FLIP ROBO

MALIGNANT COMMENTS CLASSIFIER

Submitted by:

SANDEEP

KUMAR

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to the sources Medium, TowardsDataScience, StackOverflow, Krish Naik’s youtube channel which helped me to accomplish this project.

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.

  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.

- Conceptual Background of the Domain Problem

  This project proposed a Machine Learning Approach combined with Natural Language Processing for toxicity detection and its type identification in user comments. Finally, the best score of minimu log loss was achieved through BR method with multinomial classifier from scikit multilearn.

- Review of Literature

  Nowadays users leave numerous comments on different social networks, news portals, and forums. Some of the comments are toxic or abusive. Due to numbers of comments, it is unfeasible to manually moderate them, so most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, we performed a systematic review of the state-of-the-art in toxic comment classification using machine learning methods. We extracted data from 31 selected primary relevant studies. In our analysis of every primary study we investigated: data set used, evaluation metric, used machine learning methods, classes of toxicity, and comment language.

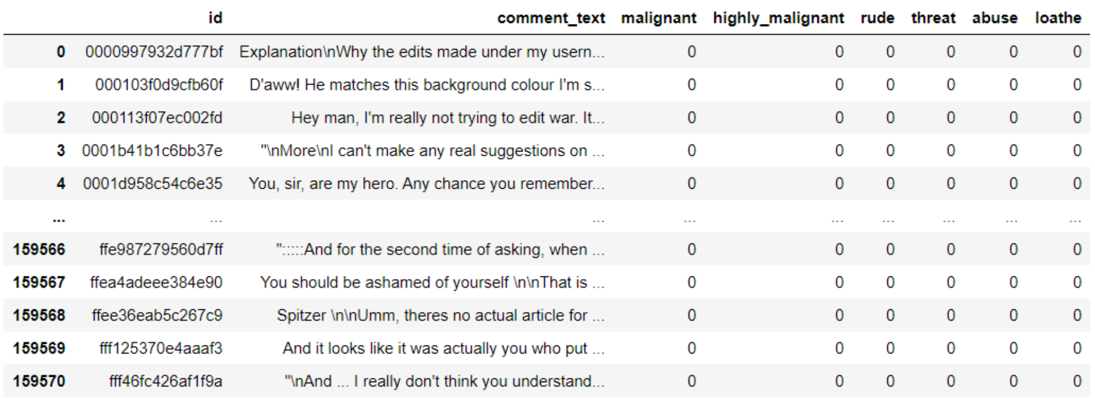
- Motivation for the Problem Undertaken

  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.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

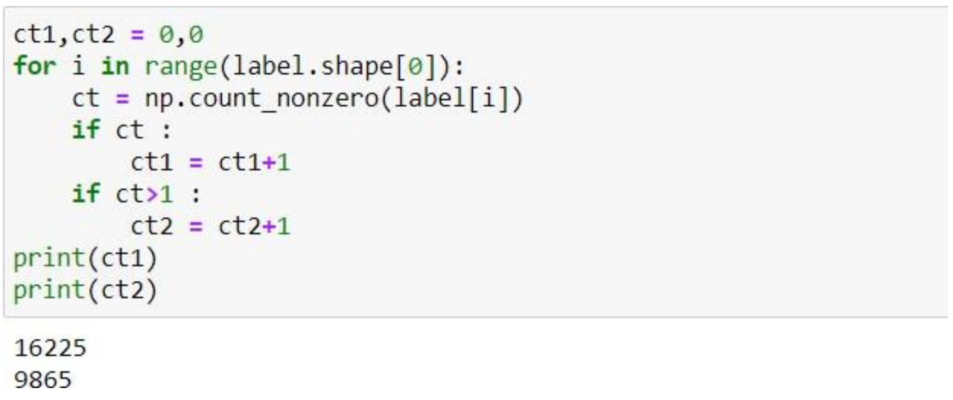
There are in total 159571 rows and 8 columns in our training dataset as shown in figure below,



	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
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

We found the occurrence of multilabelled data as shown below,



```
ct1,ct2 = 0,0
for i in range(label.shape[0]):
    ct = np.count_nonzero(label[i])
    if ct :
        ct1 = ct1+1
    if ct>1 :
        ct2 = ct2+1
print(ct1)
print(ct2)
```

```
16225
9865
```

ct1 counts samples having atleast one label

ct2 counts samples having 2 or more than 2 labels

From this analysis we make interpretation that, there are 16225 comments with single label, 9865 multilabelled comments and 133481 comments without any label i.e non toxic comments.

