FLIP ROBO

Malignant Comments Classifier

Submitted by-:
NITISH KUMAR SHARMA

ACKNOWLEDGEMENT

It is great pleasure for me to undertake this project. I feel overwhelmed doing this project entitled – “Malignant-Comments-Classifier“.

Some of the resources that helped me to complete this project are as follows:

- Internet/web
- Stack overflow
- Analytics Vidhya
- Articles published in Medium.com
- Wikipedia

INTRODUCTION

BUSINESS PROBLEM FRAMING

This project includes the real time problem for celebrities and influencers mostly as they 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.

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.

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

Online hate, described as abusive language, aggression, cyberbullying, hatefulness, insults personal attacks, provocation, racism, sexism, threats, or toxicity, has been identified as a major threat on online social media platforms. Hate speech refers to a kind of speech that denigrates a person or multiple persons based on their membership to a group, usually defined by race, ethnicity, sexual orientation, gender identity, disability, religion, political affiliation, or views.

Related to this, hate crimes are a type of violation of the law whose primary motivation is the existence of prejudices regarding the victims. This occurs when the offender chooses victims on grounds that they belong to a certain group defined basically by the attributes mentioned earlier. There is evidence that hate crimes are influenced by singular widely publicized events (terrorist attacks, uncontrolled migration, demonstrations, riots, etc.). These events usually act as triggers, and their effect is dramatically increased inside social media. This makes social media a sensor in the real world and a source of valuable information for crime forecasting. In fact, social networks are filled with messages from individuals inciting punishment against different targeted groups. When these messages are collected after a trigger event over a period of time, they can be used for the analysis of hate crimes in all the phases: climbing, stabilization, duration, and decline of the threat. Therefore, monitoring social media becomes a priority for the forecasting, detection and analysis of hate crimes.

REVIEW OF LITERATURE

Worldwide accessibility to the Internet has incredibly reshaped our perception of the world. One of the children of the World Wide Web is Social Media (SM). In recent years, social networks (and especially Twitter) have been used to spread hate messages.

We have got training data where there are six columns referring to categories of offensive words like malignant, highly malignant, rude, threat, abuse and loathe. Then two more columns are id and comment text.

So we have to identify offensive words used in the comments by building a classifier model with good accuracy that will help to recognize those words. Which will further control spreading of hate and cyber bullying.

We have used different machine learning models to predict the above. Since we have categorical target data so classification model algorithms has been used.

We will start our project with the train dataset which contains categories of offensive words. We will observe all the features with following goals in mind:

- Relevance of the features
- Distribution of the features
- Data Cleaning of the features
- Visualization of the features

After having gone through all the features and cleaning the dataset, we will move on to machine learning classification modelling:

- Pre-processing of the dataset for models
- Testing multiple algorithms with multiple evaluation metrics
- Select evaluation metric as per our specific business application
- Doing hyper-parameter tuning using GridSearchCV for the best model
- Finally saving the best model
- Predicting with the test dataset

MOTIVATION FOR THE PROBLEM UNDERTAKEN

It’s a multi-label classification problem to solve which is a completely different project that I ever tried. So it will be exciting for me to get my hands on it.

Cyber bullying on various online platforms is a serious issue which must be stopped but we can’t prevent people from expressing & writing their views, opinions & thoughts without thinking about the consequences. But we can prevent it by building ML models which will be classifying offensive words in the comments and hence restricting it so that it does not spread hate.

It will be a kind of social work to do because we can protect peoples especially celebrities, influencers etc from cyber bullying by classifying hate and offensive comments, so that we can minimize harm.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL / ANALYTICAL MODELING OF THE PROBLEM

The dataset contains two csv files: train data with 8 features and test data with 2 features. The attributes are id, comment_text, malignant, highly malignant, rude, threat, abuse and loathe. Except id and comment_text rest all are target variable that contains 1 or 0 which means comments is non toxic and toxic respectively.
To know the overview/stats of the dataset , we will be using df.describe() function which gives informations like count, min, max, mean, standard deviation values of features . By plotting of heat map we can correlate features if they are highly inter correlated or not. So we can drop columns if problem of multicollinearity arises (if vif >5) when we observe through variance inflation factor. But that's not the case here in such type of problems.
From an initial statistical overview of the dataset, we infer that our target variable is of multi label binary classification.

DATA SOURCES AND THEIR FORMATS

The train and test data that I received was in CSV format (Comma Separated Values). After reading the train data by using function df=pd.read_csv (‘ ---file path --- ’) in jupyter notebook there were 159571 rows and 8 columns. For Test data there were 153164 rows and 2 columns.

Dataset/Attributes Description :

1- ID: It includes unique Ids associated with each comment text given.

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

3- Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.

4- Highly Malignant: It denotes comments that are highly malignant and hurtful.

5- Rude: It denotes comments that are very rude and offensive.

