FLIP ROBO

Malignant Comments Classifier Project

A Project Report by

Dheerajkumar Pittala

Acknowledgement

I would like to thank FlipRobo Technologies for giving me the opportunity to work on this project. I am very grateful to DataTrained team for providing me the knowledge which helped me a lot to work on this project. Reference sources are:

1. Google
2. YouTube
3. TowardsDataScience
4. Stackoverflow
5. DataTrained Notes

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, cyberbully, 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 behavior. There has been a remarkable increase in the cases of cyberbully and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and must 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.

- Conceptual Background of the Domain Problem

In the past few years its seen that the cases related to social media hatred have increased exponentially. The social media is turning into a dark venomous pit for people now a days. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc.

In social media the people spreading or involved in such kind of activities uses filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now. The result of such activities can be dangerous. It gives mental trauma to the victims making their lives miserable. People who are not aware of mental health online hate or cyberbully become life threatening for them. Such cases are also at rise. It is also taking its toll on religions. Each day we can see an incident of fighting between people of different communities or religions due to offensive social media posts. Online hate, described as abusive language, aggression, cyberbully, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity has been identified as a major threat on online social media platforms. These kinds of activities must be checked for a better future.

- Review of Literature

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 cyberbully.

- Motivation for the Problem Undertaken

This project is provided to us by FlipRobo Technologies. The exposure to real world data and the opportunity to deploy our skill- set in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation 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 cyberbully.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

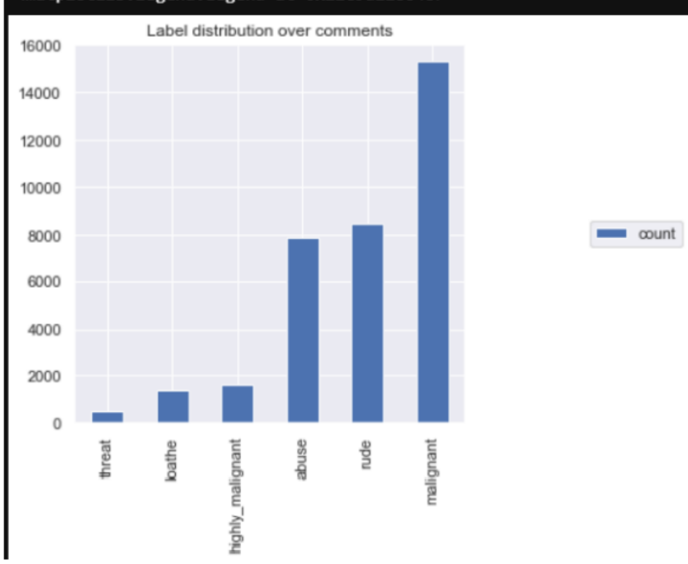
In this problem, the dataset that has been provided to us has 159571 Rows and 8 columns as shown below:

	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

Also, we have multiple labels and multiple target variables involved in our case. Distribution of various labels can be seen below:

```
cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df_distribution = train[cols_target].sum()\
                            .to_frame()\
                            .rename(columns={0: 'count'})\
                            .sort_values('count')

df_distribution.plot.bar(y='count',
                                    title='Label distribution over comments',
                                    figsize=(5, 5))\
                            .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

```
<matplotlib.legend.Legend at 0x21c9d118648>
```



- Data Sources and their formats

```
# Checking for the information of the train dataset

df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
id                  159571 non-null object
comment_text        159571 non-null object
malignant           159571 non-null int64
highly_malignant    159571 non-null int64
rude                159571 non-null int64
threat              159571 non-null int64
abuse               159571 non-null int64
loathe              159571 non-null int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

There are total 8 columns in the train dataset. 2 coulmns (id and comment_text) are of object datatype and all the target variables are of integer datatype (Boolean).

id : A unique id aligned with each comment text. Its datatype is object.

comment_text: It includes the comment text. Its datatype is object.

malignant: It is a column with binary values depicting which comments are malignant in nature. Its datatype is int.

highly_malignant: Binary column with labels for highly malignant text. Its datatype is int.

rude: Binary column with labels for comments that are rude in nature. Its datatype is int.

threat: Binary column with labels for threatening context in the comments. Its datatype is int.

abuse: Binary column with labels with abusive behaviour. Its datatype is int.

loathe: Label to comments that are full of loathe and hatred. Its datatype is int.

