FLIP ROBO

MALIGNANT COMMENTS CLASSIFICATION PROJECT

Submitted by:

Mrs. Chaitali Mohit Nakade

ACKNOWLEDGMENT

- ✓ Towards data science – A medium publication sharing concepts, ideas and codes
- ✓ James Le (May 14, 2018) - FIFA World Cup 2018: A Data-Driven Approach to deal Team Line-Ups
- ✓ Laree Pertold -Tree Hozz (18th February, 2020)
- ✓ Niklas Donges (May 14, 2018) - Predicting the Survival of Titanic Passengers

ARTICLE RESOURCES

- ✓ Notebook and Data: GitHub, Jupyter Notebook
- ✓ Libraries: numpy, pandas, sklearn, scipy, joblib, matplotlib, seaborn, statsmodels

INTRODUCTION

As we all know, now a days social media is a highest priority of peoples and commenting on any social post is just like a trend. Individuals are expressing their opinions/feelings with comments. But sometimes peoples are getting angry in it like they are posting a bad, malignant, abusive, threatening comments due to that other are getting embraced and uncomfortable. In this project we will make a model with machine learning and differentiate which comments are considered to be a malignant or which one is non-malignant.

- ## 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.

- ## Conceptual Background of the Domain Problem

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.

- ## Review of Literature

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.

- ## Motivation for the Problem Undertaken

With the help of this project, we will find out which comments is malignant, which one is highly_malignant, which one is rude, identifying threat, abuse and loath. And with that we will distinguish that.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Now we will work on model building, we are using machine learning t echniques to predict “Malignant Comments”. Here we are having dataset named as data1, test_data. There are 2 datasets present in this project whi ch are, train dataset and it contains 159571 rows and 8 columns whereas, tes t dataset contains 153164 rows and 2 columns.

Start work with dataset with importing necessary libraries and dataset.

```
#Import libreries

import pandas as pd #Data processing
import numpy as np #Linear algebra
from sklearn.preprocessing import StandardScaler #resize the distribution of values
from sklearn.linear_model import LinearRegression #algorithm
from sklearn.model_selection import train_test_split #estimate the performance
import statsmodels.api as sm
import matplotlib.pyplot as plt #data visualization
import seaborn as sn  #data visualization

import warnings
warnings.filterwarnings("ignore")
```

All these libraries are used for model building and data processing.

```
1 #Import the dataset
2
3 data1= pd.read_csv(r"C:\Users\Chaitali Nakade\OneDrive\Desktop\dataset\malignant.csv")
4 data1.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

In this model we have to make one target column with “LABEL” name. if any of malignant, highly_malignant, rude, threat, abuse and loathe is 1 then label will become 1 and if any of this will become 0 than label become 0. After that we can drop that 6 columns and work with 3 columns only.

The complete dataset is imported in variable name ‘data1’ and we can see there are some columns are need to alter them into numeric data because machine learning will not work with object data type but before that need to analyse the data.

```
for i in data1["threat"]:
    if i==0:
        label3.append(0)
    elif i==1:
        label3.append(1)
```

```
label4 = []
for i in data1["abuse"]:
    if i==0:
        label4.append(0)
    elif i==1:
        label4.append(1)
```

```
label5 = []
for i in data1["loathe"]:
    if i==0:
        label5.append(0)
    elif i==1:
        label5.append(1)
```

```
Label = [x | y | z | a | b |c for x, y, z, a, b, c in zip(label, label1, label2, label3, label4, label5)]
```

```
data1["LABEL"] = Label
```

```
data1
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	LABEL
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	0

Exploratory Data Analysis

- Data Sources and their formats

```
data1.describe()
```

	LABEL
count	159571.000000
mean	0.101679
std	0.302226
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

Express more about dataset:

- We can see in above table majority of comments are not malignant which denotes by 0 and very less comments are malignant in nature which is denoted by 1.

```
1 data1.info()
```

