FLIP ROBO

Detecting Malignant Comment Using Natural Language Processing

Submitted by:

Aditya Paruleker

INTRODUCTION

- Business Problem Framing

  Online hate, described as abusive language, aggression, cyber bullying, 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. This comments can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts. Data science can thus play a key role in this domain by protecting the online users from such hatred using various natural language processing techniques. The primary objective is to use this data science knowledge to build a model to detect the fake news. 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

  Motivation behind this project is to help the users avoid facing backlashes and hateful offensive comments from other people. The information obtained from the model can be used to block such comment and make social media platform a more safer and better place.

Analytical Problem Framing

- Data Sources and their formats

  There are two dataset; one for training, which contains 159571 rows and 8 column. The second dataset for testing contains 153164 rows and 2 columns. Both the dataset are obtained from the kaggle website.

- Preprocessing Done

a) Null Value Analysis

```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

No null values are detected in the dataset. Since all the data is of object type we skip the statistical analysis and move to data cleaning phase.

b) Data Cleaning

- Combine values of all four feature (Malignant, Highly Malignant, Abuse, Loathe, Threat) and then create a new feature called “Malignant Comment” and store 1 if the sum of all the four feature is greater than 1 or else store 0.
- Strip if any html tags using beautiful soup html parser
- Remove all the .com and https links.
- Split the text and convert it to lower case.
- Remove all the numbers from the string.

- Using regex function, strip anything other than words and whitespaces like symbols.
- Eliminate the stopwords and punctuation from the string and then lemmatize the text.
- Remove any single character if remaining in the text.
- Delete rows with empty string.

c) **Separating Target and Input features**
Before proceeding with further preprocessing, the input features labelled x are separated from target features labelled y.

d) **Vectorize**
In NLP, word vectorization is a method i.e. used to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics. And this process of converting words into numbers is called Vectorization.

e) **TF – IDF (Term frequency-inverse document frequency)**
It evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document and the inverse document frequency of the word across a set of documents.

• Hardware and Software Requirements and Tools Used

Following libraries were used in this project,

Pandas
Created by Wes McKinney in 2008, this python library is widely used for data manipulation, analyzing and cleaning of data. Apart from this, it also helps in finding correlation between columns and in deleting rows.

Numpy
Created by Travis Oliphant in 2005, this python library provides an array object called ndarray i.e. up to 50x faster than traditional python lists. It was has functions can be used in linear algebra, matrices etc

Seaborn
Seaborn is a high level interface based data visualization library that uses matplotlib library underneath the working.

Matplotlib
Unlike saeborn, matplotlib is a low level data visualization python library. Majority of its function lies in the submodule named pyplot

Scikit-Learn
It provides tools for classification, regression, clustering and dimensionality reduction through its interface in python.

Pickle
Used for serializing (pickling) and de-serializing (unpickling) python object structure so that the data can be easily transferred from system to system and then stored in a file.

NLTK
NLTK is a standard python library that provides a set of diverse algorithms for NLP.

BeautifulSoup
Beautiful Soup is a Python library for pulling data out of HTML and XML files.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

  In this study several machine learning models were tested and their results were compared before selecting the model that gave best result among all.

  Logistic Regression
  Using logistic approach for modelling, this supervised machine learning model aims to solve classification problems by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome.

  KNeighbor Classifier
  The K-NN algorithm used in this classification stores all the available data and classifies a new data point based on the similarity. It is non parametric since it does not make any underlying assumption.

  Support Vector Classifier
  The SVC approach finds a hyperplane that creates a boundary between two classes of data to classify them.

  Multinomial Naive Bayes Classifier
  The algorithm first guesses the tag of a text using the Bayes theorem and then calculates each tag's likelihood for a given sample and lastly outputs the tag with the greatest chance.

  Ensemble Methods
  Ensemble technique uses several base estimators of machine learning model to predict the output. This results are then combined to improve the robustness of model over single estimator.

GradientBoost Classifier

The model after calculating the residual i.e. the difference between actual value and predicted target value; trains the weak model for mapping the features to that residual.

- Run and Evaluate models

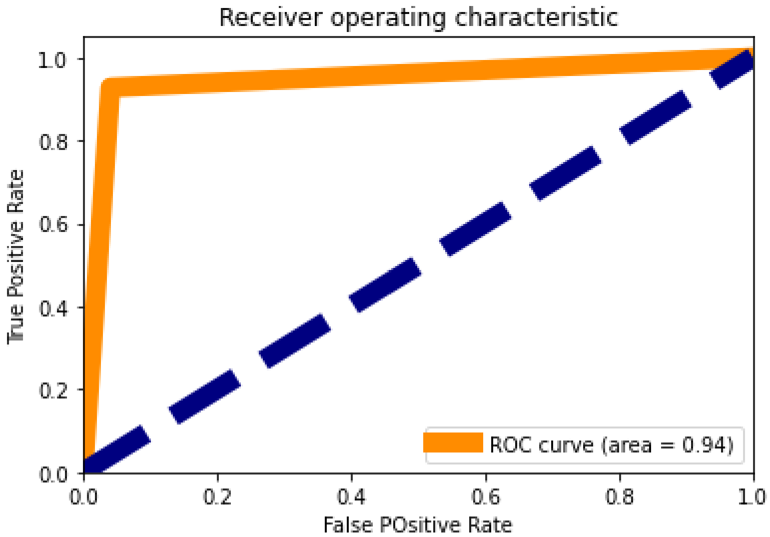
Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score,accuracy_score,classificatio
n_report,auc,roc_curve
from sklearn.model_selection import train_test_split,cross_val_sc
ore
from sklearn.metrics import log_loss


lg = LogisticRegression()

x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,
random_state=42)
lg.fit(X,y)
pred_train=lg.predict(x_train)
pred_test=lg.predict(x_test)
print("Train Accuracy : ",round(accuracy_score(y_train,pred_train
)*100,2))
print("Test Accuracy : ",round(accuracy_score(y_test,pred_test)*1
00,2))
print("cv score : ", round(cv_score.mean()*100,2))
logloss = log_loss(y_test, lg.predict_proba(x_test))
logloss
print("Classification Report:\n",classification_report(y_test,pre
d_test))
```

