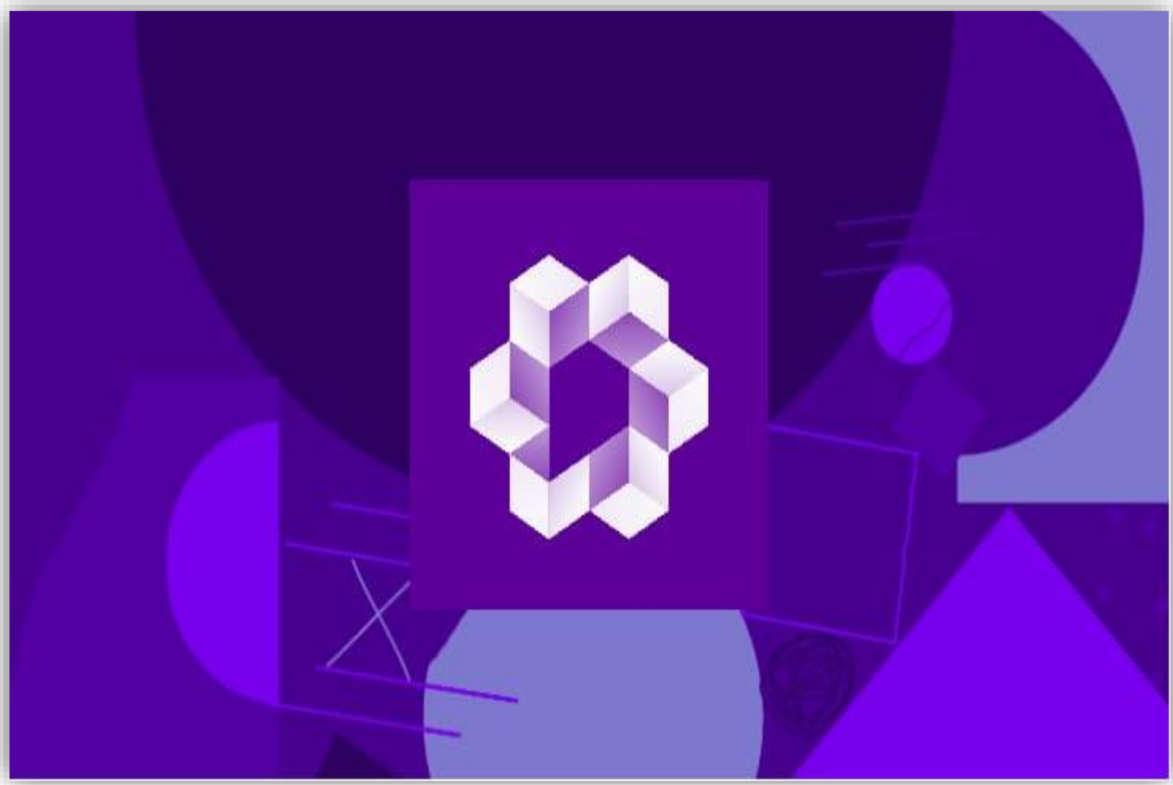
FLIP ROBO

Project Report On

“Malignant Comment Classification”



Submitted by:

Sapna Jolly

ACKNOWLEDGMENT

I would like to express my sincere thanks of gratitude to present this report on “Malignant Comments Classification” project. Working on this project was a good experience that has given me a basic knowledge about Machine Learning Model with NLP. This project also helped me in doing lots of research wherein I came to know about so many new things.

At the commencement of this project report, I would like to evince my deepest sense of gratitude to SME Mohd Kashif. Without his guidance, insightful decision, valuable comments and corrections it would not have possible to reach up to this mark.

I would like to draw my gratitude to Flip Robo Technologies and Data Trained for providing me a suitable environment and guidance to complete my work.

Finally, I would like to thank my family and friends who have helped me with their valuable suggestions and guidance and have been very helpful in various stages of project completion. Last but not the least thanks to the brilliant authors from where I have got the idea to carry out the project.

References:

I have also used few external resources that helped me to complete this project successfully. Below are the external resources that were used to create this project.

1. https://www.google.com/
2. https://scikit-learn.org/stable/index.html
3. https://www.researchgate.net/
4. https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a
5. https://towardsdatascience.com/
6. https://www.analyticsvidhya.com/

TABLE OF CONTENTS:

1. INTRODUCTION

Over the years, social media and social networking use have been increasing exponentially due to an upsurge in the use of the internet. Flood of information arises from online conversation in a daily basis as people are able discuss, express themselves and air their opinion via these platforms.

Every day, we get a tremendous amount of short content data from the blast of online correspondence, web-based business and the utilization of advanced gadgets. This volume of data requires text mining apparatuses to carry out the various report tasks in an opportune and suitable way. Detecting and controlling verbal abuse in an automated fashion is inherently an NLP task (Natural Language Processing). Text Classification is a great point for NLP.

Nowadays, every social media site and applications use machine learning approach. Machine Learning has simplified the task that may take long duration to complete without it. Most of the approaches require text analysis and classification techniques. Classification of the comments is necessary before posting on online platforms. This paper discusses different methodologies like logistic regression, support vector machine, multinomial naïve bayes etc. for comment classification into 6 different categories viz. malignant, highly malignant, rude, threat, abuse and loathe.

Business Problem Framing

Social media has given a lot of people which beyond imagination. In this era of technology, it has become the hub of information. The numbers of contents on social media are vast and rich and everything has found a place on social media that may be anything. It has given wings to its users to fly high and express their feelings. It has become a boon for the mankind but we all know that if there is good there must be bad. Likewise, social media has also got the dark side.

