MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

C S Manjunath Reddy

ACKNOWLEDGMENT

I would like to thank FlipRobo for giving me this opportunity. The DataTrained institute classes helped me to solve this problem.

The language used for the project, the Python with Pandas, NumPy, Matplotlib, Seaborn, Sklearn, NLP.

INTRODUCTION

- Business Problem Framing

  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.Our goal is to build a machine learning model with the help of past data and to predict the maligant values for new comments.

- Conceptual Background of the Domain Problem

  The domain-related concepts are Data science, NLP, Machine Learning.

- Motivation for the Problem Undertaken

  Client wants to differentiate the comments which are toxic and non toxic. We have train data and test data. With train data we will train with ML algorithms and later use the best model to predict the test data.

Analytical Problem Framing

- Data Sources and their formats

  We got the train and test data from the client.

- Data Preprocessing Done

  The comments includes all special characters, we used lemmatization, removed all punctuation and stopwords, Replaced email address with word email, replaced phone numbers with number.

- Data Inputs- Logic- Output Relationships

  The comments columns in train data showed which are toxic and which are not toxic. With the help of this train data we trained the model and used test data to predict the values.

- Hardware and Software Requirements and Tools Used

- Pandas : used to read the data and to check data insights
- NLP-Regular Expression : used to clean the text data.
- NLP-Tf-idf : To change the text data into numbers.
- Scikit-Learn : Used Ml algorithms to train the data and evaluate best model.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Treated the text data by cleaning the data with stop words, email address, numbers and removing punctuations, did lematization.

- Testing of Identified Approaches (Algorithms)

  Navie_bayes.
  Logistic Regression.
  DecisionTree Classifier

- Run and Evaluate selected models
  Logistic Regression:

```
lr.fit(x_train_smote,y_train_smote)
pred=lr.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.9096340240641712
[[39373  3539]
 [  787  4173]]
              precision    recall  f1-score   support

           0       0.98      0.92      0.95     42912
           1       0.54      0.84      0.66      4960

    accuracy                           0.91     47872
   macro avg       0.76      0.88      0.80     47872
weighted avg       0.93      0.91      0.92     47872
```

DecisionTreeClassifier:

```
dr.fit(x_train_smote,y_train_smote)
pred=dr.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.8989388368983957
[[39476  3436]
 [ 1402  3558]]
              precision    recall  f1-score   support

           0       0.97      0.92      0.94     42912
           1       0.51      0.72      0.60      4960

    accuracy                           0.90     47872
   macro avg       0.74      0.82      0.77     47872
weighted avg       0.92      0.90      0.91     47872
```

Naïve Bayes:

```
nb.fit(x_train_smote,y_train_smote)
pred=nb.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.8983121657754011
[[38764  4148]
 [  720  4240]]
              precision    recall  f1-score   support

           0       0.98      0.90      0.94     42912
           1       0.51      0.85      0.64      4960

    accuracy                           0.90     47872
   macro avg       0.74      0.88      0.79     47872
weighted avg       0.93      0.90      0.91     47872
```

