Malignant Comments Classifier Project

Submitted by:

Radhika Narayana

ACKNOWLEDGMENT

Thanks for giving me the opportunity to work in FlipRobo Technologies as Intern and would like to express my gratitude to Data Trained Institute as well for trained me in Data Science Domain.

This helps me to do my projects well and understand the concepts.

Resources Referred – Google, GitHub, Blogs for conceptual referring

Links – Medium.com, towardsdatascience.com

INTRODUCTION

- Business Problem Framing

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.

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

Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

- ## Conceptual Background of the Domain Problem

  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.

  Online hate, described as abusive language, aggression, cyberbullying, 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.

- ## Motivation for the Problem Undertaken

  The project was the first provided to me by Flip-Robo as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective.

  The main aim 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.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Here we need to find whether the given comments are malignant words or not. It is text classification problem where we need to predict the target variable from the text and, we have multiple target variables like malignant, high malignant, rude, abuse, loathe.

- Data Sources and their formats

  The Data is provided by Flip Robo Technologies, and it has Train and Test Data Set and need to train our data in Train dataset and need to load the Test dataset to make the predictions.

```
# Loading the train dataset

df = pd.read_csv("train.csv")
df
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loat
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	
...	...	...	...	...	...	...	...	
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	

159571 rows × 8 columns

```
# Loading the test dataset

te = pd.read_csv("test.csv")
te
```

	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.
...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...

153164 rows × 2 columns

- ## Data Pre-processing Done

For Data pre-processing we did some data cleaning, where we used WordNet lemmatizer to clean the words and removed special characters using Regexp Tokenizer.

Then, filtered the words by removing stop words and then used lemmatizers and joined and return the filtered words.

Used TFIDF vectorizer to convert those text into vectors and trained the train and loaded the test dataset.

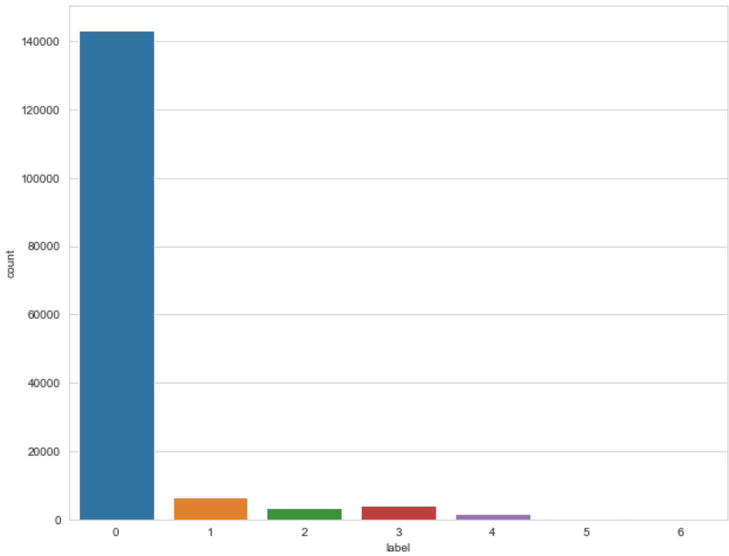
```python
#Defining the stop words
stop_words = stopwords.words('english')

#Defining the Lemmatizer
lemmatizer = WordNetLemmatizer()
```

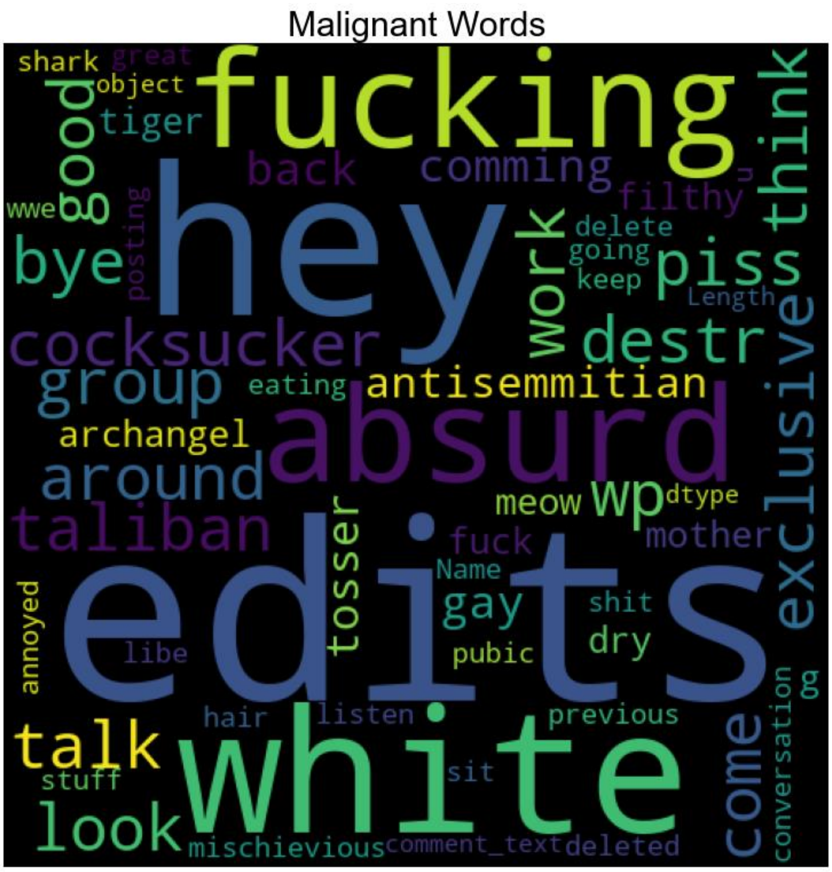
```python
#Replacing '\n' in comment_text
df['comment_text'] = df['comment_text'].replace('\n',' ')

