FLIP ROBO

Malignant comments project

Submitted by:

Yogesh Mudliar

ACKNOWLEDGMENT

It is a genuine pleasure to express my deep sense of thanks and gratitude to my guide, Ms.Khushboo Garg, for allowing me to work on this project. It was a great way to expose myself to the actual research environment.

I thank FLIPROBO for permitting me to work with them.
I take this opportunity to say heartfelt thanks to Dr. Deepika Sharma, VP-learning And development DataTraind for her overall dedication, devotion, and support towards me. I convey my sincere regards to all the DataTraind team thanks for supporting me during academic years of my post-graduation course in data science.

I express my profound sense of gratitude to my mentor Ms.Khushboo Garg, FLIPROBO for her guidance at every step of my research work.

Apart from the project, I learned a lot from her, she gave me valuable thought- "To think"; that I will benefit from, for a long time to come. I am indebted to her more than she knows.

INTRODUCTION

- ## Business Problem Framing

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

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

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

- **Conceptual Background of the Domain Problem**

With the proliferation of smart devices and mobile and social network environments, the social side effects of these technologies, including cyberbully through malicious comments and rumors, have become more serious. Malicious online comments have emerged as an unwelcome social issue worldwide.

Both cyberbully and malicious comments are increasingly viewed as a social problem due to their role in suicides and other real-world crimes. However, the online environment generally lacks a system of barriers to prevent privacy invasion, personal attacks, and cyberbully, and the barriers that do exist are weak. Social violence as an online phenomenon is increasingly pervasive, a phenomenon manifesting itself through social divisiveness.

Motivations for malicious comments identified in the study involved targeting people's mistakes. Conversely, most benevolent comments involved encouragement and compliments to help people in difficult or risky situations, showing malicious comments is a primary reason for degradation of online social networks. Moreover, the abolition of anonymity and intensification of punishment in social media can be effective in reducing malicious comments and rumors. However, potential violation of freedom of expression also risks trivializing the online social network itself. Because anonymous forms of freedom of expression have always been controversial in theoretical and normative spheres of social research. Careful consideration of any limiting of comments is necessary before a ban might be contemplated.

Requiring true identities would cause them to be more careful and responsible. Our results also suggest providers of social media services can apply text filters to their systems. Because certain texts are used repeatedly in cyberbully or malicious comments, providers should be persuaded to develop a system to detect certain texts and alert them as to when to possibly take action against the people posting them. Such a filtering function could reduce the number of all kinds of malicious comments. Conversely, social media service providers should consider posting lists that rank users most active in posting benevolent comments on their sites. Because people generally enjoy self-expression, these rankings could motivate more people to post positive comments as a way to develop a new social norm in which malicious comments are unwelcome and the people posting them are scorned.

- ### Motivation for the Problem Undertaken

Motivations for malicious comments identified in the study involved targeting people's mistakes. Conversely, most benevolent comments involved encouragement and compliments to help people in difficult or risky situations, showing malicious comments is a primary reason for degradation of online social networks. Moreover, the abolition of anonymity and intensification of punishment in social media can be effective in reducing malicious comments and rumors. However, potential violation of freedom of expression also risks trivializing the online social network itself. Because anonymous forms of freedom of expression have always been controversial in theoretical and normative spheres of social research. Careful consideration of any limiting of comments is necessary before a ban might be contemplated.

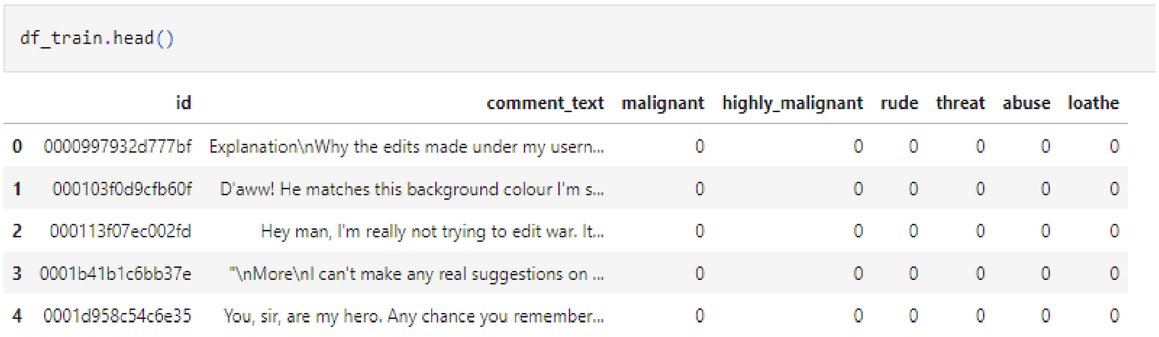
Based on data provided, a comment is assessed based on different factors. By building the model, we can assess which comments are highly likely to be hateful in varying degrees of hate thereby it will be useful for those people who are target of online hate comments by deleting those comments.

Analytical Problem Framing

- ### Data Sources and their formats

The data has been provided in different train and test datasets in a comma separated values(.csv) format.

