

A
Report On
Malignant Comment
Classification

Submitted By:
Rohit Kachhal

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzation skills.

A huge thanks to my academic team “Data trained” who are the reason behind what I am doing today.

INTRODUCTION

- **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.

- **Conceptual Background of the Domain Problem**

- ❖ 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 be used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

- Review of Literature

- ❖ Aggression by text is a complex phenomenon, and different knowledge fields try to study and tackle this problem. In this study, several related literatures are used to express different types of aggression. Some of those are hate, cyberbullying, abusive language, malignant, flaming, threatening, extremism, radicalization and hate speech. This research found a few dedicated works that addresses the effect of incorporating different text transformations on the model accuracy for sentiment classification. In this work, we performed a systematic review of the state-of-the-art in malignant comment classification using machine learning methods with NLP test processing. In our analysis of every primary study, we investigated data set used, evaluation metric, used machine learning methods, classes of malignant and non-malignant and comment language.

- Motivation for the Problem Undertaken

- ❖ The main objective of this study is to build a machine learning model that performed well on this dataset. We have several type of classification model through which based on comment we can easily find the either the comment are good or bad.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

For the given problem I have two datasets. One dataset to train the model and another dataset to test the model. The train data set contains 8 columns and test dataset contains 2 columns. 'id' column is present in both dataset and is not useful for model so let's drop it. After reading this dataset, I can say that I have multiclass binary classification problem. According to this train datasets target variables are 'malignant' , 'highly malignant' , 'rude' , 'threat' , 'abuse' , 'loathe'. First of all, I added new column with name 'label' in train dataset which contains the column wise sum. After that I get 6 different label in 'label' column. Then, by processing convert it in to label with either 0 or 1, where 0 denotes NO and 1 denotes Yes. Then I add comment length in dataset.

I used different visualization plot to see the trend of data like countplot, distplot to see the trend, distribution, skewness and outliers in dataset, heatmap to see the multicollinearity. From these plots, I can observe skewness but in target variable, so no need to worry about it. After that, data cleaning also performed and clean comment length added into dataset. I have also done hyperparameter tuning on final model to improve performance of model. After that, I saved the model and compared the performance of model.

- Data Sources and their formats

❖ DataFrame

| | 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 | 000103f0d9cfb80f | 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 | 0001b41b1c8bb37e | "\nMore!\nI can't make any real suggestions on ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0001d958c54c8e35 | You, sir, are my hero. Any chance you remember... | 0 | 0 | 0 | 0 | 0 | 0 |

The info of train dataframe is :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   id                     159571 non-null object
1   comment_text           159571 non-null object
2   malignant              159571 non-null int64
3   highly_malignant       159571 non-null int64
4   rude                   159571 non-null int64
5   threat                 159571 non-null int64
6   abuse                  159571 non-null int64
7   loathe                 159571 non-null int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

| | id | comment_text |
|---|------------------|---|
| 0 | 00001cee341fdb12 | Yo bitch Ja Rule is more succesful then you'll... |
| 1 | 0000247887823ef7 | == From RfC == \n\n The title is fine as it is... |
| 2 | 00013b17ad220c46 | " \n\n == Sources == \n\n * Zawe Ashton on Lap... |
| 3 | 00017583c3f7919a | :if you have a look back at the source, the in... |
| 4 | 00017895ad8997eb | I don't anonymously edit articles at all. |