#Function Definition for using regex operations and other text preprocessing for getting cleaned texts
def clean_comments(text):

    #convert to lower case
    lowered_text = text.lower()

    #Replacing email addresses with 'emailaddress'
    text = re.sub(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress', lowered_text)

    #Replace URLs with 'webaddress'
    text = re.sub(r'http\S+', 'webaddress', text)

    #Removing numbers
    text = re.sub(r'[0-9]', " ", text)

    #Removing the HTML tags
    text = re.sub(r"<.*?>", " ", text)

    #Removing Punctuations
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'\_',' ',text)

    #Removing all the non-ascii characters
    clean_words = re.sub(r'[^\x00-\x7f]',r'', text)

    #Removing the unwanted white spaces
    text = " ".join(text.split())

    #Splitting data into words
    tokenized_text = word_tokenize(text)

    #Removing remaining tokens that are not alphabetic, Removing stop words and Lemmatizing the text
    removed_stop_text = [lemmatizer.lemmatize(word) for word in tokenized_text if word not in stop_words if word.isalpha()]

    return " ".join(removed_stop_text)
```

```
#Converting the features into number vectors
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
```

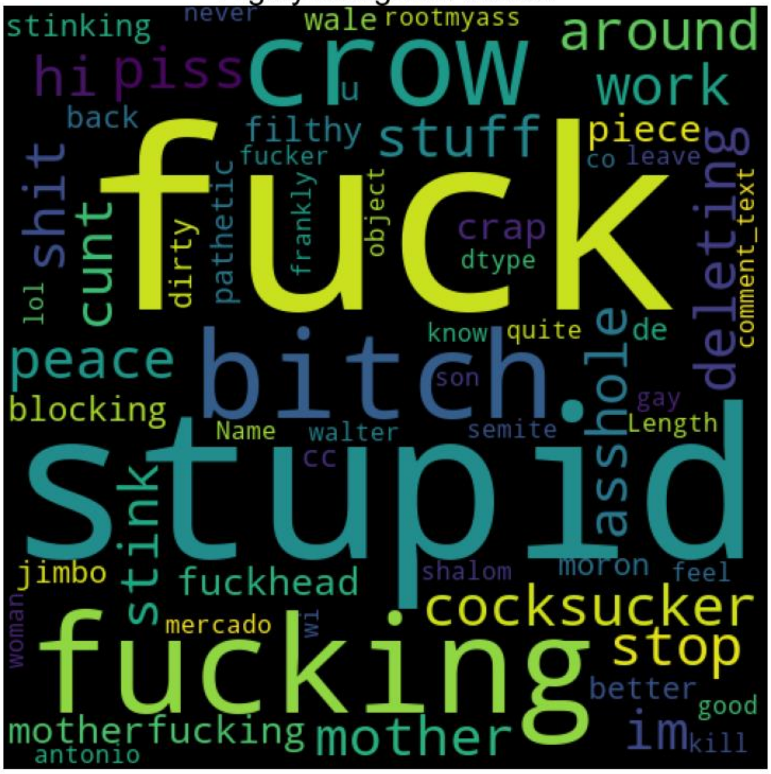
```
#Let's Separate the input and output variables represented by X and y respectively in train data and convert them
X = tf_vec.fit_transform(df['comment_text'])
