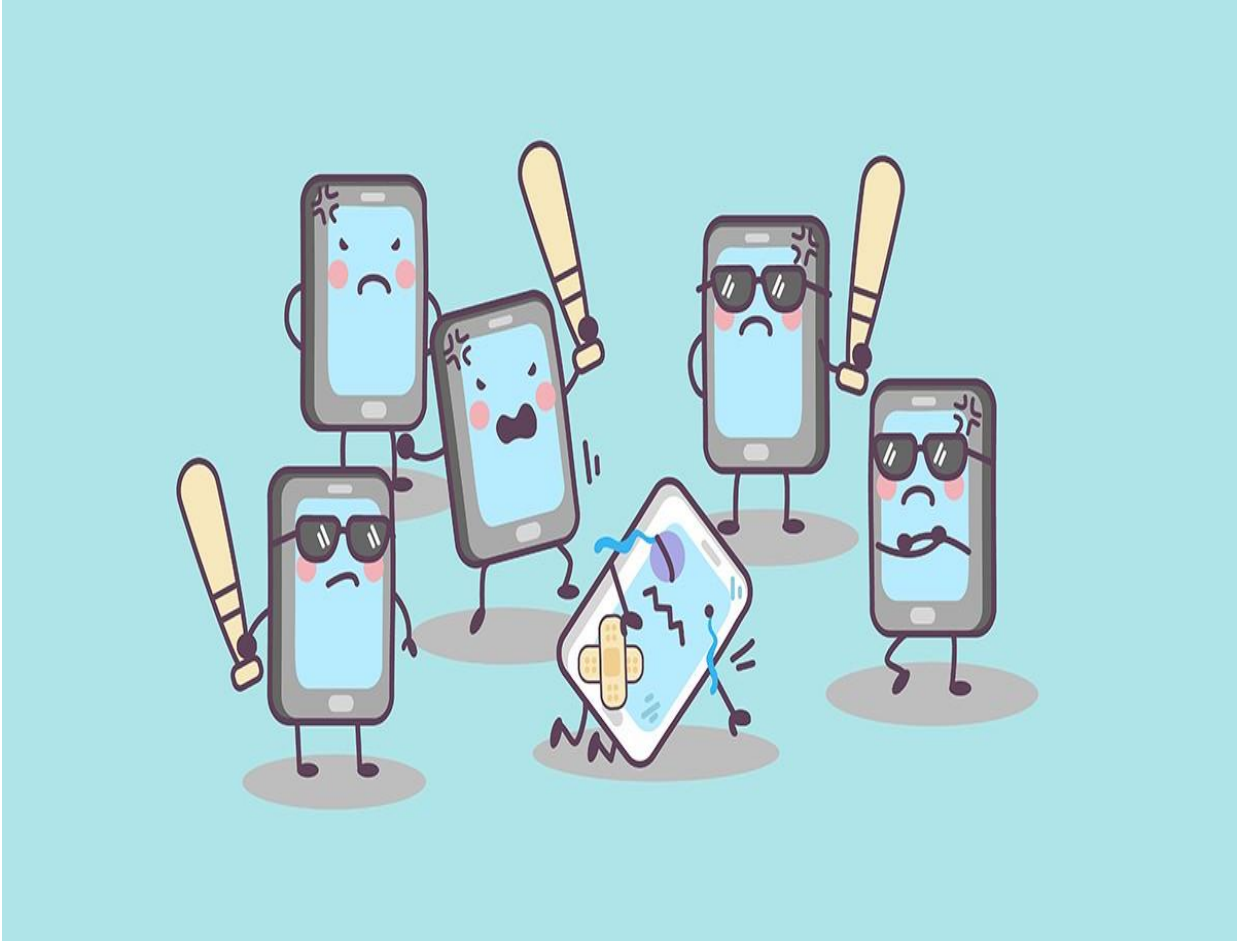
FLIP ROBO

Malignant Comment Classification



Submitted by:

Rakesh Chaudhary

ACKNOWLEDGMENT

This project would not have seen the light of the day without the following people and their priceless support and cooperation. Hence I extend my gratitude to all of them. As a part of Data Trained Education, I would first of all like to express my gratitude to FlipRobo Team and seniors for granting me permission to undertake the project report in their esteemed organization. I would also like to express my sincere thanks to Miss Khushboo Mam for supporting me and being always there for me whenever I needed. During the actual research work with FlipRobo team and other IOT that set the ball rolling for my project.

They had been a source of inspiration through their constant guidance; personal interest; encouragement and help. I convey my sincere thanks to them. In spite of their busy schedule they always found time to guide me throughout the project. I am also grateful to them for reposing confidence in my abilities and giving me the freedom to work on my project. Without their invaluable help I would not have been able to do justice to the project.

This paper presents a novel application of Natural Language Processing techniques to classify unstructured text into toxic and nontoxic categories. In the current century, social media has created many job opportunities and, at the same time, it has become a unique place for people to freely express their opinions. Meanwhile, among these users, there are some groups that are taking advantage of this framework and misuse this freedom to implement their toxic mind-set (i.e. insulting, verbal sexual harassment, threads, Obscene, etc.)

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 inoffensive, 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.

Conceptual Background of the Domain Problem

The advances in IT technologies and generalizing virtualization all over the world has led to an unprecedented participation in social mediaand there is no doubt that social media is one of the biggest hallmarks of the 21st century.According to de Bruijn, Muhonensocial media has been growing exponentially since 2004. Meanwhile, social media is a place to express individual opinions and share thoughts in line with a constructive contribution to develop a safe place for everybody practicing their rights accordingly. Based on the report by Birkland'Twitter users generate 500 million tweets per day, and in 2019 they had a 14% year-over-year growth of daily usage'. However, behind the shield of computers as virtual walls, some individuals also think they can abuse and harass other people's opinions and characters. Accordingly, a jargon word has been coined recently to address such behaviours as "cyberbullying".Based on U.S. Department of Health and Human Services, cyberbullying could be introduced as mistreatments that occurs all over digital instruments and devices such as computers, tablets, and mobile phones.

Cyberbullying may take place via short massage system (SMS), general apps with the possibility of communication between users, online social media forums, or even online gaming where individuals can virtually participate in.

Cyberbullying includes posting, sending, or sharing harmful, negative, mean, or false content about someone else either directly sent to the person or post as a general comment where other could observe it. It also includes sharing private or personal info about someone else in order to humiliation or embarrassment. Such online harassment suppresses so many of our fellow citizens from expressing their opinions. According to the US Government, two separate recent

studies have estimated that over 15% of teenagers have been victim to cyberbullying in the last 12 months.

Review of Literature

The purpose of the literature review is to:

1. Identify the foul words or foul statements that are being used.
2. Stop the people from using these foul languages in online public forum.

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

I have used 9 different 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. 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. 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.

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 than ever before, and its 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 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 Modeling of the Problem

Firstly, we do some EDAs to gain a general understanding of our data, and detecting some important metrics and trends that may come helpful for our further analysis and model building.

This dataset contains 159,571 comments. The data consists of one input feature, the string data for the comments, and six labels for different categories of malignant comments: Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.

We will now understand about the features in our dataset.

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.