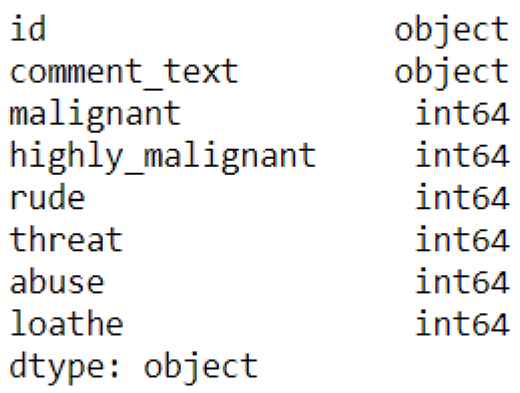
- Data Sources and their formats

The variable features of this problem statement are,

- **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.

The data types of features are shown in fig,



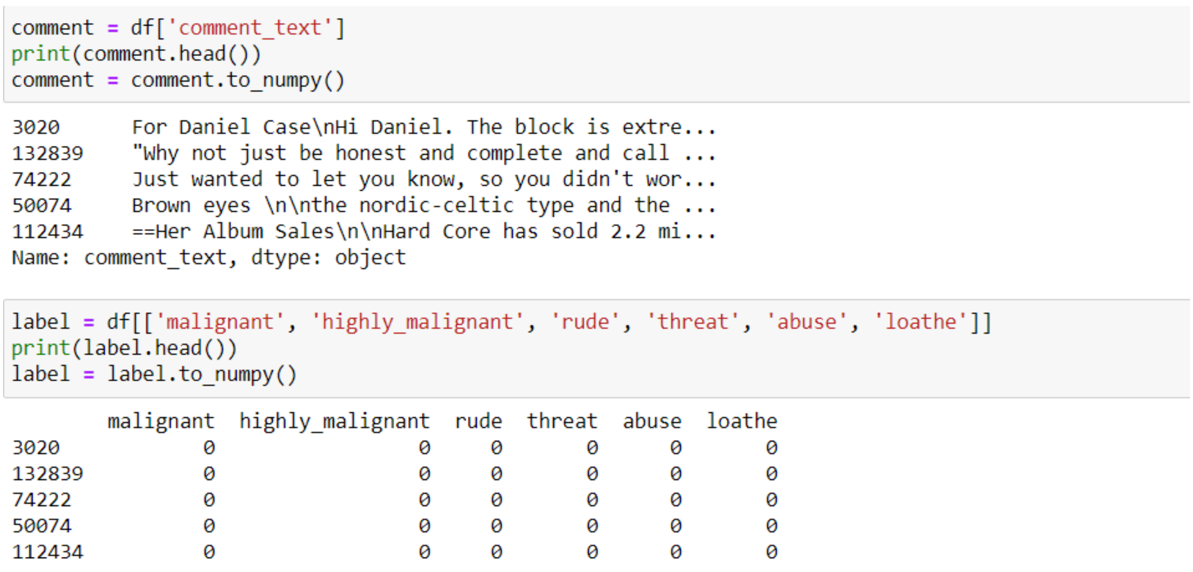
```
id                   object
comment_text         object
malignant             int64
highly_malignant      int64
rude                  int64
threat                int64
abuse                 int64
loathe                int64
dtype: object
```

Fig Data types of features

- ## Data Preprocessing Done

It was observed that running train_test_split on the heavy preprocessed dataframe sometimes resulted in system going out of memory. Hence to avoid such cases, one extra line of code was added. The df.reindex code will shuffle the indices initially, so that later splitting dataset into training and testing will give fairer results.

