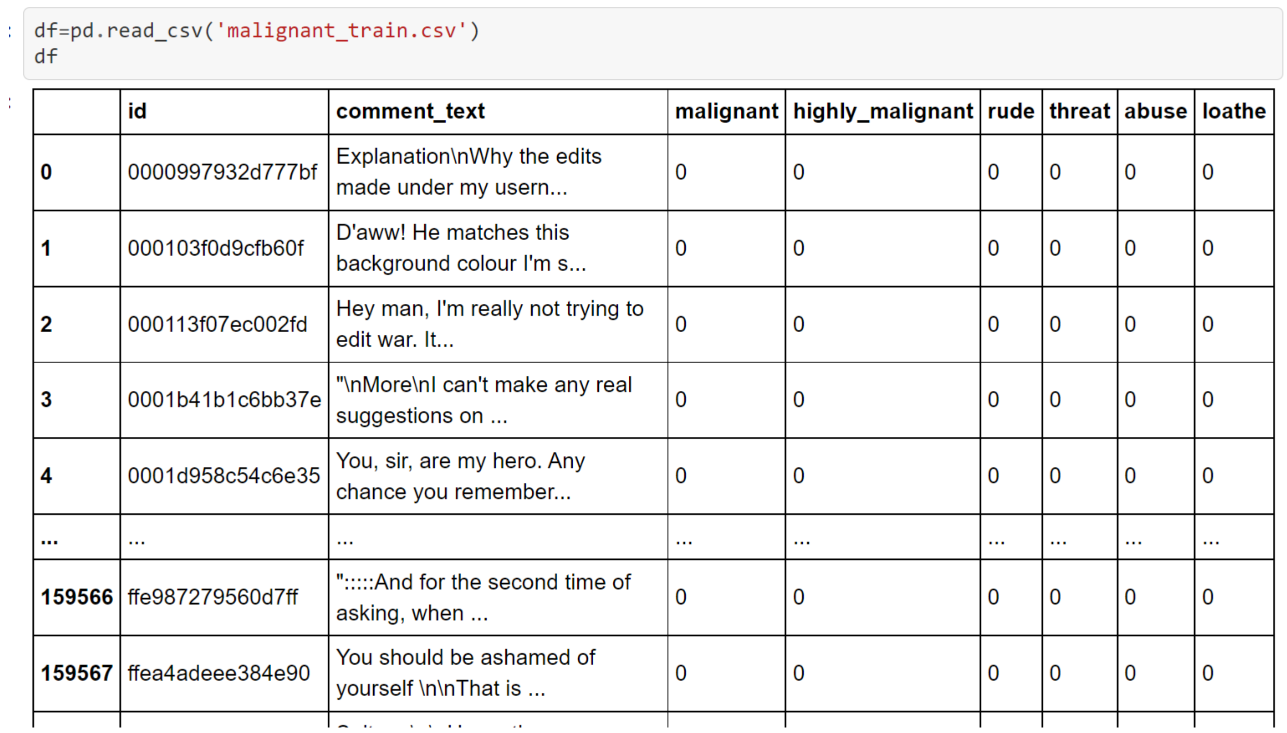
Analytical Problem Framing

Mathematical/Analytical modelling of problem

- 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 attributes were of int type,except id & comment_text(object type)
- Dataset did’nt contain any null value.

Data source and their formats

The data is provide to us from our clint database.The sample data is in .csv format
.The sample data for reference is shown below.



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

	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
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0

DataSet description

- Train dataset samples contain 8 features 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 data set includes:

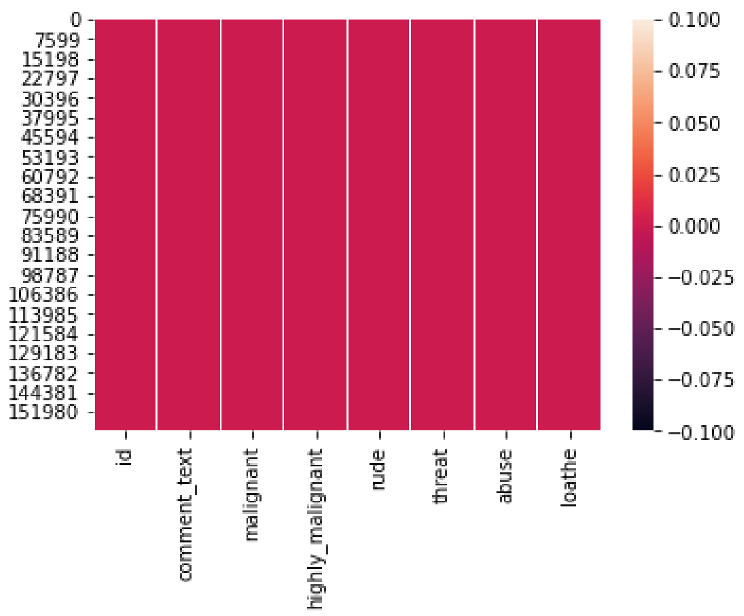
  - **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
  - **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
  - **Rude:** It denotes comments that are very rude and offensive.
  - **Threat:** It contains indication 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.

- Data Pre-processing

  Data usually comes from a variety of source & is often inconsistent,inaccurate.Data preprocessing helps to enhance the quality of dat and make it ready for various ML model.We have applied various methods for data preprocessing methods in this project .

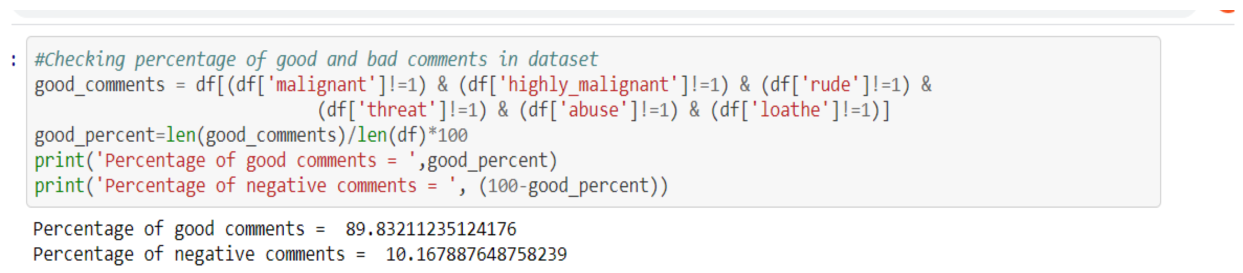
- ➢ First we check shape by using(df.shape)
- ➢ Then checked datatype of various features & found that all features are of int type except ID & comment_text which are of object datatype
- ➢ checking for null values in each column:

```
sns.heatmap(df.isnull())
```



The above heatmap shows that no null values are pesent in the train dataset.

