
# Malignant Comment Classifier PROJECT

Submitted by:

Vikram Purohit

# ACKNOWLEDGMENT

I express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project on Malignant Comment Classifier using machine learning algorithms and NLTK suite of libraries and also, for providing me with the requisite datasets for training and testing prediction accuracies of the models. I acknowledge my indebtedness to the authors of the papers titled: “Toxic Comment Classification” and “Machine learning methods for toxic comment classification: a systematic review” for providing me with invaluable knowledge and insights into what constitute as malignant and benign comments and the role of natural language processing tools and techniques in identifying them and in helping build models to classify input comments as malignant and benign.

# INTRODUCTION

## • Business Problem Framing

With the proliferation of social media there has been an emergence of conflict and hate, making online environments uninviting for users. 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. 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

Predictive modelling, Classification algorithms are some of the machine learning techniques used along with the various libraries of the NLTK suite for Classification of comments.

Using NLTK tools, the frequencies of malignant words occurring in textual data were estimated and given appropriate weightage, whilst filtering out words, and other noise which do not have any impact on the semantics of the comments and reducing the words to their base lemmas for efficient processing and accurate classification of the comments.

## • Review of Literature

Two research papers titled: “Toxic Comment Classification” by Sara Zaheri and “Machine learning methods for toxic comment classification: a systematic review” by Darko Androcec were reviewed and studied to gain insights into the nature of malignant comments, their impact on social media platforms and the various methods that are employed for training models to detect, identify and classify them.

## • 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.

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. Automatic recognition of malignant comments on online forums, and social media serves as a useful provision for moderators of public platforms as well as users who could receive warnings and filter unwanted contents. The need of advanced methods and techniques to improve identification of different types of comments posted online motivated the current project.

# Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Various Classification analysis techniques were used to build Classification models to determine whether an input Message content is benign or malignant. Machine Learning Algorithms such as Multinomial Naïve Bayes and Complement Naïve Bayes were employed which are based on the Bayes Theorem:

P(message is malignant | message content) = P(message content | malignant). P(malignant) / P(message content)

The probability of message being Malignant, knowing that Message Content has occurred could be calculated. Event of "Message Content" represents the evidence and "Message is Malignant", the hypothesis to be approved. The theorem runs on the assumption that all predictors/features are independent and the presence of one would not affect the other.

The approach to classify a comment as malignant would depend on training data labelled as various categories of malignant messages and benign messages.

# • Data Sources and their formats

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.

| | 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 |

## The data set includes:

*Figure 1 Train Dataset*

| | id | comment_text |
|---|---|---|
| 0 | 00001cee341fdb12 | Yo bitch Ja Rule is more succesful then you'll... |
| 1 | 0000247867823ef7 | == From RfC == \n\n The title is fine as it is... |
| 2 | 00013b17ad220c46 | " \n\n == Sources == \n\n * Zawe Ashton on Lap... |
| 3 | 00017563c3f7919a | :If you have a look back at the source, the in... |
| 4 | 00017695ad8997eb | I don't anonymously edit articles at all. |

*Figure 2 Test Dataset*

- **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 Done

  The dataset was checked to see if there were any null values or random characters present. None were found.

Column: ID was dropped since they don't contribute to building a good model for predicting the target variable values.

The train and test dataset contents were then converted into lowercase. Punctuations, unnecessary characters etc were removed, currency symbols, phone numbers, web urls, email addresses etc were replaced with single words. Tokens that contributed nothing to semantics of the messages were removed as Stop words. Finally retained tokens were lemmatized using WordNetLemmatizer().

The string lengths of original comments and the cleaned comments were then compared.

```
# Convert all messages to lower case
trainDF['comment_text'] = trainDF['comment_text'].str.lower()


trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress') # Replace email add

# Replace URLs with 'webaddress'
trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','webaddres


trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'£|\$', 'dollars')# Replace money symbols with 'moneysymb'

# Replacing 10 digit phone numbers with 'phonenumber'
trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'\d+(\.\d+)?','num') # Replace numbers with 'num'


trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'[^\w\d\s]',' ') #removing punctuations

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'[\_]',' ') #removing underscore characters

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'\s+[a-zA-Z]\s+', ' ') #removing single characters

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'\s+', ' ') #removing whitespace between terms with a single

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

```

```
trainDF.head()
```

| | comment_text | malignant | highly_malignant | rude | threat | abuse | loathe | Stringlength |
|---|---|---|---|---|---|---|---|---|
| 0 | explanation why the edits made under my userna... | 0 | 0 | 0 | 0 | 0 | 0 | 264 |
| 1 | d aww he matches this background colour m seem... | 0 | 0 | 0 | 0 | 0 | 0 | 112 |
| 2 | hey man m really not trying to edit war it jus... | 0 | 0 | 0 | 0 | 0 | 0 | 233 |
| 3 | more can make any real suggestions on improve... | 0 | 0 | 0 | 0 | 0 | 0 | 622 |
| 4 | you sir are my hero any chance you remember wh... | 0 | 0 | 0 | 0 | 0 | 0 | 67 |

```
import nltk
from nltk.corpus import stopwords,wordnet
```

```
from nltk.stem import WordNetLemmatizer
```