6- Threat: It contains indication of the comments that are giving any threat to someone.

7- Abuse: It is for comments that are abusive in nature.

8- Loathe: It describes the comments which are hateful and loathing in nature.

Train dataset datatypes are as follow :

```
df1.info() # to know datatype of each columns in train data
```

```
<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 dataset datatypes are as follow :

```
df2.info() # to know datatype of each columns in test data
```

```
<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
```

To check datatype of Train & Test data

```
df1.info() # to know datatype of each columns in train data
```

```
<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
```

```
df2.info() # to know datatype of each columns in test data
```

```
<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
```

DATA PREPROCESSING DONE

Imported RegEx (regular expression) to identify sequence of characters that specifies a search pattern, installed nltk library for words classification, tokenizing, stemming, parse tree visualization etc and wordcloud for visualizing loudwords used in the comments.

```
def text_cleaner(text):
    clean_text = re.sub(r'@[A-Za-z0-9]+','',text)
    clean_text = re.sub('#','',clean_text)
    clean_text = re.sub(r"'s\b",'',clean_text)
    clean_text = re.sub(r'[%$#@&}{]','',clean_text)
    clean_text = re.sub(r'[.,:;!]','',clean_text)
    letters_only = re.sub("[^a-zA-Z]",' ',clean_text)

    lower_case = letters_only.lower()
    tokens = [w for w in lower_case.split() if not w in stop_words]
    clean_text=''
    for i in tokens:
        clean_text = clean_text + lemmatizer.lemmatize(i)+ ' '
    return clean_text.strip()
```

- Converting all comments text into lower case
- Removing punctuation from comment text column
- Removing stopwords from comment text column
- Removing digits from the comment text column
- Using Lemmatizing for converting a word to its base form

```
cleaned_text=[]
for i in df1['comment_text']:
    cleaned_text.append(text_cleaner(i))
```

```
df1['cleaned_comments'] = cleaned_text
```

- After doing all data cleaning on comment text column then storing it to a empty list named cleaned text and then storing to a dataframe cleaned comments.

```
cleaned_text=[]
for i in df2['comment_text']:
    cleaned_text.append(text_cleaner(i))
```

```
df2['cleaned_comments'] = cleaned_text
```

- Doing same data cleaning for test data as we did earlier for train data.

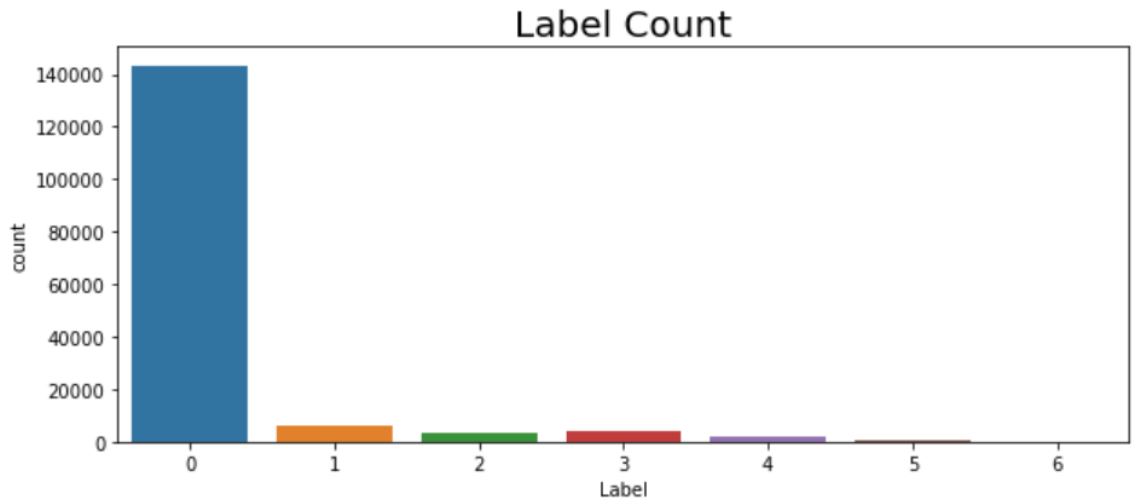
```
df1=df1.drop(columns=['comment_text','id'])
df2=df2.drop(columns=['comment_text','id'])
```

- Dropping columns id and comment text as they are now not useful for further prediction.

DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

```
# create a label feature, which is combination of all target columns.
all_labels = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df1['Label'] = df1[all_labels].sum(axis=1)
```

```
# plot label column count
plt.figure(figsize=(10,4))
sns.countplot(df1['Label'])
plt.title("Label Count",fontsize=20)
plt.show()
```



- After creating a feature named label and storing all the target columns in it.

HARDWARE / SOFTWARE REQUIREMENTS AND TOOLS USED

- Anaconda Navigator 1.10.0
- Jupyter Notebook 6.1.4 , Python 3

Libraries