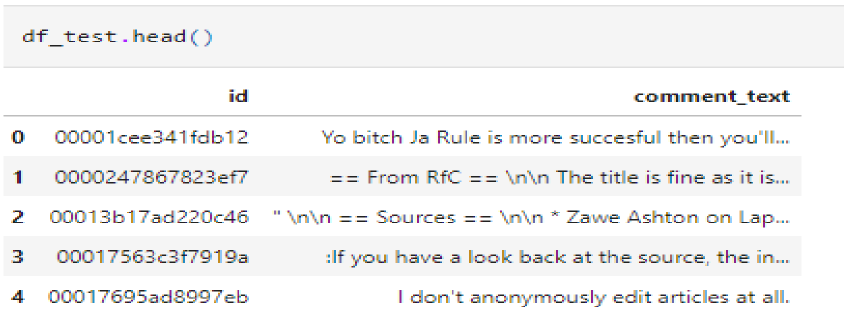
-Train data set



```
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

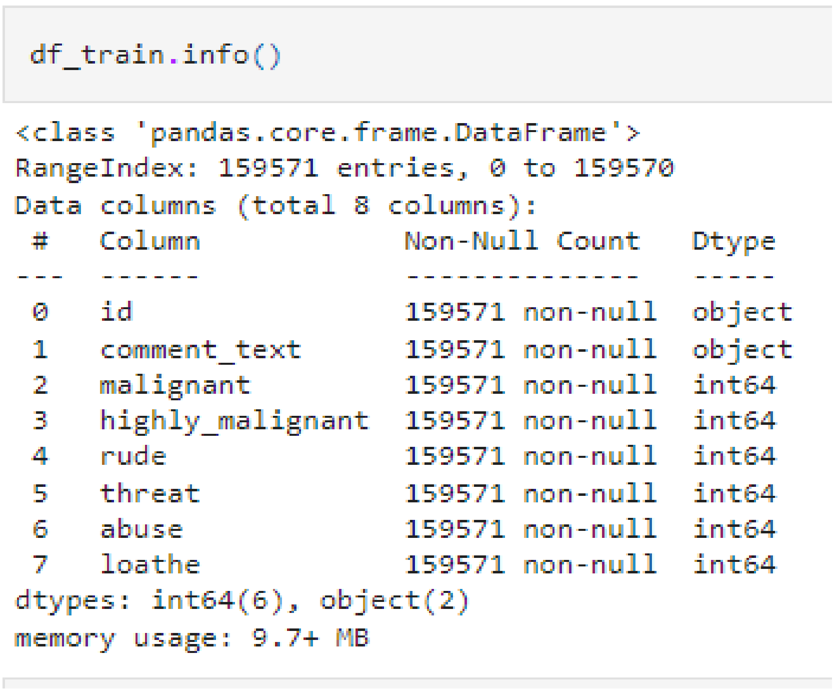
- Test data set



```
df_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.

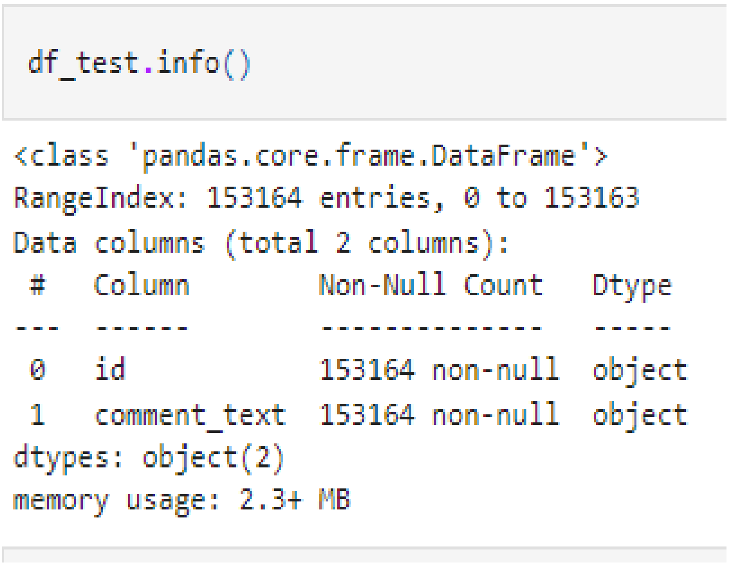
-**Train data set INFO:-** No null values and we have Object and int64 Data type.



```
df_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
```

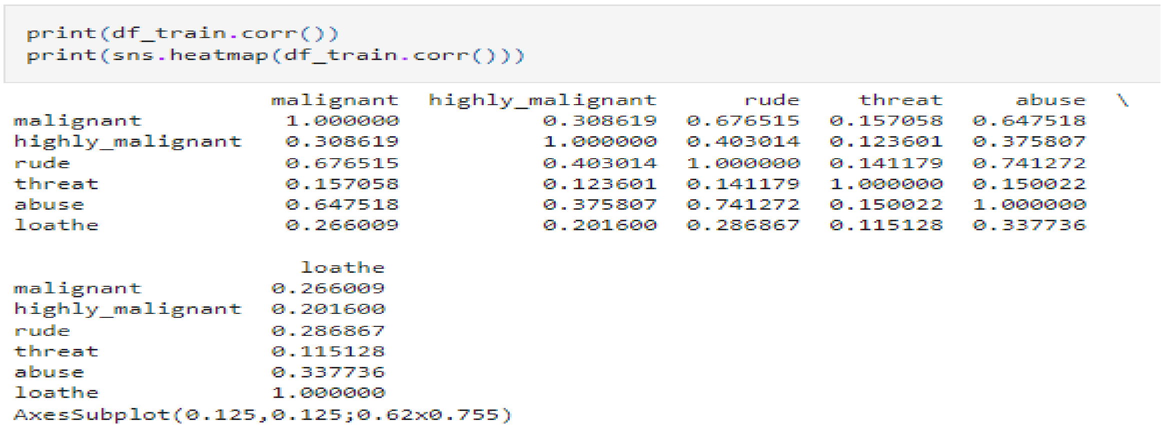
- Test data set INFO:- No null values and we have Object Data type



```
df_test.info()
```

