Malignant-Comments-Classifier

Submitted by:

Vijay Ashley Rodrigues K.

ACKNOWLEDGMENT

- I would like to thank FlipRobo Technologies for providing me this opportunity and guidance throughout the project and all the steps that are implemented.
- I have primarily referred to various articles scattered across various websites for the purpose of getting an idea on Housing related in general.
- I would like to thank the technical support team also for helping me out and reaching out to me on clearing all my doubts as early as possible
- I would like to thank my project SME Keshav Bansal for providing the flexibility in time and also for giving us guidance in creating the project.
- The following are some of the articles I referred to in this project.

- https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6837517.pdf
- https://pubmed.ncbi.nlm.nih.gov/8640897/

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.

  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.

- ## Conceptual Background of the Domain Problem

  - 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.
- 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.
- You need to build a model that can differentiate between comments and its categories.
-

- Motivation for the Problem Undertaken

    - It's highly common to receive feedback or comments for articles, products or posts of any kind these days.
    - Keeping track of what kind of comments we receive, and how to filter it based on the situation is important as we can portray how we look virtually based on the comments.

Analytical Problem Framing

- Data Sources and their formats

- The dataset is provided by FlipRobo and is available for academic purpose only and not for any kind of commercial activities.
- The dataset contains the Housing related data with 159571 records (rows) and 8 features (columns).
- The file is in .csv format and we have 2 files. One for train data and another test data.
- The dataset is in both numerical and categorical data.

	A	B	C	D	E	F	G	H
1	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
2	0000997932d777bf	Explanation	0	0	0	0	0	0
3	000103f0d9cfb60f	D'aww! He matches	0	0	0	0	0	0
4	000113f07ec002fd	Hey man, I'm really r	0	0	0	0	0	0
5	0001b41b1c6bb37e	"	0	0	0	0	0	0
6	0001d958c54c6e35	You, sir, are my hero	0	0	0	0	0	0

- Data Pre-processing Done

1. Acquire the dataset

- We have received this dataset from FlipRobo Technologies which is related to a Malignant Classification of comments on social media.
- The sample data is provided to us from our client database. It is hereby given for this exercise.

2. Import all the crucial libraries

- For this project I have used the following major libraries like Pandas-profiling, Pandas, Matplotlib and Seaborn that are used for EDA or any other pre-processing done in this scenario.

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier

from sklearn.model_selection import train_test_split

from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

3. Import the dataset

- The dataset is in CSV format and it is imported using Pandas library in Jupyter Notebook.

```
# train dataset

df_train = pd.read_csv("D:/Malignant Comments Classifier Project/train.csv")
df_train.head()
```

- The statement **pd. set_option("display.max_columns", None)** simply allows us to physically see all the feature columns.
- The statement **pd. set_option("display.max_rows", None)** simply allows us to physically see all the feature rows.
- By default, Jupyter Notebook doesn't display all the rows and columns at the same time and only selected portion from starting and ending of dataset are displayed.

4. Identifying and handling the missing values

- There are no missing values hence no actions have been taken

```
<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
```

```
: df_train.isnull().sum()

: id                0
  comment_text      0
  malignant         0
  highly_malignant  0
  rude              0
  threat            0
  abuse             0
  loathe            0
  dtype: int64
```

5. Encoding the categorical data

- I have used LabelEncoder as the data is categorical and is not ordinal in nature.

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

# Encode the training dataset

df_train.comment_text = encoder.fit_transform(df_train.comment_text)


# Encode the test dataset

df_test.comment_text = encoder.fit_transform(df_test.comment_text)
```

6. Splitting the dataset

- Splitting up of dataset between x (features) and y (target column)

Since the target values have 2 features of same data type, col "y" is split in the following manner

```
]: # train dataset with featurs only
x = df_train.drop(columns = ["malignant", "highly_malignant", "rude", "threat", "abuse", "loathe"], axis=1)

y = df_train[["malignant", "highly_malignant", "rude", "threat", "abuse", "loathe"]]

# test dataset with featurs only
x1 = df_test
```

- Split the dataset into train and test data set.
- I have chosen 200 random state and 30% of data is divided in text dataset.

I have chosed 200 random state and 30% of data is divided in text dataset

```
]: x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.30, random_state = 200)
```

7. Feature scaling

- Finding variance inflation factor in each scaled column
- This gives us a relationship between feature vs feature and we can drop it if necessary to avoid multicollinearity.
- From the below observation, I have not considered this step.
- Let's now Scale the data for further processing.
- we have used StandardScaler for further scaling up of data.