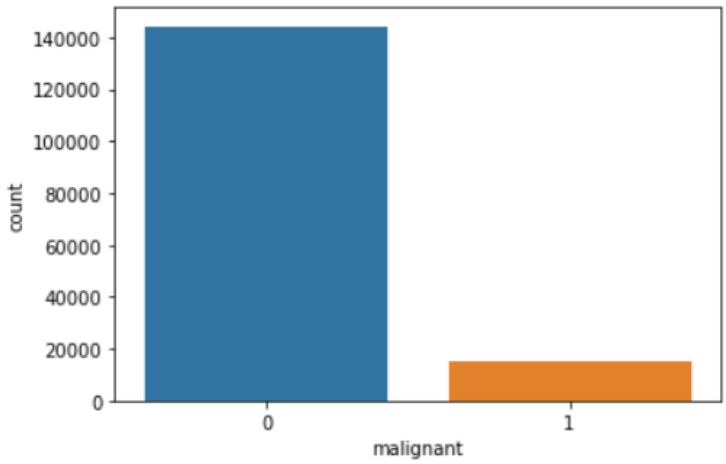
Now, we will go to graphical visualisation.

```
1 train['malignant'].value_counts()
```

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

```
1 sns.countplot(train['malignant'])
```

```
<AxesSubplot:xlabel='malignant', ylabel='count'>
```



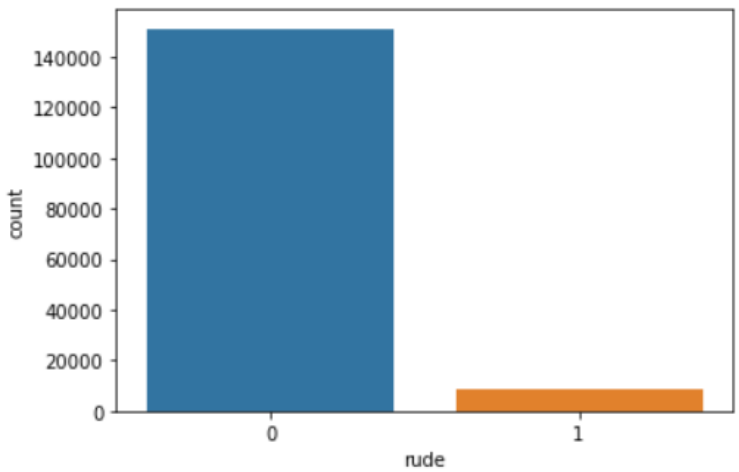
From above images we can observe that there are around 15000 comments that contains inappropriate data.

```
1 train['rude'].value_counts()
```

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

```
1 sns.countplot(train['rude'])
```

```
<AxesSubplot:xlabel='rude', ylabel='count'>
```



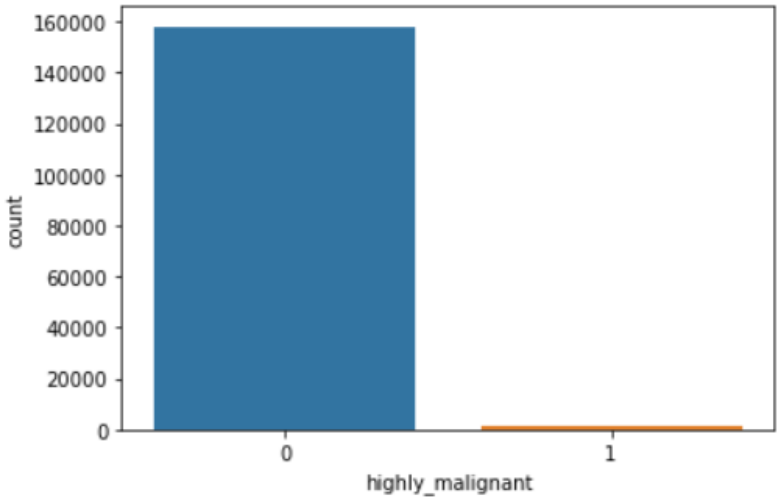
From the above image, we can observe that there are 8449 comments that contains inappropriate data and which fall under rude comments.

```
1 train['highly_malignant'].value_counts()
```

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

```
1 sns.countplot(train['highly_malignant'])
```

```
<AxesSubplot:xlabel='highly_malignant', ylabel='count'>
```



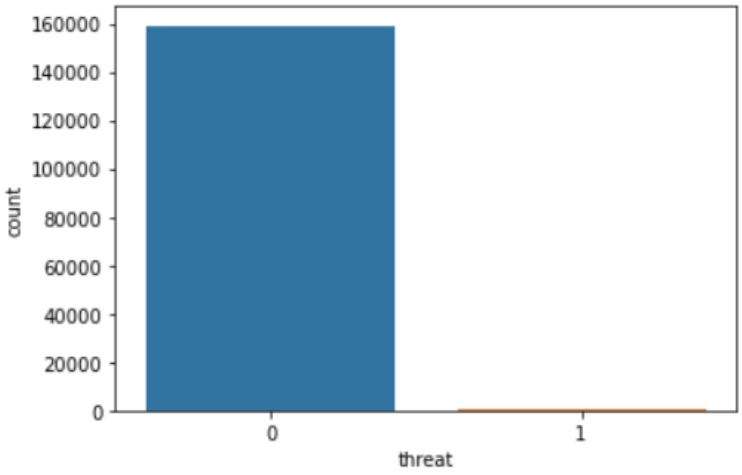
From the above image, we can observe that there are 1595 comments that contains inappropriate data and which fall under highly_malignant comments.

```
1 train['threat'].value_counts()
```

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

```
1 sns.countplot(train['threat'])
```

```
<AxesSubplot:xlabel='threat', ylabel='count'>
```



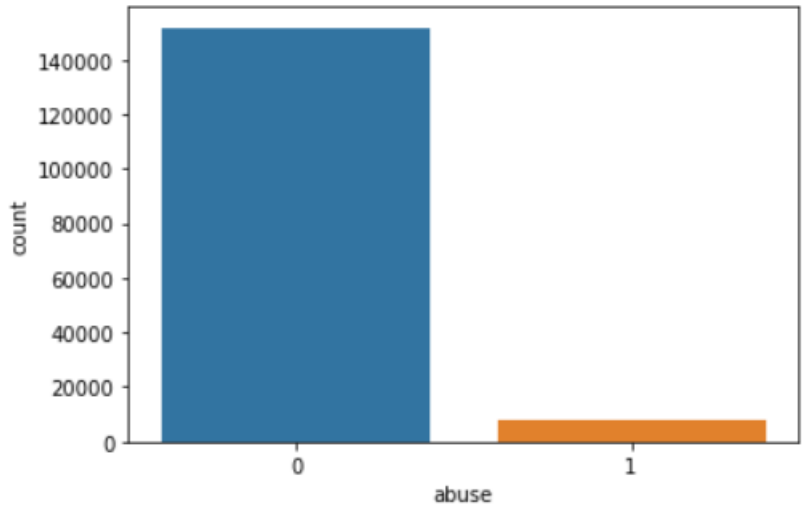
From the above image, we can observe that there are 478 comments that contains inappropriate data and which fall under threat comments.

```
1 train['abuse'].value_counts()
```

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

```
1 sns.countplot(train['abuse'])
```

```
<AxesSubplot:xlabel='abuse', ylabel='count'>
```



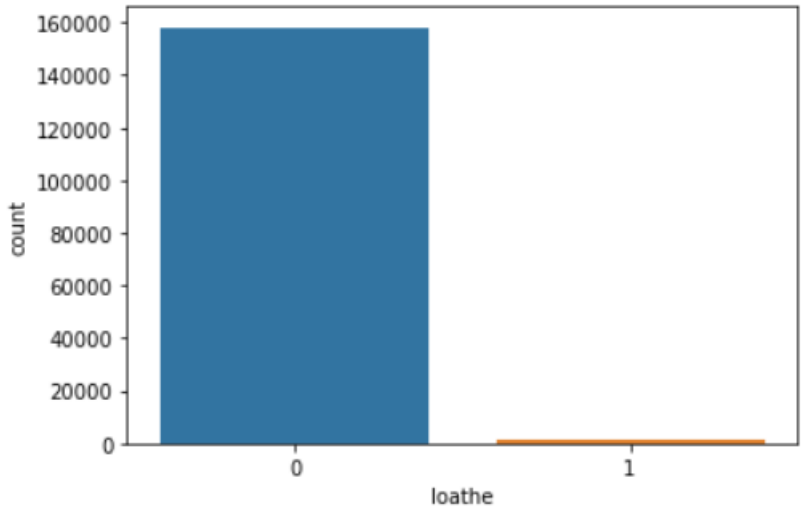
From the above image, we can observe that there are 7877 comments that contains inappropriate data and which fall under abuse comments.

```
1 train['loathe'].value_counts()
```

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

```
1 sns.countplot(train['loathe'])
```

```
<AxesSubplot:xlabel='loathe', ylabel='count'>
```



From the above image, we can observe that there are 1405 comments that contains inappropriate data and which fall under loathe comments.