Output

```
Train Accuracy :  96.04
Test Accuracy :  95.69
cv score :  95.61
0.11876382098273715
```

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

           0       0.96      0.99      0.98     28692
           1       0.93      0.62      0.74      3223

    accuracy                           0.96     31915
   macro avg       0.94      0.81      0.86     31915
weighted avg       0.96      0.96      0.95     31915

 Confusion Matrix
 [[28535   157]
 [ 1218  2005]]
```



Model Selection User Defined Function

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve,auc

def model_selection(algorithm_instance,x_train,y_train,x_test,y_test):
algorithm_instance.fit(x_train,y_train)
model_pred_train=algorithm_instance.predict(x_train)
 model_pred_test=algorithm_instance.predict(x_test)
 print("Accuracy of training model :",round(accuracy_score(y_train,model_pred_train)*100,2))
 print("Accuracy of test data :",round(accuracy_score(y_test,model_pred_test)*100,2))

 cv_score=cross_val_score(algorithm_instance,X,y,cv=5)
 print("cv score : ", round(cv_score.mean()*100,2))
print("\nClassification report for test data\n",classification_report(y_test,model_pred_test))
print("Classification report for training data\n",classification_report(y_train,model_pred_train))
```

```
    print("Confusion Matrix\n",confusion_matrix(y_test,model_pred_test))
    print("\n")

    fpr, tpr, thresholds = roc_curve(model_pred_test,y_test)
    roc_auc= auc(fpr,tpr)
    plt.figure()
    plt.plot(fpr,tpr,color='darkorange',lw=10,label='ROC curve (area =
%0.2f)' % roc_auc)
    plt.plot([0,1],[0,1],color='navy',lw=10,linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0,1.05])
    plt.xlabel("False POsitive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Receiver operating characteristic")
    plt.legend(loc='lower right')
    plt.show()
```

