NAME OF THE PROJECT

“Malignant Comment Classification”

Submitted by:

Sharad Yadav

ACKNOWLEDGMEN

I would like to thank fliprobo team that provided “Malignant comments classification” is current hot topic that how to prevent toxic trolling on social media. Here dataset is provided by fliprobo team in csv format ,to classified toxic comments.

INTRODUCTION

Problem Statement

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

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

There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as 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.

Data Set Description

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

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

The data set includes:

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

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.

Now a day's social media platform become so toxic trolling that required to do some preventive measures.

- **Conceptual Background of the Domain Problem**
  The idea behind comment classification to build classified hate troll toxic comments among also that it can be controlled and restricted from spreading hatred and cyberbullying.
- Review of Literature

Data set 159570 rows and 8 column id, and comment are object data and its classified between malignant, highly_malignant, rude, threat, abuse, loathe are in categories. Troll accounts have become prevalent on numerous social media platform so we will build a prototype model which classified the toxic bad comments.

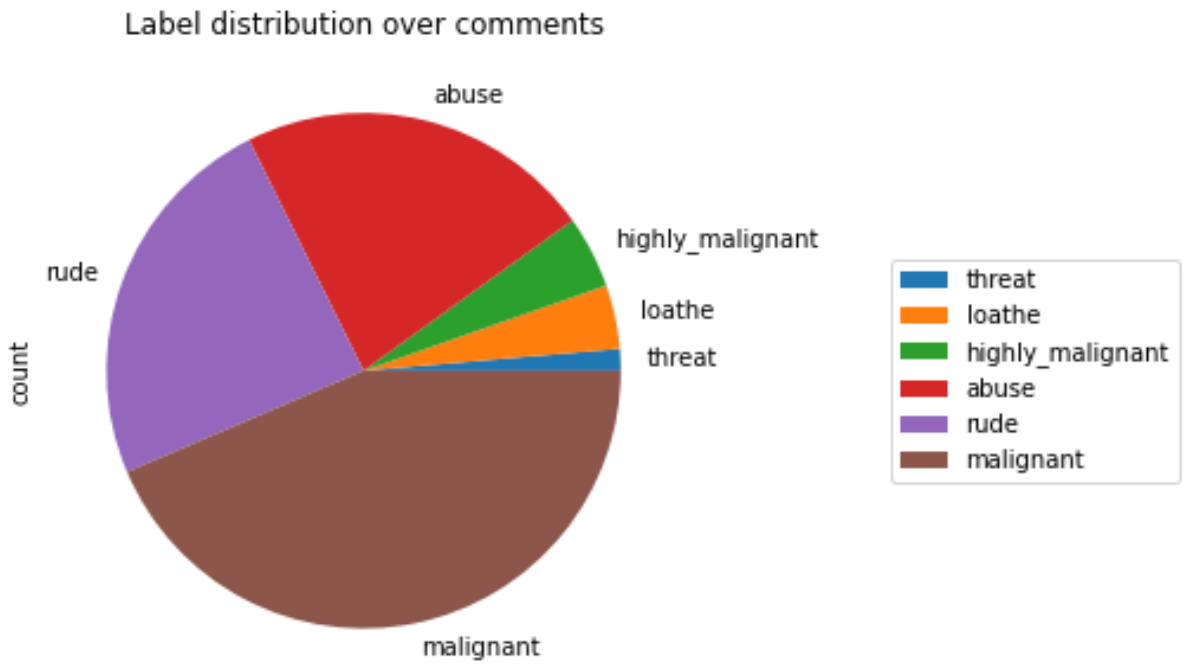
- **Motivation for the Problem Undertaken**
  In recent times as new social platform comes into our life hatred, toxic, trolling comments come in to existence which cause cyberbullying harassment of people which result mental health issue and many other consequences.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

I have to classified toxic bad comments from the dataset malignant rude abusive are most of the cases in given dataset

As per analysis of chart.