From all the features we can observe that we have few comments that are malignant which cover approximately 10% of the total comments in the dataset.

Data Sources and their formats

The data was provided by FlipRobo in CSV format. After loading the training dataset into Jupyter Notebook using Pandas and using df.head() it can be seen that there are eight columns named as " id, comment_text, "malignant, highly_malignant, rude, threat, abuse, loathe". Similarly the test file can be load using pandas and the first five rows of the dataset can be seen using df.head() method .The metadata table is provided below for better understanding of the data given [Table 1].

Variable	Definition
id	A unique id aligned with each comment text.
comment_text	It includes the comment text.
malignant	It is a column with binary values depicting which comments are malignant in nature.
highly_malignant	Binary column with labels for highly malignant text.
rude	Binary column with labels for comments that are rude in nature.
threat	Binary column with labels for threatening context in the comments.
abuse	Binary column with labels with abusive behaviour.
loathe	Label to comments that are full of loathe and hatred.

As mentioned earlier the shape of the training dataset is (159571, 8) and the shape of test dataset is (153164,2). The shape of the datasets in form of a tuple can be accessed using df.shape(). The column names of the datasets in form of a list can be seen using df.columns.values() . The datasets have no duplicated values or null values. Both the dataset have no trace of any null or duplicated values. The number of duplicated values of a dataset can be seen using df.duplicated().sum() and the null values can be seen using df.isnull().sum() . The null values can also be visualized with help of seaborn and matplotlib library. Visualization gives a better idea.

Data Pre-processing Done

In order to prepare the text data for the model building we perform text preprocessing. It is the very first step of NLP projects. Some of the preprocessing steps are:

1. Making of corpus by GENERIC NLP FILTER CODE

a) Convert all cases to lower

b) Remove punctuations

c) Remove Stop words

d) Stemming and Lemmatising

2 .Applying embedding technique. (TF-IDF Encoding.)

3. Merging all the output with one column and then predicting the comments.

Cleaning the Comments:

```
# Convert all the comments into lower case
train_df['comment_text'] = train_df['comment_text'].str.lower()
test_df['comment_text'] = test_df['comment_text'].str.lower()
```

```
### Removing Special charactors
charactors=['.',':',';','(',')','!','@','$','^','[^\w\s]','/','?','<','>']
for i in charactors:
    train_df['comment_text'] = train_df['comment_text'].str.replace(i,'')
```

```
## for test dataset:
for i in charactors:
    test_df['comment_text'] = test_df['comment_text'].str.replace(i,'')
```

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

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

```
# Replacing '\n' with ' '
train_df.comment_text = train_df.comment_text.str.replace('\n',' ')
```

```
### cleaning the unwanted text

def unwanted_text(string):
    string = re.sub(r"won't", "will not",string)
    string = re.sub(r"don't", "do not",string)
    string = re.sub(r"doesn't", "does not",string)
    string= re.sub(r"haven't", "have not", string)
    string = re.sub(r"can't", "can not", string)
    string = re.sub(r"im ", "i am", string)
    string = re.sub(r"yo ", "you ",string)
    string = re.sub(r"n\'t", " not", string)
    string = re.sub(r"\'re", " are", string)
    string = re.sub(r"\'s", " is", string)
    string = re.sub(r"\'d", " would", string)
    string = re.sub(r"\'ll", " will", string)
    string = re.sub(r"\'t", " not", string)
    string = re.sub(r"\'ve", " have", string)
    string = re.sub(r"\'m", " am", string)
    string = re.sub(r"<br>", " ", string)
    string = re.sub(r"what's", "what is ", string)
    string = re.sub(r"\'s", " ", string)
    string = re.sub(r"\'ve", " have ", string)
    string = re.sub(r"can't", "cannot ", string)
    string= re.sub(r"n't", " not ", string)
    string= re.sub(r"i'm", "i am ", string)
    string= re.sub(r"\'re", " are ", string)
    string= re.sub(r"\'d", " would ", string)
    string= re.sub(r"\'ll", " will ", string)
    string= re.sub(r"\'scuse", " excuse ",string)
    string= re.sub('\W', ' ', string)
    string= re.sub('\s+', ' ', string)
    string= string.strip(' ')
    ##removing all the urls:
    string = re.sub(r'http\S+', '', string)
    return string
```

```
## cleaning the comments
pd.set_option('display.max_colwidth', -1)
train_df['comment_text'] = train_df['comment_text'].map(lambda comment : unwanted_text(comment))
train_df.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	char_count
0	explanation why the edits made under my username hardcore metallica fan were reverted they werent vandalisms just closure on some gas after i voted at new york dolls fac and please dont remove the template from the talk page since i amretired now892053827	0	0	0	0	0	0	264
1	daww he matches this background colour i amseemingly stuck with thanks talk 2151 january 11 2016 utc	0	0	0	0	0	0	112
2	hey man i amreally not trying to edit war its just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page he seems to care more about the formatting than the actual info	0	0	0	0	0	0	233

```
## Removing the digits or numbers from the comments:
train_df['comment_text'] = train_df['comment_text'].apply(lambda element: re.sub(r"\d+", "", element))

## for text dataset
test_df['comment_text'] = test_df['comment_text'].apply(lambda element: re.sub(r"\d+", "", element))
```

```
# Replacing '\n' with ' '
train_df.comment_text = train_df.comment_text.str.replace('\n',' ')
test_df.comment_text=test_df.comment_text.str.replace('\n',' ')

# Removing all the stopwords
stop_charactors = stopwords.words('english')
train_df.comment_text= train_df.comment_text.apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_charactors)]))
test_df.comment_text= test_df.comment_text.apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_charactors)]))
```

```
## replacing space key
space_key=['\n','_','-']
for j in space_key:
    train_df['comment_text'] = train_df['comment_text'].str.replace(j,' ')
```

```
for j in space_key:
    test_df['comment_text'] = test_df['comment_text'].str.replace(j,' ')
```

```
# Removing punctuations
train_df.comment_text = train_df.comment_text.str.replace("[^\w\d\s]","")
test_df.comment_text = test_df.comment_text.str.replace("[^\w\d\s]","")
```

```
# Stemming words
snb_stem = SnowballStemmer('english')
train_df.comment_text = train_df.comment_text.apply(lambda x: ' '.join(snb_stem.stem(word) for word in word_tokenize(x)))
test_df.comment_text = test_df.comment_text.apply(lambda x: ' '.join(snb_stem.stem(word) for word in word_tokenize(x)))
```

Handling Outliers using ZScore:

```
from scipy import stats
from scipy.stats import zscore
```

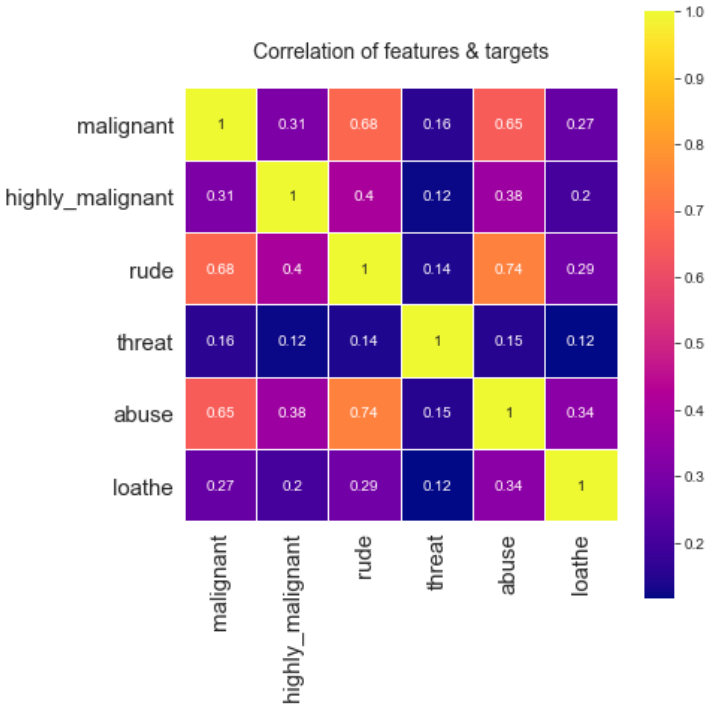
```
z_score=zscore(train_df[['char_count_after']])
abs_zscore=np.abs(z_score)
```

```
threshold=3
new_entry=(abs_zscore<threshold).all(axis=1)
df_new=train_df[new_entry]
print("The shape before: ", train_df.shape)
print("The shape after: ",df_new.shape)
```