```

- Data Inputs- Logic- Output Relationships



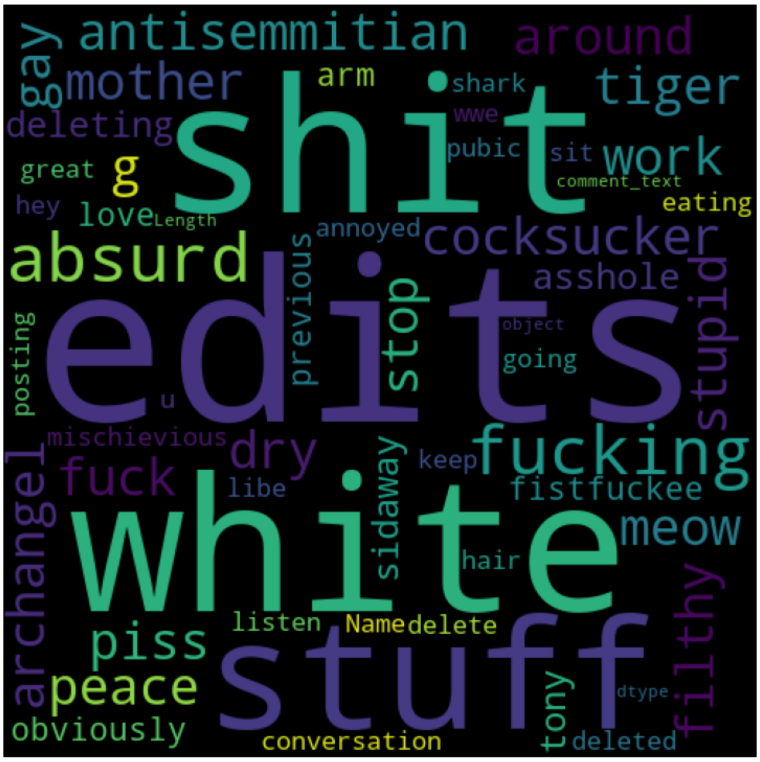


Highly Malignant Words

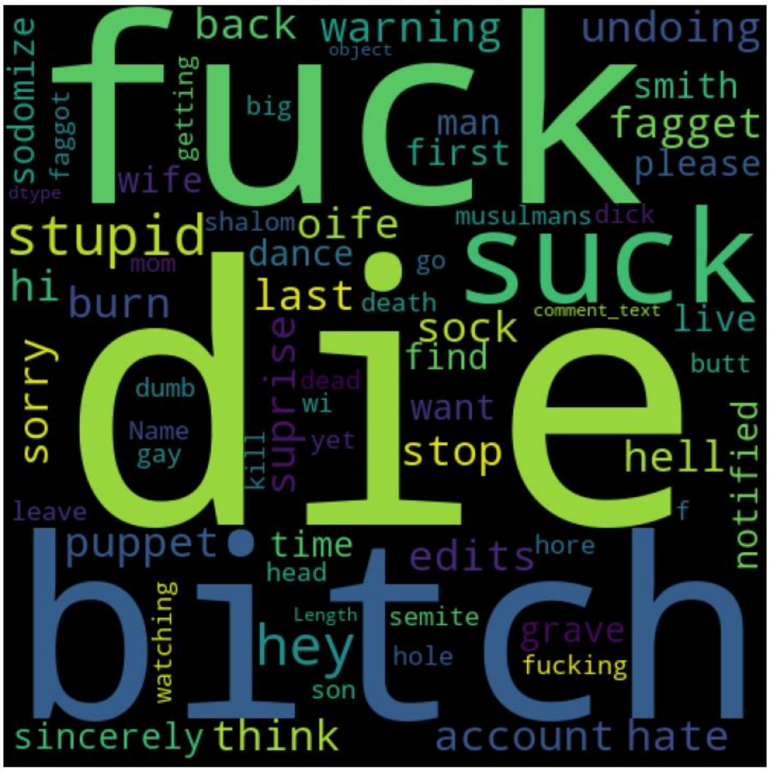


Rude Words

Abuse Words



Threaten Words



From the above graph we can see the most used words in all categories – malignant, highly malignant, abuse, loathe, rude.

- ## Hardware and Software Requirements and Tools Used

  Model training was done on Jupiter Notebook. Kernel Version is Python3.

  Hardware -- > Intel 8GB RAM, i5 processor

```
#importing the required libraries

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV
from sklearn.metrics import f1_score,accuracy_score,classification_report,confusion_matrix,roc_curve,roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

The above libraries and packages used in this project for building a model.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Converting the label into 0 and 1 as below,

```
df['label'].value_counts()
```

```
0    143346
1      6360
3      4209
2      3480
4      1760
5       385
6        31
Name: label, dtype: int64
```

```
#converting label as 0 and 1

df['label'] = [1 if out >0 else 0 for out in df['label']]
```

```
df['label'].value_counts()
```

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

- Testing of Identified Approaches (Algorithms)

  - Logistic Regression
  - Gradient Boost Classifier
  - Decision Tree Classifier
  - Naïve Bayes Multi-Nomial NB
  - Passive Aggressive Classifier

