Malignant Comments Classifier Project

Submitted by:

YOUR NAME

K. Rahul Ramanujam

ACKNOWLEDGMENT

- There aren't any specific references or going through the research papers to understand the project since the enough information and briefing was given in the problem statement.

- The professional help in a way was taken from my reporting manager 'Shubham Yadav' on tackling the problem (be it any) by breaking it down and finding the solution to it. This method has become quite handy.

- Other resource I had to look up was to see if I could find any relevant subject online but most of those are just the result pages of cyber bullying and other related articles. I had gone through those a while to see how the train and test data are relevant with the recent abuse online cases.

- The final search I did was to check the social media sites and see how much the online abuse runs on a daily basis. The fact that shook me was, more famous the personality is, more is the abuse in comments of their respective posts. The final model of this project is capable to depict the comments which are abusive and which are not.

INTRODUCTION

- ## Business Problem Framing

  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.

  Thus, we've to build a model which will detect such abusive comments and help the system to block/minimize it.

- ## Conceptual Background of the Domain Problem

  The projects like Rating prediction, Sentiment analysis and projects like those will help in reading the text input and drawing the sentiments out of those for a better desired prediction.

  The current project had to be dealt with the text input to finally send it to the model building post text-processing for final prediction.

- ## Motivation for the Problem Undertaken

  Manually going through various social media sites and checking the famous personalities comments, it had more than enough to offer as the abuse was so visible and disclosed that every possible individual who's on social media can read it.

  The initial and immediate reaction I had after going through such comments was of disgust feeling. Wonder how the celebrities take such hate and lead a normal life every single day.

  The motivation was clearly was to build a model to detect the abuse and curb such activities.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem
    1. Given data was totally a text type, it had comments from the social media post of celebrities. The only analytical approach required was to curb those sentences which are more than 400 words in length for a better model readability.

Removing excess info. (Data compromise for a better model)

```
comments = []
labels = []

for ix in range(comment.shape[0]):
    if len(comment[ix])<=400:
        comments.append(comment[ix])
        labels.append(label1[ix])
```

    2. Another step taken was to reduce the no.of values in both X_train and X_test set in order to match with the count of Y.

- Data Pre-processing Done

The provided data (Train and Test) had no null values. The datatype was of object, it was a text data. Source of the data was from the comments from users under the celebrities' post on social media.

The pre-processing steps involved were:

1) Removing Punctuations and other special characters
2) Splitting the comments into individual words
3) Removing Stop Words
4) Lemmatising
5) Applying TfIdf Vectoriser
6) Splitting dataset into Training and Testing

NLP text-preprocessing steps were used to make the data ready for the model to understand and yield the desired results as per the business requirement.

- Data Inputs- Logic- Output Relationships

  The input data was an object datatype which had texts. Source of the data was from the comments from users under the celebrities' posts on social media.

  The relationship between the output and input is that the final output relies on the sentiment that the particular input carries. If the particular input carries the sentiment that of an abuse, the output will be one which validates it.

  The basic logic used in building the relation between the input and output was to train the model better with the respective train's test data set.

- Hardware and Software Requirements and Tools Used
  1) **Pandas**, **Numpy**
  2) **Seaborn** and **Matplotlib** for plotting.
  3) **Wordcloud** to display most occurring words from the data.
  4) **WordNetLemmatizer** from **nltk** package for lemmatizing the tokens.
  5) **Stopwords** from **nltk** package for removing stopwords from the data.
  6) **TfidfVectorizer** from **sklearn** package for vectorizing the text.
  7) **XGBoost, RFC, Naïve-Bayes, SVC, KNN** are some of the libraries imported for the model building.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  1) Given data was totally a text type, it had comments from the social media post of celebrities. The only analytical approach