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

```
lem=WordNetLemmatizer()
trainDF['comment_text'] = trainDF['comment_text'].apply(lambda x: ' '.join(lem.lemmatize(t) for t in x.split()))
```

```
trainDF['Cleaned_Stringlength'] = trainDF['comment_text'].str.len()
trainDF.head()
```

| | comment_text | malignant | highly_malignant | rude | threat | abuse | loathe | Stringlength | Cleaned_Stringlength |
|---|---|---|---|---|---|---|---|---|---|
| 0 | explanation edits made username hardcore metal... | 0 | 0 | 0 | 0 | 0 | 0 | 264 | 164 |
| 1 | aww match background colour seemingly stuck th... | 0 | 0 | 0 | 0 | 0 | 0 | 112 | 83 |
| 2 | hey man really trying edit war guy constantly ... | 0 | 0 | 0 | 0 | 0 | 0 | 233 | 141 |
| 3 | make real suggestion improvement wondered sect... | 0 | 0 | 0 | 0 | 0 | 0 | 622 | 364 |
| 4 | sir hero chance remember page | 0 | 0 | 0 | 0 | 0 | 0 | 67 | 29 |

- Data Inputs- Logic- Output Relationships

The comment tokens so vectorised using TfidVectorizer are input and classified as benign(0) or malignant(1) as output by classification models.

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

  The comment content made available in Train and Test Dataset is assumed to be written in English Language in the standard Greco-Roman script. This is so that the Stopword package and WordNetLemmatizer can be effectively used.

- Hardware and Software Requirements and Tools Used Hardware Used:
    - Processor: 2.9 GHz Dual-Core Intel Core i5
    - Physical Memory: 8 GB 2133 MHz LPDDR3
    - GPU:Intel Iris Graphics 550 1536 MB
    - Software Used: Anaconda Package and Environment Manager
    - MacBook Pro (13-inch, 2016, Four Thunderbolt 3 Ports)

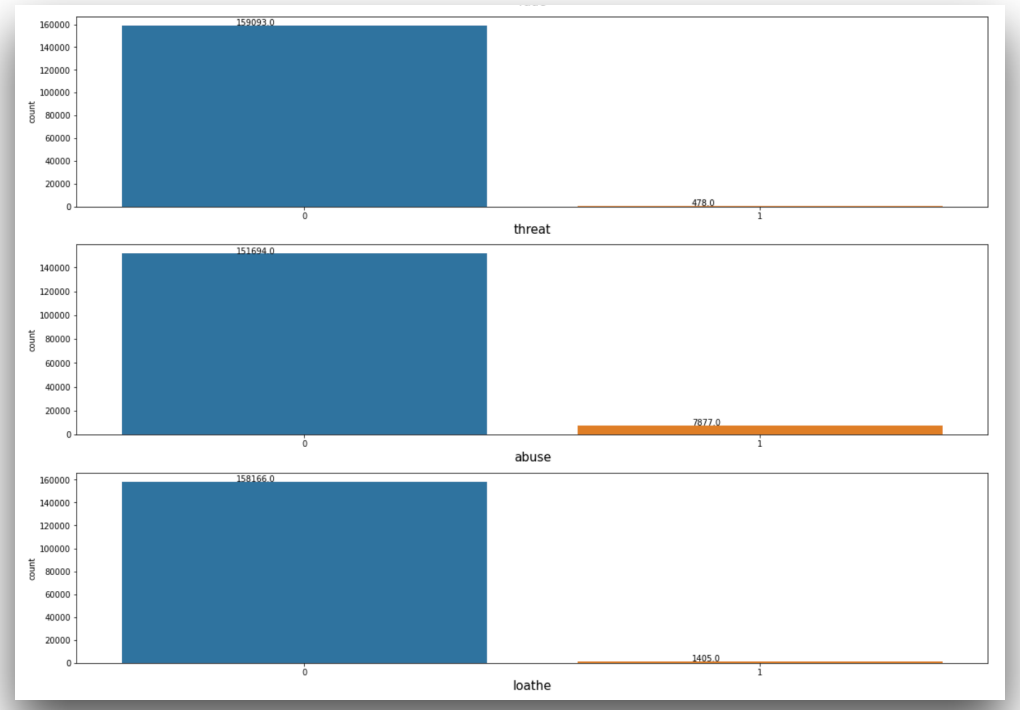
✦**Python Libraries used:**

- Pandas: For carrying out Data Analysis, Data Manipulation, Data Cleaning etc
- Numpy: For performing a variety of operations on the datasets.
- matplotlib.pyplot, Seaborn: For visualizing Data and various relationships between Feature and Label Columns
- sklearn for Modelling Machine learning algorithms, Evaluation metrics, Data Transformation etc
- imblearn.over_sampling: To employ SMOTE technique for balancing out the classes. o re, string: To perform regex operations
- Wordcloud: For Data Visualization
  NLTK: To use various Natural Language Processing Tools.