```
: from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)

x_scaled1 = scaler.fit_transform(x1)
```

- State the set of assumptions (if any) related to the problem under consideration

    - Perform proper data cleaning and text exploration.
    - Focus on data visualization and infer details from it.
    - Try to infer unique patterns from the data and try to generate new features.
    - Use 4-5 models for training, do proper hyperparameter tuning and choose the right evaluation metrics to finalize your model.
    - Test your predictions on multiple metrics like log loss, Recall and Precision.
    - Create a detailed report mentioning all the steps in the format of a sample documentation file.
    - Create a powerpoint presentation of the project.

- Hardware and Software Requirements and Tools Used

    Hardware / Software specifications

    Windows 10 64bit

    Anaconda 2021.05

    Python version – Python 3.9.5

    Jupyter Notebook 6.4 and Google Colab

    Pandas-profiling – package that performs simple EDAs for distribution of variables. This helped me give basic details about what the dataset consists of and its correlations with each other. The report is displayed using **.to_widgets()**

Pandas – This is used in the data manipulation, processing and cleaning and also to get description, stats and almost everywhere in the project.

Matplotlib and **Seaborn** Majority of the data visualizations are plotted using Seaborn and to some extent only Matplotlib is used here.

LabelEncoder I have used this **Skipy** library to convert all the non-ordered categorical data into numerical data. In our case it's only one feature "pcircle".

train_test_split module from **sklearn.model_selection** to split the data into train and test and then used **StandardScaler** to bring the values to one level before imputing to model.

Warnings: I have used "ignore" to avoid the general errs that may occur and used "FutureWarning" to avoid errors that I got when running algorithms on Google Colab. To have a generic and efficient notebook file I used this as well.

Sciypy Used xgboost to import XGBClassifier and remaining algorithms including ensemble are part of Scipy.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

For the model building I have considered the following 6 algorithms for the training and testing.

KNeighborsClassifier
RandomForestClassifier
DecisionTreeClassifier
ExtraTreesClassifier

- Run and Evaluate selected models

  - I have used a total of 4 machine learning algorithms to find the best and suited model which also includes ensemble algorithms.
  - I have considered Accuracy Score for model evaluation.
  - But we cannot simply rely on these scores as we cannot have any scope for assumption. Hence post this I have also performed Cross Validation

for all these algorithms to find the estimated performance metric when it's actually used in production.

- I have not used AUC ROC as it was not asked in the problem statement.

1) KNeighborsClassifier

- From the below algorithm we can see the accuracy score for KNeighborsClassifier is approximately **88.79%**.

```
from sklearn.neighbors import KNeighborsClassifier

k_neigh = KNeighborsClassifier()
k_neigh.fit(x_train,y_train)

y_pred = k_neigh.predict(x_test)

print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.8879094251336899
              precision    recall  f1-score   support

           0       0.56      0.22      0.32      4598
           1       0.41      0.14      0.20       465
           2       0.58      0.20      0.29      2491
           3       0.14      0.01      0.01       144
           4       0.54      0.19      0.28      2305
           5       0.27      0.02      0.03       420

   micro avg       0.55      0.19      0.29     10423
   macro avg       0.42      0.13      0.19     10423
weighted avg       0.53      0.19      0.28     10423
 samples avg       0.02      0.02      0.02     10423
```

2) RandomForestClassifier

- From the below algorithm we can see the accuracy score for RandomForestClassifier is approximately **83.88%.**

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(x_train,y_train)

y_pred = rf.predict(x_test)

print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.8388201871657754
              precision    recall  f1-score   support

           0       0.33      0.32      0.32      4598
           1       0.20      0.21      0.21       465
           2       0.28      0.29      0.28      2491
           3       0.11      0.12      0.11       144
           4       0.26      0.27      0.27      2305
           5       0.07      0.07      0.07       420

   micro avg       0.28      0.28      0.28     10423
   macro avg       0.21      0.21      0.21     10423
weighted avg       0.28      0.28      0.28     10423
 samples avg       0.03      0.03      0.02     10423
```

3) DecisionTreeClassifier

- From the below algorithm we can see the accuracy score for DecisionTreeClassifier is approximately **83.86%.**

```
: from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)

y_pred = dt.predict(x_test)

print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.8386112967914439
              precision    recall  f1-score   support

           0       0.32      0.32      0.32      4598
           1       0.20      0.21      0.21       465
           2       0.28      0.29      0.28      2491
           3       0.11      0.12      0.11       144
           4       0.26      0.27      0.27      2305
           5       0.07      0.07      0.07       420

   micro avg       0.28      0.28      0.28     10423
   macro avg       0.21      0.21      0.21     10423
weighted avg       0.28      0.28      0.28     10423
 samples avg       0.03      0.03      0.02     10423
```