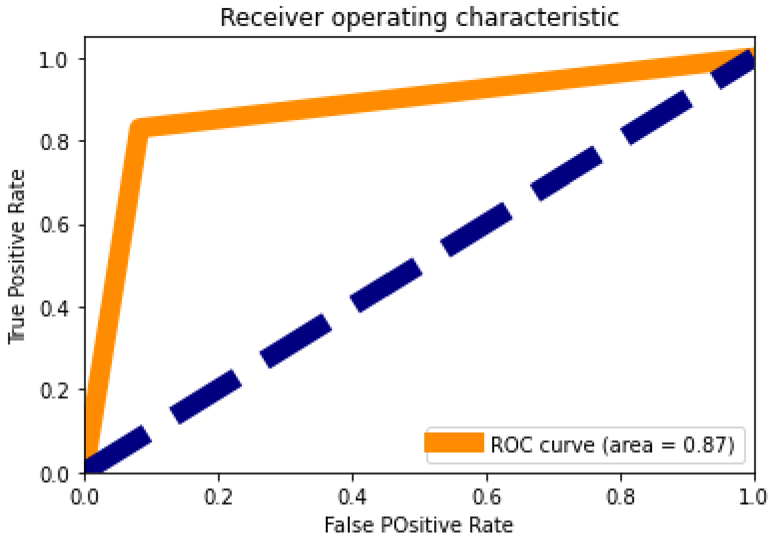
K Neighbour Classifier

```
from sklearn.neighbors import KNeighborsClassifier
k=KNeighborsClassifier()
model_selection(k,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 92.32
Accuracy of test data : 91.51
cv score :  91.51
Log loss:  1.7830867845874572

Classification report for test data
              precision    recall  f1-score   support

           0       0.92      1.00      0.95     28692
           1       0.83      0.20      0.32      3223

    accuracy                           0.92     31915
   macro avg       0.87      0.60      0.64     31915
weighted avg       0.91      0.92      0.89     31915

Classification report for training data
              precision    recall  f1-score   support

           0       0.92      1.00      0.96    114654
           1       0.92      0.27      0.42     13002

    accuracy                           0.92    127656
   macro avg       0.92      0.63      0.69    127656
weighted avg       0.92      0.92      0.90    127656
```

```
Confusion Matrix
 [[28559   133]
 [ 2578   645]]
```



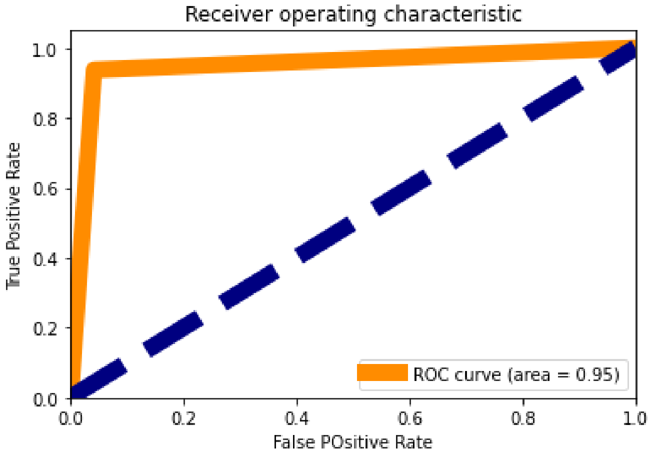
Support Vector Classifier

```
from sklearn import svm
s=svm.SVC()
model_selection(s,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 98.86
Accuracy of test data : 95.65
cv score :  95.6

Classification report for test data
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     28692
           1       0.94      0.61      0.74      3223

    accuracy                           0.96     31915
   macro avg       0.95      0.80      0.86     31915
weighted avg       0.96      0.96      0.95     31915

Classification report for training data
              precision    recall  f1-score   support

           0       0.99      1.00      0.99    114654
           1       0.99      0.89      0.94     13002

    accuracy                           0.99    127656
   macro avg       0.99      0.95      0.97    127656
weighted avg       0.99      0.99      0.99    127656
```

```
Confusion Matrix
 [[28560   132]
 [ 1256  1967]]
```