- Run and evaluate selected models

```
from sklearn.linear_model import LogisticRegression

lor = LogisticRegression()
lor.fit(x_train,y_train)
y_pred = lor.predict(x_test)
scr_lor = cross_val_score(lor,x_over,y_over,cv=5)


print("F1 score \n", f1_score(y_test,y_pred))
print("CV Score :", scr_lor.mean())
print("-------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("--------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
print("ROC AUC Score \n", roc_auc_score(y_test,y_pred))
```

```
F1 score
 0.9316651152603624
CV Score : 0.9311631098256351
-------------------------------------------------------

Classification Report
               precision    recall  f1-score   support

           0       0.94      0.92      0.93     35600
           1       0.92      0.94      0.93     36073

    accuracy                           0.93     71673
   macro avg       0.93      0.93      0.93     71673
weighted avg       0.93      0.93      0.93     71673

--------------------------------------------------------

Confusion Matrix
 [[32651  2949]
 [ 2043 34030]]
ROC AUC Score
 0.9302638824300412
```

```
from sklearn.ensemble import GradientBoostingClassifier

sv = GradientBoostingClassifier()
sv.fit(x_train,y_train)
y_pred = sv.predict(x_test)
scr_sv = cross_val_score(sv,x_over,y_over,cv=5)

print("F1 score \n", f1_score(y_test,y_pred))
print("CV Score :", scr_sv.mean())
print("-------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("--------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
print("ROC AUC Score \n", roc_auc_score(y_test,y_pred))
```

```
F1 score
 0.8143213723588609
CV Score : 0.8358587355762415
-------------------------------------------------------

Classification Report
               precision    recall  f1-score   support

           0       0.77      0.97      0.85     35600
           1       0.95      0.71      0.81     36073

    accuracy                           0.84     71673
   macro avg       0.86      0.84      0.83     71673
weighted avg       0.86      0.84      0.83     71673

--------------------------------------------------------

Confusion Matrix
 [[34384  1216]
 [10463 25610]]
ROC AUC Score
 0.8378959830829931
```

```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred = dt.predict(x_test)
scr_dt = cross_val_score(dt,x_over,y_over,cv=5)


print("F1 score \n", f1_score(y_test,y_pred))
print("CV Score :", scr_dt.mean())
print("-----------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-----------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
print("ROC AUC Score \n", roc_auc_score(y_test,y_pred))
```

```
F1 score
 0.9470121254961666
CV Score : 0.9485301689220229
-----------------------------------------------------

Classification Report
               precision    recall  f1-score   support

           0       0.96      0.93      0.94     35600
           1       0.93      0.97      0.95     36073

    accuracy                           0.95     71673
   macro avg       0.95      0.95      0.95     71673
weighted avg       0.95      0.95      0.95     71673

-----------------------------------------------------

Confusion Matrix
 [[32942  2658]
 [ 1240 34833]]
ROC AUC Score
 0.9454811692706768
```

```python
from sklearn.naive_bayes import MultinomialNB

mnb= MultinomialNB()
mnb.fit(x_train,y_train)
y_pred = mnb.predict(x_test)
scr_mnb = cross_val_score(mnb,x_over,y_over,cv=5)


print("F1 score \n", f1_score(y_test,y_pred))
print("CV Score :", scr_mnb.mean())
print("-----------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-----------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
print("ROC AUC Score \n", roc_auc_score(y_test,y_pred))
```

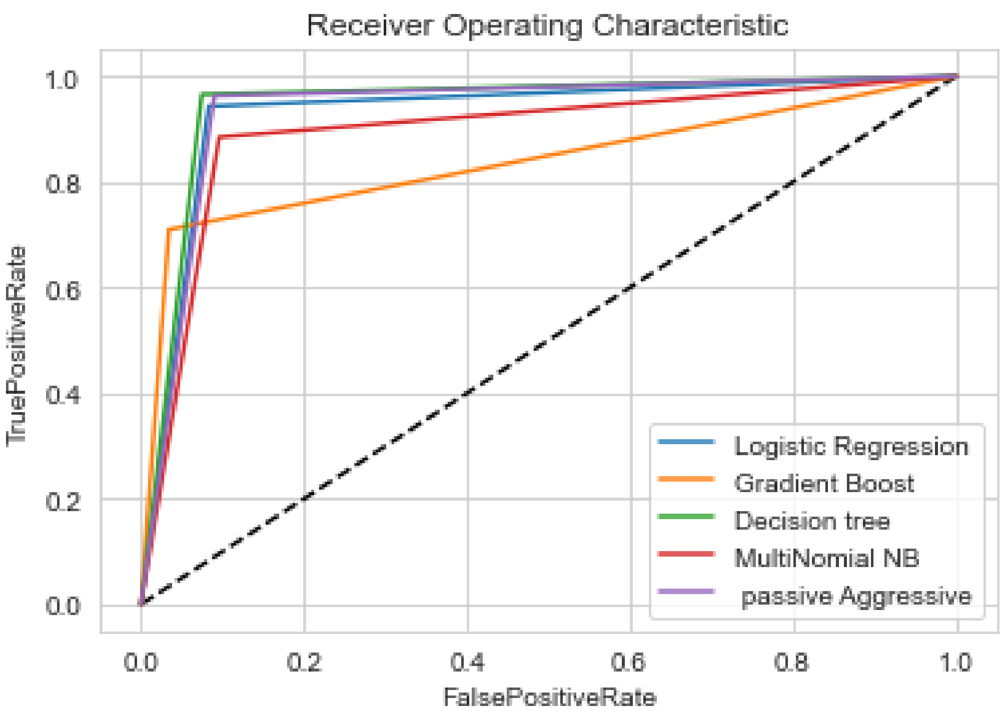
```
F1 score
 0.8940145085847128
CV Score : 0.8976567254900415
-----------------------------------------------------

Classification Report
               precision    recall  f1-score   support

           0       0.89      0.90      0.89     35600
           1       0.90      0.88      0.89     36073

    accuracy                           0.89     71673
   macro avg       0.89      0.89      0.89     71673
weighted avg       0.89      0.89      0.89     71673

-----------------------------------------------------

Confusion Matrix
 [[32186  3414]
 [ 4154 31919]]
ROC AUC Score
 0.894472872112947
```

```
from sklearn.linear_model import PassiveAggressiveClassifier

pac = PassiveAggressiveClassifier()
pac.fit(x_train,y_train)
y_pred = pac.predict(x_test)
scr_pac = cross_val_score(pac,x_over,y_over,cv=5)


print("F1 score \n", f1_score(y_test,y_pred))
print("CV Score :", scr_pac.mean())
print("-------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
print("ROC AUC Score \n", roc_auc_score(y_test,y_pred))
```

