MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

KUMAR GOURABH (kumargourabh94@gmail.com)

ACKNOWLEDGMENT

I would like to thank my mentors at Data Trained, who taught me the concepts of Data Analysis, building a machine learning model, and tuning the parameters for best outcomes.

For this task, I referred the following websites and articles when stuck:

- https://towardsdatascience.com/a-common-mistake-to-avoid-when-encoding-ordinal-features-79e402796ab4

- https://stackoverflow.com/questions/43590489/gridsearchcv-random-forest-regressor-tuning-best-params

- https://www.codegrepper.com/code-examples/delphi/scikit+pca+preserve+column+names+pca+pipeline

- https://stackoverflow.com/questions/22984335/recovering-features-names-of-explained-variance-ratio-in-pca-with-sklearn

I would also like to thank my mentor in Fliprobo, Sajid Choudhary, for providing me with the dataset and problem statement for performing this wonderful task.

INTRODUCTION

Problem Statement

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.

Multilabel vs Multiclass classification?

In multi-class classification, the data can belong to only one label out of all the labels we have. For example, a given picture of a animal may be an cat, dog or elephant only and not a combination of these.

In multi-label classification, data can belong to more than one label simultaneously. For example, in our case a comment may be malignant, threat or loathe at the same time. It may also happen that the comment is positive/neutral and hence does not belong to any of the six labels.

This is therefore a multi-label classification problem.

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. 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.

Analytical Problem Framing

In this project, the method used to represent text as numbers called **TF-IDF** (Term Frequency Inverse Document Frequency). To illustrate the concept of TF-IDF, we need a corpus. A **corpus** is a collection of documents. In a typical Natural Language Processing problem, a corpus can vary from a list of call centre logs/ transcripts, a list of social media feedback to a large collection of research documents.

We now use the TF-IDF to vectorize the words so that machine can understand the words.

1. TF – **Term Frequency** (the number of times the words/terms appear in a document.)
2. IDF - **Inverse Document Frequency**. (If a word appears in all documents, then it may not play such a big part in differentiating between the documents. IDF is a way of identifying such words)

Document Frequency(term t) = number of documents with the term t/ total number of documents = d(t)/n

Inverse Document Frequency = total number of documents / number of documents with the term t = n / d(t)

"If a word appears in all the documents, we want it at the bottom of the range of 0–1. So, a logarithmic scale intuitively makes sense to be used here as log 1 is 0. However, there are some practical considerations such as avoiding the infamous divide by 0 error, 1 is added to the denominator.

Inverse Document frequency for the default settings in TF IDF vectorizer in sklearn is calculated as below *(default settings have smooth_idf=True that adds "1" to the numerator and denominator as if an extra document were seen containing every term in the collection exactly once, which prevents zero divisions).*

$$idf(t) = \ln\left(\frac{1+n}{1+df(t)}\right) + 1$$

- *n is the total number of documents in the document set.*
- *d(t) is the number of documents in the document set that contain term."*[1]

3. Finally, we multiply TF and IDF and normalize it. By default, python uses L2 normalization. In L2 normalization, we are essentially dividing the vector by the length of the vector.

[1] https://towardsdatascience.com/a-gentle-introduction-to-calculating-the-tf-idf-values-9e391f8a13e5

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.

Data Sample

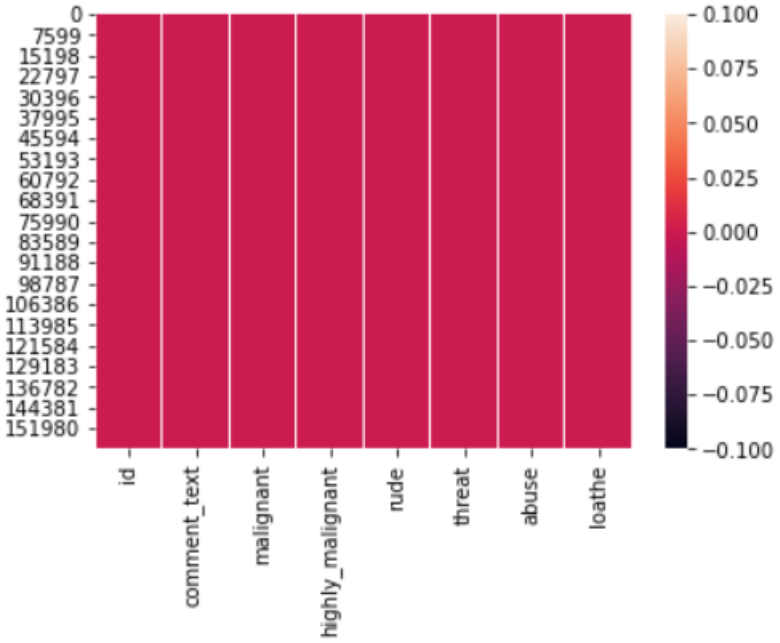
	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	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
10	0005300084f90edc	"\nFair use rationale for Image:Wonju.jpg\n\nT...	0	0	0	0	0	0
11	00054a5e18b50dd4	bbq \n\nbe a man and lets discuss it-maybe ove...	0	0	0	0	0	0
12	0005c987bdfc9d4b	Hey... what is it..\n@ | talk .\nWhat is it.....	1	0	0	0	0	0
13	0006f16e4e9f292e	Before you start throwing accusations and warn...	0	0	0	0	0	0
14	00070ef96486d6f9	Oh, and the girl above started her arguments w...	0	0	0	0	0	0

