FLIP ROBO

MALIGNANT COMMENTS CLASSIFIER PROJECT

Submitted by

Raganee Verma

(intern fliprobo **)**

ACKNOWLEDGMENT

I would like to express my special thanks to our mentor Ms. Khushboo Garg mam as well as Flip Robo Technologies who gave me the opportunity to do this project on Malignant Comments Classification, which also helped me in doing lots of knowledge and research work. wherein I came to know about so many new things, especially the Natural Language Processing and Natural Language Toolkit parts.

I am also using a few external resources that helped me to complete this project and I learn from the samples and modify things according to my project requirement. The external resources, research paper and articles that were used in creating this project are listed below:

- https://www.google.com/
- https://www.youtube.com/
- https://scikit-learn.org/stable/user_guide.html
- https://github.com/
- https://www.kaggle.com/
- https://medium.com/@dobko_m/nlp-text-data-cleaning-and-preprocessing-ea3ffe0406c1
- https://towardsdatascience.com/
- TF-IDF Vectorizerscikit-learn. Deep understanding TfidfVectorizer by… | by Mukesh Chaudhary | Medium

INTRODUCTION

- **Business Problem Framing**

The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.
Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.
There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and 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 un offensive, 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

- ## Review of Literature

The purpose of the literature review is to Identify the toxic words or toxic statements that are being used and stop the people from using these toxic languages in online public platforms.

To solve this problem, we are now building a model using our machine language technique that identifies all the toxic language and words, using which the online platforms like social media principally stops these mob using the toxic language in an online community or even block them or block them from using this language.

I have used 7different Classification algorithms and shortlisted the best on basis of the metrics of performance and I have chosen one algorithm and build a model in that algorithm.

- ## Motivation for the Problem Undertaken

One of the first lessons we learn as children is that the louder you scream and the bigger of a tantrum you throw, you more you get your way. Part of growing up and maturing into an adult and functioning member of society is learning how to use language and reasoning skills to communicate our beliefs and respectfully disagree with others, using evidence and persuasiveness to try and bring them over to our way of thinking.

Social media is reverting us back to those animalistic tantrums, schoolyard taunts and unfettered bullying that define youth, creating a dystopia where even renowned academics and dispassionate journalists transform from Dr. Jekyll into raving Mr. Hydes, raising the critical question of whether social media should simply enact a blanket ban on profanity and name calling? Actually, ban should be implemented on these profanities and taking that as a motivation I have started this project to identify the malignant comments in social media or in online public forums.

With widespread usage of online social networks and its popularity, social networking platforms have given us incalculable opportunities more than ever before, and their benefits are undeniable. Despite benefits, people may be

humiliated, insulted, bullied, and harassed by anonymous users, strangers, or peers. In this study, we have proposed a cyberbullying detection framework to generate features from online content by leveraging a pointwise mutual information technique. Based on these features, we developed a supervised machine learning solution for cyberbullying detection and multi-class categorization of its severity. Results from experiments with our proposed framework in a multi-class setting are promising both with respect to classifier accuracy and f-measure metrics. These results indicate that our proposed framework provides a feasible solution to detect cyberbullying behaviour and its severity in online social networks.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  Imported The libraries/dependencies for this project are shown below

```python
# importing all required libraries for the project
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as stats
import nltk
from nltk.corpus import wordnet
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk import pos_tag
from collections import Counter
from nltk import FreqDist
from nltk.tokenize import word_tokenize
#from nltk.stem.wordnet import WordNetLemmatizer
#from nltk.stem import wordnet
import re
import string


import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

```

In this project, we have been provided with two datasets namely train and test CSV files. I will build a machine learning model by using NLP using train dataset. And using this model we will make predictions for our test dataset.

I need to build multiple classification machine learning models. Before model building will need to perform all data pre-processing steps involving NLP. After trying different classification models with different hyper parameters then will select the best model out of it. Will need to follow the complete life cycle of data science that includes steps like -

- Data Cleaning
- Exploratory Data Analysis
- Data Pre-processing
- Model Building
- Model Evaluation
- Selecting the best model

Data Sources and their formats

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:

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 Preprocessing

The following pre-processing pipeline is required to be performed before building the classification model prediction:

- Loading the dataset
- Remove null values
- Drop column id
- Convert comment text to lower case and replace '\n' with single space.
- Keep only text data ie. a-z' and remove other data from comment text.
- Remove stop words and punctuations
- Apply Stemming using SnowballStemmer
- Convert text to vectors using TfidfVectorizer
- Load saved or serialized model
- Predict values for multi class label

Loading the dataset

Here I am loading the training dataset into the variable df and test dataset as df_test

```
df.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_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.

Identification of possible problem-solving approaches (methods)

I checked through the entire training dataset for any kind of missing values information and all these preprocessing steps were repeated on the testing dataset as well.