The info of test dataframe is :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 153164 entries, 0 to 153163
Data columns (total 2 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   id                     153164 non-null object
1   comment_text           153164 non-null object
dtypes: object(2)
memory usage: 2.3+ MB
```

❖ Descriptive Statistical Analysis: Train Dataset

| | malignant | highly_malignant | rude | threat | abuse | loathe |
|-------|---------------|------------------|---------------|---------------|---------------|---------------|
| count | 159571.000000 | 159571.000000 | 159571.000000 | 159571.000000 | 159571.000000 | 159571.000000 |
| mean | 0.095844 | 0.009998 | 0.052948 | 0.002998 | 0.049364 | 0.008805 |
| std | 0.294379 | 0.099477 | 0.223931 | 0.054650 | 0.218627 | 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 |



Outcome:

- ✓ Analysis describes that some skewness is present in the target variable.

• Data Pre-processing Done

- ❖ In both dataset 'Id' is unnecessary column so need to drop it.
- ❖ Datasets do not contain null values.
- ❖ Created new column with name 'label' in train dataset which contains the column wise sum of target variable. After that I get 6 different label in 'label' column. Then, by processing convert it in to label with either 0 or 1, where 0 denotes NO and 1 denotes Yes.
- ❖ Comment length column also added in both datasets.
- ❖ Text pre-processing technique like convert all text in lower case, replace '\n' by ' ', cleaning the dataset like removing punctuation and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization.
- ❖ After that added a new column in dataset with name 'clean_length' which contains the clean comment length.
- ❖ Skewness is present in target variable so no need to deal with it.
- ❖ After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by

comparing the frequency of appearance of a word in a document with the number of documents the words appear in.

- Data Inputs- Logic- Output Relationships

- ❖ I used count plot to visualize the trend.
- ❖ All data now are of either object or categorical.
- ❖ I have checked the correlation between the target and features using heat map .

- Hardware and Software Requirements and Tools Used

- ❖ Hardware:

- Processor: I3 or above
- RAM: 4 GB or above
- Storage: 250 GB or above

- ❖ Software:

- Anaconda
- Jupyter notebook etc

- ❖ Libraries:

- numpy
- pandas
- matplotlib
- seaborn
- model_selection: train_test_split, cross_val_score, GridSearchCV
- score: Accuracy score, recall, precision, confusion metrics, AUC-ROC

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

First of all, I dropped the unnecessary and unwanted columns from the data frame by using **drop ()**. After that, a new column is added with name 'label' in train dataset which contains the column wise sum of target variable using **sum(axis=1)**. I get 6 different label in 'label' column. Then, by processing convert it in to label with either 0 or 1, where 0 denotes NO and 1 denotes Yes. After that, text pre-processing and cleaning the data like converting the text into lower case by using **str.lower()**, replacing the '\n' by using **str.replace()**, removing punctuation and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization. After that, it is observed skewness is present only in target. After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in. Finally, using **train_test_split** split the dataset and Initiate the model.

- Testing of Identified Approaches (Algorithms)

As the problem is regression problem. In machine learning there are several algorithms for regression problem. I used total 7 algorithm to build best model on this dataset. The algorithms are:

- ❖ Logistic Regression
- ❖ K-Neighbors Classifier
- ❖ Decision Tree Classifier
- ❖ Bagging Classifier
- ❖ Gradient Boosting Classifier
- ❖ AdaBoost Classifier
- ❖ XGB Classifier

- Run and Evaluate selected models
Here are the coding and insights of all models:

❖ Logistic Regression

```
lg = LogisticRegression()
lg.fit(X_train,y_train)
lg_pred = lg.predict(X_test)
lg_accuracy = accuracy_score(y_test,lg_pred)
lg_cf = classification_report(y_test,lg_pred)
lg_cm = confusion_matrix(y_test,lg_pred)
lg_train_score = lg.score(X_train,y_train)
lg_test_score = lg.score(X_test,y_test)

print('Logistic Regression')
print('-----\n')
print('The Score on train set is :',lg_train_score)
print('The Score on test set is :',lg_test_score)
print('The Accuracy on test set is :',lg_accuracy)
print('The Classification report is :\n',lg_cf)
print('The Confusion matrix is :\n',lg_cm)
print('-----')
```

```
Logistic Regression
-----
The Score on train set is : 0.897259599459261
The Score on test set is : 0.9009650735294118
The Accuracy on test set is : 0.9009650735294118
The Classification report is :
      precision    recall  f1-score   support

     0       0.90      1.00      0.95      43128
     1       1.00      0.00      0.00       4744

   accuracy          0.90      0.90      0.90      47872
  macro avg       0.95      0.50      0.47      47872
 weighted avg       0.91      0.90      0.85      47872

The Confusion matrix is :
[[43128   0]
 [ 4741   3]]
-----
```

Cross Validation on Logistic Regression

```
lg_cv_score = cross_val_score(lg,X_train,y_train,cv=5)
print('The cross validation score is :',lg_cv_score.mean())