4) ExtraTreesClassifier

- From the below algorithm we can see the accuracy score for ExtraTreesClassifier is approximately **84.12%**.

```
from sklearn.ensemble import ExtraTreesClassifier

ext_reg = ExtraTreesClassifier()
ext_reg.fit(x_train,y_train)

y_pred = ext_reg.predict(x_test)

print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.8412224264705882
              precision    recall  f1-score   support

           0       0.33      0.31      0.32      4598
           1       0.20      0.20      0.20       465
           2       0.28      0.28      0.28      2491
           3       0.08      0.08      0.08       144
           4       0.27      0.27      0.27      2305
           5       0.07      0.06      0.06       420

   micro avg       0.29      0.28      0.28     10423
   macro avg       0.21      0.20      0.20     10423
weighted avg       0.28      0.28      0.28     10423
 samples avg       0.03      0.03      0.02     10423
```

- The below code shows us the cross validation performed over all the algorithms and I have used the CV values as 5.
- If you observe the below screenshot, we get the difference in the values.

```
: from sklearn.model_selection import cross_val_score

: scr = cross_val_score(k_neigh, x, y, cv=5)
  print("Cross Validation score of KNeighborsClassifier model is:", scr.mean())

  Cross Validation score of KNeighborsClassifier model is: 0.8875484996195173

: scr = cross_val_score(rf, x, y, cv=5)
  print("Cross Validation score of RandomForestClassifier model is:", scr.mean())

  Cross Validation score of RandomForestClassifier model is: 0.8385232995015166

: scr = cross_val_score(dt, x, y, cv=5)
  print("Cross Validation score of DecisionTreeClassifier model is:", scr.mean())

  Cross Validation score of DecisionTreeClassifier model is: 0.8383227607494531

: scr = cross_val_score(ext_reg, x, y, cv=5)
  print("Cross Validation score of ExtraTreesClassifier model is:", scr.mean())

  Cross Validation score of ExtraTreesClassifier model is: 0.8413935122190315
```

- From the above algorithms ExtraTreesClassifier seems to be an ideal algorithm in this scenario and for this type of dataset.
- The difference between the accuracy score and cross validation for this model is very less compared to other models.

Sr.No	ML Models used	Accuracy Score	Cross Validation Scores	Difference in values
1	KNeighborsClassifier	0.887909425133689	0.887548499619517	0.000360925514171995
2	RandomForestClassifier	0.838820187165775	0.838523299501516	0.000296887664258949
3	DecisionTreeClassifier	0.838611296791443	0.838322760749453	0.000288536041989973
4	ExtraTreesClassifier	0.840136196524064	0.841393512219031	-0.00125731569496701

- Let us try to tune the proposed model (ExtraTreesClassifier) to get better accuracy, if possible.

- The "parameters" have been selected from the skicit library and I have considered 6 parameters.

```
parameters = {"criterion":["gini", "entropy"],
              "max_features":["auto", "sqrt", "log2"],
              "class_weight":["balanced", "balanced_subsample"],
              "oob_score":[True, False],
              "random_state":[30, 50, 70, 100, 120],
              "n_estimators":[100, 130, 150, 170, 200]
              }
```

- RandomizedSearchCV is used to tune the parameters by fitting the same to the training dataset and used the best parameters after selection

```
from sklearn.model_selection import RandomizedSearchCV
RCV = RandomizedSearchCV(ExtraTreesClassifier(), parameters, cv=5, n_iter=10)
```

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

```
RandomizedSearchCV(cv=5, estimator=ExtraTreesClassifier(),
                   param_distributions={'class_weight': ['balanced',
                                                         'balanced_subsample'],
                                        'criterion': ['gini', 'entropy'],
                                        'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'n_estimators': [100, 130, 150, 170,
                                                         200],
                                        'oob_score': [True, False],
                                        'random_state': [30, 50, 70, 100, 120]})
```

```
RCV.best_params_
```

```
{'random_state': 120,
 'oob_score': False,
 'n_estimators': 130,
 'max_features': 'auto',
 'criterion': 'gini',
 'class_weight': 'balanced_subsample'}
```

