Malignant Comment Classification

Submitted by:

P. MANIVANNAN

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 unoffensive, 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.

- Review of Literature

The Research is done on online User comments how they can be used to Classify them. First Brief Exploratory Data Analysis is done like Handling Null values and data pre-processing is done using Regex. Then Data is then Fed to Machine Learning Algorithms. after Evaluating and Testing Different Algorithms Best Algorithm is Selected. Based on Evaluation Metrics such as Accuracy score, Cross Validation Score, Confusion Matrix

etc. then Hyper Parameter tuning is done on best model using Grid Search CV.

Analytical Problem Framing

- Data Sources and their formats

The data contains 159571 rows and 8 columns.

The data is in Csv format.

- Data Pre-processing Done
    1) The data has null values which is removed using dropna method of pandas.
    2) Since machine learning algorithms are case sensitive, we are converging all to lower cases
    3) Removing Emails and replacing them with word “emailaddress”.
    4) Removing Website address and replacing them with word “webaddress”.
    5) Removing Phone Numbers and replacing them with word “phonenumber”.
    6) Removing Currency Symbols like Dollars and Pounds and replacing with word “dollars”.
    7) Removing Numbers and replacing with word “numbr”.
    8) Removing Punctuations.
    9) Removing Whitespaces and stop words

- State the set of assumptions (if any) related to the problem under consideration
    1) Pre-processing is done using Regex

- Hardware and Software Requirements and Tools Used

The Hardware requirements are 16 Gb RAM,500 GB SSD, at least 6 Cores processor

The software requirements are Windows, Anaconda Framework and libraries used are Pandas, NumPy, Sklearn, SciPy, Matplotlib, Seaborn

We use Pandas for reading the Dataset, Creating Dataframe, much more to handle data in dataset

Certain functions of NumPy are used like log , cbrt ,sqrt skewness transformation . absolute (abs) function used along with Zscore.

SKlearn Library is used for Machine Learning Algorithms like SVC, Random Forest, Linear Regression etc and Metrics for analysing them.

Scipy is scientific python library used for Zscore method etc.

Matplotlib and Seaborn are used for visualizations plotting graphs charts, etc

Finally, joblib module is used for saving our model for future use.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Since the Target Variable is classification type I should Use Classification Approach to Solve this Problem and build a Classification Model

- Testing of Identified Approaches (Algorithms)
  1) Decision Tree Classifier
  2) K Neighbors Classifier
  3) Random Forest Classifier

4) Ada Boost Classifier
5) Logistic Regression

- Run and Evaluate selected models

  I have custom written a Code Block for Running all Algorithms and Storing them in DataFrame and Ranking them based on their Accuracy Score. The best part is the code finds the best random state for each algorithm and prints them

Code:

This code block is written by me which ranks the best alogrithm

If you see the code, I have created a list of algorithms which I need to run. And here is the output:

Output1:

```
# LogisticRegression
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 0.9595161997869274
Test accuracy is 0.9552974598930482
[[42733   217]
 [ 1923  2999]]
              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
```

```
# DecisionTreeClassifier
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(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9990420684160108
Test accuracy is 0.9400693516042781
[[41574  1376]
 [ 1493  3429]]
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     42950
           1       0.71      0.70      0.71      4922

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

```
# RandomForestClassifier
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)))
cvs=cross_val_score(RF, x, y, cv=5, 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 0.9990241631527588
Test accuracy is 0.9561956885026738
cross validation score : 95.68969278050851
[[42394   556]
 [ 1541  3381]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.86      0.69      0.76      4922

    accuracy                           0.96     47872
   macro avg       0.91      0.84      0.87     47872
weighted avg       0.95      0.96      0.95     47872
```

```
#AdaBoostClassifier
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 0.9507873839515125
Test accuracy is 0.9490307486631016
[[42557   393]
 [ 2047  2875]]
              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
```

Here you can see the Model Name, Accuracy Score and Cross Validation Score.

Based on this best model is selected and further proceeded for hyperparameter tuning. Here Logistic Regression performed best compared to others.

Here Logistic Regression performed best

Hyperparameter Tuning

Code:

```
parameters = {'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
 'penalty': ['none', 'l1', 'l2', 'elasticnet'],
 'C': [100, 10, 1.0, 0.1, 0.01]}
```

```
rf = LogisticRegression()
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=.20,random_state=56)
from sklearn.model_selection import GridSearchCV
gsv = GridSearchCV(rf,parameters)
gsv.fit(train_x,train_y)
```

```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [100, 10, 1.0, 0.1, 0.01],
                         'penalty': ['none', 'l1', 'l2', 'elasticnet'],
                         'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag',
                                    'saga']})
```

Output:

```
gsv.best_params_
```

```
{'C': 1.0, 'penalty': 'l1', 'solver': 'liblinear'}
```

The Best Parameters are listed now we need to run the model with the best parameters

```
# LogisticRegression
LG = LogisticRegression(C=1.0, penalty = 'l1',solver='liblinear')

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 0.9629808682262151
Test accuracy is 0.9592037098930482
[[42591   359]
 [ 1594  3328]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     42950
           1       0.90      0.68      0.77      4922

    accuracy                           0.96     47872
   macro avg       0.93      0.83      0.88     47872
weighted avg       0.96      0.96      0.96     47872
```

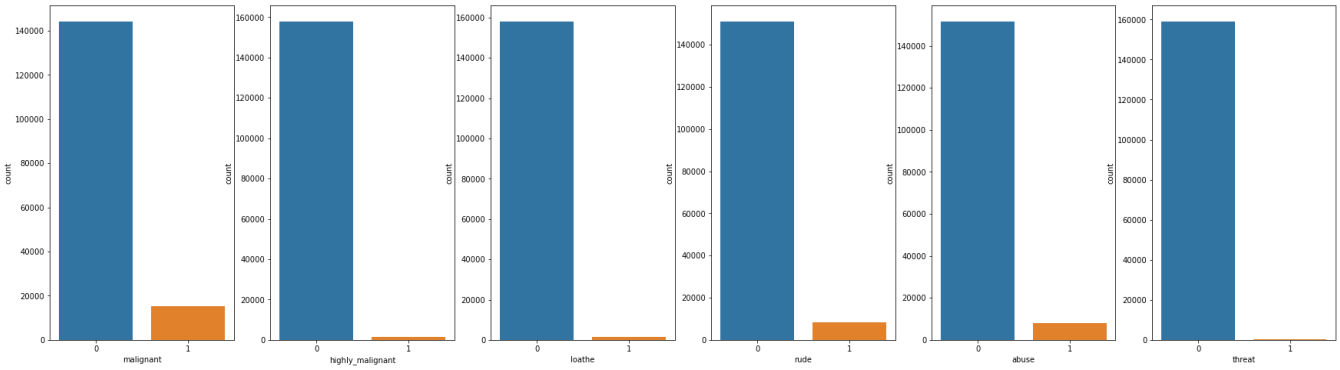
The Best Accuracy Score for Logistic Regression After Hyper parameter tuning for Training is 96% and Test Accuracy is 95%

- Key Metrics for success in solving problem under consideration

Since it's a Classification problem The key metrics used are Accuracy Score, Cross Validation score, confusion matrix, Classification Report

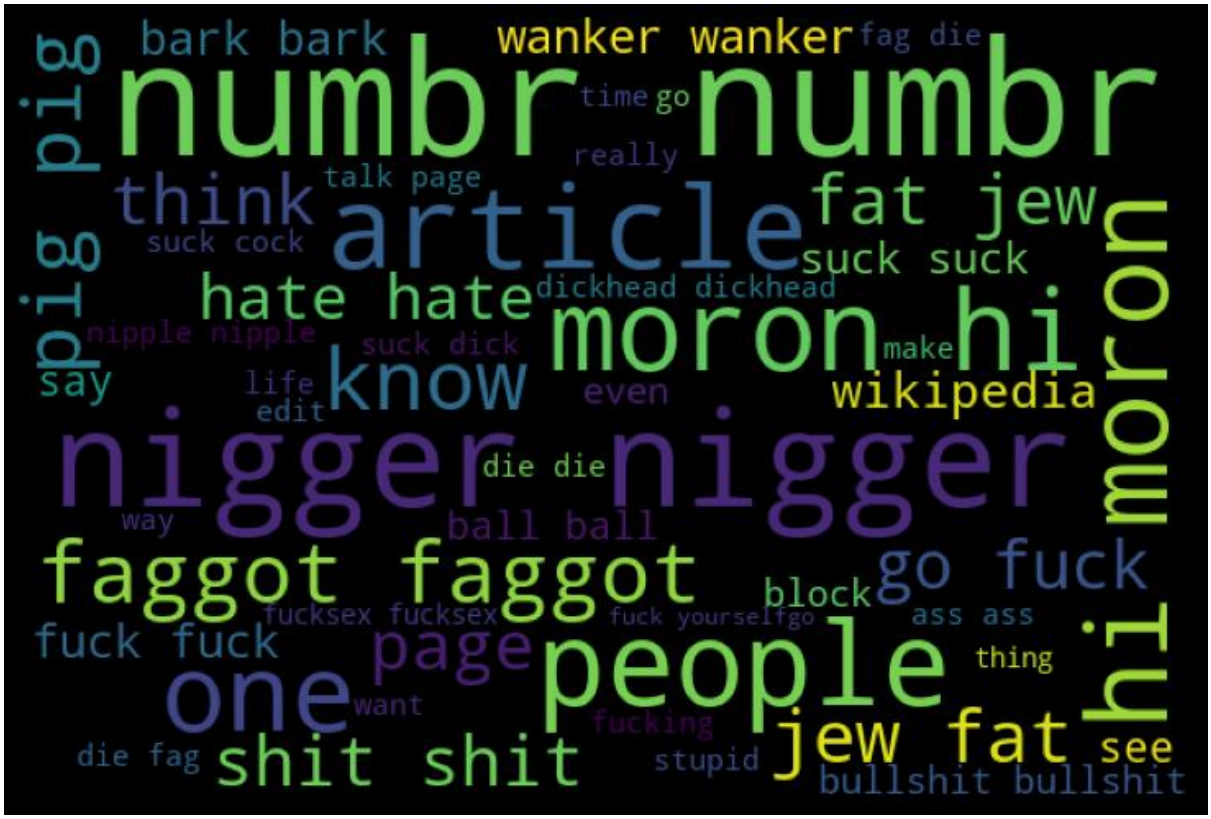
Visualizations

For visualizations the Matplotlib & Seaborn Library are used.



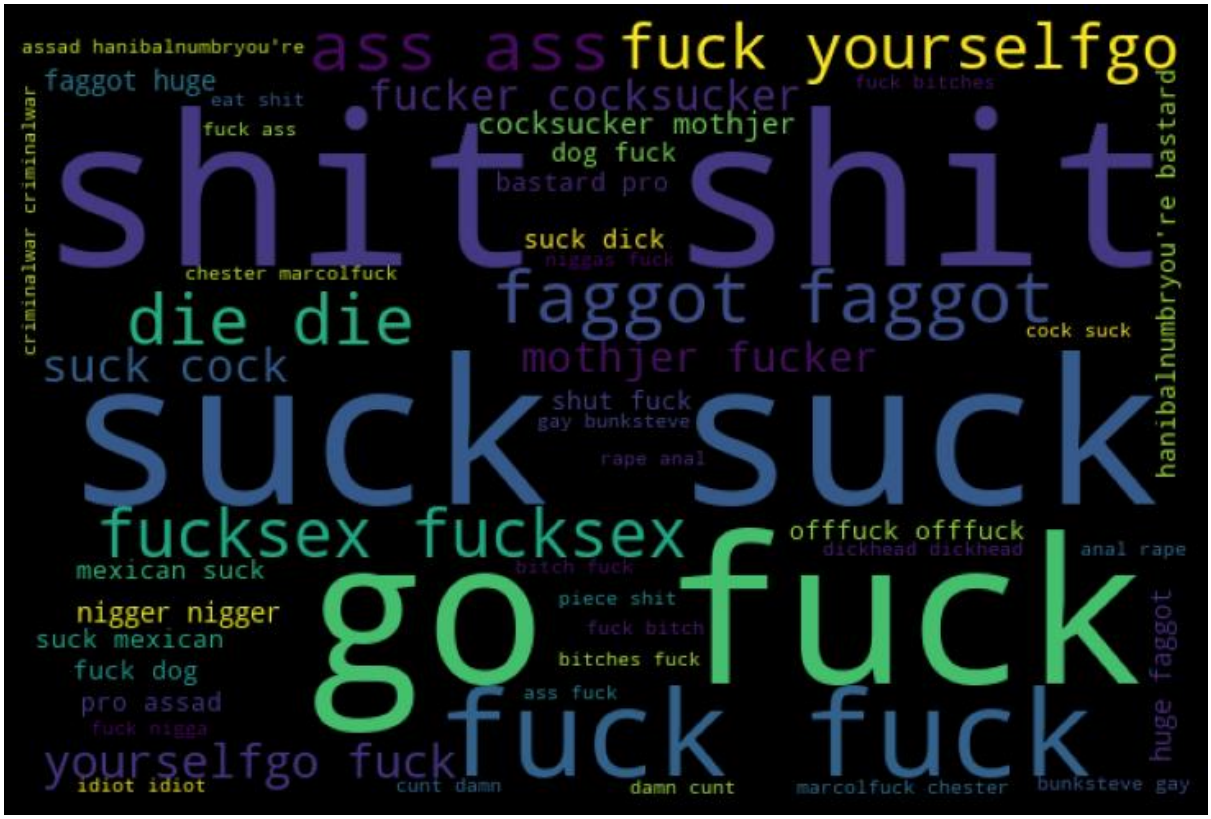
Observations:

Malignant Comments are higher compared to others



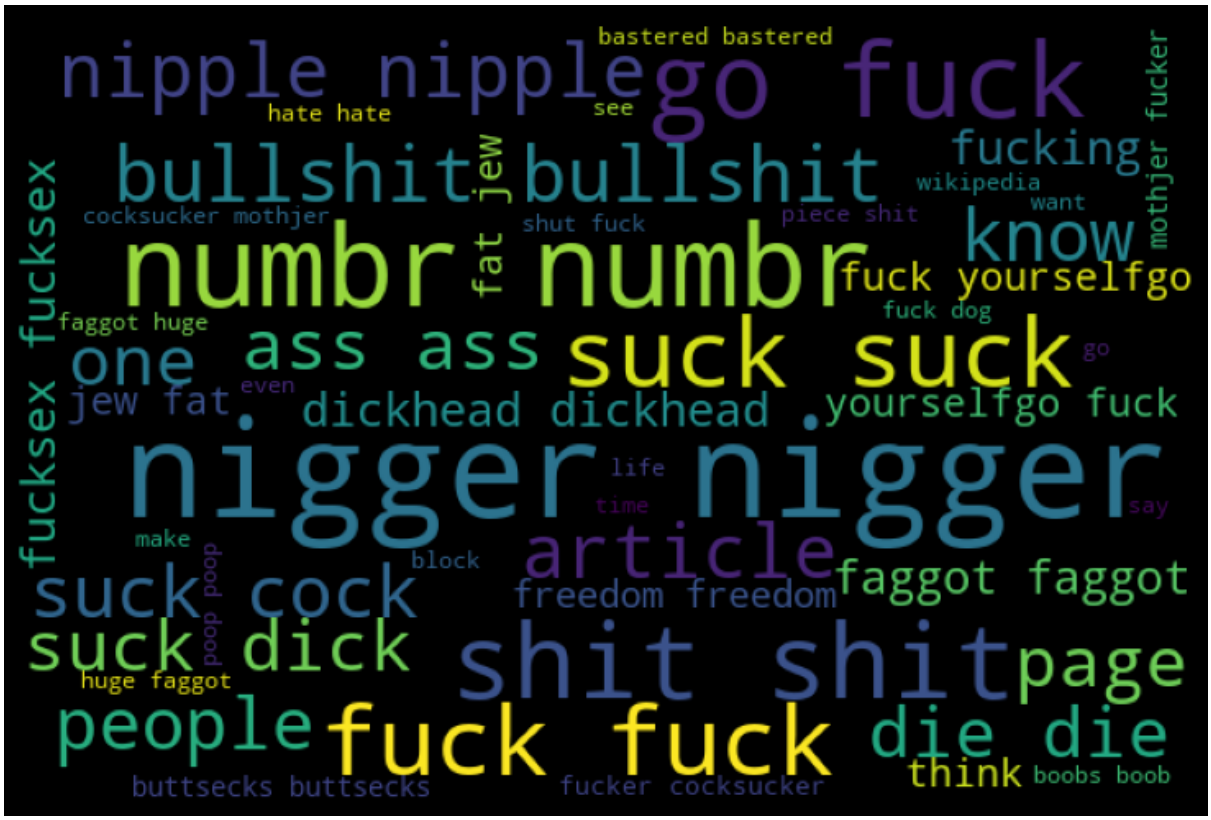
Observation:

Based on the Word Cloud plot these are words which is most often repeated in malignant comments.



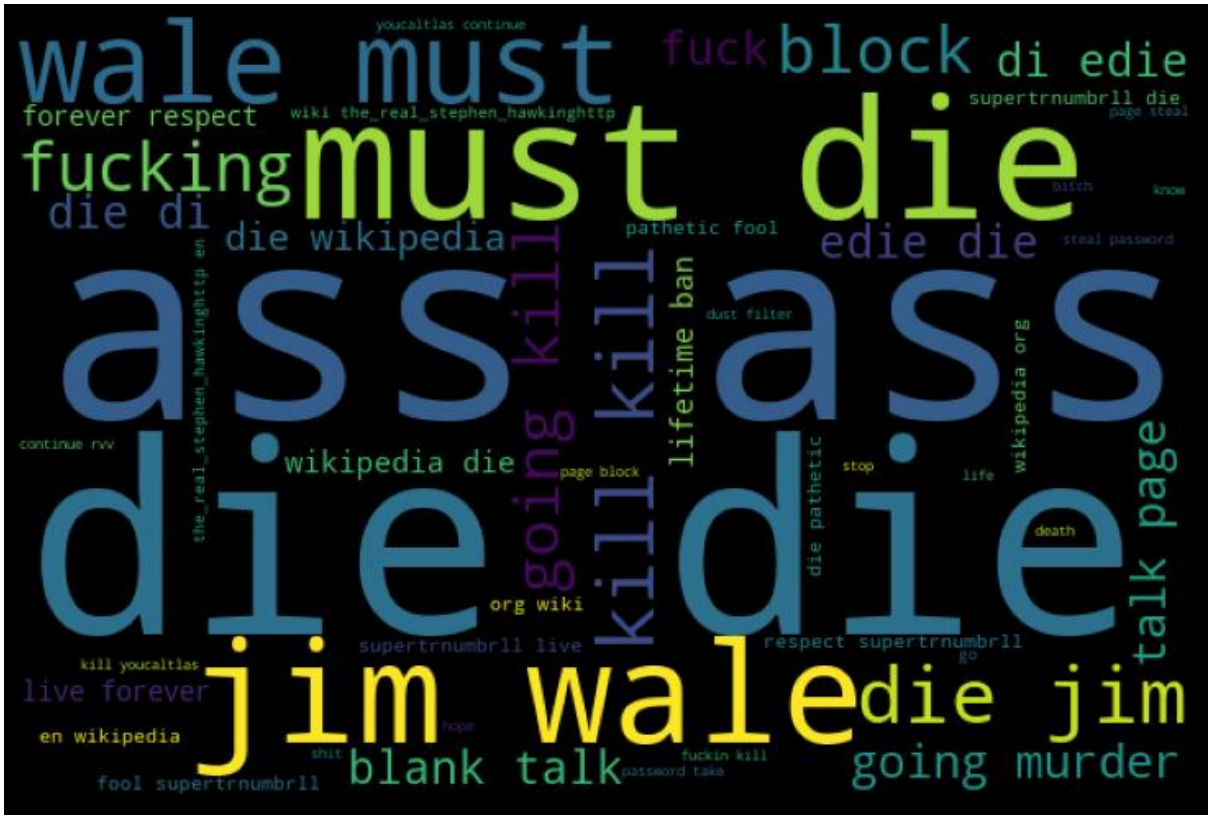
Observation:

Based on the Word Cloud plot these are words which is most often repeated in Highly Malignant comments.



Observation:

Based on the Word Cloud plot these are words which is most often repeated in Rude comments.



Observation:

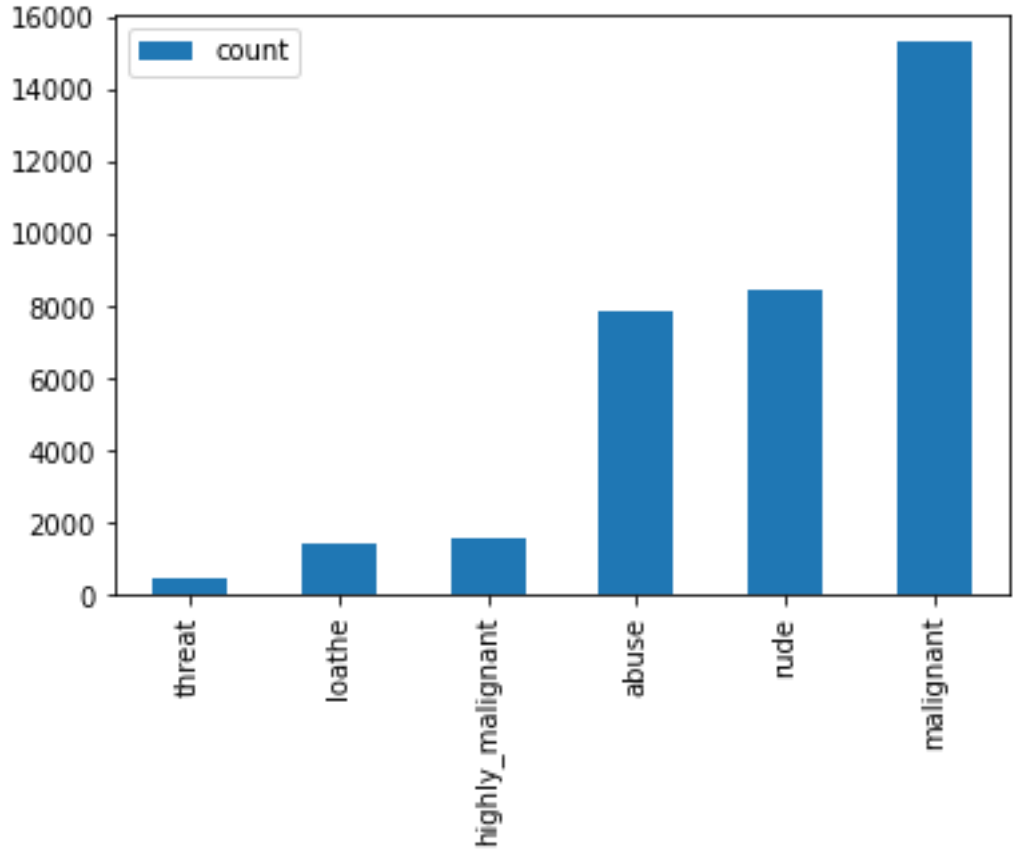
Based on the Word Cloud plot these are words which is most often repeated in threat comments.

Observation:

Based on the Word Cloud plot these are words which is most often repeated in abuse comments.

Observation:

Based on the Word Cloud plot these are words which is most often repeated in loathe comments.



Observation:

Type of Classification Comments plotted. We can malignant comments are higher.

- ## Interpretation of the Results

  Based on the Word Cloud plot, we can see some words influence the Comments Classification Prediction.

CONCLUSION

- Key Findings and Conclusions of the Study

  We can clearly see that Certain words affect the Comments Classification of the NLP Program.

- Learning Outcomes of the Study in respect of Data Science

  We can learn many useful insights from this study. the words people most often use to as threat rude etc. So based on these words NLP process the Comments classification.