PROJECT NAME

MALIGNANT COMMENTS CLASSIFICATION

Submitted by

Aarti A Ambhore

Business Problem Framing:

We are required to model the comments classification malignant/highly-malignant with the available independent variables. This model will then be used by the management to understand to classify the comments based on input.

Conceptual Background of the Domain Problem:

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, but “u are an idiot” is clearly offensive.

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.

- Review of Literature: Now a days balancing an environment on social media platform is extremely important. Describing as “highly malignant

and harmful passing through the medium of electronic text", cyber bullying puts targets under attack from a barrage of degrading, threatening, and/or sexually explicit messages and images conveyed using web sites, instant messaging, blogs, chat rooms, cell phones, web sites, e-mail, and personal online profiles. Thus, the task of finding and removing toxic communication from social media forums is very crucial.

- Motivation for the Problem Undertaken:

This project helps me understand the toxic comments classification problem in social media platform, its customer comments. With the right set of datasets in hand I have built a model that helps the enterprise take the right decision that is whether to focus on a malignant/highly-malignant set of customers. This also motivate learn about text classification problem in social media platform in details. This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes many categories of comments, we can do good amount of data exploration and derive some interesting features using the comments text column available.

We built a model that can differentiate between comments and its categories.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem:
  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.

I have used a Random Forest classifier model to classify the comments in terms malignant/highly malignant and also used cross validation to remove overfitting problem while predicted the correct outcome and validate the model.

- Data sources are provided internally by the enterprise.

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.

Machine Learning Models:

1. Decision Tree Classifier
2. Random Forest Classifier
3. Logistic Regression
4. Gradient Boosting Classifier
5. Ada Boost Classifier
6. KNeighbors Classifier

Out of these models the **Random Forest Classifier** model is the best fit model. Which is showing the accuracy about **95.72%.**

Decision Tree Classifier:

```
In [65]: df = DecisionTreeClassifier()

In [66]: df.fit(x_train,y_train)

Out[66]: DecisionTreeClassifier()
```

```
In [67]: df.score(x_test,y_test)

Out[67]: 0.9408741970860097

In [68]: y_pred = df.predict(x_test)

In [69]: from sklearn.metrics import confusion_matrix,classification_report
         print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.97	0.97	28859
1	0.70	0.67	0.69	3056
accuracy			0.94	31915
macro avg	0.83	0.82	0.83	31915
weighted avg	0.94	0.94	0.94	31915

Random Forest Classifier:

```
In [72]: rf = RandomForestClassifier()

In [73]: rf.fit(x_train,y_train)

Out[73]: RandomForestClassifier()
```

```
In [74]: rf.score(x_test,y_test)

Out[74]: 0.9572301425661914

In [75]: y_pred = rf.predict(x_test)
         print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.99	0.98	28859
1	0.84	0.69	0.75	3056
accuracy			0.96	31915
macro avg	0.90	0.84	0.87	31915
weighted avg	0.95	0.96	0.96	31915

Logistic Regression:

```
In [78]:  1 lm = LogisticRegression()

In [79]:  1 lm.fit(x_train,y_train)

Out[79]: LogisticRegression
         LogisticRegression()

In [80]:  1 lm.score(x_test,y_test)

Out[80]: 0.9564154786150713
```

Gradient Boosting Classifier:

```
In [81]:  1 gb = GradientBoostingClassifier()
          2 gb.fit(x_train,y_train)

Out[81]: GradientBoostingClassifier
         GradientBoostingClassifier()

In [82]:  1 gb.score(x_test,y_test)

Out[82]: 0.9413755287482375

In [83]:  1 y_pred = gb.predict(x_test)
          2 print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       0.94      1.00      0.97     28859
                    1       0.94      0.41      0.57      3056

             accuracy                           0.94     31915
            macro avg       0.94      0.71      0.77     31915
         weighted avg       0.94      0.94      0.93     31915
```

Ada Boost Classifier:

```
In [85]:  1 ad=AdaBoostClassifier()
          2 ad.fit(x_train,y_train)

Out[85]: AdaBoostClassifier
         AdaBoostClassifier()

In [86]:  1 ad.score(x_test,y_test)

Out[86]: 0.9474855083816387

In [87]:  1 y_pred = ad.predict(x_test)
          2 print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       0.95      0.99      0.97     28859
                    1       0.87      0.53      0.66      3056

             accuracy                           0.95     31915
            macro avg       0.91      0.76      0.81     31915
         weighted avg       0.94      0.95      0.94     31915
```

KNeighbors Classifier:

```
In [89]:
1 from sklearn import preprocessing, neighbors
2 clf = neighbors.KNeighborsClassifier()
3 clf.fit(x_train,y_train)
```

```
Out[89]:
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
In [90]:
1 clf.score(x_test,y_test)
```

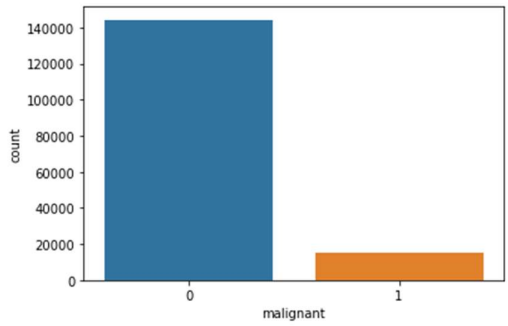
```
Out[90]: 0.9228262572458091
```

Visualization:

Malignant Comment:

```
In [15]:
1 column_name=["malignant","highly_malignant","rude","threat","abuse","loathe"]
2 for i in column_name:
3     print(i)
4     print("\n")
5     print(df_train[i].value_counts())
6     sns.countplot(df_train[i])
7     plt.show()
```

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

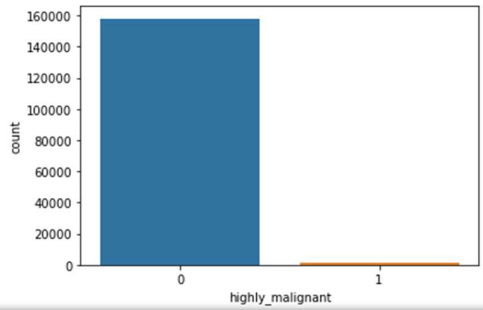


From the above graph we can say that there are 144277 counts for malignant comment.

