FLIP ROBO

MALIGNANT COMMENTS CLASSIFIER

Submitted by:

AAYUSHI LASHKARI

ACKNOWLEDGMENT

I would like to express my special thanks to my mentor Mr. Shubham Yadav who gave me his support and assistance throughout the project. I am thankful to flip robo technologies and Data Trained institute to give me guidelines and knowledge to complete this project.

Links and websites that I preferred:

https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

https://prakhartechviz.blogspot.com/2019/02/multi-label-classification-python.html

https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

https://machinelearningmastery.com/multi-class-imbalanced-classification/

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.

- ## Conceptual Background of the Domain Problem

As now a days everyone is using social media and aware About the scenario that which type of comments people do it may be abusive , rude or anything else . Moreover to handle the data it needs to do pre processing of text data .
But to understand the problem domain it is not a out of box topic now a days.

- ## Review of Literature

The research starts with collecting the comments from different social media platforms in order to restrict and
Stop the increasing hatred through social media platform.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.

• Motivation for the Problem Undertaken

Motivation behind this problem works in two ways first is to learn a kind of new problem statement and how to work on that also how to handle this size of dataset and which technique to use for that .
And secondly by this model we can identify that how cyber bullying
And hatred can be stop or restrict.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

The data consist of 1.5 lakh rows (approx.) and 8 columns . The large number of rows is a problem in terms of model building and data cleaning .

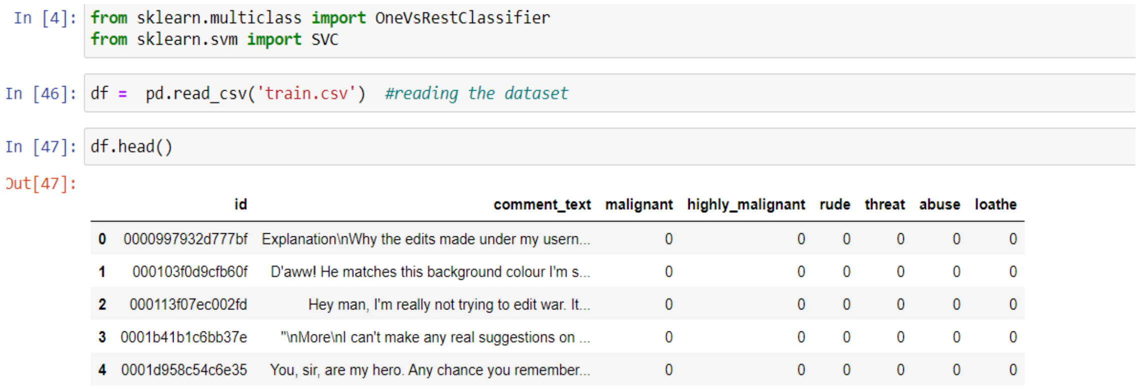
Also it is multi label classification problem it is important to evaluate that which model to use**.**

- ## Data Sources and their formats

The data is collected from the different social media platform to analysis the type of comments and hate in that environments.

And based on the data created six type of comments that are Malignant , Highly malignant , Abuse , Rude , Loathe , threat.

I have first read the data and converted it into the data frame.



```
In [4]: from sklearn.multiclass import OneVsRestClassifier
        from sklearn.svm import SVC
```

```
In [46]: df =  pd.read_csv('train.csv')  #reading the dataset
```

```
In [47]: df.head()
```

Out[47]:

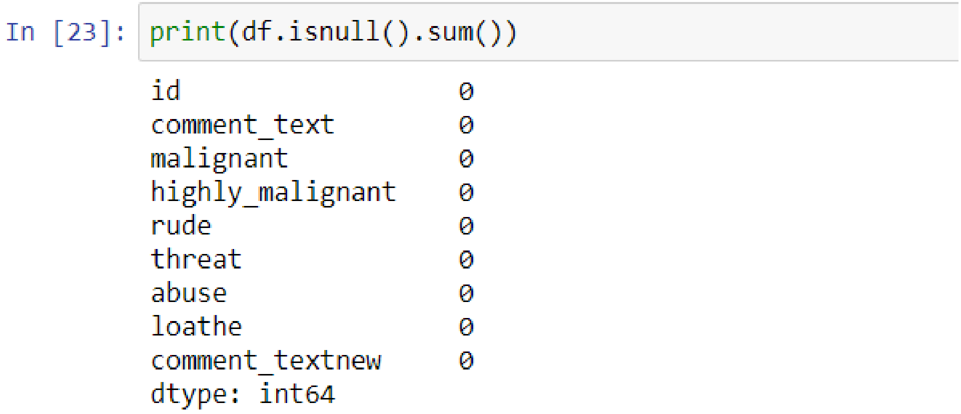
	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	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

- ## Data Preprocessing Done

Data pre processing is most important step to prepare the data for model building as in this problem statement we have only one independent column that is a text columns so following steps I have performed before model building :