We then separated comment text data and outcome labels as shown below,



```
comment = df['comment_text']
print(comment.head())
comment = comment.to_numpy()
```

```
3020      For Daniel Case\nHi Daniel. The block is extre...
132839    "Why not just be honest and complete and call ...
74222     Just wanted to let you know, so you didn't wor...
50074     Brown eyes \n\nthe nordic-celtic type and the ...
112434    ==Her Album Sales\n\nHard Core has sold 2.2 mi...
Name: comment_text, dtype: object
```

```
label = df[['malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe']]
print(label.head())
label = label.to_numpy()
```

```
        malignant  highly_malignant  rude  threat  abuse  loathe
3020            0                 0     0       0      0       0
132839          0                 0     0       0      0       0
74222           0                 0     0       0      0       0
50074           0                 0     0       0      0       0
112434          0                 0     0       0      0       0
```

Some very large length comments can be seen, in our dataset. These pose serious problems like adding excessively more words to the training dataset, causing training time to increase and accuracy to decrease! Hence, a threshold of 400 characters will be created and only comments which have length smaller than 400 will be used further.

Hence, after removing comments longer than 400 characters, we are still left with 115893 comments, which seems enough for training purposes.

Preprocessing involved the following steps, but these were performed in a slightly different manner:

Removing Punctuations and other special

characters Splitting the comments into

individual words Removing Stop Words

Stemming and

Lemmatising Applying

Hash Vectorizer

Splitting dataset into Training and Testing

- Set of assumptions related to the problem under consideration

  By looking into the target vaariable label we assumed that it was a Multiclass classification type of problem.

  We assumed that running train_test_split on the heavy preprocessed dataframe sometimes results in system going out of memory. Hence to avoid such cases, one extra line of code was added. The df.reindex code shuffled the indices initially, so that later splitting dataset into training and testing will give fairer results.

  We assumed that the id column was unique id to all the rows and we will be not needing it.

- Hardware and Software Requirements and Tools Used

  This project was done on laptop with i5 processor with quad cores and eight threads with 8gb of ram and latest GeForce GTX 1650 GPU on Anaconda, jupyter notebook.

  The tools, libraries and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, nltk, WordNetLemmatizer, PosterStemmer, StopWords, Hash vectorizer, joblib.

  Through pandas library we loaded our csv file 'Data file' into dataframe and performed data manipulation and

  analysis. With the help of numpy we worked with

  arrays.

  With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization.

  With WordNetLemmatizer we applied lemmatization

  technique. With PoterStemmer we applied stemming

  technique.

  With StopWords we removed the extra words which were not giving any information.

With Hash vectorizer we converted comments to vectors. Through joblib we saved our model in csv format.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  We first seperated comment text and data outcome labels.

  Preprocessing involved the following steps, but these were performed in a slightly different manner:

  Removing Punctuations and other special

  characters Splitting the comments into

  individual words Removing Stop Words

  Stemming and

  Lemmatising Applying

  Hash Vectorizer

  Splitting dataset into Training and Testing

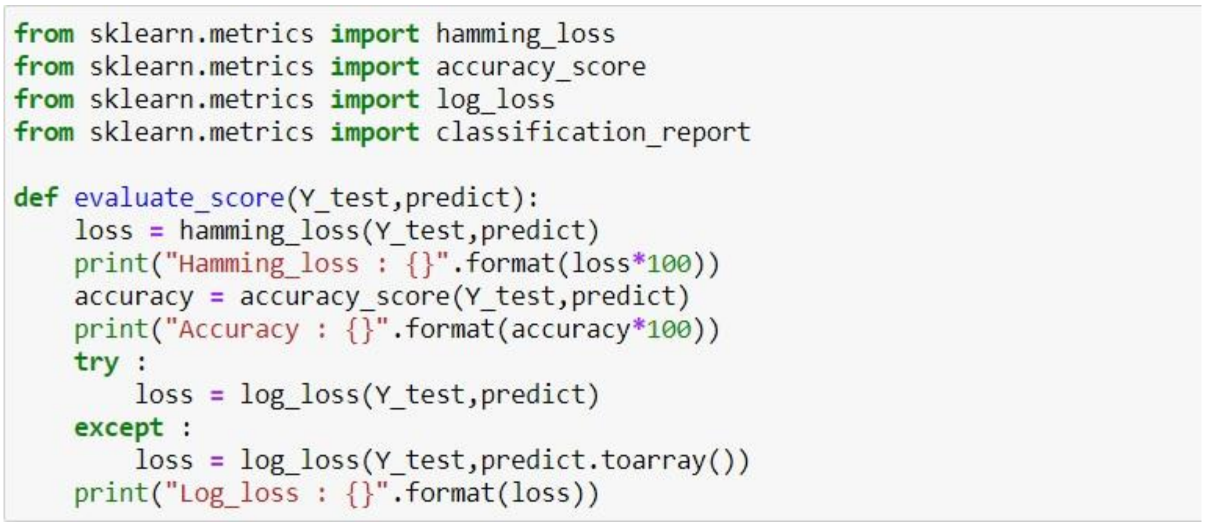
- Testing of Identified Approaches (Algorithms)