- **Data Sources and their formats**

- Dataset is in csv file, to read that data I have to used pandas library to read file further describe method to analysis get overview of data distribution, data info to identify object integer data types. Here this is all about comment which is of course object data I have to visualize more to get insight of toxic comments .

```python
# lets start with importing necessary library
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV
import statsmodels as sm
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.decomposition import PCA
import scikitplot as skplt
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
import pickle
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
import warnings
warnings.filterwarnings("ignore")
```

```python
 import nltk
nltk.download()
```

```python
data.describe()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- Data Preprocessing Done

The following steps were taken to process the data

1. I have downloaded and installed NLTK tool to remove extra punctuation present in comment, word cloud is used for visualization of loud frequent data. To convert a

string of words into a matrix of words with column headers represent by words and their frequency

2. Stop words accepted convert to lower case and regular expression as its parameters.
3. Splitting data set into Training and testing by train_test_split method

```
#comment text cleaning as many / n & so many special character present in comments
# Convert all messages to lower case
data['comment_text'] = data['comment_text'].str.lower()

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

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

# Replace money symbols with 'moneysymbol
data['comment_text'] = data['comment_text'].str.replace(r'£|\$', 'dollers')

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


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


data['comment_text'] = data['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in string.punctuation))

stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
data['comment_text'] = data['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

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

Removing extra lines and string from comments

- Data Inputs- Logic- Output Relationships
  Since comment is huge number and it's difficult to fetch only bad words to vectorization and label to that comments first and then I have applied Logistic ,decision tree and ensambel techniques to classified the data.

- **Hardware and Software Requirements and Tools Used**

Below required sklearn library and various other python techniques required to build model

```
# lets start with importing necessary library
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV
import statsmodels as sm
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.decomposition import PCA
import scikitplot as skplt
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
import pickle
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
import warnings
warnings.filterwarnings("ignore")
```

Model/s Development and Evaluation

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

- Testing of Identified Approaches (Algorithms)
  1. Logistic Regression
  2. Decision Tree Classifier
  3. Gradient Boosting Classifier
  4. Random Forest Classifier

- **Run and Evaluate selected models**

1. Logistic regression

```
1 from sklearn.linear_model import LogisticRegression
```

```
1 Logistic=LogisticRegression()
2 Logistic.fit(x_train,y_train)
```

```
]: LogisticRegression()
```

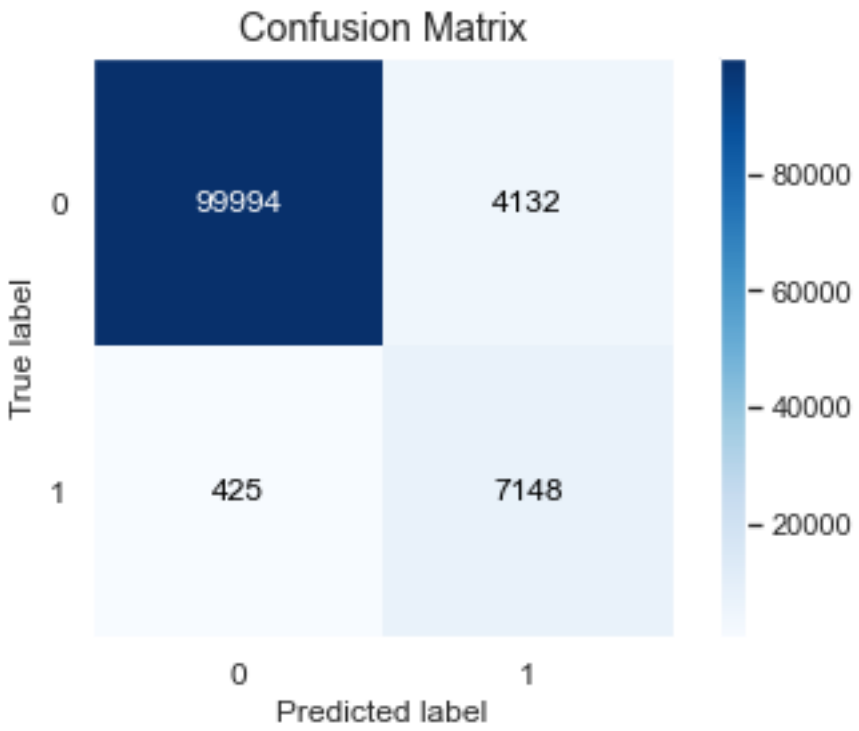
```
1 y_pred=Logistic.predict(x_train)
```

```
1 #classification report
2 print(classification_report(y_train,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98    100419
           1       0.94      0.63      0.76     11280

    accuracy                           0.96    111699
   macro avg       0.95      0.81      0.87    111699
weighted avg       0.96      0.96      0.96    111699
```

```
1 #modle score
2 Logistic.score(x_train,y_train)
```

```
]: 0.959202857680015
```



