NAME OF THE PROJECT

MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

Ram Kumar

ACKNOWLEDGMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend my sincere thanks to SME. Khushboo Garg .

We are highly indebted to Flip Robo technology for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

Thanks all.

Ram kumar

.

INTRODUCTION

- 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 ironments 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.

- 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.

Analytical Problem Framing

Import library and load the dataset

```
# Import Library:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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

```
# Loading the train  dataset:
train=pd.read_csv('Train.csv')
train.head()
```

	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

```
# Loading the test  dataset:
test=pd.read_csv('Test.csv')
test.head()
```

	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.

```
# Here checking train data shape:
train.shape
```

```
(159571, 8)
```

```
# Here checking train data shape:
test.shape
```

```
(153164, 2)
```

```
# Here checking train data info:
train.info()
```

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

- Display all column name of dataset.

```
# Here checking train data info:
train.info()
```

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

```
# Here checking test data info:
test.info()
```

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

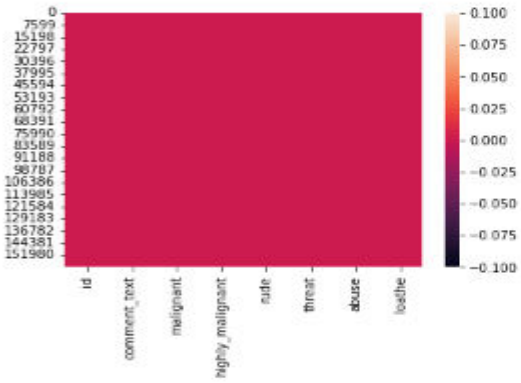
```
# Here checking train data type:

train.dtypes
```

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

dtype: int64

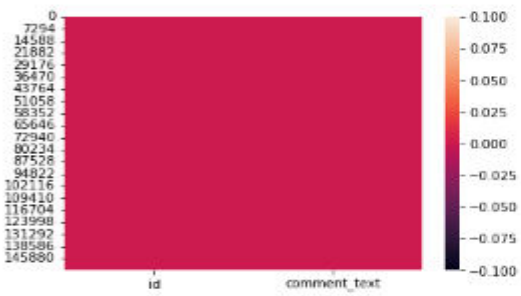
```
#Checking for null values using heatmap
sns.heatmap(train.isnull())
```

```
<AxesSubplot:>
```




```
#Checking for null values using heatmap
sns.heatmap(test.isnull())
```

```
<AxesSubplot:>
```



- Display statistical summary.

```
#summary statistics.
train.describe().style.background_gradient()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

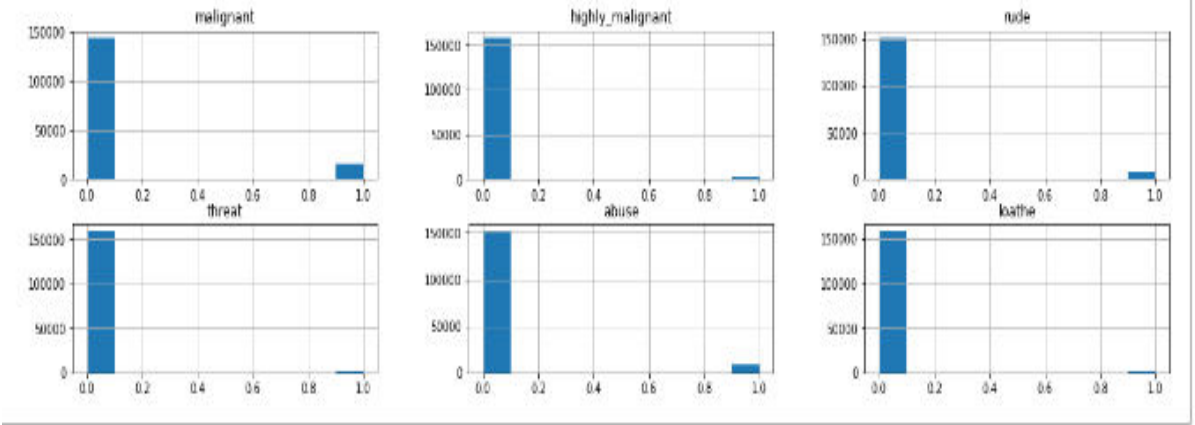
```
#summary statistics.
test.describe().style.background_gradient()
```