```
The shape before:  (159571, 9)
The shape after:  (156135, 9)
```

Data Inputs- Logic- Output Relationships

Correlation with Heat map:-



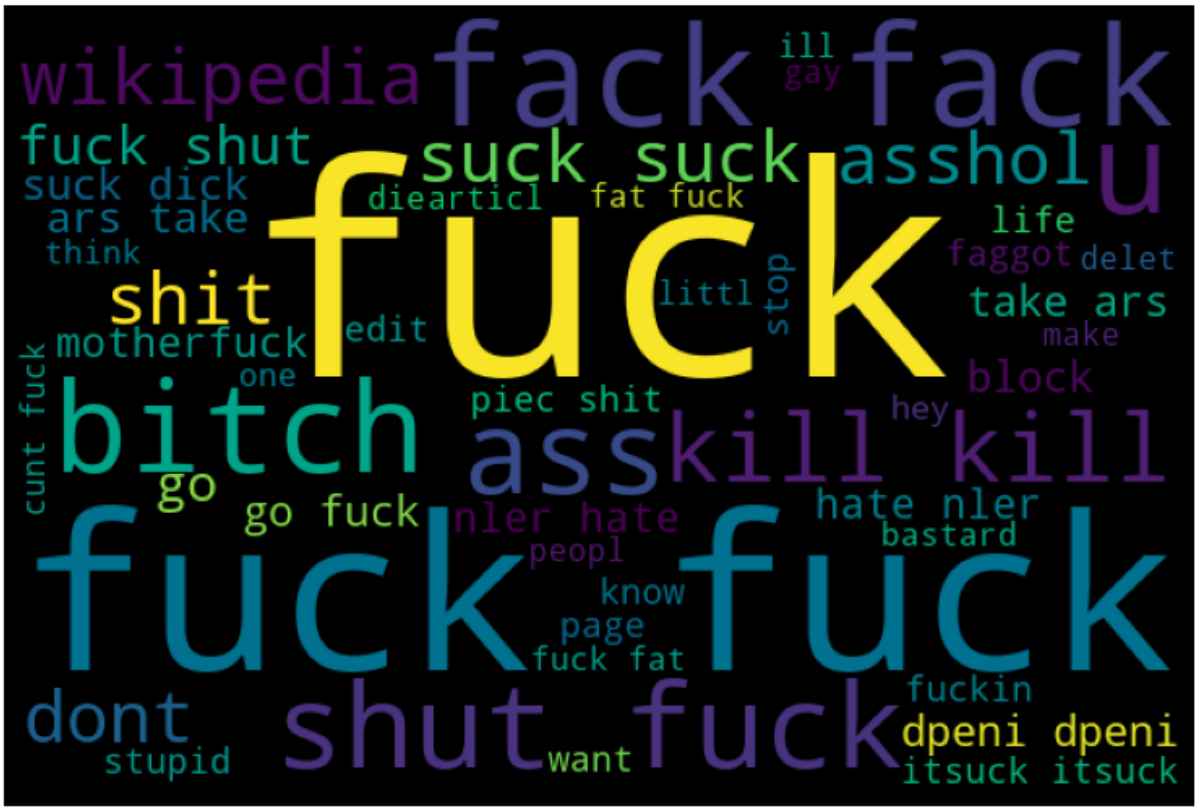
observation of the heatmap:-

- `Rude` & `Abuse` Feature are respectevely 68% and 65% able to tell us about the target variable.
- `Loathe` feature are very less correlated with our Target varibale.
- All features are positively correlated with target.
- `Rude` & `Abuse` Feature are also correlated with each other with 74% correlation, which is showing multi-collinearity problem here.

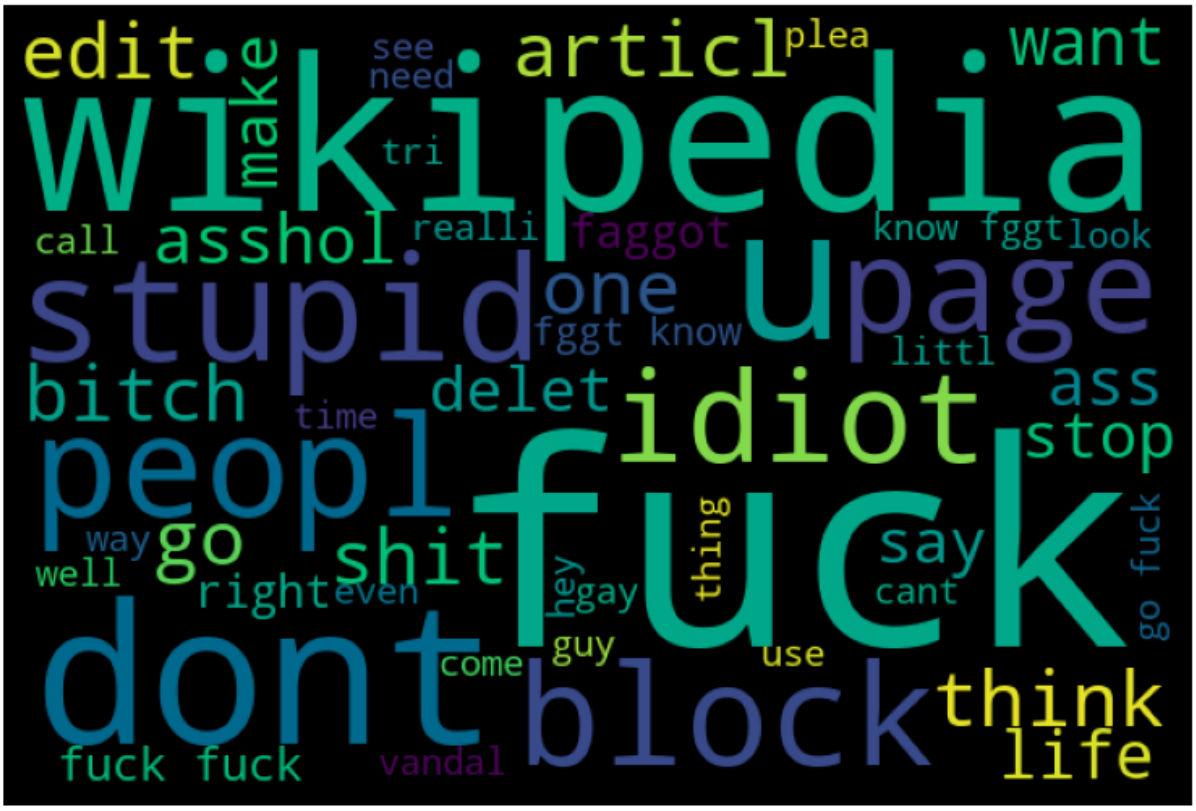
Word Cloud:- Printing Top 50 Accruing Words:

**********Malignant**********

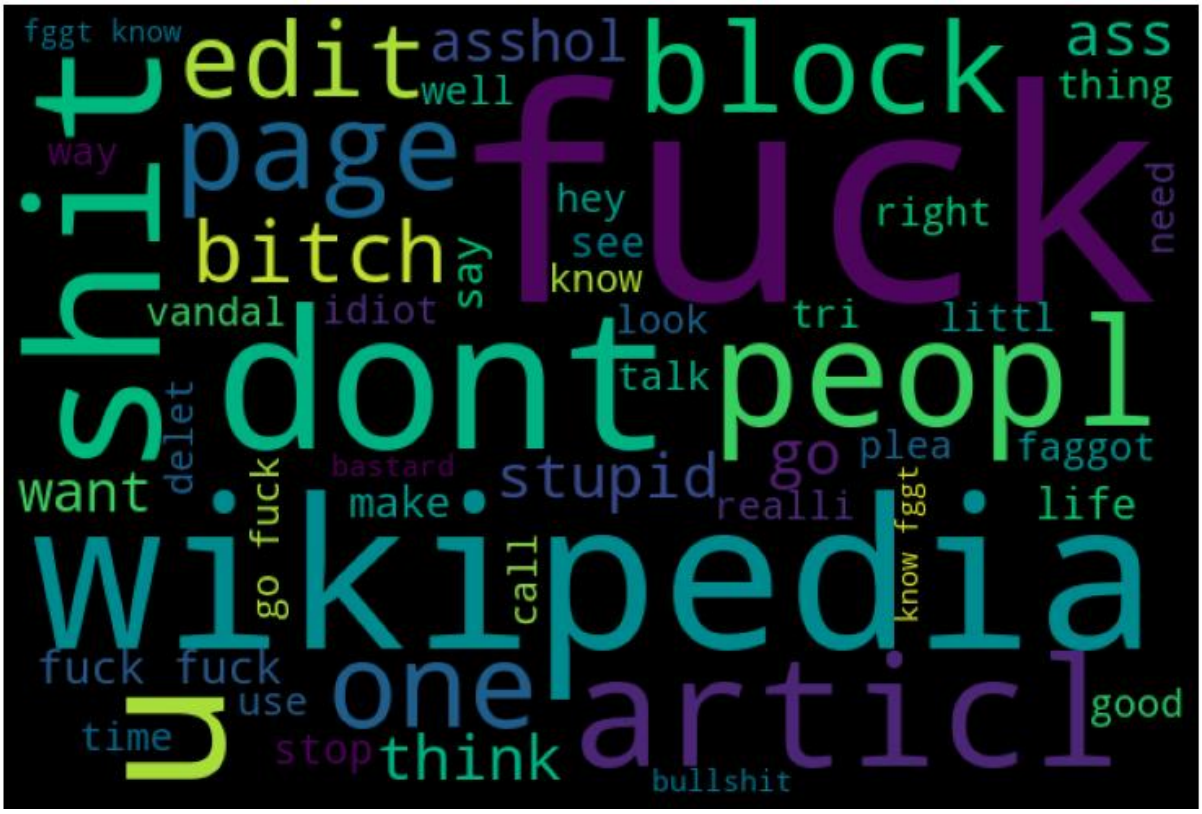
**********Highly_Malignant**********



**********Abuse**********

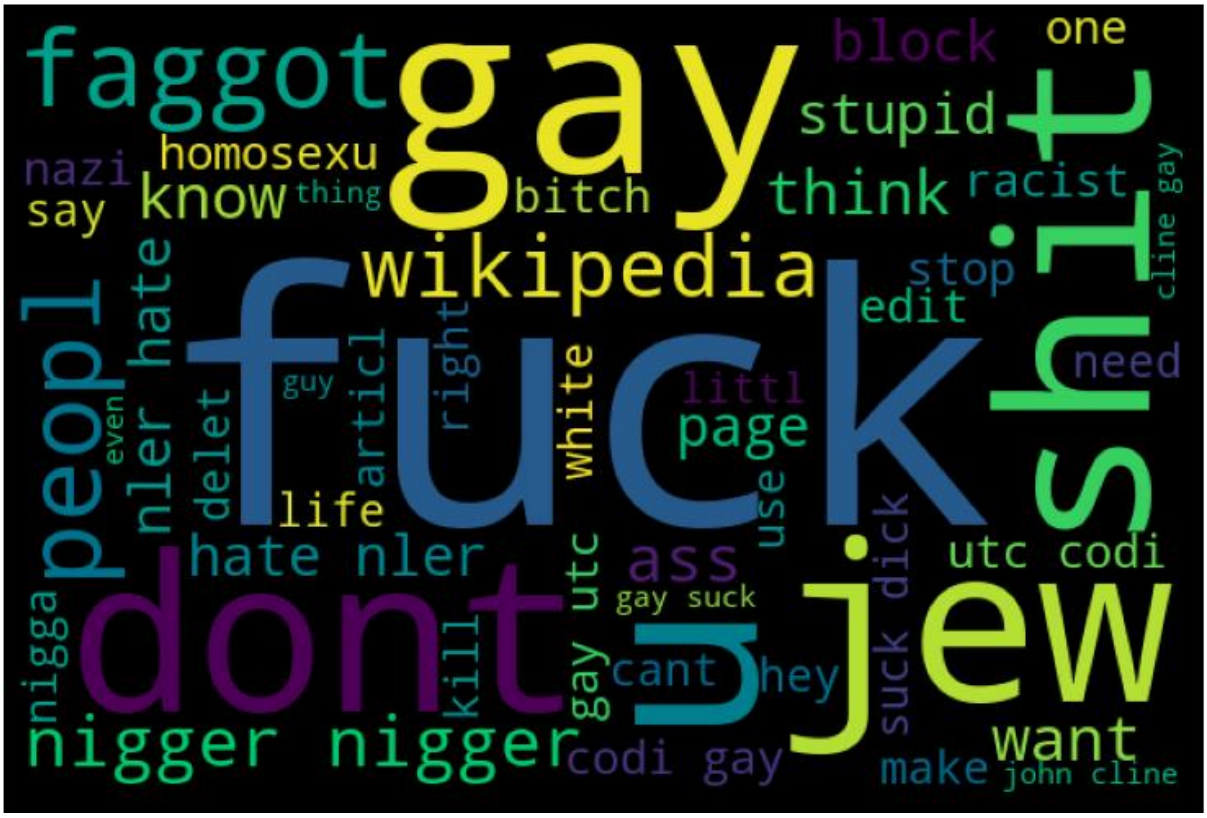

wikipedia
fuck
dont
stupid
upage
peopl
block
idiot
bitch
edit
articl
want
asshol
one
delet
ass
stop
shit
go
say
think
life
make
see
need
plea
tri
call
realli
faggot
know fggt
look
fggt know
littl
time
way
well
right
even
hey
gay
thing
cant
go fuck
guy
come
use
fuck fuck
vandal

**********Rude**********


fuck
shit
dont
peopl
wikipedia
page
block
edit
bitch
articl
one
u
asshol
ass
thing
fggt know
well
way
hey
see
know
right
need
say
vandal
idiot
look
tri
littl
talk
go
plea
faggot
want
delet
bastard
stupid
make
realli
life
go fuck
call
know fggt
fuck fuck
use
time
stop
think
good
bullshit

**********Threat**********

**********Loathe**********



Observation of Word Cloud:

- we can see in wordcloud of malignant comments, it is clear that it mostly consists of words like fuck, nigger, moron, hate, suck ect.
- we can see in wordcloud of highly_malignant comments, it is clear that it mostly consists of words like ass, fuck, bitch, shit, die, suck, faggot ect.
- also we can see in wordcloud of rude comments, it is clear that it mostly consists of words like nigger, ass, fuck, suck, bullshit, bitch etc.
- we can see in wordcloud of threat comments, it is clear that it mostly consists of words like die, must die, kill, murder etc.
- we can see in wordcloud of abuse comments, it is clear that it mostly consists of words like moron, nigger, fat, jew, bitch etc.
- and at the last we can see in wordcloud of loathe comments, it is clear that it mostly consists of words like nigga, stupid, nigger, die, gay cunt etc.

Hardware and Software Requirements and Tools Used

In this project the below mentioned Hardware, IDE, Language, Packages were used.

HARDWARE	LAPTOP: ASUS TUF A17 OS: WIN 10 HOME BASIC PROCESSOR: AMD RYZEN 7 4800H RAM: 16GB VRAM: 6GB NVIDIA GTX 1660Ti
LANGUAGE	Python 3.8
IDE	JUPYTER NOTEBOOK 6.0.3
PACKAGES	PANDAS, NLTK, SKLEARN, MATPLOTLIB, SEABORN

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Pre-Processing for Model Building:

As we already see that we observe that there are 156135 rows and only 6 columns in this final data frame. This leads us to believe that we'll need to add more columns or delete some columns to our spreadsheet to categorize the comments based on our model's confidence.

Now what we'll do add all the columns, it means that ['malignant','highly_malignant','rude','threat','abuse','loathe'] columns will be added. If any comment falls in any category of them then we will assume that that is the 'Bad Comment'.