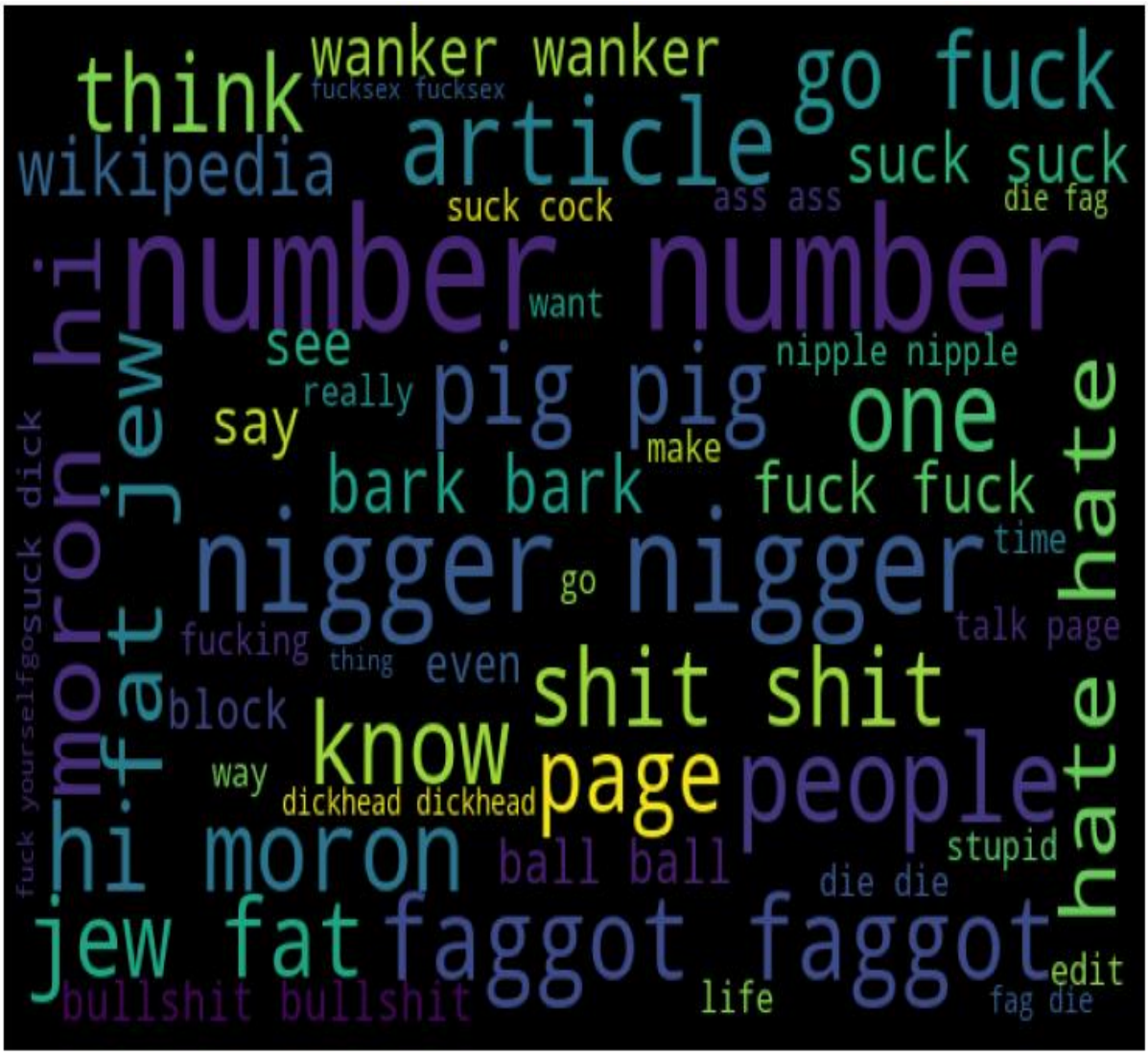
- Key Metrics for success in solving problem under consideration

Key Metrics are Confusion Matrix and Classification report . Confusion matrix shows True positive and false positive rate. Classification report shows F1-score(which shows how good the model trained).

