FLIP ROBO

MALIGNANT COMMENTS CLASSIFIER PROJECT

Words frequented in Clean Comments

Submitted by:

JASMINE KAUR

FLIP ROBO TECHNOLOGIES

ACKNOWLEDGMENT

This project was my first experience with NLP project. It was a challenging one but at the same time it taught me several things. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am thankful to Flip Robo Technologies, Bangalore for the constant supervision as well as guidance regarding the project. I want to thank my SME Ms. Khushboo Garg for providing an amazing dataset and addressing my queries immediately.

I would like to express my special thanks to our institute DataTrained for giving the basic knowledge that made me capable of making this project a reality. With the help of their constant guidance and motivation, I am able to perform the tasks efficiently. In the passage of this project, I got a chance to enhance my technical as well non-technical skills.

INTRODUCTION

- ## Business Problem Framing

  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 cyber bullying.

  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, cyber bullying, 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 cyber bullying 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

  ### MULTILABEL VS MULTICLASS CLASSIFICATION

  In multi-class classification, the data can belong to only one label out of all the labels we are considering in the project. For example, a given picture of a flower may be a sunflower, rose or lily only and not a combination of these.

In multi-label classification, data can belong to more than one label simultaneously. For example, in our case a comment may be malignant, threat or loathe at the same time. It may also happen that the comment is positive/neutral and hence does not belong to any of the six labels.

In the past few years, we have seen the cases related to social media hatred have been increased exponentially. Social media is turning a platform where people can spread hatred anonymously. The reason of online hate is the difference in opinion, culture, race, religion, occupation, nationality etc. People use filthy language due to these differences to prove their point and make the other party feel hurtful. This is one of the major concerns today.

- ## Review of Literature

  Sentiment classification regarding toxicity has been intensively researched in the past few years, largely in the context of social media data where researchers have applied various machine learning systems to try and tackle the problem of toxicity as well as the related, more well-known task of sentiment analysis. The motivation behind the project is to build a model that can detect toxic comments and find a bias with respect to the mention of selected identities.

- ## Motivation for the Problem Undertaken

  The upsurge in the volume of unwanted comments called malignant comments has created an intense need for the development of more dependable and robust malignant comments filters. Machine learning methods are used to detect and filter malignant comments. Build a model which can be used to predict in terms of a probability for comments to be malignant. In the project, Label 1 indicates that the comment is malignant while Label 0 indicates that the comment is not malignant.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  We have built a machine learning model. The first step in building a machine learning model is to perform data pre-processing steps using NLP. We have tried different models and select the best model.

  The steps in building the model are as follows:

  1. Data cleaning
  2. EDA
  3. Data pre-processing
  4. Model building
  5. Model evaluation
  6. Selecting the best model

- Data Sources and their formats

  We were provided with a dataset that contains a training dataset which has around 159000 sample points and a test dataset which has around 153000 sample points. All the samples in training dataset contain 8 field namely 'ID', ' comments', 'Malignant', 'Highly Malignant', 'Rude', 'Threat', 'Abuse', and 'Loathe'.

  The label column is created that can be either 1 or 0. Label 1 denotes if there is at least one sort of bad language used in the comment and Label 0 is used when the comment is either positive or neutral.

- Data Preprocessing Done

  Due to the wide range of the available data, it is necessary to clean the data to bring out the best out of data. So, we dropped unnecessary column such as ID. Now, we are left with 7 attributes.

CLEANING THE DATA USING NLP:

Data cleaning is the process of preparing data for analysis by removing and modifying the data that is incorrect, incomplete, irrelevant, or not formatted properly. We remove or modify such data because it is not required in the analysing process as it might hinder the process or provide inaccurate results.

We have created a new column namely ‘length before cleaning’ which shows the total length of the comments before cleaning the data.

After this, we have performed following cleaning steps to clean the comment text.

- Replaced the extra lines or ‘\n’ from the text.
- Transform the text into lower case.
- Replaced the email addresses with the text ‘emailaddress’.
- Replaced the URLs with the text ‘webaddress’.
- Removed the numbers.
- Removed the HTML tags.
- Removed the punctuations.
- Removed all the non-ascii characters.
- Removed the unwanted white spaces.
- Removed the remaining tokens that not alphabetic.
- Removed the stop words.