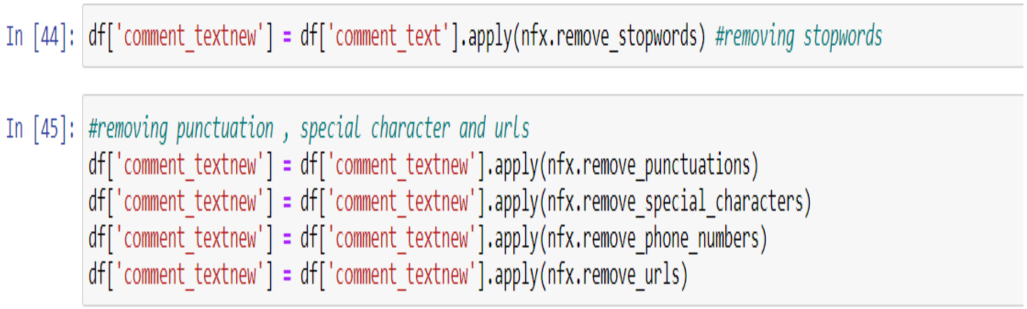
1. Finding Null values :

There is no any null values in the data set



```
In [23]: print(df.isnull().sum())

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

2. Removing stop words , punctuations, special characters and urls from the comments :



```
In [44]: df['comment_textnew'] = df['comment_text'].apply(nfx.remove_stopwords) #removing stopwords

