PROJECT NAME:-

MALIGNANT COMMENTS CLASSIFIER

SUBMITTED BY:-

ATUL DAHIMA

INTRODUCTION

PEOBLEM 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.

Data Set Description

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:

- **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
- **Rude:** It denotes comments that are very rude and offensive.
- **Threat:** It contains indication of the comments that are giving any threat to someone.
- **Abuse:** It is for comments that are abusive in nature.
- **Loathe:** It describes the comments which are hateful and loathing in nature.
- **ID:** It includes unique Ids associated with each comment text given.

- **Comment text:** This column contains the comments extracted from various social media platforms.

START MODEL BUILDING:-

- ***Mathematical/ Analytical Modeling of the Problem***

```
In [1]: #import all the required libraries
        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [3]: train_data=pd.read_csv(r"C:\Users\ankus\Downloads\train.csv")
        print("train data:-",train_data)

        test_data=pd.read_csv(r"C:\Users\ankus\Downloads\test.csv")
        print("test data:-",test_data.head)
```

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

        malignant  highly_malignant  rude  threat  abuse  loathe
0               0                 0     0       0      0       0
1               0                 0     0       0      0       0
2               0                 0     0       0      0       0
3               0                 0     0       0      0       0
4               0                 0     0       0      0       0
...           ...               ...   ...     ...    ...     ...
159566          0                 0     0       0      0       0
159567          0                 0     0       0      0       0
```

We will start the model building with importing some important required libraries like:- Pandas, Numpy, Seaborn ,Matplotlib etc. and import the datset using pandas (train and test dataset separately)

Few libraries are used initially for different purpose like:-

- *Numpy*:- for mathematical operation
- *Pandas*:- for data preprocessing
- *Matplotlib and Seaborn*:- for data visualization

In [4]:
```
print("train data info:-",train_data.info())
print("test data info:-",test_data.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
train data info:- None
<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
test data info:- None
```

In [5]:
```
print("train data:-",train_data.shape)
print("test data:-",test_data.shape)
```

```
train data:- (159571, 8)
test data:- (153164, 2)
```

After importing the dataset we will check the information about both the dataset i.e. how many values does a column have, null values in the columns, data type of the columns and some other information

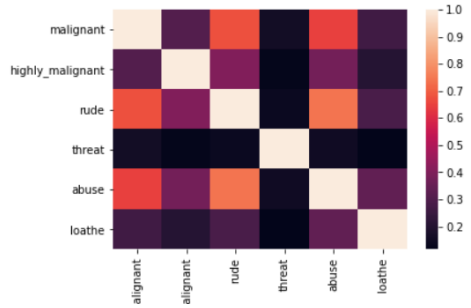
In [6]:
```
#data description
print("train data description:=",train_data.describe())
print("test data description:-",test_data.describe())
```

```
train data description:=             malignant  highly_malignant           rude         threat  \
count  159571.000000     159571.000000  159571.000000  159571.000000
mean        0.095844          0.009996       0.052948       0.002996
std         0.294379          0.099477       0.223931       0.054650
min         0.000000          0.000000       0.000000       0.000000
25%         0.000000          0.000000       0.000000       0.000000
50%         0.000000          0.000000       0.000000       0.000000
75%         0.000000          0.000000       0.000000       0.000000
max         1.000000          1.000000       1.000000       1.000000

              abuse         loathe
count  159571.000000  159571.000000
mean        0.049364       0.008805
std         0.216627       0.093420
min         0.000000       0.000000
25%         0.000000       0.000000
50%         0.000000       0.000000
75%         0.000000       0.000000
max         1.000000       1.000000
test data description:-                       id                                       comment_text
count             153164                                                 153164
unique            153164                                                 153164
top     6e69f6e6474d8135  == St. Volodymyr's Cathedral == \n\n Dear Davi...
freq                   1                                                      1
```

After this check the data description which tells the mathematical information about the dataset i.e. mean , median , mode of the columns , std of the columns and some other values like 25th%, 50th%, 75th% of the columns which helps further in the model building

```
In [7]: print(train_data.corr())
        print(sns.heatmap(train_data.corr()))
```

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

                 loathe
malignant        0.266009
highly_malignant 0.201600
rude             0.286867
threat           0.115128
abuse            0.337736
loathe           1.000000
AxesSubplot(0.125,0.125;0.62x0.755)
```

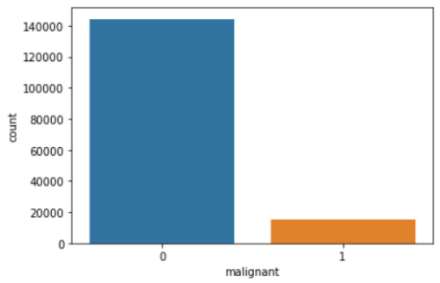


Now plot the correlation graph for all the colums using heatmap which show the correlation among all the columns and tells how the columns are correlated to each other .

