MALIGNANT CLASSIFIER

Submitted by:

KEERTHI PERUMAL

ACKNOWLEDGMENT

my sincere thanks to Flip Robo Technologies, Bangalore for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I want to thank my SME Mr. Shubham Yadav for providing the Dataset and helping to solve the problem and addressing out our Query in right time & members of Flip Robo for their kind co-operation and encouragement which help me in completion of this project.

I was able to complete my tasks properly and were up to the mark in all the tasks assigned. During the process, I got a chance to see the stronger side of my technical and non-technical aspects and also strengthen my concepts.

INTRODUCTION

INTRODUCTION

Problem Statement

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.

Business Problem

Framing 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. This is a multi-label classification problem.

Conceptual Background of the Domain Problem

In the past few years, it's seen that the cases related to social media hatred have increased exponentially. Social media is turning into a dark venomous pit for people nowadays. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc. In social media the people spreading or involved in such

kind of activities use filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.

Review of Literature

Studies regarding toxicity has been intensively researched in the past few years, largely in the context of social media data where researchers have applied various machine learning systems to try and tackle the problem of toxicity as well as the related, more well-known task of sentiment analysis. Comment abuse classification research begins with combining of TF-IDF with sentiment/contextual features. The motivation for our project is to build a model that can detect toxic comments and find the bias with respect to the mention of select identities.

Motivation for the Problem Undertaken

The upsurge in the volume of unwanted comments called malignant comments has created an intense need for the development of more dependable and robust malignant comments filters. Machine learning methods of recent are being used to successfully detect and filter malignant comments. Build a model which can be used

to predict in terms of a probability for comments to be malignant. In this case, Label ‘1’ indicates that the comment is malignant, while, Label ‘0’ indicates that the comment is not malignant.

ANALYTICAL PROBLEM FRAMING

Model Building Phase

You need to build a machine learning model. Before model building do all data pre-processing steps involving NLP. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science.

Include all the steps like

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model Data Sources and their formats

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes ‘Id’, ‘Comments’, ‘Malignant’, ‘Highly malignant’, ‘Rude’, ‘Threat’, ‘Abuse’ and ‘Loathe’.

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments

which have multiple labels. The first attribute is a unique ID associated with each comment.

The data set includes:
- Malignant: It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- Highly Malignant: It denotes comments that are highly malignant and hurtful.
- Rude: It denotes comments that are very rude and offensive.
- Threat: It contains indication of the comments that are giving any threat to someone. - Abuse: It is for comments that are abusive in nature.
- Loathe: It describes the comments which are hateful and loathing in nature.
- ID: It includes unique Ids associated with each comment text given.
- Comment text: This column contains the comments extracted from various social media platforms.

Both train and test csv(s) are loaded respectively, where, in training dataset, the independent variable is Comment text which is of 'object'type and rest 6 categories or labels are the dependent features whose values needs to be predicted, are of Boolean in nature being 'int64' type.

Data Preprocessing

```
df_train['malignant'].value_counts()
```

```
0    144277
1     15294
Name: malignant, dtype: int64
```

```
df_train['highly_malignant'].value_counts()
```

```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```