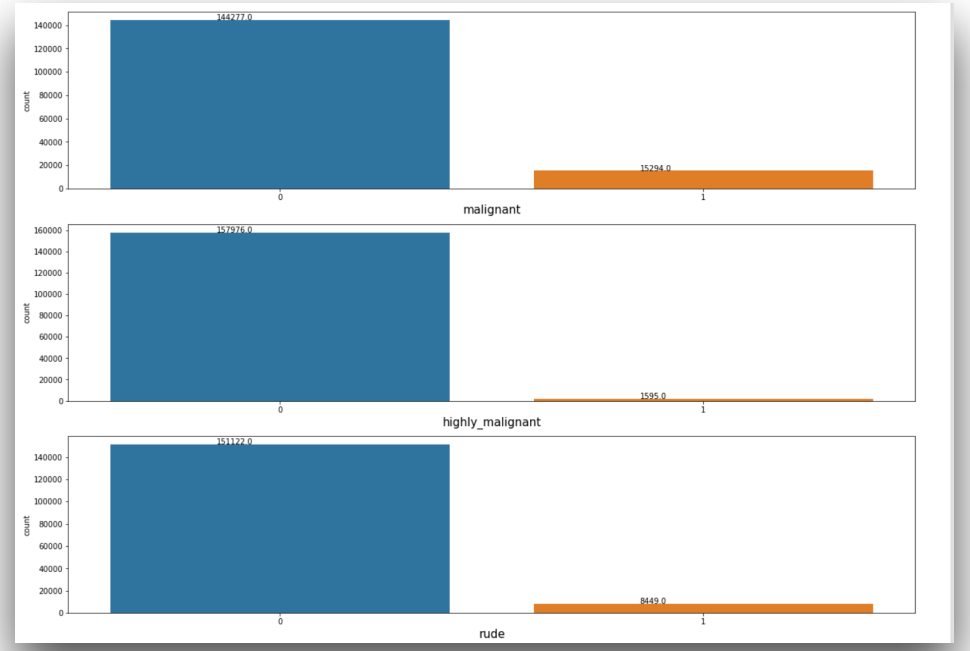
# Exploratory Data Analysis Visualizations

- ✦Barplots
- ✦Countplots
- ✦Distplots
- ✦WordClouds were used to visualise the data of all the columns and their relationships with Target variable.