  The algorithms we used for the training and testing are as follows:-

  - Binary Relevance (BR) Method with MultinomialNB classifiers (from scratch)
  - BR Method with Multinomial classifier (from scikit-multilearn)
  - BR Method with SVM classifier (from scikit-multilearn)

- ## Run and Evaluate selected models

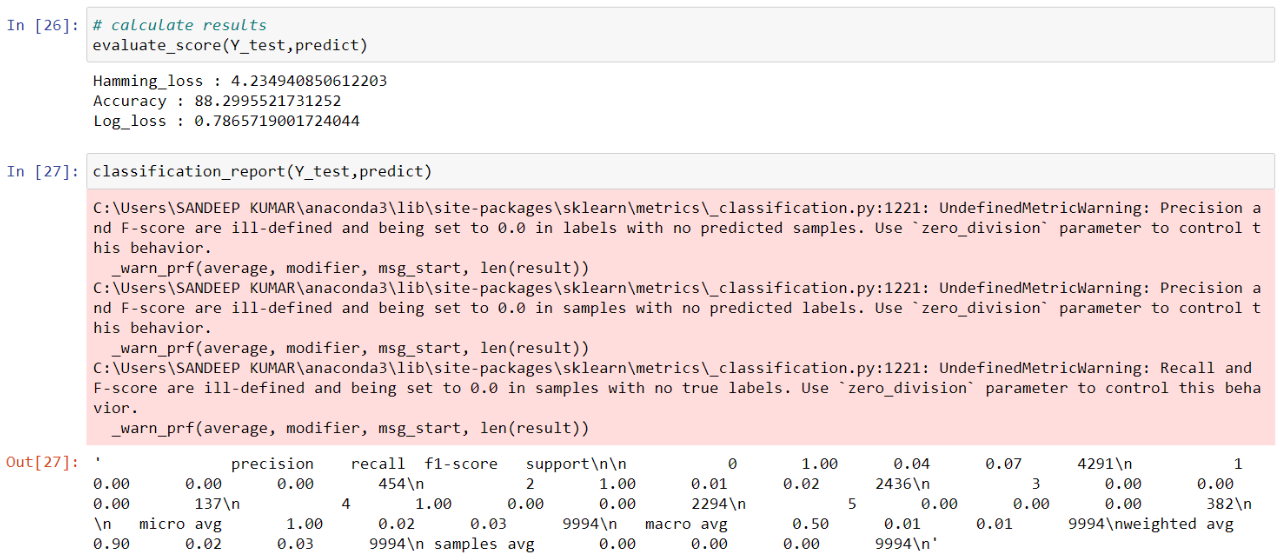
  Here we define all the evaluation metrics,



```python
from sklearn.metrics import hamming_loss
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss
from sklearn.metrics import classification_report

def evaluate_score(Y_test,predict):
    loss = hamming_loss(Y_test,predict)
    print("Hamming_loss : {}".format(loss*100))
    accuracy = accuracy_score(Y_test,predict)
    print("Accuracy : {}".format(accuracy*100))
    try :
        loss = log_loss(Y_test,predict)
    except :
        loss = log_loss(Y_test,predict.toarray())
    print("Log_loss : {}".format(loss))
```