- ➢ After that we check the summary statistics of our dataset bu using df.describe(). This part tells about the statistics of our dataset i.e. mean, median, max value ,min values and also it tell whether outliers are present in our dataset or not. Since values are either 0 or 1,so no outliers are present
- ➢ Checking distribution of positive and negative comments.



```
#Checking percentage of good and bad comments in dataset
good_comments = df[(df['malignant']!=1) & (df['highly_malignant']!=1) & (df['rude']!=1) &
                   (df['threat']!=1) & (df['abuse']!=1) & (df['loathe']!=1)]
good_percent=len(good_comments)/len(df)*100
print('Percentage of good comments = ',good_percent)
print('Percentage of negative comments = ', (100-good_percent))
```

```
Percentage of good comments =  89.83211235124176
Percentage of negative comments =  10.167887648758239
```

We see that around 10% comments are negative,which belong to different category.

- Cleaning the raw data-It involves deletion of words or special characters that do not add meaning to the text.
- Important cleaning steps are:
    1. Lowering case
    2. Handling of special characters
    3. Removal of stopwords
    4. Handling of hyperlinks
    5. Removing leading and trailing white space
    6. Replacing urls with web address
    7. Converted words to most suitable base form by using lemmatization

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

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

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

    # Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
df['comment_text']=df['comment_text'].str.replace(r'£|\$', 'dollers')

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


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

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

    #REPLACING SPECIAL CHARACTERS  BY WHITE SPACE
df['comment_text']=df['comment_text'].str.replace(r"[^a-zA-Z0-9]+", " ")

     #REPLACING one or 2  character length word   BY WHITE SPACE
df['comment_text']=df['comment_text'].str.replace(r'\b[a-zA-Z]{1,2}\b', '')
```

```python
import string
punct=string.punctuation
```

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

stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])

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

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

```python
from nltk.tokenize import RegexpTokenizer
tokenizer=RegexpTokenizer(r'\w+')
df['comment_text'] = df['comment_text'].apply(lambda x: tokenizer.tokenize(x.lower()))
```

```python
# writing function for the entire dataset
# Lemmatizing to get root words and further reducing characters

from nltk.stem import SnowballStemmer, WordNetLemmatizer
stemmer = SnowballStemmer("english")
import gensim
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text,pos='v'))

#Tokenize and Lemmatize
def preprocess(text):
    result=[]
    for token in text:
        if len(token)>=3:
            result.append(lemmatize_stemming(token))

    return result
```

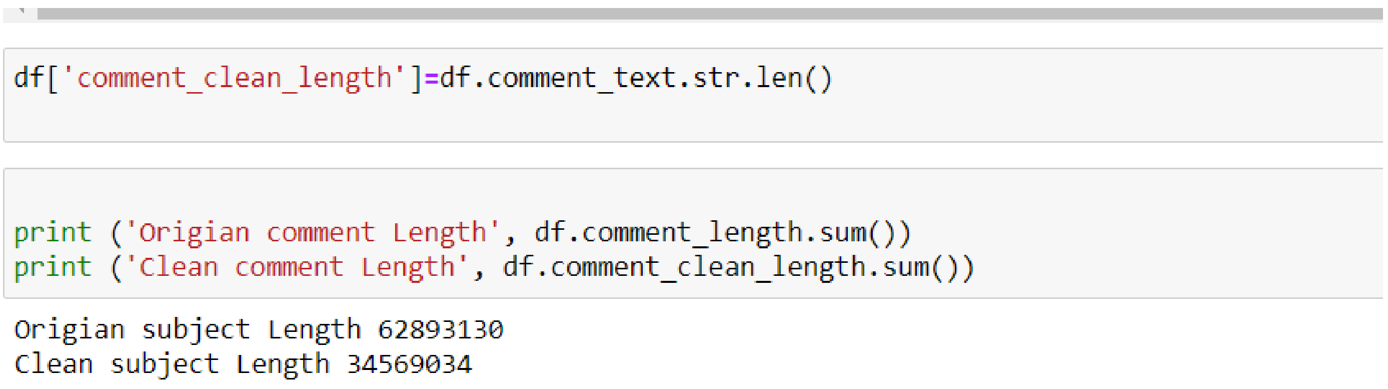
```python
# Processing comment_text with above Function
processed_comment = []

for doc in df.comment_text:
    processed_comment.append(preprocess(doc))

print(len(processed_comment))
processed_comment[:3]
```

Adding additional attribute :

To compare the length of comment_text before preprocessing and after preprocessing an addition column was added:

Adding new column "Comment_length" which is the length of comment in number.

Adding one more column "comment_clean_length" which is count of length of comment column by mapping



```
df['comment_clean_length']=df.comment_text.str.len()
```

```
print ('Origian comment Length', df.comment_length.sum())
print ('Clean comment Length', df.comment_clean_length.sum())
```