Fig: Top 15 rows of the training dataset

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

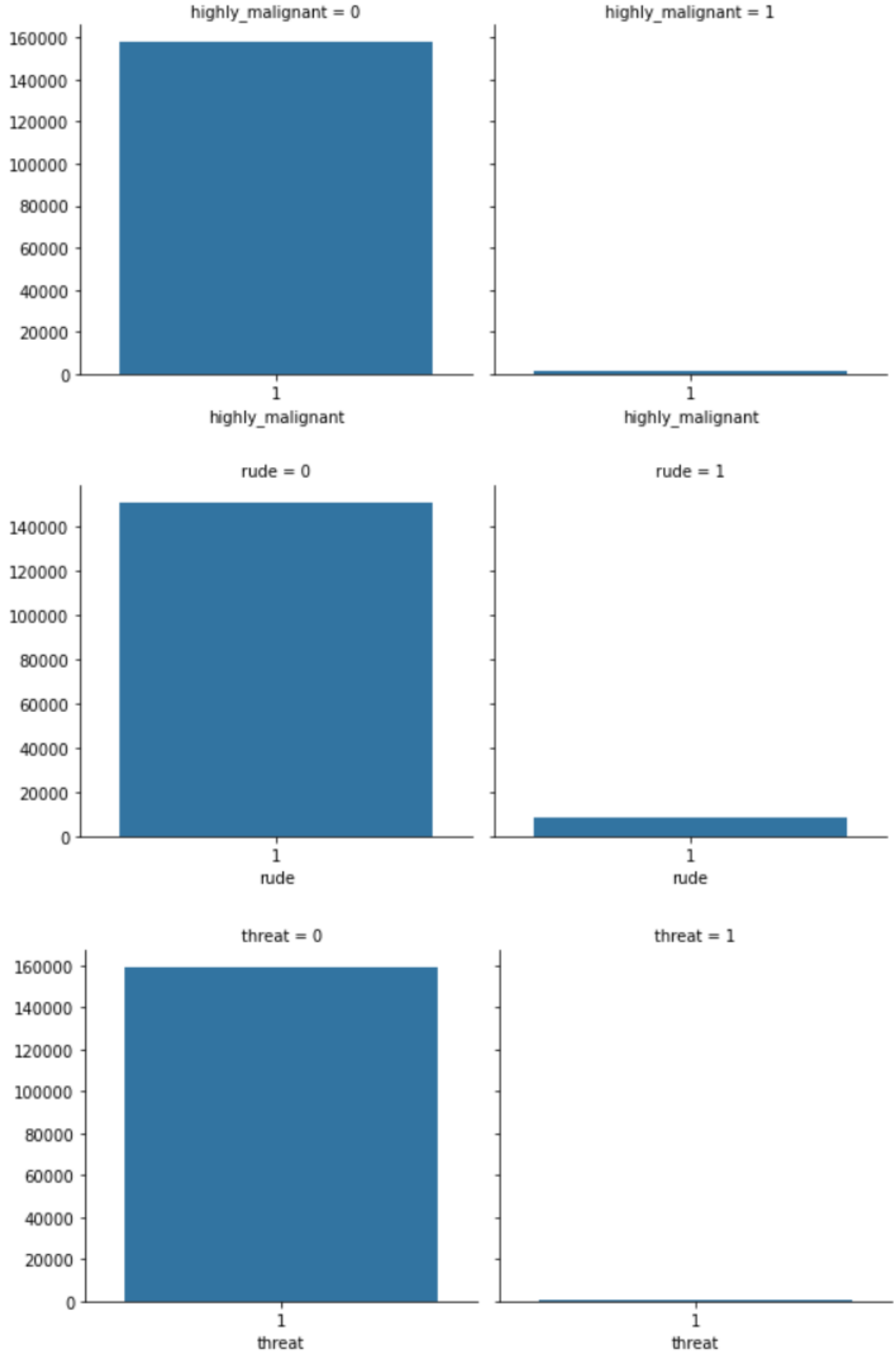
Fig: Top 10 rows of the test dataset

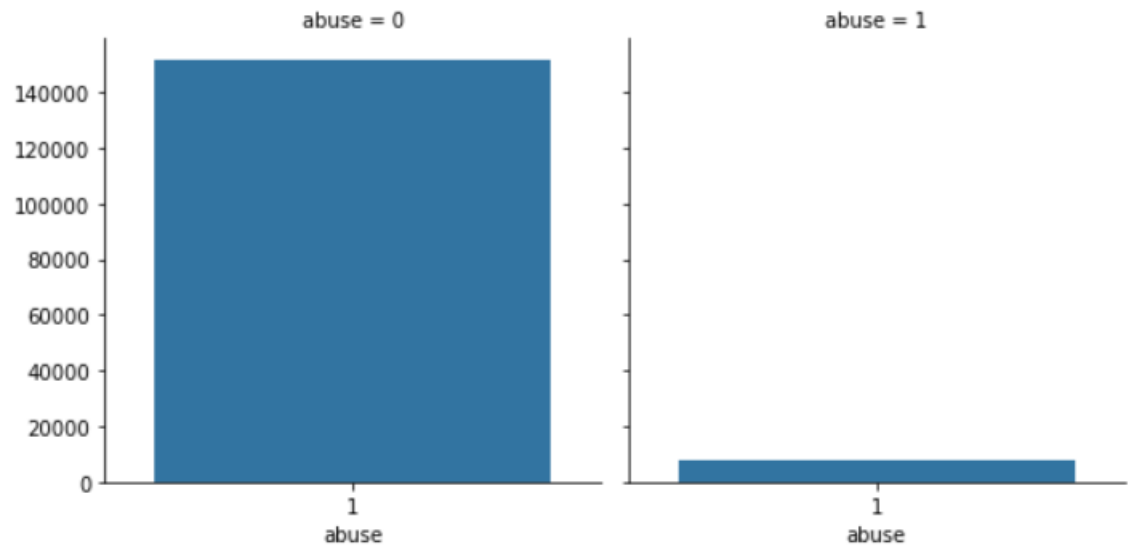
```
In [5]: df.shape
Out[5]: (159571, 8)

In [6]: df_test.shape
Out[6]: (153164, 2)

In [7]: # Checking for null values
        sns.heatmap(df.isnull())
Out[7]: <AxesSubplot:>
```