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.

Conceptual Background of the Domain Problem

In the past few years, it is seen that the cases related to social media hatred have increased exponentially. The social media is turning into a dark venomous pit for people now a days. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc. In social media the people spreading or involved in such kind of activities uses filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.

The result of such activities can be dangerous. It gives mental trauma to the victims making their lives miserable. People who are not well aware of mental health online hate or cyberbullying become life threatening for them. Such cases are also at rise. It is also taking its toll on religions. Each and every day we can see an incident of fighting between people of different communities or religions due to offensive social media posts.

Online hate, described as abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity has been identified as a major threat on online social media platforms. These kinds of activities must be checked for a better future.

Review of Literature

Aggression by text is a complex phenomenon, and different knowledge fields try to study and tackle this problem. In this study, several related literatures are used to express different types of aggression. Some of those are hate, cyberbullying, abusive language, malignant, flaming, threating, extremism, radicalization and hate speech. This research found a few dedicated works that addresses the effect of incorporating different text transformations on the model accuracy for sentiment classification. In this work, we performed a systematic review of the state-of-the-art in malignant comment classification using machine learning methods with NLP text processing. In our analysis of every primary study, we investigated data set used, evaluation metric, used machine learning methods, classes of malignant and non-malignant, and comment language.

Motivation for the Problem Undertaken

The main objective of this study is to investigate which method from a chosen set of machine learning techniques performs the best. So far, we have a range of publicly available models served through the Perspective API, including toxicity/malignant comments. But the current models still make errors, and they don't allow users to select which type of toxicity they are interested in finding.

The project which is given by Flip ROBO as a part of the internship programme which gives an insight to identify major factors that lead to cyberbullying and online abusive comments. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation was to classify the news in order to bring awareness and reduce unwanted chaos and make a good model which will help us to know such kind of miscreants. 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.

2. ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem:

We are provided with two different datasets. One for training and another one to test the efficiency of the model created using the training dataset. The training data provided here has both dependent and independent variables. As it is a multiclass problem it has 6 independent/target variables. Here the target variables named "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.

Clearly it is a binary classification problem as the target columns giving binary outputs and all independent variables has text so it is clear that it is a supervised machine learning problem where we can use the techniques of NLP and classification-based algorithms of Machine Learning. Here we will use NLP techniques like word tokenization, lemmatization, stemming and tfidf vectorizer then those processed data will be used to create best model using various classification based supervised ML algorithms like Logistic Regression, Multinomial NB, LGBM Classifier, XGB Classifier, Gradient Boosting Classifier, LinearSVC, Decision Tree Classifier and Adaboost Classifier.

Data Sources and their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The data set contains the training set, which has approximately 159571 samples and the test set which contains nearly 153164 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'. In the particular dataset all the columns are of object data type. The attribution information is as follows:

Variables	Definition
id	It includes unique Ids associated with each comment text given
comment_text	The comments extracted from various social media platforms
malignant	It denotes the comments are 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

Data Pre-processing Done

Data pre-processing is the process of converting raw data into a well-readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. I have used following pre-processing steps:

- Importing necessary libraries and loading dataset as a data frame.
- Checked some statistical information like shape, number of unique values present, info, null values, value counts, duplicated values etc.
- Checked for null values and did not find any null values. And removed Id.
- Done feature engineering and created new columns viz label: which contain both good and bad comments which is the sum of all the labels, comment_length: which contains the length of comment text.
- Visualized each feature using seaborn and matplotlib libraries by plotting categorical plots like pie plot, count plot, distribution plot and wordcloud for each label.
- Done text pre-processing techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization. Then created new column as clean_length after cleaning the data. All these steps were done on both train and test datasets. Checked correlation using heatmap.
- After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our

modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in.

Mathematically,

TF-IDF = TF(t*d)*IDF(t,d)

- Balanced the data using Randomoversampler method.

Data Inputs- Logic- Output Relationships

The train dataset consists of multilabel and features. The features are independent and label is dependent as the values of our independent variables changes as our label varies.