```
import pandas as pd # for handling dataset
import numpy as np  # for mathematical computation

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
                            plot_roc_curve, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier

from sklearn.decomposition import PCA
from scipy.stats import skew

# for visualization
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import style
import seaborn as sns
import pickle

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

Libraries and Packages used:

- Pickle for saving & loading machine learning model.
- GridSearchCV for Hyper-parameter tuning.
- Cross validation score to cross check if the model is overfitting or not.
- Seaborn and matplotlib for visualization.
- Wordcloud to visualize sense of loud words used
- Lemmatizing for converting words to its base form

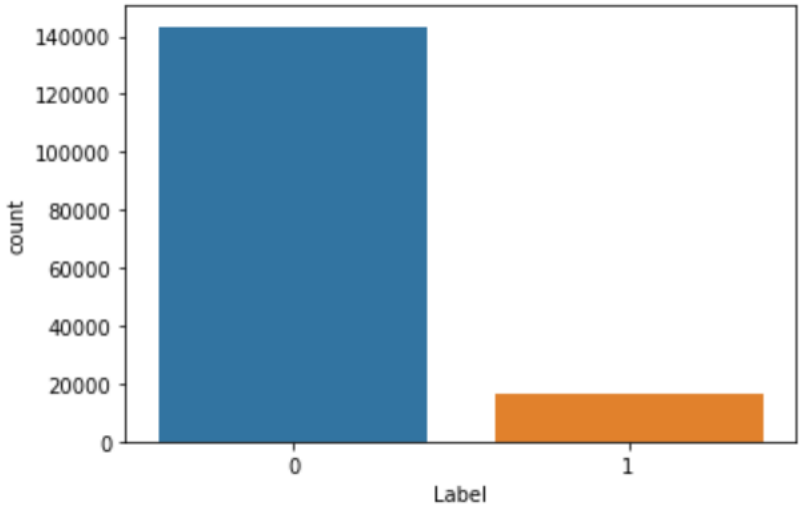
MODEL/s DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

```
# Here, we convert label column in form of 0 and 1 (scaling).
df1['Label'] = df1['Label']>0
df1['Label'] = df1['Label'].astype(int)
```

```
# Here, we plot our label column
sns.countplot(df1['Label'])
plt.show()

df1['Label'].value_counts()
```



- From the above we can observe that the data was highly imbalanced so we have used SMOTETomek to balance the dataset.

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

Algorithms used:

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- KNN Classifier
- Support Vector Machines
- Gradient Boosting Classifier
- Stochastic Gradient Descent

RUN AND EVALUATE SELECTED MODELS

1.Logistic Regression

```
log_reg = LogisticRegression()
log_reg.fit(x_train,y_train)
```

```
LogisticRegression()
```

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

```
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

```
0.9289551155944359
```

```
# Confusion Matrix
conf_mat =confusion_matrix(y_test,y_pred)
conf_mat
```

```
array([[32881,  3013],
       [ 2079, 33700]], dtype=int64)
```

```
print('\n----------------Classification Report--------------------')
print (classification_report(y_test,y_pred,digits=2))
```

```
----------------Classification Report--------------------
              precision    recall  f1-score   support

           0       0.94      0.92      0.93     35894
           1       0.92      0.94      0.93     35779

    accuracy                           0.93     71673
   macro avg       0.93      0.93      0.93     71673
weighted avg       0.93      0.93      0.93     71673
```

Accuracy score : 92.89%

2. Decision Tree Classifier

```
dt_clf = DecisionTreeClassifier()
dt_clf.fit(x_train,y_train)
pred=dt_clf.predict(x_train)
dt_clf_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{dt_clf_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#********************* Test Score ***********************

pred = dt_clf.predict(x_test)
dt_clf_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{dt_clf_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

Training Result : 99.71% Test Result : 94.44%

3. Random Forest Classifier

```
rand_clf = RandomForestClassifier(random_state=101)
rand_clf.fit(x_train,y_train)
pred=dt_clf.predict(x_train)
rand_clf_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{rand_clf_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#********************* Test Score ***********************

pred = rand_clf.predict(x_test)
rand_clf_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{rand_clf_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

Training Result : 99.71% Test Result : 97.52%

4. KNN Classifier

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
pred=knn.predict(x_train)
knn_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{knn_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#********************** Test Score ************************

pred = knn.predict(x_test)
knn_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{knn_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

Training Result : 66.91% Test Result : 61.50%

5. Support-Vector Machines

```
svc = SVC(kernel = 'rbf',C=1)
svc.fit(x_train,y_train)
pred=svc.predict(x_train)
svc_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{svc_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#********************** Test Score ************************

pred = svc.predict(x_test)
svc_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{svc_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