2. Decision Tree Classifier

```
1 from sklearn.tree import DecisionTreeClassifier
```

```
1 dt=DecisionTreeClassifier()
2 dt.fit(x_train,y_train)
```

```
3]: DecisionTreeClassifier()
```

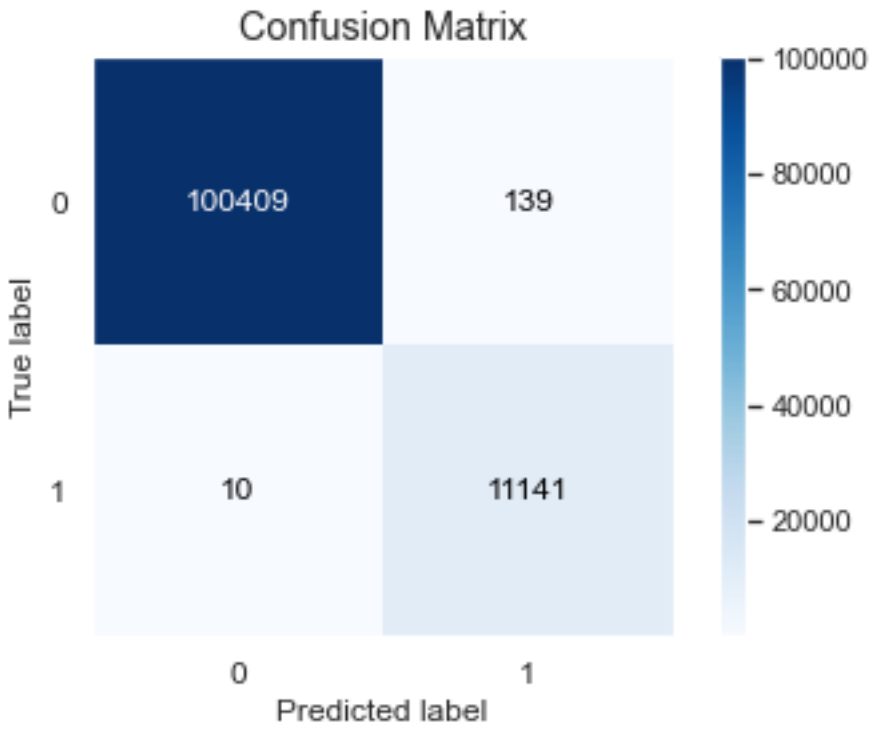
```
1 y_pred=dt.predict(x_train)
```

```
1 #modle score
2 dt.score(x_train,y_train)
```

```
9]: 0.9986660578877161
```

```
1 #classification report
2 print(classification_report(y_train,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    100419
           1       1.00      0.99      0.99     11280

    accuracy                           1.00    111699
   macro avg       1.00      0.99      1.00    111699
weighted avg       1.00      1.00      1.00    111699
```



2. Gradient Boosting Classifier

```
1 from sklearn.ensemble import GradientBoostingClassifier
```

```
1 gbc=GradientBoostingClassifier()
2 gbc.fit(x_train,y_train)
```

```
]: GradientBoostingClassifier()
```