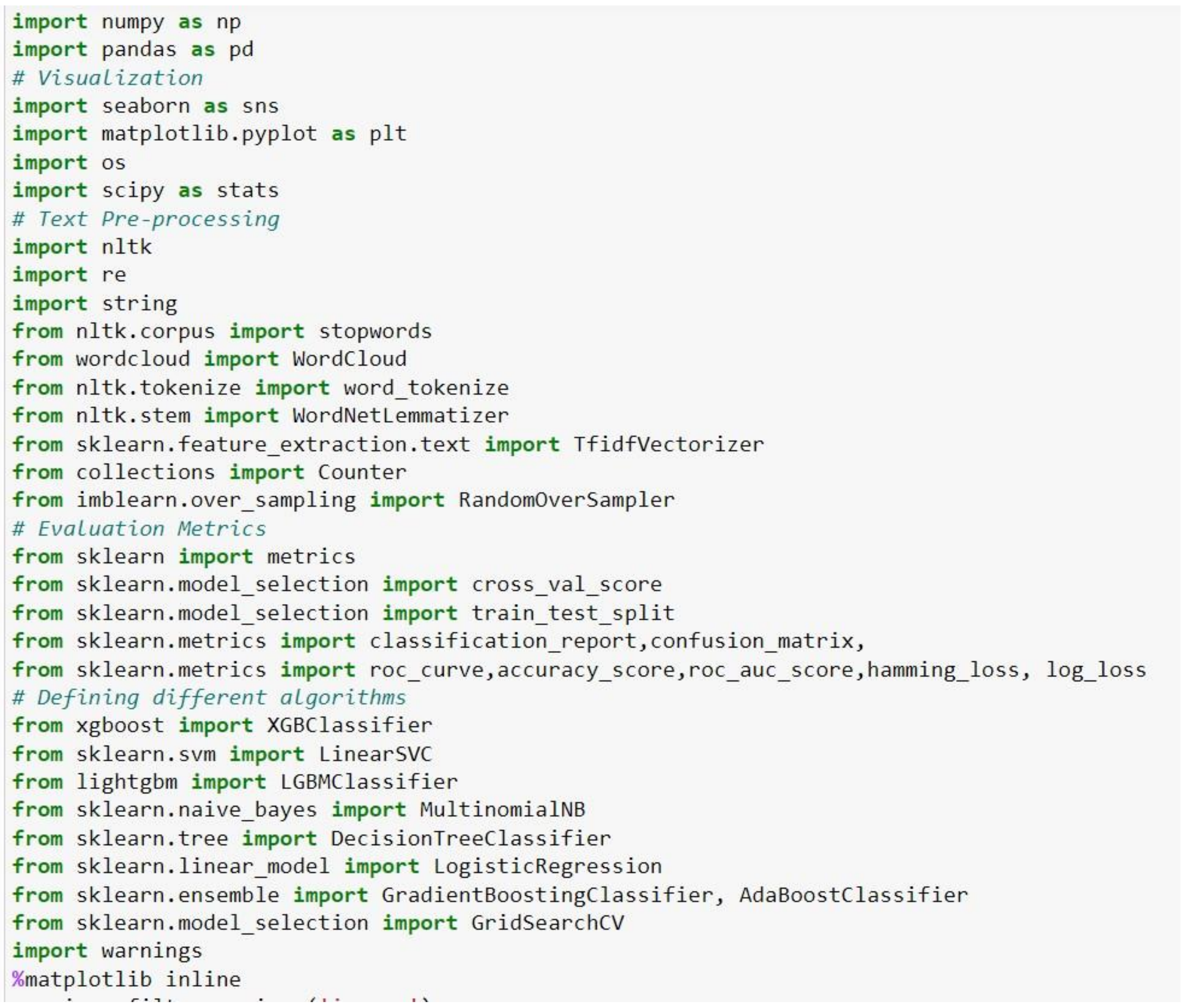
- I checked the distribution of skewness using dist plots and used count plots to check the counts available in each column as a part of univariate analysis.
- Got to know sense of loud words in every label using wordcloud which gives the words frequented in the labels.
- I have checked the correlation between the label and features using heat map.

Hardware & Software Requirements & Tools Used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware	Processor: core i5 RAM: 12 GB ROM/SSD: 512 GB
Software	Distribution: Anaconda Navigator Programming language: Python Browser based language shell: Jupyter Notebook

Libraries required:



```
import numpy as np
import pandas as pd
# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import os
import scipy as stats
# Text Pre-processing
import nltk
import re
import string
from nltk.corpus import stopwords
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from imblearn.over_sampling import RandomOverSampler
# Evaluation Metrics
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix,
from sklearn.metrics import roc_curve,accuracy_score,roc_auc_score,hamming_loss, log_loss
# Defining different algorithms
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC
from lightgbm import LGBMClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
import warnings
%matplotlib inline
```

- **import numpy as np:** It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.
- **import pandas as pd:** Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. The data-frame term is coming from Pandas only.
- **import matplotlib.pyplot as plt:** Matplotlib and Seaborn acts as the backbone of data visualization through Python.

  Matplotlib: It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays.

- **import seaborn as sns: Seaborn** is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data.

With the above sufficient libraries, we can perform pre-processing and data cleaning and model building.

3. MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible Problem-solving approaches (Methods):

In this project there were 6 features which defines the type of comment like malignant, hate, abuse, threat, loathe but we created another feature named as "label" which is combined of all the above features and contains the labelled data into the format of 0 and 1 where 0 represents "NO" and 1 represents "Yes". In this NLP based project we need to predict the multiple labels which are binary. I have converted text into feature vectors using TF-IDF vectorizer and separated our feature and labels. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models.

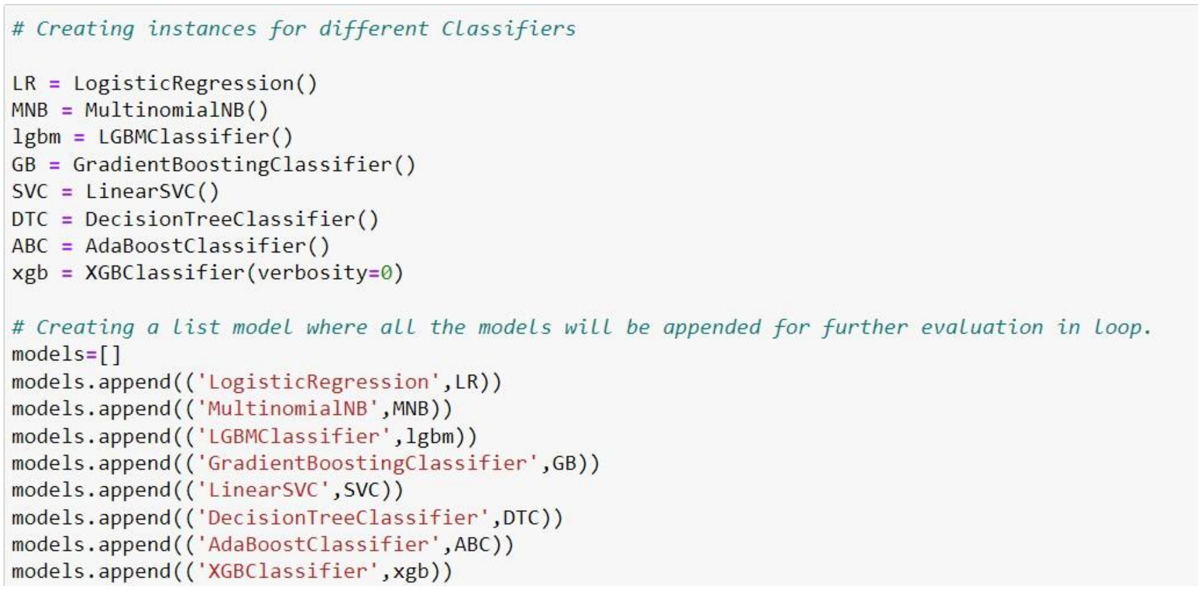
Testing of Identified Approaches (Algorithms)