```
# Creating function to see information of dataset
def features_info(dataset):

    print("\nRows in dataset =",dataset.shape[0])
    print("\nColumns in dataset =",dataset.shape[1])
    print("\nFeatures names =\n",dataset.columns)
    print("\nDataset types : \n",dataset.dtypes)
    print("\nNull values in dataset :\n",dataset.isnull().sum())
    print("\nAny duplicated values :",dataset.duplicated().values.any(),"\n")
    for i in dataset.columns:
        print("Total unique values in {} = {}".format(i,dataset[i].nunique()))
```

```
# Let's call the function
features_info(df)
```

```
Rows in dataset = 159571

Columns in dataset = 8

Features names =
 Index(['id', 'comment_text', 'malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe'],
      dtype='object')


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

Null values in dataset :
 id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64

Any duplicated values : False
```

```
Total unique values in id = 159571
Total unique values in comment_text = 159571
Total unique values in malignant = 2
Total unique values in highly_malignant = 2
Total unique values in rude = 2
Total unique values in threat = 2
Total unique values in abuse = 2
Total unique values in loathe = 2
```

Observation:

1. Here we can see all the names of the columns present in our train dataset with Malignant as our target column.

2. Here we can see all the names of the columns present in our train dataset with Malignant as our target column.

3. Id and comment is object datatype

4 .and all other column is integer type.

5. There are no null values in dataset and the columns which shows the tells the type of comment are having two unique values either 0 or 1. 0 represent NO while 1 represents Yes.

6. Here we can see that there are two types of dtype present in the train dataset i.e., object and integer dtype. Here we can see that there is 1st column name id, ids are unique for all the comments dataset and it won't help in our model building, it will make the model more complex and less accurate. so, we must drop this column.

7. No any duplicate value is present.

```
# lets check the information about the train dataset
df.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
```

```
# lets check the information regarding the test dataset
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
```

checking ratio of data which contains malignant comments and normal

```python
# checking ratio of data which contains malignant comments and normal or unoffensive comments.
output_labels = df.columns[2:]

# counting non-zero rows i.e. Malignant Comments
malignant_comments = len(df[df[output_labels].any(axis=1)])

# counting rows containing zero i.e. Normal Comments
normal_comments = len(df)-malignant_comments

print(f"Total Malignant Comments: {malignant_comments} ({round(malignant_comments*100/len(df),2)}%)")
print(f"Total Normal Comments: {normal_comments} ({round(normal_comments*100/len(df),2)}%)")
```

Our data frame consists 10.17% of Malignant Comments and 89.83% of Normal Comments. Hence, it is clear that the dataset is imbalanced and needs to be treated accordingly during train test split of model training.

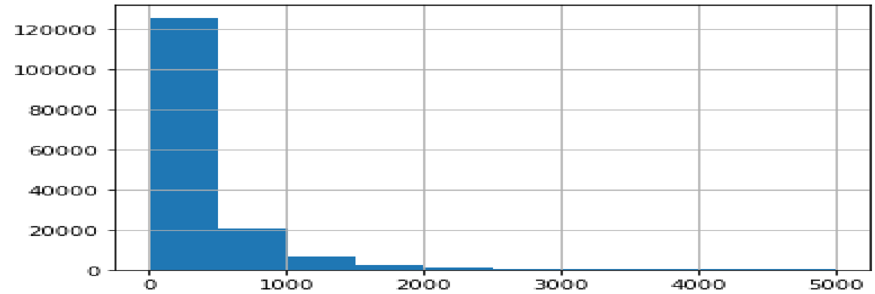
- <u>**Data Visualization**</u>

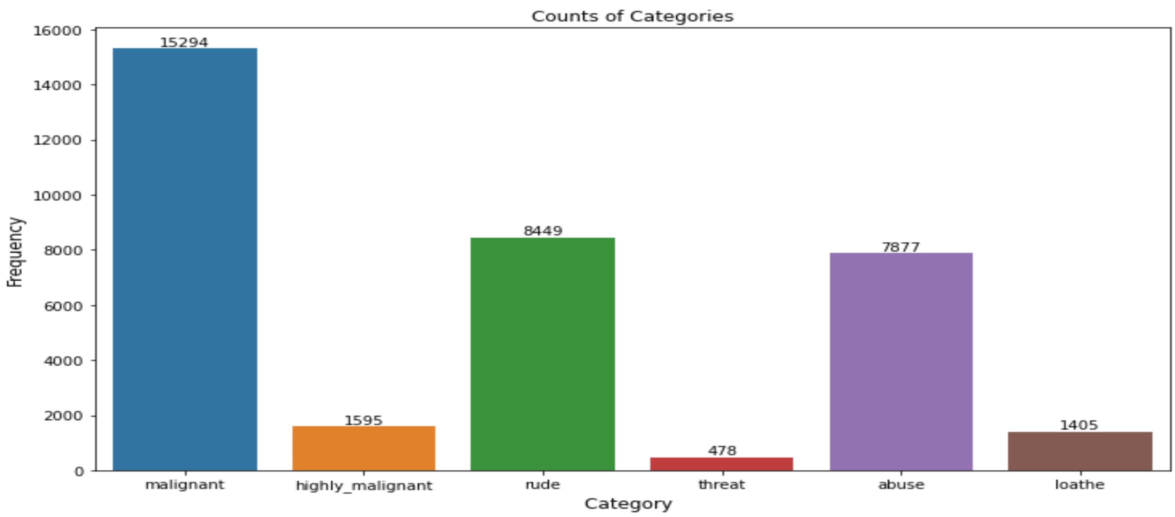
For the Visualization we have used Matplotlib and Seaborn library to plot the numerical data into graphs.

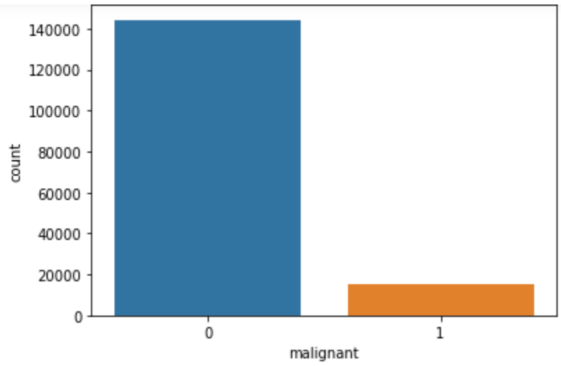
```python
# Let's check the average and maximum Length of a comment