If any columns are highly correlated to each other we can remove one of those columns

```
In [9]: col=['malignant','highly_malignant','loathe','rude','abuse','threat']
        for i in col:
            print(i)
            print("\n")
            print(train_data[i].value_counts())
            sns.countplot(train_data[i])
            plt.show()
```

```
  warnings.warn(
```




```
highly_malignant
```

In [10]:
```
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
```

In [11]:
```
train_data['length'] = train_data['comment_text'].str.len()
train_data.head()
```

Out[11]:

	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'awww! 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

In above code we again import some libraries like WordLemmatizer for lemmatization and import nltk and download stopwords using nltk and import string

In [12]:
```
# Convert all messages to lower case
train_data['comment_text'] = train_data['comment_text'].str.lower()

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

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

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


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


train_data['comment_text'] = train_data['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_data['comment_text'] = train_data['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

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

In above code we change the case of the text and replace some special character , hyperlinks and numbers with their respectives to make the code much easier and shorter than before

In [13]:
```
train_data['clean_length'] = train_data.comment_text.str.len()
train_data.head()
```

Out[13]:

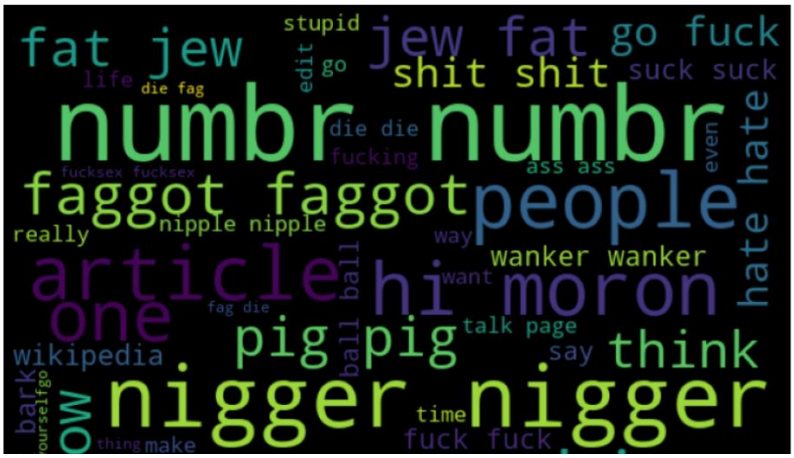
	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length	clean_length
0	0000997932d777bf	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	180
1	000103f0d9cfb60f	d'aww! match background colour i'm seemingly s...	0	0	0	0	0	0	112	111
2	000113f07ec002fd	hey man, i'm really trying edit war. guy const...	0	0	0	0	0	0	233	149
3	0001b41b1c6bb37e	can't make real suggestion improvement wondere...	0	0	0	0	0	0	622	397
4	0001d958c54c6e35	you, sir, hero. chance remember page that's on?	0	0	0	0	0	0	67	47

In [14]:
```
# Total length removal
print ('Origian Length', train_data.length.sum())
print ('Clean Length', train_data.clean_length.sum())
```

```
Origian Length 62893130
Clean Length 43575187
```

Here we check the original length and clean length after doing some cleaning on the comment_text column

In [16]:
```
#Getting sense of loud words which are offensive
from wordcloud import WordCloud
hams = train_data['comment_text'][train_data['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()
```

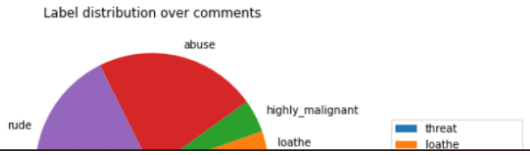


Here we import the wordcloud to check the offensive word used oftenly ,the bigger size shows used more frequently and lower the size of the word shows that the word is used less

In [17]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,roc_curve,roc_auc_score,auc
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

In [18]:
```
cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df_distribution = train_data[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))
```

Out[18]: <matplotlib.legend.Legend at 0x1c2b1da3400>



Now after import the classification model and other required libraries start doing the model building pn the train dataset

And the pie graph show the ratio of the malignant, highly malignant, nude , bad , loathe comments

In [19]:
```
target_data = train_data[cols_target]

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

```
In [21]: #  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_data['comment_text'])
         x = features

In [22]: train_data.shape

Out[22]: (159571, 11)

In [23]: test_data.shape

Out[23]: (153164, 2)

In [24]: y=train_data['bad']
         x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=56,test_size=.30)

In [25]: y_train.shape,y_test.shape

Out[25]: ((111699,), (47872,))
```

In the above code we convert the text into vectors using Tf-idf vectorizer

Now start implementing the classification model for predicting that malignant comment or not

Logistic Regression:-

Logistic Regression

```
In [26]: # 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 0.9553183489304813
[[42729   221]
 [ 1918  3004]]
              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
```

Decision Tree Classifier:-

Decision Tree Classifier

In [27]:
```
# 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 0.9406333556149733
[[41652  1298]
 [ 1544  3378]]
              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
```

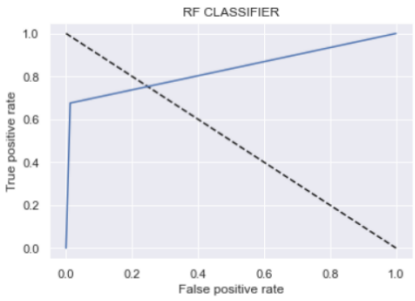
Random forest classifier:-

Random Forest Classifier

In [28]:
```
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)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988630157834896
Test accuracy is 0.9552347927807486
[[42404   546]
 [ 1597  3325]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.68      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
```

In [29]:
```
#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()
```



In [30]:
```
test_data =tf_vec.fit_transform(test_data['comment_text'])
test_data
```

Out[30]:
```
<153164x10000 sparse matrix of type '<class 'numpy.float64'>'
        with 2940344 stored elements in Compressed Sparse Row format>
```

In [31]:
```
prediction=RF.predict(test_data)
prediction
```

Out[31]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [32]:
```
import joblib
joblib.dump(RF,"malignant comments classifier.pkl")
```

Out[32]:
```
['malignant comments classifier.pkl']
```

Finally we will predict for the test dataset using the model we build on the train dataset and then save the model