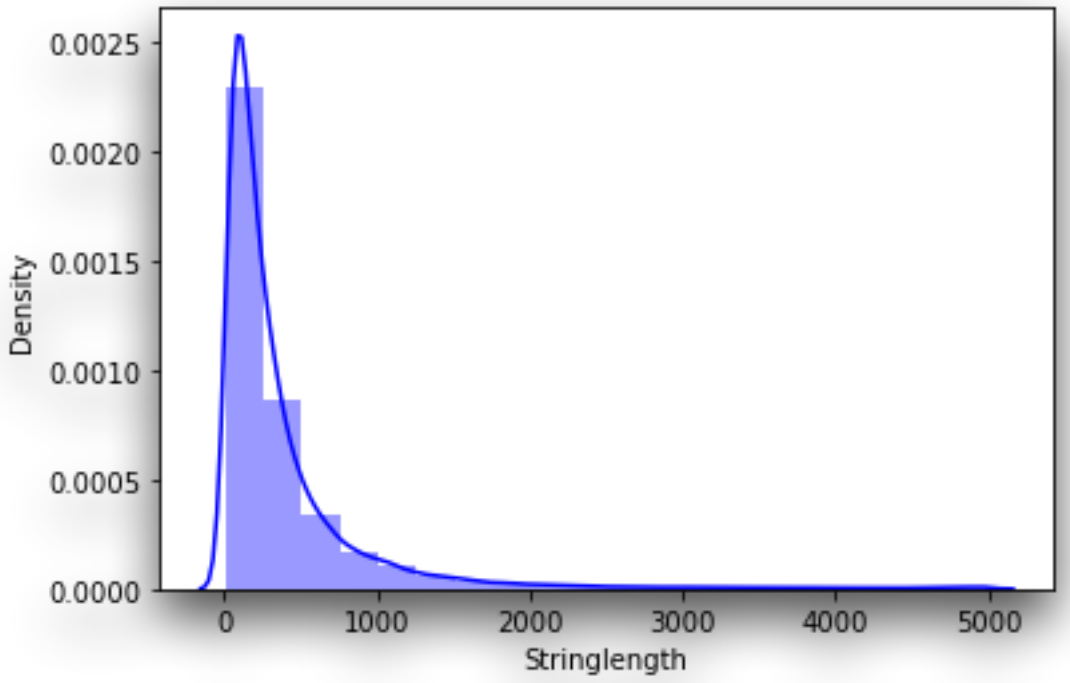
## Analyzing the Feature Columns

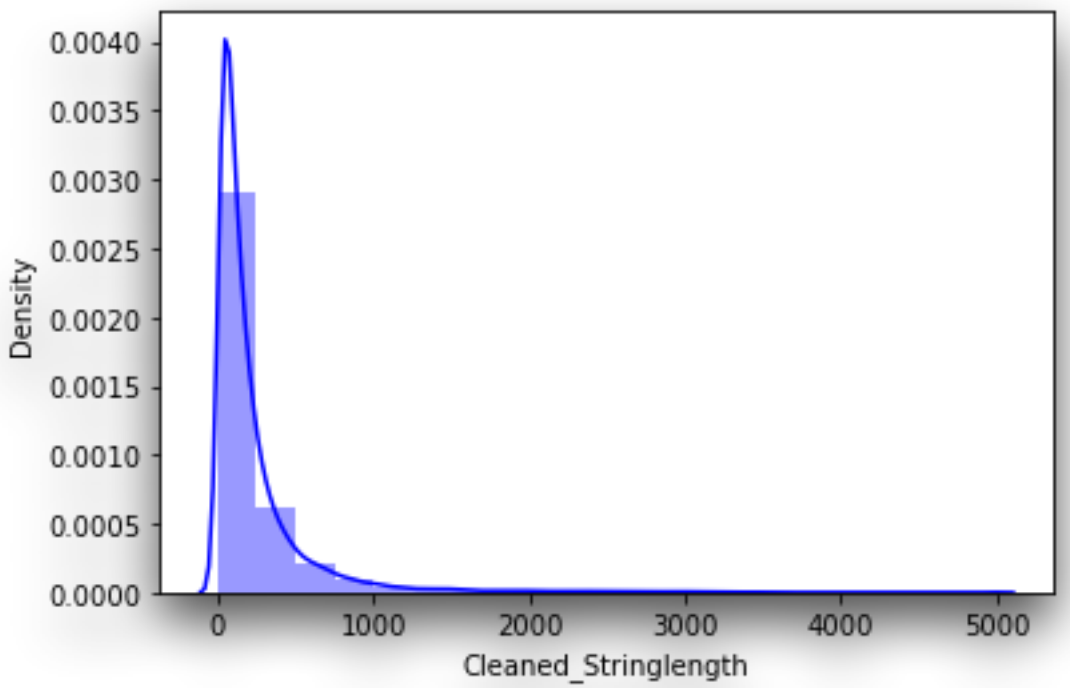



From the graphs about it is observed that majority of the comments are benign.

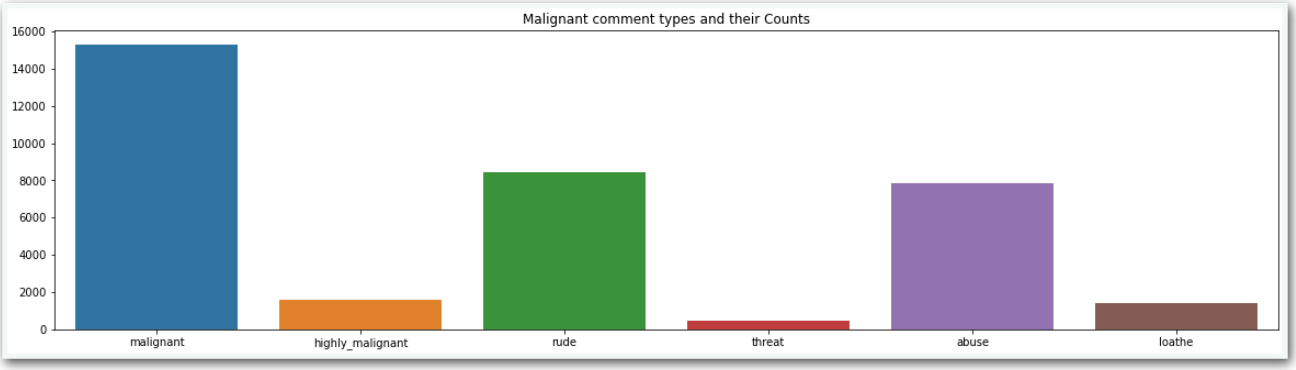



# Unprocessed vs Cleaned string lengths






Above graphs show that the string length of comments was drastically brought down after processing.




The above graph shows the composition of toxic comments, of which majority are malignant followed by rude comments, abusive comments, highly malignant comments, hateful comments and threats.

# Word Clouds of the most frequent words under various categories of Malignant Comments









# Feature Engineering

The comments data could belong to more than one label simultaneously(rude comments are at the same time malignant and in some cases can also be deemed hateful, abusive comments are hateful and can be highly malignant at the same time, threats are highly malignant too etc.)

