FLIP ROBO

Malignant Comments Classifier

Submitted By:- **Dhrubajyoti Mandal**

Acknowledgment:-

- *First, I would like to express my gratitude towards Flip Robo Technologies for their kind co- operation and encouragement which help me in completion of this project.*

- *I would like to express my special gratitude and thanks to industry persons and my mentor Miss. Sapna Verma for giving me such attention and time as and whenever required.*

Contents:-

Problem Statement:-

- 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.
- 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.

Features:-

1. **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
2. **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
3. **Rude:** It denotes comments that are very rude and offensive.
4. **Threat:** It contains indication of the comments that are giving any threat to someone.
5. **Abuse:** It is for comments that are abusive in nature.
6. **Loathe:** It describes the comments which are hateful and loathing in nature.
7. **ID:** It includes unique Ids associated with each comment text given.
8. **Comment text:** This column contains the comments extracted from various social media platforms.

Data Processing:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

```
df_train = pd.read_csv(r"C:\Users\Dhruv\Data Science with Python\Flip Robo Internship\010. Malignant-Comments-Classifier\Malignar
df_test = pd.read_csv(r"C:\Users\Dhruv\Data Science with Python\Flip Robo Internship\010. Malignant-Comments-Classifier\Malignant
```

Here, I have imported the libraries, and loaded the dataset.

```
df_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

```
df_train.drop(columns = {'id'}, inplace = True)
```

```
df_train.shape
```

```
(159571, 7)
```

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   comment_text      159571 non-null  object
 1   malignant         159571 non-null  int64
 2   highly_malignant  159571 non-null  int64
 3   rude              159571 non-null  int64
 4   threat            159571 non-null  int64
 5   abuse             159571 non-null  int64
 6   loathe            159571 non-null  int64
dtypes: int64(6), object(1)
memory usage: 8.5+ MB
```

```
1]: df_train['comment_text'].value_counts()
```

```
1]: Explanation\nWhy the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure o
n some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.8
9.205.38.27
1
John Barbirolli \n\nWarm thanks for promoting the article and for your kind words about it. At your service for GA, peer review
or FA etc if I can be of any use. -
1
"\n\nHe is an Alumnus, a scientist and an Interaction Design pioneer.  If you want to reword the article, by all means User:Kma
rkey  call Carnegie Mellon for verification.  Also, if you read Category:Suspected_Wikipedia_sockpuppets_of_Young_Zaphod it men
tions ""Herb Gilliland"" because of an article involving Mr. Gilliland.  \n "
1
Reply on my talk page. -
1
"\n\n Some tips for you! \n\nHi , I thought I'd drop a few notes on your talk page with some help on writing articles o)\n\nFir
st of all, it may be best for you to do a bit of reading, starting with the Wikipedia manual of style, which will give you a lo
t of information about how Wikipedia prefers its articles to be written.  It's not as hard to follow as it might look; quite a
bit of the information there probably won't be vital for you at first.\n\nSecond, I recommend you make a user sandbox - which i
s just an area you can use to practise in, and to make notes in, and to get things ready in.  If you click this red link: use
r:/Sandbox, that will let you create that page (it gives you an edit window to start work in).  Anything, anywhere, on the help
and information pages which gives you an example, try it out in your sandbox until you're familiar with it.\n\nFor your articl
e, the next thing you want to do is start collecting as much information as you can about it.  Google searches (particularly in