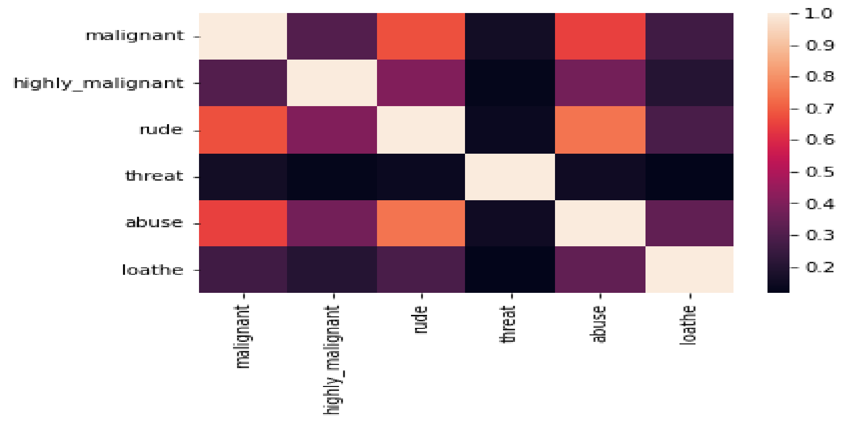
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153164 entries, 0 to 153163
Data columns (total 2 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   id            153164 non-null  object
 1   comment_text  153164 non-null  object
dtypes: object(2)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

-Correlation of train data set



```
print(df_train.corr())
print(sns.heatmap(df_train.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)
```



- ## Data Preprocessing Done

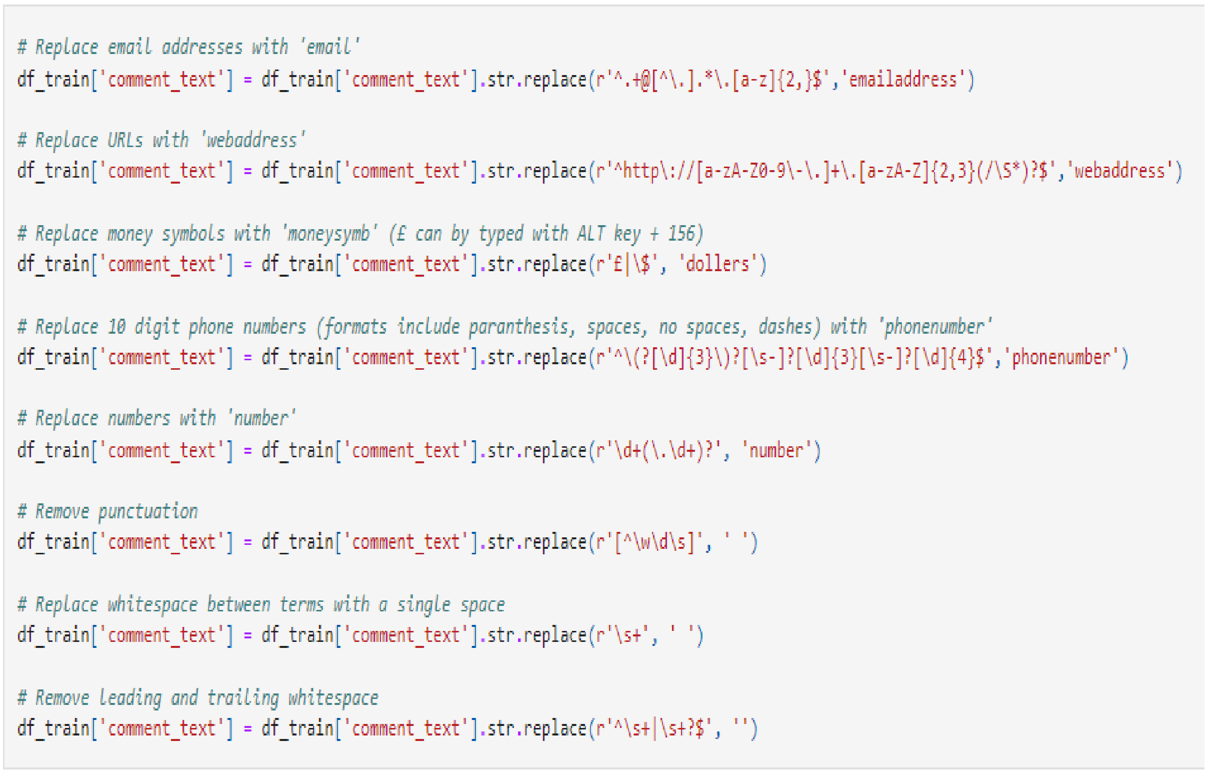
Removing major errors, duplicates, and outliers— all of which are inevitable problems when aggregating data from numerous sources.