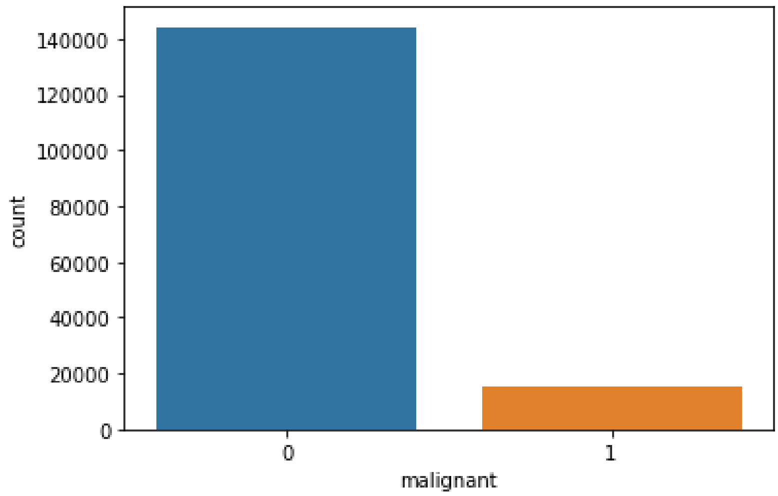
```
F1 score
 0.9387429744920017
CV Score : 0.9372672098758568
-------------------------------------------------------

Classification Report
               precision    recall  f1-score   support

           0       0.96      0.91      0.93     35600
           1       0.92      0.96      0.94     36073

    accuracy                           0.94     71673
   macro avg       0.94      0.94      0.94     71673
weighted avg       0.94      0.94      0.94     71673

--------------------------------------------------------

Confusion Matrix
 [[32398  3202]
 [ 1332 34741]]
ROC AUC Score
 0.9365655278606395
```

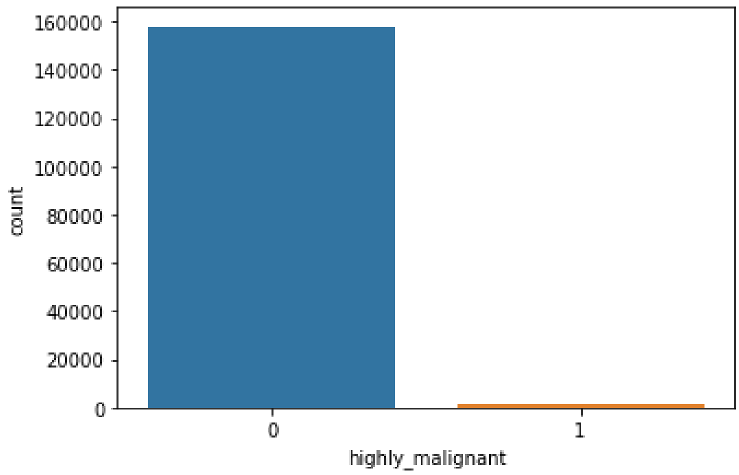
- Key Metrics for success in solving problem under consideration