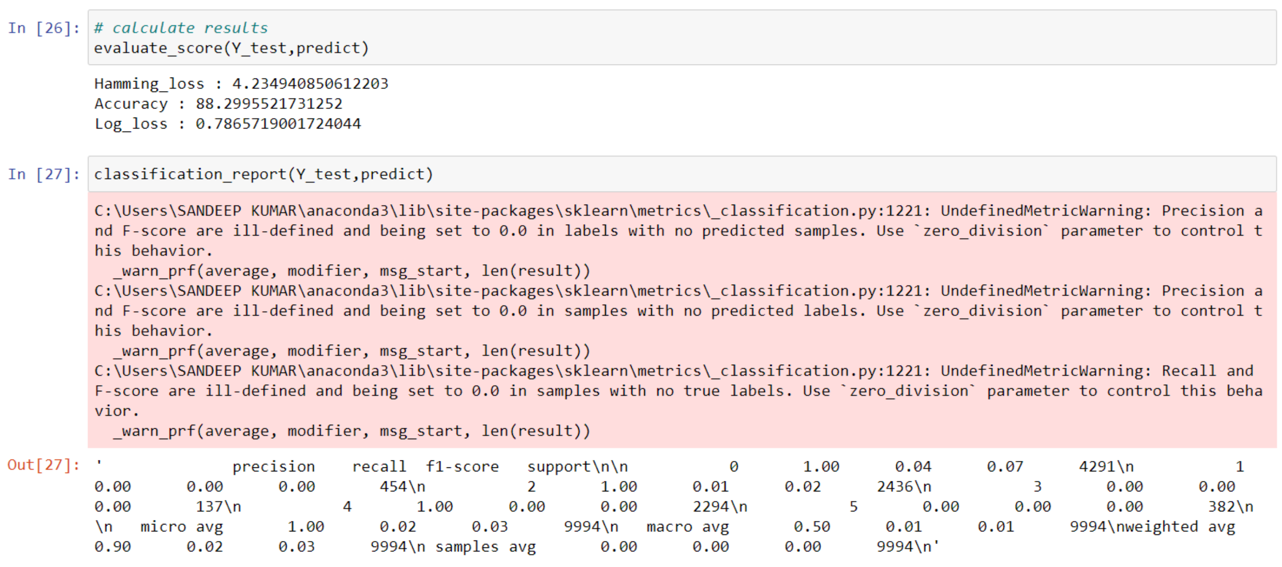
The results observed over different evaluation metrics are shown in fig,

Binary Relevance (BR) Method with MultinomialNB classifiers (from scratch) :-



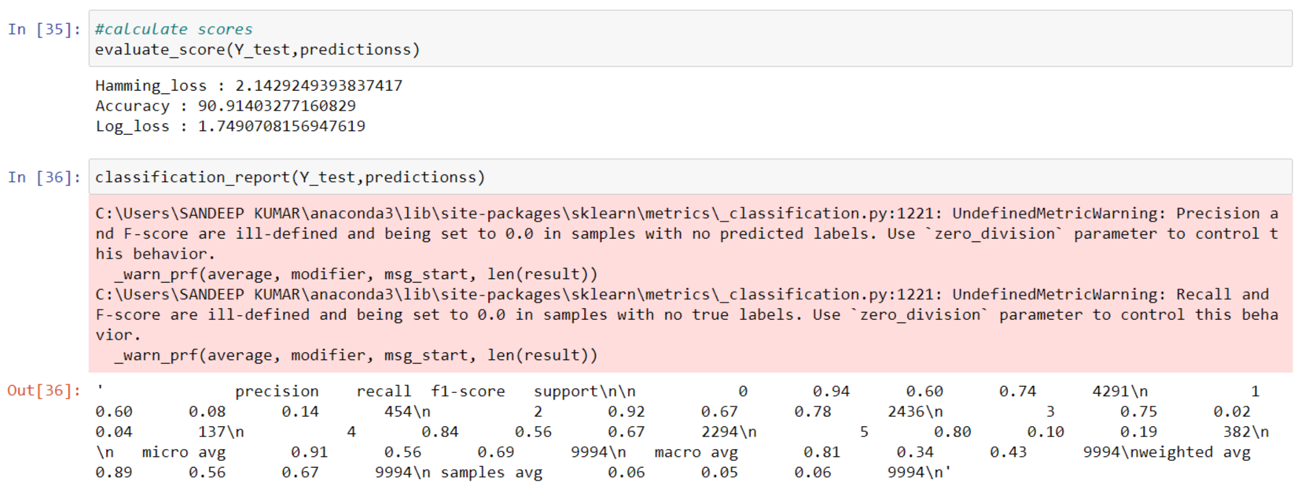
```
In [26]: # calculate results
         evaluate_score(Y_test,predict)

         Hamming_loss : 4.234940850612203
         Accuracy : 88.2995521731252
         Log_loss : 0.7865719001724044

In [27]: classification_report(Y_test,predict)

         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision a
         nd F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control t
         his behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision a
         nd F-score are ill-defined and being set to 0.0 in samples with no predicted labels. Use `zero_division` parameter to control t
         his behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Recall and
         F-score are ill-defined and being set to 0.0 in samples with no true labels. Use `zero_division` parameter to control this beha
         vior.
           _warn_prf(average, modifier, msg_start, len(result))

Out[27]: '              precision    recall  f1-score   support\n\n           0       1.00      0.04      0.07      4291\n           1
         0.00      0.00      0.00       454\n           2       1.00      0.01      0.02      2436\n           3       0.00      0.00
         0.00       137\n           4       1.00      0.00      0.00      2294\n           5       0.00      0.00      0.00       382\n
         \n   micro avg       1.00      0.02      0.03      9994\n   macro avg       0.50      0.01      0.01      9994\nweighted avg
         0.90      0.02      0.03      9994\n samples avg       0.00      0.00      0.00      9994\n'
```