In [45]: #removing punctuation , special character and urls
         df['comment_textnew'] = df['comment_textnew'].apply(nfx.remove_punctuations)
         df['comment_textnew'] = df['comment_textnew'].apply(nfx.remove_special_characters)
         df['comment_textnew'] = df['comment_textnew'].apply(nfx.remove_phone_numbers)
         df['comment_textnew'] = df['comment_textnew'].apply(nfx.remove_urls)
```

- Data Inputs- Logic- Output Relationships

Based on the comments the comments are classified though there are some common words like fuck , ass etc which is most common in each kind of category but so far analysed based on the intensity of the words used it is classifying the comment.

- ## Hardware and Software Requirements and Tools Used

Tool:

Jupyter NoteBook 6.1.4:

- Web-based interactive computing notebook Environment.

Software Requirement:

- The client environment may be Windows, macOS, or Linux.

Hardware Requirement:

- CPU: 2 x 64-bit, 2.8 GHz, 8.00 GT/s CPUs or better.
- Memory: minimum RAM size of 32 GB, or 16 GB RAM with 1600 MHz DDR3 installed, for a typical installation with 50 regular users.

Libraries:

- Pandas: For reading CSV file, Converting dataset into a data frame, handling date datatype, and more.
- Seaborn and matplotlib: For EDA and Visualization.

Model/s Development and Evaluation

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

Before building of the model data cleaning is important
And in this problem I have removed the stop words and
Punctuations , special characters and urls from the comment text . Also this steps are very time consuming
Because of large data set .
Because of large data set I have first divided the data into two parts then perform model building.
I have used OneVsRest classifier and different ensemble techniques for this problem.

Also Binary relevance and Powerset , though the model which is giving
Best accuracy is onevsrest classifier with svc .

- Testing of Identified Approaches (Algorithms)

