MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

Vivek Rai

ACKNOWLEDGMENT

I would like to express my sincere gratitude to all those who provided me the chance to take a step in this project. I went through google and Kaggle for the references and my old projects helped me understand the way to solve this use case.

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.

- Conceptual Background of the Domain Problem

  - A large proportion of online comments present on public domains are usually constructive, however a significant proportion are toxic in nature. Dataset is obtained online which are processed to remove noise from the dataset. The comments contain lot of errors which increases the number of features manifold, making the machine learning model to train the dataset by processing the dataset, in the form of transformation of raw comments before feeding it to the Classification models using a machine learning technique known as the term frequency-inverse document frequency (TF-IDF)

technique. The logistic regression technique is used to train the processed dataset, which will differentiate toxic comments from non-toxic comments. The multi-headed model comprises toxicity (severe-toxic, obscene, threat, insult, and identity-hate) or Non-Toxicity Evaluation, using confusion metrics for their prediction.

- So, primarily we must focus which social handles are used mostly among all and what are the motto of particular social handles who is facing this type of comments.

Review of Literature

- Aggression by text is a complex phenomenon, and different knowledge fields try to study and tackle this problem.

- This analysis of related work focuses on a computer science perspective of aggression identification, a recent emerging area. Currently, the scientific study of automatic identification of aggressive text, using information technology techniques, is increasing. In this study, several related literature are used to express different types of aggression. Some of those are hate (Tarasova et al.8), cyber bullying (Adamic9), abusive language (Nobata et al.3), toxicity .

Motivation for the Problem Undertaken

- Because of the most demanding and most exploratory technology i.e. NLP and specifically using machine learning in this type of problem can help me to learn more and explore me more. The following problem statement is also a useful use case because as said earlier comments classification is becoming more important nowadays.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  - Project contains train and test dataset as well.
  - In the train data set there are 159,571 rows and 8 columns.
  - There are no null values in the dataset
  - Most of the data are numeric in nature which are binary.
  - Comments are objects in nature and consist of text.

- Data Sources and their formats

  Train Data

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
id                  159571 non-null object
comment_text        159571 non-null object
malignant           159571 non-null int64
highly_malignant    159571 non-null int64
rude                159571 non-null int64
threat              159571 non-null int64
abuse               159571 non-null int64
loathe              159571 non-null int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

- Data Preprocessing Done

```
# Convert all messages to lower case
train['comment_text'] = train['comment_text'].str.lower()

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

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

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

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


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


train['comment_text'] = train['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'])
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))

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

- As there are no null values in the dataset, but as the comment column in text format so they require a lot of text pre-processing.
- As, the number of offensive comments are very less in number.

- Data Inputs- Logic- Output Relationships

  - There are features which are highly correlated
  - It is possible because one comment can be classified into multiple categories
  - One comment can be rude and abusive at the same time.
  - Most of the text length are within 500 characters, with some up to 5,000 characters long

- State the set of assumptions (if any) related to the problem under consideration

- For the ease of the training and predicting I assumed and made a label of bad words. Means all the bad comments are classify as bad word and all good comments whose value are 0 are classified as good comments
- This will help the model to classify the comment easily. Also, I had made a separate model which helps to predict the prob of comment text in which column it has high prob.

Hardware and Software Requirements and Tools Used

```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,roc_curve,roc_auc_score,auc
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