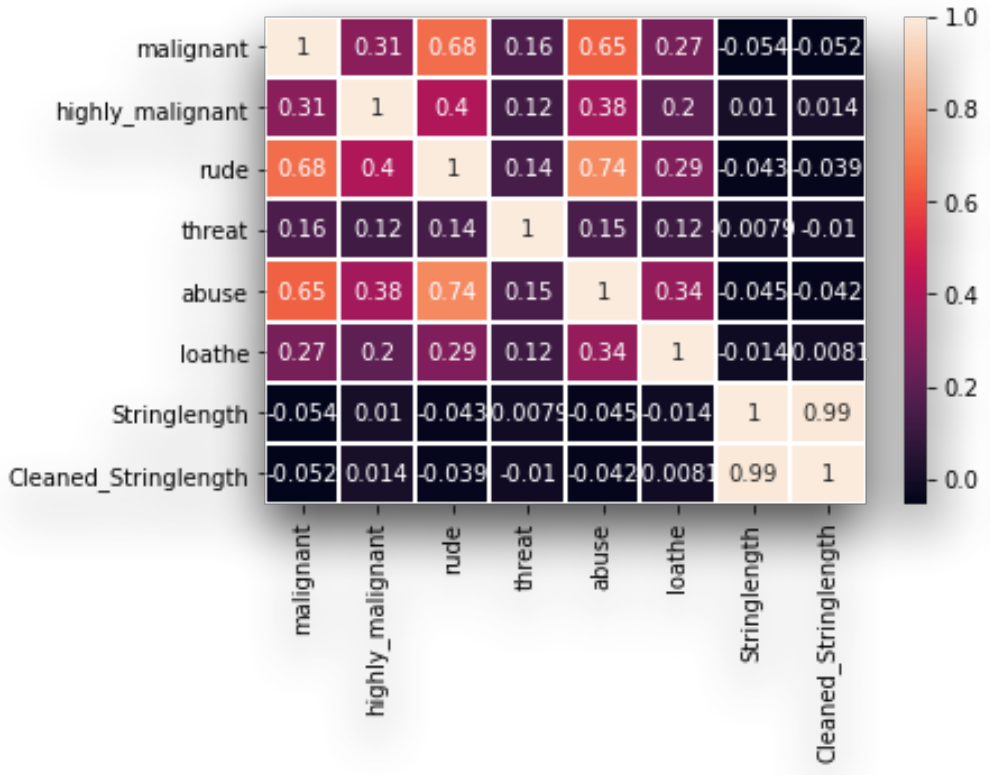
Since each of the categories had very small data available to work with, a new column: ‘comment_type’ was created which only had binary classes: 0 which represented all the benign comments and 1 which represented all the comments which fell under malignant, highly malignant, abusive, hateful, rude, threat features. This column acted as Target Label column for malignant comment classification.

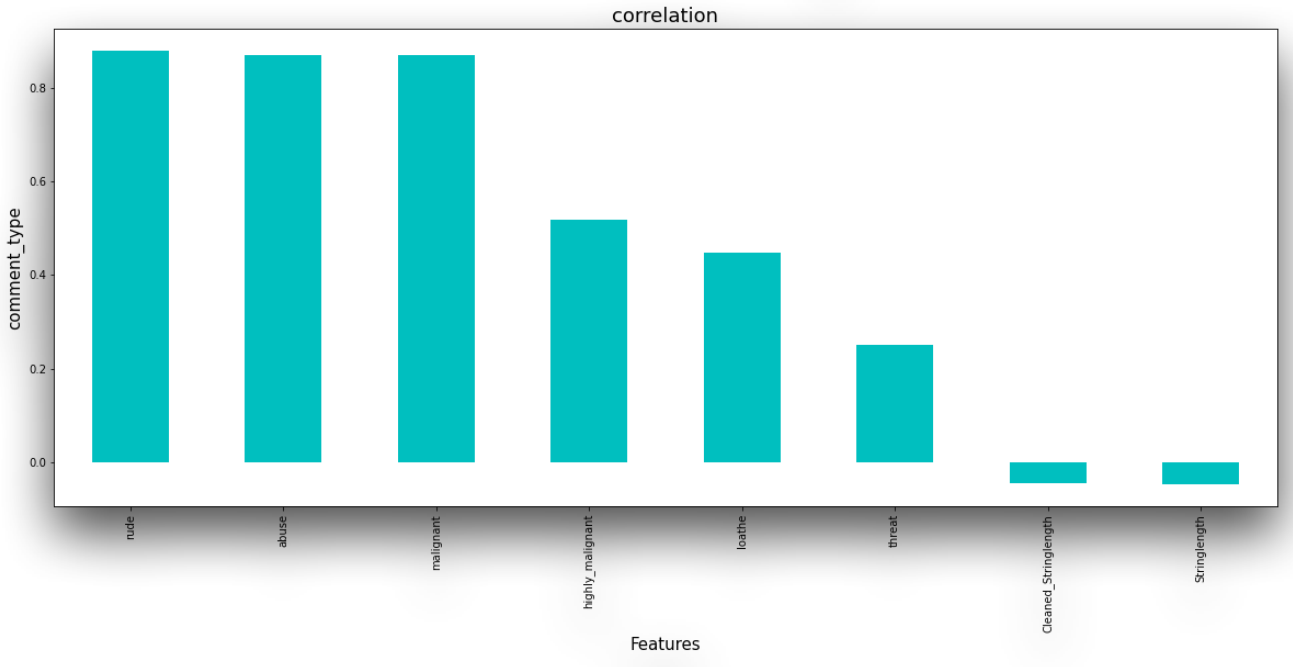
# Visualising data in Target column

The classes appear to be imbalanced with 90% of comments being benign (0) and only 10% being malignant (1).