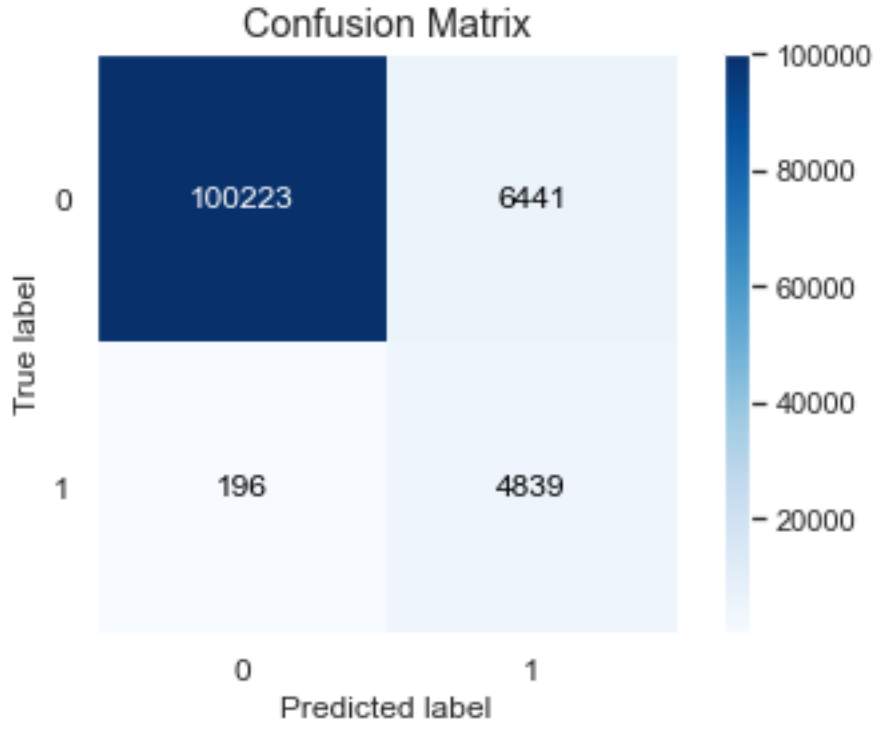
```
1 y_pred=gbc.predict(x_train)
```

```
1 #model score
2 gbc.score(x_train,y_train)
```

```
]: 0.9404649996866579
```

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

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97     45637
           1       0.43      0.96      0.59      2235

    accuracy                           0.94     47872
   macro avg       0.71      0.95      0.78     47872
weighted avg       0.97      0.94      0.95     47872
```



4. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

```
y_pred=rfc.predict(x_train)
```

```
#model score
rfc.score(x_train,y_train)
```

```
0.9987108210458464
```

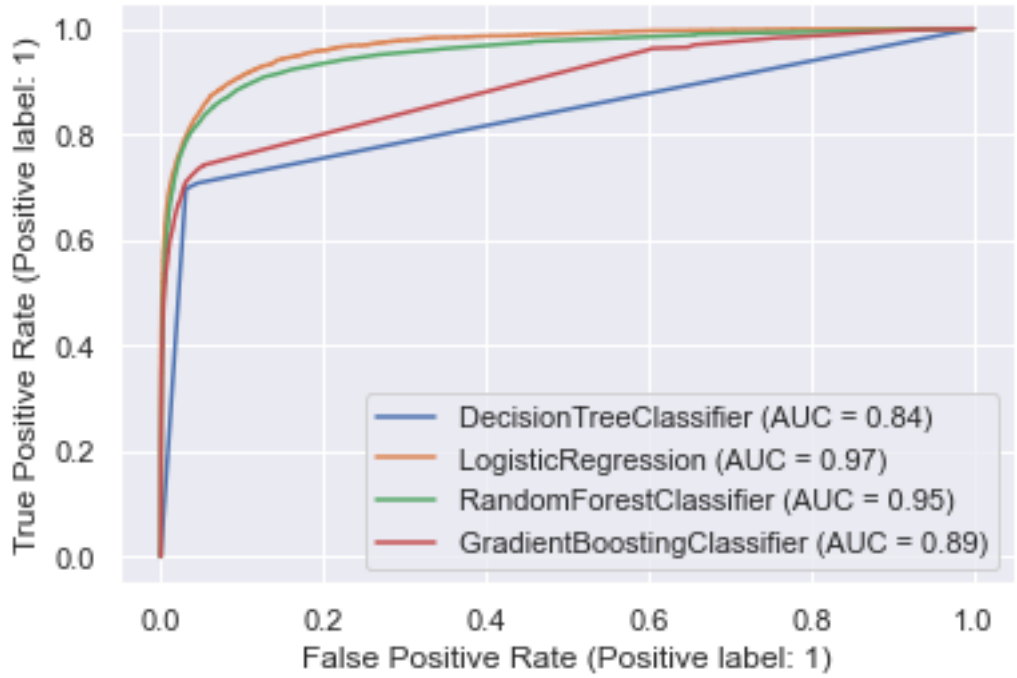
```
print(classification_report(y_pred,y_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    100525
           1       0.99      1.00      0.99     11174