The data-set is provided by FlipRobo Technologies as part of the ongoing Data Science internship program. We have been provided with two CSV files namely Train dataset (To train the model) and Test Dataset (Use to test/predict the results)

Loading the train dataset ¶

```
df_train = pd.read_csv('train.csv')
# looking for the dataset
df_train.head()
```

[2]:

	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

```
# Checking for the shape of the train dataset
df_train.shape
```

[3]: (159571, 8)

The train dataset contains 1,59,571 rows and 8 columns including the target columns.

Loading the test dataset:

```
df_test = pd.read_csv('test.csv')
# looking for the dataset
df_test.head()
```

[4]:

	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.

```
# Checking for the shape of the test dataset
df_test.shape
```

[5]: (153164, 2)

The test dataset contains 1,53,164 rows and 2 columns.

- # Data Preprocessing Done

```
# Converting all the comments to lower case

df_train['comment_text']=df_train['comment_text'].str.lower()

df_train.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length
0	0000997932d777bf	explanation\nwhy the edits made under my usern...	0	0	0	0	0	0	264
1	000103f0d9cfb60f	d'aww! he matches this background colour i'm s...	0	0	0	0	0	0	112
2	000113f07ec002fd	hey man, i'm really not trying to edit war. it...	0	0	0	0	0	0	233
3	0001b41b1c6bb37e	"\nmore\ni can't make any real suggestions on ...	0	0	0	0	0	0	622
4	0001d958c54c6e35	you, sir, are my hero. any chance you remember...	0	0	0	0	0	0	67

```
#Replacing email address with 'email'

df_train['comment_text']=df_train['comment_text'].str.replace(r'^.+@[^\.].*\[a-z]{2,}$','emailaddress')

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

#Replacing money symbol with 'moneysymb'(£ can type with ALT key+156)
df_train['comment_text']=df_train['comment_text'].str.replace(r'£|\$','dollers')

#Replacing 10 digit phone number(format include paranthesis, space, no spaces,dashes) with 'phone number'
df_train['comment_text']=df_train['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

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

#Replacing number with 'numbr'
df_train['comment_text']=df_train['comment_text'].str.replace(r'^\d+(\.\d+)?','numbr')

#Removing punctuation
df_train['comment_text']=df_train['comment_text'].str.replace(r'[^\w\d\s]',' ')

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

df_train.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length
0	0000997932d777bf	explanation why the edits made under my userna...	0	0	0	0	0	0	264
1	000103f0d9cfb60f	d aww he matches this background colour i m s...	0	0	0	0	0	0	112
2	000113f07ec002fd	hey man i m really not trying to edit war it...	0	0	0	0	0	0	233
3	0001b41b1c6bb37e	more i can t make any real suggestions on imp...	0	0	0	0	0	0	622
4	0001d958c54c6e35	you sir are my hero any chance you remember...	0	0	0	0	0	0	67

```
# Removing the stopwords

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

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

```
#Checking for the length of the comments after removing the stopwords

df_train['clean_length']=df_train.comment_text.str.len()
df_train.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	0000997932d777bf	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	171
1	000103f0d9cfb60f	aww matches background colour seemingly stuck ...	0	0	0	0	0	0	112	83
2	000113f07ec002fd	hey man really trying edit war guy constantly ...	0	0	0	0	0	0	233	141
3	0001b41b1c6bb37e	make real suggestions improvement wondered sec...	0	0	0	0	0	0	622	374
4	0001d958c54c6e35	sir hero chance remember page	0	0	0	0	0	0	67	29

- Data Inputs- Logic- Output Relationships

```
# Checking for the information of the train dataset

df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
id                  159571 non-null object
comment_text        159571 non-null object
malignant           159571 non-null int64
highly_malignant    159571 non-null int64
rude                159571 non-null int64
threat              159571 non-null int64
abuse               159571 non-null int64
loathe              159571 non-null int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

[7]:

```
# Checking for the information of the test dataset

df_test.info()
```

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

The test dataset contains 2 columns and both are of object datatype.

[8]:

```
# Checking if there is any missing values present in the train dataset

df_train.isnull().sum()
```

Out[8]:

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

There is no missing values in the train dataset.

[9]:

```
# Checking for the missing values in the test dataset

df_test.isnull().sum()
```

Out[9]:

```
id              0
comment_text    0
dtype: int64
```

There is no missing values in the test dataset.

Train and Test Datasets are free of NULL values

- Hardware and Software Requirements and Tools Used

We have used the following Software & Libraries

1. Jupyter Notebook
2. Python 3
3. Pandas
4. Numpy
5. Matplotlib
6. Seaborn
7. NLTK
8. SkLearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches and Testing of Identified Approaches

Upon doing some research on such problems, it was identified that for such problems, RF models are the best as they are non-linear tree based models. I had an intuition of RF model to be the one, but still I tested my dataset on the following algorithms:

1. Logistic Regression
2. Decision Trees
3. Random Forest
4. KNeighborsClassifier

- Run and Evaluate selected models

Different models were tried upon after doing the train test split. Following is the modelwise dataframe obtained after running all the models:

	Model	Accuracy_score	Cross_val_score	Difference	Roc_auc_curve
0	KNeighborsClassifier	91.723763	91.779834	-0.056071	61.056311
1	LogisticRegression	95.531835	95.563730	-0.031895	80.186820
2	DecisionTreeClassifier	94.082136	94.153699	-0.071563	82.760771
3	RandomForestClassifier	95.577791	95.678412	-0.100622	83.630276

- Key Metrics for success in solving problem under Consideration

The key matrices used in solving the problem were applied in the same code of model building. Those matrices were Accuracy Score, Cross validation Score, AUC ROC Score, log_loss and learning score. Cross validation score was used to create a more generic model so that it performs well under different circumstances and in various permutations and combinations of data. Also, as we can see, log loss score is also inversely proportional to accuracy score and it is closer to zero in case of RF and Extra Trees. It has helped us in strengthening our conclusion of cross val scores. In addition to this, AUC ROC score is one of the key metric for evaluation as it tells us how capable the model is in distinguishing between the positive and negative classes. It means that it observes the True Positive Rate and False Positive Rate for users who paid the loan and are falsely marked as defaulters

[45]:

	Model	Accuracy_score	Cross_val_score	Difference	Roc_auc_curve
0	KNeighborsClassifier	91.723763	91.779834	-0.056071	61.056311
1	LogisticRegression	95.531835	95.563730	-0.031895	80.186820
2	DecisionTreeClassifier	94.082136	94.153699	-0.071563	82.760771
3	RandomForestClassifier	95.577791	95.678412	-0.100622	83.630276

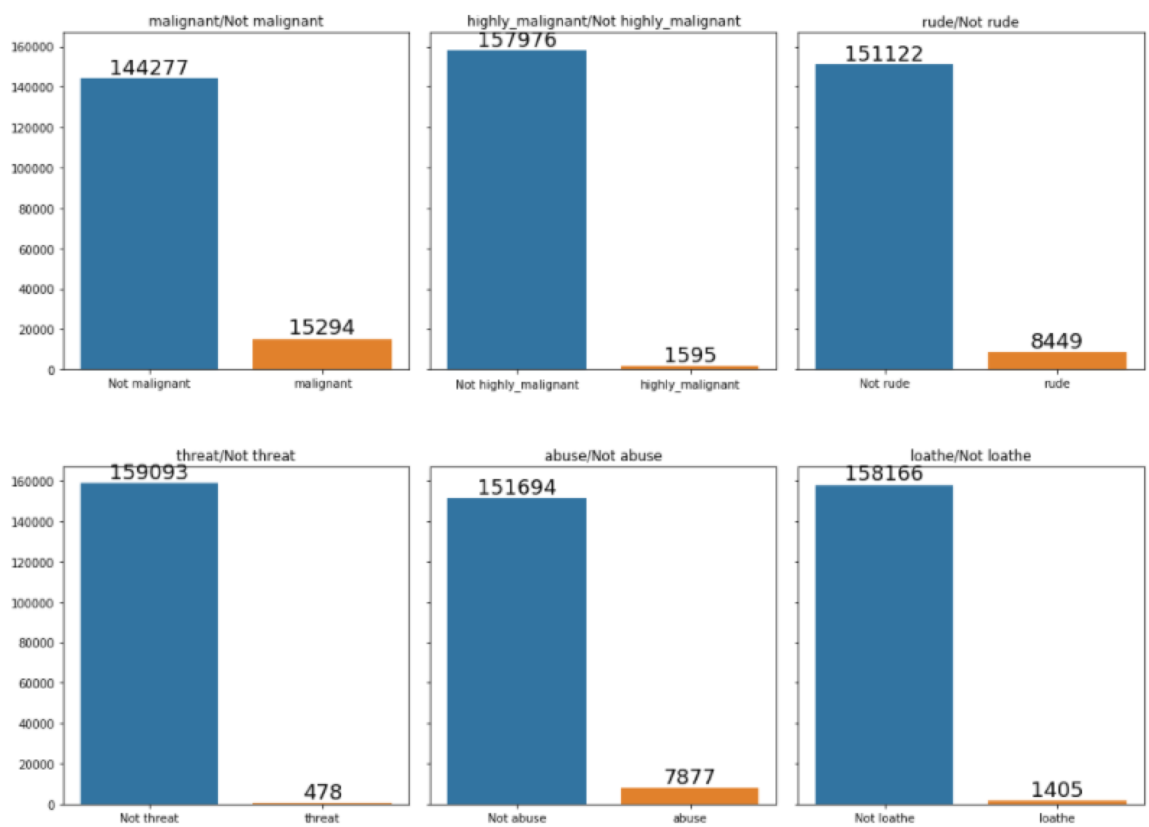
From the above table, we found that the minimum difference between the accuracy score and cross validation score is for LogisticRegression. So, the best fit model for our project is LogisticRegression.

EDA and Data Visualisation

Firstly, we visualized various target columns that we have in our dataset by the following code:

```
target_columns = ['malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe']
```

```
fig, axes = plt.subplots(2, 3, figsize=(14, 10), sharey=True)

i = -1
j = 0
for c in target_columns:
    plt.tight_layout(h_pad=5.0)
    if j % 3 == 0:
        i+=1
        j = 0
        ax=sns.barplot(['Not '+c,c], df_train[c].value_counts().values, ax=axes[i,j])
    else:
        ax=sns.barplot(['Not '+c,c], df_train[c].value_counts().values, ax=axes[i,j])

    #adding the text labels
    rects = ax.patches
    labels = df_train[c].value_counts().values
    for rect, label in zip(rects, labels):
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2, height + 5, label, ha='center', va='bottom', fontsize=18)
    axes[i,j].set_title(c+'/Not '+c)
    j += 1
```

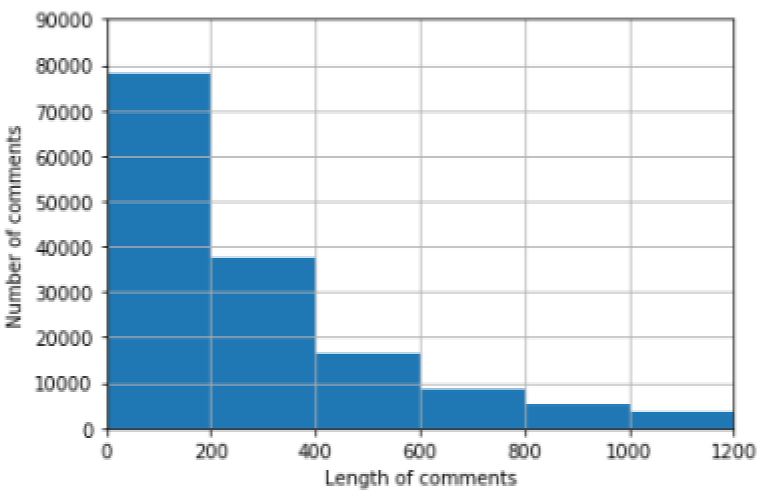


The above graph shows that most of the data in the train dataset are non-offensive.

Analysis of Length of comments through

```python
#Analysing the lengths of comments through visualisation
x = [len(comment[i]) for i in range(comment.shape[0])]

