FLIP ROBO

Malignant Comment Classification

Submitted by:

Yatika Taneja

ACKNOWLEDGMENT

.

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.
I wish to thank, all the faculties in data trained academy as this project utilized knowledge gained from every course that formed the Data science program.

INTRODUCTION

Objective of the study

Cyberbullying is one of the biggest issues in the world of internet. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

The goal of the study 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.

Business Model

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.

Analytical Problem Framing

Data Sources

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.

Data Description

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.

Data Preprocessing

The fundamental steps involved in data pre-processing are,:

1. Cleaning the raw data
2. Tokenizing the cleaned data

- ### Cleaning the Raw Data

This phase involves the deletion of words or characters that do not add value to the meaning of the text. Some of the standard cleaning steps are listed below:

- **Lowering case**
- **Removal of special characters**
- **Removal of stop words**
- **Removal of hyperlinks**
- **Removal of numbers**
- **Removal of whitespaces**

Lowering Case

Lowering the case of text is essential for the following reasons:

- The words, ‘TEXT’, ‘Text’, ‘text’ all add the same value to a sentence
- Lowering the case of all the words is very helpful for reducing the dimensions by decreasing the size of the vocabulary

```
train['comment_text']=train['comment_text'].str.lower()
```

```
train.head()
```

Removal of special characters

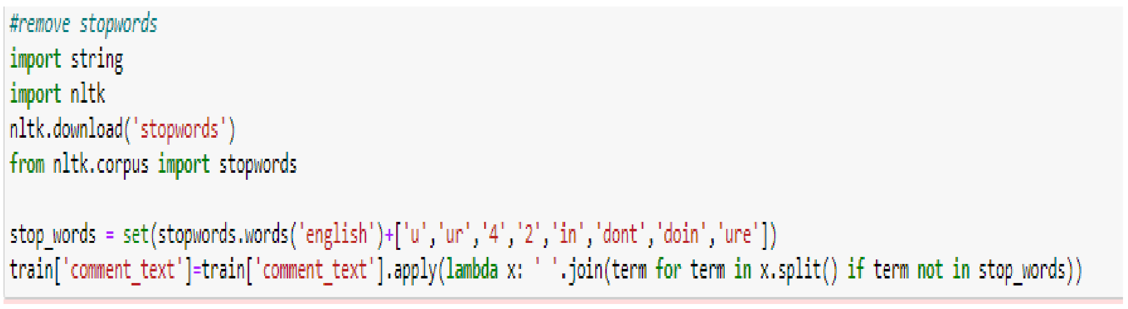
```
#remove punctuation
train['comment_text']=train['comment_text'].str.replace(r'[^\w\d\s]',' ')

#replace whitespace with a single space
train['comment_text']=train['comment_text'].str.replace(r'\s+',' ')

#remove leading and trailing whitespace
train['comment_text']=train['comment_text'].str.replace(r'^\s+|\s+?$','')
```

```
# remove '\\n'
train['comment_text'] = train['comment_text'].str.replace('\\n',' ')

# remove any text starting with User...
train['comment_text'] = train['comment_text'].str.replace("\[\[User.*",'')

# remove IP addresses or user IDs
train['comment_text'] = train['comment_text'].str.replace("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}",'')

#remove http links in the text
train['comment_text'] = train['comment_text'].str.replace("(http://.*?\s)|(http://.*)",'')
```

Removal of stop words

Stop words are commonly occurring words in a language like ‘the’, ‘a’, and so on.

Most of the time they can be removed from the text because they don’t provide valuable information.

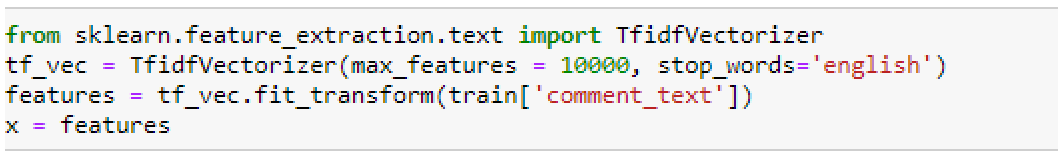


```
#remove stopwords
import string
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english')+['u','ur','4','2','in','dont','doin','ure'])
train['comment_text']=train['comment_text'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```

- **Tokenizing the Cleaned Data**

Tokenization is the process of splitting text into smaller chunks, called tokens.

Each token is an input to the machine learning algorithm as a feature.