```
df_train['rude'].value_counts()
```

```
0    151122
1      8449
Name: rude, dtype: int64
```

```
df_train['threat'].value_counts()
```

```
0    159093
1       478
Name: threat, dtype: int64
```

```
df_train['abuse'].value_counts()
```

```
0    151694
1      7877
Name: abuse, dtype: int64
```

```
df_train['loathe'].value_counts()
```

```
0    158166
1      1405
```

STATISTICAL SUMMARY

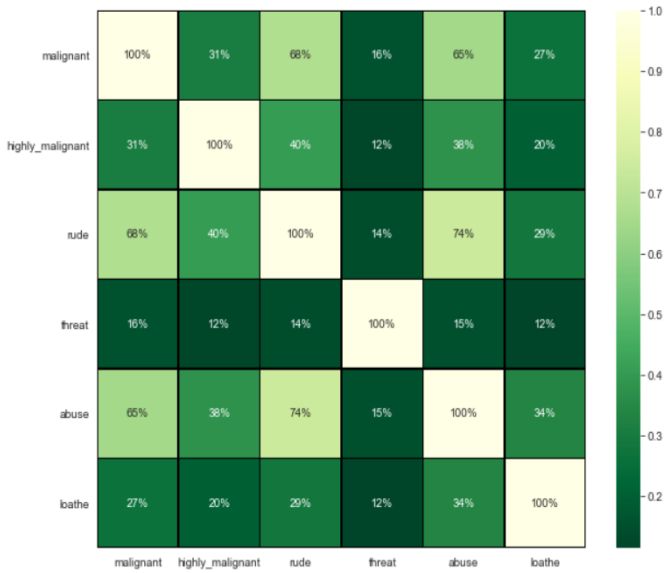
```
df_train.describe()    #Statistical summary of the dataset
```

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- The minimum value and the maximum value of the attributes is same i.e., 0 and 1 respectively.
- The mean and standard deviation is nearly 0-1 of all the attributes in the training dataset.
- Here, with this statistical analysis, it is interpreted that there are no outliers as well as skewness present in this training dataset.

- The count of each field is equal which shows that there are no missing values present.

DATA CORRELATION



- The highest positive correlation is seen in between fields 'rude' and 'abuse'.
- Attribute 'threat' is negatively correlated with each and every other feature of this training dataset.
- Overall the correlation among the attributes is not positive.

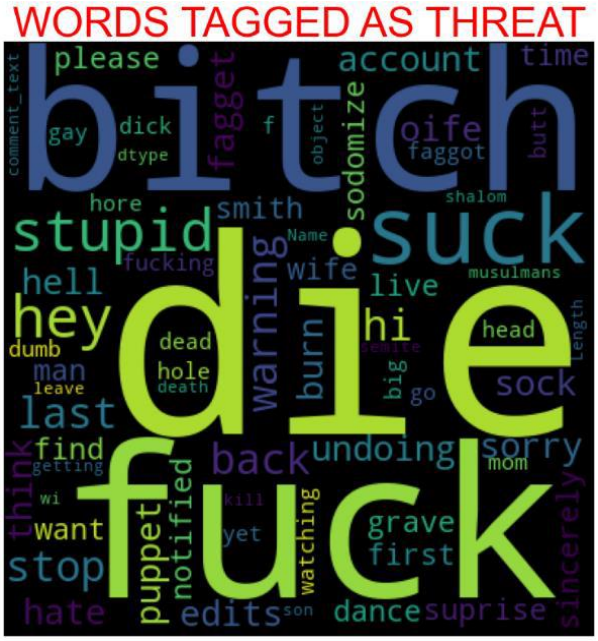
DROPPING COLUMNS

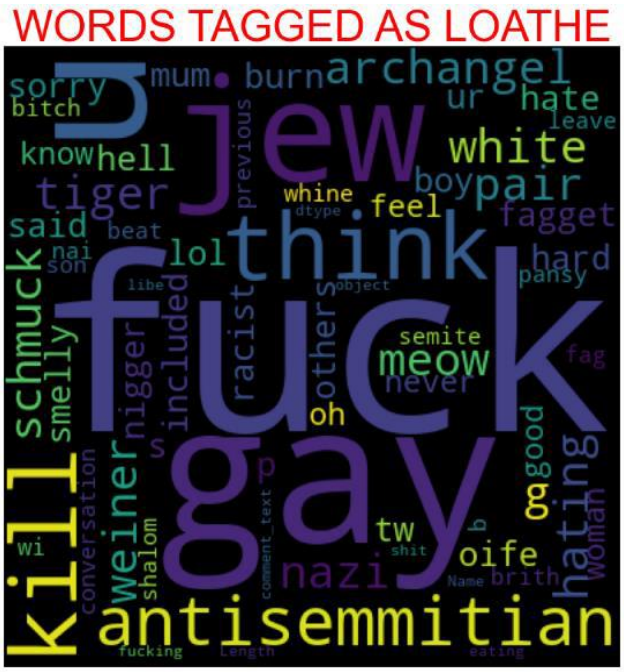
- Due to the wide range of given data, it is extremely fruitful to clean, shape and set the data in the most suitable form. Dropping unnecessary columns declines the chances of producing errors. Thus, column ‘ID’ was dropped from the train dataset as every comment has its own unique id. After dropping the same, the dataset is now having 7 attributes in total including the target variables.

Cleaning the data using NLP

- Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. This data is usually not necessary or helpful when it comes to analysing data because it may hinder the process or provide inaccurate results. Before cleaning the data, a new column is created named ‘length_before_cleaning’ which shows the total length of the comments respectively before cleaning the text.

Plotting wordcloud for each feature





we can see that the large texts have more weightage in their respective type of comments whereas small texts have the lesser weightages

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used Jupyter notebook to do my python programming and analysis.

For using an CSV file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:

```
#Importing warning library to avoid any warnings
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

#Importing Required libraries for NLP
import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

#Importing the Oversampling library and Counter for handling imbalanced dataser
from collections import Counter
from imblearn.over_sampling import RandomOverSampler

#Libraries for model building
from sklearn.model_selection import train_test_split
#Importing required libraries
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn.metrics import hamming_loss, log_loss

#Saving the model
import pickle
```

```
#Making a for loop and calling the algorithm one by one and save data to respective model using append function
Model=[]
score=[]
cvs=[]
rocscore=[]
h_loss=[]
l_loss=[]
for name,model in models:
    print('******************************',name,'**************************')
    print('\n')
    Model.append(name)
    model.fit(x_train_os,y_train_os)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('accuracy_score: ',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,X,y,cv=5,scoring='accuracy').mean()
    print('cross_val_score: ',sc)
    cvs.append(sc*100)
    print('\n')
    false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
    roc_auc= auc(false_positive_rate,true_positive_rate)
    print('roc_auc_score: ',roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')
    hloss = hamming_loss(y_test, pre)
    print("Hamming_loss:", hloss)
    h_loss.append(hloss)
    print('\n')
    try :
        loss = log_loss(y_test, pre)
    except :
            loss = log_loss(y_test, pre.toarray())
```

After running the above for loops for the algorithms, the output will be as follows we found random forest classifier as best fit

```
RandomForestClassifier()

accuracy_score:  0.9558405748663101

cross_val_score:  0.9570786587653991

roc_auc_score:  0.8289565962677834

Hamming_loss: 0.04415942513368984

Log_loss : 1.5252209620475623
```

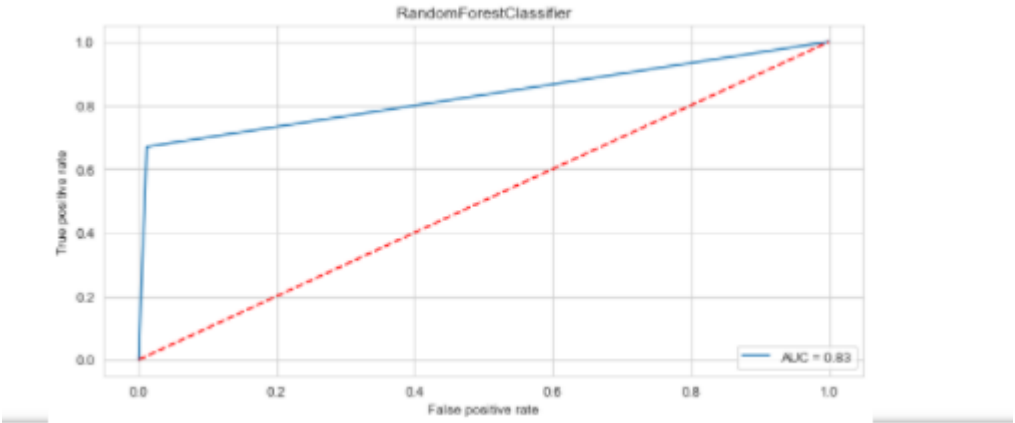
Classification report:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	43004
1	0.87	0.67	0.76	4868
accuracy			0.96	47872
macro avg	0.91	0.83	0.87	47872
weighted avg	0.95	0.96	0.95	47872

Confusion matrix:

```
[[42498   506]
 [ 1608  3260]]
```

AUC_ROC curve:



ut[74]:

	Model	Accuracy_score	Cross_val_score	roc_auc_score	Hamming_loss	Log_loss
0	Logistic Regression	95.487968	95.608224	80.300911	0.045120	1.558405
1	MultinomialNB	94.635695	94.660057	75.172106	0.053643	1.852768
2	DecisionTreeClassifier	94.048713	94.116725	82.842634	0.059513	2.055524
3	KNeighborsClassifier	91.723763	91.803649	62.238599	0.082762	2.858516
4	RandomForestClassifier	95.584057	95.707866	82.895660	0.044159	1.525221
5	AdaBoostClassifier	94.489472	94.591749	76.502532	0.055105	1.903276
6	GradientBoostingClassifier	93.946357	94.030870	71.500261	0.060536	2.090857

After running the for loop of classification algorithms and the required metrics, we can see that the best 2 performing algorithms are RandomForestClassifier and XGBoostClassifier because the loss values are less and their scores are the best among all. Now, we will try Hyperparameter Tuning to find out the best parameters and using them to improve the scores and metrics values

HYPERTUNNING

If we run GridSearchCV and RandomSearchCV, it takes more than 2 hours to run the code as the dataset is huge and the best params are not obtained from it due to more computational power requirement. The AUC Score, f1-score and recall value is high when we use randomforest with over sampling data. So, we choose RandomForestClassifier model with over sampled data as our best model among all models.

```
rfc = RandomForestClassifier()
rfc.fit(x_train_os,y_train_os)
```

```
RandomForestClassifier()
```

```
pred=rfc.predict(x_test)
print('Accuracy score: ',accuracy_score(y_test,pre)*100)
print('Cross validation score: ',cross_val_score(rfc,X,y,cv=5,scoring='accuracy').mean()*100)
false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
roc_auc= auc(false_positive_rate,true_positive_rate)
print('roc_auc_score: ',roc_auc)
hloss = hamming_loss(y_test, pre)
print("Hamming_loss:", hloss)
loss = log_loss(y_test, pre)
print("Log loss:", loss)
print('Classification report: \n')
print(classification_report(y_test,pre))
print('Confusion matrix: \n')
print(confusion_matrix(y_test,pre))
```

```
Accuracy score:  94.90516377005348
Cross validation score:  95.66838520306182
roc_auc_score:  0.8547804251569878
Hamming_loss: 0.05094836229946524
Log loss: 1.7597134016088005
Classification report:

              precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.76      0.74      0.75      4868

    accuracy                           0.95     47872
   macro avg       0.86      0.85      0.86     47872
weighted avg       0.95      0.95      0.95     47872
```

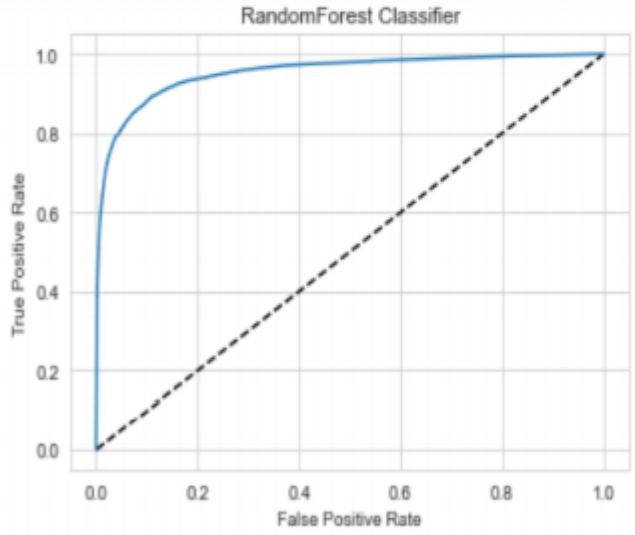
Confusion matrix:

[[41848 1156]

After finding the metrics values and other required scores, we will plot the auc roc curve with the help of auc score obtained with the help of the fitted model above