print('Average length of comments: {:.3f}'.format(sum(x)/len(x)) )
bins = [1,200,400,600,800,1000,1200]
plt.hist(x, bins=bins)
plt.xlabel('Length of comments')
plt.ylabel('Number of comments')
plt.axis([0, 1200, 0, 90000])
plt.grid(True)
plt.show()
```

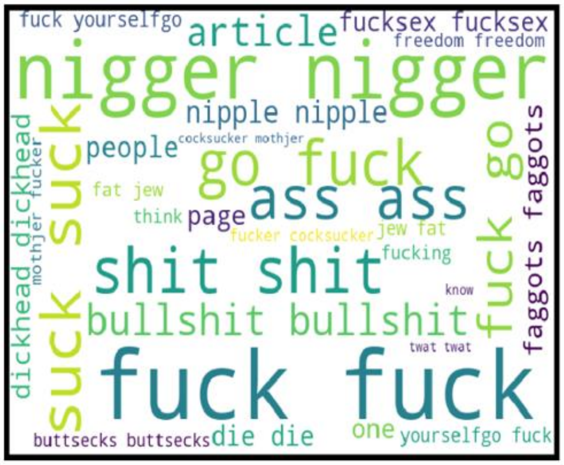
```
Average length of comments: 394.139
```



The average length of comments was found to be 394.

Word Cloud Of Malignant And Highly Comments

Word cloud of rude and threat malignant comments

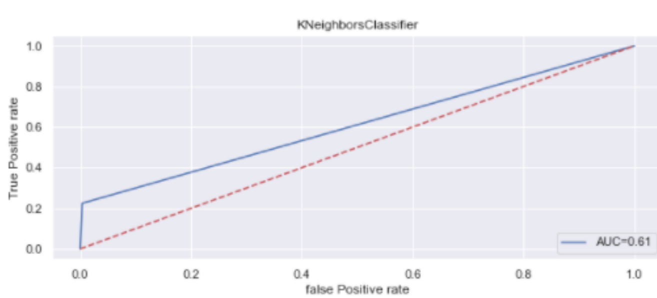


Word cloud of Abuse & Loathe comments

Word cloud not malignant comments

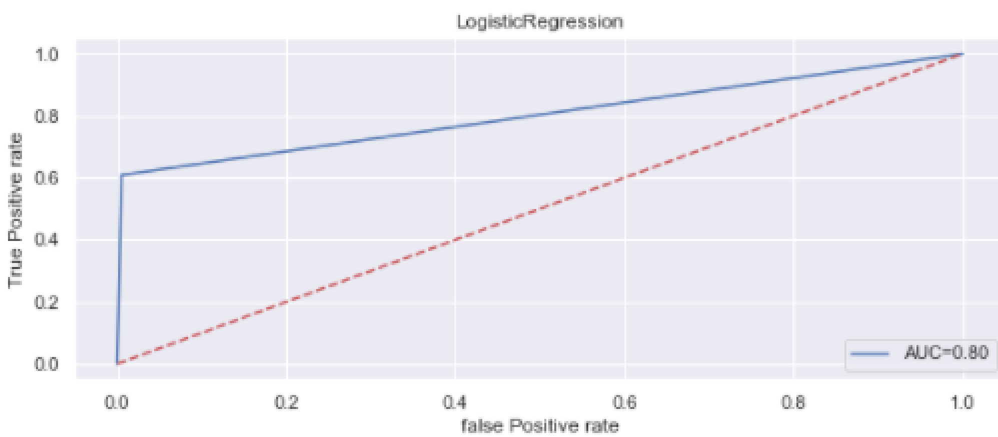
Visualizations and Interpretations

```
KNeighborsClassifier(n_neighbors=6)
Accuracy_score=  0.9172376336898396
Cross_val_score= 0.9177983431914599
roc_auc_score=  0.6105631081187836
classification_report
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     42950
           1       0.88      0.22      0.36      4922

    accuracy                           0.92     47872
   macro avg       0.90      0.61      0.66     47872
weighted avg       0.91      0.92      0.89     47872


Confusion Matrix
 [[42805   145]
 [ 3817  1105]]
```



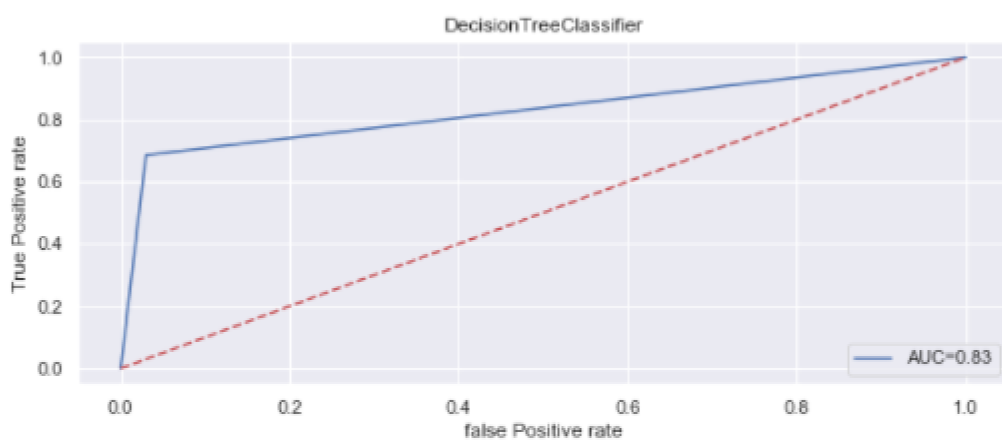
The area under the curve is 0.61, which means that 61% of the predictions by the model are correct