Training Result : 99.42% Test Result : 98.50%

6. Gradient Boosting Classifier

```
gbdt_clf = GradientBoostingClassifier()
gbdt_clf.fit(x_train,y_train)
pred=gbdt_clf.predict(x_train)
gbdt_clf_report = pd.DataFrame(classification_report(y_train,pred, output_dict=True))

print("\n==================Train Result=====================")

print(f"Accuracy Score: {accuracy_score(y_train,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{gbdt_clf_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_train,pred)}\n")

#********************* Test Score ************************

pred = gbdt_clf.predict(x_test)
gbdt_clf_report = pd.DataFrame(classification_report(y_test,pred, output_dict=True))
print("\n==================Test Result=====================")
print(f"Accuracy Score: {accuracy_score(y_test,pred) * 100:.2f}%")
print("_____________________________________________")
print(f"CLASSIFICATION REPORT:\n{gbdt_clf_report}")
print("_____________________________________________")
print(f"Confusion Matrix:\n {confusion_matrix(y_test,pred)}\n")
```

Training Result : 83.45% Test Result : 83.47%

7. Stochastic Gradient Descent

```
sgd=SGDClassifier(loss='modified_huber',shuffle=True,random_state=101)
sgd.fit(x_train,y_train)
y_pred = sgd.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
accuracy
```

Test Result : 92.45%

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

Precision: It is the ratio between the True Positives and all the Positives. It can be seen as a measure of quality, higher precision means that an algorithm returns more relevant results than irrelevant ones

Recall : It is the measure of our model correctly identifying True Positives. It is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

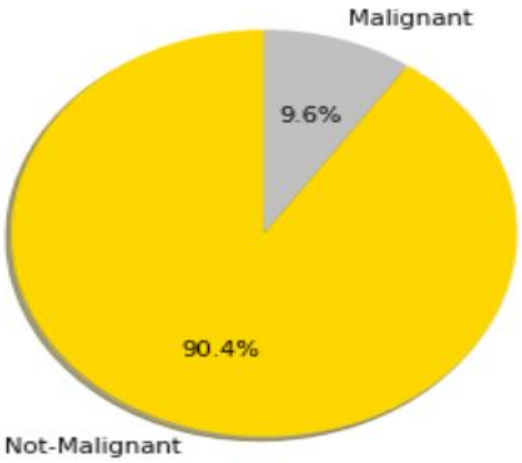
Accuracy score : It is the ratio of the total number of correct predictions and the total number of predictions. It is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar

F1-score : It is used when the False Negatives and False Positives are crucial. Hence F1-score is a better metric when there are imbalanced classes.

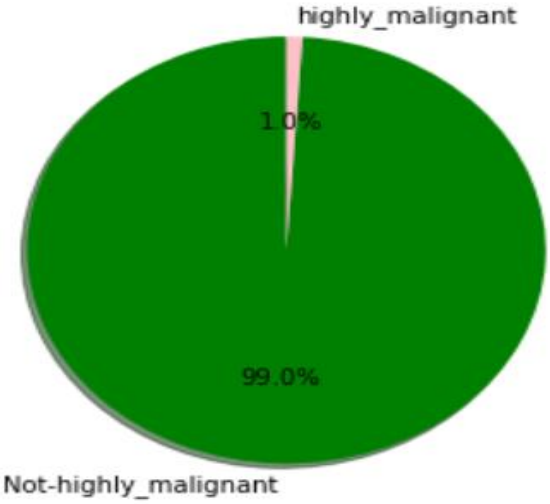
Cross_val_score : To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross- validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

roc _auc _score : ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

VISUALIZATION

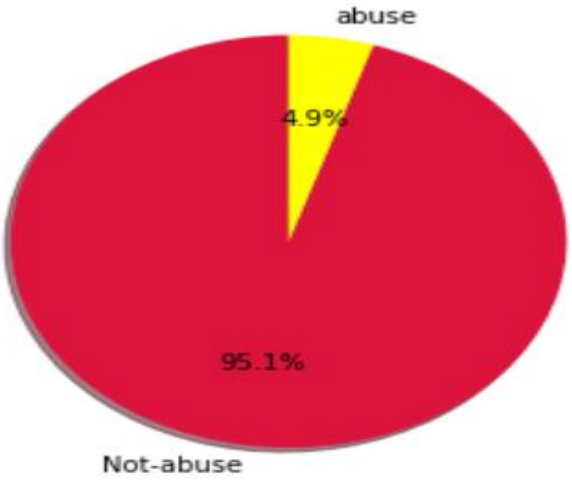


Comments which are stated as malignant is 9.6% and rest 90.4% is not-malignant.