Removing unwanted data points— extracting irrelevant observations that have no bearing on your intended analysis.

Bringing structure to your data— general 'housekeeping', i.e. fixing typos or layout issues, which will help you map and manipulate your data more easily.

Filling in major gaps— as you're tidying up, you might notice that important data are missing. Once you've identified gaps, you can go about filling them.

-CLEANING TRAIN DATA SET



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

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

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
df_train['comment_text'] = df_train['comment_text'].str.replace(r'£|\$', 'dollers')

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

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

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

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

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

-CLEANING TEST DATA SET

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

# Replace URLs with 'webaddress'
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 with 'moneysymb' (£ can by typed with ALT key + 156)
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')

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

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

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

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

-STEMMING AND LEMMATIZATION

Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling. Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma. For instance, stemming the word 'Caring' would return 'Car'

```
#create objects for stemmer and lemmatizer
lemmatiser = WordNetLemmatizer()
stemmer = PorterStemmer()
```

```
for i in range(len( df_train['comment_text'])):
    df_train['comment_text'][i]= df_train['comment_text'][i].lower()
    Y = []
    for word in df_train['comment_text'][i].split():
        Y.append((lemmatiser.lemmatize(word,pos="v")))
        df_train['comment_text'][i]= " ".join(Y)
```

```
for i in range(len(df_test['comment_text'])):
    df_test['comment_text'][i]= df_test['comment_text'][i].lower()
    Y = []
    for word in df_test['comment_text'][i].split():
        Y.append((lemmatiser.lemmatize(word,pos="v")))
        df_test['comment_text'][i]= " ".join(Y)
```

- ## Data Inputs- Logic- Output Relationships

The comments columns in train data showed which malignant comments are and which are not malignant comments. With the help of this train data we trained the model and used test data to predict the values.

- ## Hardware and Software Requirements and Tools Used

-We used this software:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
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,f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings('ignore')
```

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods)**

1. Read dataset and make it in proper format.

2. Encode labels

3. Convert all cases to lower: I will now convert the text to all lowercase. Our machine learning algorithms recognize words that start with a capital letter as different words, and we will convert them to lowercase. Thus, our machine learning algorithms will not perceive words that start with a capital letter as a different word.

4. Remove punctuations

5. Remove Stop words

6. Check stats of messages

7. Convert all texts into vectors

8. Import classifier

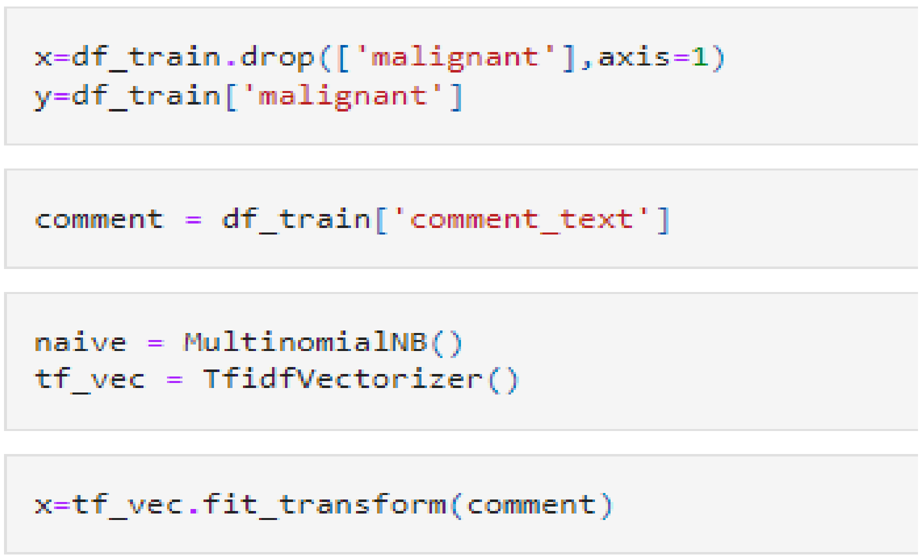
9. Train and test

10. Check the accuracy/confusion matrix

- **Testing of Identified Approaches (Algorithms)**

-Multinomial Naive Bayes

The Multinomial Naive Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a newspaper story, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the greatest chance.



```
x=df_train.drop(['malignant'],axis=1)
y=df_train['malignant']
```

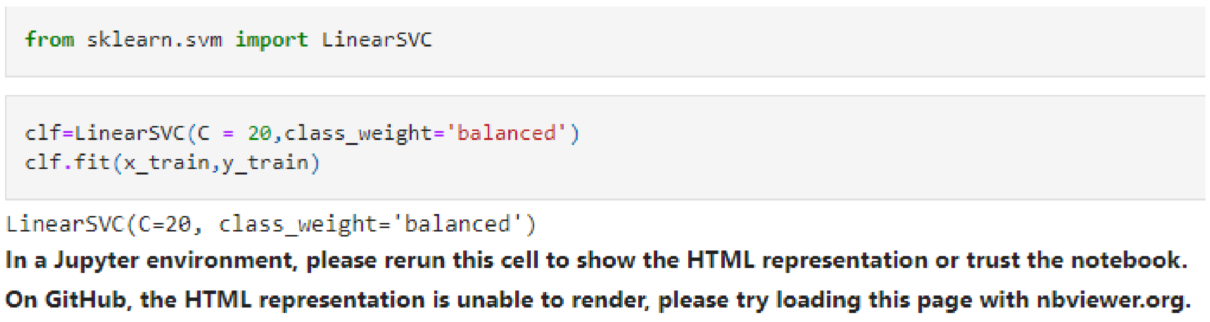
```
comment = df_train['comment_text']
```

```
naive = MultinomialNB()
tf_vec = TfidfVectorizer()
```

```
x=tf_vec.fit_transform(comment)
```

-Linear SVC

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.