Highly Malignant Comment:

```
In [15]:
column_name=["malignant","highly_malignant","rude","threat","abuse","loathe"]
for i in column_name:
    print(i)
    print("\n")
    print(df_train[i].value_counts())
    sns.countplot(df_train[i])
    plt.show()
```

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

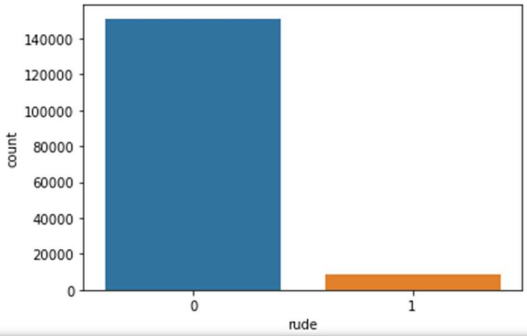


Counts for highly malignant comments are 157976.

Rude Comments:

```
In [15]:
column_name=["malignant","highly_malignant","rude","threat","abuse","loathe"]
for i in column_name:
    print(i)
    print("\n")
    print(df_train[i].value_counts())
    sns.countplot(df_train[i])
    plt.show()
```

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

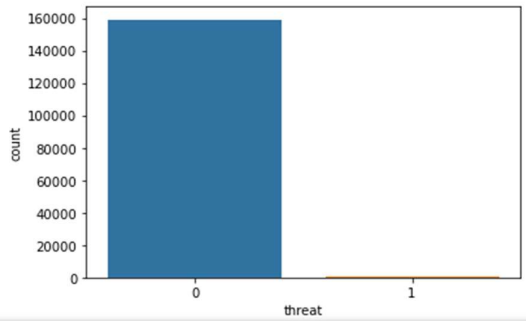


There are 151122 counts for the rude comments.

Threat Comments:

```
In [15]:
column_name=["malignant","highly_malignant","rude","threat","abuse","loathe"]
for i in column_name:
    print(i)
    print("\n")
    print(df_train[i].value_counts())
    sns.countplot(df_train[i])
    plt.show()
```

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

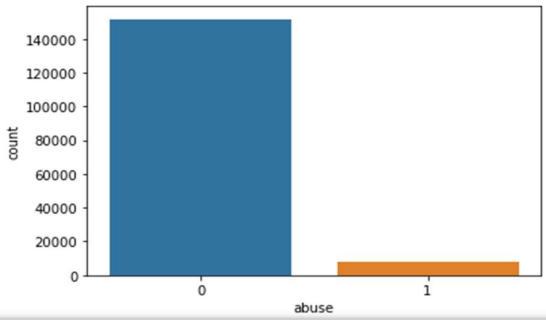


From the graph we say that there are 159093 counts for threat comments.

Abuse Comments:

```
In [15]:
column_name=["malignant","highly_malignant","rude","threat","abuse","loathe"]
for i in column_name:
    print(i)
    print("\n")
    print(df_train[i].value_counts())
    sns.countplot(df_train[i])
    plt.show()
```

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

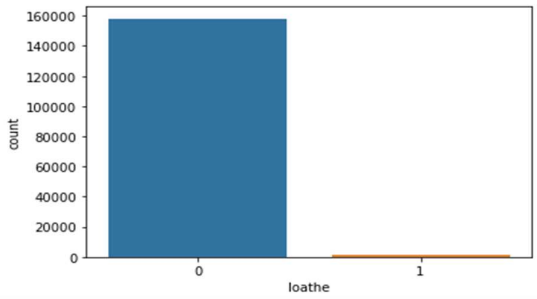


There are 151694 counts for the abuse comments.

Loathe Comments:

```
In [15]:
column_name=["malignant","highly_malignant","rude","threat","abuse","loathe"]
for i in column_name:
    print(i)
    print("\n")
    print(df_train[i].value_counts())
    sns.countplot(df_train[i])
    plt.show()
```

```
0    158166
1      1405
Name: loathe, dtype: int64
```



From the above graph we can say that there are 158166 counts for the loathe comments.

From the all above count plots we can say that there is high count for the Threat Comments followed by Loathe comments.

Interpretation of the Results:

- Random Forest Classifier model predict the best result for the given dataset.
- The accuracy of this model is 95.72%.
- The confusion matrix and classification report for the given model is given below:

```
In [75]:
1 y_pred = rf.predict(x_test)
2 print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     28859
           1       0.84      0.69      0.75      3056

    accuracy                           0.96     31915
   macro avg       0.90      0.84      0.87     31915
weighted avg       0.95      0.96      0.96     31915
```

```
In [76]:
1 cm=confusion_matrix(y_test,df.predict(x_test))
```

```
In [77]:
1 print(cm)
```

```
[[27970   889]
 [  998  2058]]
```

Conclusion:

Key Findings and Conclusions of the Study:

- I used various classification methods and out of all machine learning algorithm used, Random Forest classifier yields the best results.
- These malignant comments classification can be used by social media companies to filter and classify some keywords as highly malignant and set their own policy going forward for the customers and other shareholders.

THE END