BR method with Multinomial classifier from scikit multilearn:-



```
In [26]: # calculate results
         evaluate_score(Y_test,predict)

         Hamming_loss : 4.234940850612203
         Accuracy : 88.2995521731252
         Log_loss : 0.7865719001724044

In [27]: classification_report(Y_test,predict)

         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision a
         nd F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control t
         his behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision a
         nd F-score are ill-defined and being set to 0.0 in samples with no predicted labels. Use `zero_division` parameter to control t
         his behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Recall and
         F-score are ill-defined and being set to 0.0 in samples with no true labels. Use `zero_division` parameter to control this beha
         vior.
           _warn_prf(average, modifier, msg_start, len(result))

Out[27]: '              precision    recall  f1-score   support\n\n           0       1.00      0.04      0.07      4291\n           1
         0.00      0.00      0.00       454\n           2       1.00      0.01      0.02      2436\n           3       0.00      0.00
         0.00       137\n           4       1.00      0.00      0.00      2294\n           5       0.00      0.00      0.00       382\n
         \n   micro avg       1.00      0.02      0.03      9994\n   macro avg       0.50      0.01      0.01      9994\nweighted avg
         0.90      0.02      0.03      9994\n samples avg       0.00      0.00      0.00      9994\n'
```

BR method with SVM classifier:-



```
In [35]: #calculate scores
         evaluate_score(Y_test,predictionss)

         Hamming_loss : 2.1429249393837417
         Accuracy : 90.91403277160829
         Log_loss : 1.7490708156947619

In [36]: classification_report(Y_test,predictionss)

         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision a
         nd F-score are ill-defined and being set to 0.0 in samples with no predicted labels. Use `zero_division` parameter to control t
         his behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Recall and
         F-score are ill-defined and being set to 0.0 in samples with no true labels. Use `zero_division` parameter to control this beha
         vior.
           _warn_prf(average, modifier, msg_start, len(result))

Out[36]: '              precision    recall  f1-score   support\n\n           0       0.94      0.60      0.74      4291\n           1
         0.60      0.08      0.14       454\n           2       0.92      0.67      0.78      2436\n           3       0.75      0.02
         0.04       137\n           4       0.84      0.56      0.67      2294\n           5       0.80      0.10      0.19       382\n
         \n   micro avg       0.91      0.56      0.69      9994\n   macro avg       0.81      0.34      0.43      9994\nweighted avg
         0.89      0.56      0.67      9994\n samples avg       0.06      0.05      0.06      9994\n'
```