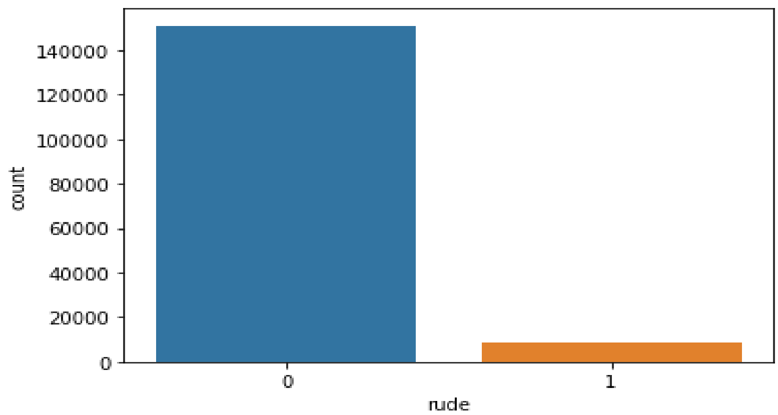
  Key Metrices used were the Accuracy Score, Cross validation Score and AUC & ROC Curve as this was binary classification as you can see in the above image in models used.

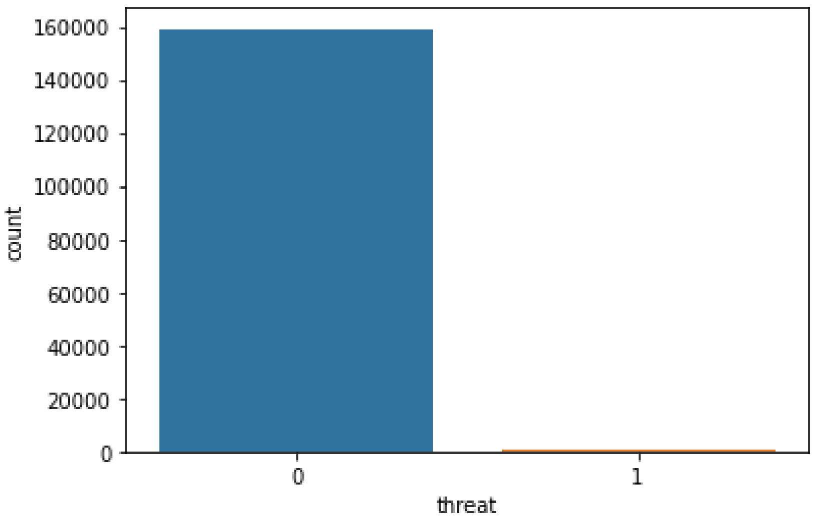


- Visualizations
  Used Count plot and distribution plot and for the different target variables.
  Heat map for test the correlation between features and variables.