-

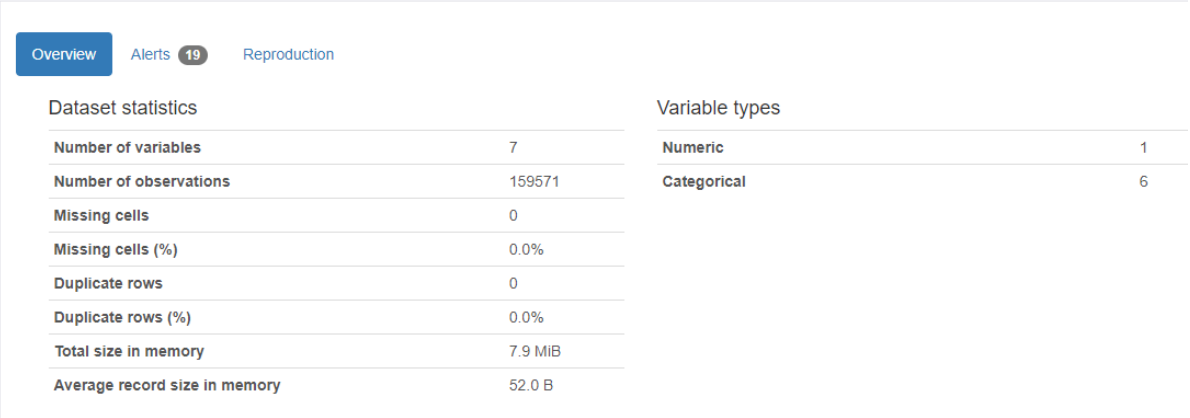
```
mod_final = MultiOutputClassifier(ExtraTreesClassifier(random_state = 120, oob_score = False, n_estimators = 130,
                                                       max_features= "auto", criterion= "gini", class_weight = "balanced_subsampl

mod_final.fit(x_train,y_train)
pred = mod_final.predict(x_test)
```

- Key Metrics for success in solving problem under consideration

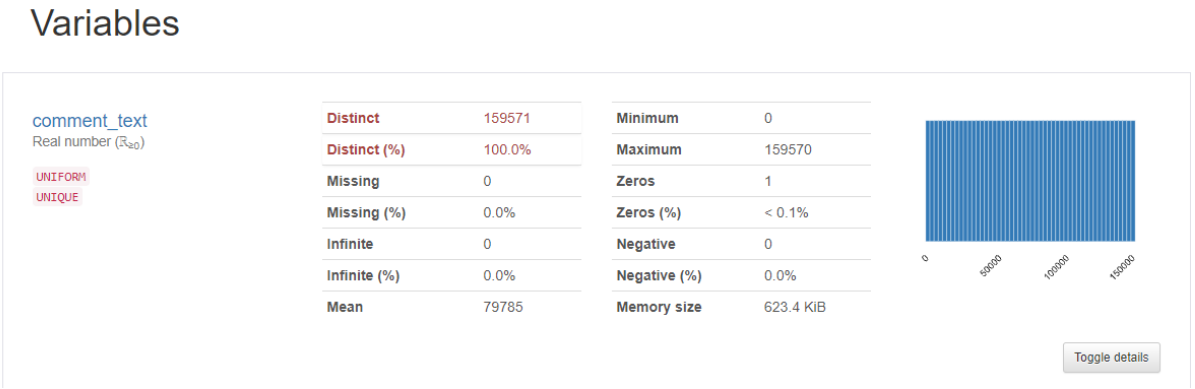
  - Using sklearn.metrics I have used Accuracy Score, Precision and Recall to check for all possible errors.

- Visualizations

  - The following figure gives us the total count or shows us the overview of the dataset itself.
  - All the attributes such as variables, observations, how many are numerical and categorical etc are all displayed using this figure.

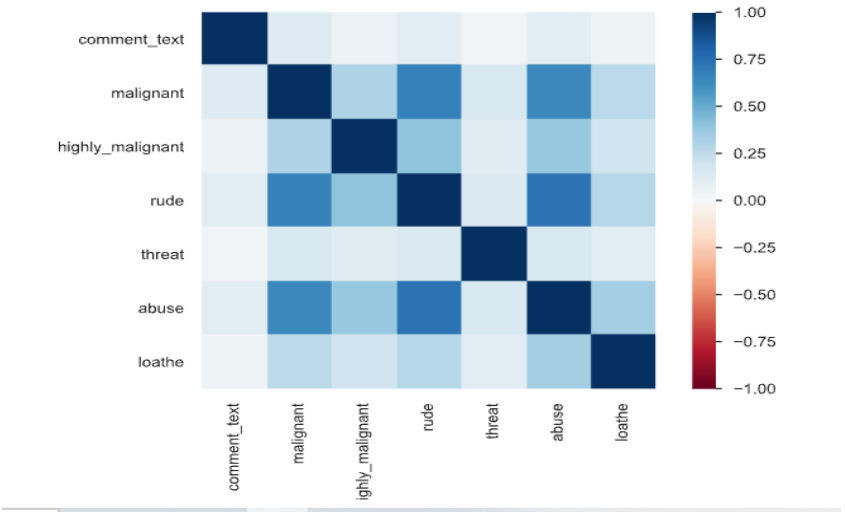


  - By toggling as mentioned we get to see the details of individual features / variables of the dataset.
  - The below figure gives us details on the variable “comment_text” and we can toggle within to get details of remaining features as well.