```
from sklearn.svm import LinearSVC
```

```
clf=LinearSVC(C = 20,class_weight='balanced')
clf.fit(x_train,y_train)
```

```
LinearSVC(C=20, class_weight='balanced')
```

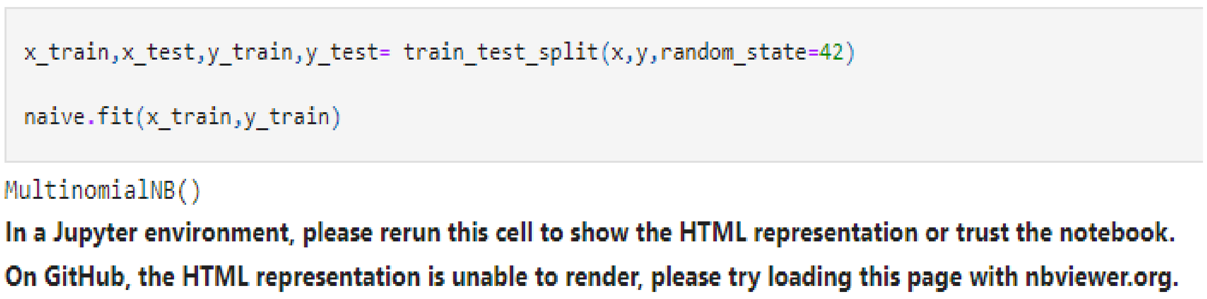
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

-from sklearn.feature_extraction.text import TfidfVectorizer Transforms text to feature vectors that can be used as input to estimator. Vocabulary Is a dictionary that converts each token (word) to feature index in the matrix, each unique token gets a feature index.

 -from sklearn.model_selection import train_test_split The train_test_split function of the sklearn. model_selection package in Python splits arrays or matrices into random subsets for train and test data, respectively.

- ## Run and Evaluate selected models

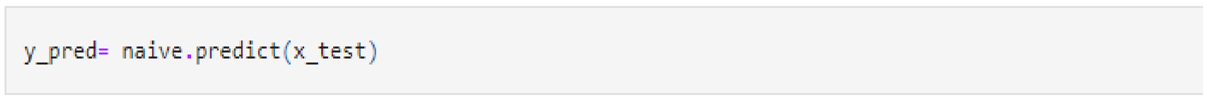
-Multinomial Naive Bayes



```
x_train,x_test,y_train,y_test= train_test_split(x,y,random_state=42)

naive.fit(x_train,y_train)
```

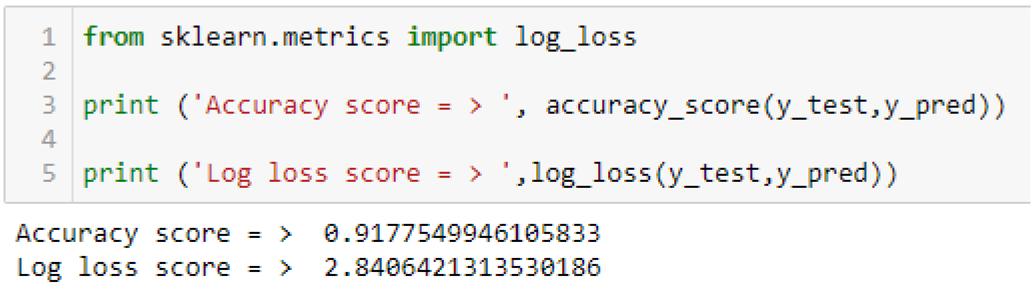
```
MultinomialNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.



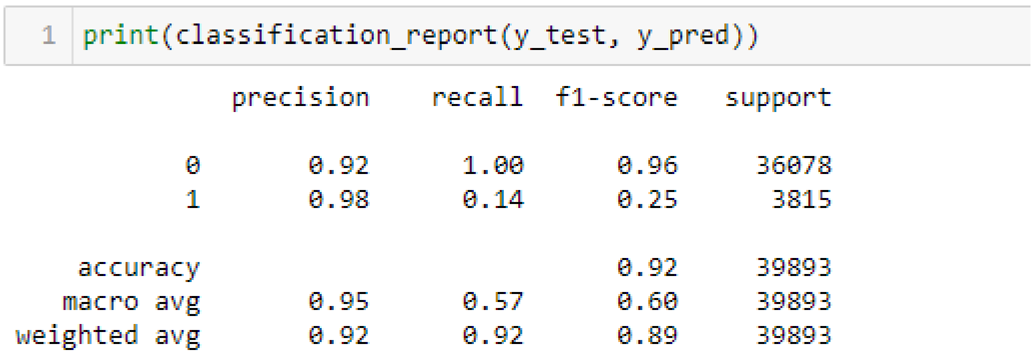
```
y_pred= naive.predict(x_test)
```

-We can observe the accuracy score



```
from sklearn.metrics import log_loss

print ('Accuracy score = > ', accuracy_score(y_test,y_pred))

print ('Log loss score = > ',log_loss(y_test,y_pred))
```