Since the target variable is categorical in nature, from this I can conclude that it is a classification type problem hence I have used following classification algorithms. After the pre-processing and data cleaning I left with 10 columns including targets. The algorithms used on training the data are as follows:

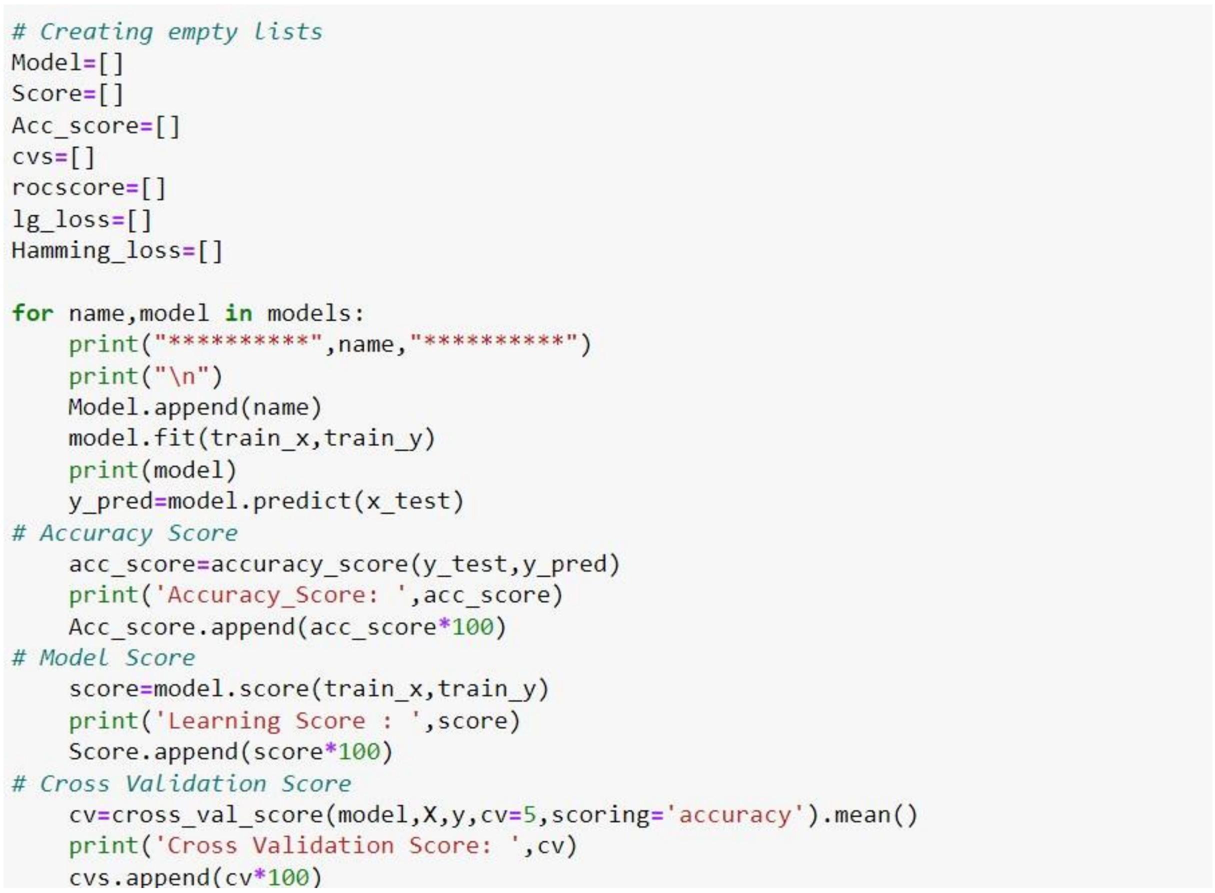
1. Logistic Regression
2. MultinomialNB
3. LightGBM Classifier
4. LinearSVC
5. Gradient Boosting Classifier
6. Decision Tree Classifier
7. Extreme Gradient Boosting Classifier (XGB)
8. AdaBoost Classifier