lens = df.comment_text.str.len()
lens.mean(), lens.std(), lens.max()
```

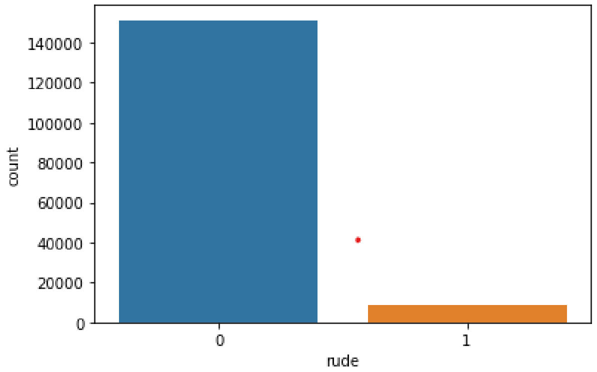
```
(394.1386655469979, 590.7254974843385, 5000)
```

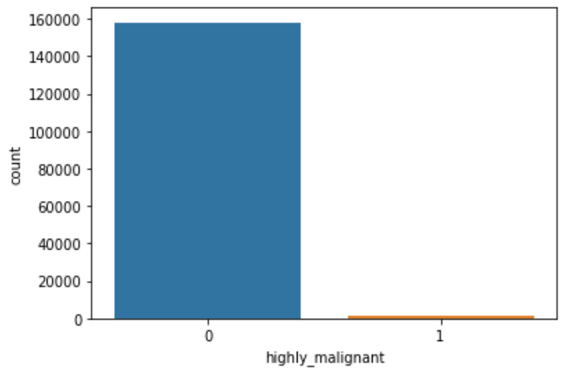
```python
# Let's Plot the Length in a histogram

