MALIGNANT COMMENTS CLASSIFIER PROJECT

SUBMITTED BY : **SINDHU SHREE N**

INTERNSHIP BATCH : **19**

ACKNOWLEDGMENT

With great pleasure, I sincerely express my deep sense of gratitude and heartfelt thanks to several individuals from whom I received impetus motivation during the internship project work. I am very grateful to **SHUBHAM YADAV** sir internship 19th batch guide, for the help and I do express my deep gratitude to sir for his guidance, keen interest and constant encouragement throughout the internship project work. I am thanking him for personal concern, affinity and great spirit with which he has guided the work. And I extend my sincere thanks for the https://www. Kaggle.com/malignant-data-preprocessing, a public data platform which provided a reference to complete this project.

INTRODUCTION

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.

To protect users from being exposed to offensive language on online forums or social media sites, companies have started flagging comments and blocking users who are found guilty of using unpleasant language. Several Machine Learning models have been developed and deployed to filter out the unruly language and protect internet users from becoming victims of online harassment and cyberbullying.

Business Problem Framing

“To build a multi-headed model that’s capable of detecting different types of toxicity like malignant, threats, abuse, loathe.”

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.

Conceptual Background of the Domain Problem

Sentiment classification regarding malignant has been intensively researched in the past few years, largely in the context of social media data where researches have applied various machine learning systems to try and tackle the problem of the malignant as well as the related, which is a task of sentiment analysis.

Review of Literature

Comment abuse classification research initially began with Yin et al’s application of combining TF-IDF with sentiment features. They compared the performance of this model with a simple TF-IDF model and reported 6% increase in F1 score of the classifier on chat style datasets.

Motivation for the Problem Undertaken

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.

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.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

1. Included in ‘Data-cleaning ipynb’.
2. Selecting relevant features.
3. Exploratory the data analysis and data cleaning.
4. Null value imputation
5. Training a machine learning model

6. Hyperparameter tunning
7. Evaluate the model
8. Predictions of the model

Data Sources and their formats

The data is obtained from various online platforms and stored it as a CSV file. Which was imported and performed the data cleaning process.

Data Pre-processing Done

The data set for building the classification model was acquired from the competition site and it included the training set as well as the test set. The steps elaborated below will describe the entire process from Data Pre-Processing to Model Testing.

1. Checking for missing values

   The first and foremost step after importing the training and testing sets in pandas dataframe is to check for the null values. Uing 'isnull' function on both training and testing sets. Here I discovered no missing values so the data is clean and hence I can proceed to the next step.

2. Text Normalising

   As I did not found any missing values now I can proceed for the data preprocessing. Since our data is directly extracted from online platforms it contains special characters, numbers, unnecessary spaces etc., hence the text has to be normalize.

   1) Converting data to lower case
   2) Removing Punctuation
   3) Removing unnecessary white spaces between the characters
   4) Removing "\n"
   5) Replacing all Email id's to 'emailaddress'.
   6) Replacing phone numbers to 'phonenumber'.

3. Lemmatization:

   Since the data is now clean and consistent, it is the right time to perform ***Lemmatization***. Lemmatization is the process of grouping

together the different inflected forms of a word so they can be analyzed as a single item. For example, we do not want the Machine Learning algorithm to treat running, runs, and run as three separate words because, in truth, they are not. Lemmatization helps reduce the words “running” and “runs” to their root form, i.e. run. To implement Lemmatization, I imported “WordNetLemmatizer” from the “nltk” library, and applied it to clean the data that I procured from Step 2.

4. Removing stopwords:

   As we all know, it is one of the most crucial steps in text preprocessing for use-cases that involve text classification. Removing stopwords ensures that more focus is on those words that define the meaning of the text. To remove stopwords from my data, first we imported stopwords from nltk.corpus and used lambda function to apply it for the data.

Data Inputs- Logic- Output Relationships

Since it is a classification model where the target feature is discrete data, so we used **correlation matrix** to find the relation between the features.

Set of assumptions (if any) related to the problem under consideration

Since it is a real-world problem and based on the users comments no assumptions are needed. So I have made no assumptions here.