Run and evaluate selected models

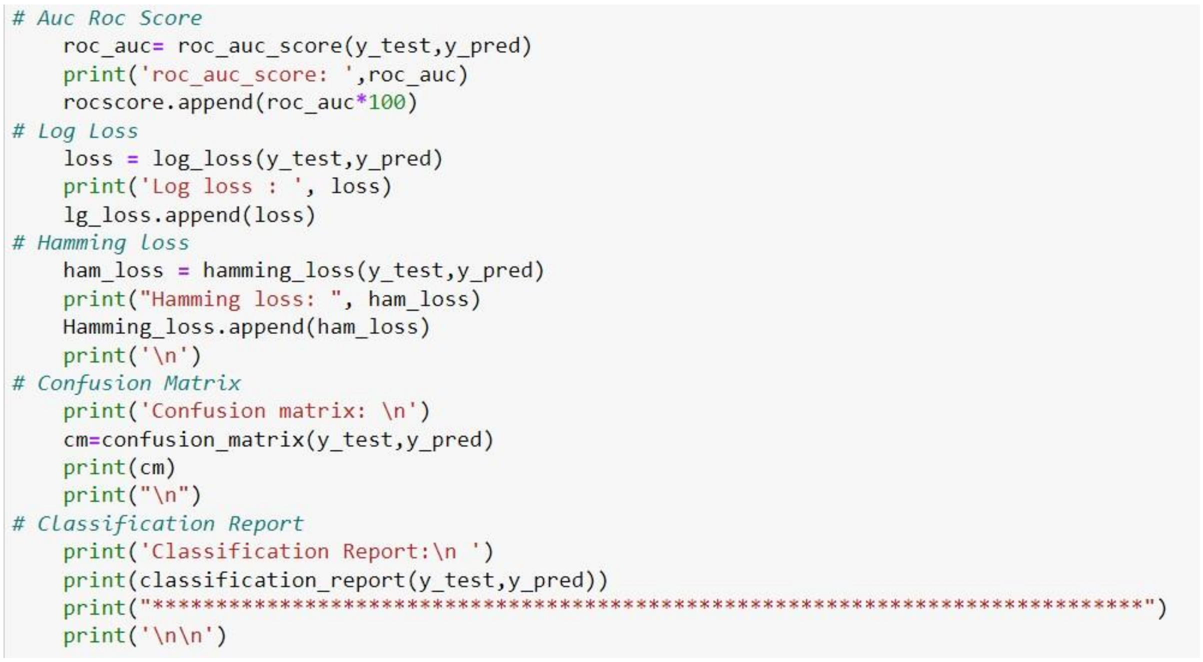
I have used 8 classification algorithms after choosing random state as 42. First, I have created 8 different classification algorithms and are appended in the variable models. Then, ran a for loop which contained the accuracy of the models along with different evaluation metrics.



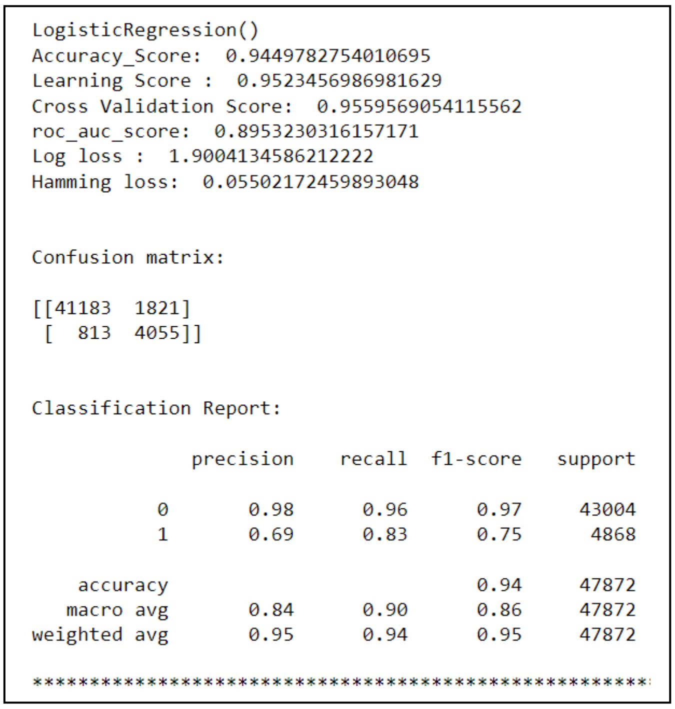
```
# Creating instances for different Classifiers

LR = LogisticRegression()
MNB = MultinomialNB()
lgbm = LGBMClassifier()
GB = GradientBoostingClassifier()
SVC = LinearSVC()
DTC = DecisionTreeClassifier()
ABC = AdaBoostClassifier()
xgb = XGBClassifier(verbosity=0)

# Creating a list model where all the models will be appended for further evaluation in loop.
models=[]
models.append(('LogisticRegression',LR))
models.append(('MultinomialNB',MNB))
models.append(('LGBMClassifier',lgbm))
models.append(('GradientBoostingClassifier',GB))
models.append(('LinearSVC',SVC))
models.append(('DecisionTreeClassifier',DTC))
models.append(('AdaBoostClassifier',ABC))
models.append(('XGBClassifier',xgb))
```