```
Origian subject Length 62893130
Clean subject Length 34569034
```

After executing all these steps it was found that all the words & special characters were removed from the dataset which were of no use and consuming memory

- Data Inputs- Logic- Output Relationships

  For this data's input and output logic we will analyse words frequency for each label, so that we can get the most frequent words that were used in different features.

Hardware and Software Requirements and Tools Used:

• Framework-Annconda

• IDE-Jupyter –Notebook

• Coding Language-Python

• Hardware used: system memory 8GB,

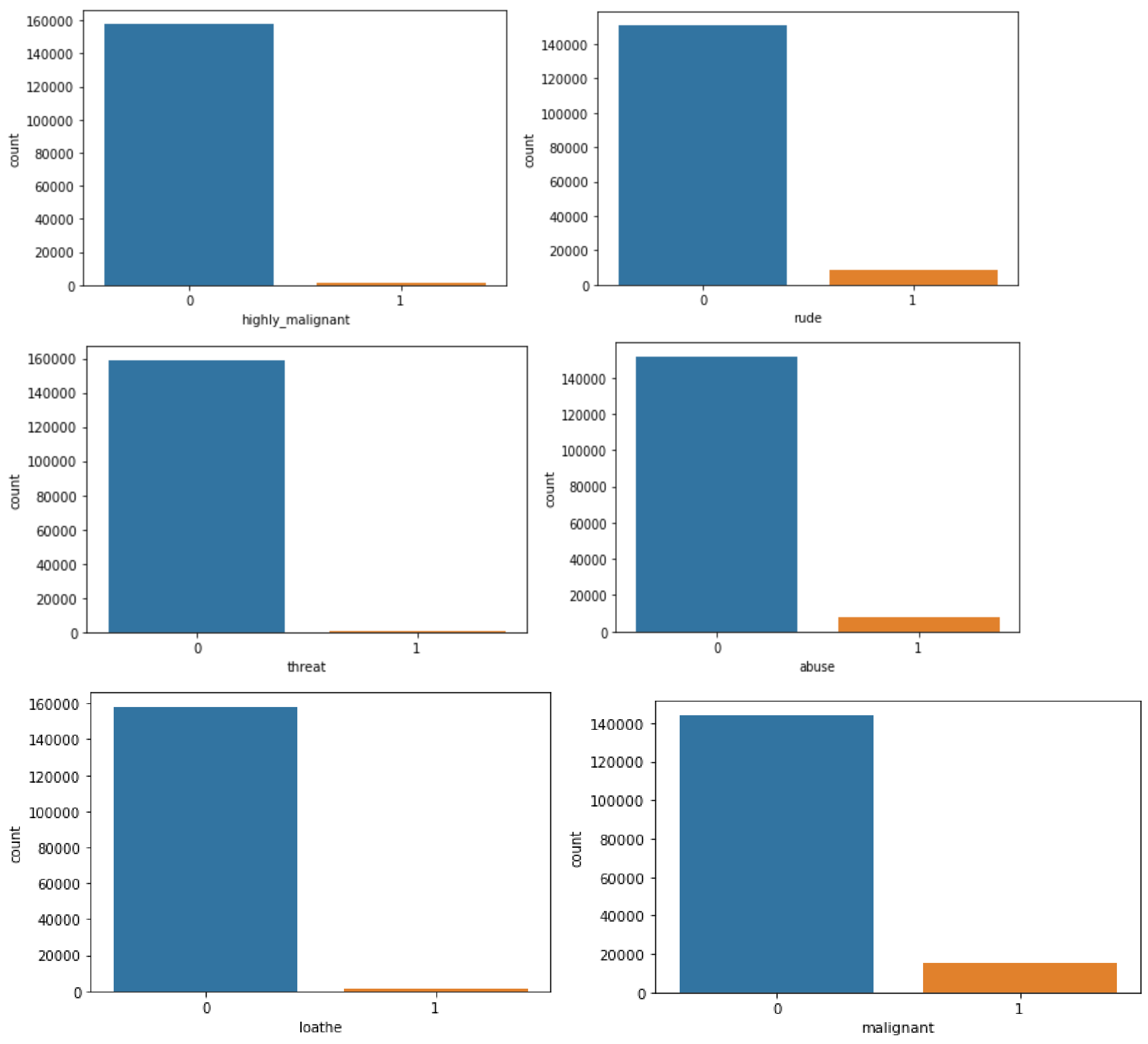
• Processor: core i3

Libraries used: Below are the python library used in this project. •

- ➢ Pandas: To read the Data file in form of data.
- ➢ Matplotlib: This library is typically used to plot the figures for better visualisation of data.
- ➢ Seaborn: A advanced version of Matplotlib
- ➢ Scikit Learn: This is the most important library for Machine Learning since it contains various Machine Learning Algorithms which are used in this project. Scikit Learn also contains Preprocessing library which is used in data preprocessing. Apart from this it contains very useful joblib library for serialization purpose using which final model have been saved in this project.
- ➢ NLTK: Natural language took kit is one of the most used library for building NLP projects.

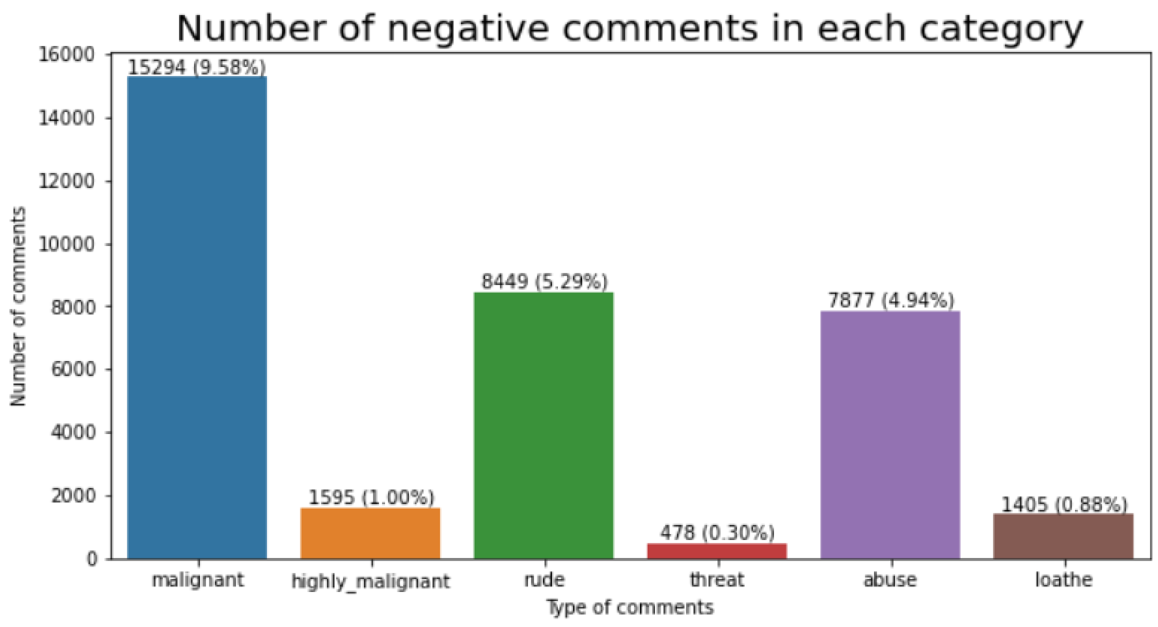
Visualizations

To better analyse the distribution of comments in all categories I have used count plot as below showing distribution of comments belonging to that each label:



We observe that the data is unbalanced.

Again checking percentage of negative comments in each category:



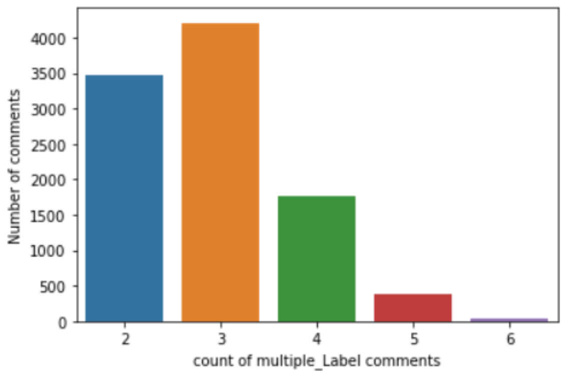
From above we observe that:

- ➢ Highest unwanted comments were of malignant type followed by rude and abuse.
- ➢ Threat comments were least in number with only 0.30%

Some of the comments belonged to more than one category i.e comments belong to different label .To check this I have used below plot

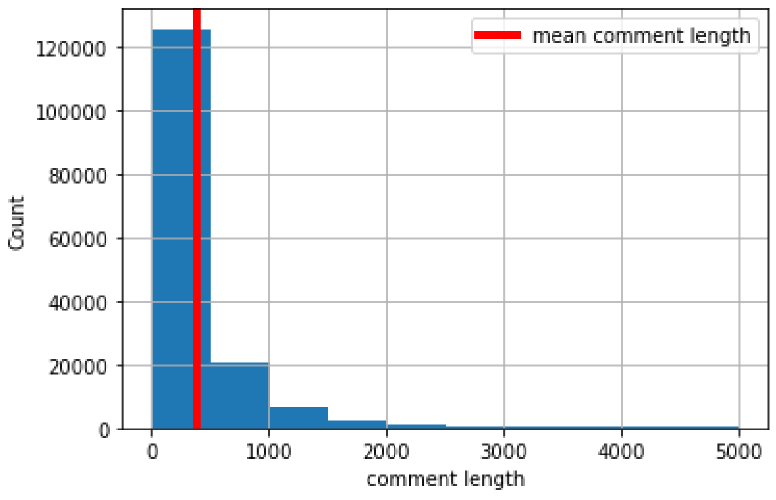
```
sns.countplot(x='label_sum',data=df[df['label_sum']>1])
plt.xlabel('count of multiple_Label comments', fontsize=10)
plt.ylabel('Number of comments',fontsize=10)
df['label_sum'].value_counts()/159571*100
```