Smote Technique was used to balance out the classes

# Finding Correlation

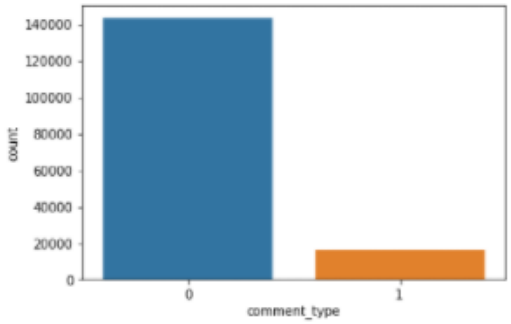


From the graphs above it is observed that columns: Rude,Abuse, Malignant have highest positive correlation with comment_type.

```
1 sns.countplot(trainDF['comment_type'])
```

```
<AxesSubplot:xlabel='comment_type', ylabel='count'>
```




```
1 print('Benign comment ratio = ',round(len(trainDF[trainDF['comment_type']==0])/len(trainDF.comment_type),2)*100,'%')
```

```
Benign comment ratio =  90.0 %
```

```
1 print('Malignant comment ratio = ',round(len(trainDF[trainDF['comment_type']==1])/len(trainDF.comment_type),2)*100,'%')
```

```
Malignant comment ratio =  10.0 %
```

**Classes are imbalanced**

## Balancing out classes in Label column using SMOTE technique.

```
1 from imblearn.over_sampling import SMOTE as sm
2
3 smt_x,smt_y = sm().fit_resample(X,y)
```

# Model/s Development and Evaluation

☑ Identification of possible problem-solving approaches (methods)
The model algorithms used were as follows:

- **Logistic Regression**: It is a classification algorithm used to find the probability of event success and event failure. It is used when the dependent variable is binary(0/1, True/False, Yes/No) in nature. It supports categorizing data into discrete classes by studying the relationship from a given set of labelled data. It learns a linear relationship from the given dataset and then introduces a non- linearity in the form of the Sigmoid function. It not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative).

**Logistic Regression**

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import log_loss

lo=LogisticRegression()
lo.fit(x_train,y_train)
lo.score(x_train,y_train)
predlo=lo.predict(x_test)

print('Accuracy Score is''\n',accuracy_score(y_test,predlo))
print('Confusion Matrix''\n',confusion_matrix(y_test,predlo))
print(classification_report(y_test,predlo))
print('Log loss : ', log_loss(y_test,predlo))
```

```
Accuracy Score is
 0.935852107875002
Confusion Matrix
 [[29084  2539]
 [ 1507 29943]]
              precision    recall  f1-score   support

           0       0.95      0.92      0.93     31623
           1       0.92      0.95      0.94     31450

    accuracy                           0.94     63073
   macro avg       0.94      0.94      0.94     63073
weighted avg       0.94      0.94      0.94     63073

Log loss :  2.2156218900588827
```

- **Multinomial Naïve Bayes Classifier**: Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

**Multinomial Naive Bayes**

```
from sklearn.naive_bayes import MultinomialNB

mnb=MultinomialNB()
mnb.fit(x_train,y_train)
pred_mnb=mnb.predict(x_test)
print('Accuracy score''\n',accuracy_score(y_test,pred_mnb))
print('Confusion Matrix''\n',confusion_matrix(y_test,pred_mnb
print('Classification Report''\n',classification_report(y_tes
print('Log loss : ', log_loss(y_test,pred_mnb))
```

```
Accuracy score
 0.9112615540722655
Confusion Matrix
 [[28789  2834]
 [ 2763 28687]]
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     31623
           1       0.91      0.91      0.91     31450

    accuracy                           0.91     63073
   macro avg       0.91      0.91      0.91     63073
weighted avg       0.91      0.91      0.91     63073

Log loss :  3.0649532690918826
```

- **Gradient Boosting Classifier** : Gradient Boosting Classifier uses decision trees as base learners; combining many weak learners to make a strong learner. As a result it is referred to as an ensemble

**Gradient Boosting Classifier**

```
from sklearn.ensemble import GradientBoostingClassifier

gbk = GradientBoostingClassifier()
gbk.fit(x_train, y_train)
pred_gbk = gbk.predict(x_test)
print('Accuracy score''\n',accuracy_score(pred_gbk,y_test))
print('Confusion Matrix''\n',confusion_matrix(y_test,pred_gbk))
print('Classification Report''\n',classification_report(y_test,pred_gbk))
print('Log loss : ', log_loss(y_test,pred_gbk))
```

```
Accuracy score
 0.8319566216923248
Confusion Matrix
 [[25553  6070]
 [ 4529 26921]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.81      0.83     31623
           1       0.82      0.86      0.84     31450

    accuracy                           0.83     63073
   macro avg       0.83      0.83      0.83     63073
weighted avg       0.83      0.83      0.83     63073

Log loss :  5.804089619426056
```

learning method since it uses the output of many models in the final prediction. It uses the power of parallel processing and supports regularization.

- **RandomForestClassifier**: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A random forest produces good predictions that can be understood easily. It reduces overfitting and can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

**Random Forest Classifier**

```
from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
pred_rfc=rfc.predict(x_test)
print('Accuracy score''\n',accuracy_score(y_test,pred_rfc))
print('Confusion Matrix''\n',confusion_matrix(y_test,pred_rfc))
print('Classification Report''\n',classification_report(y_test,pred_rfc))
print('Log loss : ', log_loss(y_test,pred_rfc))
```