lens.hist();
```

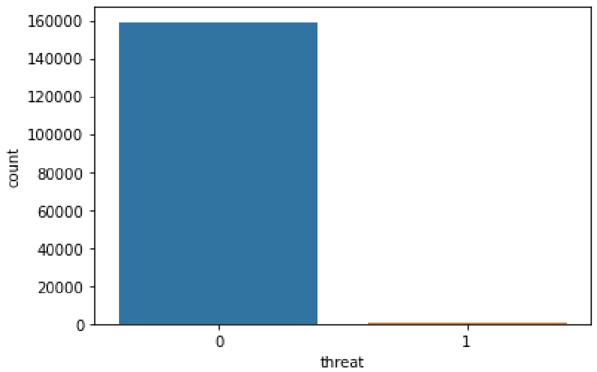


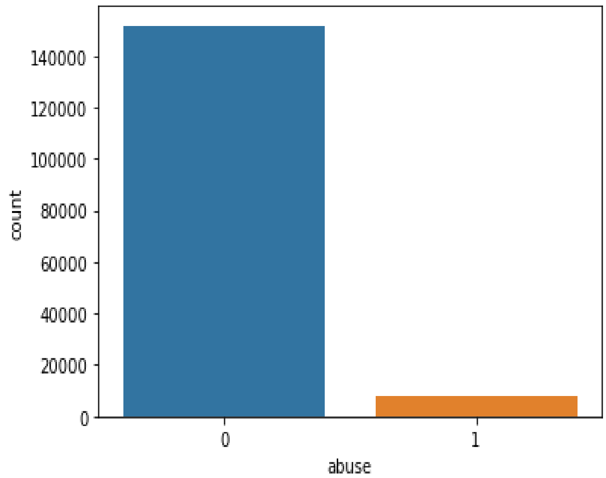


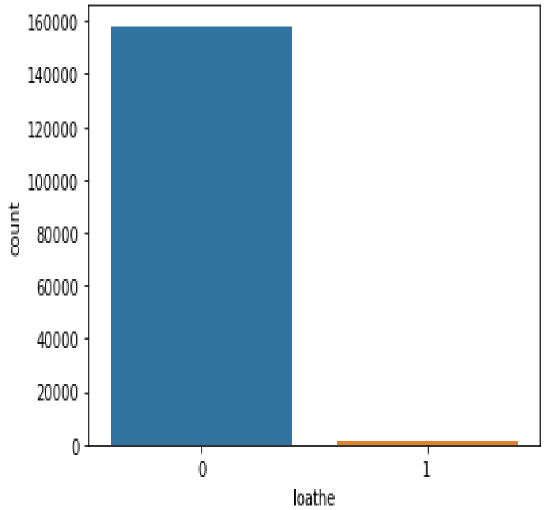












Based on the above graphs we can say that there is less percentage of negative comments which are in form of malignant, abusive, loathe, threat and highly malignant in nature

- **Data Cleaning**

Then we went ahead and performed multiple data cleaning and data transformation steps. I have added an additional column to store the original length of our comment text column.

```
# Adding new column comment_length to check length of comment_text characters

df['comment_length']=df.comment_text.str.len()
df
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_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
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67

Since there was no use of the "id" column we can drop it.

```
# Dropping column 'id' since it's of no use
df.drop(['id'],axis=1,inplace=True)
```

Creating a new feature having Negative Comments and Non-Negative Comments from all features combinly as label and that is our target column.

```
#Scaling the label column
df['label'] = df['label'] >0
df['label'] = df['label'].astype(int)
df.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	label
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264	0
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112	0
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233	0
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622	0
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67	0

```
print(df['label'].value_counts())
sns.countplot(df['label'], palette='rainbow')
plt.title('Counting of the labels after scaling',fontsize=25)
plt.show()
```

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

Text Preprocessing using NLP:

In natural language processing, text preprocessing is **the practice of cleaning and preparing text data**. NLTK and re are common Python libraries used to handle many texts preprocessing tasks.

Removing and replacing unwanted characters in the comment text column

```
# function to filter using POS tagging. This will be called inside the below function
def get_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN


# Function for data cleaning...
def Processed_data(comments):
    # Replace email addresses with 'email'
    comments=re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', comments)

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

    # getting only words(i.e removing all the special characters)
    comments = re.sub(r'[^\w]', ' ', comments)

    # getting only words(i.e removing all the" _ ")
    comments = re.sub(r'[\_]', ' ', comments)

    # getting rid of unwanted characters(i.e remove all the single characters left)
    comments=re.sub(r'\s+[a-zA-Z]\s+', ' ', comments)
```

```
    # Removing extra whitespaces
    comments=re.sub(r'\s+', ' ', comments, flags=re.I)

    #converting all the letters of the review into lowercase
    comments = comments.lower()

    # splitting every words from the sentences
    comments = comments.split()

    # iterating through each words and checking if they are stopwords or not,
    comments=[word for word in comments if not word in stopwords.words('english')]

    # remove empty tokens
    comments = [text for text in comments if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(comments)


    # considering words having length more than 3only
    comments = [text for text in comments if len(text) > 3]

    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS ..
    comments = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]