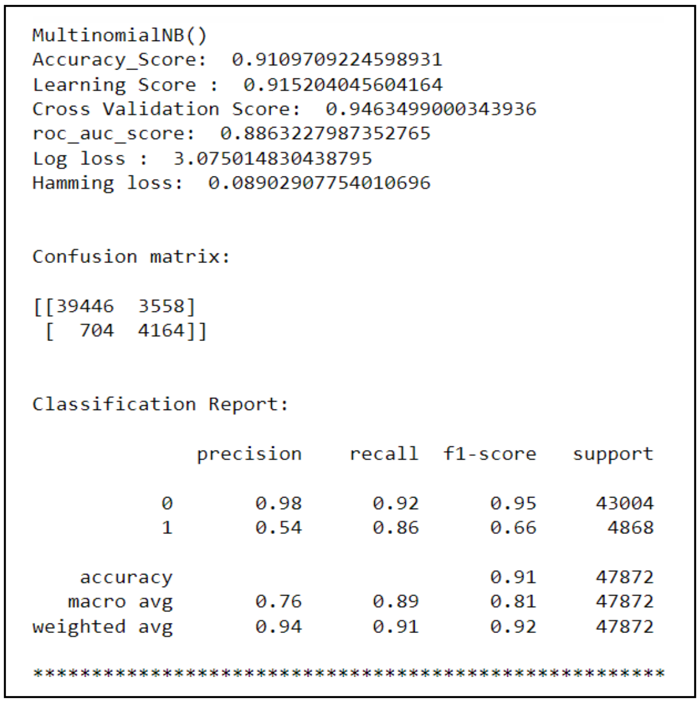
```
# Creating empty lists
Model=[]
Score=[]
Acc_score=[]
cvs=[]
rocscore=[]
lg_loss=[]
Hamming_loss=[]

for name,model in models:
    print("**********",name,"**********")
    print("\n")
    Model.append(name)
    model.fit(train_x,train_y)
    print(model)
    y_pred=model.predict(x_test)
# Accuracy Score
    acc_score=accuracy_score(y_test,y_pred)
    print('Accuracy_Score: ',acc_score)
    Acc_score.append(acc_score*100)
# Model Score
    score=model.score(train_x,train_y)
    print('Learning Score : ',score)
    Score.append(score*100)
# Cross Validation Score
    cv=cross_val_score(model,X,y,cv=5,scoring='accuracy').mean()
    print('Cross Validation Score: ',cv)
    cvs.append(cv*100)
```



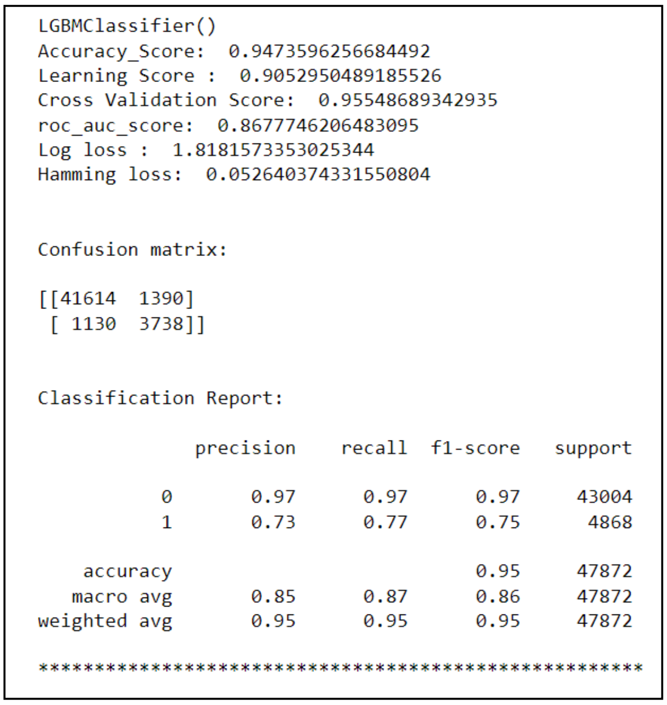
```
# Auc Roc Score
    roc_auc= roc_auc_score(y_test,y_pred)
    print('roc_auc_score: ',roc_auc)
    rocscore.append(roc_auc*100)
# Log Loss
    loss = log_loss(y_test,y_pred)
    print('Log loss : ', loss)
    lg_loss.append(loss)
# Hamming loss
    ham_loss = hamming_loss(y_test,y_pred)
    print("Hamming loss: ", ham_loss)
    Hamming_loss.append(ham_loss)
    print('\n')
# Confusion Matrix
    print('Confusion matrix: \n')
    cm=confusion_matrix(y_test,y_pred)
    print(cm)
    print("\n")
# Classification Report
    print('Classification Report:\n ')
    print(classification_report(y_test,y_pred))
    print("**************************************************************************")
    print('\n\n')
```



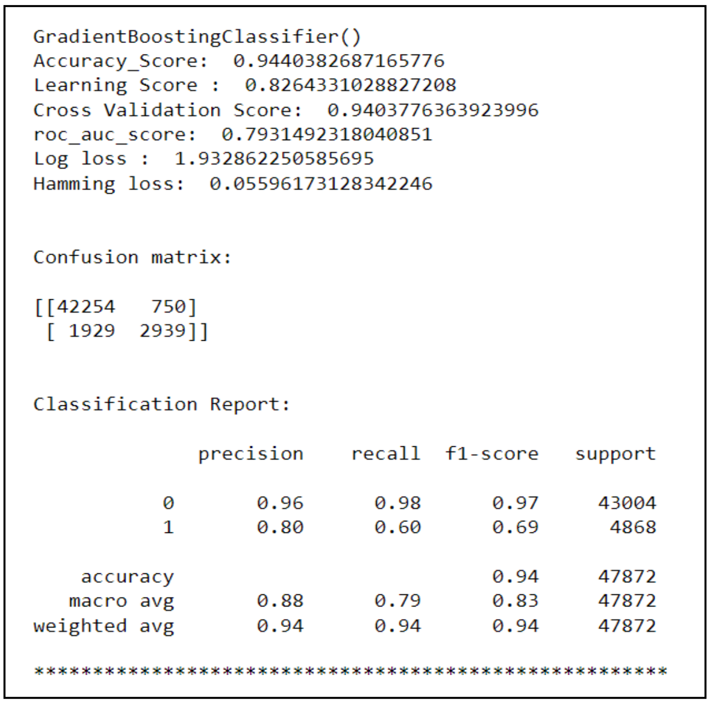
```
LogisticRegression()
Accuracy_Score:  0.9449782754010695
Learning Score :  0.9523456986981629
Cross Validation Score:  0.9559569054115562
roc_auc_score:  0.8953230316157171
Log loss :  1.9004134586212222
Hamming loss:  0.05502172459893048


Confusion matrix:

[[41183  1821]
 [  813  4055]]


Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.96      0.97     43004
           1       0.69      0.83      0.75      4868

    accuracy                           0.94     47872
   macro avg       0.84      0.90      0.86     47872
weighted avg       0.95      0.94      0.95     47872

*******************************************************
```