```
from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(train['comment_text'])
x = features
```

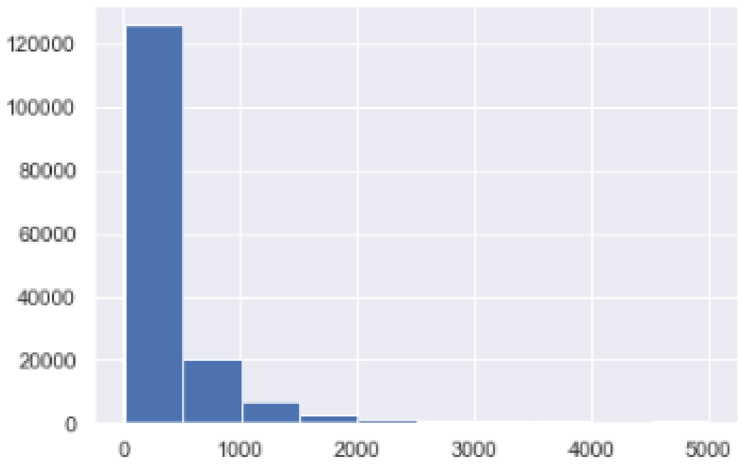
Tools Used

Python 3.7.2,
Jupyter Notebook
Numpy
Pandas
Matplotlib
Seaborn
Scikit-learn
Scipy
NLTK

Data Visualization

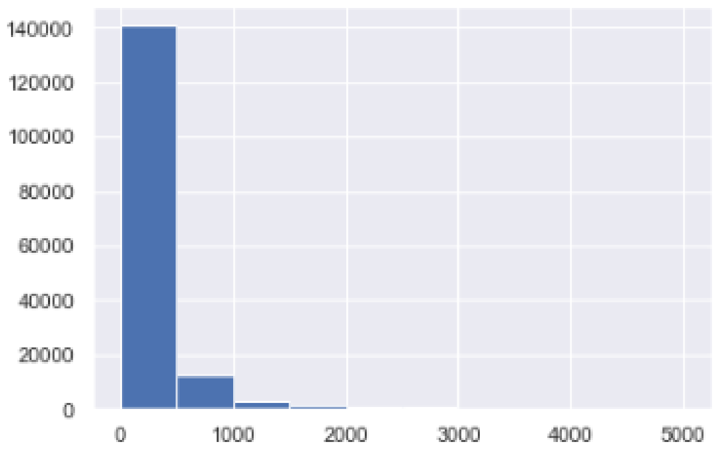
1. Comment distribution before cleaning