All the text cleaning is performed by defining a function and applying the same using apply() to the comment_text column of the training dataset.

TOKENIZATION

Word tokenization is the process of splitting a large sample of text into words. This is a requirement in natural language processing tasks where each word needs to be captured and subjected to further analysis.

After cleaning the text, each comment i.e. the corpus is split into words. Thus, the text is tokenized into words using the word_tokenize().

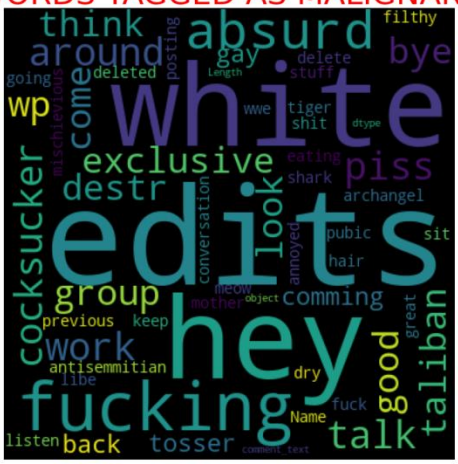
LEMMATIZATION

Lemmatization in NLTK refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word known as lemma.

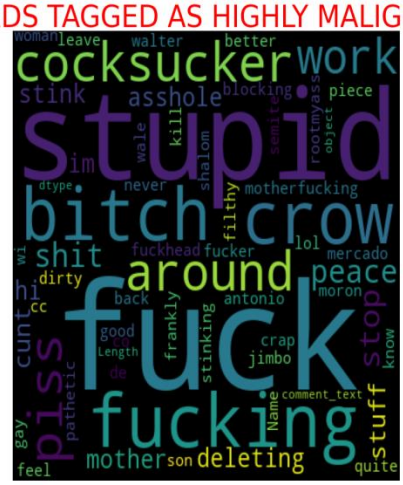
The NLTK lemmatization method is based on WorldNet's built-in morph function. Thus, the words are lemmatized using WordNetLemmatizier() after importing the necessary library to perform the same and then creating the instance for it.

PLOTTED WORDCLOUD FOR EACH FEATURE:

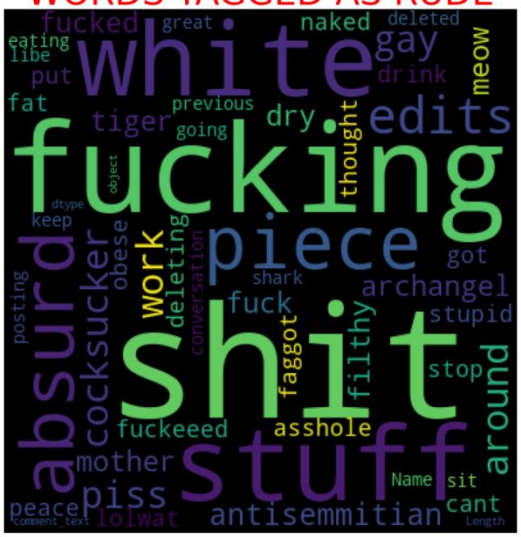
WORDS TAGGED AS MALIGNANT



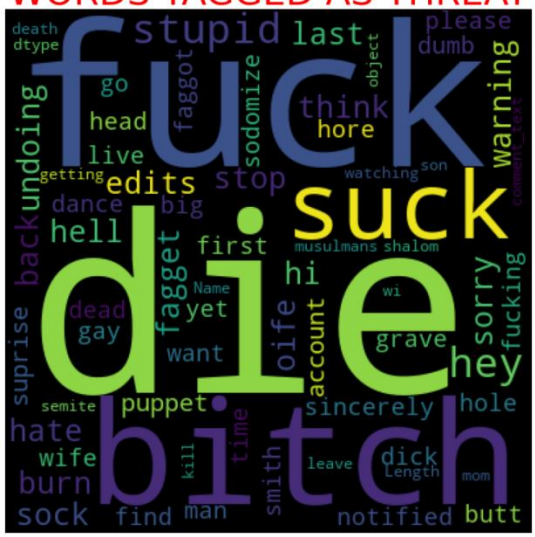
WORDS TAGGED AS HIGHLY MALIGNANT



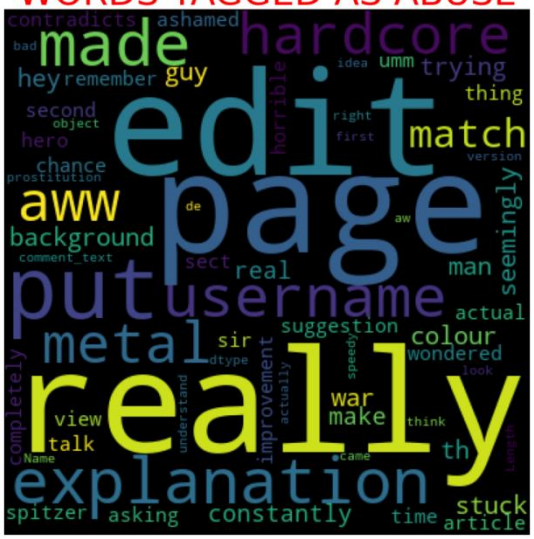
WORDS TAGGED AS RUDE



WORDS TAGGED AS THREAT



WORDS TAGGED AS ABUSE



WORDS TAGGED AS LOATHE

From the above wordclouds, we can see that the large texts have more weightage in their respective type of comments whereas small texts have less weightage.

- Hardware and Software Requirements and Tools Used

  For doing the project, a laptop is used as hardware device. I have used google colab to code the entire project. MS Excel is used for reading the data. Python libraries are used to perform programming and analysis.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
  - **a.** We have separated the independent and dependent features and converted the data into number feature
  - **b.** Splitted the training and testing data, along with handling the imbalanced dataset using RandomOverSampler.
  - **c.** We completed model building process.
- Testing of Identified Approaches (Algorithms)

********** KNeighborsClassifier **********

KNeighborsClassifier()

accuracy_score: 0.7150654565226484

cross_val_score: 0.9166708490499401

roc_auc_score: 0.6530926561056766

Hamming_loss: 0.28493454347735164

Log_loss : 9.841483803725552

Classification report:

	precision	recall	f1-score	support
0	0.94	0.73	0.82	42958
1	0.19	0.58	0.29	4860
accuracy			0.72	47818
macro avg	0.57	0.65	0.56	47818
weighted avg	0.86	0.72	0.77	47818

Confusion matrix:

```
[[31397 11561]
 [ 2064  2796]]
```

********** RandomForestClassifier **********

RandomForestClassifier()

accuracy_score: 0.9530302396587059

cross_val_score: 0.956817151204079

roc_auc_score: 0.830244250437359

Hamming_loss: 0.046969760341294076

Log_loss : 1.622289286721371

Classification report:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	42958
1	0.83	0.68	0.75	4860
accuracy			0.95	47818
macro avg	0.90	0.83	0.86	47818
weighted avg	0.95	0.95	0.95	47818

Confusion matrix:

```
[[42286   672]
 [ 1574  3286]]
```

********** AdaBoostClassifier **********

AdaBoostClassifier()

accuracy_score: 0.9309046802459325

cross_val_score: 0.9459257723137311

roc_auc_score: 0.8194810003914246

Hamming_loss: 0.0690953197540675

Log_loss : 2.38649701170052

Classification report:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	42958
1	0.65	0.68	0.67	4860
accuracy			0.93	47818
macro avg	0.81	0.82	0.81	47818
weighted avg	0.93	0.93	0.93	47818

Confusion matrix:

```
[[41211  1747]
 [ 1557  3303]]
```

********** GradientBoostingClassifier **********

GradientBoostingClassifier()

accuracy_score: 0.9451880045171275

cross_val_score: 0.9402353912573943

roc_auc_score: 0.7992370383015509

Hamming_loss: 0.05481199548287256

Log_loss : 1.8931518806123366

Classification report:

	precision	recall	f1-score	support
0	0.96	0.98	0.97	42958
1	0.80	0.62	0.70	4860
accuracy			0.95	47818
macro avg	0.88	0.80	0.83	47818
weighted avg	0.94	0.95	0.94	47818