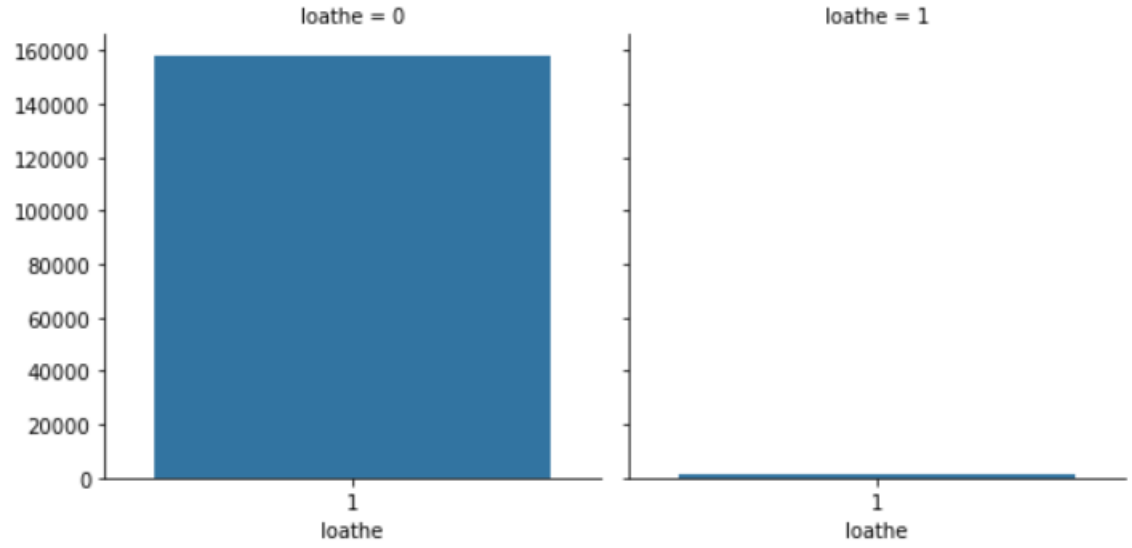


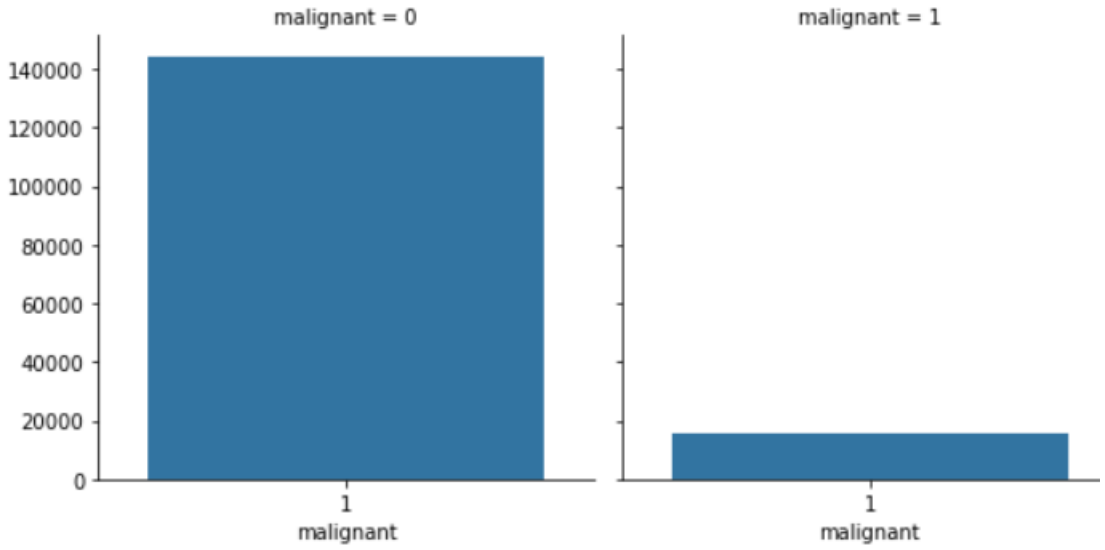
The above heatmap shows there are no Null Values in the dataset.

EDA (Exploratory Data Analysis)