Hardware and Software Requirements and Tools Used

Hardware requirements:

PROCESSOR: Intel(R) Core(TM) i3 CPU

MONITOR : Any display unit

HARD DISK : 240GB SSD

RAM : 8.00GB

Software requirements:

OPERATING SYSTEM: Windows 10 Pro

FRONT END : Jupyter Notebook (Anaconda3)

BACK END : Excel 2013

Tools Used:

1) Pandas Library

2) Numpy Library

3) Seaborn Library

4) Matplotlib

5) Scikit_learn

6) nltk library

7) TF-IDF

Model/s Development and Evaluation

Since we have completed the data pre-processing and feature engineering part of our project, we move on to the model creation and model assessment part of the project. Before trying to fit a classification models on the training data, I randomly split the data into train-set and test-set. The test set accounts for 30% of the training data.

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

1) Logistic Regression
2) Decision Tree Classifier
3) Random Forest classifier
4) KNeighbors Classifier
5) AdaBoost Classifier

Run and Evaluate selected models

1) **Logistic Regression**: It is the most commonly used type of classifier for binary classification. In this model we can observe that the precision, recall and f1 score are 96%, 99% and 98% respectively.

```
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("\n","Confusion matrix:\n",confusion_matrix(y_test,y_pred_test))
print("\n","Classification report:\n",classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9592207629432672
Test accuracy is 0.9553601270053476

 Confusion matrix:
 [[42769   236]
 [ 1901  2966]]

 Classification report:
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     43005
           1       0.93      0.61      0.74      4867

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

2) **Decision Tree Classifier:** It is a type of classification model by building a decision tree. In this case it has a precision, recall and f1-score are 96%, 97% and 97% respectively.

```
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","Confusion matrix:\n",confusion_matrix(y_test,y_pred_test))
print("\n","Classification report:\n",classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9986033894663336
Test accuracy is 0.9398604612299465

 Confusion matrix:
 [[41644  1361]
 [ 1518  3349]]

 Classification report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     43005
           1       0.71      0.69      0.70      4867

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

3) **Random Forest Classifier:** It is a type of ensemble learning method that uses numerous decision trees to achieve higher prediction accuracy and model stability. Every tree classifies a data instance based on attributes, and the forest chooses the classification that received most instances. It has a precision of 97% , recall of 99% and f1 score 98%.

```
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("\n","Confusion matrix:\n",confusion_matrix(y_test,y_pred_test))
print("\n","Classification report:\n",classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9986839631509682
Test accuracy is 0.957449030748663

 Confusion matrix:
 [[42493   512]
 [ 1525  3342]]

 Classification report:
               precision    recall  f1-score   support

           0       0.97      0.99      0.98     43005
           1       0.87      0.69      0.77      4867

    accuracy                           0.96     47872
   macro avg       0.92      0.84      0.87     47872
weighted avg       0.96      0.96      0.96     47872
```

4) **KNeighbors Classifier**: It is one of the simplest algorithms used in machine learning for regression and classification problem. It use data and classify new data points based on similarity measures. Classification is done by a majority vote its neighbors. Here it has the precision 92%, recall 100% and f1 score of 96%.

```
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("\n","Confusion matrix:\n",confusion_matrix(y_test,y_pred_test))
print("\n","Classification report:\n",classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9222195364327344
Test accuracy is 0.9178434157754011

 Confusion matrix:
 [[42830   175]
 [ 3758  1109]]

 Classification report:
               precision    recall  f1-score   support

           0       0.92      1.00      0.96     43005
           1       0.86      0.23      0.36      4867

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

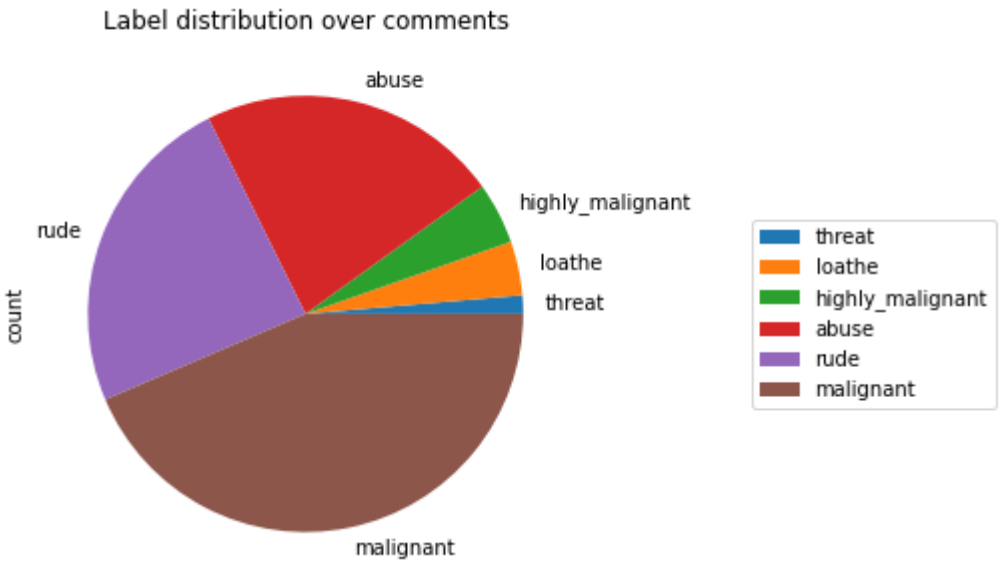
5) **AdaBoost Classifier:** It is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset.