The cross validation score is : 0.8972148363483289
```

❖ K-Neighbors Classifier

```
kc = KNeighborsClassifier()
kc.fit(X_train,y_train)
kc_pred = kc.predict(X_test)
kc_accuracy = accuracy_score(y_test,kc_pred)
kc_cf = classification_report(y_test,kc_pred)
kc_cm = confusion_matrix(y_test,kc_pred)
kc_train_score = kc.score(X_train,y_train)
kc_test_score = kc.score(X_test,y_test)

print('K-Neighbors Classifier')
print('-----\n')
print('The Score on train set is :',kc_train_score)
print('The Score on test set is :',kc_test_score)
print('The Accuracy on test set is :',kc_accuracy)
print('The Classification report is :\n',kc_cf)
print('The Confusion matrix is :\n',kc_cm)
print('-----')
```

```
K-Neighbors Classifier
-----
The Score on train set is : 0.8972864573541393
The Score on test set is : 0.9009650735294118
The Accuracy on test set is : 0.9009650735294118
The Classification report is :
      precision    recall  f1-score   support

     0       0.90      1.00      0.95      43128
     1       1.00      0.00      0.00       4744

   accuracy          0.90      0.90      0.90      47872
  macro avg       0.95      0.50      0.47      47872
 weighted avg       0.91      0.90      0.85      47872

The Confusion matrix is :
[[43128   0]
 [ 4741   3]]
-----
```

Cross validation on K-Neighbors Classifier

```
kc_cv_score = cross_val_score(kc,X_train,y_train,cv=5)
print('The cross validation score is :',kc_cv_score.mean())

The cross validation score is : 0.8972595991057147
```

❖ Decision Tree Classifier

```
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
dt_pred = dt.predict(X_test)
dt_accuracy = accuracy_score(y_test,dt_pred)
dt_cf = classification_report(y_test,dt_pred)
dt_cm = confusion_matrix(y_test,dt_pred)
dt_train_score = dt.score(X_train,y_train)
dt_test_score = dt.score(X_test,y_test)

print('Decision Tree Classifier')
print('-----\n')
print('The Score on train set is :',dt_train_score)
print('The Score on test set is :',dt_test_score)
print('The Accuracy on test set is :',dt_accuracy)
print('The Classification report is :\n',dt_cf)
print('The Confusion matrix is :\n',dt_cm)
print('\n-----')
```

Decision Tree Classifier

```
The Score on train set is : 0.9056213573980072
The Score on test set is : 0.901800635026738
The Accuracy on test set is : 0.901800635026738
The Classification report is :
      precision    recall  f1-score   support

     0       0.90       1.00       0.95       43128
     1       0.85       0.01       0.02        4744

 accuracy          0.90       0.90       0.90       47872
 macro avg         0.88       0.51       0.48       47872
 weighted avg      0.90       0.90       0.86       47872
```

The Confusion matrix is :
[[43119 9]
[4692 52]]

Cross validation on Decision Tree Classifier

```
dt_cv_score = cross_val_score(dt,X_train,y_train,cv=5)
print('The cross validation score is :',dt_cv_score.mean())
```

The cross validation score is : 0.8978862821174637

❖ Bagging Classifier

```
bc = BaggingClassifier()
bc.fit(X_train,y_train)
bc_pred = bc.predict(X_test)
bc_accuracy = accuracy_score(y_test,bc_pred)
bc_cf = classification_report(y_test,bc_pred)
bc_cm = confusion_matrix(y_test,bc_pred)
bc_train_score = bc.score(X_train,y_train)
bc_test_score = bc.score(X_test,y_test)

print('Bagging Classifier')
print('-----\n')
print('The Score on train set is :',bc_train_score)
print('The Score on test set is :',bc_test_score)
print('The Accuracy on test set is :',bc_accuracy)
print('The Classification report is :\n',bc_cf)
print('The Confusion matrix is :\n',bc_cm)
print('\n-----')
```

Bagging Classifier

```
The Score on train set is : 0.9047260942354005
The Score on test set is : 0.9017170788770054
The Accuracy on test set is : 0.9017170788770054
The Classification report is :
      precision    recall  f1-score   support

     0       0.90       1.00       0.95       43128
     1       0.84       0.01       0.02        4744

 accuracy          0.90       0.90       0.90       47872
 macro avg         0.87       0.50       0.48       47872
 weighted avg      0.90       0.90       0.86       47872
```

The Confusion matrix is :
[[43119 9]
[4696 48]]

Cross validation on Bagging Classifier

```
bc_cv_score = cross_val_score(bc,X_train,y_train,cv=5)
print('The cross validation score is :',bc_cv_score.mean())
```

The cross validation score is : 0.89781466050337

❖ Gradient Boosting Classifier