	id	comment_text
count	153164	153164
unique	153164	153164
top	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll ever be whats up with you and hating you sad mofuckas...i should bitch slap ur pethedic white faces and get you to kiss my ass you guys sicken me. Ja rule is about pride in da music man. dont diss that shit on him. and nothin is wrong bein like tupac he was a brother too...fuckin white boys get things right next time.,
freq	1	1

- Display histogram of all columns.

```
#display histogram of all columns.
train.hist(figsize=(20,15), layout=(6,3), sharex=False)
```

```
array([[<AxesSubplot:title={'center':'malignant'}>,
        <AxesSubplot:title={'center':'highly_malignant'}>,
        <AxesSubplot:title={'center':'rude'}>],
       [<AxesSubplot:title={'center':'threat'}>,
        <AxesSubplot:title={'center':'abuse'}>,
        <AxesSubplot:title={'center':'loathe'}>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
       [<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



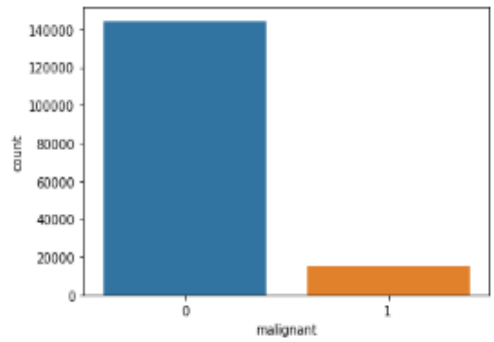
- Display correlation of columns using heatmap.

```
]: #check correlation matrix with heatmap.
corr_mat = train.corr()
plt.figure(figsize=(15,8))
sns.heatmap(corr_mat, annot=True, fmt = '.1f',cmap = 'Blues')
plt.title('Correlation matrix')
plt.show()
```

- Display barplot of all columns.

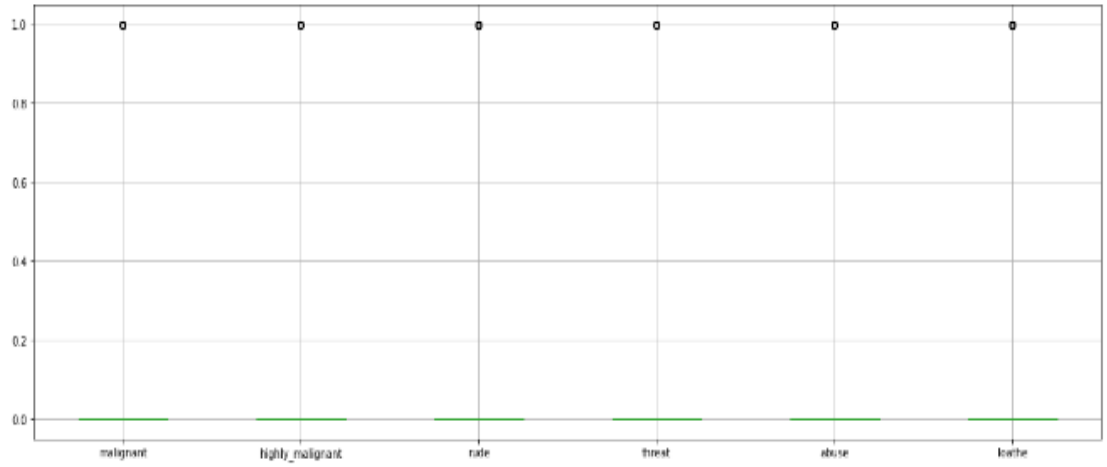
```
: col=['malignant','highly_malignant','loathe','rude','abuse','threat']
for i in col:
    print(i)
    print("\n")
    print(train[i].value_counts())
    sns.countplot(train[i])
    plt.show()
```

```
0    144277
1     15294
Name: malignant, dtype: int64
```



- Display outliers of all columns.