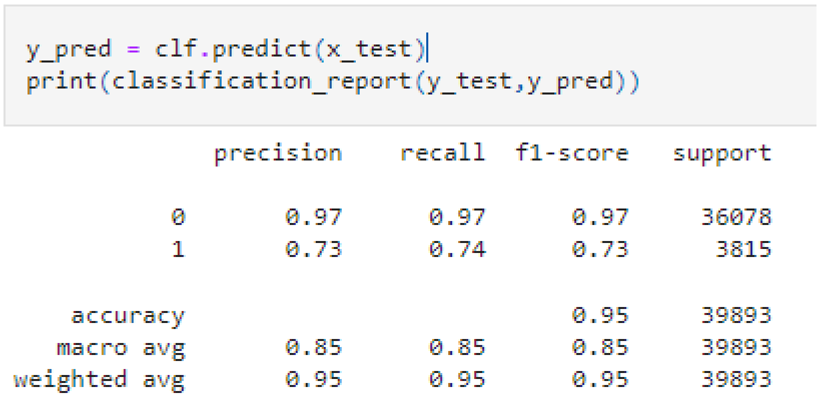
```
Accuracy score = >  0.9177549946105833
Log loss score = >  2.8406421313530186
```



```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     36078
           1       0.98      0.14      0.25      3815

    accuracy                           0.92     39893
   macro avg       0.95      0.57      0.60     39893
weighted avg       0.92      0.92      0.89     39893
```

-Linear SVC



```
y_pred = clf.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     36078
           1       0.73      0.74      0.73      3815

    accuracy                           0.95     39893
   macro avg       0.85      0.85      0.85     39893
weighted avg       0.95      0.95      0.95     39893
```

- Key Metrics for success in solving problem under consideration

1. Read dataset and make it in proper format.

2. Encode labels

3. Convert all cases to lower: I will now convert the text to all lowercase. Our machine learning algorithms recognize words that start with a capital letter as different words, and we will convert them to lowercase. Thus, our machine learning algorithms will not perceive words that start with a capital letter as a different word.

4. Remove punctuations

5. Remove Stop words

6. Check stats of messages

7. Convert all texts into vectors

8. Import classifier

9. Train and test

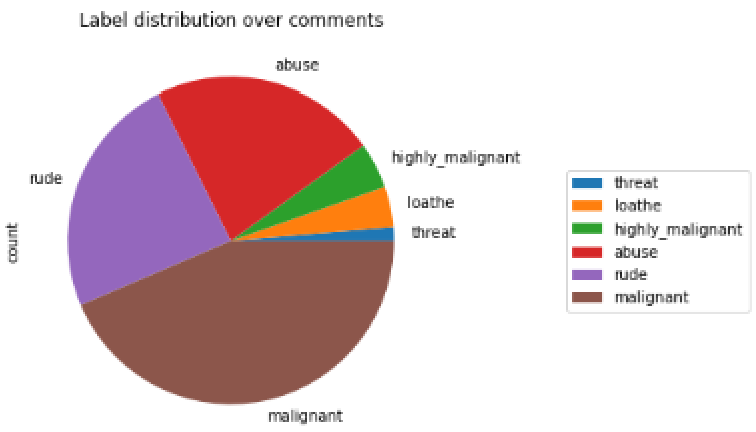
10. Check the accuracy/confusion matrix

- Visualizations

  -We can observe that malignant there are 50% of malignant comments present in our train data set:

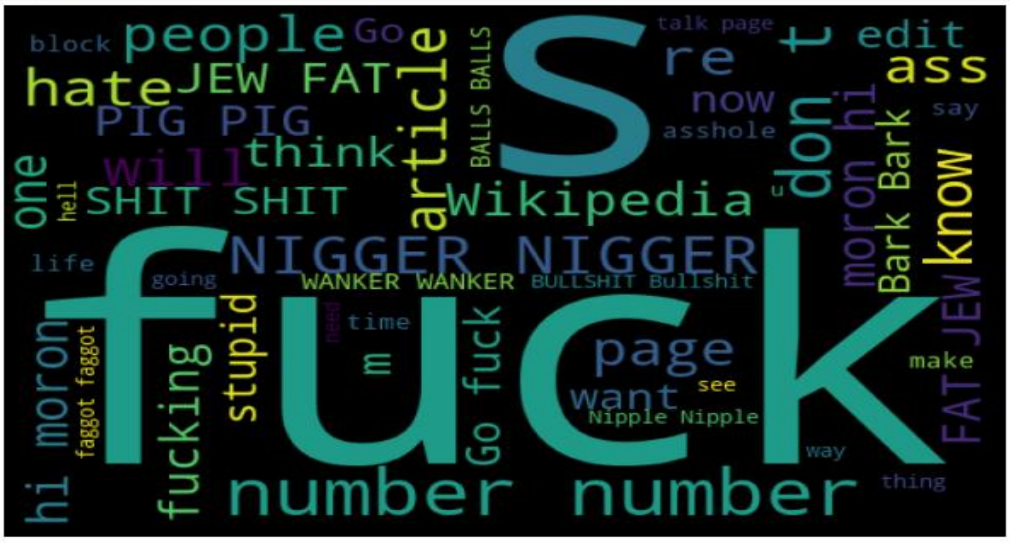
```
df_distribution = df_train[M].sum()\
                            .to_frame()\
                            .rename(columns={0:'count'})\
                            .sort_values('count')
df_distribution.plot.pie(y = 'count',
                          title = 'Label distribution over comments',
                          figsize=(5,5))\
.legend(loc = 'center left', bbox_to_anchor = (1.3, 0.5))
```

```
<matplotlib.legend.Legend at 0x12a18d44fd0>
```