```
gc = GradientBoostingClassifier()
gc.fit(X_train,y_train)
gc_pred = gc.predict(X_test)
gc_accuracy = accuracy_score(y_test,gc_pred)
gc_cf = classification_report(y_test,gc_pred)
gc_cm = confusion_matrix(y_test,gc_pred)
gc_train_score = gc.score(X_train,y_train)
gc_test_score = gc.score(X_test,y_test)

print('Gradient Boosting Classifier')
print('-----\n')
print('The Score on train set is :',gc_train_score)
print('The Score on test set is :',gc_test_score)
print('The Accuracy on test set is :',gc_accuracy)
print('The Classification report is :\n',gc_cf)
print('The Confusion matrix is :\n',gc_cm)
print('\n-----')
```

Gradient Boosting Classifier

The Score on train set is : 0.8972864573541393
The Score on test set is : 0.9009650735294118
The Accuracy on test set is : 0.9009650735294118
The Classification report is :
precision recall f1-score support

| | | | | |
|--------------|------|------|------|-------|
| 0 | 0.90 | 1.00 | 0.95 | 43128 |
| 1 | 1.00 | 0.00 | 0.00 | 4744 |
| accuracy | | | 0.90 | 47872 |
| macro avg | 0.95 | 0.50 | 0.47 | 47872 |
| weighted avg | 0.91 | 0.90 | 0.85 | 47872 |

The Confusion matrix is :
[[43128 0]
[4741 3]]

Cross validation on Gradient Boosting Classifier

```
gc_cv_score = cross_val_score(gc,X_train,y_train,cv=5)
print('The cross validation score is :',gc_cv_score.mean())
```

The cross validation score is : 0.8972595991057147

❖ AdaBoost Classifier

```
ac = AdaBoostClassifier()
ac.fit(X_train,y_train)
ac_pred = ac.predict(X_test)
ac_accuracy = accuracy_score(y_test,ac_pred)
ac_cf = classification_report(y_test,ac_pred)
ac_cm = confusion_matrix(y_test,ac_pred)
ac_train_score = ac.score(X_train,y_train)
ac_test_score = ac.score(X_test,y_test)

print('AdaBoost Classifier')
print('-----\n')
print('The Score on train set is :',ac_train_score)
print('The Score on test set is :',ac_test_score)
print('The Accuracy on test set is :',ac_accuracy)
print('The Classification report is :\n',ac_cf)
print('The Confusion matrix is :\n',ac_cm)
print('\n-----')
```

AdaBoost Classifier

The Score on train set is : 0.8981011468321113
The Score on test set is : 0.9010486296791443
The Accuracy on test set is : 0.9010486296791443
The Classification report is :
precision recall f1-score support

| | | | | |
|--------------|------|------|------|-------|
| 0 | 0.90 | 1.00 | 0.95 | 43128 |
| 1 | 1.00 | 0.00 | 0.00 | 4744 |
| accuracy | | | 0.90 | 47872 |
| macro avg | 0.95 | 0.50 | 0.48 | 47872 |
| weighted avg | 0.91 | 0.90 | 0.85 | 47872 |

The Confusion matrix is :
[[43128 0]
[4737 7]]

Cross validation on AdaBoost Classifier

```
ac_cv_score = cross_val_score(ac,X_train,y_train,cv=5)
print('The cross validation score is :',ac_cv_score.mean())
```

The cross validation score is : 0.8972595991057147

❖ XGB Classifier

```
xc = xgb.XGBClassifier()
xc.fit(X_train,y_train)
xc_pred = xc.predict(X_test)
xc_accuracy = accuracy_score(y_test,xc_pred)
xc_cf = classification_report(y_test,xc_pred)
xc_cm = confusion_matrix(y_test,xc_pred)
xc_train_score = xc.score(X_train,y_train)
xc_test_score = xc.score(X_test,y_test)

print('XGB Classifier')
print('-----\n')
print('The Score on train set is :',xc_train_score)
print('The Score on test set is :',xc_test_score)
print('The Accuracy on test set is :',xc_accuracy)
print('The Classification report is :\n',xc_cf)
print('The Confusion matrix is :\n',xc_cm)
print('-----\n')
```

```
XGB Classifier
-----

The Score on train set is : 0.897259599459261
The Score on test set is : 0.9009650735294118
The Accuracy on test set is : 0.9009650735294118
The Classification report is :
      precision    recall  f1-score   support