Books and Scholar) will be your best friend for this!  Once you've found the information, the next most important thing is to s
tart writing up each fact in your own words (very important, this), and make a note at the same time of exactly where that info
rmation came from.  Build in the references as you go along; I'm going to copy in, down below this, a whole heap of help on doi
ng references, which was produced by one of our best teachers ().\n\nHere's another place that you'll find incredibly useful -
citation templates which you can copy and paste into your sandbox, between  tags; you just fill in the blanks from your sources
into the template, and you'll end up with nicely formatted inline citations o)  It all helps.  Remember to add a references sec
tion to your sandbox (make a new line, and put ==References== on it, and type {{reflist}} on the next line, so that you can see
how your citations look as you do them.  Remember to save your page often! You don't want to lose your work.\n\nHopefully this
will give you a good start and make life easier for you.  (talk ♪stalk!) \nHow references work\n\n Simple references \n\nThese
require two parts;\n\na)\nChzz is 98 years old. ""The book of Chzz"", Aardvark Books, 2009. \n\nHe likes tea.  [http://www.nice
cupofteaandasitdown.com Tea website] \n\nb) A section called ""References"" with the special code ""{{reflist}}"";\n== Referenc
es ==\n{{reflist}}\n\n(an existing article is likely to already have one of these sections)\n\nTo see the result of that, pleas
e look at user:chzz/demo/simpleref. Edit it, and check the code; perhaps make a test page of your own, such as user:/reftest an
d try it out.\n\n Named references \n\nChzz was born in 1837. \n""The book of Chzz"", Aardvark Books, 2009. \n \n\nChzz lives i
n Footown.\n\nNote that the second usage has a / (and no closing ref tag). This needs a reference section as above; please see
user:chzz/demo/namedref to see the result.\n\n Citation templates \n\nYou can put anything you like between  and , but using ci
tation templates makes for a neat, consistent look;\n\nChzz has 37 Olympic medals.  {{Citation\n | last = Smith\n | first = Joh
n\n | title = Olympic medal winners of the 20th century\n | publication-date = 2001\n | publisher = [[Cambridge University Pres
s]]\n | page = 125\n | isbn = 0-521-37169-4\n}}\n\n\nPlease see user:chzz/demo/citeref to see the result.\n\nFor more help and
tips on that subject, see user:chzz/help/refs.\nSomething to make your life easier!\n\nHi there !  I've just come across one of
your articles, and noticed that you had to create titles for your url links manually, or were using bare urls as references.\n
\nYou  might want to consider using this tool - it makes your life a whole heap easier, by filling in complete citation templat
es for your links.  All you do is install the script on Special:MyPage/common.js, or or    1
                                                                                                                ..
, which also mean that I will avoid edit-warring as WP:DRD says
1
nutcase \n\nyou're a sick bloody nutcase
1
"\n\nDon't worry about the above too much - but copyrighted files like this can't be used when it's ""easy"" to get free shots.
That's why so many articles on living people don't have an image, they're waiting for free images to appear.  You also need to
find some more sources about Kelly - IMDb isn't classed as a very reliable source because like wikipedia most of it is user gen
erated content, so it isn't verifiably correct.   "
1
please reffer tkz.jpg
1
"\nAnd ... I really don't think you understand.  I came here and my idea was bad right away.  What kind of community goes ""you
have bad ideas"" go away, instead of helping rewrite them.   "
1
Name: comment_text, Length: 159571, dtype: int64
```

```
df_train['malignant'].value_counts()
```