   # considering words having length more than 3 only
    comments = [text for text in comments if len(text) > 3]
    comments = ' '.join(comments)
    return comments
```

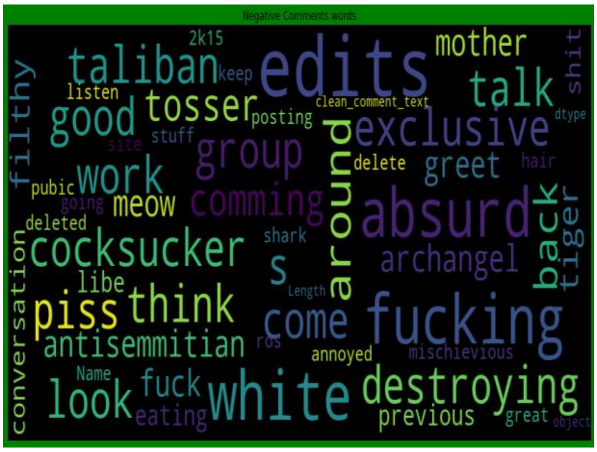
```
# Cleaning  and storing the comments in a separate feature.
df["clean_comment_text"] = df["comment_text"].apply(lambda x: Processed_data(x))
```

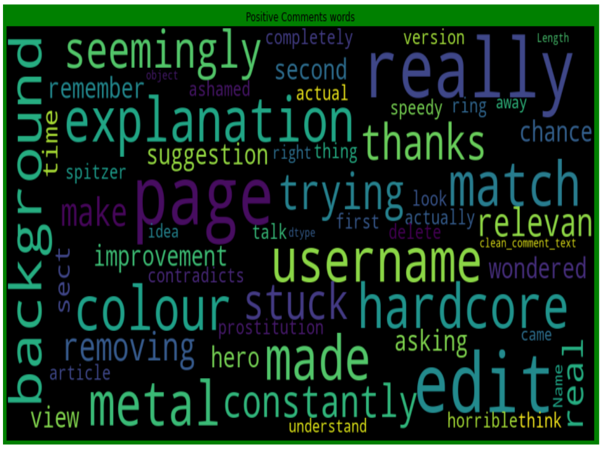
```
# Adding new feature clean_comment_length to store length of cleaned comments in clean_comment_text characters
df['clean_comment_length'] = df['clean_comment_text'].apply(lambda x: len(str(x)))
df.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	label	clean_comment_text	clean_comment_length
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264	0	explanation edits made username hardcore metal...	141
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112	0	match background colour seemingly stuck thanks...	64
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233	0	really trying edit constantly removing relevan...	125

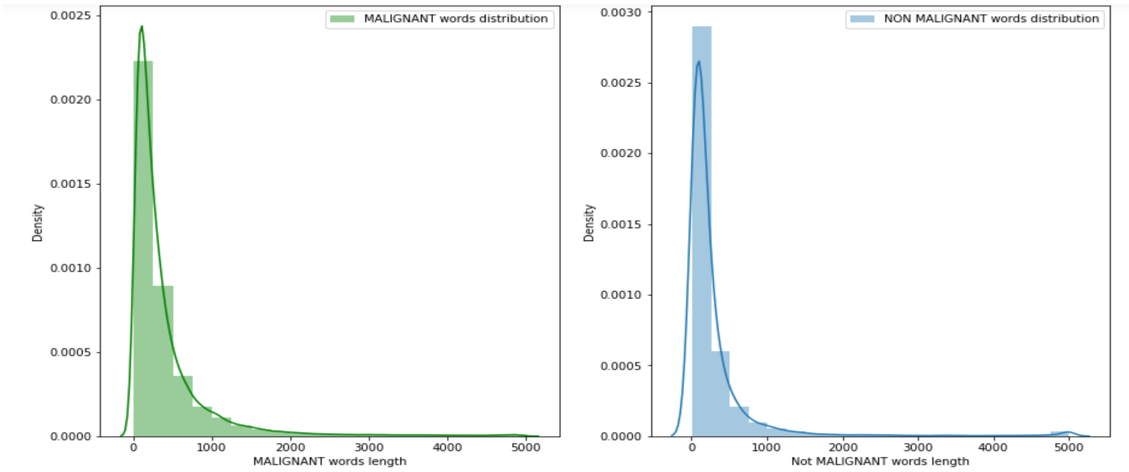
Word Cloud

Getting sense of loud words in each of the output labels .I have analysed the input output logic with word cloud and I have word clouded the sentenced that as classified as foul language in every category. A tag/word cloud is a novelty visual representation of text data, typically used to depict keyword metadata on websites or to visualize free-form text. It’s an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance





Comments length distribution BEFORE cleaning



Training and Testing Model on our train dataset

The complete list of all the algorithms used for the training and testing

1. LogisticRegression
2. MultinomialNB
3. DecisionTreeClassifier
4. KNeighborsClassifier
5. RandomForestClassifier
6. GradientBoostingClassifier
7. Support Vector Classifier