Algorithms used in this problem statement:

1. OneVsRestClassifier :

With SVC:
Accuracy score : 0.91
Hamming loss : 0.02
With Decision Tree:
Accuracy score: 0.87
Hamming loss: 0.03
With KNN:
Accuracy score : 0.88
Hamming loss : 0.02

2. Ensemble Techniques:

1. RandomForest :

Accuracy score : 0.90

Hamming Loss: 0.028

2. Gradient Boost :

Accuracy score : 0.906

Hamming Loss : 0.024

3. Ada Boost:

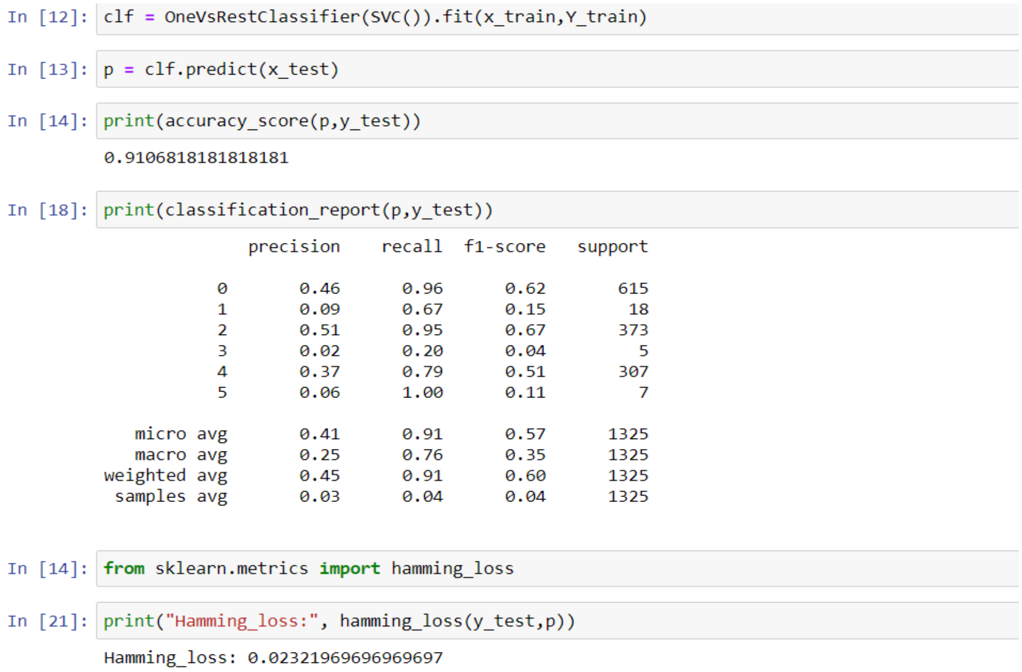
Accuracy score : 0.908

Hamming Loss: 0.02

- Run and Evaluate selected models

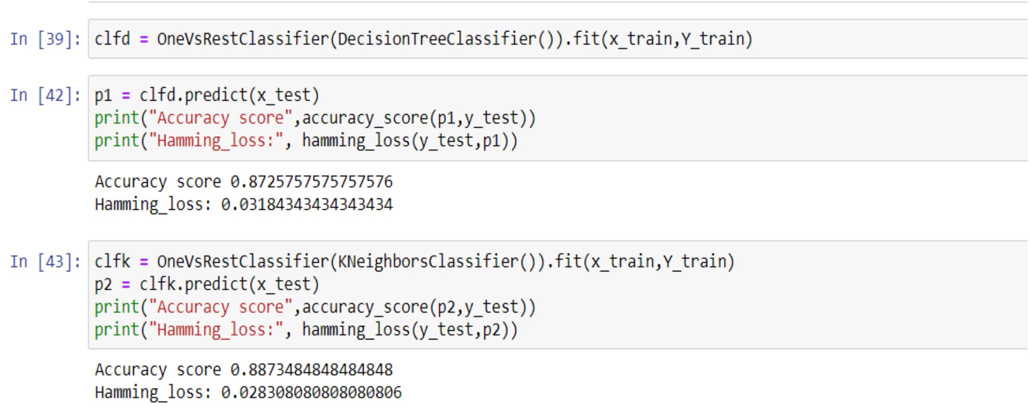
All the traditional Algorithm does not work for multi label or multi class classification . For Those problem we have to go for other solutions like : OneVsRest or OneVsAll classifier .The One-vs-Rest strategy splits a multi-class classification into one binary classification problem per class.

1. OneVsRestClassifier with SVC:



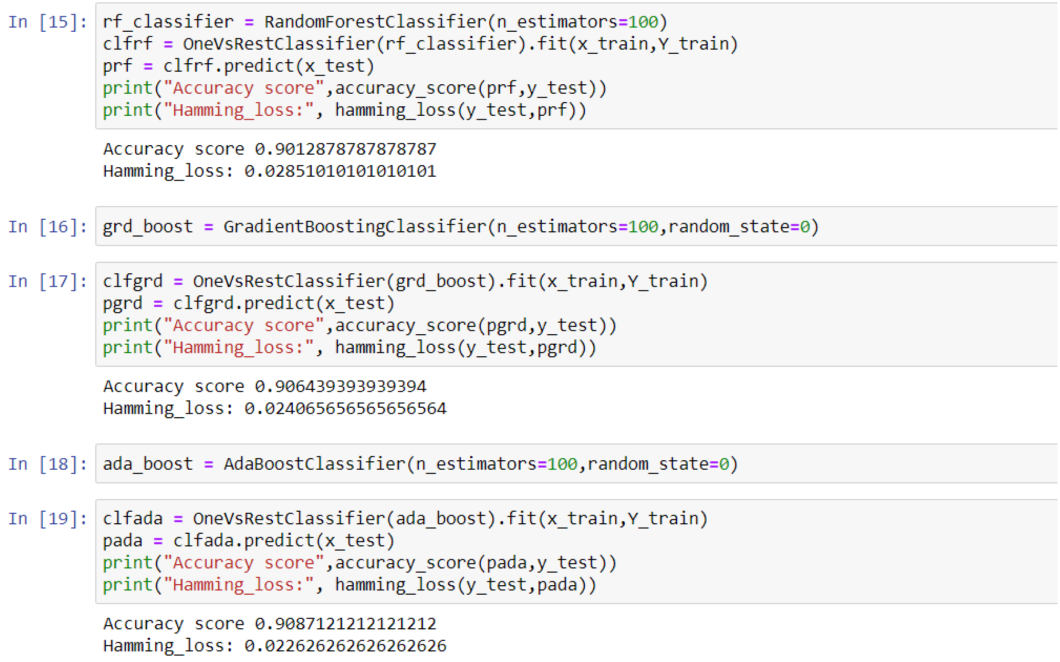
```
In [12]: clf = OneVsRestClassifier(SVC()).fit(x_train,Y_train)

In [13]: p = clf.predict(x_test)

In [14]: print(accuracy_score(p,y_test))

         0.9106818181818181

In [18]: print(classification_report(p,y_test))

                       precision    recall  f1-score   support

                    0       0.46      0.96      0.62       615
                    1       0.09      0.67      0.15        18
                    2       0.51      0.95      0.67       373
                    3       0.02      0.20      0.04         5
                    4       0.37      0.79      0.51       307
                    5       0.06      1.00      0.11         7

            micro avg       0.41      0.91      0.57      1325
            macro avg       0.25      0.76      0.35      1325
         weighted avg       0.45      0.91      0.60      1325
          samples avg       0.03      0.04      0.04      1325

In [14]: from sklearn.metrics import hamming_loss

In [21]: print("Hamming_loss:", hamming_loss(y_test,p))