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

```
df_train['highly_malignant'].value_counts()
```

```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```

```
df_train['rude'].value_counts()
```

```
0    151122
1      8449
Name: rude, dtype: int64
```

```
df_train['threat'].value_counts()
```

```
0    159093
1       478
Name: threat, dtype: int64
```

```
df_train['abuse'].value_counts()
```

```
0    151694
1      7877
Name: abuse, dtype: int64
```

```
df_train['loathe'].value_counts()
```

```
0    158166
1      1405
Name: loathe, dtype: int64
```

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

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

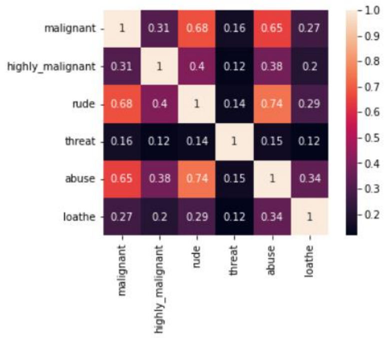
```
df_train.describe().T
```

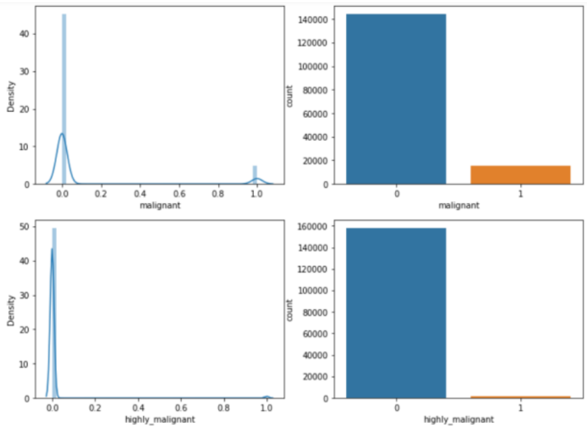
	count	mean	std	min	25%	50%	75%	max
malignant	159571.0	0.095844	0.294379	0.0	0.0	0.0	0.0	1.0
highly_malignant	159571.0	0.009996	0.099477	0.0	0.0	0.0	0.0	1.0
rude	159571.0	0.052948	0.223931	0.0	0.0	0.0	0.0	1.0
threat	159571.0	0.002996	0.054650	0.0	0.0	0.0	0.0	1.0
abuse	159571.0	0.049364	0.216627	0.0	0.0	0.0	0.0	1.0
loathe	159571.0	0.008805	0.093420	0.0	0.0	0.0	0.0	1.0

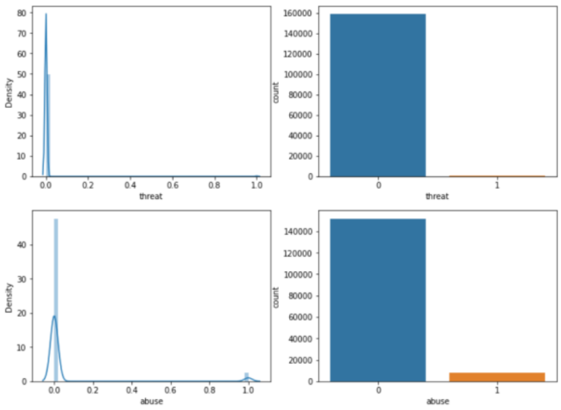
```
# Eda
df_train.corr()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128
abuse	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736
loathe	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000

```python
sns.heatmap(df_train.corr(), annot = True, square = True)
plt.show()
```







Data Preprocessing

```python
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize, regexp_tokenize
import string
```

```python
df_train['length']=df_train['comment_text'].str.len()
df_train.head(2)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112

```python
# convert to lower
df_train['comment_text']=df_train['comment_text'].str.lower()
```

```python
# replace email address
df_train['comment_text']=df_train['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddr')
```

```python
# replace web address
df_train['comment_text']=df_train['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddress')
```

```python
# replace money symbols
df_train['comment_text']=df_train['comment_text'].str.replace(r'£|\$', 'moneysymb')
```

```python
# replace 10 digit phone numbers with 'phonenumber'
df_train['comment_text']=df_train['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumbr')
```

```python
# replace normal numbers with 'numbr'
df_train['comment_text']=df_train['comment_text'].str.replace(r'\d+(\.\d+)?','numbr')
```

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

```
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_words))
```

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

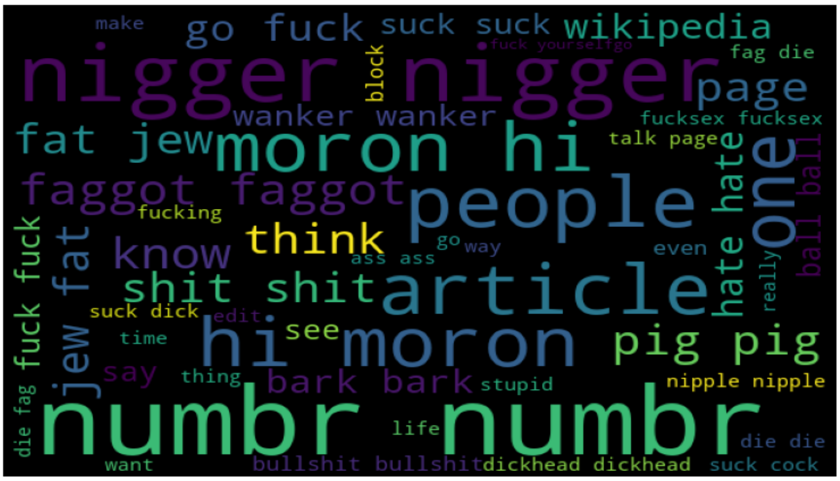
```
df_train['clean_length'] = df_train.comment_text.str.len()
df_train.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	180
1	d'aww! match background colour i'm seemingly s...	0	0	0	0	0	0	112	111
2	hey man, i'm really trying edit war. guy const...	0	0	0	0	0	0	233	149
3	can't make real suggestion improvement wondere...	0	0	0	0	0	0	622	397
4	you, sir, hero. chance remember page that's on?	0	0	0	0	0	0	67	47

```
# Total length removal
print ('Original Length', df_train.length.sum())
print ('Clean Length', df_train.clean_length.sum())
```

```
Original Length 62893130
Clean Length 43577387
```

```
#Analysing of Loud words which are offensive,hurting and not accpectable
from wordcloud import WordCloud
hams = df_train['comment_text'][df_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()
```




```
df_test['length'] = df_test['comment_text'].str.len()
df_test.head(2)
```