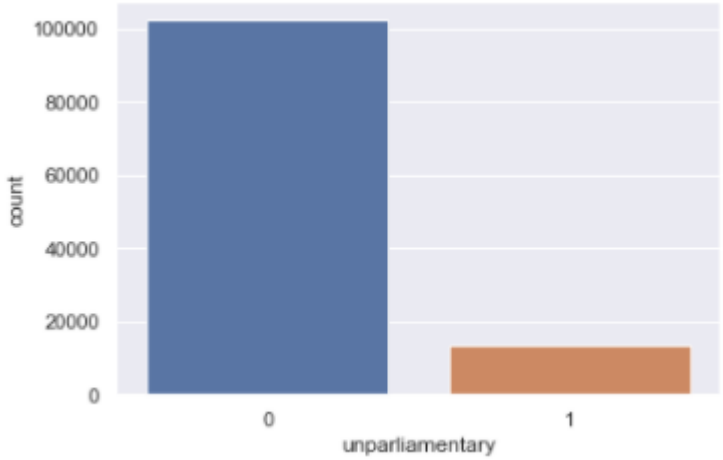
required was to curb those sentences which are more than 400 words in length for a better model readability

Removing excess info. (Data compromise for a better model)

```
comments = []
labels = []

for ix in range(comment.shape[0]):
    if len(comment[ix])<=400:
        comments.append(comment[ix])
        labels.append(label1[ix])
```

2) Creating a separate and new column named 'Unparliamentary' in the lablelss columns was a new and unique to come up with, it was then assigned to Y as a target variable.
3) SMOTE approach was used for Y (target variable) In order to tackle the imbalance nature of its data.



4) SMOTE helped the data to get its balanced distribution thus, avoiding the model from skipping the minority data.

- Run and Evaluate selected models
  1) **XGBoost**: One of the fast-paced ensemble techniques used here to train and test the data.

```
import xgboost
xgb = xgboost.XGBClassifier()
xgb.fit(X_train, Y_train)
y_pred_train = xgb.predict(X_train)

y_pred_test = xgb.predict(X_test)

print(confusion_matrix(Y_test,y_pred_test))
print(classification_report(Y_test,y_pred_test))

print('\n')

print('Training accuracy is {}'.format(accuracy_score(Y_train, y_pred_train)))

print('\n')

print('Test accuracy is {}'.format(accuracy_score(Y_test,y_pred_test)))
```

```
[16:24:45] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0, the default evaluation metric used with the objective 'binary:logistic' was
t eval_metric if you'd like to restore the old behavior.
[[35417   666]
 [ 4714    96]]
              precision    recall  f1-score   support

           0       0.88      0.98      0.93     36083
           1       0.13      0.02      0.03      4810

    accuracy                           0.87     40893
   macro avg       0.50      0.50      0.48     40893
weighted avg       0.79      0.87      0.82     40893


Training accuracy is 0.9577733333333334


Test accuracy is 0.868437140830949
```

2) **Naïve-Bayes**:

```
MNB = MultinomialNB()
MNB.fit(X_train, Y_train)
predictions = MNB.predict(X_test)
```

```
# Model evaluation

print(classification_report(Y_test, predictions))
print(confusion_matrix(Y_test, predictions))

MNB_f1 = round(f1_score(Y_test, predictions, average='weighted'), 3)
MNB_accuracy = round((accuracy_score(Y_test, predictions)*100),2)

print("Accuracy : " , MNB_accuracy , " %")
print("f1_score : " , MNB_f1)
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.91     36083
           1       0.12      0.07      0.09      4810

    accuracy                           0.83     40893
   macro avg       0.50      0.50      0.50     40893
weighted avg       0.79      0.83      0.81     40893

[[33560  2523]
 [ 4464   346]]
Accuracy :  82.91  %
f1_score :  0.81
```

3) **Random Forest Classifier**: It is an ensemble model used here, gave a accuracy score of .82. The hyperparameter tuning such as finding the right parameters using GridSearchCV was crucial.

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

parameters={"n_estimators":[10,100,500]}

rfc=RandomForestClassifier()

clf = GridSearchCV(rfc, parameters, cv=10)
clf.fit(X_train,Y_train)
clf.best_params_
```

```
{'n_estimators': 500}
```

```
rfc = RandomForestClassifier(n_estimators=500)

rfc.fit(X_train,Y_train)
rfc_predict=rfc.predict(X_test)
```

```
print(classification_report(Y_test,rfc_predict))
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.90     36083
           1       0.12      0.08      0.09      4810

    accuracy                           0.83     40893
   macro avg       0.50      0.50      0.50     40893
weighted avg       0.79      0.83      0.81     40893
```

```
print("Model Acc:",rfc.score(X_test,Y_test))
```

```
Model Acc: 0.8264006064607634
```

4) **KNN:** The kind of model which uses the distance based method to process the data has efficiently used GridSearchCV to find the right parameters like n_neighbours and CV to get the better performance.