```
#checking outliers with boxplot.
train.iloc[:,1:8].boxplot(figsize=[20,8])
plt.subplots_adjust(bottom=0.25)
plt.show()
```

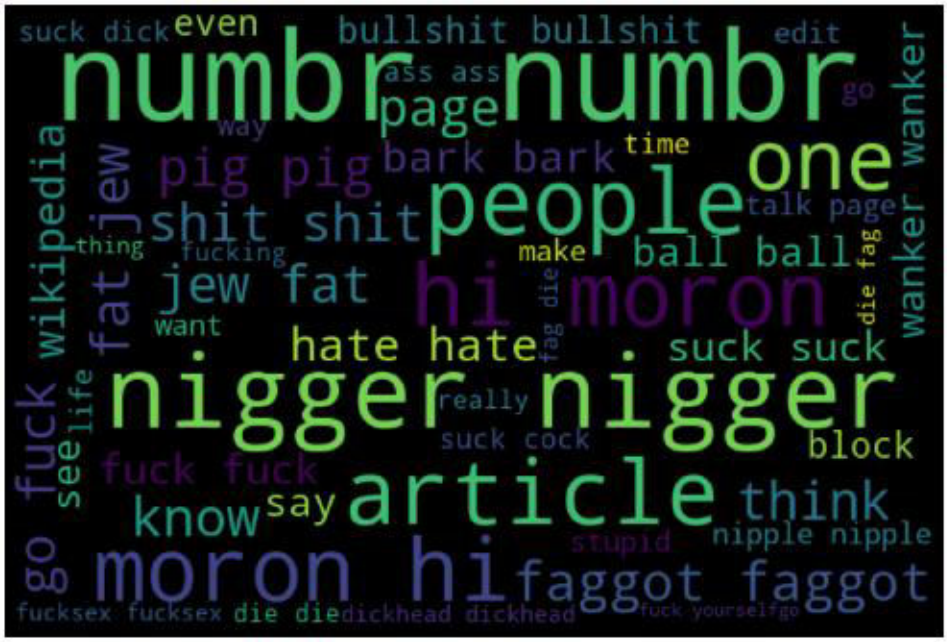


- filling empty values and Replace values.

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

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

# Replace URLs with 'webaddress'
train['comment_text'] = train['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)
train['comment_text'] = train['comment_text'].str.replace(r'£|\$', 'dollers')

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


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


train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in string.punctuation))

stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

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

- **Getting sense of loud words which are offensive**

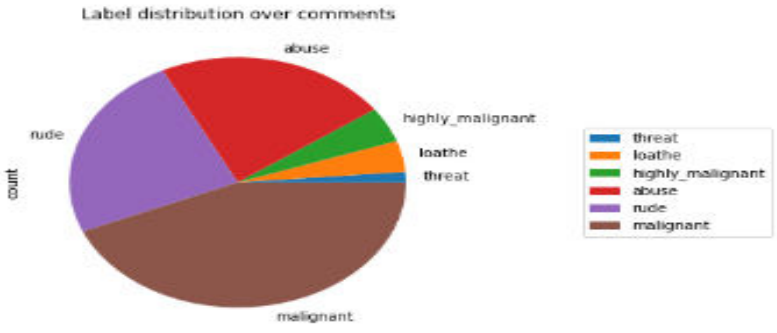
```
#Getting sense of loud words which are offensive
from wordcloud import WordCloud
hams = train['comment_text'][train['malignant']==1]
spam_cloud = WordCloud(width=600,height=400,background_color='black',max_words=50).generate(' '.join(hams))
plt.figure(figsize=(10,8),facecolor='k')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



- **Display distribution values:**

```
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.pie(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 0x1937da0c130>
```