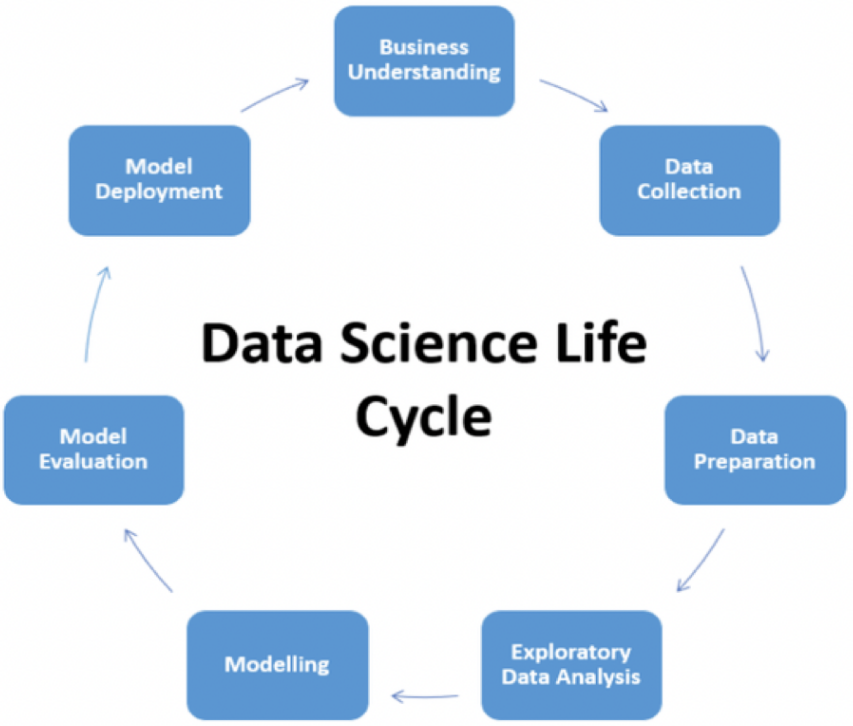
- Above are the libraries which I used to pre-process, predict and visualize the project :
- Pandas - It is used to play with data frame and helps in to get more insight of the data, like describing the data and the types of the all features.
- Seaborn, Matplotlib – Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots and so on. Seaborn on the other hand,

provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.

- Sklearn – Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction..

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)



  - Read the use case and search from google to know more about the use case and in which field or domain it will be applicable more.

  - As the data set is already there, this step is excluded.

- Data Preparation includes the data cleaning and describing the data which I followed text pre-processing.

- EDA involves visualization which is helpful to get more insight from the data and get to know about the most used comment and check the most frequent words.

- Modelling involves creating the model with a suitable algorithm which provides the best result. I tried multiple algo and applied hyperparameter tuning.

- Model Evaluation, for this I used confusion matrices and mainly focus on False negative and tried to reduce the False negative which is type 2 error and on F1 score as the dataset is imbalance.

- Testing of Identified Approaches (Algorithms)

Decision Tree:

```
DT = DecisionTreeClassifier()
DT.fit(x_train, y_train)
y_pred_train = DT.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = DT.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print("\n")
cvs=cross_val_score(DT, x, y, cv=10, scoring='accuracy').mean()
print('cross validation score :',cvs*100)
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988898736783678
Test accuracy is 0.9396933489304813


cross validation score : 94.09479258877076
[[41589  1361]
 [ 1526  3396]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.71      0.69      0.70      4922

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

Random Forest:

```
RF = RandomForestClassifier()

RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9988361578886114
Test accuracy is 0.9551721256684492
[[42417   533]
 [ 1613  3309]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.67      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.83      0.87     47872
weighted avg       0.95      0.96      0.95     47872
```

Logistic regression:

```
LG = LogisticRegression(C=1, max_iter = 3000)

LG.fit(x_train, y_train)

y_pred_train = LG.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = LG.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9595520103134316
Test accuracy is 0.9553183489304813
[[42729   221]
 [ 1918  3004]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.93      0.61      0.74      4922

    accuracy                           0.96     47872
   macro avg       0.94      0.80      0.86     47872
weighted avg       0.95      0.96      0.95     47872
```

Ada boost classifier:

```
ada=AdaBoostClassifier(n_estimators=100)
ada.fit(x_train, y_train)
y_pred_train = ada.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = ada.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.951118631321677
Test accuracy is 0.9490307486631016
[[42553   397]
 [ 2043  2879]]
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     42950
           1       0.88      0.58      0.70      4922

    accuracy                           0.95     47872
   macro avg       0.92      0.79      0.84     47872
weighted avg       0.95      0.95      0.94     47872
```

XG boost classifier:

```
xgb = xgboost.XGBClassifier()
xgb.fit(x_train, y_train)
y_pred_train = xgb.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = xgb.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9614052050600274
Test accuracy is 0.9526236631016043
[[42689   261]
 [ 2007  2915]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     42950
           1       0.92      0.59      0.72      4922

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

KNN Classifier:

```
knn=KNeighborsClassifier(n_neighbors=9)
knn.fit(x_train, y_train)
y_pred_train = knn.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = knn.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.922300110117369
Test accuracy is 0.9173629679144385
[[42809   141]
 [ 3815  1107]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     42950
           1       0.89      0.22      0.36      4922

    accuracy                           0.92     47872
   macro avg       0.90      0.61      0.66     47872
weighted avg       0.91      0.92      0.89     47872
```

- Run and Evaluate selected models

  - In classification problem there are various metrics that are accuracy score, confusion matrix, classification repot, Roc Auc curve which help to check the efficiency of the model

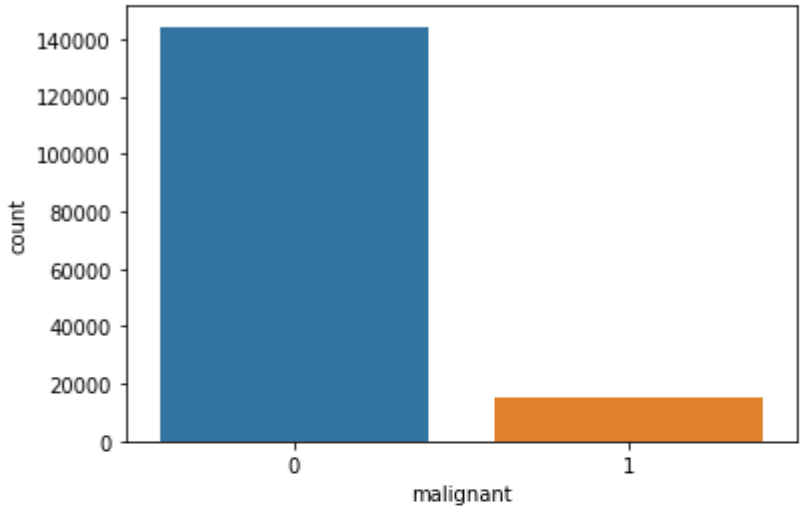
    Evaluation Matrices:
  - Accuracy - it determines how often a model predicts default and non default correctly.
  - Precision-it calculates whenever our models predicts it is default how often it is correct.
  - Recall- Recall regulate the actual default that the model is actually predict.
  - Precision Recall Curve - PRC will display the trade off between Precision and Recall threshold.
  - F1 score - the F1-score, is a measure of a model's accuracy on a dataset. It is used to evaluate binary classification systems, which classify examples into 'positive' or 'negative'.
  - Cross Validations:
  - K Fold cross validations , K = 10
  - So, in this case accuracy score is good but most important is confusion matrix in which we must decrease the False Positive that is type 2 error.