```
from sklearn.neighbors import KNeighborsClassifier
import warnings
warnings.filterwarnings("ignore")

kc=KNeighborsClassifier()

neighbors={"n_neighbors":range(1,10)}
clf = GridSearchCV(kc, neighbors, cv=5)
clf.fit(X_train,Y_train)
clf.best_params_
```

```
{'n_neighbors': 1}
```

```
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train,Y_train)
knn_predict=knn.predict(X_test)
```

```
print("Model Acc:",knn.score(X_test,Y_test))
```

```
Model Acc: 0.8723843206601827
```

5) **SVC:**

```
from sklearn.svm import SVC

svc=SVC()

parameters={"kernel":["linear", "poly", "rbf"],"C":[0.01,0.1,1]}
clf = GridSearchCV(svc, parameters, cv=5)
clf.fit(X_train,Y_train)
clf.best_params_
```

```
{'C': 1, 'kernel': 'linear'}
```

```
svc=SVC(C=1,kernel="linear")

svc.fit(X_train,Y_train)
svc_predict=svc.predict(X_test)
```

```
print("Model Acc:",svc.score(X_test,Y_test))
```

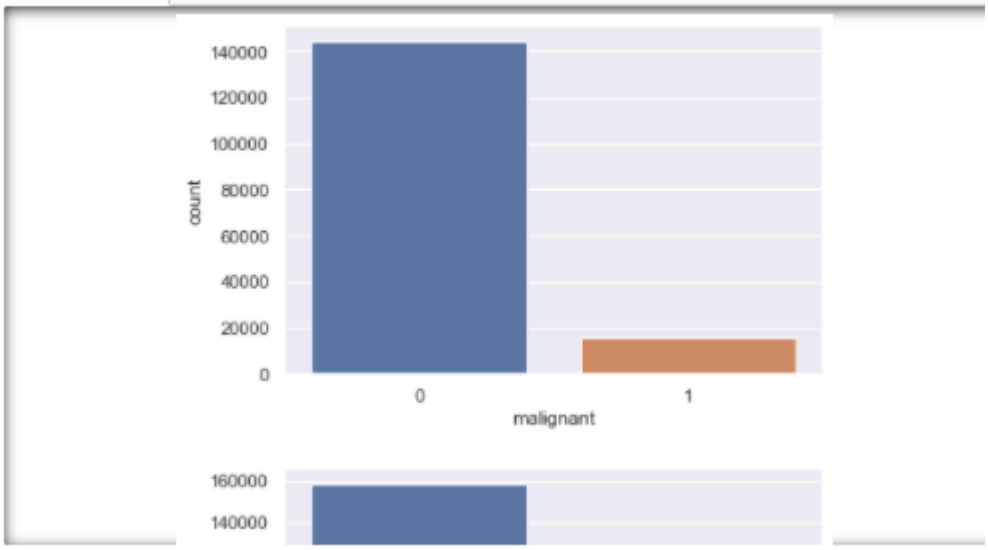
```
Model Acc: 0.8659003831417624
```

- Visualizations

1) **Countplot**:

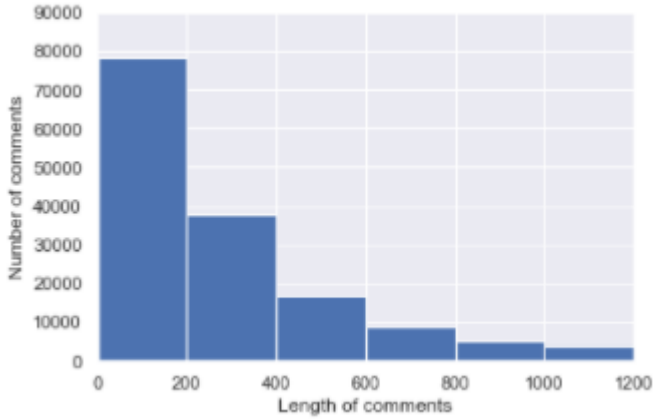
The basic nature of countplots is to represent the count of column values pictorially.