	id	comment_text	length
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50

```
# Convert to lower
df_test['comment_text']=df_test['comment_text'].str.lower()

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

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

# Replace money symbols
df_test['comment_text'] = df_test['comment_text'].str.replace(r'£|\$', 'dollers')

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

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

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

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


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

	id	comment_text	length	clean_length
0	00001cee341fdb12	yo bitch ja rule succesful ever whats hating s...	367	249
1	0000247867823ef7	== rfc == title fine is, imo.	50	29
2	00013b17ad220c46	== source == zawe ashton lapland —	54	34
3	00017563c3f7919a	:if look back source, information updated corr...	205	117
4	00017695ad8997eb	anonymously edit article all.	41	29

```
print ('Origial Length:', df_test.length.sum())
print ('Clean Length:', df_test.clean_length.sum())
```

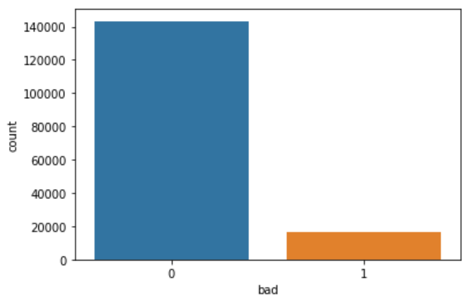
```
Origial Length: 55885733
Clean Length: 38993729
```

```
# importing important libraries

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier


from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,roc_curve,roc_auc_score,auc,f1_score
from sklearn.model_selection import cross_val_score,GridSearchCV
```

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

df_train['bad'] =df_train[target_columns].sum(axis =1)
print(df_train['bad'].value_counts())
df_train['bad'] = df_train['bad'] > 0
df_train['bad'] = df_train['bad'].astype(int)
print(df_train['bad'].value_counts())
```

```
0    143346
1      6360
3      4209
2      3480
4      1760
5       385
6        31
Name: bad, dtype: int64
0    143346
1     16225
Name: bad, dtype: int64
```

```
sns.countplot(df_train['bad'])
plt.show()
```



```
#  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(df_train['comment_text'])
x = features
```

```
df_train.shape
```

```
(159571, 10)
```

```
df_test.shape
```

```
(153164, 4)
```

```
y=df_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,))
```

```
# Logistic Regression
LG = LogisticRegression()
#for trainoing data
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)))

# for testing data
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 0.9595967734715619
Test accuracy is 0.9553392379679144
[[42729   221]
 [ 1917  3005]]
              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
```

```
# DecisionTree Regression
DTC = DecisionTreeClassifier()
#for trainoing data
DTC.fit(x_train, y_train)
y_pred_train = DTC.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))

# for testing data
y_pred_test = DTC.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 0.939651570855615
[[41593  1357]
 [ 1532  3390]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.71      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
```

```
# KNeighborsClassifier
knn = KNeighborsClassifier()
#for trainoing data
knn.fit(x_train, y_train)
y_pred_train = knn.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))

# for testing data
y_pred_test = knn.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 0.9296591733139956
Test accuracy is 0.9181567513368984
[[42604   346]
 [ 3572  1350]]
              precision    recall  f1-score   support

           0       0.92      0.99      0.96     42950
           1       0.80      0.27      0.41      4922

    accuracy                           0.92     47872
   macro avg       0.86      0.63      0.68     47872
weighted avg       0.91      0.92      0.90     47872
```

```python
# Random Forest Regression
RF = RandomForestClassifier()
#for trainoing data
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)))

# for testing data
y_pred_test = RF.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 0.9988272052569853
Test accuracy is 0.9546916778074866
[[42400   550]
 [ 1619  3303]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.67      0.75      4922

    accuracy                           0.95     47872
   macro avg       0.91      0.83      0.86     47872
weighted avg       0.95      0.95      0.95     47872
```

```python
# AdaBoostClassifier Regression
ada = AdaBoostClassifier()
#for trainoing data
ada.fit(x_train, y_train)
y_pred_train = ada.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))

# for testing data
y_pred_test = ada.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 0.9463737365598618
Test accuracy is 0.9454169451871658
[[42587   363]
 [ 2250  2672]]
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     42950
           1       0.88      0.54      0.67      4922

    accuracy                           0.95     47872
   macro avg       0.92      0.77      0.82     47872
weighted avg       0.94      0.95      0.94     47872
```