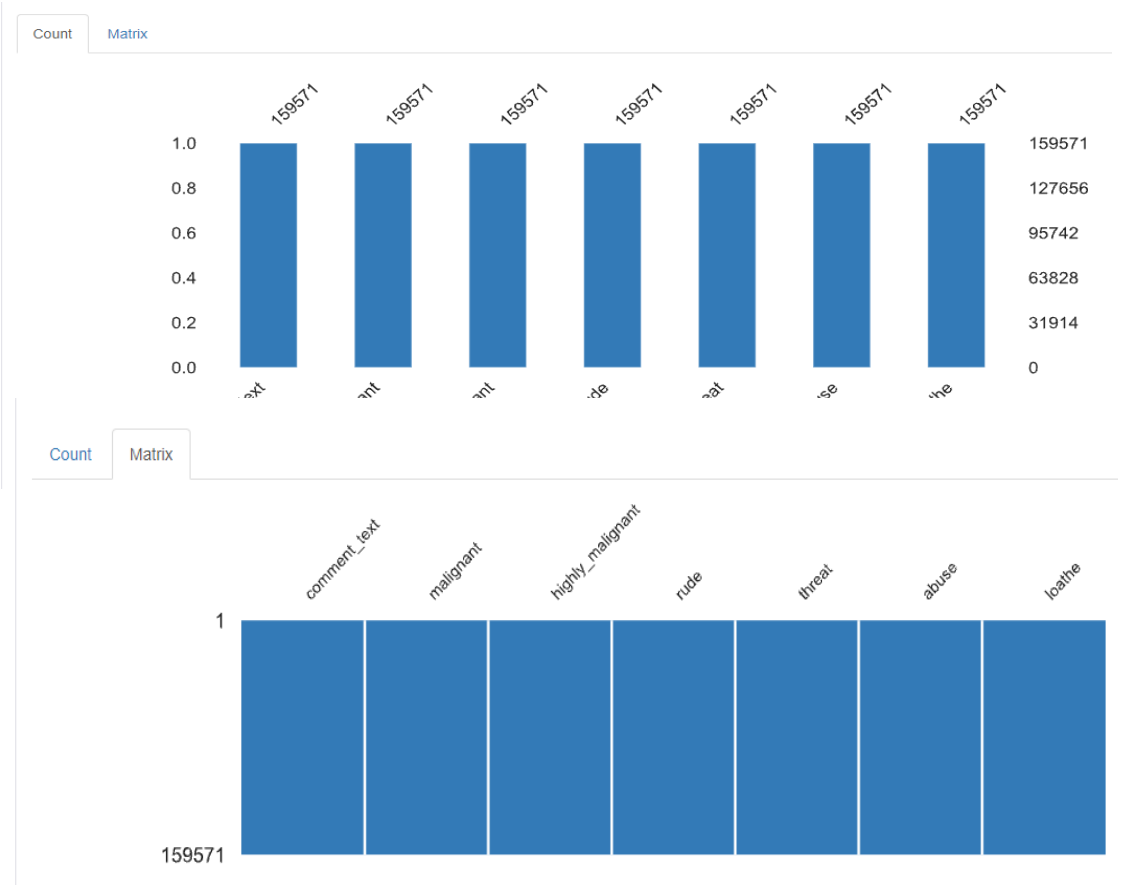


  - The below figure gives us details on the correlation matrix of the dataset and by default we have Pearson Correlation matrix.

- But Pandas profiling lets us observe and compare other correlation methods also like Spearman's, Kendall's, Cramer's and Phik Correlation.



- The following figure gives us both the count and matrix of missing values in the dataset if any.
- In the following case we do not have any missing values.



CONCLUSION

- Key Findings and Conclusions of the Study

    - From the EDA , I could observe that it’s not easy to categorize the comments completely under the given malignant labels.
    - Majority of the comments are at times just spam or only symbols that are neither malignant in any way but it also doesn’t make any sense.

- Learning Outcomes of the Study in respect of Data Science

    - With the help of this dataset, I was able to work on a single feature dataset.
    - I believe we can also proceed with neural network algorithms in such scenarios