```
## target_features = df_new[['malignant','highly_malignant','rude','threat','abuse','loathe']]
df_new['Target'] =df_new[['malignant','highly_malignant','rude','threat','abuse','loathe']].sum(axis =1)

### we created char_count, clean_char_count and Clean_word count feature only for visualization purpose.
##So they are not contribute anything in model building so we will drop them now.
df_new.drop(columns=['char_count','char_count_after','Clean_word_count'],axis=1,inplace=True)
df_new.head(20)
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	Target
0	explan edit made usernam hardcor metallica fan revert werent vandal closur gas vote new york doll fac pleas dont remov templat talk page sinc amretir	0	0	0	0	0	0	0
1	daww match background colour amseem stuck thank talk januari utc	0	0	0	0	0	0	0
2	hey man amreal tri edit war guy constant remov relev inform talk edit instead talk page seem care format actual info	0	0	0	0	0	0	0
3	cant make real suggest improv wonder section statist later subsect type accid think refer may need tidi exact format ie date format etc later noon els first prefer format style refer want pleas let know appear backlog articl review guess may delay review turn list relev form eg wikipediagood articl nominationstransport	0	0	0	0	0	0	0
4	sir hero chanc rememb page that	0	0	0	0	0	0	0
5	congratul well use tool well talk	0	0	0	0	0	0	0

```
df_new.Target.value_counts()
```

```
0    140240
1    6240
3    4148
2    3432
4    1677
5    368
6    30
Name: Target, dtype: int64
```

```
## Anything is greater than 0, we will assume that this is a harmful comment.
## adding all of them which are greater then 0
for i in df_new.Target:
    if i in [2,3,4,5,6]:
        df_new.Target=df_new.Target.replace(i,1)
```

```
df_new.Target.value_counts()
```

```
0    140240
1    15895
Name: Target, dtype: int64
```

We have created Target column and separate all the harmful comments and non toxic comments. But we can see that our dataset is seems that our dataset is imbalanced, which leads to our model to biased. So we have to handle it by some over sampling tools.

Vectorization:

```
#  Convert text into vectors using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tf_vector = TfidfVectorizer(max_features = 3000, stop_words='english')
features = tf_vector.fit_transform(df_new['comment_text'])
features.shape
```

```
(156135, 3000)
```

```
x=features
y=df_new.Target
print(x.shape)
print(y.shape)
```

```
(156135, 3000)
(156135,)
```

Testing of Identified Approaches (Algorithms)

```
## importing smote for imbalanced dataset:

from imblearn.over_sampling import SMOTE
```

```
### importing train test
from sklearn.model_selection import train_test_split

### importing the models
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier,ExtraTreesClassifier
rf=RandomForestClassifier()
adc=AdaBoostClassifier()
gbdt=GradientBoostingClassifier()
etc=ExtraTreesClassifier()
from sklearn.svm import SVC
svc=SVC()
from sklearn.linear_model import SGDClassifier
sgdc=SGDClassifier()
from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
mnb=MultinomialNB()
gnb=GaussianNB()
bnb=BernoulliNB()
```

```
### importing Evaluating matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score, precision_score, confusion_matrix, accuracy_score, classification_report
```

Run and Evaluate selected models

Handling Imbalanced data:-

```
over_sampling=SMOTE(0.80)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=5)
x_train_new,y_train_new=over_sampling.fit_resample(x_train,y_train)
```

```
# Lets check the shapes of traning and test data
print("x_train", x_train_new.shape)
print("x_test", x_test.shape)
print("y_train", y_train_new.shape)
print("y_test", y_test.shape)
```

```
x_train (189342, 3000)
x_test (39034, 3000)
y_train (189342,)
y_test (39034,)
```

```
print(y_train_new.value_counts())
print('\n')
print(y_train_new.value_counts(normalize=True))
```

```
0    105190
1    84152
Name: Target, dtype: int64


0    0.555556
1    0.444444
Name: Target, dtype: float64
```

Now we can not say that our data is imbalanced...we have handle that using oversampling method.

Define a function for Printing the Scores:

```
def print_score(Model,independent,dependent,train=True):
    Model.fit(x_train_new,y_train_new)
    if train:
        prediction=Model.predict(x_train_new)
        print('After Oversampling the  new shape of Xtrain is : ',x_train_new.shape)
        print("After oversampling the new shape of Ytrain is : ",y_train_new.shape)
        print("\n*******************Traning Scores**************************\n")
        print("Accuracy Score is {} for Training Model.".format(accuracy_score(y_train_new,prediction)))
        print("\n F1 Score for the model is : \n", f1_score(y_train_new,prediction))
        print("\n*******************Confusion Matrix**************************\n")
        print(confusion_matrix(y_train_new,prediction))
        print("\n\n Training Classification Report \n",classification_report(y_train_new,prediction))

    if train==False:
        pred=Model.predict(x_test)
        print("\n\n")
        print("**********************Testing Scores*****************************\n")
        print("Accuracy score for testing is : ", accuracy_score(y_test,pred))
        print("\n F1 Score for testing is : ", f1_score(y_test,pred))
        print("Confusion Matrix : \n",confusion_matrix(y_test,pred))
        print("\n The Classification report for Testing \n", classification_report(y_test,pred))
```