- Visualizations

  Mostly I used seaborn, word cloud and matplotlib for visualization and EDA

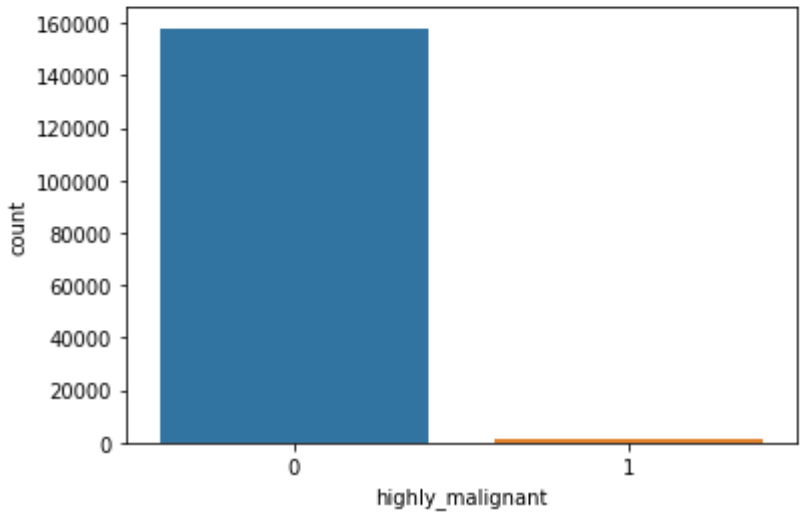
```
malignant


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



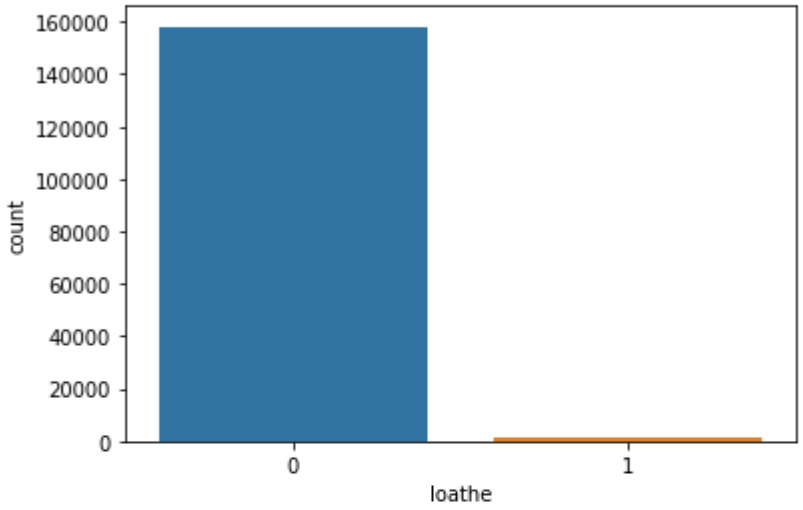

```
highly_malignant


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



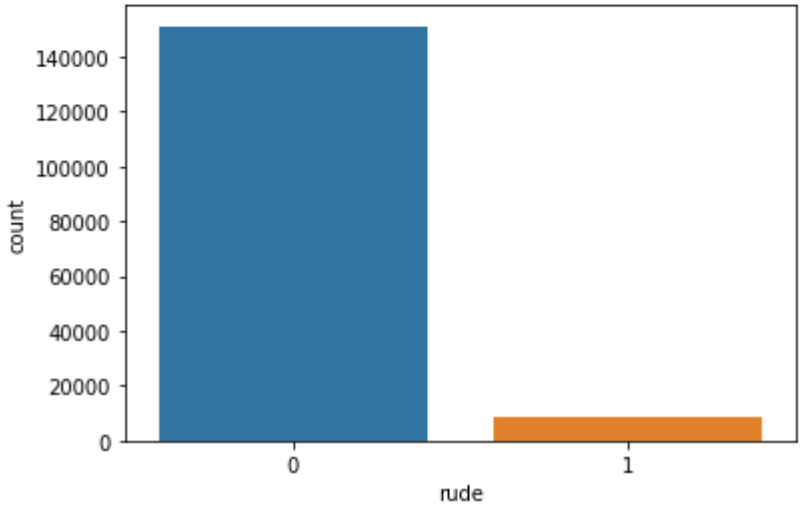
```
loathe


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



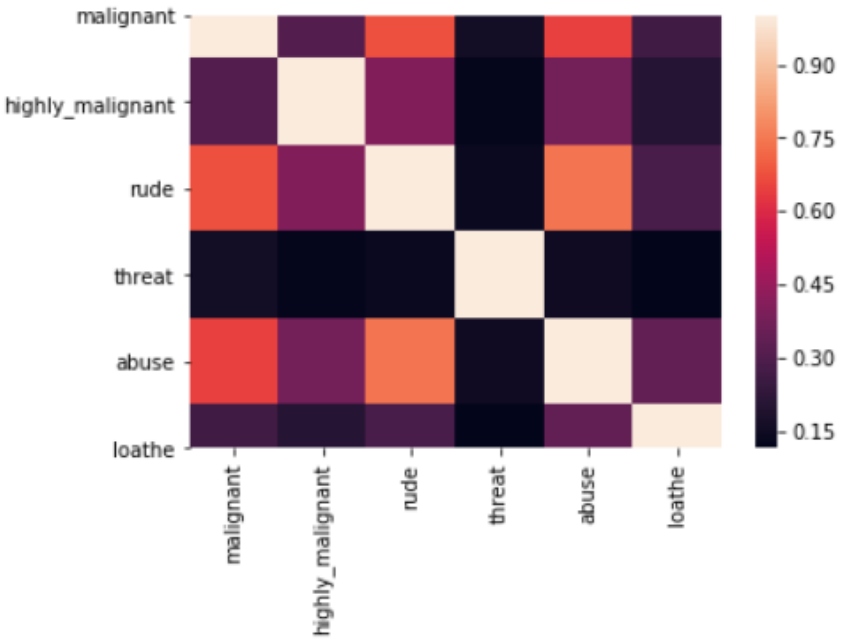