```
0    89.832112
1     3.985687
3     2.637697
2     2.180847
4     1.102957
5     0.241272
6     0.019427
Name: label_sum, dtype: float64
```



- 3.98% comments belongs to only one label
- 0.01% comments belong to all the labels.
- 2.63% comments belongs to 3 different labels

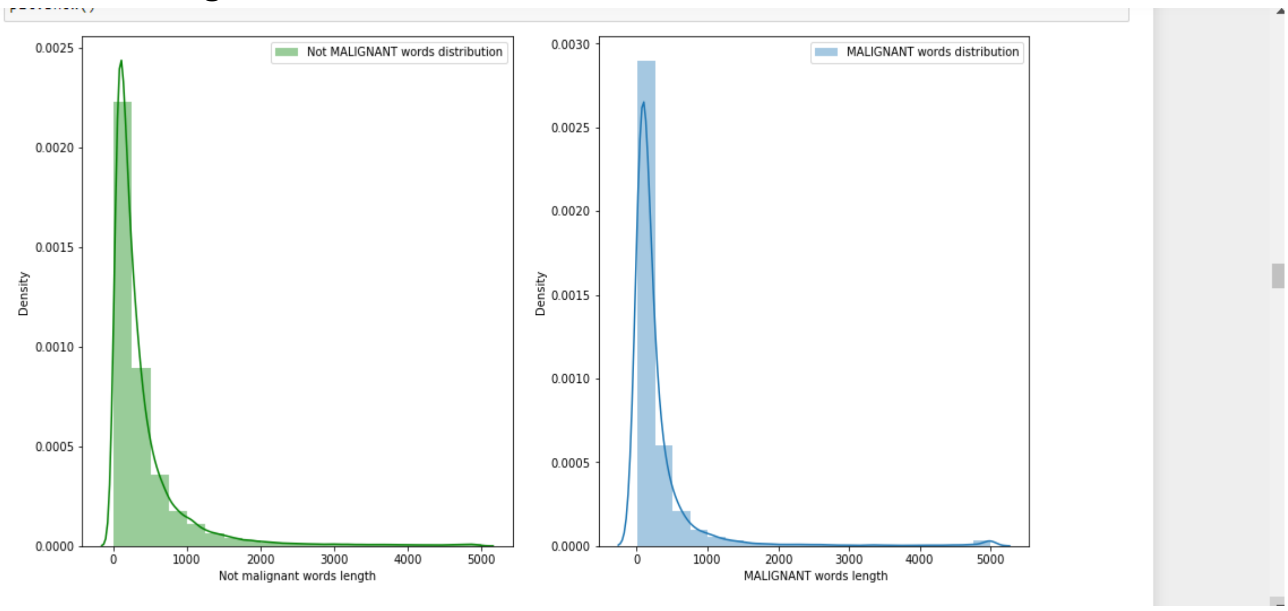
- ➢ Checking average comment length before cleaning



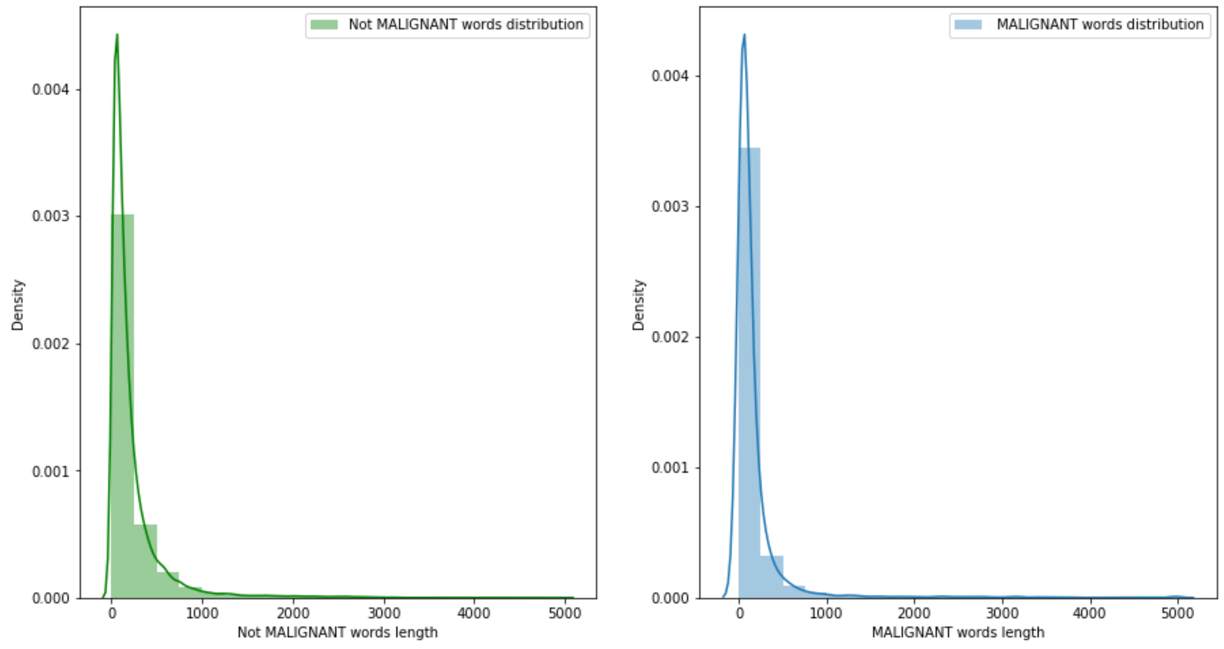