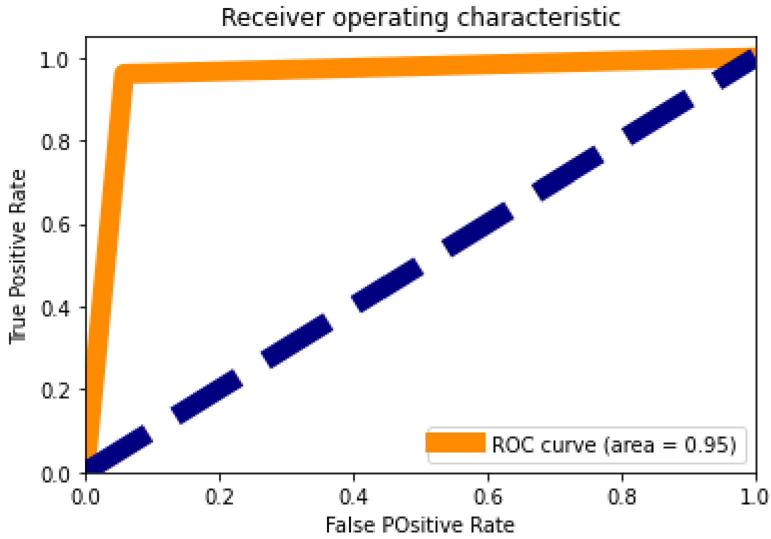
Multinomial Naïve Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
model_selection(mnb,x_train,y_train,x_test,y_test)
```

Output

```
Accuracy of training model : 94.56
Accuracy of test data : 94.26
cv score :  94.17
Log loss:  0.15682396693381728

Classification report for test data
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     28692
           1       0.96      0.45      0.61      3223

    accuracy                           0.94     31915
   macro avg       0.95      0.72      0.79     31915
weighted avg       0.94      0.94      0.93     31915

Classification report for training data
              precision    recall  f1-score   support

           0       0.94      1.00      0.97    114654
           1       0.97      0.48      0.64     13002

    accuracy                           0.95    127656
   macro avg       0.96      0.74      0.81    127656
weighted avg       0.95      0.95      0.94    127656
```

```
Confusion Matrix
 [[28632    60]
 [ 1771  1452]]
```



GradientBoost Regressor

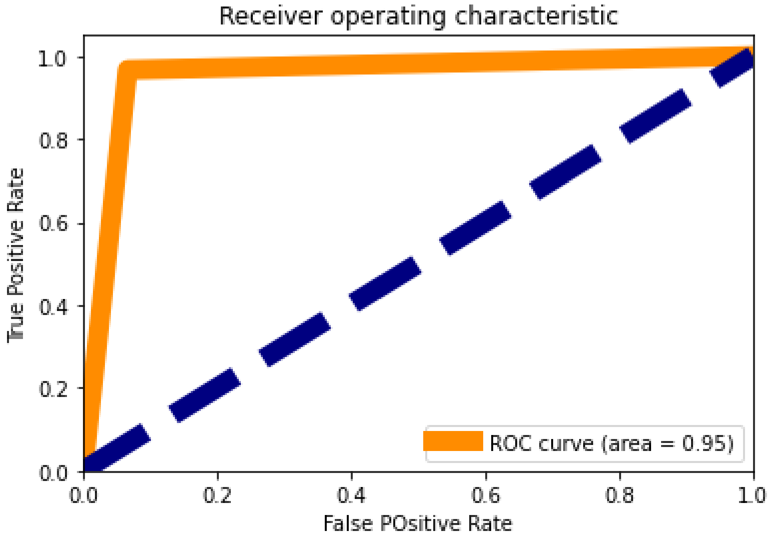
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

parameter={'loss':['log_loss', 'deviance', 'exponential'],
      'criterion':['friedman_mse', 'squared_error', 'mse'],
      'max_features':['auto', 'sqrt', 'log2'],
      'n_estimators':range(0,100,50)}

rf3=GradientBoostingClassifier()
clf=GridSearchCV(rf3,parameter)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'criterion': 'friedman_mse', 'loss': 'deviance', 'max_features': 'auto', 'n_estimators': 50}
```

```
rf3=GradientBoostingClassifier(criterion='friedman_mse', loss='deviance
', max_features= 'auto', n_estimators= 50)
model_selection(rf3,x_train,y_train,x_test,y_test)
```