    accuracy                           1.00    111699
   macro avg       0.99      1.00      1.00    111699
weighted avg       1.00      1.00      1.00    111699
```

```
#confusion matrix of Random forest
confusion_matrix(y_pred,y_train)
```

```
array([[100400,    125],
       [    19,  11155]], dtype=int64)
```

Roc _auc plot and model saving

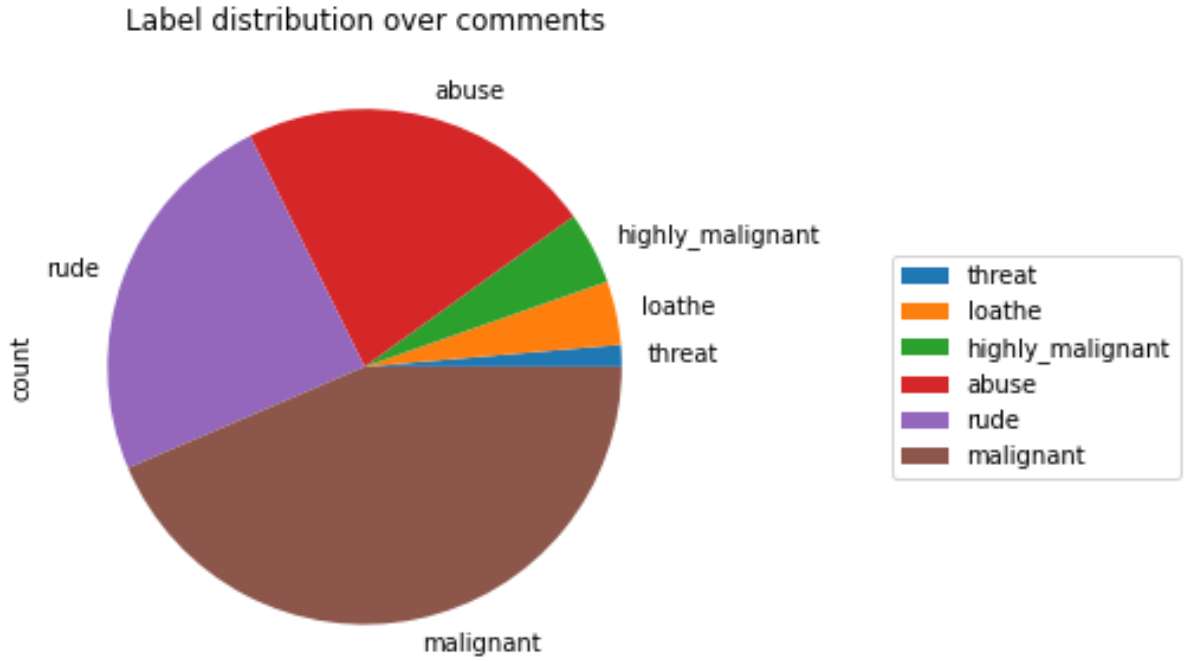