Confusion matrix:

```
[[42203   755]
 [ 1866  2994]]
```

********** XGBoostClassifier **********

XGBClassifier()

accuracy_score: 0.9458572085825422

```
cross_val_score:  0.937776047160854

roc_auc_score:  0.795138667359467

Hamming_loss: 0.05414279141745786

Log_loss : 1.8700370365781034

Classification report:

              precision    recall  f1-score   support

           0       0.96      0.98      0.97     42958
           1       0.81      0.61      0.69      4860

    accuracy                           0.95     47818
   macro avg       0.89      0.80      0.83     47818
weighted avg       0.94      0.95      0.94     47818


Confusion matrix:

[[42284   674]
 [ 1915  2945]]
```

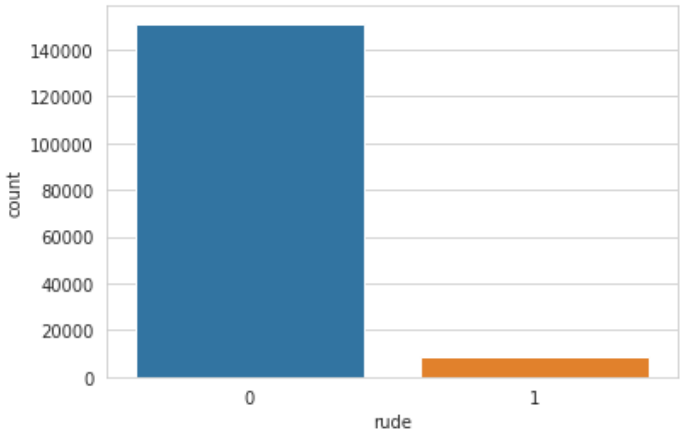
- Run and Evaluate selected models

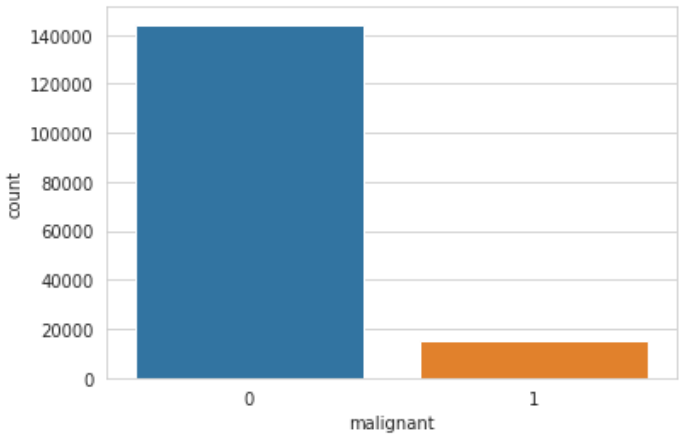
  Since the difference between the accuracy score and cross validation score is minimum in the case of random forest classifier and also its loss is very less. So, we will try hyperparameter tuning on Random Forest Classifier.

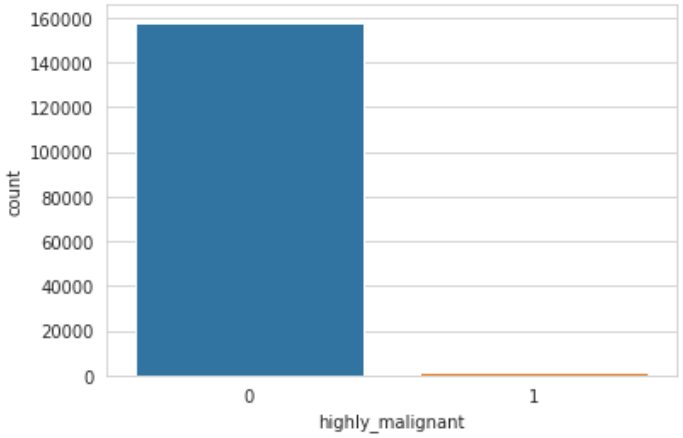
- Visualizations

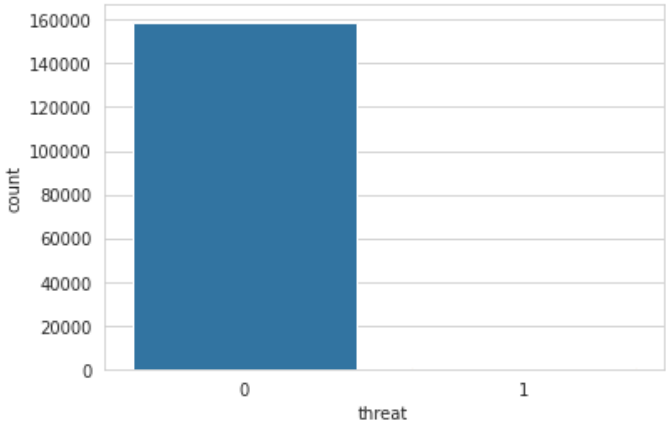
  The use of tables, graphs, and charts play a vital role in presenting the data being used to draw these conclusions. Thus, data visualization is the best way to explore the data as it allows in-depth analysis.