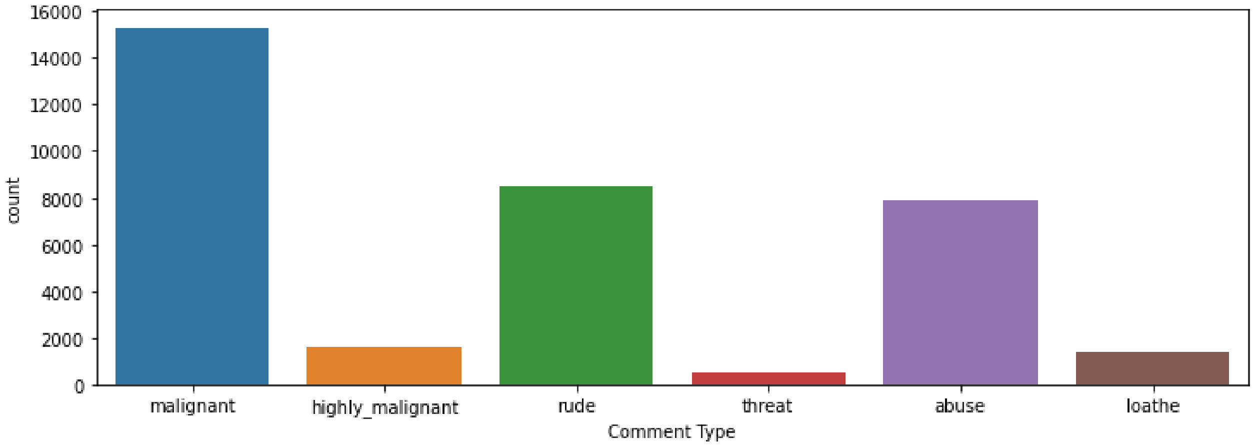
Output

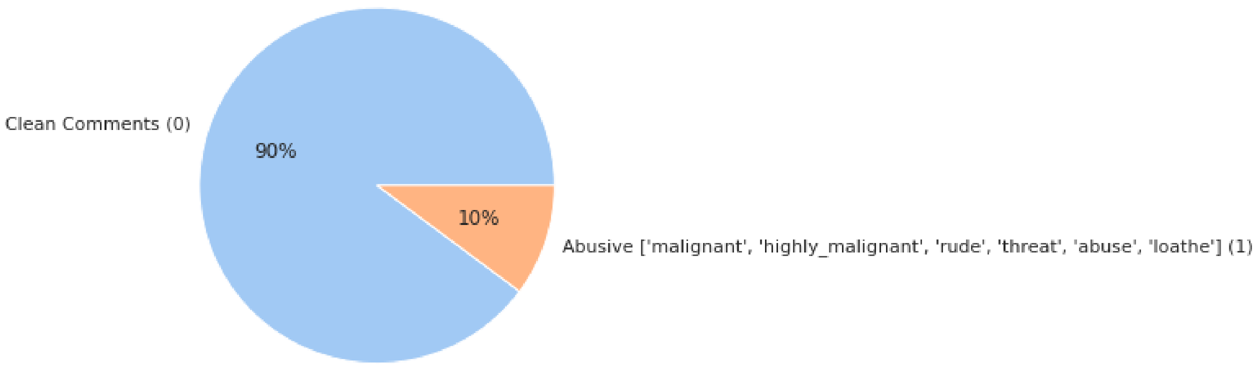
```
Accuracy of training model : 93.44
Accuracy of test data : 93.46
cv score :  93.39

Classification report for test data
              precision    recall  f1-score   support

           0       0.93      1.00      0.96     28692
           1       0.97      0.36      0.53      3223

    accuracy                           0.93     31915
   macro avg       0.95      0.68      0.75     31915
weighted avg       0.94      0.93      0.92     31915

Classification report for training data
              precision    recall  f1-score   support

           0       0.93      1.00      0.96    114654
           1       0.97      0.37      0.53     13002

    accuracy                           0.93    127656
   macro avg       0.95      0.68      0.75    127656
weighted avg       0.94      0.93      0.92    127656

Confusion Matrix
 [[28651    41]
 [ 2047  1176]]
```