Defining a function for Cross Validation:

```
def Cross_validation(clf,X,Y):
    clf.fit(x_train_new,y_train_new)
    pred_y=clf.predict(x_test)
    for i in range(5,6):
        score=cross_val_score(clf,X,Y,cv=i)
        mean=score.mean()
        print("\nAt Random State {}, the cross validation score of the model is {}. And accuracy Score is {} ".format(i,mean,accuracy_score(y_test,
        print("\n The difference between corss val score and Accuaracy score is : ", mean-accuracy_score(y_test,pred_y))
```

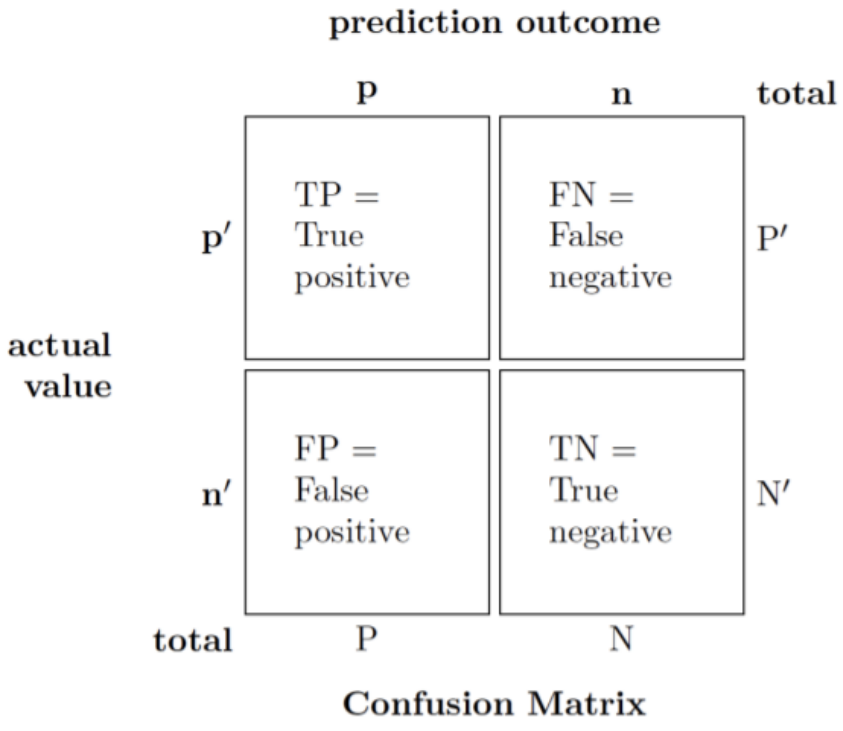
Key Metrics for success in solving problem under consideration

Metrics:

In order to be able to evaluate the performance of each algorithm, several metrics are defined accordingly.

Confusion Matrix:

It is very informative performance measures for classification tasks. Ci,j an element of matrix tells how many of items with label i are classified as label j. Ideally we are looking for diagonal Confusion matrix where no item is miss-classified. The matrix in Figure 1 is a good representation for our binary classification. Positive (P) represents toxic label and n (negative) represents non-toxic label.



Confusion Matrix

Elements of confusion matrix; P (positive) represents toxic label and n (negative) represents non-toxic label.

Accuracy:

This metric measures how many of the comments are labeled correctly. However, in our data set, where most of comments are not toxic, regardless of performance of model, a high accuracy was achieved.

$$Precision := \frac{TP + TN}{N' + P'}$$

Precision and Recall:

Precision and recall in were designed to measure the model performance in its ability to correctly classify the toxic comments. Precision explains what fraction of toxic classified comments are truly toxic, and Recall measures what fraction of toxic comments are labeled correctly.

$$Precision := \frac{TP}{P} \quad Recall := \frac{TP}{P'}$$

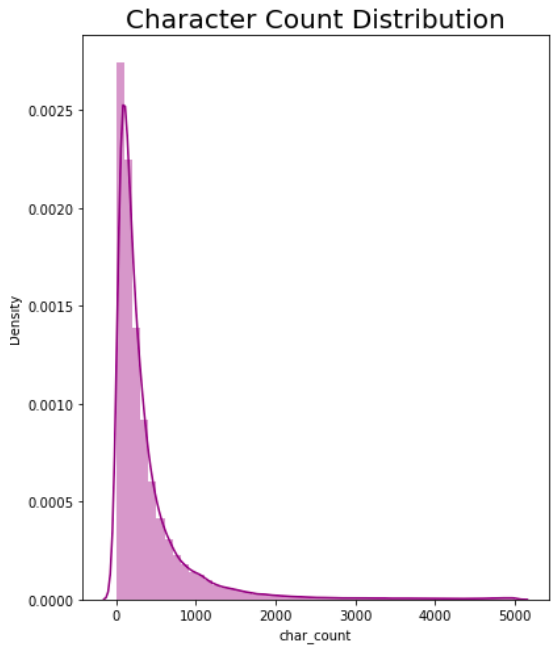
F Score:

Both Precision and Recall are important for checking the performance of the model. However, implementing a more advanced metric that combines both Precision and Recall together is quite informative and applicable (Equation 9). In this equation, setting $\beta = 1$ leads equation to return harmonic mean of Precision and Recall.

$$F_{beta} = (1 + \beta)^2 . \frac{Precision.recall}{(\beta^2 .precision) + recall} \quad Recall := \frac{TP}{P'}$$