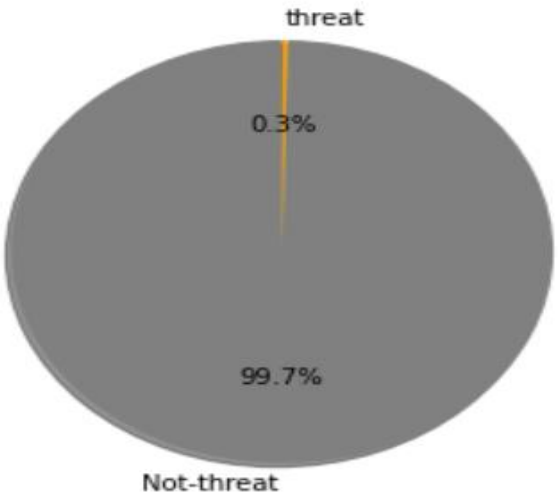


Comments which are stated as highly malignant is 1% and rest 99% is not-highly malignant.

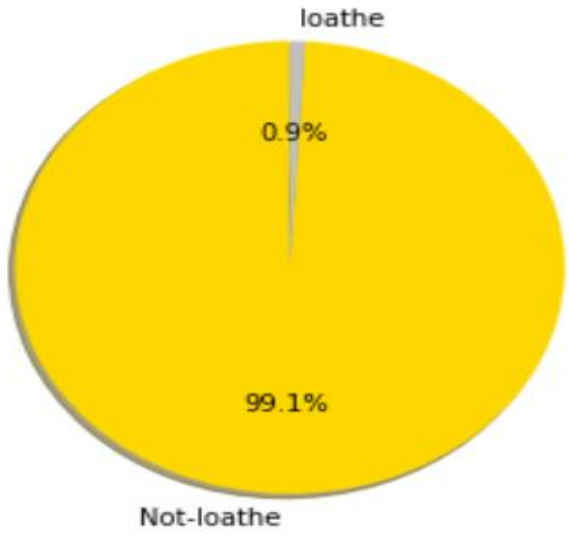
Comments which are stated as rude is 5.3% and rest 94.7% is not-rude.



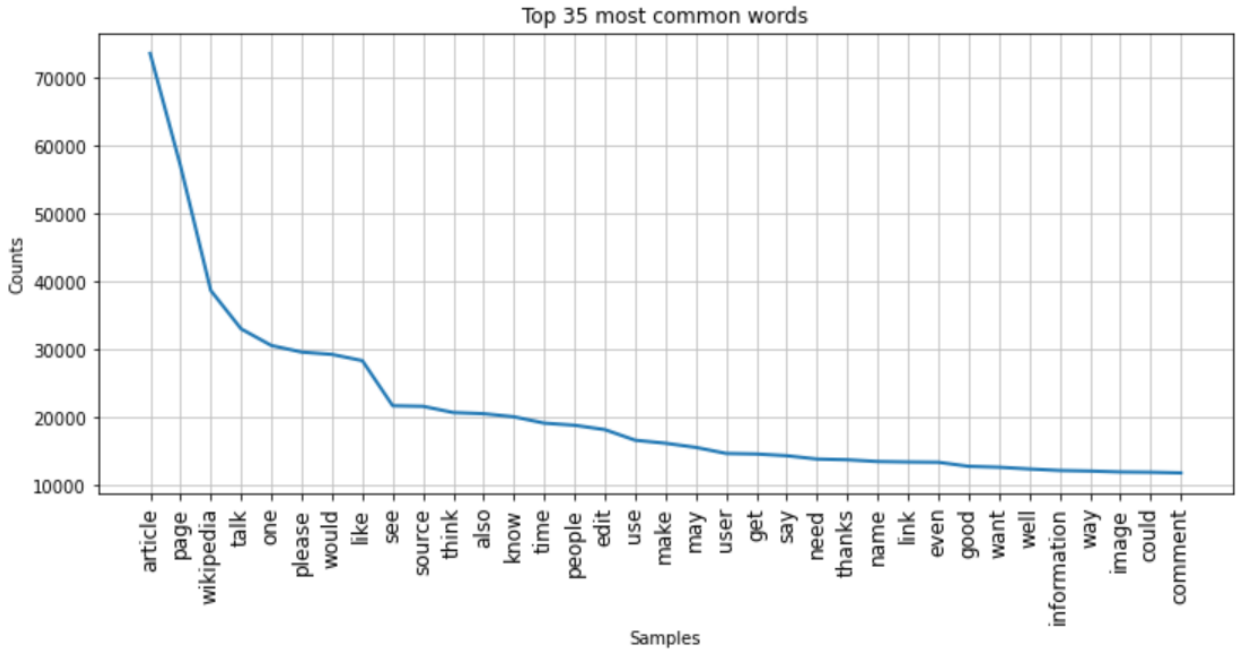
Comments which are stated as abuse is 4.9% and rest 95.1% is not-abuse.



Comments which are stated as threat is 0.3% and rest 99.7% is not-threat.