      0       0.90      1.00      0.95      43128
      1       1.00      0.00      0.00        4744

 accuracy          0.95
 macro avg          0.95      0.50      0.47      47872
weighted avg          0.91      0.90      0.85      47872

The Confusion matrix is :
[[43128   0]
 [ 4741   3]]
```

Cross validation on XGB Classifier

```
xc_cv_score = cross_val_score(xc,X_train,y_train,cv=5)
print('The cross validation score is :',xc_cv_score.mean())
```

The cross validation score is : 0.8972148363483289

- Key Metrics for success in solving problem under consideration

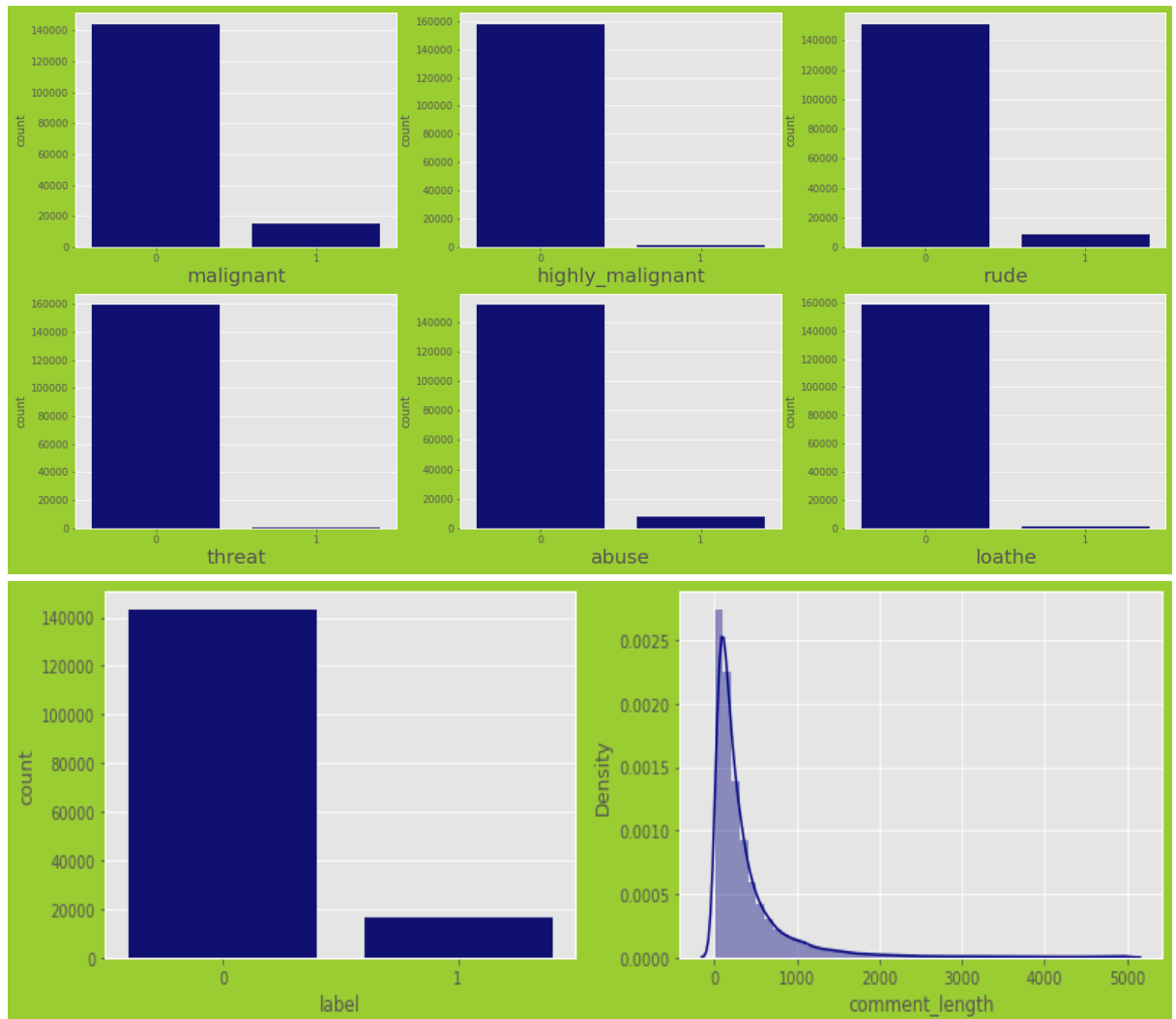
Following are the metrics to observe good model:

- ❖ Accuracy simply measures how often the classifier correctly predicts. Accuracy can be defined as the ratio of the number of correct predictions and the total number of predictions.
- ❖ Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives.
- ❖ Recall explains how many of the actual positive cases we were able to predict correctly with our model. It is a useful metric in cases where False Negative is of higher concern than False Positive.
- ❖ It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall.

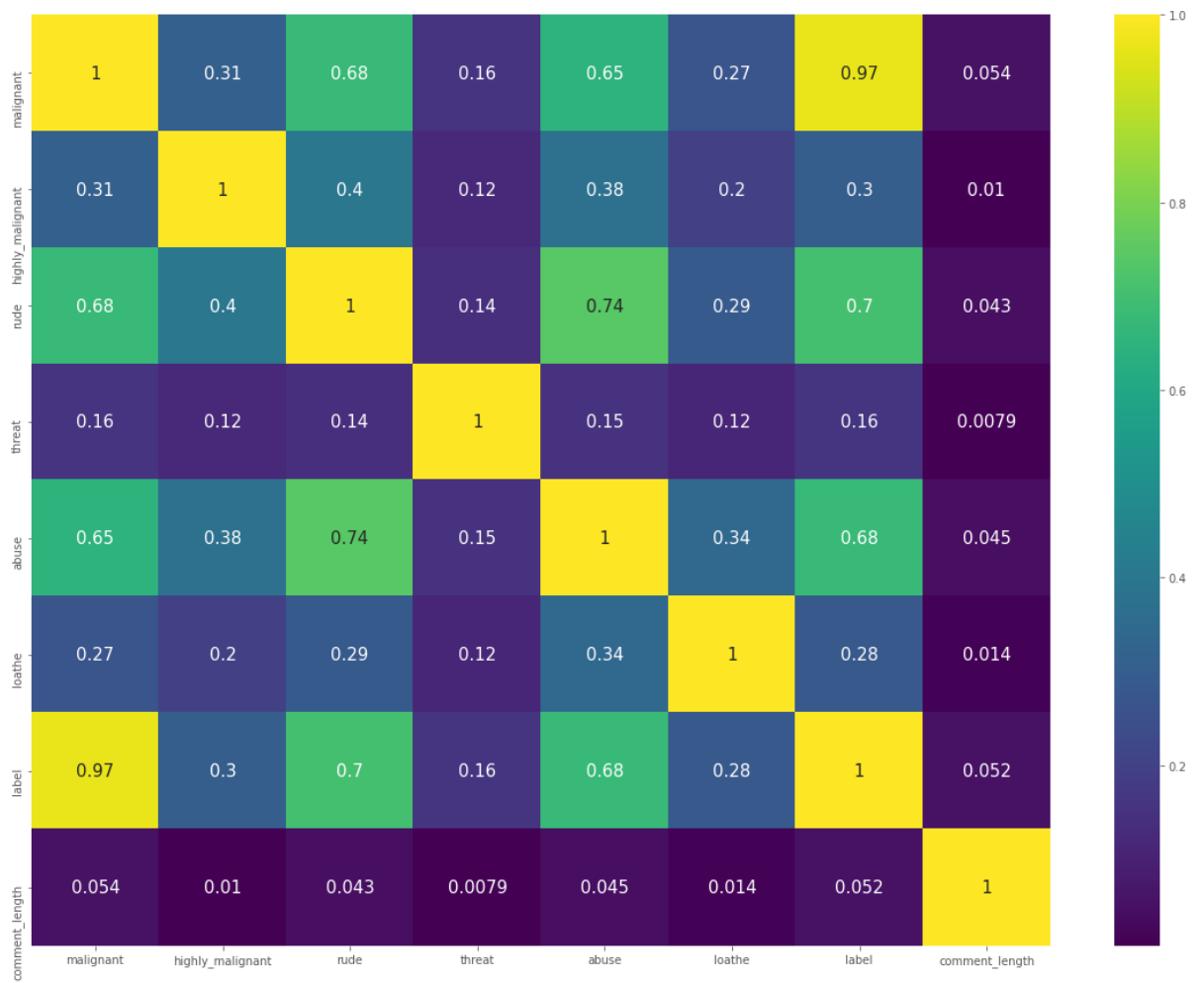
- ❖ **Cross_val_score:** Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.