Run and evaluate selected models

I created a classification function that included the evaluation metrics details for the generation of our Classification Machine Learning models

```
# Importing libraries for model training

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier


from sklearn.model_selection import cross_val_score, cross_val_predict, train_test_split
from sklearn.model_selection import GridSearchCV


# Importing evaluation metrics for model performance....
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import log_loss
```

```
#splitting the data into training and testing
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=0.30,stratify=y)
```

```python
# Creating instances for different Classifiers

LR=LogisticRegression()
MNB=MultinomialNB()
DT=DecisionTreeClassifier()
KNN=KNeighborsClassifier()
RFC=RandomForestClassifier()
GBC=GradientBoostingClassifier()
SV=SVC()
```

```python
# Creating a list model where all the models will be appended for further evaluation in loop.
models=[]
models.append(('LogisticRegression',LR))
models.append(('MultinomialNB',MNB))
models.append(('DecisionTreeClassifier',DT))
models.append(('KNeighborsClassifier',KNN))
models.append(('RandomForestClassifier',RFC))
models.append(('GradientBoostingClassifier',GBC))
models.append(('SVC',SV))
```

```python
# Lists to store model name, Learning score, Accuracy score, cross_val_score, Auc Roc score.

Model=[]
Score=[]
Acc_score=[]
cvs=[]
rocscore=[]
lg_loss=[]
```

```python
# For Loop to Calculate Accuracy Score, Cross Val Score, Classification Report, Confusion Matrix

for name,model in models:
    print(name)
    Model.append(name)
    print(model)

    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42,stratify=y)
    model.fit(x_train,y_train)

# Learning Score
    score=model.score(x_train,y_train)
    print('Learning Score : ',score)
    Score.append(score*100)
    y_pred=model.predict(x_test)
    acc_score=accuracy_score(y_test,y_pred)
    print('Accuracy Score : ',acc_score)
    Acc_score.append(acc_score*100)


# Cross_val_score
    cv_score=cross_val_score(model,x,y,cv=5,scoring='roc_auc').mean()
    print('Cross Val Score : ', cv_score)
    cvs.append(cv_score*100)

# Roc auc score
    false_positive_rate,true_positive_rate, thresholds=roc_curve(y_test,y_pred)
    roc_auc=auc(false_positive_rate, true_positive_rate)
    print('roc auc score : ', roc_auc)
    rocscore.append(roc_auc*100)

# Log Loss
    loss = log_loss(y_test,y_pred)
    print('Log loss : ', loss)
    lg_loss.append(loss)

# Classification Report
    print('Classification Report:\n',classification_report(y_test,y_pred))
    print('\n')

    print('Confusion Matrix:\n',confusion_matrix(y_test,y_pred))
    print('\n')

    [==-=]\
    =\]\
plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    plt.plot(false_positive_rate,true_positive_rate,label='AUC = %0.2f'% roc_auc)
    plt.plot([0,1],[0,1],'r--')
    plt.legend(loc='lower right')
    plt.ylabel('True_positive_rate')
    plt.xlabel('False_positive_rate')
```

```
LogisticRegression
LogisticRegression()
Learning Score :  0.9578331050412269
Accuracy Score :  0.9534592245989305
Cross Val Score :  0.9645083965798673
roc auc score :  0.7930189291978497
Log loss :  1.6074654435002567
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.92      0.59      0.72      4868

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


Confusion Matrix:
 [[42764   240]
 [ 1988  2880]]
```

```
MultinomialNB
MultinomialNB()
Learning Score :  0.9386476154665664
Accuracy Score :  0.9344293114973262
Cross Val Score :  0.925433564221135
roc auc score :  0.6835088413934397
Log loss :  2.2647324339375343
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96     43004
           1       0.97      0.37      0.53      4868

    accuracy                           0.93     47872
   macro avg       0.95      0.68      0.75     47872
weighted avg       0.94      0.93      0.92     47872


Confusion Matrix:
 [[42939    65]
 [ 3074  1794]]
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
Learning Score :  0.9985944368347076
Accuracy Score :  0.9392755681818182
Cross Val Score :  0.8332057025091579
roc auc score :  0.8260213721878081
Log loss :  2.097370421734832
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     43004
           1       0.71      0.68      0.70      4868

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.83     47872
weighted avg       0.94      0.94      0.94     47872


Confusion Matrix:
 [[41636  1368]
 [ 1539  3329]]
```

```
KNeighborsClassifier
KNeighborsClassifier()
Learning Score :  0.9184325732549082
Accuracy Score :  0.8829378342245989
Cross Val Score :  0.6803766000263364
roc auc score :  0.6183237278112976
Log loss :  4.04321952832571
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.95      0.94     43004
           1       0.40      0.29      0.33      4868

    accuracy                           0.88     47872
   macro avg       0.66      0.62      0.63     47872
weighted avg       0.87      0.88      0.87     47872


Confusion Matrix:
 [[40875  2129]
 [ 3475  1393]]
```

```
RandomForestClassifier
RandomForestClassifier()
Learning Score :  0.9985675789398294
Accuracy Score :  0.9536054478609626
Cross Val Score :  0.9551636689479004
roc auc score :  0.8040304882303662
Log loss :  1.6024169583679295
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     43004
           1       0.89      0.62      0.73      4868

    accuracy                           0.95     47872
   macro avg       0.93      0.80      0.85     47872
weighted avg       0.95      0.95      0.95     47872


Confusion Matrix:
 [[42651   353]
 [ 1868  3000]]
```

```
GradientBoostingClassifier
GradientBoostingClassifier()
Learning Score :  0.9428911628573219
Accuracy Score :  0.9399857954545454
Cross Val Score :  0.8949387391092702
roc auc score :  0.7190277516750081
Log loss :  2.0728197802510886
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     43004
           1       0.93      0.44      0.60      4868

    accuracy                           0.94     47872
   macro avg       0.94      0.72      0.78     47872
weighted avg       0.94      0.94      0.93     47872


Confusion Matrix:
 [[42849   155]
 [ 2718  2150]]
```