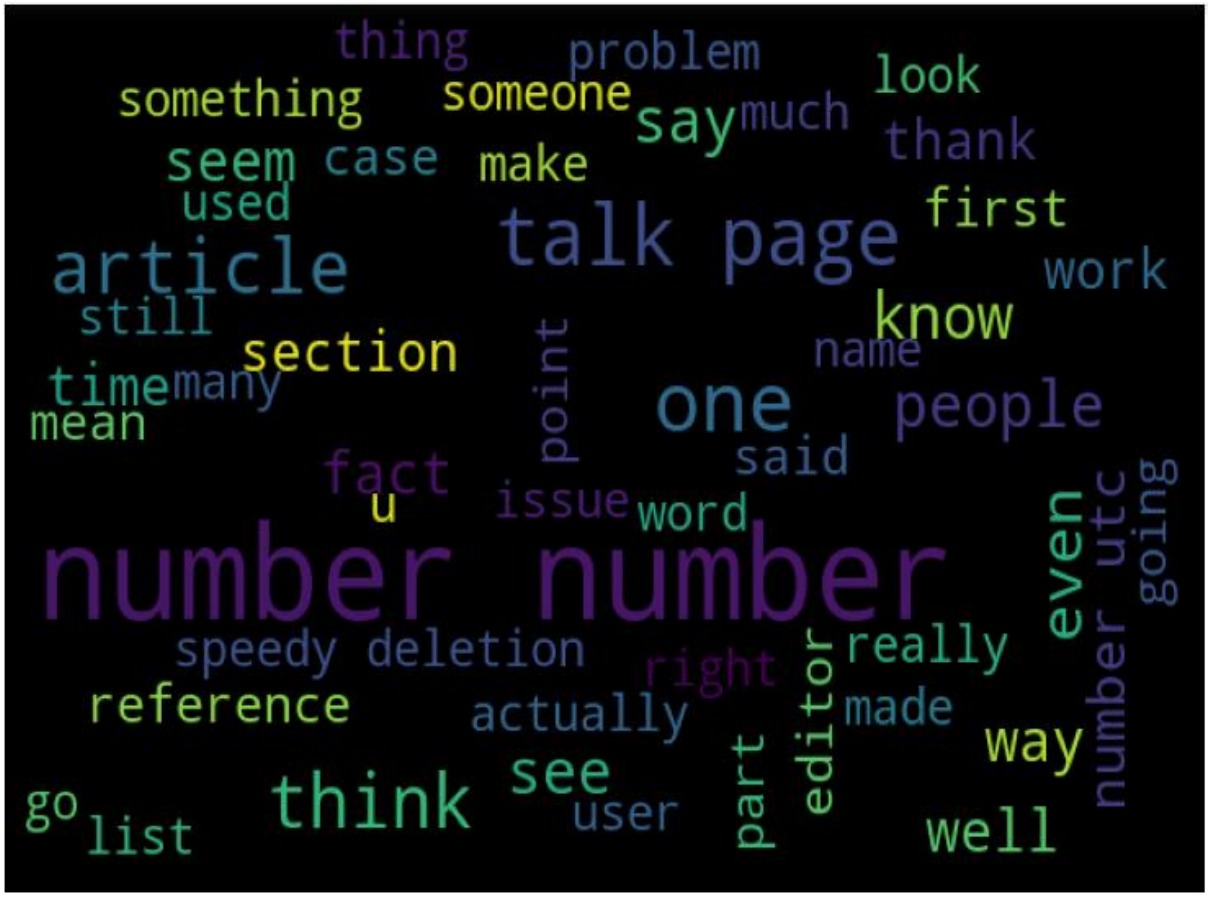
- Visualizations

  With the help of Word clouds or tag clouds are graphical representations of word frequency that give greater prominence to words that appear more frequently in a source text.

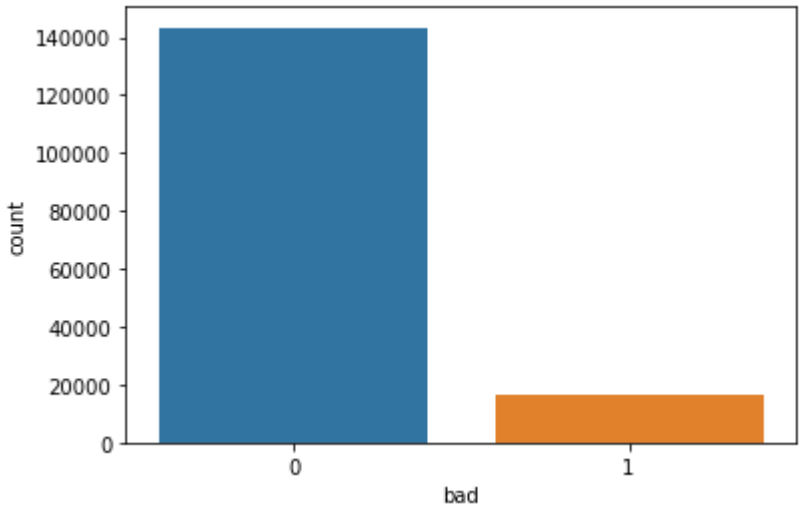
  → From below visualization we can see that the words which shows the maligant is yes.



→ From below visualization we can see that the words which shows the malignant is NO.



→The below count plot shows how many comments are toxic(malignant/bad) and how many are not.



- Interpretation of the Results

  From the Visualizations we came to know that the data is imbalanced. So, we have to do Oversampling or Under Sampling. If we do under sampling we will miss some data. So in this project we will use Oversampling.

CONCLUSION

- Key Findings and Conclusions of the Study

  Key Findings – Find how to clean the text data.

  Trained the data with multiple algorithms and checked SVC (support vector classifier) is the better algorithm with better f1 score and accuracy.

- Learning Outcomes of the Study in respect of Data Science

I used DecsiosnTree Classifier, Navie-byes, Logistic Regression, SVC algorithms to train the data. I used NLP techniques (Regex, bag of words) to clean and change the text data into numerical format. When I trained the data with the algorithms it shows Logistic Regression working better with better F1 score when compared with other algorithms.