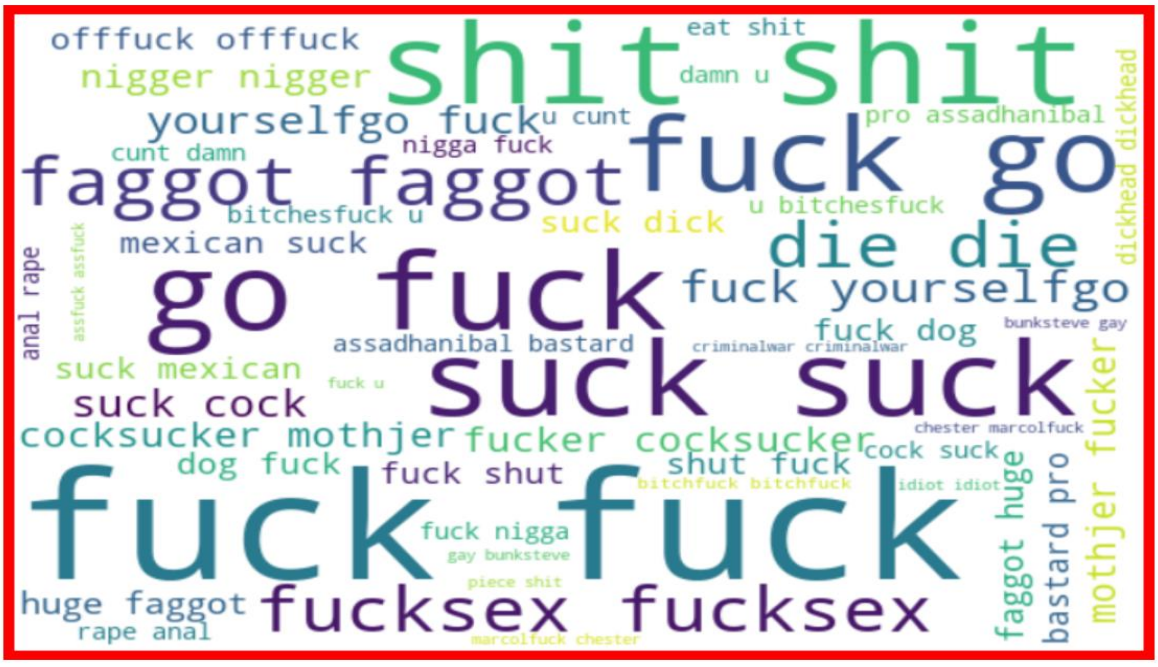
Comments which are stated as loathe is 0.9% and rest 99.1% is not-loathe.



- These are 35 most frequently used common words from the cleaned comments column according to its number of occurences in a descending order.

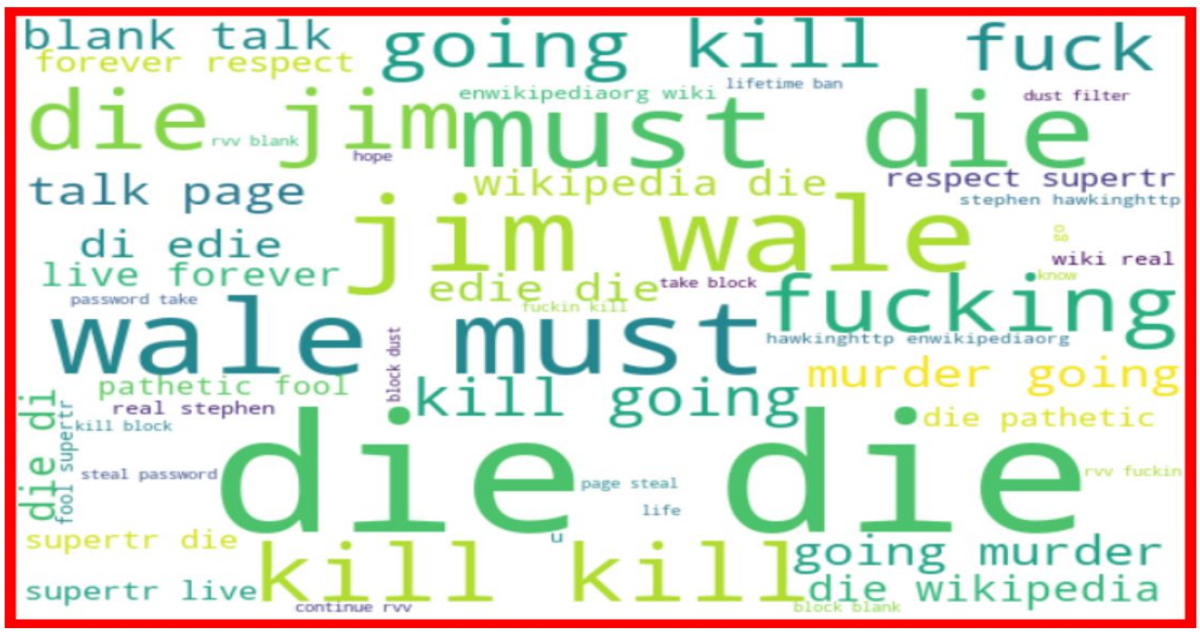
OFFENSIVE WORDS :