```
In [234]: for i in col:
              sns.countplot(train[i])
              plt.show()
```

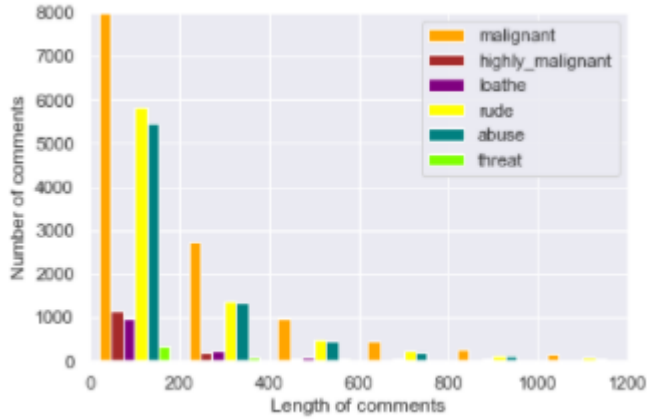


The above plot had represented all the column/features. It clearly helped to understand that the distribution of 0 and 1 is imbalanced. The count of 0 was higher than 1.

2) **Analyse the no. of comments having lengths varying from 0 to 1200:**

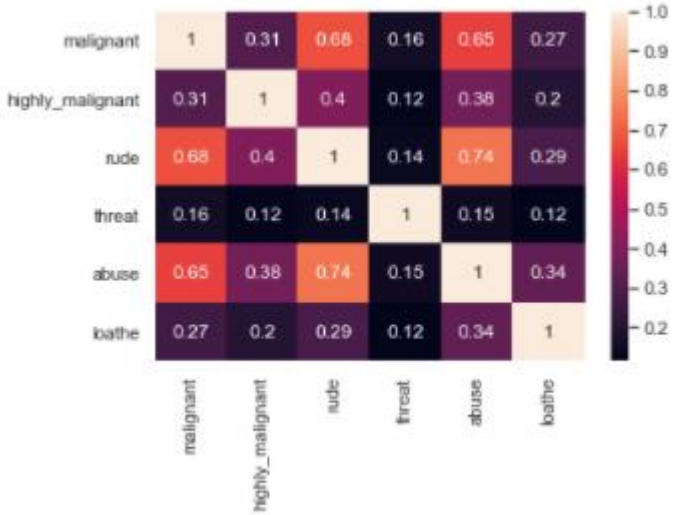


3) **Features classification depending on their lengths.**

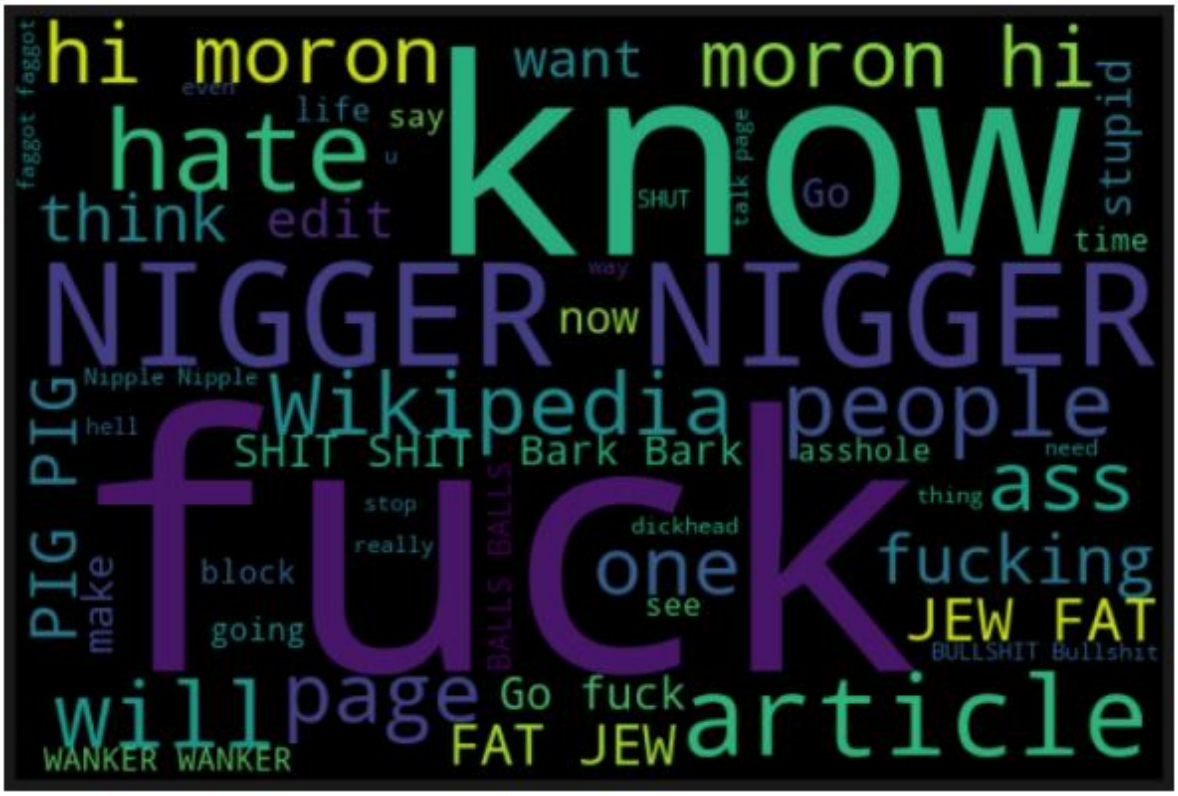


The plot has the countplot in varied color showing each feature and its count in the data. With the help of this plot, it becomes easy to understand which features are more dominant.

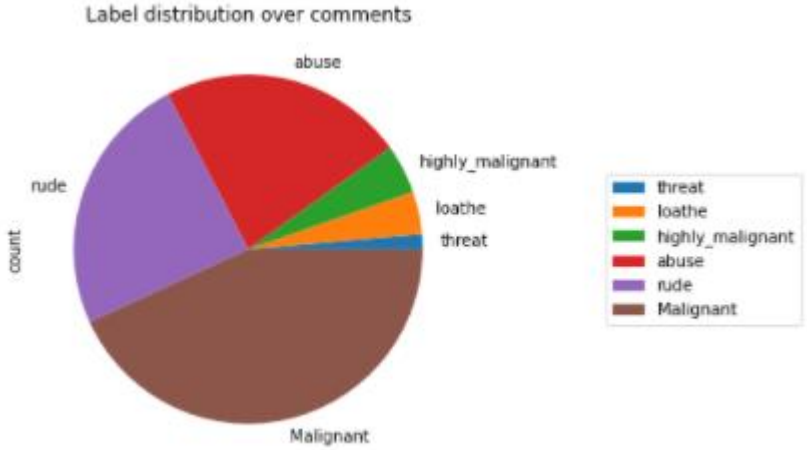
4) **HeatMap**: It represents the correlation of features. For understanding the data better, correlation plays a crucial part.



5) **WordCloud**: The representation of the most used words in the data can be shown in a single picture. Bigger the words are, bigger are their usage in the data.



6) **Pie diagram**: The wheel or sphere shows the distribution clearly to understand how much the data is spread features wise.



CONCLUSION

- Key Findings and Conclusions of the Study
  1) The major finding was the model was able to understand the underlying sentiment of the texts and predict the value correctly.
  2) NLP methods have come in handy to get noise off the data. The sentiment analysis was also helpful for the model to predict the values correctly.
  3) Abuse is an abuse, the target variable was an amalgamation of all the abuse levels into one, it became more easier for understanding which one was an abuse and which one wasn't.
- Learning Outcomes of the Study in respect of Data Science
  1) The power of visualization was realized more in this project as it had more information to infer and process the upcoming steps accordingly. The relationships like correlation became more easier by plotting it.
  2) The data was in lakhs, it needed visualization to see its distribution because it is not possible to check every single value manually. Thus, visualization have come in handy.
  3) The abuse in texts were mixed with the other regular texts. It was tough to focus on those in particular, but thanks to the visual techniques, the job became easier to spot those.
- Limitations of this work and Scope for Future Work

  The scope for this domain is going to be never ending as the online abuse is quite common and relevant in the real world. The steps being taken to control have been exemplary but it isn't enough. There's a significant team working on this issue, thus, the scope is high and above.