```
Accuracy score
 0.9839233903572051
Confusion Matrix
 [[30842   781]
 [  233 31217]]
Classification Report
              precision    recall  f1-score   support

           0       0.99      0.98      0.98     31623
           1       0.98      0.99      0.98     31450

    accuracy                           0.98     63073
   macro avg       0.98      0.98      0.98     63073
weighted avg       0.98      0.98      0.98     63073

Log loss :  0.5552763266379033
```

- **Complement Naïve Bayes Classifier**: Complement Naive Bayes is somewhat an adaptation of the standard Multinomial Naive Bayes algorithm.

**Complement Naive Bayes**

```
from sklearn.naive_bayes import ComplementNB

cnb=ComplementNB()
cnb.fit(x_train,y_train)
pred_cnb=cnb.predict(x_test)
print('Accuracy score''\n',accuracy_score(y_test,pred_cnb))
print('Confusion Matrix''\n',confusion_matrix(y_test,pred_cnb))
print('Classification Report''\n',classification_report(y_test,pred_cnb))
print('Log loss : ', log_loss(y_test,pred_cnb))
```

```
Accuracy score
 0.9108176240229575
Confusion Matrix
 [[28823  2800]
 [ 2825 28625]]
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     31623
           1       0.91      0.91      0.91     31450

    accuracy                           0.91     63073
   macro avg       0.91      0.91      0.91     63073
weighted avg       0.91      0.91      0.91     63073

Log loss :  3.080285638770586
```

Complement Naive Bayes is particularly suited to work with imbalanced datasets. In complement Naive Bayes, instead of calculating the probability of an item belonging to a certain class, we calculate the probability of the item belonging to all the classes.

- **Passive Aggressive Classifier:** Passive-Aggressive algorithms do not require a learning rate and are called so because if the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model. If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

**Passive Aggressive Classifier** ¶

```
from sklearn.linear_model import PassiveAggressiveClassifier
pac=PassiveAggressiveClassifier()

pac.fit(x_train,y_train)
pred_pac=pac.predict(x_test)
print('Accuracy score''\n',accuracy_score(y_test,pred_pac))
print('Confusion Matrix''\n',confusion_matrix(y_test,pred_pac))
print('Classification Report''\n',classification_report(y_test,pred_pac))
print('Log loss : ', log_loss(y_test,pred_pac))
```

```
Accuracy score
 0.9552581928876064
Confusion Matrix
 [[29109  2514]
 [  308 31142]]
Classification Report
              precision    recall  f1-score   support

           0       0.99      0.92      0.95     31623
           1       0.93      0.99      0.96     31450

    accuracy                           0.96     63073
   macro avg       0.96      0.96      0.96     63073
weighted avg       0.96      0.96      0.96     63073

Log loss :  1.5453591421745834
```

Out of all Models RandomForest Classifier gave the best result 98.34%.

Lets check the Cross Validation Score.

## ☑ Analyzing Accuracy of The Models

Classification Report consisting of Precision,Recall, Support and F1- score were the metrics used to evaluate the Model Performance.

Precision is defined as the ratio of true positives to the sum of true and false positives. Recall is defined as the ratio of true positives to the sum of true positives and false negatives.The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is. Support is the number of actual occurrences of the class in the dataset. It doesn't vary between models; it just diagnoses the performance evaluation process.

Log Loss quantifies the accuracy of a classifier by penalizing false classifications.

## Model Cross Validation

Cross validation is a technique for assessing how the statistical analysis generalises to an independent data set.It is a technique for evaluating machine learning models by training several models on subsets of the available input data and evaluating them on the complementary subset of the data. Using cross-validation, there are high chances that we can detect over-fitting with ease. Model Cross Validation scores were then obtained for assessing how the statistical analysis generalises to an independent data set. The models were evaluated by training several models on subsets of the available input data and evaluating them on the complementary subset of the data.

## Cross Validation

```
from sklearn.model_selection import cross_val_score as cvs
```

```
print(cvs(lo,smt_x,smt_y,cv=5).mean())
```

```
0.9347383708875979
```

```
print(cvs(rfc,smt_x,smt_y,cv=5).mean())
```

```
0.9837212216779463
```

```
print(cvs(mnb,smt_x,smt_y,cv=5).mean())
```

```
0.9089545598822175
```

```
print(cvs(cnb,smt_x,smt_y,cv=5).mean())
```

```
0.9089615359369099
```

```
print(cvs(pac,smt_x,smt_y,cv=5).mean())
```

```
0.9545226407214468
```

Random forest is the best model

**Random Forest gave the best result and also best fits the model.**

# ☑Interpretation of the Results

Based on comparing the results of, Precision, Recall, Accuracy Scores with Cross validation scores and log loss scores, it is determined that Random Forest Classifier, is the best model for the dataset.

## Hyper Parameter Tuning

GridSearchCV was used for Hyper Parameter Tuning of the Random Forest Classifier model.

## Hyper Parameter Tuning