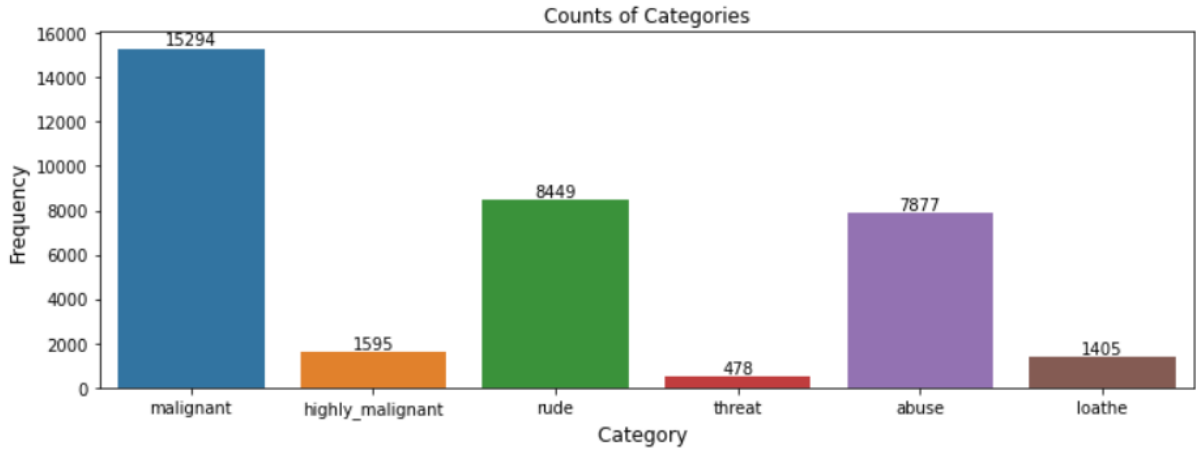




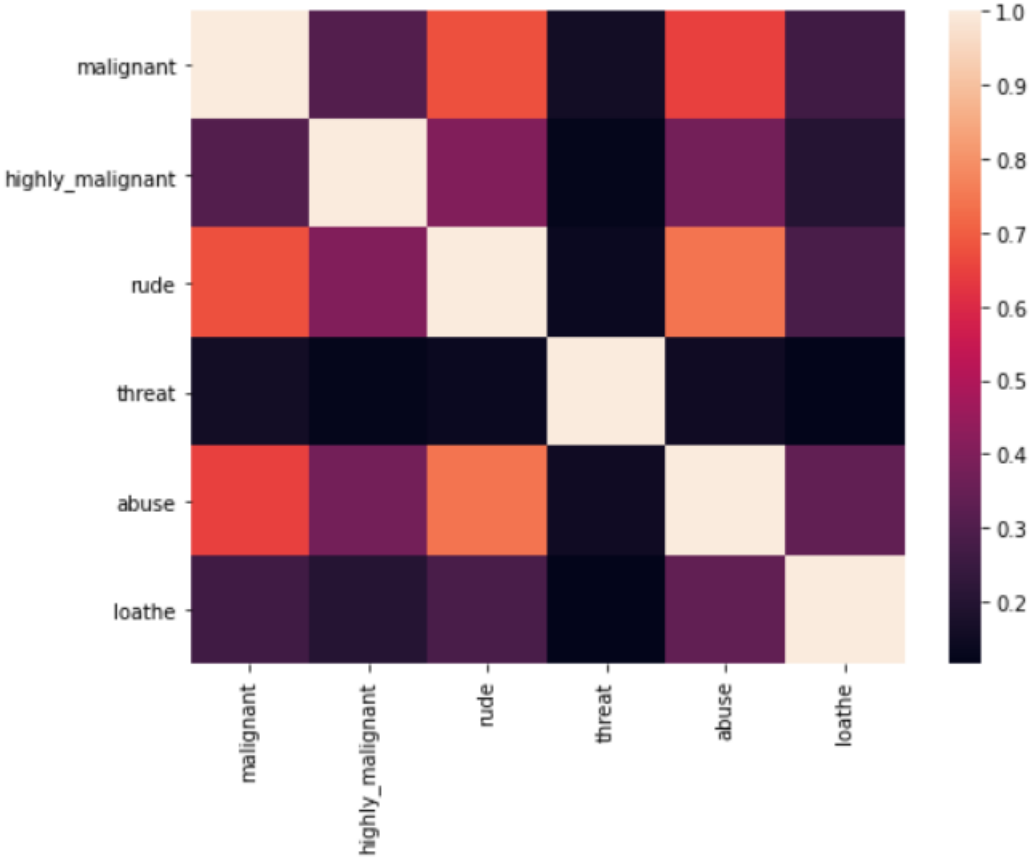
While maximum Categories belong to Malignant, a lot of comments are abusive and rude as well; while threat comments are the minimum

```
# Verifying above statement
counts=df.iloc[:,1:].sum()
counts
```

```
malignant          15294
highly_malignant    1595
rude                8449
threat               478
abuse               7877
loathe              1405
dtype: int64
```




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



Added New Feature for character lengths and Comment Label as shown below.

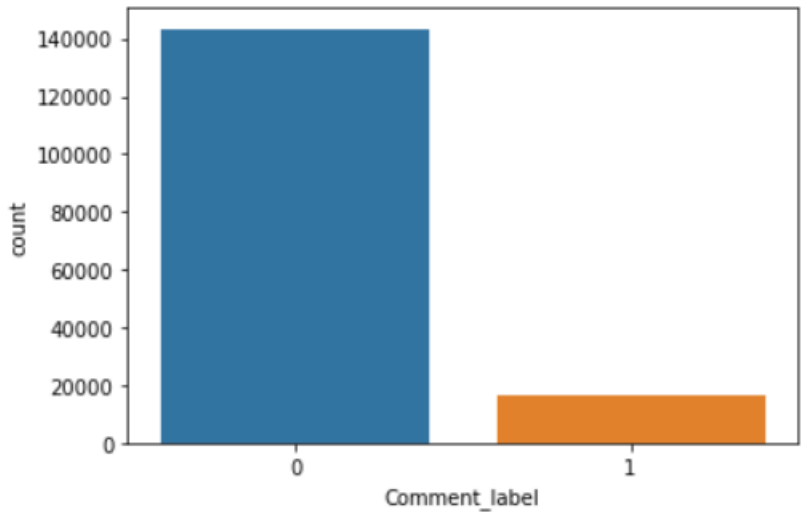
Comment_label 1: Negative Comment

Comment_label 0: Neutral/Positive Comment

```
# Creating a new feature having Negative Comments and Non-Negative Comments.
df['Comment_label'] = df[categories].max(axis=1)
df.head(15)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	Comment_label
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264	0
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112	0
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233	0
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622	0
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67	0
5	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0	65	0
6	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0	44	1

```
0    143346
1     16225
Name: Comment_label, dtype: int64
```



Here, 1 indicates those comments which are either of the 6 mentioned classes. The label is 0 if:

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

Pre-processing the comments for faster and accurate predictions

The comments need to be modified before we can use them for modelling.

```
# Fuction to remove short words
def clean_text(text):
    text = text.lower()
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r"\'scuse", " excuse ", text)
    text = re.sub('\W', ' ', text)
    text = re.sub('\s+', ' ', text)
    text = text.strip(' ')
    return text
```

Using above function, I have replaced slangs and short words with meaningful words.