```
rude


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

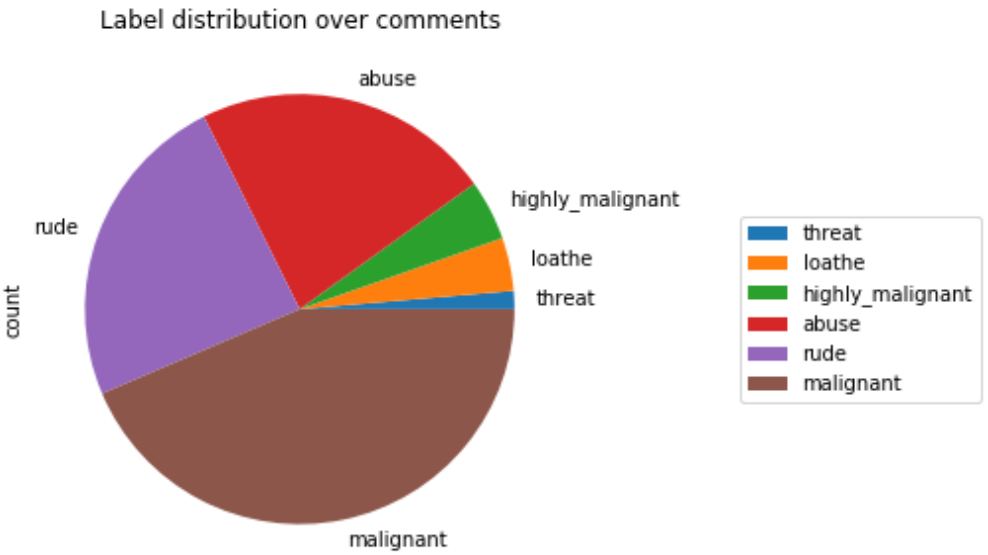


Wordcloud:

Correlation heatmap :



Abuse and malignant, rude and abuse are highly correlated.



CONCLUSION

- Key Findings and Conclusions of the Study

  - In this project there are some variables like malignant and rude which are highly correlated. It is possible because one comment text may have a combination of multiple features.
  - Removing the column id does not impact the model training.
  - Using Tree, model can reduce the false negative values
  - It has future scope in various use cases likewise in election, social media etc, where every day there are multi offensive comments spread.
  - So, in the future it may be used very well to easily classify the comments as bad or good.
  - Random forest is well suitable for this project as it uses trees internally and it uses multiple weak learners and generates the strong model and generates a low bias and low variance model.

- Learning Outcomes of the Study in respect of Data Science

    - While implementing this project most of the time is taken doing the evaluation of metrices.
    - Gain knowledge and get more insight into various stemmers and vectors.
    - Tree algorithms are well suitable for this project as it uses trees internally and it uses multiple weak learners and generates the strong model and generates a low bias and low variance model.

- Limitations of this work and Scope for Future Work

    - It has future scope in various use cases likewise in election, social media etc, where every day there are multi offensive comments spread.
    - So, in the future it may use very well to easily classify the comments as bad or good.