```
# RandomForestClassifier
from sklearn.model_selection import GridSearchCV

#creating the parameter list to pass in GrideSearchcv
parameters={'max_features':['sqrt','log2'],
            'max_depth':[4,5,6,7],
            'criterion':['gini','entropy']}

gcv=GridSearchCV(RandomForestClassifier(),parameters,cv=5,scoring="accuracy")
gcv.fit(x_train,y_train)
gcv.best_params_
```

```
{'criterion': 'entropy', 'max_depth': 7, 'max_features': 'sqrt'}
```

```
gcv_pred=gcv.best_estimator_.predict(x_test)#predicting with best parameters
accuracy_score(y_test,gcv_pred)#checking the final score

print('Accuracy score''\n',accuracy_score(y_test,gcv_pred))
print('Confusion Matrix''\n',confusion_matrix(y_test,gcv_pred))
print('Classification Report''\n',classification_report(y_test,gcv_pred))
```

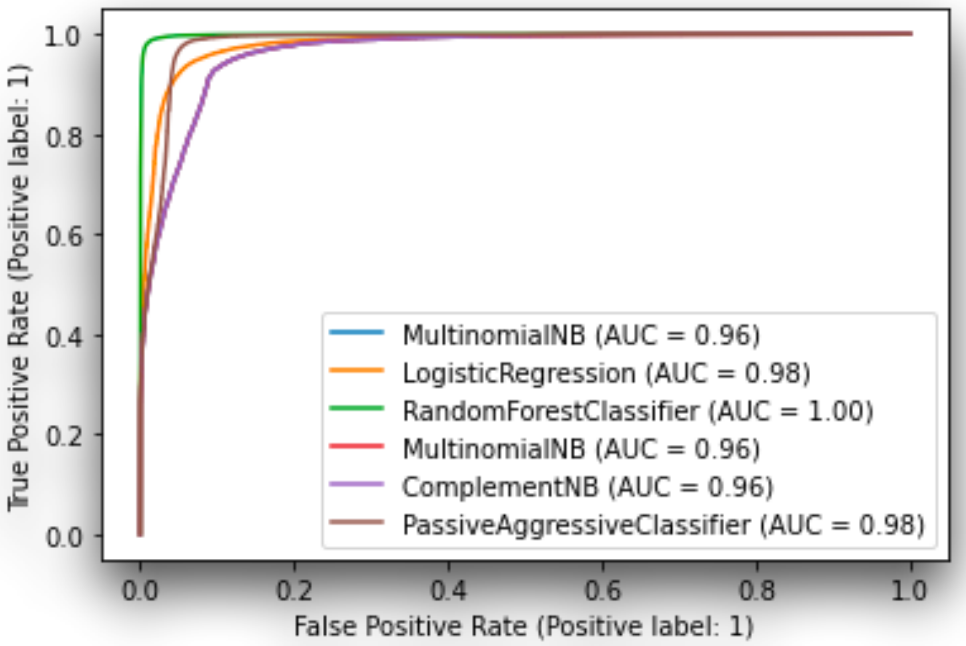
```
Accuracy score
 0.8092686252437652
Confusion Matrix
 [[22830  8793]
 [ 3237 28213]]
Classification Report
               precision    recall  f1-score   support

           0       0.88      0.72      0.79     31623
           1       0.76      0.90      0.82     31450

    accuracy                           0.81     63073
   macro avg       0.82      0.81      0.81     63073
weighted avg       0.82      0.81      0.81     63073
```

Thus after hyperparameter tuning its score is reduced may be due to low parameters, so moving forward with Random forest classifier before hyper tuning.

## ROC AUC Scores

The AUC value lies between 0.5 to 1 where 0.5 denotes a bad classifier and 1 denotes an excellent classifier. The score is used to summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

The AUC-ROC curve helps us visualize how well our machine learning classifier is performing. ROC curves are appropriate when the observations are balanced between each class.




**Thus Random Forest is giving very good AUC ROC score.**

The model was saved and the Test Dataset was then prepared for final classification work by the model. This model was then tested using the Test Dataset. The model performed with good amount of accuracy.

# ☑CONCLUSION

## • Key Findings and Conclusions of the Study

The Model has 98.34% accuracy. But since the dataset was highly imbalanced that is not the best metric for measuring its efficiency. Recall score of 0.99 for Benign (0) and 0.99 for Malignant(1), on the other hand, means that the model is optimized better to detect actual malignant comments. However, there is a need to strike a balance between precision and recall and have low false positives, which unnecessarily consume time and low false negatives which means only very few toxic comments deceive the model. F1 score

of 0.98 provides a nuanced way to catch positive results without harming the usefulness of the model.

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

The various data pre-processing and feature engineering steps in the project lent cognizance to various efficient methods for processing textual data. The NLTK suite is very useful in pre-processing text-based data and building classification models.

## ☑ Limitations of this work and Scope for Future Work

The models were trained on a highly imbalanced dataset where the total malignant comments formed only 10% of the entire available data, which seriously affected the training and accuracy of the models. By training the models on more diverse data sets, longer comments, and a more balanced dataset, more accurate and efficient classification models can be built.

# THANK YOU