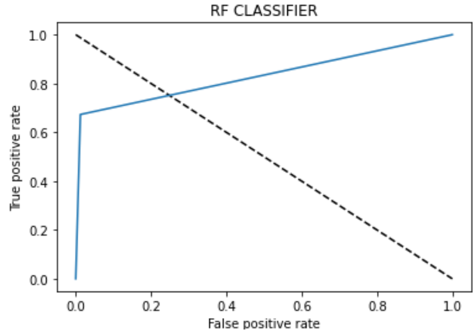
```python
# xgboost Regression
xgb = XGBClassifier()
#for trainoing data
xgb.fit(x_train, y_train)
y_pred_train = xgb.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))

# for testing data
y_pred_test = xgb.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 0.9614410155865316
Test accuracy is 0.9526445521390374
[[42686   264]
 [ 2003  2919]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     42950
           1       0.92      0.59      0.72      4922

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

```python
# Hypertuning the model with Random forest Classifier:
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=5, 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 0.9553183489304813
cross validation score : 95.65522512911213
[[42420   530]
 [ 1609  3313]]
              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
```

```python
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()
```




```python
df_test['malignant']=RF.predict(test_data)
df_test['highly_malignant']=RF.predict(test_data)
df_test['rude']=RF.predict(test_data)
df_test['threat']=RF.predict(test_data)
df_test['abuse']=RF.predict(test_data)
df_test['loathe']=RF.predict(test_data)
df_test[['id','comment_text','malignant','highly_malignant','rude','threat','abuse','loathe']].to_csv('Malignant_comment_submissi
```

```python
submission = pd.read_csv(r'Malignant_comment_submission.csv')
submission.shape
```

```
(153164, 8)
```

```python
import joblib
joblib.dump(RF,"MalignantComment Prediction.pkl")
```

```
['MalignantComment Prediction.pkl']
```

```python
submission.sample(10)
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
124981	d0c854ca11363c46	please vandalize pages, edit tastykake. contin...	0	0	0	0	0	0
54483	5a8e28b38ed9a3c8	:as writes article page, green color mean ""vi...	0	0	0	0	0	0
148233	f7bcdb53f87dee1d	== test ==	0	0	0	0	0	0
109976	b772f5fdba830c5e	thats stupid thing i've ever heard. take score...	0	0	0	0	0	0
77404	811648bf4e4b5fe4	== vagina == eat taste good	0	0	0	0	0	0
21728	24443a4d6e551d80	block evading sock puppet dalai lama ding dong	1	1	1	1	1	1
147388	f65a735450cc8359	== sorry == mean link oasis academy: mediacity...	0	0	0	0	0	0
37977	3f03ec0668a96204	:good idea. i've sorely tempted tell dick it. —	0	0	0	0	0	0
80966	872309c71bc8f3eb	== hate wikipedia!!!! == article accent (music...	0	0	0	0	0	0
45866	4c192f735ed70026	sockpuppet== evidence incontrovertible, there'...	0	0	0	0	0	0

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.
- 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

It is possible to classify the comments content into the required categories of Malignant and Non Malignant. However, using this kind of project an awareness can be created to know what is good and bad. It will help to stop spreading hatred among people.

Limitations of this work and Scope for Future Work

- Machine Learning Algorithms like Decision Tree Classifier took enormous amount of time to build the model and Ensemble techniques were taking a lot more time thus I have not included Ensemble models.
- Using Hyper-parameter tuning would have resulted in some more accuracy.
- Every effort has been put on it for perfection but nothing is perfect and this project is of no exception. There are certain areas which can be enhanced. Comment detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field.

Thank You