- Offensive words appearing in malignant column



- Offensive words appearing in highly malignant column

- Offensive words appearing in rude column



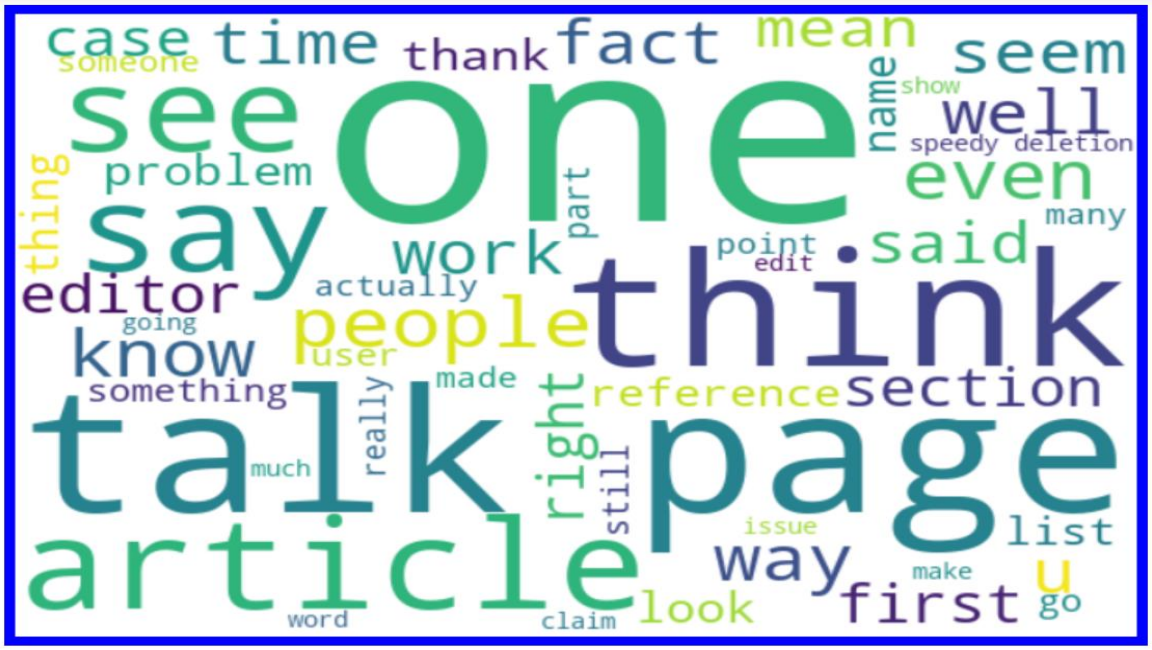
- Offensive words appearing in threat column

- Offensive words appearing in abuse column

- Offensive words appearing in loathe column

NON-OFFENSIVE WORDS :

- Non-Offensive words appearing in malignant column



- Non-Offensive words appearing in highly malignant column

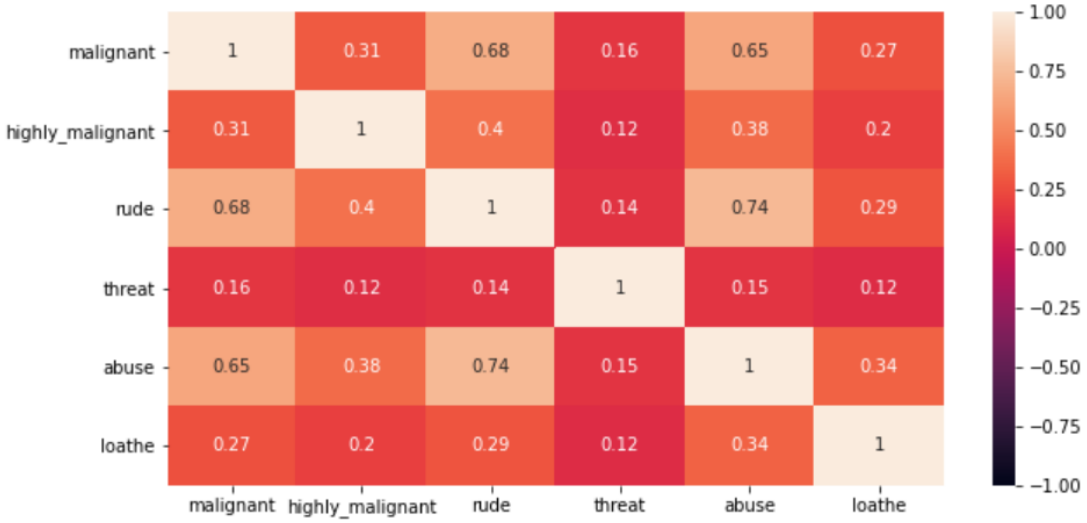
- Non-Offensive words appearing in rude column