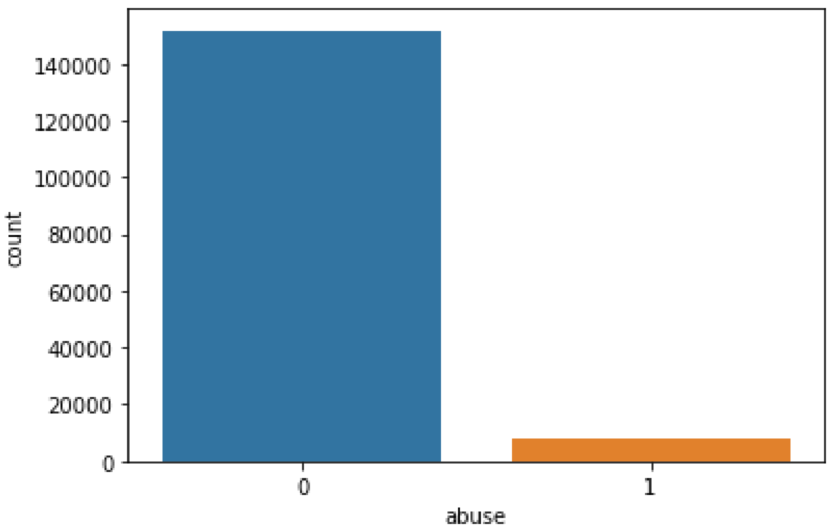




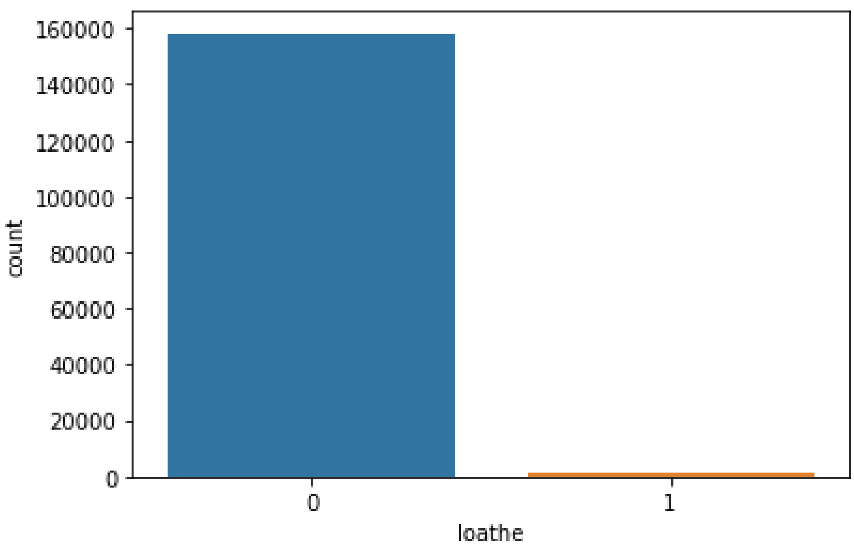




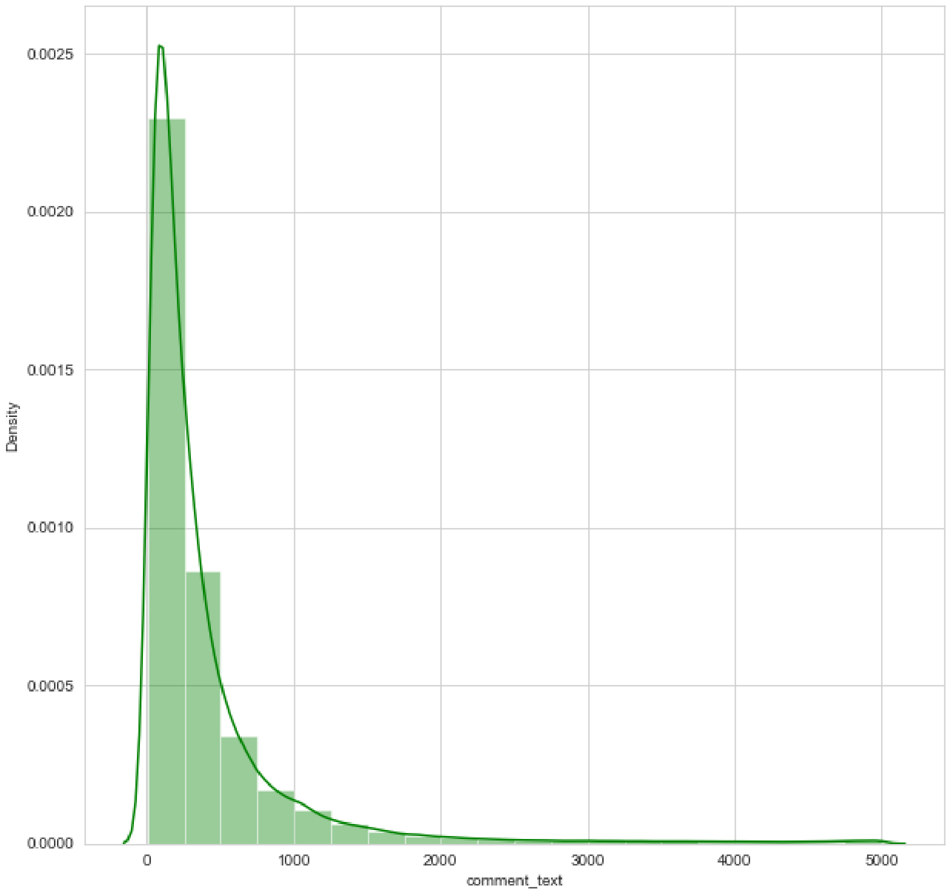
160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
threat



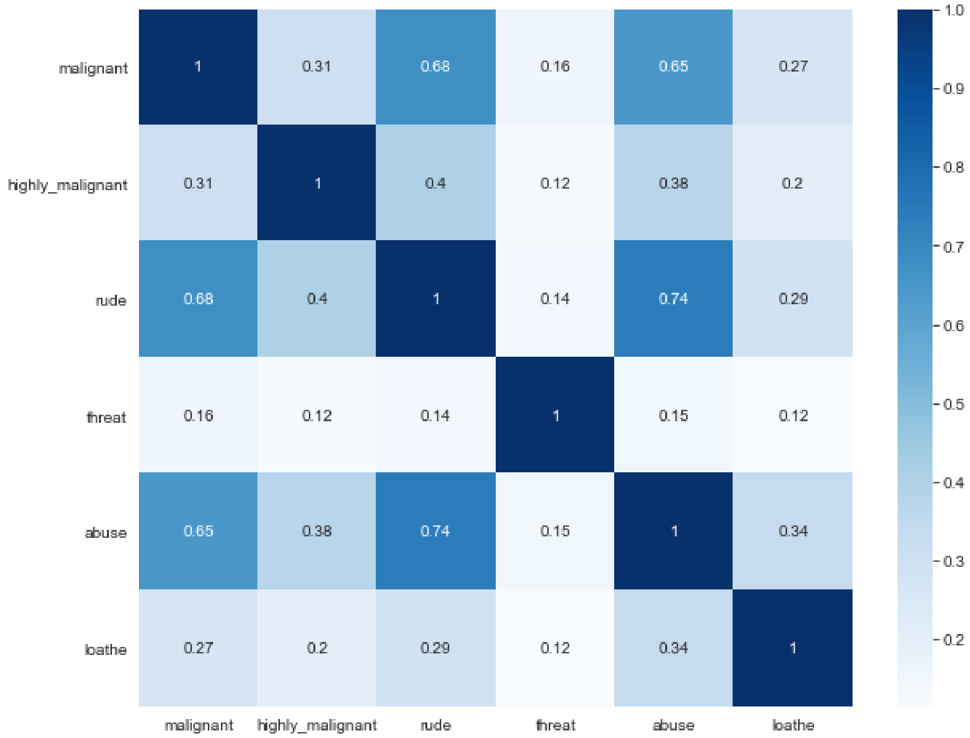
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
abuse