```
#  histogram plot for original length
sns.set()
train['length'].hist()
plt.show()
```

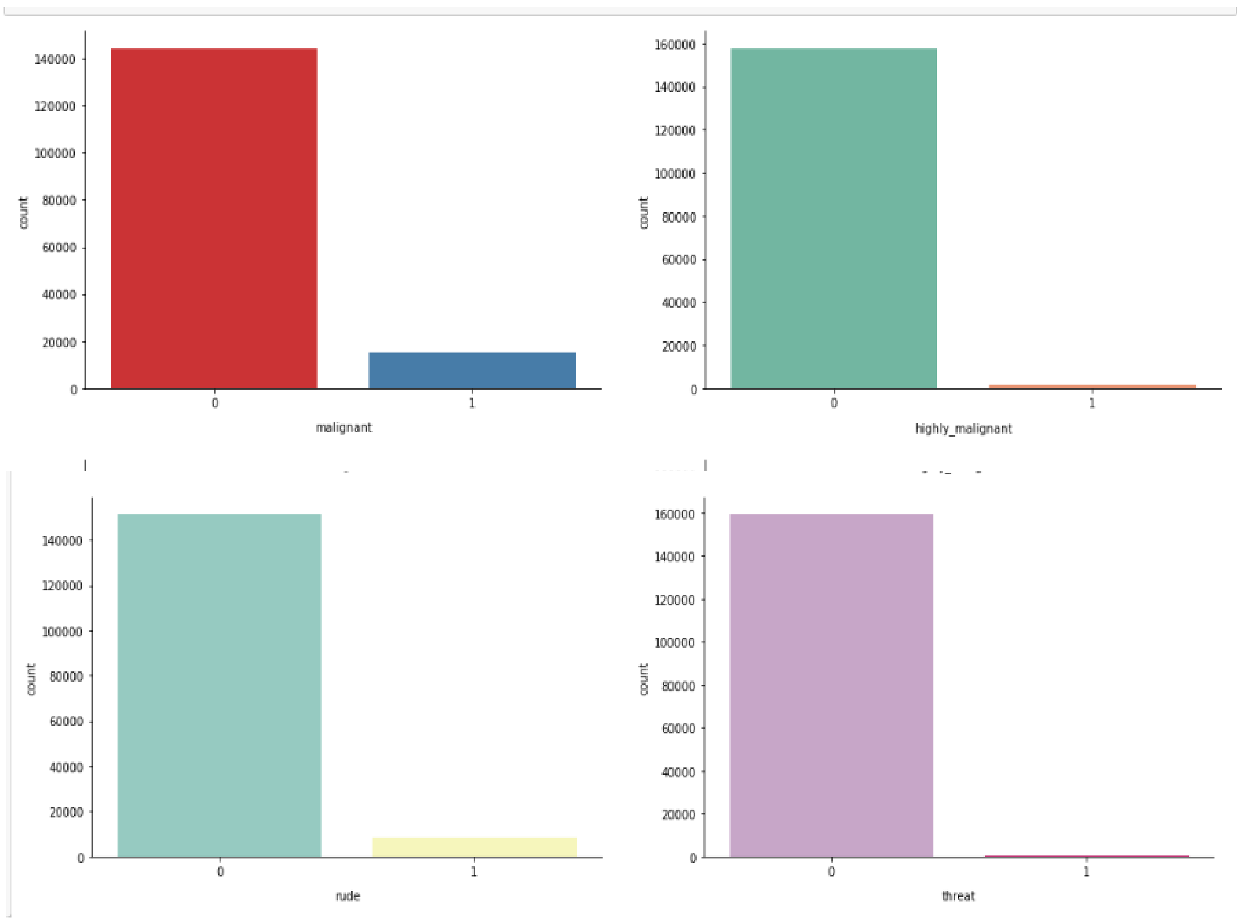


2. Comment distribution after cleaning

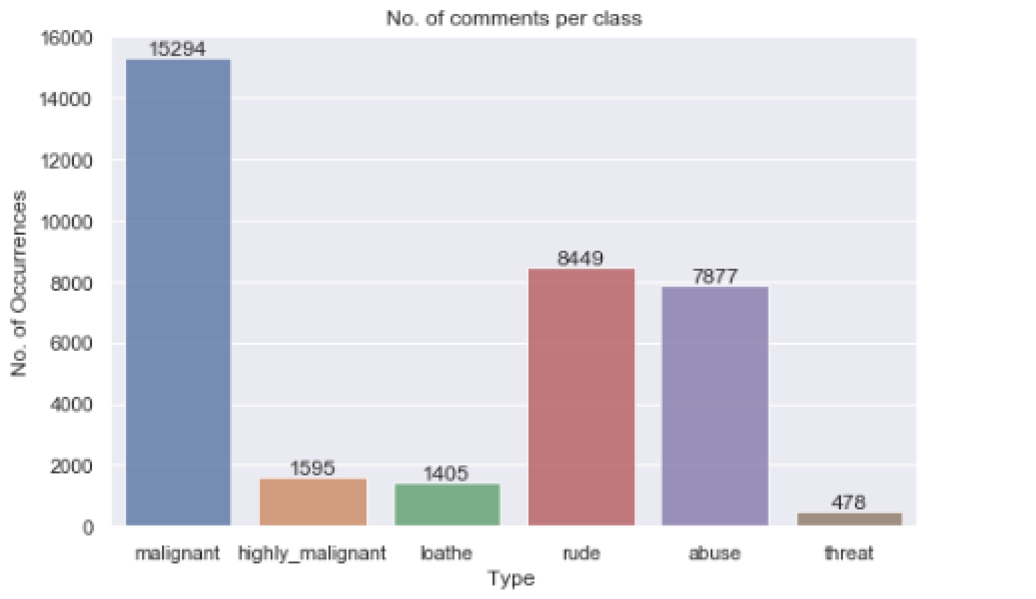
```
#  histogram plot for cleaned length
sns.set()
train['clean_length'].hist()
plt.show()
```



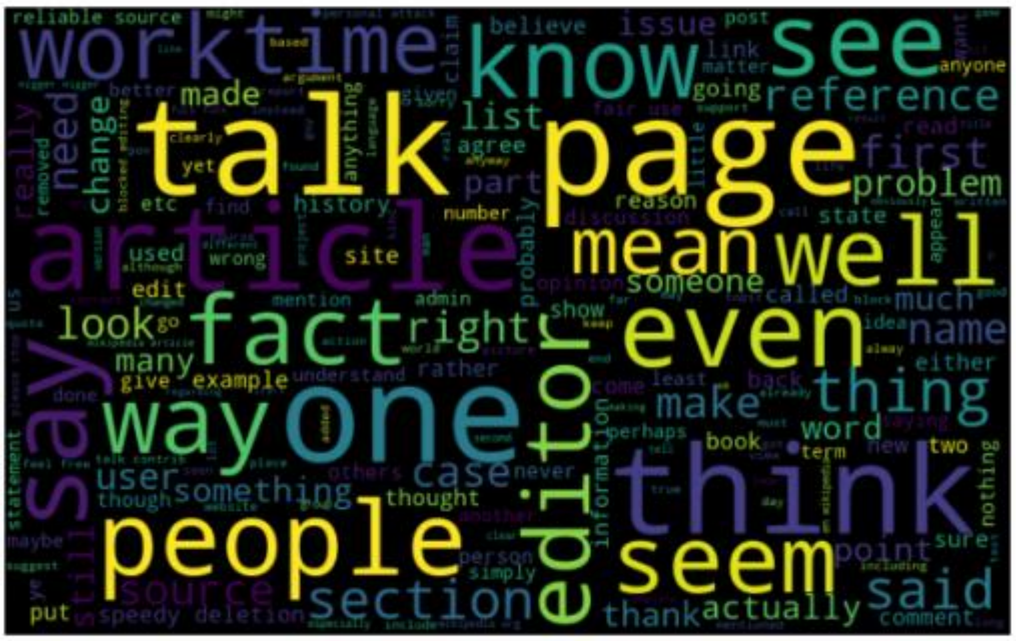
3. Count plot for each class



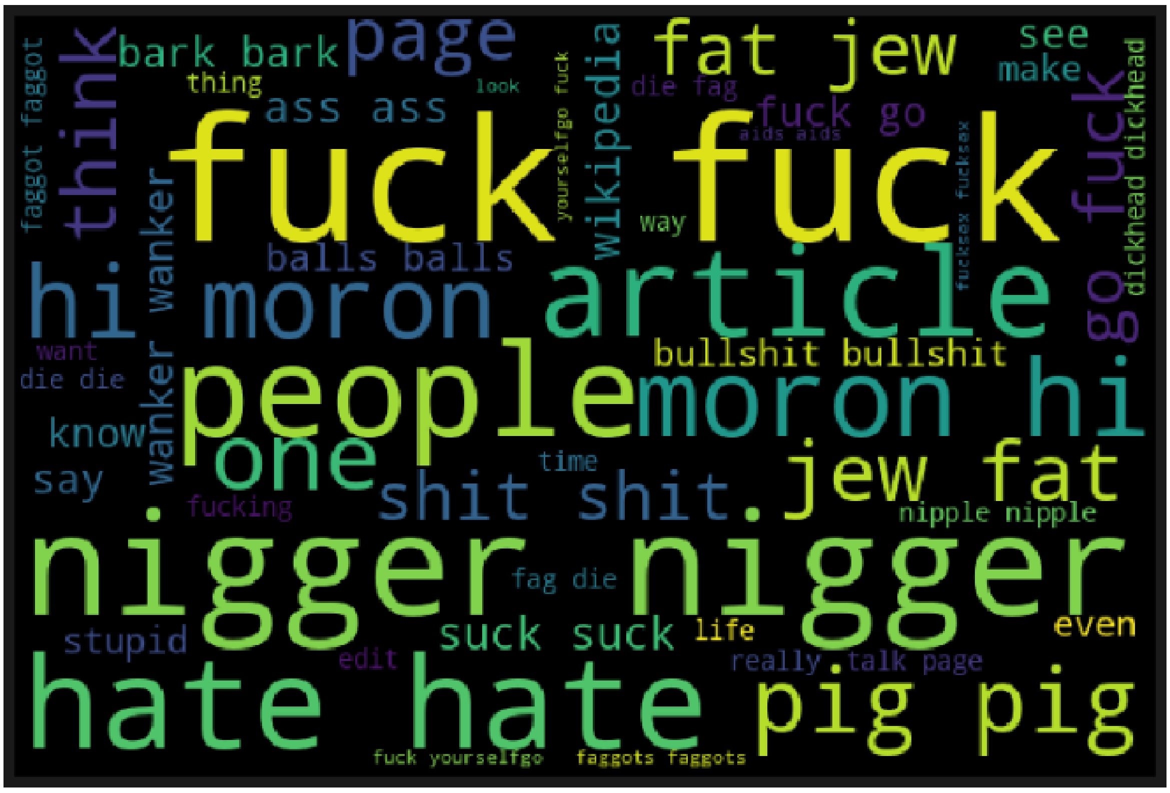
4. No. of Comments per Class



5. Word cloud for all comments



6. Word Cloud for malign comments



Model/s Development and Evaluation

Models Applied

Malignant Comment Classification is a classification problem. Some algorithms like Naive Bayes Classifier, Decision Trees work well for malign comment classification.
Algorithms like KNN, Linear Regression don't really work well due to inherent disadvantages such as curse of dimensionality.

Naive Bayes

Naive Bayes is the easiest classification algorithm (fast to build, regularly used for spam detection). It is a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features.

Why use Naive Bayes?

1. NB is very simple, easy to implement and fast because essentially you're just doing a bunch of counts.
2. If the NB conditional independence assumption holds, then it will converge quicker than discriminative models like logistic regression.
3. NB needs works well even with less sample data.
4. NB is highly scalable. It scales linearly with the number of predictors and data points.
5. NB can be used for both binary and multi-class classification problems and handles continuous and discrete data .
6. NB is not sensitive to irrelevant features.