         Hamming_loss: 0.02321969696969697
```

2. OneVsRest classifier with decision tree and KNN:



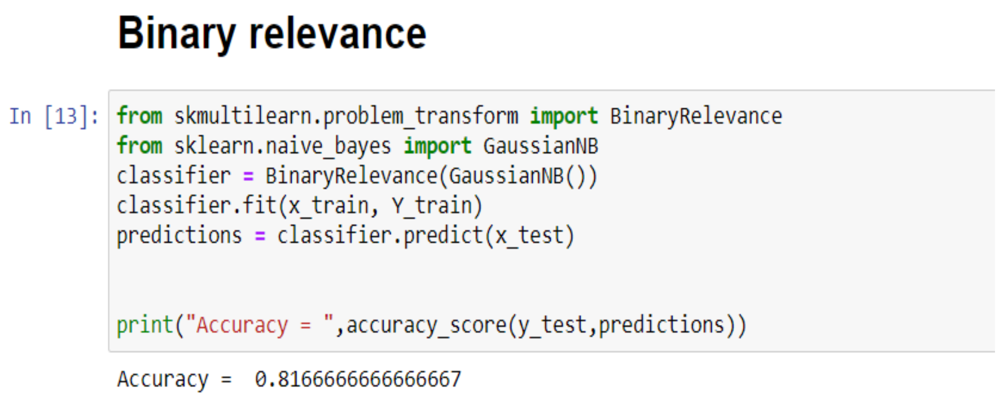
```
In [39]: clfd = OneVsRestClassifier(DecisionTreeClassifier()).fit(x_train,Y_train)

In [42]: p1 = clfd.predict(x_test)
         print("Accuracy score",accuracy_score(p1,y_test))
         print("Hamming_loss:", hamming_loss(y_test,p1))

         Accuracy score 0.8725757575757576
         Hamming_loss: 0.03184343434343434

In [43]: clfk = OneVsRestClassifier(KNeighborsClassifier()).fit(x_train,Y_train)
         p2 = clfk.predict(x_test)
         print("Accuracy score",accuracy_score(p2,y_test))
         print("Hamming_loss:", hamming_loss(y_test,p2))

         Accuracy score 0.8873484848484848
         Hamming_loss: 0.028308080808080806
```

3. Ensemble Techniques (Random Forest , Gradient Boost and Ada Boost Classifier):



```
In [15]: rf_classifier = RandomForestClassifier(n_estimators=100)
         clfrf = OneVsRestClassifier(rf_classifier).fit(x_train,Y_train)
         prf = clfrf.predict(x_test)
         print("Accuracy score",accuracy_score(prf,y_test))
         print("Hamming_loss:", hamming_loss(y_test,prf))

         Accuracy score 0.9012878787878787
         Hamming_loss: 0.02851010101010101

In [16]: grd_boost = GradientBoostingClassifier(n_estimators=100,random_state=0)

In [17]: clfgrd = OneVsRestClassifier(grd_boost).fit(x_train,Y_train)
         pgrd = clfgrd.predict(x_test)
         print("Accuracy score",accuracy_score(pgrd,y_test))
         print("Hamming_loss:", hamming_loss(y_test,pgrd))

         Accuracy score 0.906439393939394
         Hamming_loss: 0.024065656565656564

In [18]: ada_boost = AdaBoostClassifier(n_estimators=100,random_state=0)

In [19]: clfada = OneVsRestClassifier(ada_boost).fit(x_train,Y_train)
         pada = clfada.predict(x_test)
         print("Accuracy score",accuracy_score(pada,y_test))
         print("Hamming_loss:", hamming_loss(y_test,pada))

         Accuracy score 0.9087121212121212
         Hamming_loss: 0.022626262626262626
```

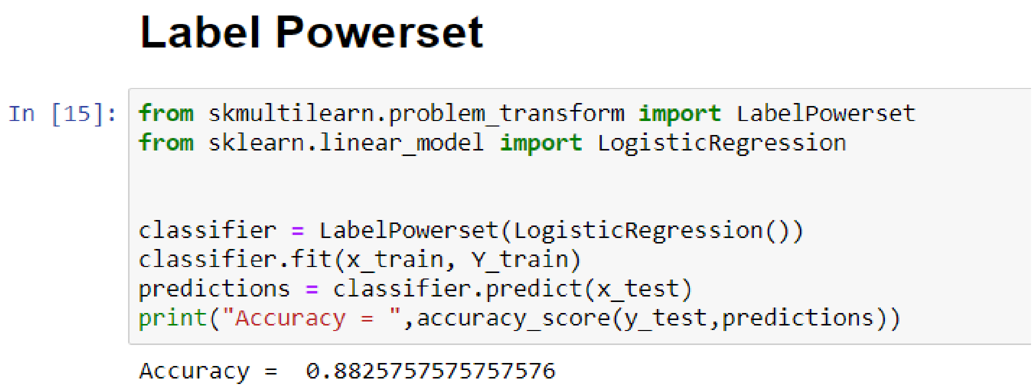
4. Binary Relevance



Binary relevance

```
In [13]: from skmultilearn.problem_transform import BinaryRelevance
         from sklearn.naive_bayes import GaussianNB
         classifier = BinaryRelevance(GaussianNB())
         classifier.fit(x_train, Y_train)
         predictions = classifier.predict(x_test)


         print("Accuracy = ",accuracy_score(y_test,predictions))