- Dataset Visualizations

	text	count
0	malignant	15294
1	highly_malignant	1595
2	rude	8449
3	threat	478
4	abuse	7877
5	loathe	1405



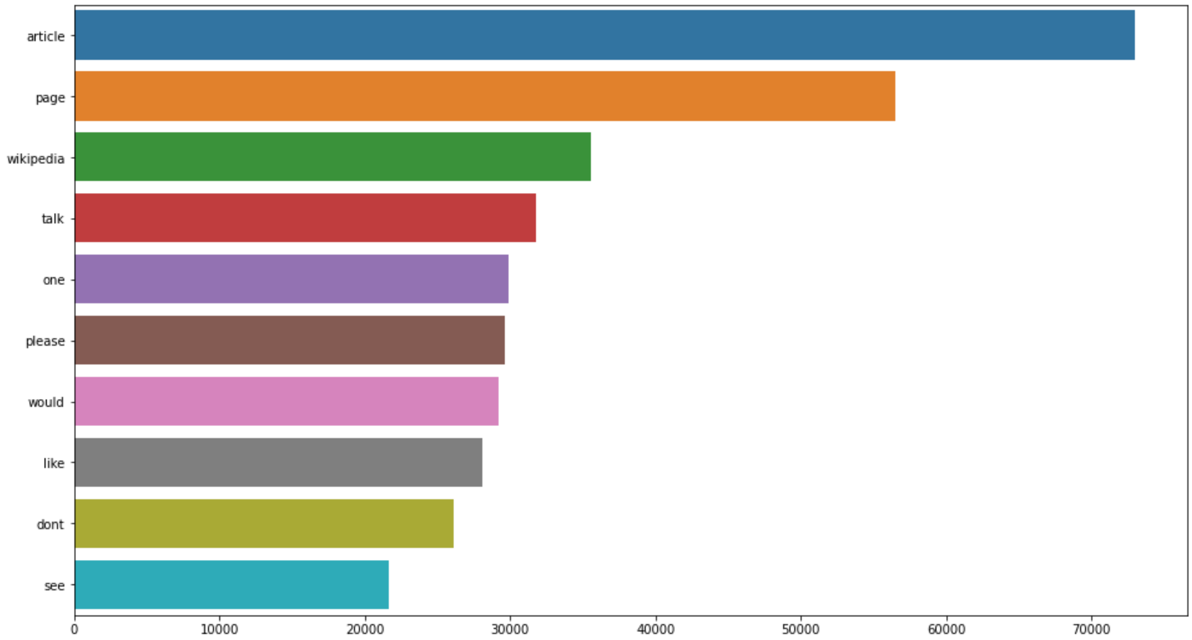


- Post Data cleaning Visualizations

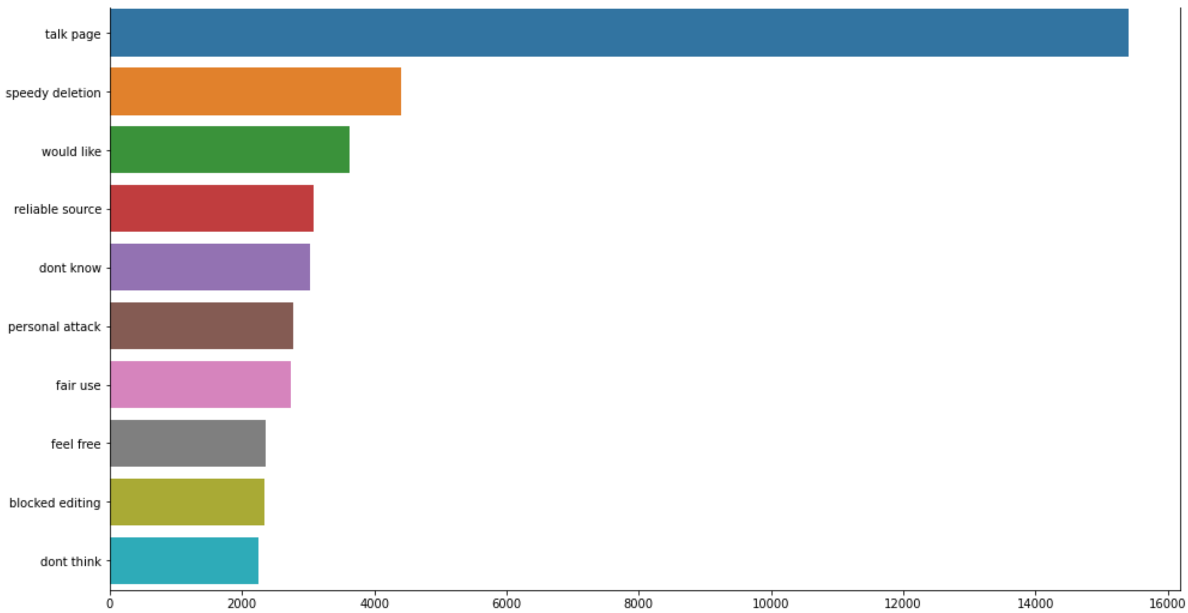
a) Word Cloud (Malignant Comments)