Compare performance of all models

```
# Displaying scores :
results=pd.DataFrame({'Model': Model,'Learning Score': Score,'Accuracy Score': Acc_score,'Cross Val Score':cvs,
                     'Auc_score':rocscore,'Log_Loss':lg_loss})
results
```

	Model	Learning Score	Accuracy Score	Cross Val Score	Auc_score	Log_Loss
0	LogisticRegression	95.783311	95.345922	96.450840	79.301893	1.607465
1	MultinomialNB	93.864762	93.442931	92.543356	68.350884	2.264732
2	DecisionTreeClassifier	99.859444	93.927557	83.320570	82.602137	2.097370
3	KNeighborsClassifier	91.843257	88.293783	68.037660	61.832373	4.043220
4	RandomForestClassifier	99.856758	95.360545	95.516367	80.403049	1.602417
5	GradientBoostingClassifier	94.289116	93.998580	89.493874	71.902775	2.072820
6	SVC	98.161129	95.437834	96.385547	80.008861	1.575721

Hyperparameter Tuning

Looking at all the Scores, I have selected Random Forest as a best model and use RandomizedSearchCv for hyperparameter tunning.

```
from sklearn.model_selection import RandomizedSearchCV
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=.30,stratify=y)
parameters={'bootstrap': [True, False],
 'max_depth': [10, 50, 100, None],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [100, 300, 500, 800, 1200]}


RFC=RandomForestClassifier()

# Applying Randomized Search CV for hyperparameter tuning with scoring= "accuracy"
rand = RandomizedSearchCV(estimator = RFC, param_distributions = parameters,
                          n_iter = 10, cv = 3, verbose=2, random_state=42, n_jobs = -1,scoring='accuracy')
rand.fit(x_train,y_train)
rand.best_params_
```

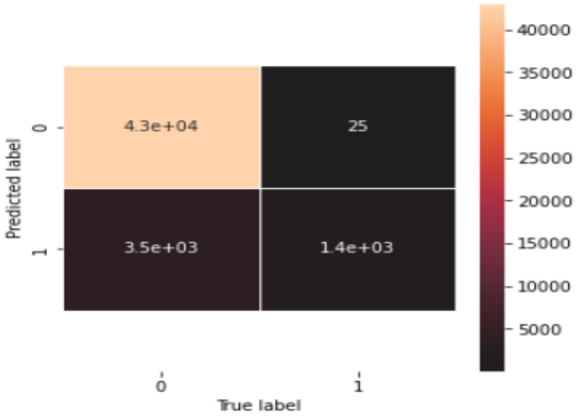
```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
{'n_estimators': 500,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_depth': 100,
 'bootstrap': False}
```