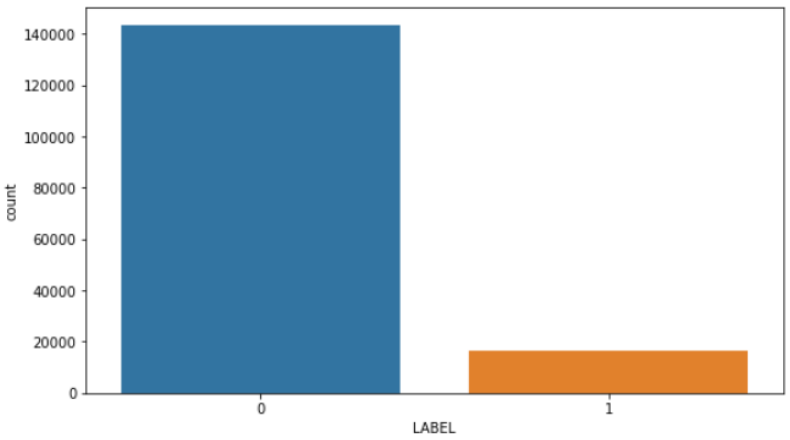
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 3 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   id            159571 non-null  object
 1   comment_text  159571 non-null  object
 2   LABEL         159571 non-null  int64 
dtypes: int64(1), object(2)
memory usage: 3.7+ MB
```

- There are 159571 Rows and 3 Columns in train dataset
- We can see no columns are having null values.
- We can find here two columns are having object data types and rest one column is having integer datatype.

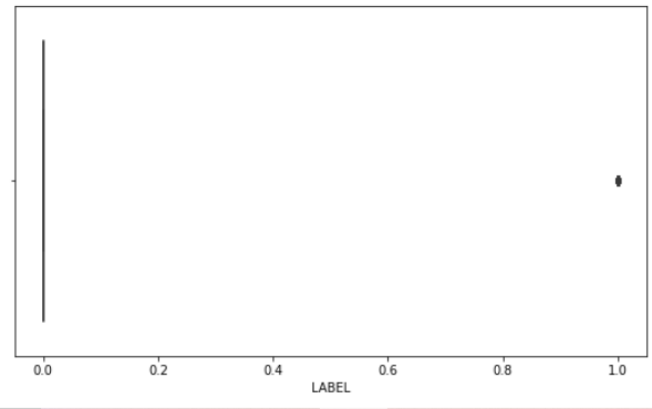
- Data Inputs- Logic- Output Relationships (Visualization of data)

Univariate analysis: Proceeding with the plotting and analysing the data using seaborn, matplotlib libraries –

Plotting count plots for categorical data –



Majority of comments are non-malignant in nature with 140000 above peak and less than 2000 are malignant in nature. Due to this it will create class imbalance issue and model will get biased but we can solve this issue with scaling. We will discuss about this later.



Some outliers are present in target column but it is target variable. So, no need to remove outliers from that.

- Data Pre-processing Done
- Encode the columns which are having object data type

  Encoding data is very important for model because data will not work with object data type. In this project I will encode the object columns with label encoder. Label encoder is work on alphabetical order and encode the column. I am encoding all the columns which are having object data type.

```
#import label encoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
for i in data1.columns:
    if data1[i].dtypes=="object":
        data1[i]=le.fit_transform(data1[i].values.reshape(-1,1))
```

	id	comment_text	LABEL
0	0	72698	0
1	1	68359	0
2	2	79594	0
3	4	35519	0
4	5	146426	0
...	...	...	...
159566	159505	44289	0
159567	159510	145720	0
159568	159524	122203	0
159569	159535	60037	0
159570	159541	31106	0

After encoding all columns, the dataset will be in numeric format and it will look like.

The above steps have to be done with test dataset also.

- Data Inputs- Logic- Output Relationships

a. **Notebook and Data:** GitHub, Jupyter Notebook

b. **Libraries:** numpy, pandas, sklearn, scipy, joblib, matplotlib, seaborn, statsmodels

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

Now we will train several machine learning models and compare their results. We need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation. After that I will compare the difference of accuracy score and cross validation score of all models. The model which is having minimum difference will be consider as a best model and further procedures will done on that model. Following are the basic steps of model building.

x= features, y=Target