```python
#saving model LogisticRegression
import pickle
file='pikle_rf_model'
with open(file ,'wb') as file:
    pickle.dump(Logistic, file)
```

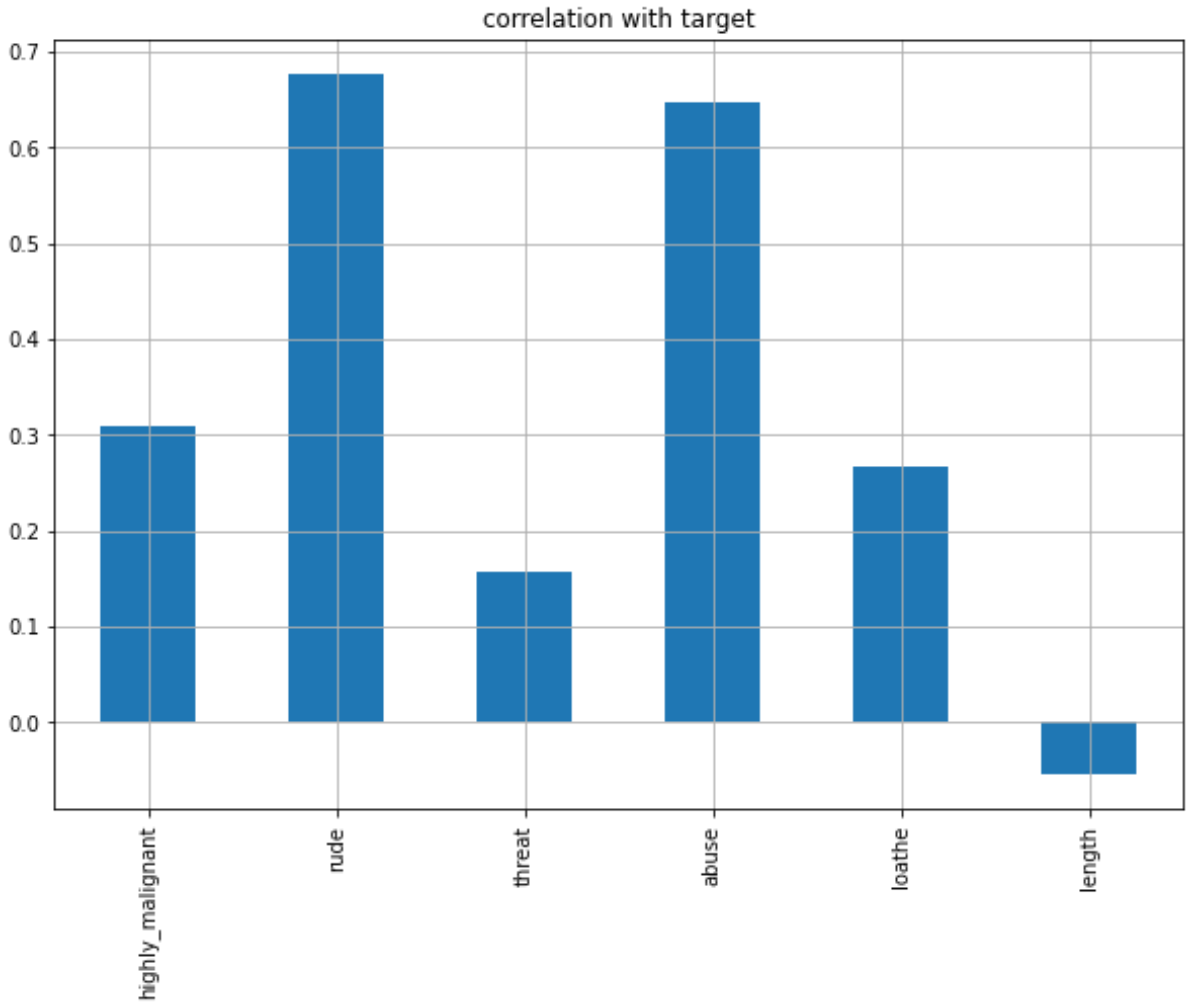
- **Key Metrics for success in solving problem under consideration**

  Our final Model is Logistic regression chosen best among all because of high score type 1 type 2error is less AUC_ROC under curve cover most area ,further we can tune the model by hyper parameter tuning but that required more time as data set is huge.

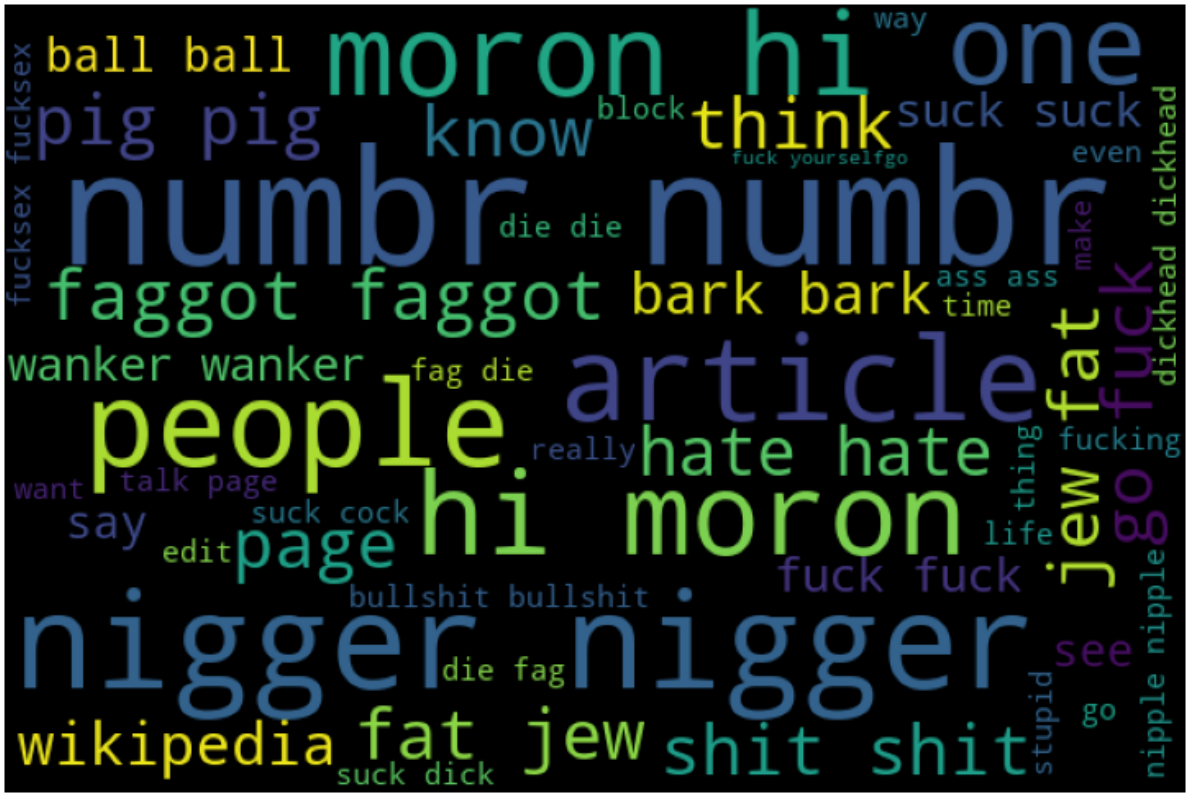
- **Visualizations**



Toxic comment pie chart analysis



Correlation with Target Variable



Loud word among toxic comments

- **Interpretation of the Results**

CONCLUSION

- **Key Findings and Conclusions of the Study**
  I have found some loud toxic word and classified among all comments to prevents such bad words, as social media uses increase troll and toxicity will increase this will definitely hep to prevent to classified bad words and control such offensive comments
- **Learning Outcomes of the Study in respect of Data Science**

This project is quite different and relatable to present scenario, but comments classification and visualization is very challenging data science point of view this need more time to build some analysis of extra features work.

- Limitations of this work and Scope for Future Work

  The current project to identify toxic comments classification, future I will add following points

  Analysis of which age group being toxic towards particular age group.

  Add automatic prevention and somatization of comments.

  Build feedback model to increase efficiency