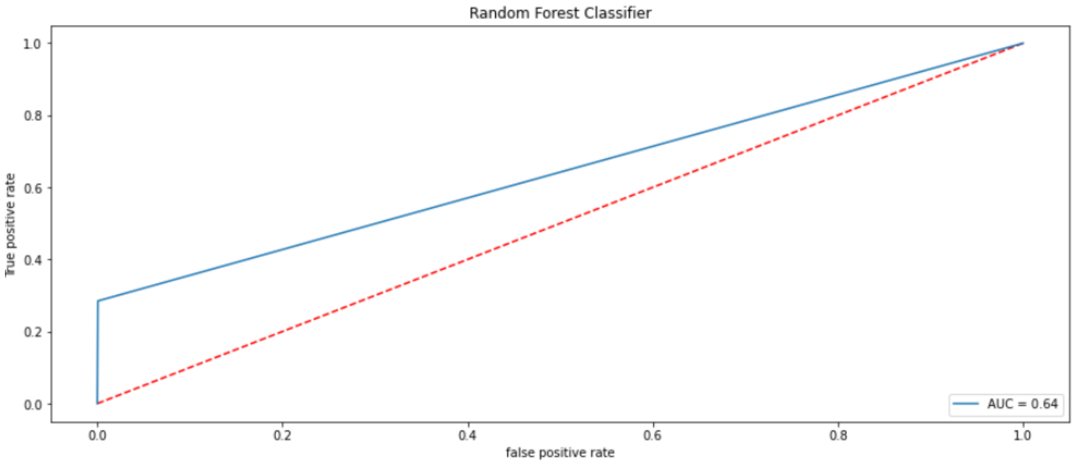
```
RFC=RandomForestClassifier(n_estimators= 500,
                           min_samples_split= 2,
                           min_samples_leaf=1,
                           max_depth= 100,
                           bootstrap= False)
RFC.fit(x_train,y_train)
RFC.score(x_train,y_train)
pred=RFC.predict(x_test)
print('Accuracy Score:',accuracy_score(y_test,pred))
print('Log loss : ', log_loss(y_test,pred))
print('Confusion Matrix:',confusion_matrix(y_test,pred))
print('Classification Report:','\n',classification_report(y_test,pred))
```

```
Accuracy Score: 0.9267421457219251
Log loss :  2.5302370656518955
Confusion Matrix: [[42979    25]
 [ 3482  1386]]
Classification Report:
               precision    recall  f1-score   support

           0       0.93      1.00      0.96     43004
           1       0.98      0.28      0.44      4868

    accuracy                           0.93     47872
   macro avg       0.95      0.64      0.70     47872
weighted avg       0.93      0.93      0.91     47872
```

```
# Confusion matrix Visualization
fig, ax =plt.subplots(figsize=(5,5))
sns.heatmap(confusion_matrix(y_test, pred),annot=True,linewidths=1,center=0)
plt.xlabel("True label")
plt.ylabel("Predicted label")
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

```
(2.5, -0.5)
```





Prediction using test data

```
x_testing_data=Tf_idf_test(df_test['clean_comment_text'])
```

```
x_testing_data.shape
```

```
(153164, 43320)
```

```
Prediction=RFC.predict(x_testing_data)
df_test['Predicted values']=Prediction
df_test
```

	id	comment_text	comment_length	clean_comment_text	clean_comment_length	Predicted values
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367	bitch rule succesful ever whats hating mofucka...	194	0
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50	title fine	10	0
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	54	source zawe ashton lapland	26	0
3	00017563c3f7919a	:If you have a look back at the source, the in...	205	look back source information updated correct f...	109	0
4	00017695ad8997eb	I don't anonymously edit articles at all.	41	anonymously edit article	24	0

```
df_test['Predicted values'].value_counts()
```

```
0    153154
1        10
Name: Predicted values, dtype: int64
```

```
df_test[df_test['Predicted values']==1].head(20)
```

	id	comment_text	comment_length	clean_comment_text	clean_comment_length	Predicted values
14628	189161f153135211	== White Argentine might be doomed == \n\n Hol...	1165	white argentine might doomed hola ianvs estos ...	843	1
25129	29e086bfcfb5d262	: But pro gov source said and show that jan + ...	292	source said show rebel still present petolucem...	135	1
38132	3f5131744222236d	" \n\n == A penny for your thoughts == \n\n I ...	324	penny thought noticed edit thought might inter...	183	1
43022	474ec43de24c8127	" \n\n == Split or Rewrite? == \n\n PAR lamps ...	610	split rewrite lamp used many application theat...	325	1
63906	6a7a8e09f23c40e8	== Kimi == \n\n Can you look over Raikkonen's ...	301	kimi look raikkonen 2007 section seems unneces...	196	1

Store data and save file

```
df_test.to_csv('Malignant_Predict.csv')
```

```
# Pickle file.
import joblib
joblib.dump(RFC,'Malignant_Predict.pkl')
```

```
['Malignant_Predict.pkl']
```

CONCLUSION

Our research has shown that harmful or toxic comments in the social media space have many negative impacts to society. The ability to readily and accurately identify comments as toxic could provide many benefits while mitigating the harm. Also, our project has shown the capability of readily available algorithms to be employed in such a way to address this challenge. In our specific study, Random Forest Algorithm provides Maximum Accuracy and cross validation score with 95% so we decide this is our best model and save it.

The finding of the study is that only few users over online use unparliamentary language. And most of these sentences have more stop words and are being quite long. As discussed before few motivated disrespectful crowds use these foul languages in the online forum to bully the people around and to stop them from doing these things that they are not supposed to do. Our study helps the online forums and social media to induce a ban to profanity or usage of profanity over these forums.

Problems faced while working in this project:

More computational power was required as it took more than 2 hours Imbalanced dataset and bad comment texts

Good parameters could not be obtained using hyperparameter tuning as time was consumed more