- Non-Offensive words appearing in threat column

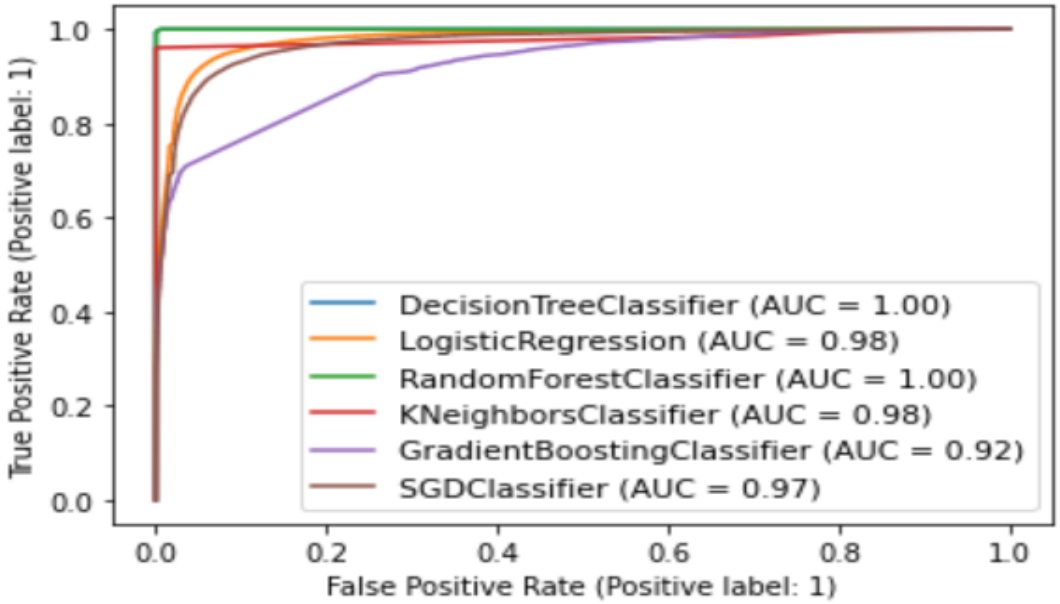
- Non-Offensive words appearing in abuse column

- Non-Offensive words appearing in loathe column

- Plotting of heatmap to check correlation between features



- ROC-AUC Curve for the fitted model



INTERPRETATION OF RESULTS

- From the above study it has been observed that highly abusive words like ‘fuck’, ‘suck’, ‘dickhead’, ‘cock’, etc. has been mostly used.
- Threating words like die, kill, murder, rape has been used.
- Use of Faggot word has been used which refers to a gay man, hence disrespecting their community .
- Mostly abusive words are based on racism, sexist, body shaming etc.
- Black people are abused and targeted by using words like nigger.
- Multinomial models are the best performing models
- In conclusion we can say, malignant comments are widely spread through social media. It not only hurts people’s emotion but also result in act of violence, building model to classify these comments will help to reduce the spread.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

From the whole project evaluation these are the inferences that I could draw from the visualization of data.

- Malignant comments were mostly used to spread hate and cyber bullying.
- Analysis 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.
- The baseline model is a Logistic Regression model fit to TF-IDF vectorized comment text with using only words for tokens, limited to 8000 features.
- These days in social media, data is normally and usually in short messages such as interpersonal organizations, news in website pages etc.
- Cyber Bullying affects an individual feelings and makes him feel embarrassed about oneself.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

Through Visualization it was clear that the target variable was a imbalanced dataset. Also the features that depicted a lot of story telling when visualization was done for all of them. Visualization gives meaning to a data which helps drawing inference from it.

This project allowed me to work with two different deep learning models and additionally, I was able to implement them on a Natural Language Processing use-case. The various data pre-processing and feature engineering steps in the project made me cognizant of the efficient methods that can be used to clean textual data.

Logistic Regression was the best model to be deployed in production because its accuracy and CV value was least among all models.

The most difficult yet most interesting aspect of the project was understanding the relationship between the size of input data and the performance of various machine learning algorithms.

The challenges that I faced while working on this project was that the dataset provided was large, so a lot of time was taken to run the classifier model especially SVM model. This gives the idea when we will deal with millions and trillions of data we have to do proper data cleaning and processing because we cannot run again and again our algorithms as it will take days to complete. Hence time & money loss will be there.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

For further improvement of results, Recurrent neural networks offer extremely high performance on natural language processing problems, and if the architecture for the inference step were implemented efficiently the computational overhead would be minimal.

More accurate and promising results can be obtained by further improving the word embedding by using more finely pre-processed data as well as advancements in developing the proposed LSTM systems.