- In the train dataset, the average length of comments are 400, & we also notice that the frequency reduces as the number of characters increase

Then we have plotted graph to show distribution of word count before cleaning and after cleaning

Before cleaning:

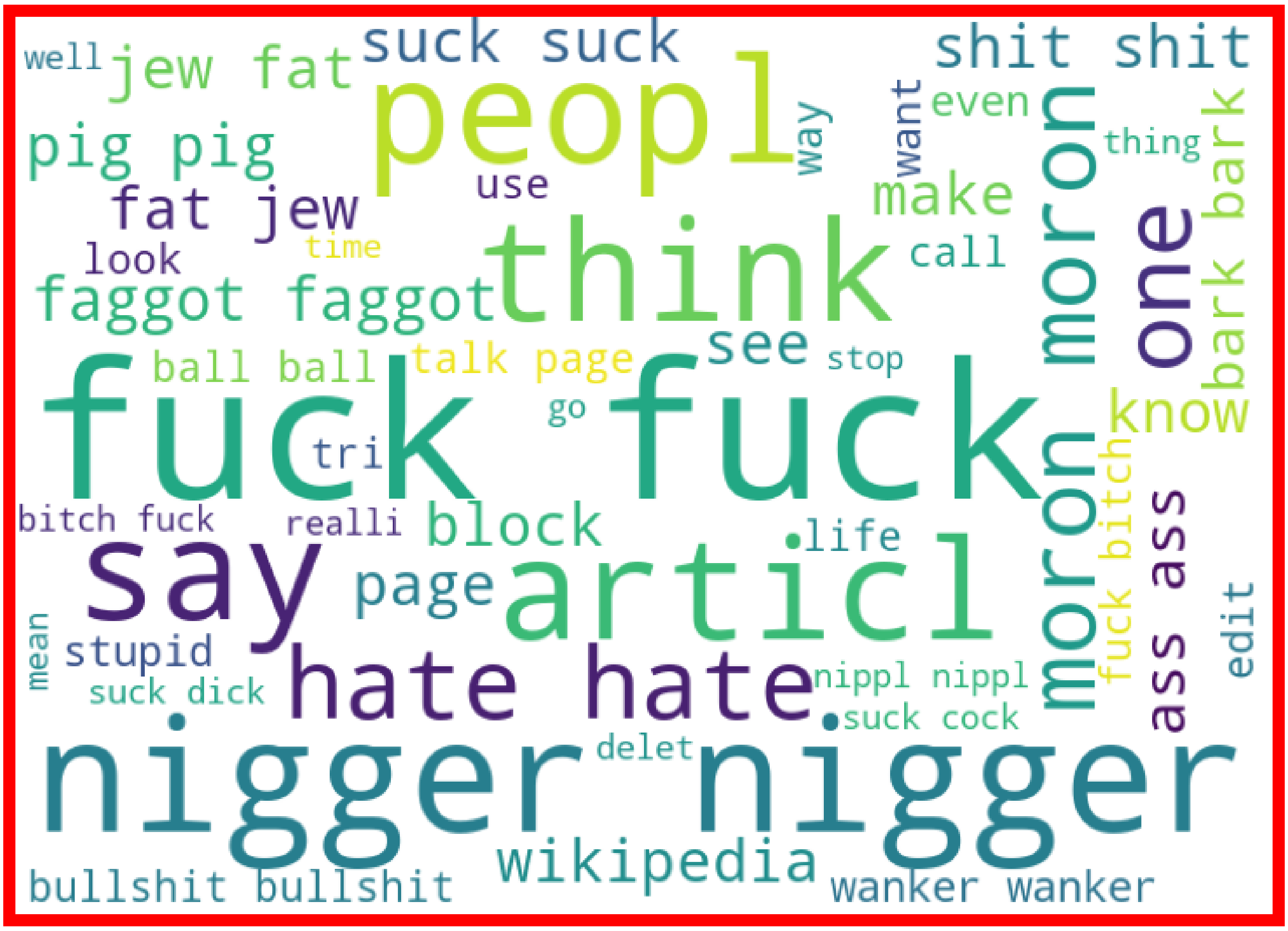


After cleaning:

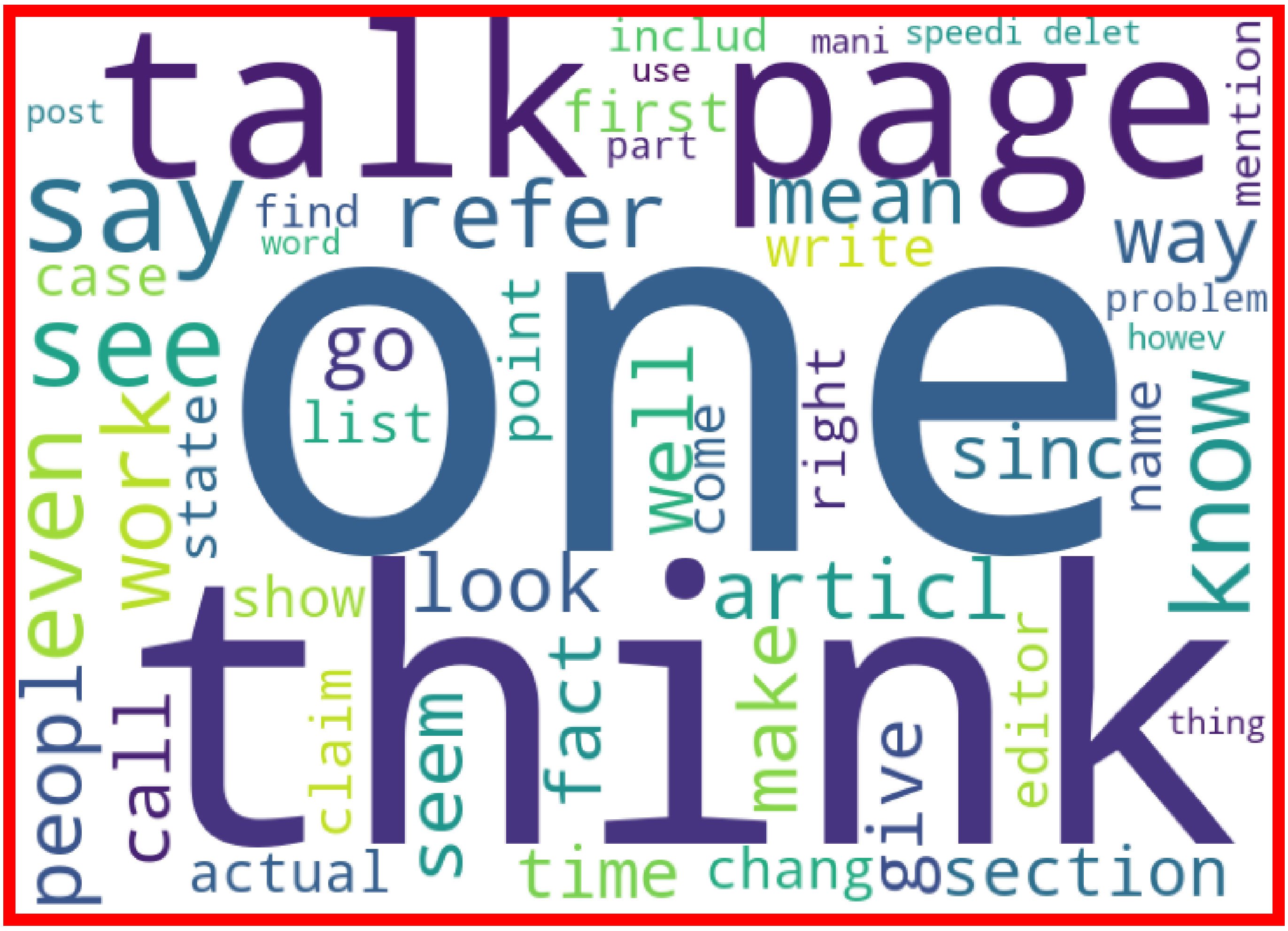


To get better view of words contained in comments . A word dictionary (wordcloud) was made showing the first 50 words highly occurred words in positive & negative comments.