         Accuracy =  0.8166666666666667
```

5. Label Power set



Label Powerset

```
In [15]: from skmultilearn.problem_transform import LabelPowerset
         from sklearn.linear_model import LogisticRegression


         classifier = LabelPowerset(LogisticRegression())
         classifier.fit(x_train, Y_train)
         predictions = classifier.predict(x_test)
         print("Accuracy = ",accuracy_score(y_test,predictions))

         Accuracy =  0.8825757575757576
```

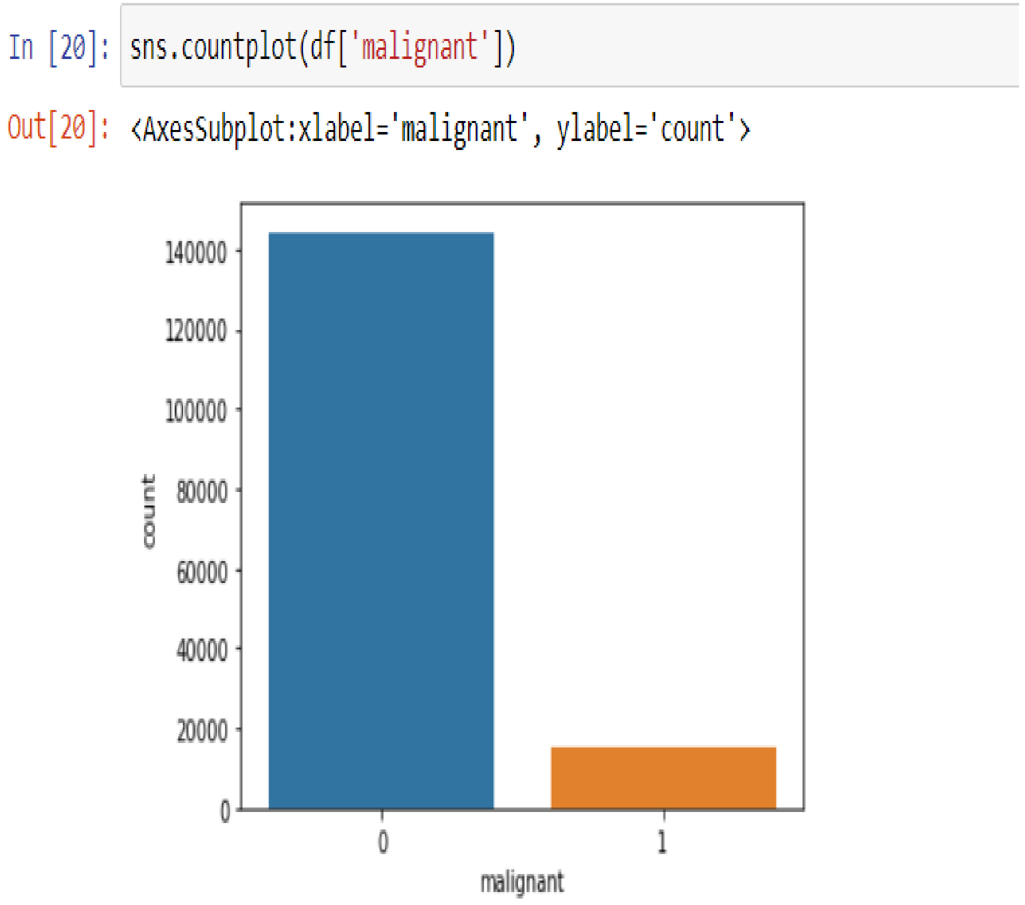
- Key Metrics for success in solving problem under consideration

As I have used six Algorithms for this problem statements but not used each metrics for every algorithm
Reason being it will uselessly utilize CPU power and time.
So for selecting the best model I have used Accuracy score and hamming loss.
Model with best accuracy score and hamming loss I have
Used Classification report and confusion matrix also.

Hamming loss is the fraction of wrong labels to the total number of labels. In multi-class classification, hamming loss is calculated as the hamming distance between y_true and y_pred .

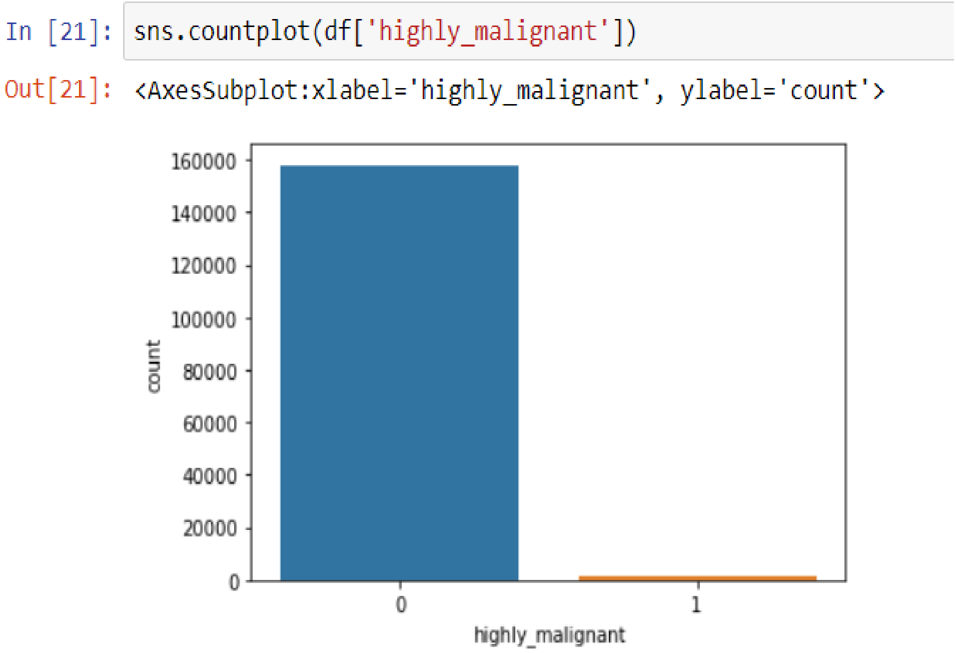
- Visualizations

1. Malignant comments :



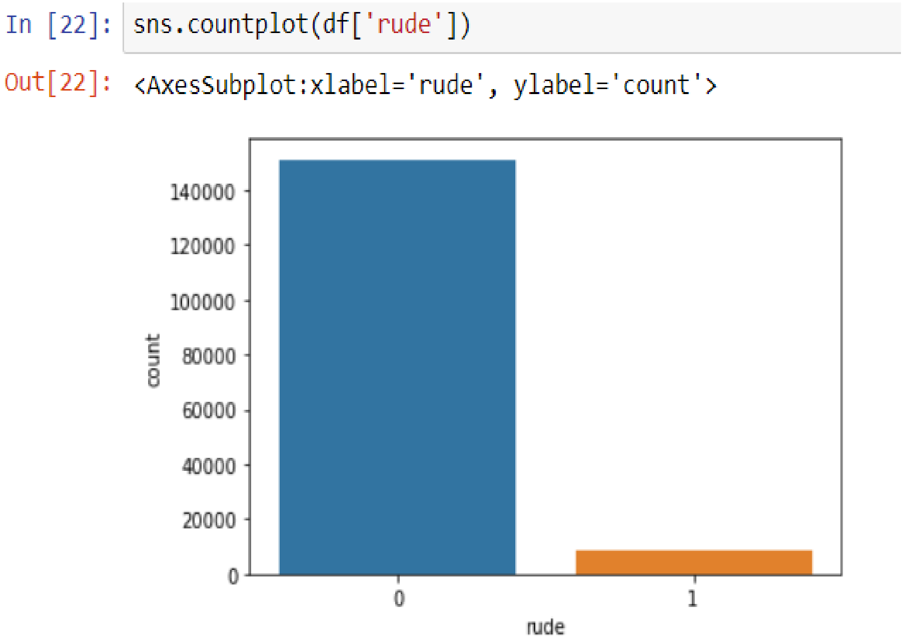
```
In [20]: sns.countplot(df['malignant'])