```
# function to filter using POS tagging. This will be called inside the below function
def get_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

The lemmatized words will be returned based on the root word, as per the Parts of Speech

```
# Function for data cleaning.
def Processed_data(comments):
    # Replace email addresses with 'email'
    comments=re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', comments)

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

    # getting only words(i.e removing all the special characters)
    comments = re.sub(r'[^\w]', ' ', comments)

    # getting only words(i.e removing all the" _ ")
    comments = re.sub(r'[\_]', ' ', comments)

    # getting rid of unwanted characters(i.e remove all the single characters left)
    comments=re.sub(r'\s+[a-zA-Z]\s+', ' ', comments)

    # Removing extra whitespaces
    comments=re.sub(r'\s+', ' ', comments, flags=re.I)
```

The above function (continued below) shows all the pre-processing steps needed to clean the data.

```
    #converting all the letters of the review into lowercase
    comments = comments.lower()

    # splitting every words from the sentences
    comments = comments.split()

    # iterating through each words and checking if they are stopwords or not,
    comments=[word for word in comments if not word in set(STOPWORDS)]

    # remove empty tokens
    comments = [text for text in comments if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(comments)

    # considering words having length more than 3only
    comments = [text for text in comments if len(text) > 3]

    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS
    comments = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]

   # considering words having length more than 3 only
    comments = [text for text in comments if len(text) > 3]
    comments = ' '.join(comments)
    return comments
```

After Performing all the steps, our data now looks as shown below.

Train Data

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	Comment_label	clean_comment_text	clean_comment_length
0	explanation why the edits made under my userna...	0	0	0	0	0	0	264	0	explanation edits username hardcore metallica ...	123
1	d aww he matches this background colour i am s...	0	0	0	0	0	0	112	0	match background colour seemingly stuck thanks...	64
2	hey man i am really not trying to edit war it ...	0	0	0	0	0	0	233	0	trying edit constantly removing relevant infor...	112
3	more i cannot make any real suggestions on imp...	0	0	0	0	0	0	622	0	real suggestion improvement wondered section s...	315
4	you sir are my hero any chance you remember wh...	0	0	0	0	0	0	67	0	hero chance remember page	25

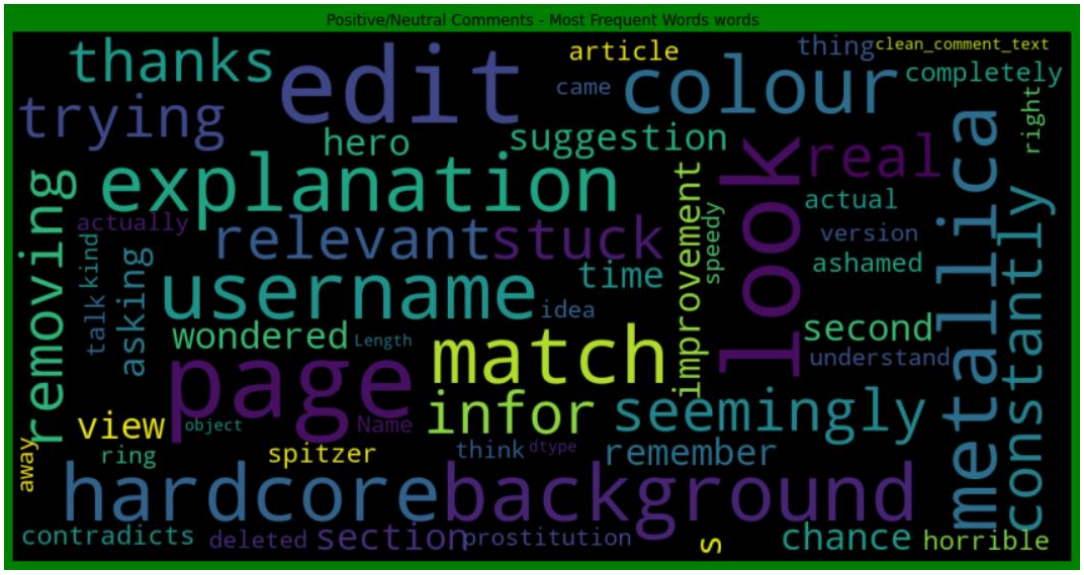
Test Data

	id	comment_text	comment_length	clean_comment_text	clean_comment_length
0	00001cee341fdb12	yo bitch ja rule is more succesful then you wi...	367	bitch rule succesful whats hating mofuckas bit...	184
1	0000247867823ef7	from rfc the title is fine as it is imo	50	title fine	10
2	00013b17ad220c46	sources zawe ashton on lapland	54	source zawe ashton lapland	26
3	00017563c3f7919a	if you have a look back at the source the info...	205	look source information updated correct form g...	104
4	00017695ad8997eb	i do not anonymously edit articles at all	41	anonymously edit article	24

TOP 10 Words in Each Category (Word,Counts)

	Most Frequent Words in Negative Comments	Most Frequent Words in Non negative Comments
0	(fuck, 10080)	(article, 72349)
1	(suck, 4696)	(page, 54390)
2	(wikipedia, 3939)	(wikipedia, 44662)
3	(like, 3937)	(talk, 36421)
4	(shit, 3707)	(like, 24494)
5	(nigger, 3434)	(source, 21257)
6	(fucking, 3337)	(think, 19270)
7	(page, 2870)	(time, 17841)
8	(hate, 2713)	(know, 17689)
9	(faggot, 2496)	(edit, 17248)

The below word cloud represents the most frequent words in the non-negative comments.



The below word cloud represents the most frequent words in the negative comments.

Preparing Data for Modelling

I have created word tokens and added them to a dictionary of words.

```
# Tokenizing
data=[]
from nltk.tokenize import word_tokenize
for j,i in enumerate(df['clean_comment_text']):
    a=word_tokenize(i,'english')
    data.append(a)
```

Creating Word dictionary

```
dictionary = corpora.Dictionary(data)
print(dictionary)
```

```
Dictionary(167609 unique tokens: ['closure', 'doll', 'edits', 'explanation', 'hardcore']...)
```

As, mentioned earlier, I am vectorizing the words using TF-IDF

```
# TF-IDF(term frequency-inverse document frequency) vectorizer
def Tf_idf_train(text):
    tfid = TfidfVectorizer(min_df=3,smooth_idf=False)
    return tfid.fit_transform(text)
```

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

```
(159571, 43246)
```

```
# For y
y = df['Comment_label'].values
y.shape
```

```
(159571,)
```

We are now ready to build a model.