```python
#Getting sense of loud words in negative comment
from wordcloud import WordCloud

spams = df['comment_text'][df['comment_label']==1]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=50).generate(' '.join(spams))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



```python
]: #Getting sense of loud words in neutral/positive comment
   from wordcloud import WordCloud

   spams = df['comment_text'][df['comment_label']==0]

   spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=50).generate(' '.join(spams))

   plt.figure(figsize=(10,8),facecolor='r')
   plt.imshow(spam_cloud)
   plt.axis('off')
   plt.tight_layout(pad=0)
   plt.show()
```



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

Only assumptions which were taken related to the problem was that we dropped the id column as it was not of much importance

Model/s Development and Evaluation

- problem-solving approaches:

Understanding the problem is the first crucial steps in solving any problem. From the given dataset it can be concluded that it is a MultiLabel Classification problem.Therefore I run my preprocessed data on 6 classification algorithm.

Training Classifier:

We converted all the comment text into vectors , using TF-IDF. Then we have split features and label.

```
# Convert text into vectors using TF-IDF
# Split feature and label
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
```

```
# TF-IDF(term frequency-inverse document frequency) vectorizer
def Tf_idf(text):
    tfid = TfidfVectorizer(min_df=2,smooth_idf=False)
    return tfid.fit_transform(text)
```

```
# Let's define x, y for modelling
x=Tf_idf(df['comment_text'])
x.shape
```

```
(159571, 54484)
```

```
y = df['comment_label']
y.shape
```

```
(159571,)
```

```
# Train_train_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=42,stratify=y)
```

Testing of Identified Approaches (Algorithms)

- **Below are the algorithms used for this problem:**
- `from sklearn.neighbors import KNeighborsClassifier`
- `from sklearn.linear_model import LogisticRegression`
- `from sklearn.tree import DecisionTreeClassifier`
- `from sklearn.ensemble import RandomForestClassifier`
- `from sklearn.ensemble import AdaBoostClassifier`
- `from sklearn.naive_bayes import MultinomialNB`

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,r
oc_curve,auc,log_loss
from sklearn.model_selection import cross_val_score
```

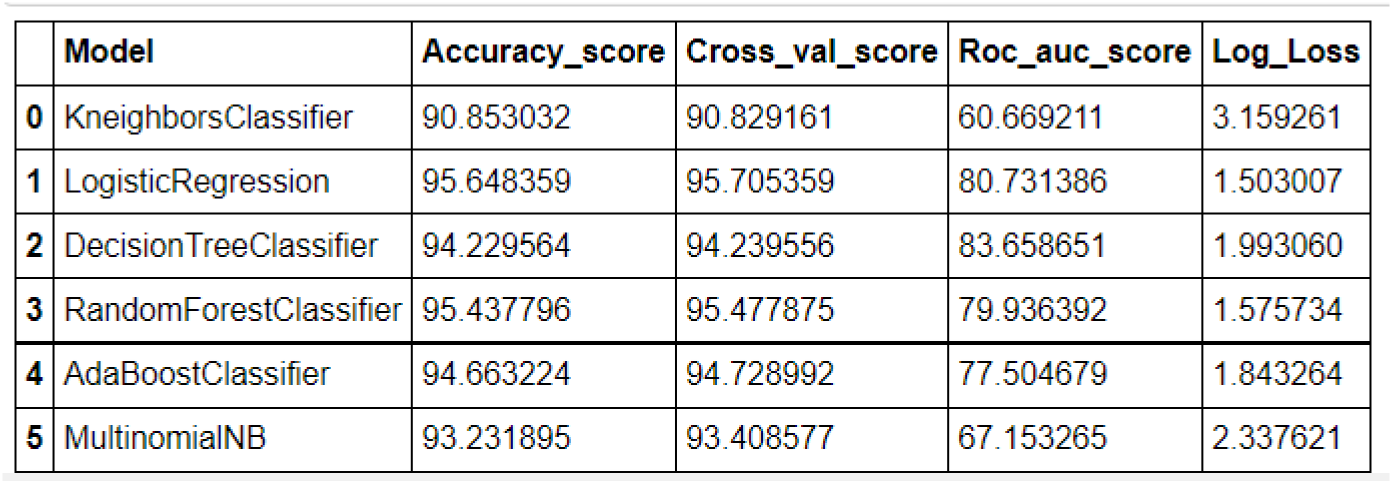
Run and Evaluate selected models

In my approach I have first prepared a method which gives all necessary classification metrics of an algorithm like classification metrics , auc_roc score, confusion matrix,log_loss

```
models=[]
models.append(('KneighborsClassifier',KNN))
models.append(('LogisticRegression',LR))
models.append(('DecisionTreeClassifier',DT))
models.append(('RandomForestClassifier',RF))
models.append(('AdaBoostClassifier',AD))
models.append(('MultinomialNB',MNB))
model_list=[]
```