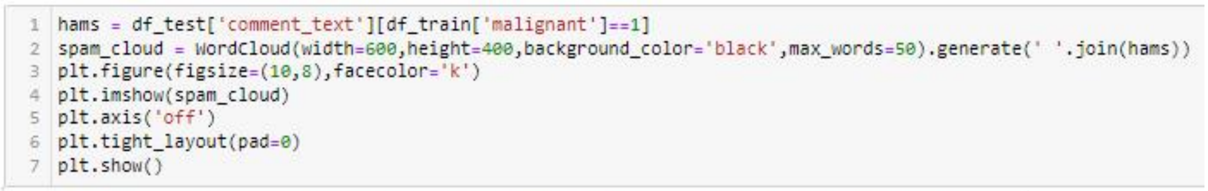


-Train Word Cloud Significant textual data points can be highlighted using a word cloud

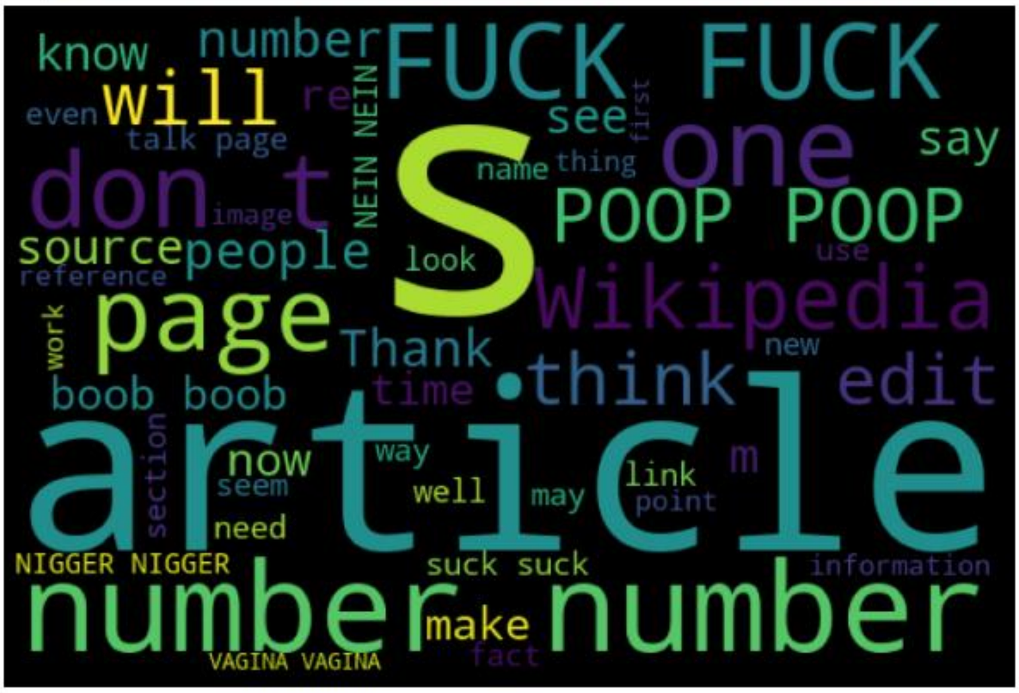
```
from wordcloud import WordCloud
hams = df_train['comment_text'][df_train['malignant']==1]
spam_cloud = WordCloud(width=600,height=400,background_color='black',max_words=50).generate(' '.join(hams))
plt.figure(figsize=(10,8),facecolor='k')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



-Test Word Cloud Significant textual data points can be highlighted using a word cloud

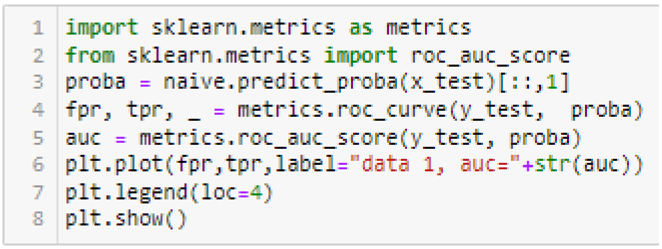


```python
hams = df_test['comment_text'][df_train['malignant']==1]
spam_cloud = WordCloud(width=600,height=400,background_color='black',max_words=50).generate(' '.join(hams))
plt.figure(figsize=(10,8),facecolor='k')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