b) Word Cloud (Non Malignant Comments)

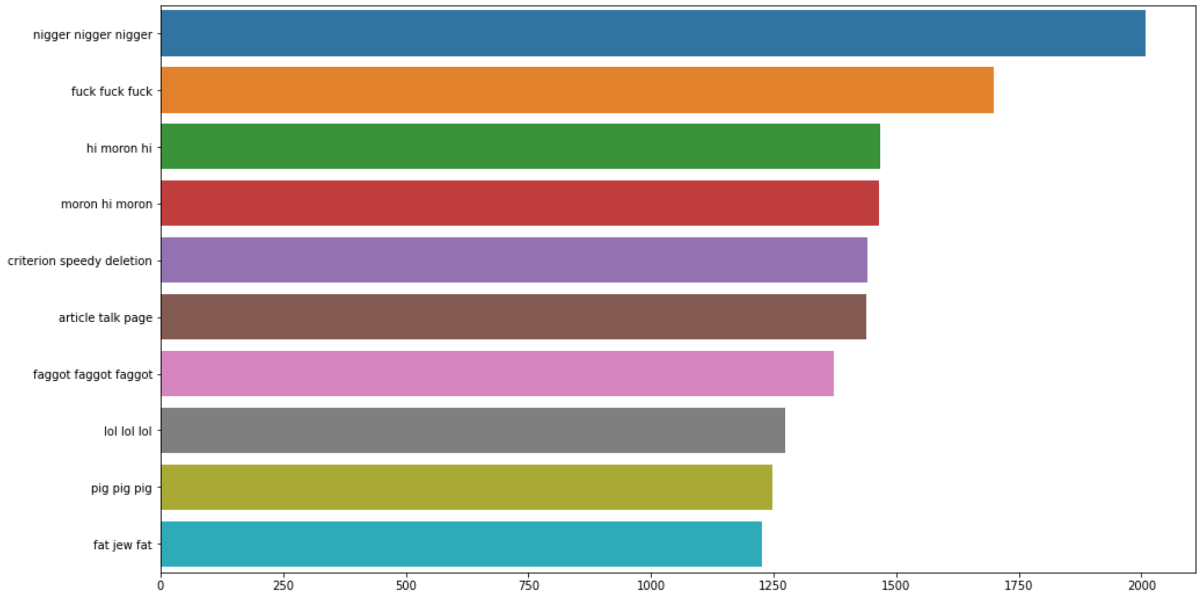
c) Unigram Analysis



d) Bigram Analysis



e) Trigram Analysis



- 10% of the comment in the dataset are malignant in nature whereas the remaining 90% are non malignant in nature.
- Out of 6 categories of the social media comment types, malignant comments had highest count of 15294 comments.
- In unigram analysis, word 'article' is found to have highest number of repetition.
- In bigram analysis, 'talk page' is found to have highest number of occurrence.
- In Trigram analysis, 'niger niger niger' is found to have highest number of occurrence.

CONCLUSION

- Key Findings and Conclusions of the Study

  - In this study we found that logistic regression classifier performs slightly better than rest of the algorithm tested.
  - Article is the most repeated word in entire dataset.

  Thus we were successfully able to detect the malignant comment with minimum error.