```
LogisticRegression()


Accuracy_score=  0.9553183489304813
Cross_val_score= 0.9556373011751551

roc_auc_score=  0.8018681986131497

classification_report
               precision    recall  f1-score   support

           0       0.96      1.00      0.98     42950
           1       0.93      0.61      0.74      4922

    accuracy                           0.96     47872
   macro avg       0.95      0.80      0.86     47872
weighted avg       0.95      0.96      0.95     47872


Confusion Matrix
 [[42737   213]
 [ 1926  2996]]
```



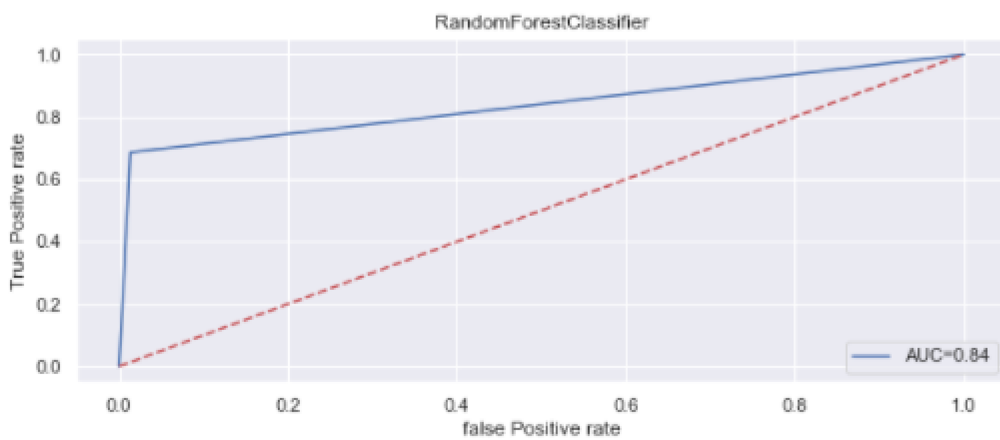
The area under the curve is 0.80, which means that 80% of the predictions by the model are correct

```
DecisionTreeClassifier()


Accuracy_score=  0.9408213569518716
Cross_val_score= 0.9415369888354681



roc_auc_score=  0.8276077093697775

classification_report
               precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.72      0.69      0.70      4922

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


Confusion Matrix
 [[41667  1283]
 [ 1550  3372]]
```



The area under the curve is 0.83, which means that 83% of the predictions by the model are correct

```
RandomForestClassifier()


Accuracy_score=  0.9557779077540107
Cross_val_score= 0.9567841246465967



roc_auc_score=  0.836302756056176

classification_report
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.69      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.84      0.87     47872
weighted avg       0.95      0.96      0.95     47872


Confusion Matrix
 [[42379   571]
 [ 1546  3376]]
```



The area under the curve is 0.84, which means that 84% of the predictions by the model are correct

Conclusion

- ## Key Findings and Conclusions of the Study

After doing the whole study, following were the key findings:

1. As a social media channel, we were able to filter out most of the comments as malignant with an accuracy score of 95% and AUC ROC score of 84%, which are decent numbers.

2. 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 behavior.

3. From the above analysis the below mentioned results were achieved which depicts the chances and conditions of a comment being a hateful comment or a normal comment.

4. With the increasing popularity of social media, more and more people consume feeds from social media and due differences they spread hate comments to instead of love and harmony. It has strong negative impacts on individual users and broader society.

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

This project helped us to know the concepts of NLP and its application in the field of Data Science. This also helped me to sharpen my knowledge in the different classification models.

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

There would be a lot of comments which will not be classified in this. For example, comments posted in languages other than English will not be classified by the model because the model was trained only in English language. Moreover, we would not be able to identify sarcasm and figure of speech, which again can be a malignant comment, but our machine cannot understand the same