Decision Trees

Decision trees are used for classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria is different for classification and regression trees. Information theory is a measure to define this degree of disorganization in a system known as Entropy. If the sample is completely homogeneous, then the entropy is zero and if the sample is an equally divided (50% – 50%), it has entropy of one. It chooses the split which has lowest entropy compared to parent node and other splits. The lesser the entropy, the better it is.

Why use Decision Trees?

1. Decision trees implicitly perform variable screening or feature selection. When we fit a decision tree to a training dataset, the top few nodes on which the tree is split are essentially the most important variables within the dataset and feature selection is completed automatically
2. Decision trees are easy to understand, easy to represent visually and easy to communicate.
3. Nonlinear relationships between parameters do not affect tree performance. Also trees can explain the non-linearity in an intuitive manner.

Random Forest

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observation and 5 randomly chosen initial variables to build a CART model. It will repeat the process 10 times and then make a final prediction on each observation. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction. Random forest gives much more accurate predictions when compared to simple CART/CHAID or regression models in many scenarios. These cases generally have high number of predictive variables and huge sample size. This is because it captures the variance of several input variables at the same time and enables high number of observations to participate in the prediction.

Model Evaluation

Metrics Used:

Following are the metrics we used to evaluate the performance of ML techniques:

1. Precision

Precision refers to the closeness of two or more measurements to each other. In Machine Learning, precision is the fraction of relevant instances among the retrieved instances. Precision = TP / (TP + FP) (Where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative).

2. Accuracy

Accuracy refers to the closeness of a measured value to a standard or known value. Accuracy = (TP+TN) / ALL

3. Recall

Recall is how many of the true positives were recalled (found), i.e. how many of the correct hits were also found. Recall = TP / (TP + FN)

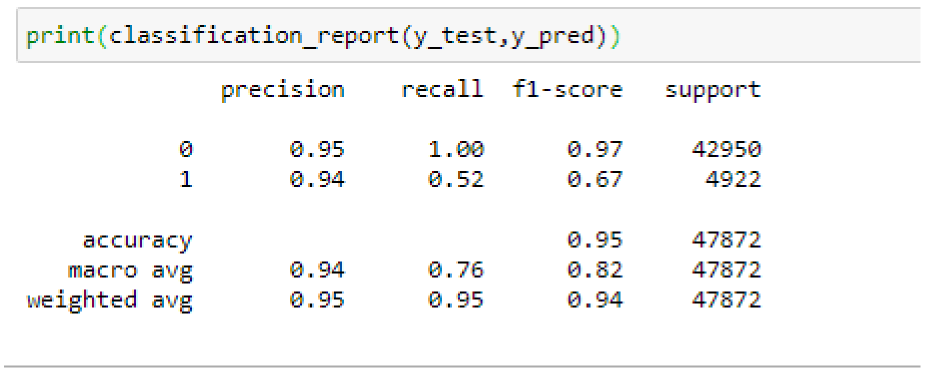
4. F-Score

F-scores are a statistical method for determining accuracy accounting for both precision and recall. It is essentially the harmonic mean of precision and recall.

Comparison of Performance of the Models:

1. Naive Bayes

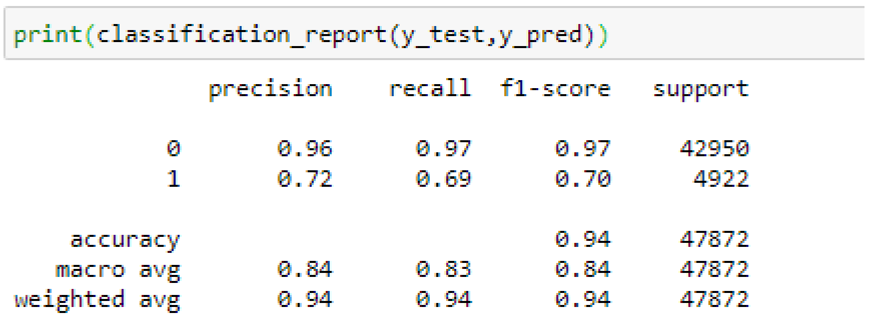
For this problem, Naive Bayes algorithm did not work well. Following were the confusion matrix and scores:



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

              precision    recall  f1-score   support

           0       0.95      1.00      0.97     42950
           1       0.94      0.52      0.67      4922

    accuracy                           0.95     47872
   macro avg       0.94      0.76      0.82     47872
weighted avg       0.95      0.95      0.94     47872
```