```
x = data1.drop(columns = 'LABEL', axis=1)
y = data1['LABEL']
x1 = test_data
```

The next step is to bring the data to a common scale, since there are certain columns with very small values and some columns with high values. This process is important as values on a similar scale allow the model to learn better.

Now, check the variance inflation factor to avoid multicollinearity problem. In this model I am having all the values of variance inflation factor is less than 10 so safe to proceed further.

	vif	Features
0	1.000008	id
1	1.000008	comment_text

- ## Testing of Identified Approaches (Algorithms)

Find best random state: first of all, find the best random state
And then, splitting our dataset by train test split with random state: 35.

```
from sklearn.tree import DecisionTreeClassifier
maxAccu = 0
maxRS = 0
for i in range(1,200):
    x_train, x_test, y_train, y_test = train_test_split(x_res,y_res, test_size=.30, random_state=i)
    mod= DecisionTreeClassifier()
    mod.fit(x_train, y_train)
    pred = mod.predict(x_test)
    acc=accuracy_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu, "on Random_state ", maxRS)
```

```
Best accuracy is  0.9503418286671007 on Random_state  35
```

```
x_train,x_test,y_train,y_test = train_test_split(x_res, y_res, test_size=0.2, random_state = 35)
```

- ## Run and Evaluate selected models

We now proceed to the main step of our machine learning, fitting the model and predicting the outputs. We fit the data into multiple classification models to compare the performance of all models and select the best model –

1. Logistic regression:

Logistic regression is a type of supervised machine learning algorithm. It gives the single result which is made with combination of multiple decision tree output.

- ✓ After applying Logistic regression method, the output of accuracy score is: **56.25%**
- ✓ And cross validation score is**: 51.22%**

2. Random forest classifier:

It is used in both classification as well as regression model. It made up with decision tree. It gives the single result which is made with combination of multiple decision tree output. In the similar way of logistic regression, I am applying the random forest algorithm and find the accuracy score and cross validation score also.

- ✓ After applying Random Forest Classifier method, the output of accuracy score is: **96.75%**
- ✓ And cross validation score is**: 93.67%**

3. Decision Tree classifier

Decision tree uses the tree illustration to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

- ✓ After applying Decision Tree classifier method, the output of R2 score is: **95.41%**
- ✓ And cross validation score is**: 92.84%**

```
from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier()
DT.fit(x_train, y_train)
```

```
DecisionTreeClassifier()
```

```
pred_DT = DT.predict(x_test)
print(accuracy_score(y_test,pred_DT))
```

```
0.954115000261602
```

```
print(confusion_matrix(y_test,pred_DT))
print(classification_report(y_test,pred_DT))
```

```
[[26209  2612]
 [   19 28499]]
              precision    recall  f1-score   support

           0       1.00      0.91      0.95     28821
           1       0.92      1.00      0.96     28518

    accuracy                           0.95     57339
   macro avg       0.96      0.95      0.95     57339
weighted avg       0.96      0.95      0.95     57339
```

```
from sklearn.model_selection import cross_val_score
cv_score= cross_val_score(DT, x_res, y_res, cv=5)
cv_mean=cv_score.mean()
cv_mean
```

```
0.9284040902322441
```

4. KNeighbors Classifier

✓ After applying KNeighbors classifier method, the output of R2 score is: **89.49%**

✓ And cross validation score is**: 85.04%**

- **Interpretation of the Results:**
  - Comparison of all models:

Sr. No.	Algorithms	Accuracy score (a)	Cross -Validation Score(b)	Difference (a-b)
1.	Logistic regressor	82.89%	44.51%	5.03 %
2.	Random Forest classifier	96.75%	93.67%	3.07%
3.	Decision Tree Classifier	95.41%	92.84%	2.56%
4.	KNeighbors Classifier	89.49%	85.04%	4.46%

As shown in above table Decision Tree Classifier is

having minimum difference, so Decision Tree Classifier is a best model. Now, I will work on best model with hyper parameter tunning.

- ## Hyper Parameter Tuning:

Grid search cross validation (GCV) in hyperparameter tuning will help us to finding the best hyperparameter and tune it with training data set and give best accuracy score:

```
# Decision tree Classifier
Parameters = {'max_depth': [2, 3, 5, 10, 20],
              'min_samples_leaf': [1, 10, 20, 50, 100],
              'criterion': ["gini", "entropy"],
              'splitter' : ["best", "random"]
             }
```

```
GCV=GridSearchCV(DecisionTreeClassifier(),Parameters,cv=5)
```

```
GCV.fit(x_train, y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [1, 10, 20, 50, 100],
                         'splitter': ['best', 'random']})
```

```
GCV.best_params_ # printing the best parameters found by GridSearchCV
```

```
{'criterion': 'gini',
 'max_depth': 20,
 'min_samples_leaf': 1,
 'splitter': 'random'}
```

```
mod = DecisionTreeClassifier( criterion= 'gini', max_depth= 50, min_samples_leaf= 1, splitter='best')

mod.fit(x_train, y_train)
pred =mod.predict(x_test)
print(accuracy_score(y_test, pred)*100)
```

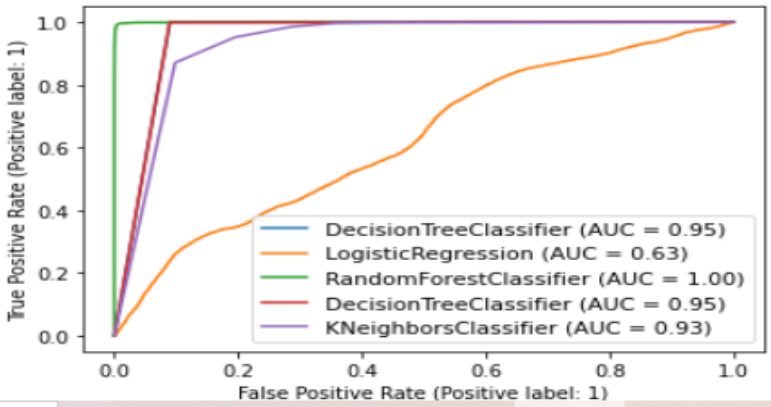
```
95.40452397146794
```

In above figure we can see the default hyperparameter which I was tunned with Decision tree classifier with GCV and then find the best parameters, now fit it in to model and find the accuracy score:

After hyperparameter tuning the accuracy score is raise to: 95.40%

- **AUC ROC Curve:**

```
disp = plot_roc_curve(dt, x_test, y_test)

plot_roc_curve(lr, x_test, y_test, ax=disp.ax_)

plot_roc_curve(rf, x_test, y_test, ax=disp.ax_)

plot_roc_curve(dt, x_test, y_test, ax=disp.ax_)

plot_roc_curve(knn, x_test, y_test, ax=disp.ax_)

plt.legend(prop = {'size':11}, loc='lower right')

plt.show()
```



Now save the model.

Saving and loading the model:

I save this model with name “Malignant1.pkl”

```
model = joblib.load("Malignant1.pkl")
```

```
prediction = model.predict(x_test)
```

Predicted output is:

```
predicted_values.head(20)
```

	Actual	Predicted
264386	1	1
49850	0	0
78885	0	0
87162	0	0
110562	0	0
3876	0	0
260908	1	1
182253	1	1
225691	1	1
19025	0	0
78099	0	0
10444	0	0
207832	1	1
125483	0	0
75374	0	0
253398	1	1
269010	1	1
25984	0	0
193234	1	1
276900	1	1

○ Conclusion:

Hence, at the end, we were effectively able to train our classification model to predict the Malignant project an accuracy score, and have achieved the required task successfully.

○ Learning Outcomes of the Study in respect of Data Science:

Best model is Decision tree classifier, accuracy score and cross validation score is having small difference compare to other algorithms.