- ❖ **AUC-ROC:** The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'.
- ❖ **AUC** is the measure of the ability of a classifier to distinguish between classes.

- Visualizations

- ❖ Univariate



❖ Correlation matrix



- Interpretation of the Results

- ❖ Score of all model

| | Model Name | Train Score | Test Score | Accuracy Score | Cross Validation Score |
|---|------------------------------|-------------|------------|----------------|------------------------|
| 0 | Logistic Regression | 0.897280 | 0.900965 | 0.900965 | 0.897215 |
| 1 | KNeighbors Classifier | 0.897286 | 0.900965 | 0.900965 | 0.897260 |
| 2 | Decision Tree Classifier | 0.905821 | 0.901801 | 0.901801 | 0.897886 |
| 3 | Bagging Classifier | 0.904726 | 0.901717 | 0.901717 | 0.897815 |
| 4 | Gradient Boosting Classifier | 0.897286 | 0.900965 | 0.900965 | 0.897260 |
| 5 | AdaBoost Classifier | 0.898101 | 0.901049 | 0.901049 | 0.897260 |
| 6 | XGB Classifier | 0.897280 | 0.900965 | 0.900965 | 0.897215 |

- 🏆 Outcome:

XGB Classifier and Decision Tree Classifier models are good. Based on accuracy Decision Tree Classifier has better accuracy. The difference between accuracy and cross validation score, XGB Classifier has less difference. So, I am choosing XGB Classifier as model on this dataset because it is performing well on this dataset.

❖ Hyperparameter Tuning