```
#AUC_ROC Curve of Randomforest Classifier with oversampled data
y_pred_prob=rfc.predict_proba(x_test)[:,1]
fpr,tpr,thresholds=roc_curve(y_test,y_pred_prob)
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='RandomForest Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('RandomForest Classifier')
plt.show()

auc_score=roc_auc_score(y_test,rfc.predict(x_test))
print(auc_score)
```




```
0.830404398757655
```

Finalizing the model

We will final the model by predicting the values and saving the model in a pickle file, which will be used for prediction of test data

```
rfc_prediction=rfc.predict(X)
#Making a dataframe of predictions
malignant_prediction=pd.DataFrame({'Predictions':rfc_prediction})
malignant_prediction
```

	Predictions
0	0
1	0
2	0
3	0
4	0
...	...
159566	0
159567	0
159568	0
159569	0
159570	0

159571 rows × 1 columns

```
#Saving the model
import pickle
filename='MalignantCommentsClassifier_Project.pkl'   #Specifying the filename
pickle.dump(rfc,open(filename,'wb'))
```

CONCLUSION

Key Findings and Conclusions of the Study

-> After the completion of this project, we got an insight of how to preprocess the data, analysing the data and building a model.

-> First, we imported both training and testing data, which had nearly 150000+ records.

-> We did all the required pre-processing steps like checking null values, datatypes check, dropping unnecessary columns, etc.

-> We used the training data for doing Exploratory Data Analysis using various plots and recorded the observations.

-> While observing the results, we found that the dataset was in highly imbalanced side and we need to handle it, in order to avoid overfitting problem.

-> Using NLP, we pre-processed the comment text and did other steps.

-> As the problem was a multi-class classifier, we took a new feature known as label and combined the comment_labels output together using sum() and then stored in that feature. For a binary classification problem, we scaled the data accordingly.

-> After applying Tf-idf Vectoriser, we used an oversampling technique called RandomOverSampler for handling the imbalanced data. There, we took 75% of the high points data and sampled it to the low points data so that both weights could be balanced equally and we could get proper result.

-> Then, we split the data using train_test_split and then we started the model building process by running as many algorithms in a for loop, with difference metrics like cross_val_score, confusion matrix, auc_score, log loss, hamming loss, etc.

-> We found that RandomForestClassifier performing well.

-> The major problem with this dataset occurred in this step. It took me nearly 2 hrs to run the code for finding out the best parameters itself as the dataset is large and more

computational power was required. Even though we found the best algorithms, it took me 2 hrs to get the results.

-> Therefore, without hyperparameter tuning, we finalized RandomForest as the best performing algorithm by predicting the outputs, saving the model and storing the results in a csv file

-> Then, by using the model we got, another set of predictions were done by using the test data and the results were stored in a separate csv file

The next step was to perform Problems faced while working in this project:

- More computational power was required as it took more than 2 hours
- Imbalanced dataset and bad comment texts
- Good parameters could not be obtained using hyperparameter tuning as time was consumed morehyperparameter tuning technique to these models for finding out the best parameters and trying to improve our scores.

Areas of improvement:

- Could be provided with a good dataset which does not take more time.
- Less time complexity
- Providing a proper balanced dataset with less errors

Learning Outcomes

Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stopwords. We were also able to learn to convert strings into vectors through hash vectorizer. In this project we applied different evaluation metrics like log loss, hamming loss besides accuracy.

My point of view from my project is that we need to use proper words which are respectful and also avoid using abusive, vulgar and worst words in social media. It can cause many problems which could affect our lives. Try to be polite, calm and composed while handling stress and negativity and one of the best solutions is to avoid it and overcoming in a positive manner