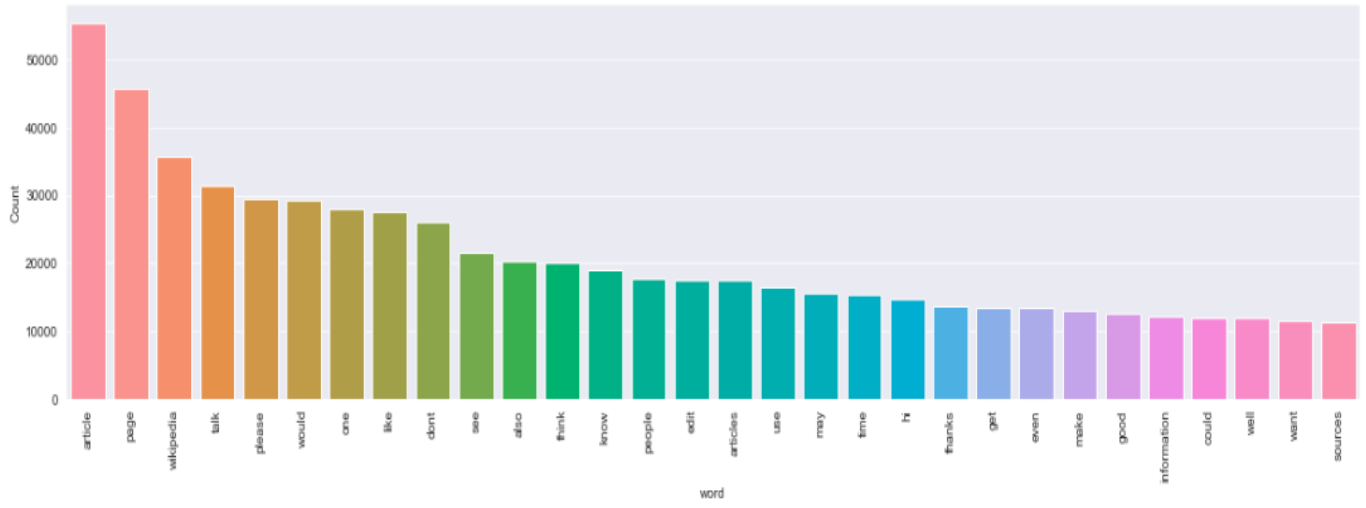
Visualizations

Character Count:-

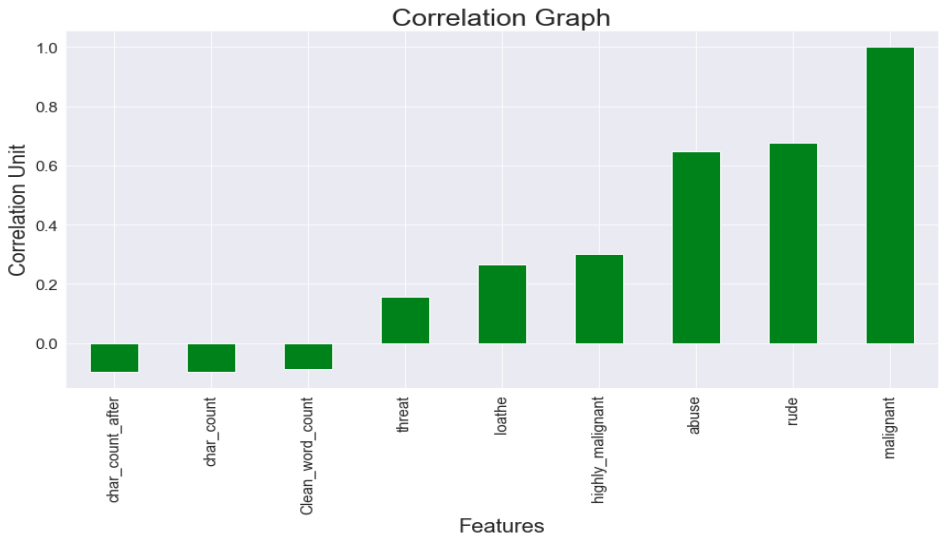


- Maximum comments are in the range of 500 characters.
- Some comments are too long, there are up to 5000 alphabets present.

Most Accruing words:-



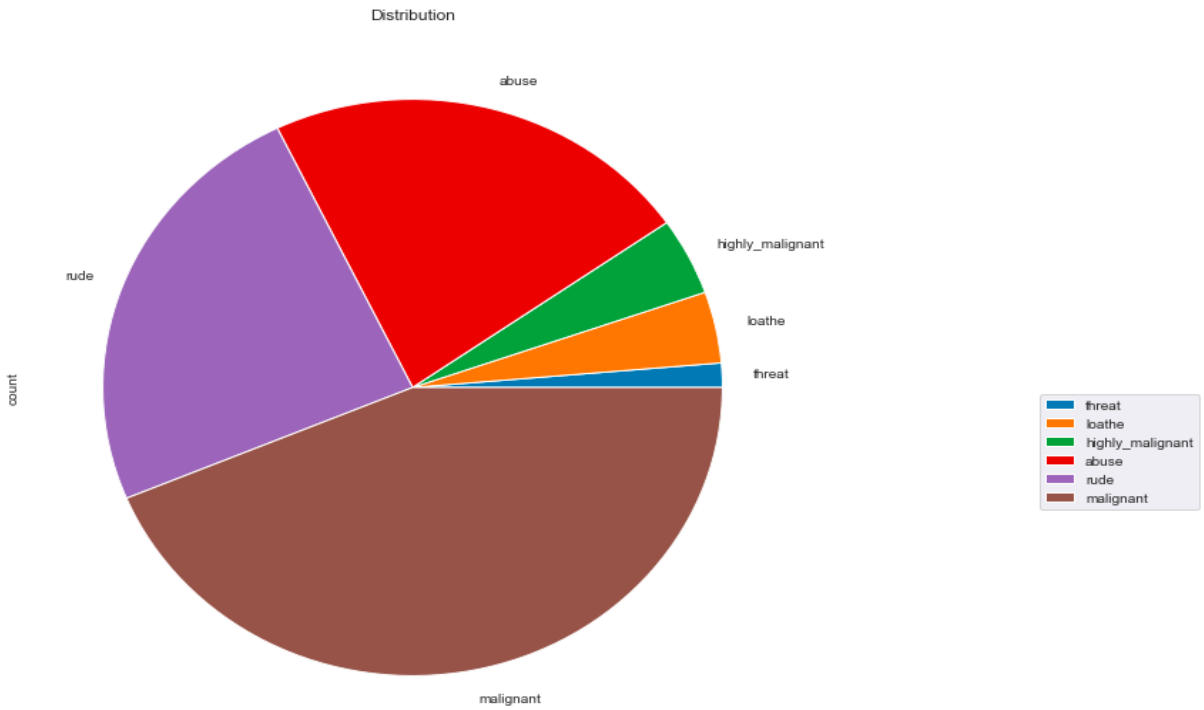
Correlation Graph:



Observations:

- As we assume that abuse and rude are highly correlated with target variable.
- All the character and word count is negatively correlated with the target variable.

Distribution Graph:-



Interpretation of the Results

Model's Name	Testing Accuracy Score	F1 Score	Cross Validation Score	Differnce
Logistic Regression	91.53%	66.80%	95.62%	-4.09%
DecisionTree Classifier	91.55%	63.30%	93.89%	-2.34%
Random Forest Classifier	94.44%	72.21%	95.48%	-1.04%

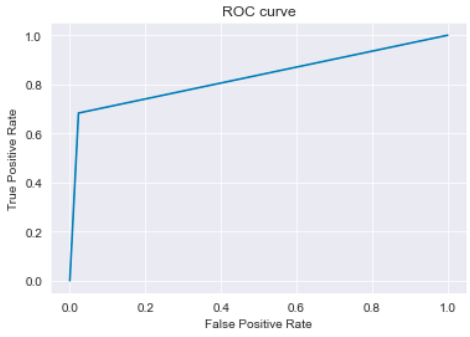
AdaBoostClassifier	93.46%	65.79%	94.73%	-1.27%
GradientBoostingClassifier	94.08%	67.65%	94.25%	-0.17%
ExtraTreesClassifier	94.96%	73.45%	95.47%	-0.51%
SVC	90.83%	65.11%	95.70%	-4.87%
Multinomial NB	92.05%	67.69%	94.62%	-2.57%
Bernouli NB	62.49%	36.25%	76.76%	-14.27%
SGDC Classifier	90.00%	62.68%	94.92%	-4.92%
KNN Classifier	33.23%	22.39%	92.16%	-58.93%

Conclusion:-

As we can see that all the models are showing almost equal accuracy But Based on the F1 Score we observe that Random Forest Classifier & Extra Trees Classifier are showing highest score. In that situation, it is little more difficult to choose our best fit model. So for that we will check the ROC AUC score for Both the models and based on that we will choose our best fit model.

ROC AUC Curve:

```
from sklearn.metrics import roc_curve,auc
fpr, tpr, threshold= roc_curve(y_test,prediction)
plt.plot(fpr, tpr)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC curve")
plt.show()
print("AUC value is {} ".format(auc(fpr, tpr)))
```




```
AUC value is 0.829993103655707
```

CONCLUSION

Key Findings and Conclusions of the Study

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do good amount of data exploration and derive some interesting features using the comments text column available.

We need to build a model that can differentiate between comments and its categories.

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 0denotes 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.

Learning Outcomes of the Study in respect of Data Science

- It is possible to classify the comments content into the required categories of authentic and However, using this kind of project an awareness can be created to know what is fake and authentic.
- The survey discovered that just a small percentage of online users use unparliamentarily language.
- And the majority of these phrases contain a lot of stop words and are pretty lengthy.
- As previously said, a few motivated rude mobs use harsh language in internet forums to harass individuals and prevent them from doing what they are not permitted to do.
- Our research aids online forums and social media in enforcing a prohibition on swearing or the use of profanity on these platforms.

Limitations of this work and Scope for Future Work

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

Areas of improvement:

- Could be provided with a good dataset which does not take more time.
- Less time complexity
- Providing a proper balanced dataset with less errors.