Out[20]: <AxesSubplot:xlabel='malignant', ylabel='count'>
```

2. Highly Malignant:



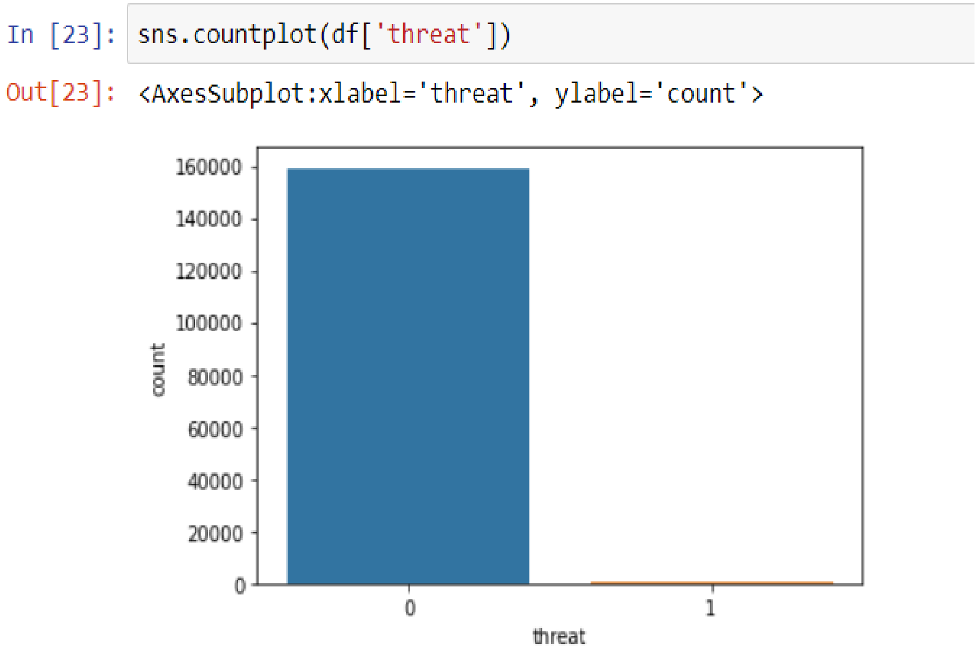
```
In [21]: sns.countplot(df['highly_malignant'])
Out[21]: <AxesSubplot:xlabel='highly_malignant', ylabel='count'>
```

3. Rude:



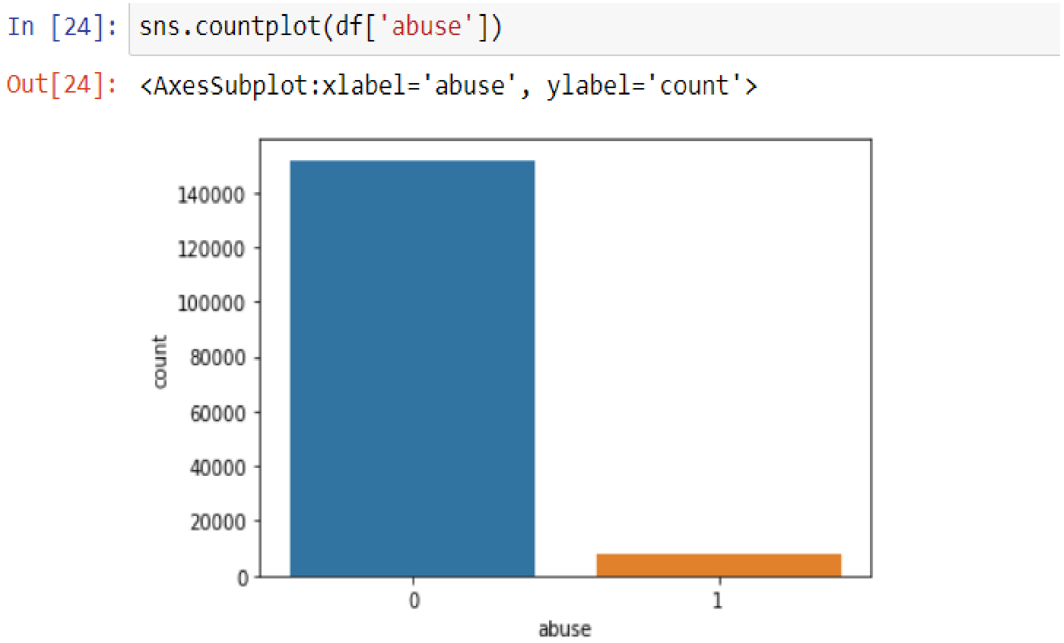
```
In [22]: sns.countplot(df['rude'])
Out[22]: <AxesSubplot:xlabel='rude', ylabel='count'>
```

4. Threat



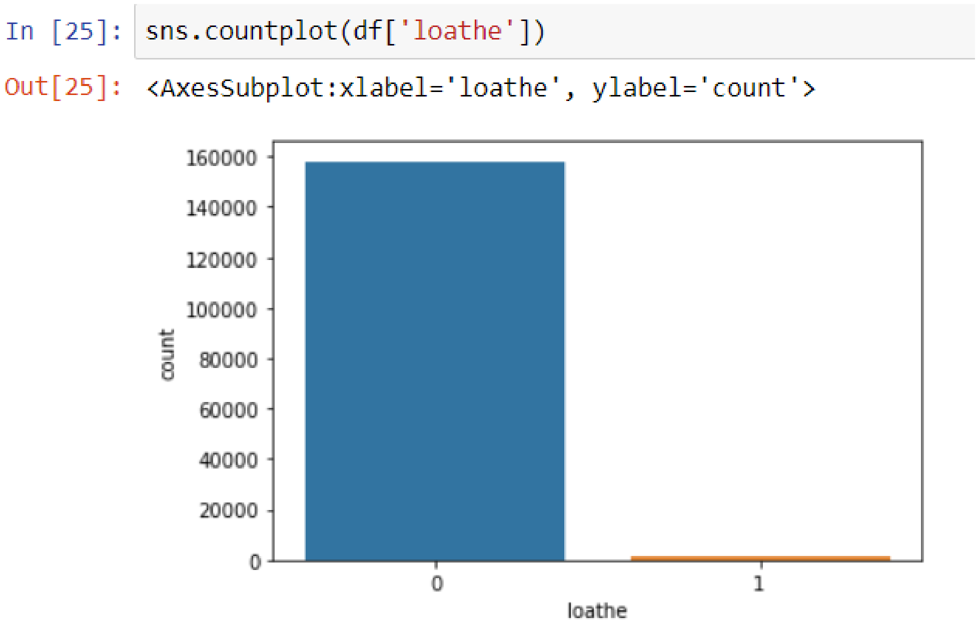
```
In [23]: sns.countplot(df['threat'])