Model/s Development and Evaluation

- **Feature engeenering:**

```
#  Convert text into vectors using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(train['comment_text'])
x = features
```

```
train.shape
test.shape
```

```
(153164, 2)
```

```
y=train['bad']
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.30)
```

```
y_train.shape,y_test.shape
```

```
((111699,), (47872,))
```

- Testing of Identified Approaches (Algorithms)

```
# LogisticRegression
LG = LogisticRegression(C=1, max_iter = 3000)

LG.fit(x_train, y_train)

y_pred_train = LG.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = LG.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9595520103134316
Test accuracy is 0.9552974598930482
[[42729   221]
 [ 1919  3003]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.93      0.61      0.74      4922

    accuracy                           0.96     47872
   macro avg       0.94      0.80      0.86     47872
weighted avg       0.95      0.96      0.95     47872
```

- Run and Evaluate selected models

```
weighted avg       0.91      0.92      0.89     47872
```

```
]: # RandomForestClassifier
RF = RandomForestClassifier()
RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
cvs=cross_val_score(RF, x, y, cv=10, scoring='accuracy').mean()
print('cross validation score :',cvs*100)
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988540631518635
Test accuracy is 0.9552347927807486
cross validation score : 95.67903834298129
[[42413   537]
 [ 1606  3316]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.67      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.83      0.87     47872
weighted avg       0.95      0.96      0.95     47872
```

- Creating RMSE:

```
: #display MAE, MSE and RMSE:
from sklearn.metrics import mean_squared_error, mean_absolute_error
print('error:')

print('mean absolute error',mean_absolute_error(y_test,y_pred_test))
print('mean squared error',mean_squared_error(y_test,y_pred_test))

print('Root mean squared error',np.sqrt(mean_squared_error(y_test,y_pred_test)))
```

```
error:
mean absolute error 0.044765207219251334
mean squared error 0.044765207219251334
Root mean squared error 0.21157789870223857
```

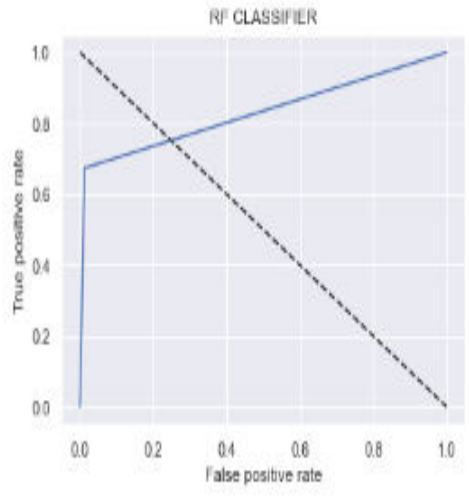
- Decision Tree:

```
# DecisionTreeClassifier
DT = DecisionTreeClassifier()

DT.fit(x_train, y_train)
y_pred_train = DT.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = DT.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988898736783678
Test accuracy is 0.940194685828877
[[41627  1323]
 [ 1540  3382]]
              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      0.83     47872
weighted avg       0.94      0.94      0.94     47872
```

- Interpretation of the Results

```
#Plotting the graph which tells us about the area under curve , more the area under curve more will be the better prediction
# model is performing good :
fpr,tpr,thresholds=roc_curve(y_test,y_pred_test)
roc_auc=auc(fpr,tpr)
plt.plot([0,1],[1,0],'k--')
plt.plot(fpr,tpr,label = 'RF Classifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('RF CLASSIFIER')
plt.show()
```



- **Prediction of test dataset:**

```
test_data =tf_vec.fit_transform(test['comment_text'])
test_data
```

```
<153164x10000 sparse matrix of type '<class 'numpy.float64'>'
        with 2940344 stored elements in Compressed Sparse Row format>
```

```
prediction=RF.predict(test_data)
prediction
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
import joblib
joblib.dump(RF,"malig.pkl")
```

```
['malig.pkl']
```

- Hardware and Software Requirements and Tools Used

- **Language :-** Python
- **Tool:-** Jupyter
- **OS:-** Windows 10
- **RAM:-** 8gb

CONCLUSION

- 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.
- 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.