160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
loathe



0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Density
0
1000
2000
3000
4000
5000
comment_text



malignant
highly_malignant
rude
threat
abuse
loathe
1
0.31
0.68
0.16
0.65
0.27
0.31
1
0.4
0.12
0.38
0.2
0.68
0.4
1
0.14
0.74
0.29
0.16
0.12
0.14
1
0.15
0.12
0.65
0.38
0.74
0.15
1
0.34
0.27
0.2
0.29
0.12
0.34
1
malignant
highly_malignant
rude
threat
abuse
loathe
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

- Interpretation of the Results

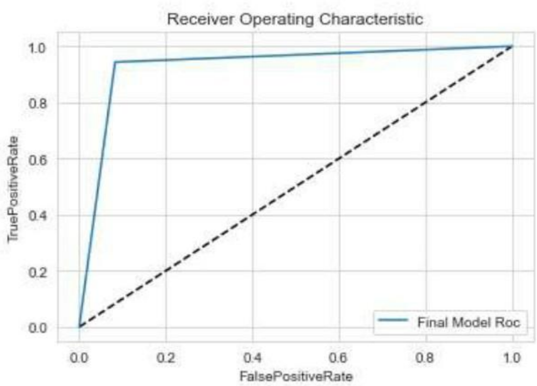
```
]: #Lets try to improve the accuracy of model by hyper parameter tuning,

param = {'C': [1.0,1.2,1.4,1.6,1.8],
        'fit_intercept':[True], 'max_iter': [1000]}

# Applying randomized search CV to increase the accuracy,

rg = RandomizedSearchCV(pac, param_distributions = param, cv= 5)
rg.fit(x_train,y_train)
rg.best_params_
```

```
]: {'max_iter': 1000, 'fit_intercept': True, 'C': 1.0}
```

```
]: #final model accuracy,

model = PassiveAggressiveClassifier(C = 1.0, max_iter = 1000, fit_intercept = Tru

model.fit(x_train,y_train)
y_pred = model.predict(x_test)


print("F1 score \n", f1_score(y_test,y_pred))
print("-------------------------------------------------------\n")
print("Classification Report \n", classification_report(y_test,y_pred))
print("-------------------------------------------------------\n")
print("Confusion Matrix \n", confusion_matrix(y_test,y_pred))
print("ROC AUC Score \n", roc_auc_score(y_test,y_pred))
```

```
F1 score
 0.9380530973451328
-------------------------------------------------------

Classification Report
               precision    recall  f1-score   support

           0       0.96      0.90      0.93     35600
           1       0.91      0.97      0.94     36073

    accuracy                           0.94     71673
   macro avg       0.94      0.94      0.94     71673
weighted avg       0.94      0.94      0.94     71673

--------------------------------------------------------

Confusion Matrix
 [[32193  3407]
 [ 1199 34874]]
ROC AUC Score
 0.9355297984237331
```

```
#Roc Curve for final model,

y_pred_fin = model.predict(x_test)
fpr , tpr, thresholds = roc_curve(y_test, y_pred_fin)


plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr1, tpr1, label= "Final Model Roc")
plt.legend()
plt.xlabel("FalsePositiveRate")
plt.ylabel("TruePositiveRate")
plt.title('Receiver Operating Characteristic')
plt.show()
```



CONCLUSION

- Key Findings and Conclusions of the Study

  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.

  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.

- Learning Outcomes of the Study in respect of Data Science

  It is possible to differentiate the comments into Malignant and Non – Malignant. However, using this project will help to create awareness among the people. It will help people to stop spreading hatred to people.

- Limitations of this work and Scope for Future Work

  This project is different than the previous project provided by Flip-Robo technologies as it is text classifier using ML techniques which is challenging.

  Models like decision tree classifier has taken more time and random forest and SVC algorithms are taking more time so, I didn’t include those algorithms.