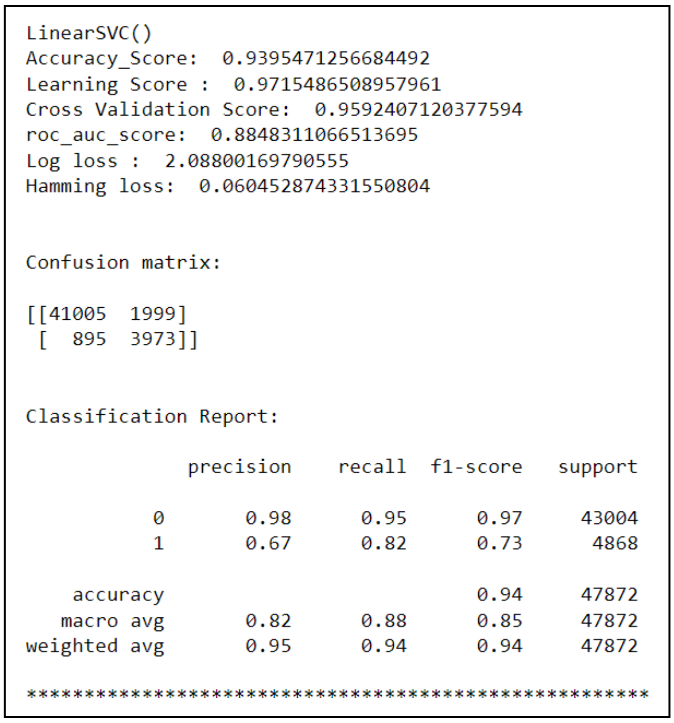
```
MultinomialNB()
Accuracy_Score:  0.9109709224598931
Learning Score :  0.915204045604164
Cross Validation Score:  0.9463499000343936
roc_auc_score:  0.8863227987352765
Log loss :  3.075014830438795
Hamming loss:  0.08902907754010696


Confusion matrix:

[[39446  3558]
 [  704  4164]]


Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.92      0.95     43004
           1       0.54      0.86      0.66      4868

    accuracy                           0.91     47872
   macro avg       0.76      0.89      0.81     47872
weighted avg       0.94      0.91      0.92     47872

*******************************************************
```



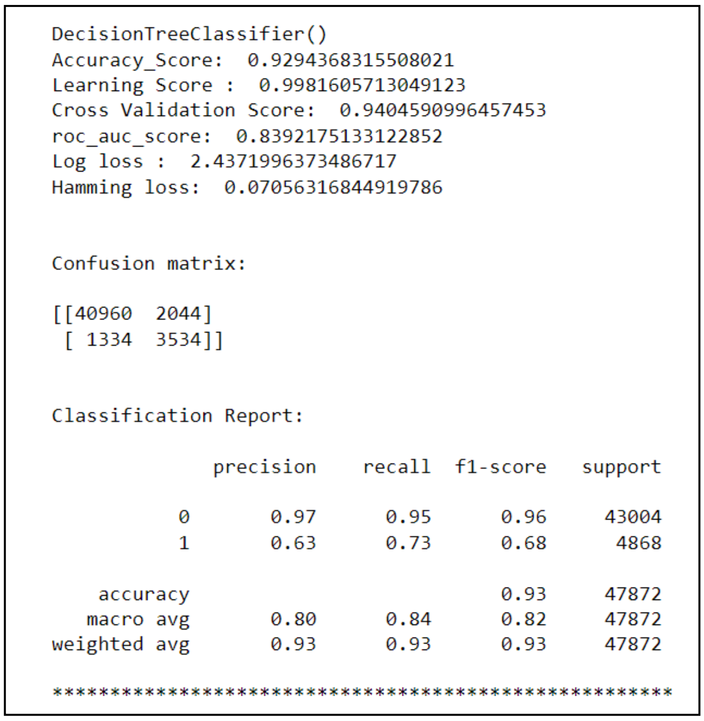
```
LGBMClassifier()
Accuracy_Score:  0.9473596256684492
Learning Score :  0.9052950489185526
Cross Validation Score:  0.95548689342935
roc_auc_score:  0.8677746206483095
Log loss :  1.8181573353025344
Hamming loss:  0.05264037433155080 4


Confusion matrix:

[[41614  1390]
 [ 1130  3738]]


Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.73      0.77      0.75      4868

    accuracy                           0.95     47872
   macro avg       0.85      0.87      0.86     47872
weighted avg       0.95      0.95      0.95     47872

*******************************************************
```