```
adab=AdaBoostClassifier(n_estimators=100)
adab.fit(x_train,y_train)
y_pred_train=adab.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,y_pred_train)))
y_pred_test=adab.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print("\n","Confusion matrix:\n",confusion_matrix(y_test,y_pred_test))
print("\n","Classification report:\n",classification_report(y_test,y_pred_test))
```

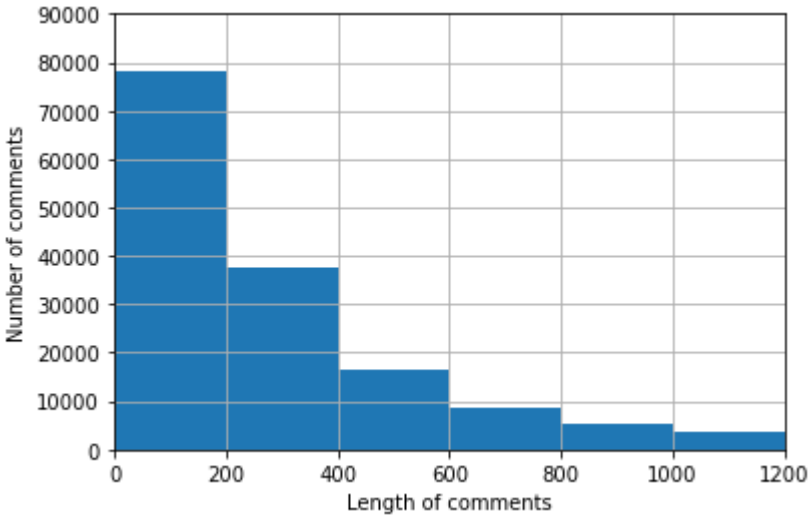
```
Training accuracy is 0.9503755628967135
Test accuracy is 0.9486129679144385

 Confusion matrix:
 [[42567   438]
 [ 2022  2845]]

 Classification report:
               precision    recall  f1-score   support

           0       0.95      0.99      0.97     43005
           1       0.87      0.58      0.70      4867