```
score=[]
cvs=[]
rocscore=[]
logloss=[]

for name, model in models:
    print('*************************',name,'*************************',end='\n\n')

    model_list.append(name)
    model.fit(x_train,y_train)
    print(model,end='\n\n')
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('Accuracy score =',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y, cv=10, scoring='accuracy').mean()
    print('cross validation score =',sc)
    cvs.append(sc*100)
    print('\n')
    loss = log_loss(y_test,pre)
    print('Log loss : ', loss)
    logloss.append(loss)
    false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
    roc_auc=auc(false_positive_rate,true_positive_rate)
    print('roc_auc_score = ', roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')
    print('classification_report\n',classification_report(y_test,pre))
    print('\n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    print(sns.heatmap(cm,annot=True))
    plt.subplot(912)
    plt.title(name)
    plt.plot(false_positive_rate,true_positive_rate, label='AUC= %0.2f'%roc_auc)
    plt.plot([0,1],[0,1],'r--')
    plt.legend(loc='lower right')
    plt.ylabel('True positive rate')
    plt.xlabel('False positive rate')
    print('\n\n')
```

```
result=pd.DataFrame({'Model': model_list, 'Accuracy_score': score, 'Cross_val_score
':cvs,'Roc_auc_score': rocscore,'Log_Loss':logloss})
result
```

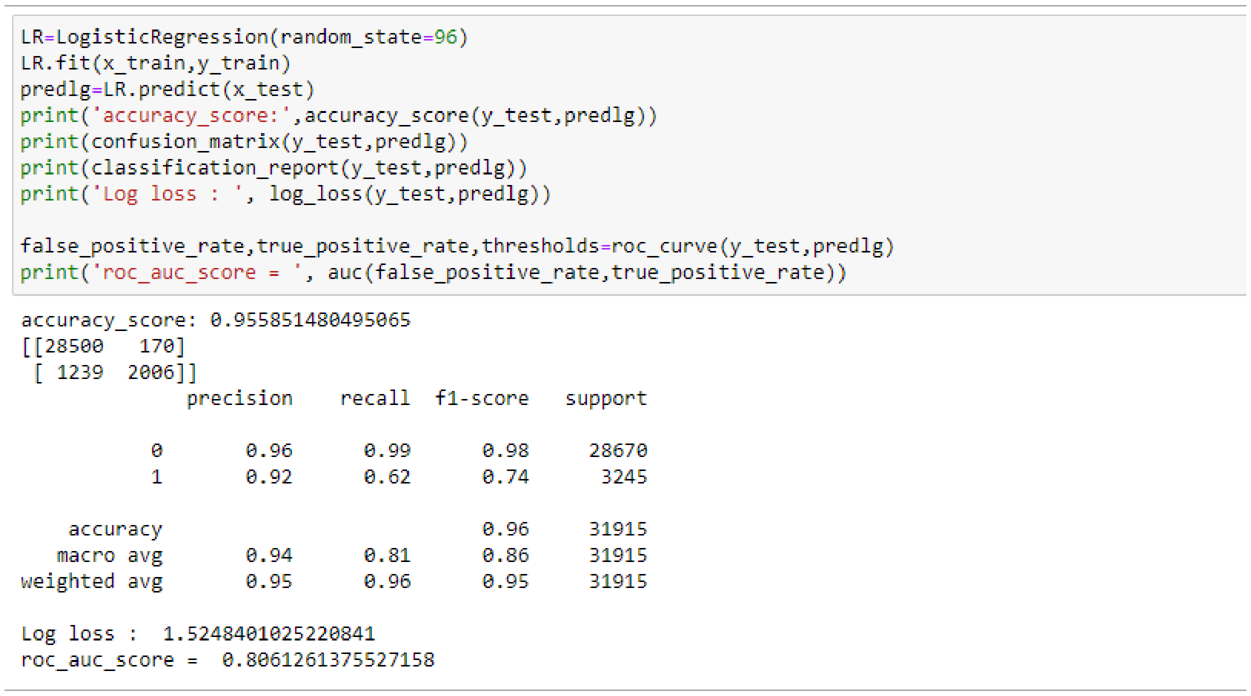


	Model	Accuracy_score	Cross_val_score	Roc_auc_score	Log_Loss
0	KneighborsClassifier	90.853032	90.829161	60.669211	3.159261
1	LogisticRegression	95.648359	95.705359	80.731386	1.503007
2	DecisionTreeClassifier	94.229564	94.239556	83.658651	1.993060
3	RandomForestClassifier	95.437796	95.477875	79.936392	1.575734
4	AdaBoostClassifier	94.663224	94.728992	77.504679	1.843264
5	MultinomialNB	93.231895	93.408577	67.153265	2.337621

We choose Logistic Regression model as the final one,as it gives hightest accuracy score & also log_loss value is minimum which indicates better prediction

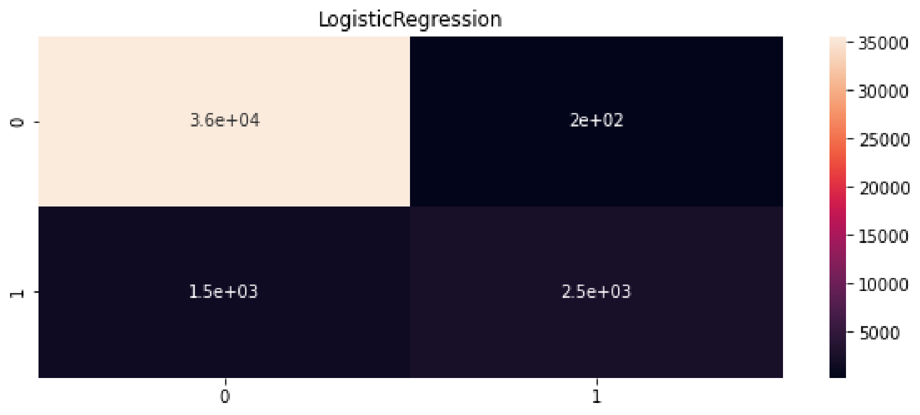
```
#finding best random_state for logistic regression

max_accuracy_score=0
for r_state in range(30,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=r_state,test_si
ze=.20,stratify=y)
    LR=LogisticRegression()
    LR.fit(x_train,y_train)
    LR_pred=LR.predict(x_test)
    accuracy_scr=accuracy_score(y_test,LR_pred)
    if accuracy_scr>max_accuracy_score:
        max_accuracy_score=accuracy_scr
        final_r_state=r_state