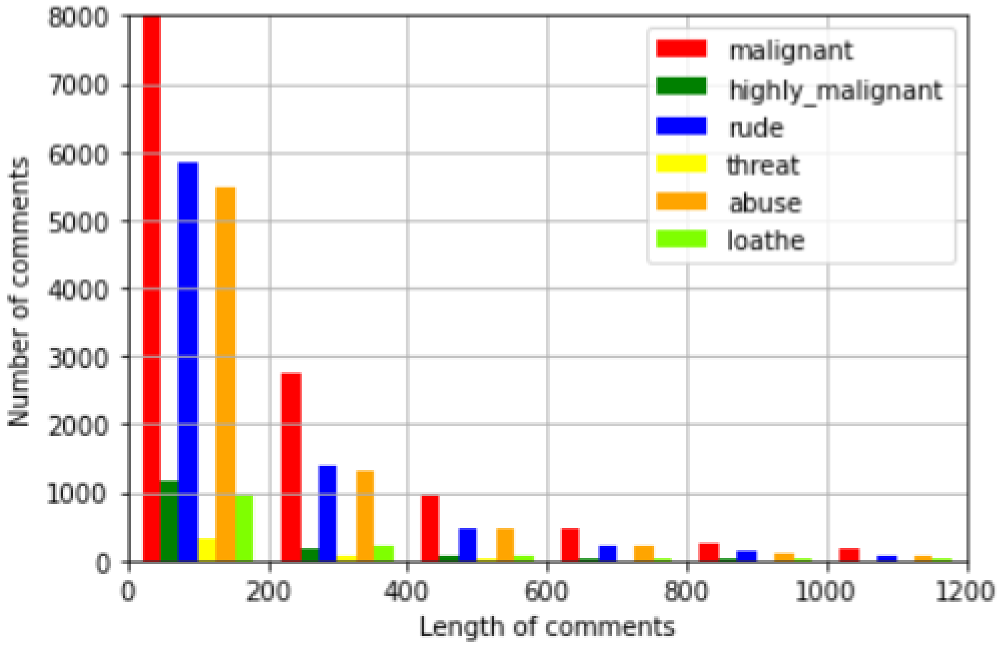
Fig Results observed

- ## Key Metrics for success in solving problem under consideration

  We used the metric log loss by selecting BR method with Multinomial classifier from scikit multilearn model which was giving us best(minimum) log loss score.

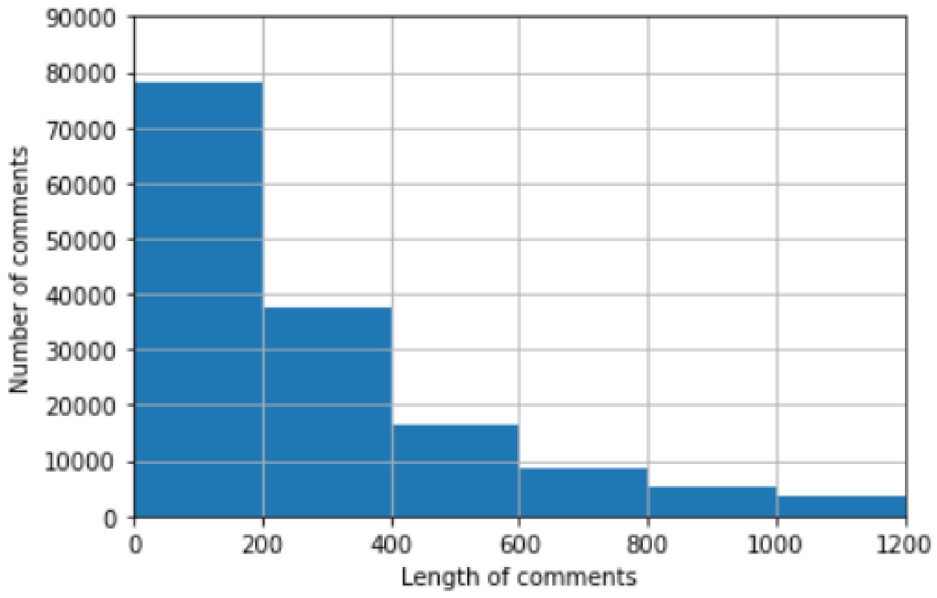
- ## Visualizations

Number of comments classified as malignant, highly malignant, rude, threat, abuse, loathe depending on the length.



Number of comments having length varying from 0 to 1200.