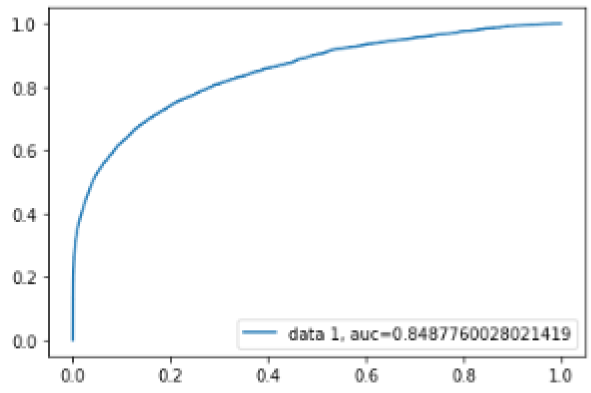


- ## Interpretation of the Results

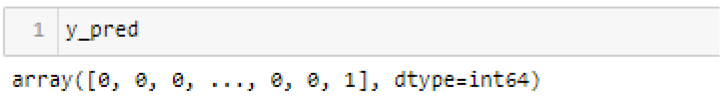
-We plotted AUC-RUC curve and got AUC= 0.84, another common metric is AUC, area under the receiver operating characteristic (ROC) curve. The Receiver operating characteristic curve plots the true positive (TP) rate versus the false positive (FP) rate at different classification thresholds.



```
import sklearn.metrics as metrics
from sklearn.metrics import roc_auc_score
proba = naive.predict_proba(x_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  proba)
auc = metrics.roc_auc_score(y_test, proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

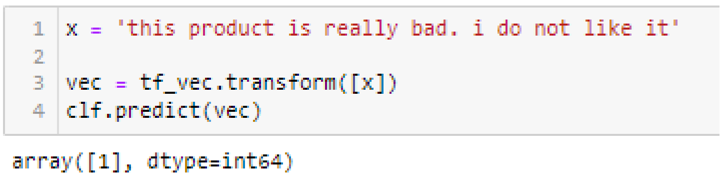


-We also test our model by some good and bad comments:



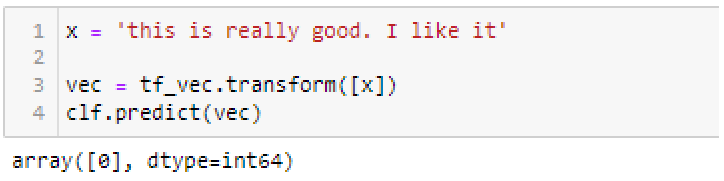
```
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```



```
x = 'this product is really bad. i do not like it'

vec = tf_vec.transform([x])
clf.predict(vec)
```

```
array([1], dtype=int64)
```



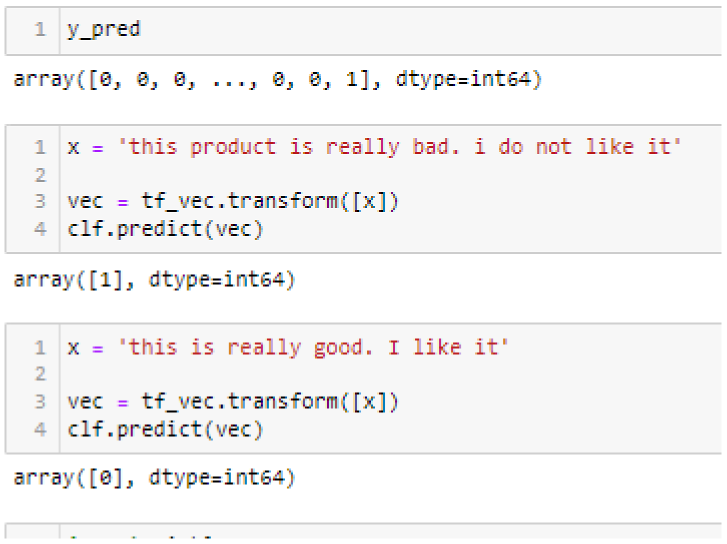
```
x = 'this is really good. I like it'

vec = tf_vec.transform([x])
clf.predict(vec)
```

```
array([0], dtype=int64)
```

CONCLUSION

- **Key Findings and Conclusions of the Study**

-We build an application that can predict malignant comments to the given reviews



```
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

```
x = 'this product is really bad. i do not like it'

vec = tf_vec.transform([x])
clf.predict(vec)
```

```
array([1], dtype=int64)
```

```
x = 'this is really good. I like it'

vec = tf_vec.transform([x])
clf.predict(vec)
```

```
array([0], dtype=int64)
```

- Learning Outcomes of the Study in respect of Data Science

-Defining the question First I define the problem statement

-Collecting the data

-Cleaning the data

-Removing major errors, duplicates

-Removing unwanted data points

-Bringing structure to your data

-Filling in major gaps

-Analyzing the data -Sharing the results

• Limitations of this work and Scope for Future Work

This paper proposed a Machine Learning Approach combined with Natural Language Processing for toxicity detection and its type identification in user comments. Finally, the Mean Validation Accuracy, so obtained, is 92% which is by far the highest ever numeric accuracy reached by any Comment Toxicity Detection Model.

The research done in this paper is intended to enhance fair online talk and views sharing in social media. A more robust model can be developed by applying Grid Search Algorithm on the same dataset over the Machine Learning Algorithms for every pipeline, being used in order to obtain better results and accurate classifications