Model Building

I am using the 5 Models as listed below:

```
models
```

```
[('LogisticRegression', LogisticRegression()),
 ('MultinomialNB', MultinomialNB()),
 ('PassiveAggressiveClassifier', PassiveAggressiveClassifier()),
 ('DecisionTreeClassifier', DecisionTreeClassifier()),
 ('RandomForestClassifier', RandomForestClassifier())]
```

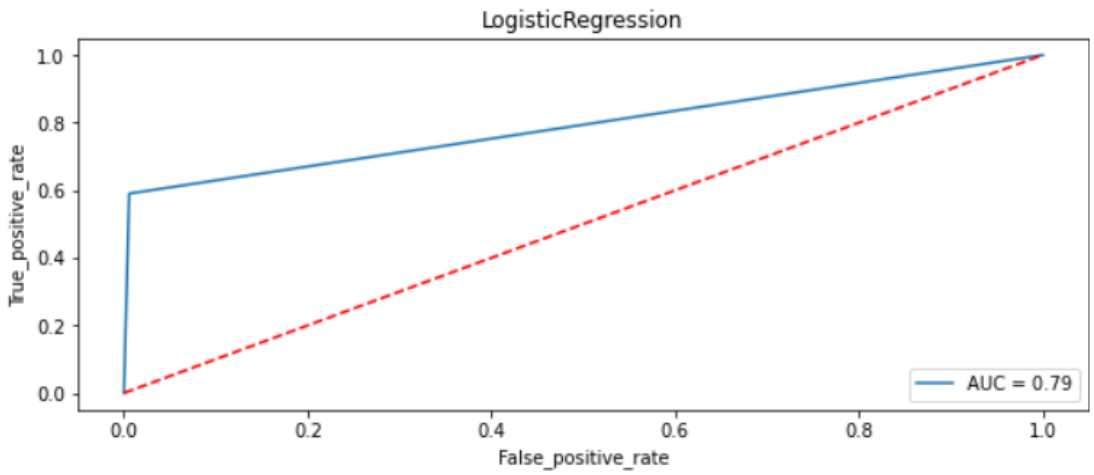
I have used 70% data for training and 30% for testing

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42,stratify=y)
model.fit(x_train,y_train)
```

After Training the model on the above Classifier Algorithms, these are the results obtained:

1. **Logistic Regression**

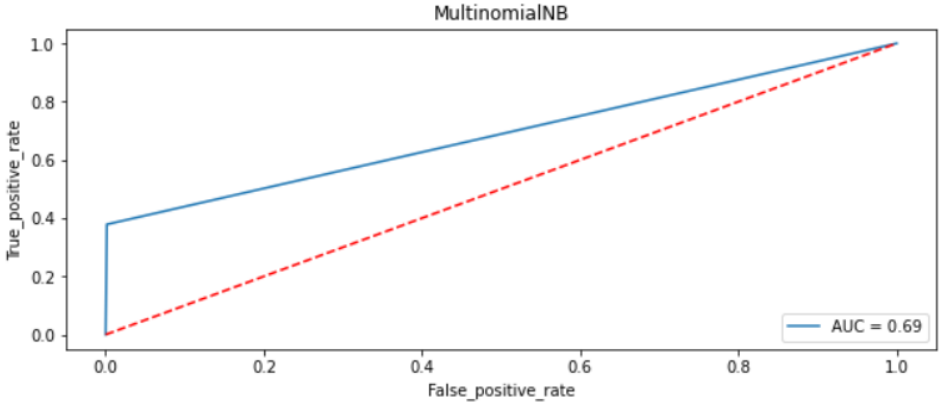
```
LogisticRegression
LogisticRegression()
Learning Score :  0.9577972945147226
Accuracy Score :  0.953125
Cross Val Score :  0.9640015948273865
roc auc score :  0.7921953066680771
Log loss :  1.619009302513781
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.92      0.59      0.72      4868

    accuracy                           0.95     47872
   macro avg       0.94      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872

Confusion Matrix:
 [[42755   249]
 [ 1995  2873]]
```



2. Multinomial NB

```
Learning Score :  0.9397845996830768
Accuracy Score :  0.9354946524064172
Cross Val Score :  0.9265902378998178
roc auc score :  0.688473877991285
Log loss :  2.2279368123856305
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.97     43004
           1       0.97      0.38      0.54      4868

    accuracy                           0.94     47872
   macro avg       0.95      0.69      0.75     47872
weighted avg       0.94      0.94      0.92     47872
```

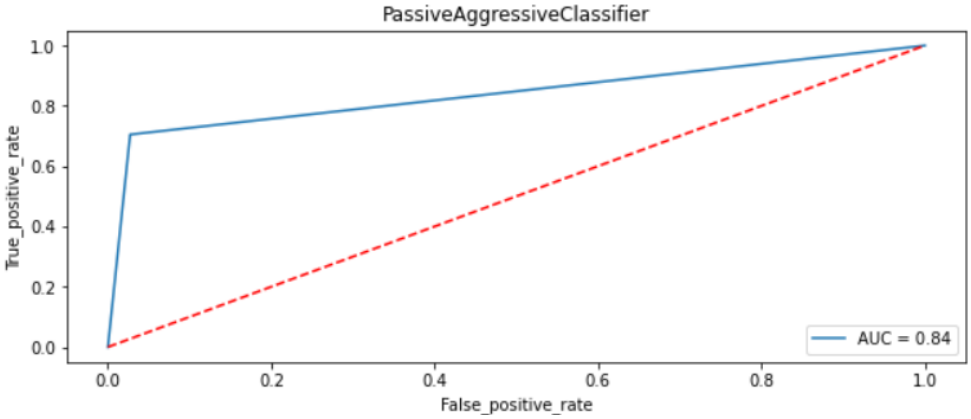
```
Confusion Matrix:
 [[42942    62]
 [ 3026  1842]]
```



3. Passive Aggressive Classifier