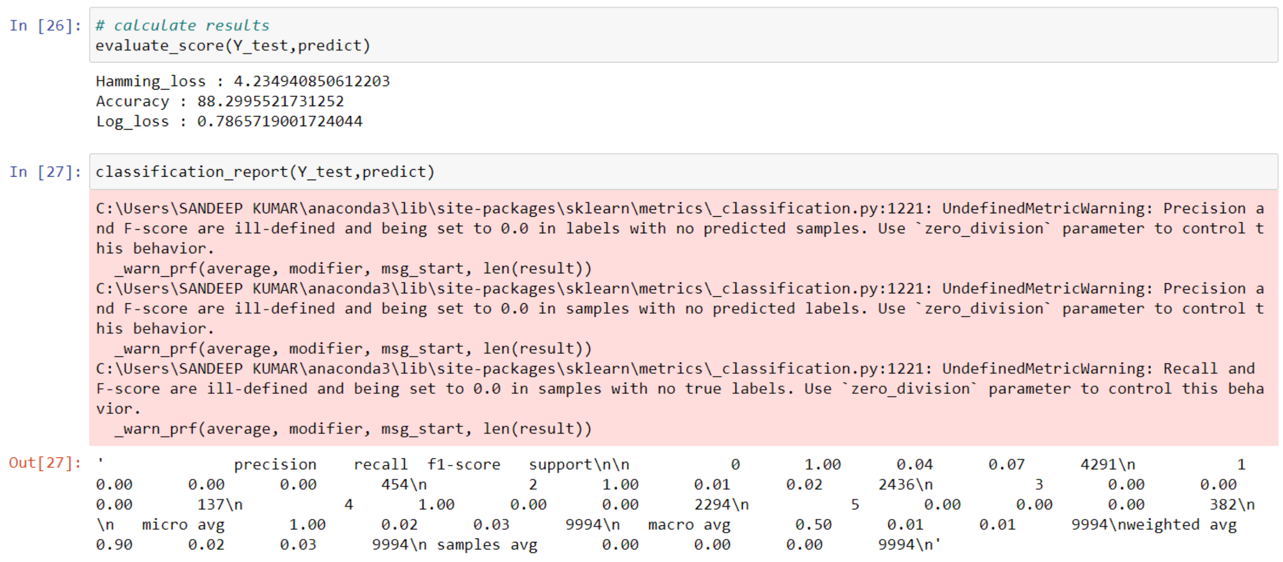
```
average length of comment: 394.139
```



Average length of comments is 394.139

- ## Interpretation of the Results

  We used the metric log loss by selecting BR method with Multinomial classifier from scikit multilearn model which was giving us best(minimum) log loss score of 0.7.



```
In [26]: # calculate results
         evaluate_score(Y_test,predict)

         Hamming_loss : 4.234940850612203
         Accuracy : 88.2995521731252
         Log_loss : 0.7865719001724044

In [27]: classification_report(Y_test,predict)

         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision a
         nd F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control t
         his behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision a
         nd F-score are ill-defined and being set to 0.0 in samples with no predicted labels. Use `zero_division` parameter to control t
         his behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         C:\Users\SANDEEP KUMAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Recall and
         F-score are ill-defined and being set to 0.0 in samples with no true labels. Use `zero_division` parameter to control this beha
         vior.
           _warn_prf(average, modifier, msg_start, len(result))

Out[27]: '              precision    recall  f1-score   support\n\n           0       1.00      0.04      0.07      4291\n           1
         0.00      0.00      0.00       454\n           2       1.00      0.01      0.02      2436\n           3       0.00      0.00
         0.00       137\n           4       1.00      0.00      0.00      2294\n           5       0.00      0.00      0.00       382\n
         \n   micro avg       1.00      0.02      0.03      9994\n   macro avg       0.50      0.01      0.01      9994\nweighted avg
         0.90      0.02      0.03      9994\n samples avg       0.00      0.00      0.00      9994\n'
```

CONCLUSION

- ## Key Findings and Conclusions of the Study

  This project proposed a Machine Learning Approach combined with Natural Language Processing for toxicity detection and its type identification in user comments. Finally, the best score of minimu log loss was achieved through BR method with multinomial classifier from scikit multilearn.

- ## Learning Outcomes of the Study in respect of Data Science

  Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stopwords. We were also able to learn to convert strings into vectors through hash vectorizer. In this project we applied different evaluation metrics like log loss, hamming loss besides accuracy.

- Limitations of this work and Scope for Future Work

While we couldn’t reach out goal of minimum log loss in malignant comments classifier, we did end up creating a system that can with get very close to that goal. As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules
and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and vesatility to the project.