print('max accuracy score corresponding to ',final_r_state,'is',max_accuracy_score)
```

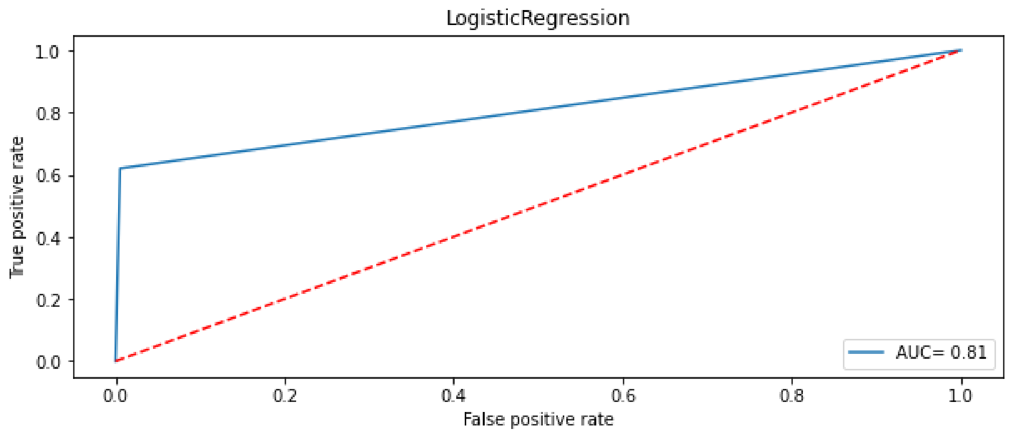
```
max accuracy score corresponding to  96 is 0.9588908036973 21
```



```
LR=LogisticRegression(random_state=96)
LR.fit(x_train,y_train)
predlg=LR.predict(x_test)
print('accuracy_score:',accuracy_score(y_test,predlg))
print(confusion_matrix(y_test,predlg))
print(classification_report(y_test,predlg))
print('Log loss : ', log_loss(y_test,predlg))

false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,predlg)
print('roc_auc_score = ', auc(false_positive_rate,true_positive_rate))
```

```
accuracy_score: 0.955851480495065
[[28500   170]
 [ 1239  2006]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     28670
           1       0.92      0.62      0.74      3245

    accuracy                           0.96     31915
   macro avg       0.94      0.81      0.86     31915
weighted avg       0.95      0.96      0.95     31915

Log loss :  1.5248401025220841
roc_auc_score =  0.8061261375527158
```



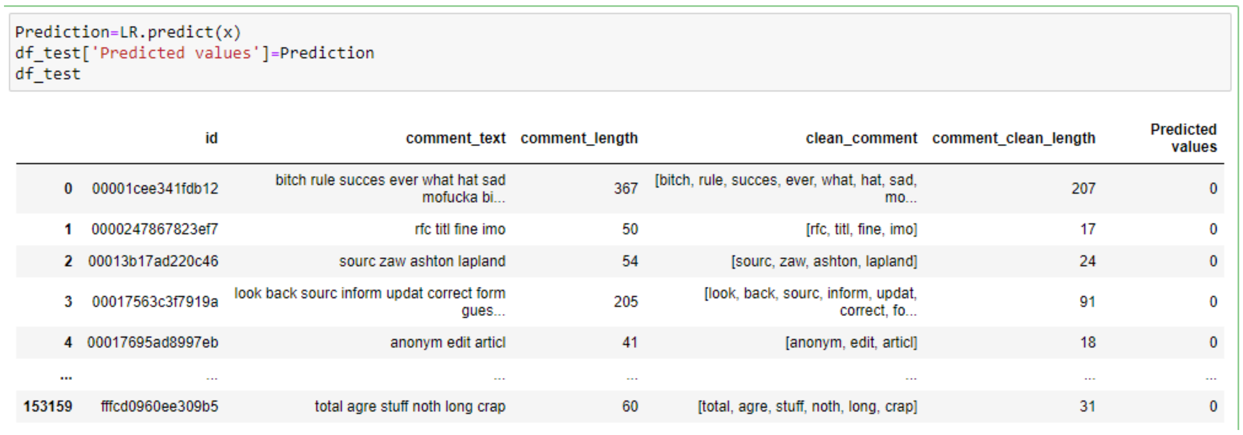


```
#Saving the best model

import joblib

joblib.dump(LR,'malignant_comment.pkl')
```

```
['malignant_comment.pkl']
```

Predicting for Test Data

All the preprocessing step performed fot train data .ie df has been performed for test data .ie df_test also



```
Prediction=LR.predict(x)
df_test['Predicted values']=Prediction
df_test
```

	id	comment_text	comment_length	clean_comment	comment_clean_length	Predicted values
0	00001cee341fdb12	bitch rule succes ever what hat sad mofucka bi...	367	[bitch, rule, succes, ever, what, hat, sad, mo...	207	0
1	0000247867823ef7	rfc titl fine imo	50	[rfc, titl, fine, imo]	17	0
2	00013b17ad220c46	sourc zaw ashton lapland	54	[sourc, zaw, ashton, lapland]	24	0
3	00017563c3f7919a	look back sourc inform updat correct form gues...	205	[look, back, sourc, inform, updat, correct, fo...	91	0
4	00017695ad8997eb	anonym edit articl	41	[anonym, edit, articl]	18	0
...	...	...	...	...	...	...
153159	fffcd0960ee309b5	total agre stuff noth long crap	60	[total, agre, stuff, noth, long, crap]	31	0

```
df_test['Predicted values'].value_counts()
```

```
0    152484
1       680
Name: Predicted values, dtype: int64
```

```
df_test.to_csv('Malignant_comment_Predict.csv')
```

```
# Pickle file.
import joblib
joblib.dump(LR,'Malignant_comment_Predict.pkl')
```

```
['Malignant_comment_Predict.pkl']
```

CONCLUSION

Interpretation of the Results

The confusion matrix gives the number of TP (28500) ,FP(2006) ,FN(1239) and TN (170).

It implies total 31915 rows were considered for testing out of which 30506 were correctly predicted by the model.

The ROC-AUC curve shows the ability to distinguish between 0 and 1 labels. In my final model I got ROC-AUC score of 80%% which implies that probability of my model to distinguish between 0 and 1

- Key Findings and Conclusions of the Study

  Dataset was highly imbalanced with very high number normal comments (90%) comparison to abnormal comments. Many comments belong to more than 1 feature

- Learning Outcomes of the Study in respect of Data Science

I found visualisation a very useful technique to infer insights from dataset.

The ROC AUC plot gives large info about the false positive rate and True positive rate at various thresholds

Use of Tf-Idfwas very helpful in conversion of text to numeric

Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stopwords.

In this project we applied different evaluation metrics like log loss, besides accuracy accuracy.

Limitations:

Since the dataset is large,It takes too much time for hyperparameter tuning .

Alsothe dataset had very less abnormal comments the trained model will be limited in scope for abnormal comments. More data of abnormal comments can definitely help improving the results of the model.