    accuracy                           0.95     47872
   macro avg       0.91      0.79      0.84     47872
weighted avg       0.95      0.95      0.94     47872
```

Visualization:

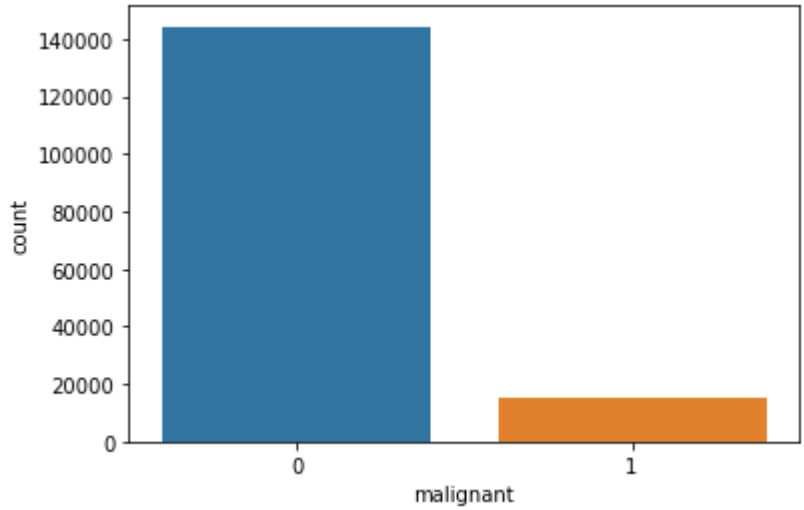
I have visualized the data in different aspects and used pie charts, histograms, barplots, countplot and finally Auc roc curve which are showed below.

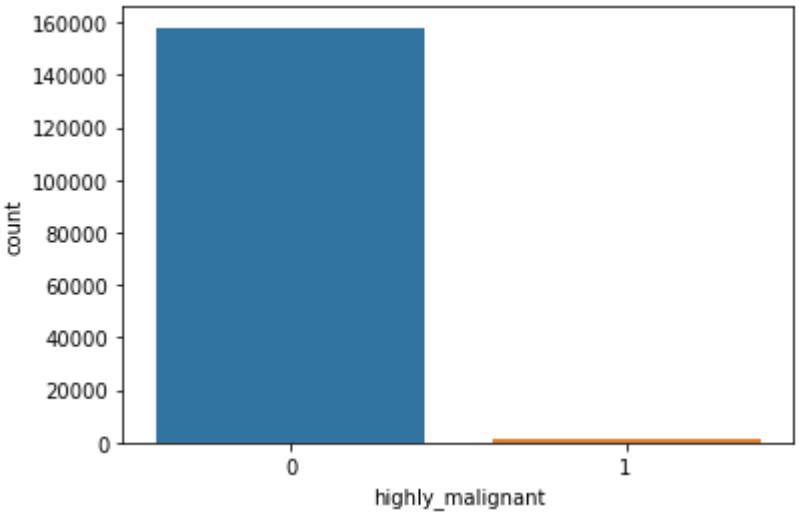


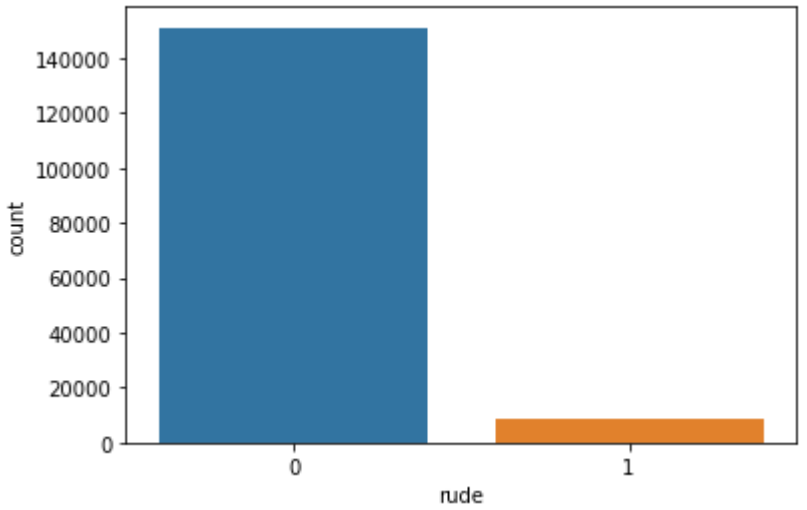
Here we can see the amount of toxic words used in the comments. So there are almost 40% of malignant words and 45% combination of rude and abuse words and remaining 15% are highly malignant, loathe and threat words used.

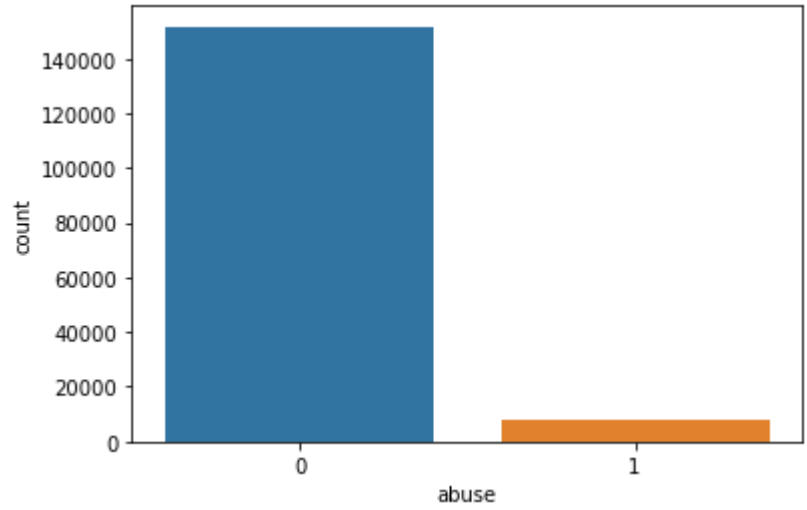


This is showing that as the number of comments increasing it has a decreased length.

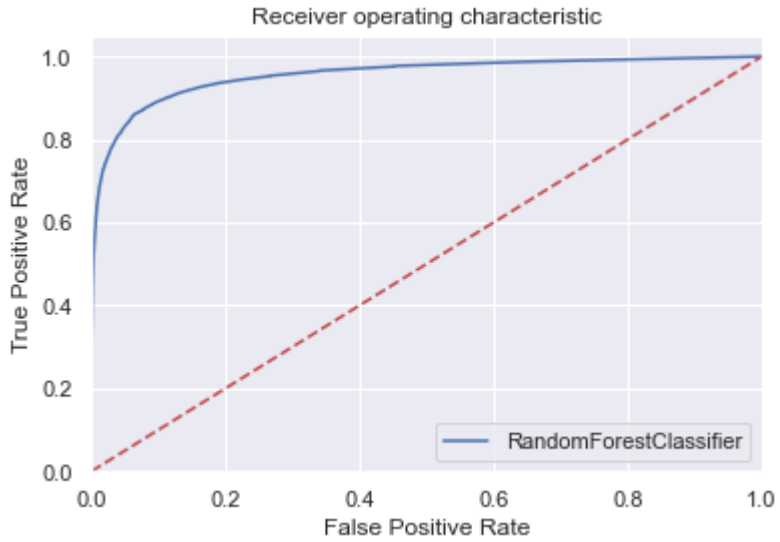








All these above graphs are showing the count of toxic words used using bar plots. In all the graphs the count of usage of toxic words are very less when compare normal texts.



It is a Auc Roc Curve which is plotted between false positive rate and true positive rate. The number of words are toxic from the group of tested words is 90% accurate.

Interpretation of the Results

On comparing all the results of these five models mentioned above we have observed that all models are performing well but the training set and testing set score is high almost near in random forest classifier and also has a comparative cross validation score. Thus we have choosed this model for hyperparameter tuning and there also we got a improved result. Hence **Random Forest Classifier** is the best fit for the present dataset.

CONCLUSION

Key Findings and Conclusions of the Study:

After evaluating the results procured during the training phase of my project and the results that I received from the websites, I can claim that the random Forest classifier performs better other classification models. And using it we have predicted the output for the testing set which is in another file.

Learning Outcomes of the Study in respect of Data Science:

This project allowed me to work with five different classification models and further, I was able to implement them on Natural Language Processing use-case. The various data pre-processing and feature engineering steps in the project made me to have knowledge of the efficient methods that can be used to clean textual data. I understood the working of various classification models such as Decision tree, KNeighbors, Logistic Regression, Random Forest and AdaBoost Classifier. I got introduced to the concepts of lemmatization, stopwords and advantages of using it in textual data. Finally doing hyper-parameter tuning for the model helped me to achieve optimum results.

Limitations of this work and Scope for Future Work

- Limited Access to information
- Time Limits
- Conflicts on biased views and personal issues
- How to structure my project research limitation correctly
- How to set my project research limitation.
- Formulation of my objectives and aims

- Implementation of my data collection methods
- Scope of discussions
- Finding my error on the codes

****