```
Learning Score :  0.9903848736336046
Accuracy Score :  0.9455422794117647
Cross Val Score :  0.935036741773213
roc auc score :  0.8388000653777252
Log loss :  1.8809225766737447
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.75      0.70      0.72      4868

    accuracy                           0.95     47872
   macro avg       0.86      0.84      0.85     47872
weighted avg       0.94      0.95      0.94     47872
```

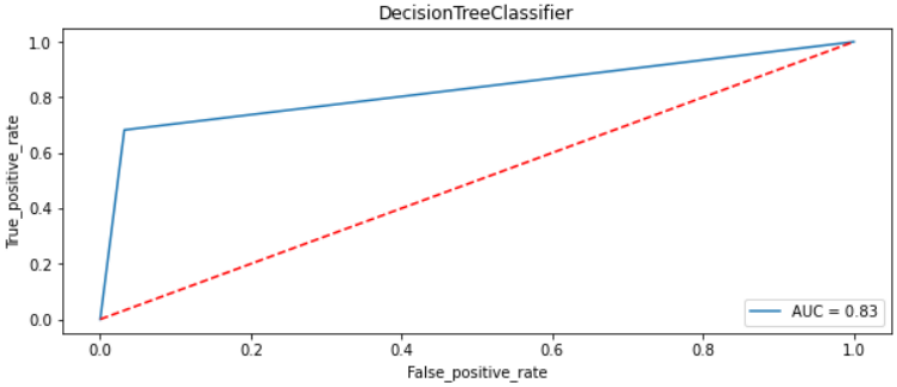
```
Confusion Matrix:
 [[41834  1170]
 [ 1437  3431]]
```



4. Decision tree Classifier

```
Learning Score :  0.9982363315696648
Accuracy Score :  0.938983121657754
Cross Val Score :  0.8341628470387448
roc auc score :  0.8251299185484036
Log loss :  2.1074712670538776
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     43004
           1       0.71      0.68      0.69      4868

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.83     47872
weighted avg       0.94      0.94      0.94     47872
```

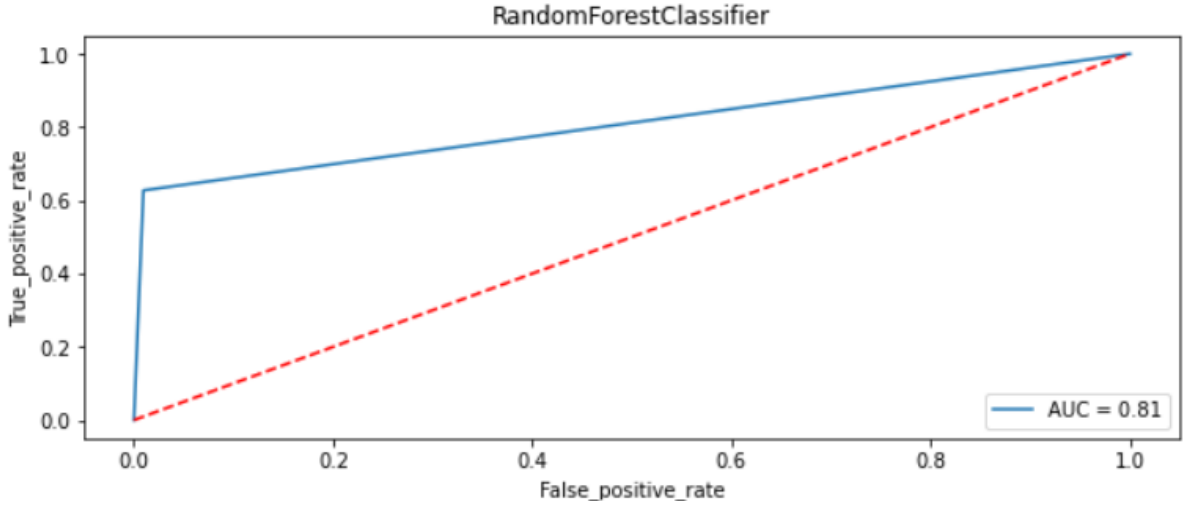
```
Confusion Matrix:
[[41630  1374]
 [ 1547  3321]]
```



5. Random Forest Classifier

```
Learning Score :  0.9982273789380388
Accuracy Score :  0.9535636697860963
Cross Val Score :  0.9548075137447611
roc auc score :  0.8088347268836737
Log loss :  1.6038608406090822
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.88      0.63      0.73      4868

    accuracy                           0.95     47872
   macro avg       0.92      0.81      0.85     47872
weighted avg       0.95      0.95      0.95     47872
```

```
Confusion Matrix:
[[42596   408]
 [ 1815  3053]]
```



Summary of all Models:

	Model	Learning Score	Accuracy Score	Cross Val Score	Auc_score	Log_Loss
0	LogisticRegression	95.779729	95.312500	96.400159	79.219531	1.619009
1	MultinomialNB	93.978460	93.549465	92.659024	68.847388	2.227937
2	PassiveAggressiveClassifier	99.038487	94.554228	93.503674	83.880007	1.880923
3	DecisionTreeClassifier	99.823633	93.898312	83.416285	82.512992	2.107471
4	RandomForestClassifier	99.822738	95.356367	95.480751	80.883473	1.603861

After having a look at all the 5-model performance, I have selected Random Forest Classifier as the final model as it has the minimum log loss and highest accuracy score. All other metrics are also good for Random Forest Classifier.

Final Model

I am using Randomized Search CV to tune the Hyperparameters to see if it increase the scores.