Out[23]: <AxesSubplot:xlabel='threat', ylabel='count'>
```

5. Abuse



```
In [24]: sns.countplot(df['abuse'])

Out[24]: <AxesSubplot:xlabel='abuse', ylabel='count'>
```

6. Loathe

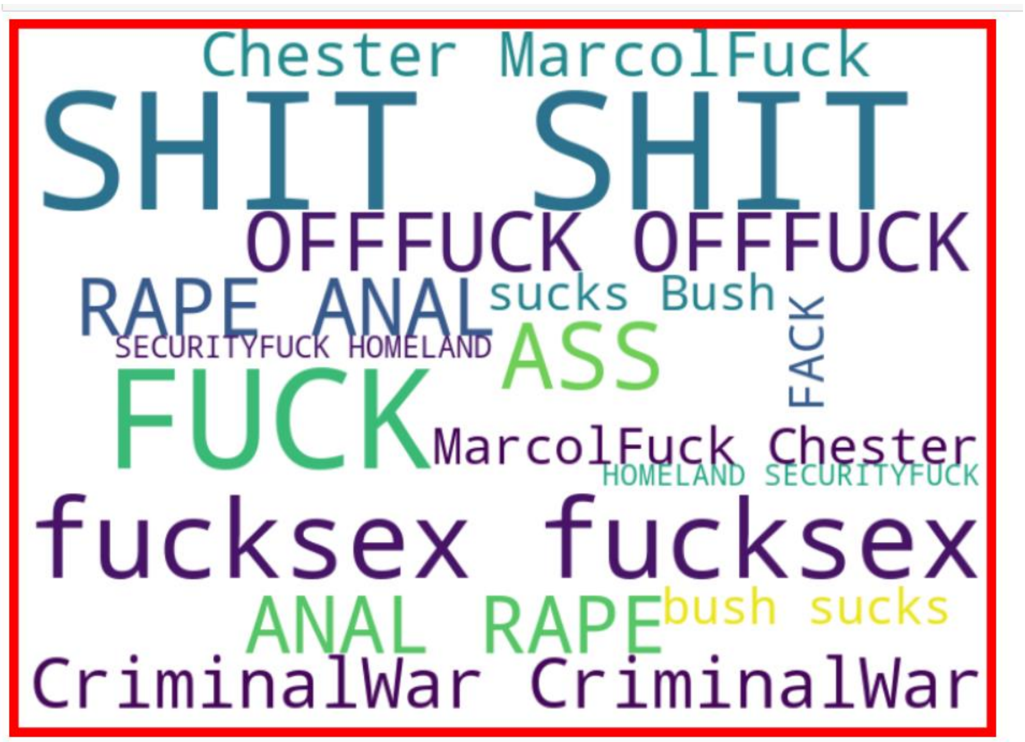


```
In [25]: sns.countplot(df['loathe'])

Out[25]: <AxesSubplot:xlabel='loathe', ylabel='count'>
```

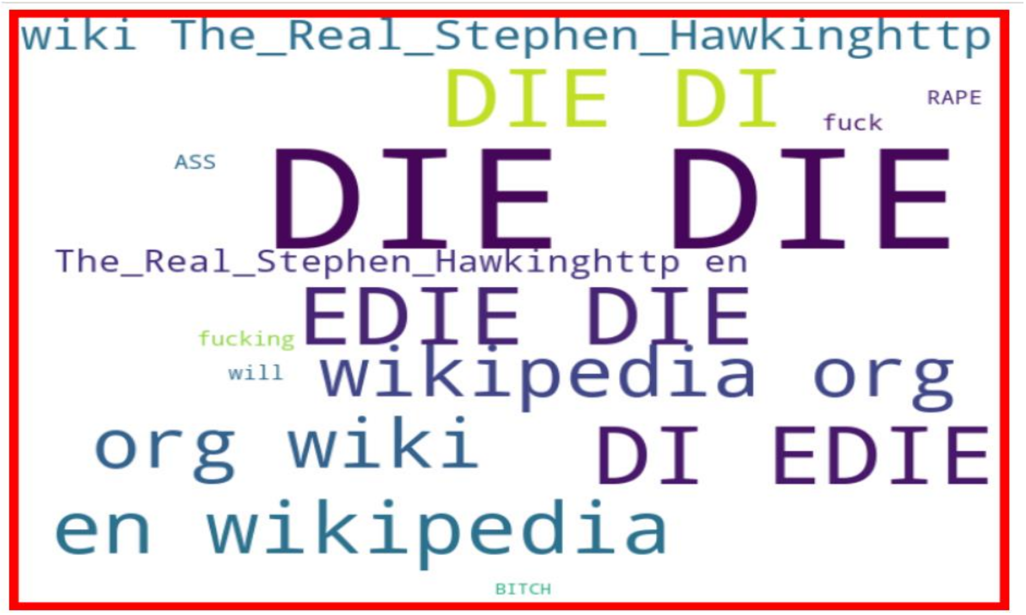
7. Word Cloud of Malignant :

NIGGER NIGGER
moron hi
fucking
people
know
page
will
hi moron
article
think
fuck
hate
Wikipedia
one
ass

8. Word cloud of abuse is no and highly malignant is yes:



9. Cloud word when all the labels are yes



- Interpretation of the Results

1. All the labels are highly unbalanced because of which the f1 score for each label is different and probably less.
2. If we look at the classification report the F1 score less for those Label which are more and more unbalanced.
3. Looking at the word clouds it is clearly concluded that what are the main words each labels.

CONCLUSION

- Key Findings and Conclusions of the Study

In the given problem statement the sampling of the data
Is more important also the size of the dataset is a challenge because of it models takes time to build and give the predictions.

More the labels are unbalanced less value for f1 score.
To get more accurate values need to use sampling before modelling.

- Learning Outcomes of the Study in respect of Data Science

This problem gives knowledge of different multi label and multi class classification and also how to use different types of metrics and how to treat large size dataset.
Specially in case of word to vector phase size of dataset
Matters a lot.

- Limitations of this work and Scope for Future Work

As it is a multi class classification problem we can explore more

Algorithms and hyper parametric tuning to get more accurate results.

Also I have not used any hyper parametric tuning because of the large dataset my system did not support though I tried couple of times.