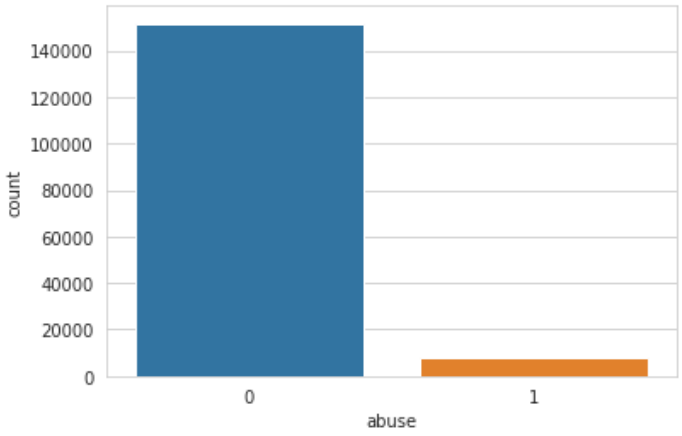
  a). Plotting countplot for all features:

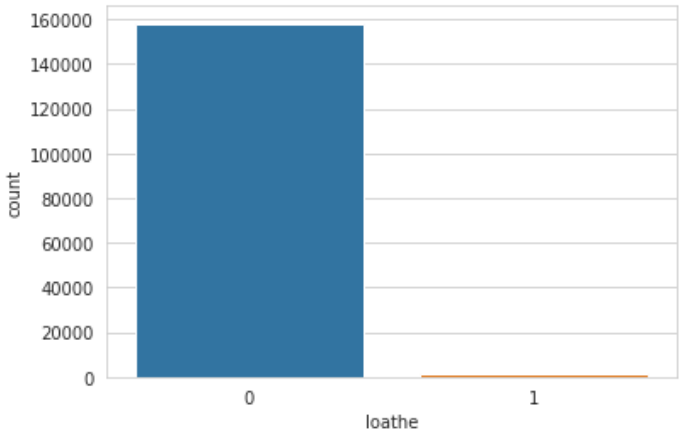

140000
120000
100000
80000
60000
40000
20000
0
count
0
1
rude


140000
120000
100000
80000
60000
40000
20000
0
count
0
1
malignant

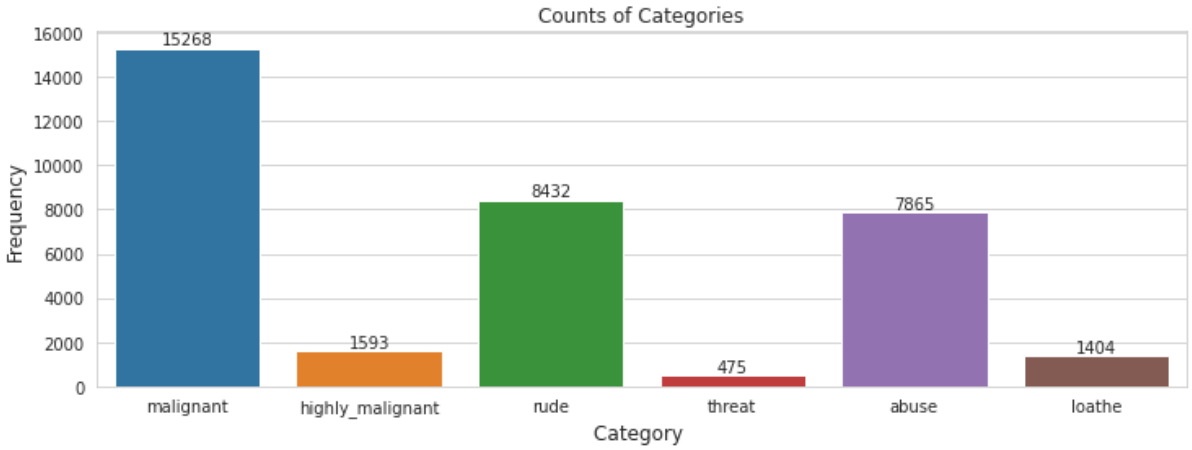

160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
highly_malignant


160000
140000
120000
100000
80000
60000
40000
20000
0
count
0
1
threat

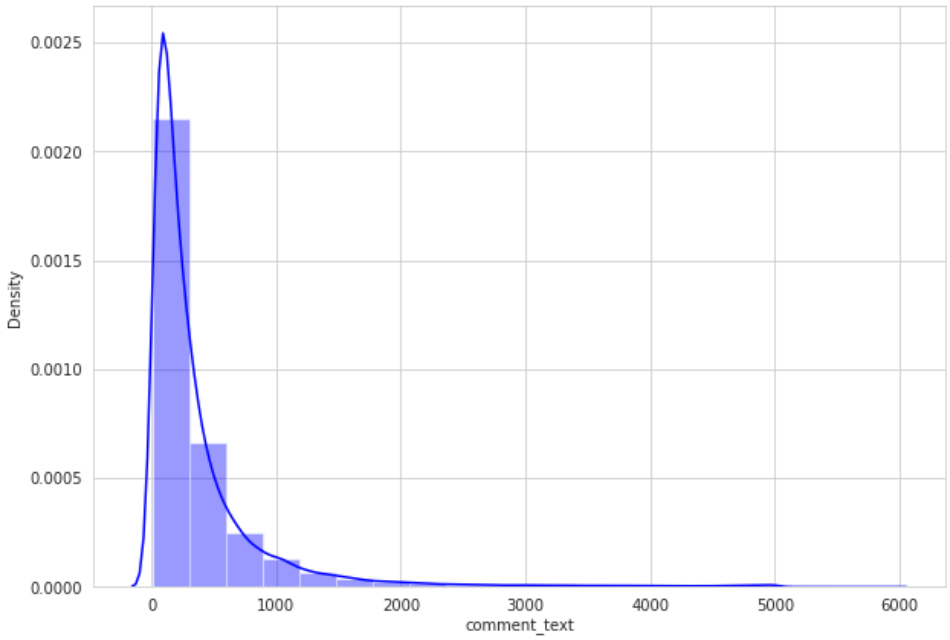




b) Plotting the counts for each category



c) Distribution of comments length



The graph shows the comment length frequency. The graph is right skewed. This means that most of the comments are short in length. The maximum length is 5000 words and minimum length is 5 words.

CONCLUSION

- Key Findings and Conclusions of the Study
  - After the completion of this project, we got an insight of how to process the data, analyse the data, and build a model.
  - First we imported both the training and testing data, both of them were huge in size.
  - We did all the required pre-processing steps like checking the null values, datatypes, dropping unnecessary coloumns etc.
  - We used the training data for doing EDA using various plots and recorded the observations.
  - While observing the results, we found that the dataset was in highly imbalanced side and we need to handle it, in order to avoid the problem of overfitting.
  - Using NLP, we pre-processed the comment text.
  - As the problem was a multi-label classifier, we took a new feature known as label and combined the comment_labels

output together using sum() and stored in that feature. For a binary classification problem, we scaled the data accordingly.

- After applying vectoriser, we used an oversampling technique called RandomOverSampler for handling the imbalanced data.
- We, then split the data using train_test_split and then we started the model building process by running as many algorithm in a for loop, with different metrics like cross_val_score, confusion_matrix, auc_score, log loss, hamming loss.
- We found that RandomForest Classifier is performing best. So, we applied hyperparameter tuning on it. We got 95% score after performing it.
- We saved the model and stored in csv file.

- Learning Outcomes of the Study in respect of Data Science

  More computational power was required as it took a lot of time to handle this much amount of data.

- Limitations of this work and Scope for Future Work

  With the help of this project, we were able to learn various NLP techniques like lemmatization, stemming, removal of stopwords. We also learnt how to convert strings into vectors through hash vectorizer. In this project, we applied different evaluation metrics like log loss, hamming loss along with accuracy.

  As this was the first project in this area, there are many improvisation or better techniques could have followed in order to make this a better project.