```
clf = xb.XGBClassifier()

param = {'learning_rate' : np.arange(0.01,0.8),
        'max_depth' : range(2,30),
        'subsample' : np.arange(0.1,1)
        }

grd = GridSearchCV(clf,param_grid=param)
grd.fit(X_train,y_train)

clf = grd.best_estimator_

clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)

ac_con = confusion_matrix(y_test,y_pred)

print("\n Best parameter",grd.best_params_)
print("\n Confusion Matrix \n",ac_con)
print("\n Accuracy after hyper parameter tuning",accuracy_score(y_test,y_pred))
```

Best parameter {'learning_rate': 0.01, 'max_depth': 2, 'subsample': 0.1}

Confusion Matrix
[[43128 0]
[4744 0]]

Accuracy after hyper parameter tuning 0.9009024064171123

📌 Outcome:

- ✓ After hyperparameter tuning, I am not able to improve the performance of this model. So, default parameter gives better result.

CONCLUSION

- Key Findings and Conclusions of the Study
 - ❖ From the above dataset study and analysis, I can conclude that every comment has a possibility of either malignant comment or normal comment.
 - ❖ With the popularity of social media or social channel, peoples used to comment on services provided by platform. These comments are either malignant comment or normal comment.
 - ❖ I found positive comments are higher in compare to negative comments. It's also observed that in real word malignant comments are harmful and hateful sometime for service provider.
- Findings are :
 - In training dataset, we have only 10% of data which is spreading hate on social media.
 - In this 10%, data most of the comments are malignant, rude or abuse.
 - Some of the comments are very long while some are very short.
- Learning Outcomes of the Study in respect of Data Science
 - ❖ Project is very interesting as it is different from other. I have learnt many new things and gain idea about new techniques and also came to know about the method to deal with uncleaned text data. I have also learnt how to deal with multiclass target features. Tools used for visualizations give a better understanding of dataset. As well-known variety of machine learning algorithms are available that can deal with the binary classification problem. After working on this dataset I can conclude that XGB Classifier gives best result in compared to other models.

- ❖ It is possible to classify the comments content into the required categories of authentic and however, using this kind of project an awareness can be created to know what is fake and authentic.
- Limitations of this work and Scope for Future Work
 - ❖ This project was amazing to work on, it creates new ideas to think about but there were some limitations in this project like unbalanced dataset. Every effort has been put on it for perfection but nothing is perfect and this project is of no exception. There are certain areas which can be enhanced.
 - ❖ In future work, we can focus on performance and error analysis of the model as lots of comments are misclassified into the hate category. Previous work has achieved success using various algorithms on data in English language but in future, we can consider having data in regional languages. We can also work on after work of the detection of the malignant comments like automatic blocking of the user, auto-deletion of the malignant comments like automatic blocking of the user, auto-deletion of harmful comments on social media platforms. Comment detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field.