High accuracy suggests that the model is very good at correctly classifying the malign comments. Precision value is also good at 0.85, means the model has a low false positive rate. But the model has a low recall value. This indicates that results are complete to a large extent. In other words, probability of detection is low. Together, having a high precision and low recall means that the most of non malign predictions are correct, but the model is not predicting all the malign in the test data.

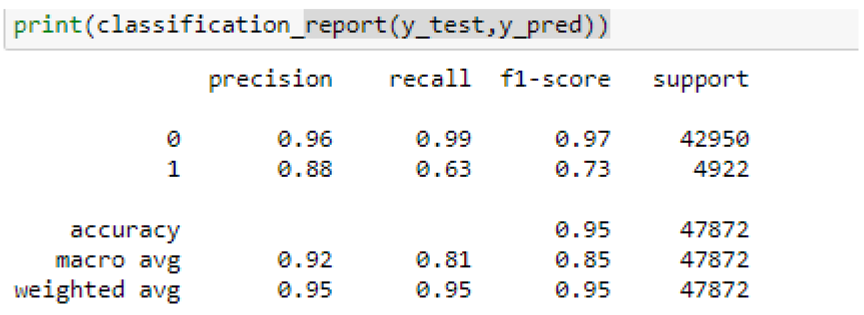
3. Decision Tree



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

              precision    recall  f1-score   support

           0       0.96      0.97      0.97     42950
           1       0.72      0.69      0.70      4922

    accuracy                           0.94     47872
   macro avg       0.84      0.83      0.84     47872
weighted avg       0.94      0.94      0.94     47872
```

Precision, Recall and F-Score of the decision tree are higher than Naive Bayes model. But accuracy is lower compared to Naïve Bayes Model.

4. Random Forest



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

              precision    recall  f1-score   support

           0       0.96      0.99      0.97     42950
           1       0.88      0.63      0.73      4922

    accuracy                           0.95     47872
   macro avg       0.92      0.81      0.85     47872
weighted avg       0.95      0.95      0.95     47872
```

Random Forest algorithm had a highest precision, accuracy, recall and f-score out of 3. Having high precision and recall suggests that the model is correctly predicting positive class.

It follows that the model also has a highest F-score, since it is directly proportional to Precision and Accuracy.

Result:

Since Random Forest Classifier have highest accuracy, precision, recall and f-score. It is the best fit for Malign Comment Classification.

CONCLUSION

In this study, a Malign Comment Classification system is introduced which makes use of a NLP and Machine Learning for its implementation.
The classification algorithms used in this approach are Naïve Bayes, Decision Tree Classifier and Random Forest Classifier.
It was observed that Random forest Classifier is the best fit for Malign Comment Classification system with 95% accuracy.
..