```
# Using RandomForestClassifier for final model
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=.30,stratify=y)
parameters={'bootstrap': [True, False],
 'max_depth': [10, 50, 100, None],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [100, 300, 500, 800, 1200]}


RFC=RandomForestClassifier()

# Applying Randomized Search CV for hyperparameter tuning with scoring= "accuracy"
rand = RandomizedSearchCV(estimator = RFC, param_distributions = parameters,
                          n_iter = 10, cv = 3, verbose=2, random_state=42, n_jobs = -1,scoring='accuracy')
rand.fit(x_train,y_train)
rand.best_params_
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

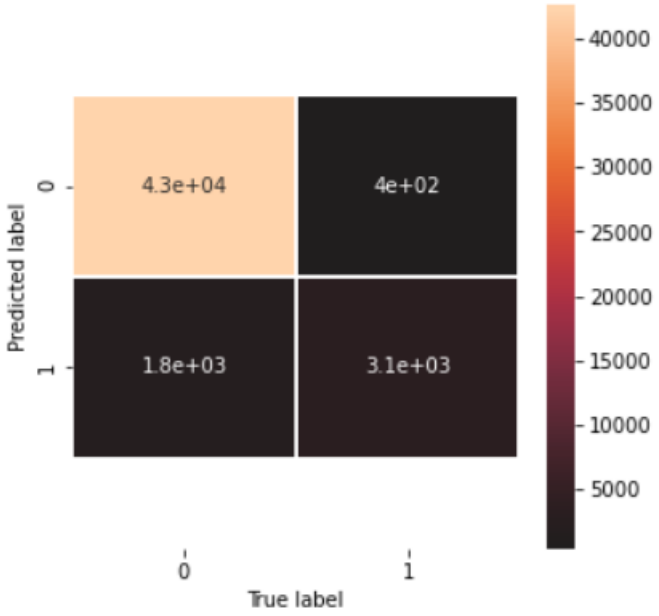
```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 52.0min finished
```

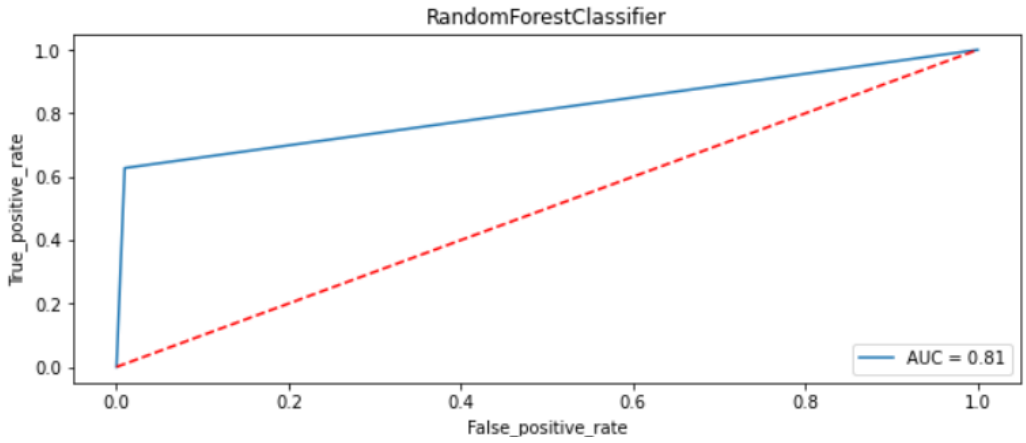
```
{'n_estimators': 500,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_depth': 100,
 'bootstrap': False}
```

However, I have trained the model using default values as they were giving better accuracy and other parameters.

Model Results:

```
Accuracy Score: 0.9538561163101604
Log loss :  1.5937599785872167
Confusion Matrix: [[42603   401]
 [ 1808  3060]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.88      0.63      0.73      4868

    accuracy                           0.95     47872
   macro avg       0.92      0.81      0.85     47872
weighted avg       0.95      0.95      0.95     47872
```





We have got very good results for our model, and this can be used for the given test data to predict the results.

PREDICTION

```
x_testing_data=Tf_idf_test(df_test['clean_comment_text'])
```

```
x_testing_data.shape
```

```
(153164, 43246)
```

```
Prediction=RFC.predict(x_testing_data)
df_test['Predicted values']=Prediction
df_test
```

The below snapshot shows some of the Comments that were identified as Negative comments by the model.

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

```
0    147740
1      5424
Name: Predicted values, dtype: int64
```

```
df_test[df_test['Predicted values']==1].head(50)
```

1001	01ac9ea1271d7409	ou...	134	persian...	98	1
1039	01ba472e48b261d8	illusions about standard croatian the debate o...	4624	illusion standard croatian debate page purely ...	2403	1
1042	01bb8d78d902e20f	have you linked it from any of the articles th...	61	linked article contain	22	1
1116	01d89aada563412a	ec we are not talking about giving things equa...	226	talking giving thing equal time talking totall...	117	1
1170	01ef0facbfec9010	if you cannot see how that source headline is ...	408	source headline phrase factory farming talk re...	191	1
1223	02039486a5ec7226	kashmir hi i have reverted your edit persian i...	179	kashmir reverted edit persian official languag...	88	1
1232	0206f791bf5c6d01	i am following wikipedia disambiguation decidi...	149	following wikipedia disambiguation deciding di...	86	1
1340	023ca2f6f57bd009	dr marmilade you are a bastard pro assad you a...	69	marmilade bastard assad wedge assad	35	1
1342	023d9196bb09858a	oppose the title should be iranian persians is...	185	oppose title iranian persian nonsense thing wa...	94	1

Conclusion

Using a Random Forest Model, I have successfully predicted the comments given in the test data to be Negative vs Non-Negative (Positive and Neutral).

Limitations:

Some of the limitations can be:

- The model might not be able to understand sarcasm.
- Sometimes non negative comments can be wrongly classified as negative ones, leading to loss of constructive feedback or comments.