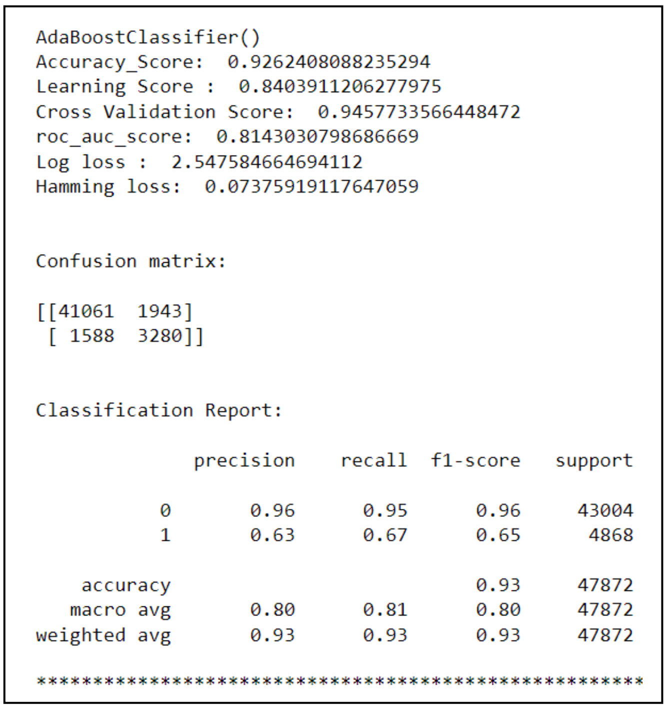
```
GradientBoostingClassifier()
Accuracy_Score:  0.9440382687165776
Learning Score :  0.8264331028827208
Cross Validation Score:  0.9403776363923996
roc_auc_score:  0.7931492318040851
Log loss :  1.932862250585695
Hamming loss:  0.05596173128342246


Confusion matrix:

[[42254   750]
 [ 1929  2939]]


Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.98      0.97     43004
           1       0.80      0.60      0.69      4868

    accuracy                           0.94     47872
   macro avg       0.88      0.79      0.83     47872
weighted avg       0.94      0.94      0.94     47872

*******************************************************
```



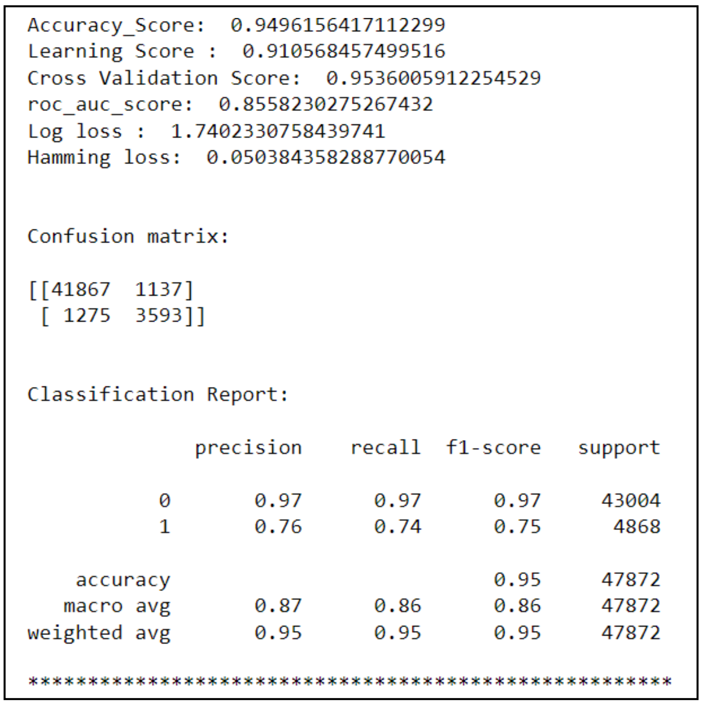
```
LinearSVC()
Accuracy_Score:  0.9395471256684492
Learning Score :  0.9715486508957961
Cross Validation Score:  0.9592407120377594
roc_auc_score:  0.8848311066513695
Log loss :  2.08800169790555
Hamming loss:  0.06045287433155080 4


Confusion matrix:

[[41005  1999]
 [  895  3973]]


Classification Report:

              precision    recall  f1-score   support

           0       0.98      0.95      0.97     43004
           1       0.67      0.82      0.73      4868

    accuracy                           0.94     47872
   macro avg       0.82      0.88      0.85     47872
weighted avg       0.95      0.94      0.94     47872

*******************************************************
```



```
DecisionTreeClassifier()
Accuracy_Score:  0.9294368315508021
Learning Score :  0.9981605713049123
Cross Validation Score:  0.9404590996457453
roc_auc_score:  0.8392175133122852
Log loss :  2.4371996373486717
Hamming loss:  0.07056316844919786


Confusion matrix:

[[40960  2044]
 [ 1334  3534]]


Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.95      0.96     43004
           1       0.63      0.73      0.68      4868

    accuracy                           0.93     47872
   macro avg       0.80      0.84      0.82     47872
weighted avg       0.93      0.93      0.93     47872

*******************************************************
```

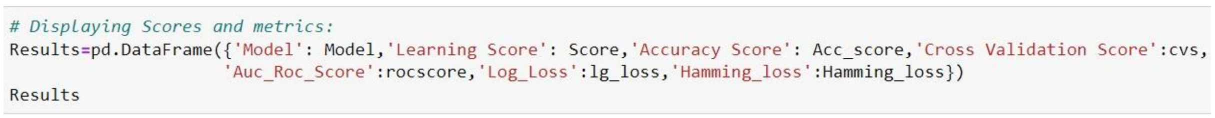


```
AdaBoostClassifier()
Accuracy_Score:  0.9262408088235294
Learning Score :  0.8403911206277975
Cross Validation Score:  0.9457733566448472
roc_auc_score:  0.8143030798686669
Log loss :  2.547584664694112
Hamming loss:  0.07375919117647059


Confusion matrix:

[[41061  1943]
 [ 1588  3280]]


Classification Report:

              precision    recall  f1-score   support

           0       0.96      0.95      0.96     43004
           1       0.63      0.67      0.65      4868

    accuracy                           0.93     47872
   macro avg       0.80      0.81      0.80     47872
weighted avg       0.93      0.93      0.93     47872

******************************************************
```

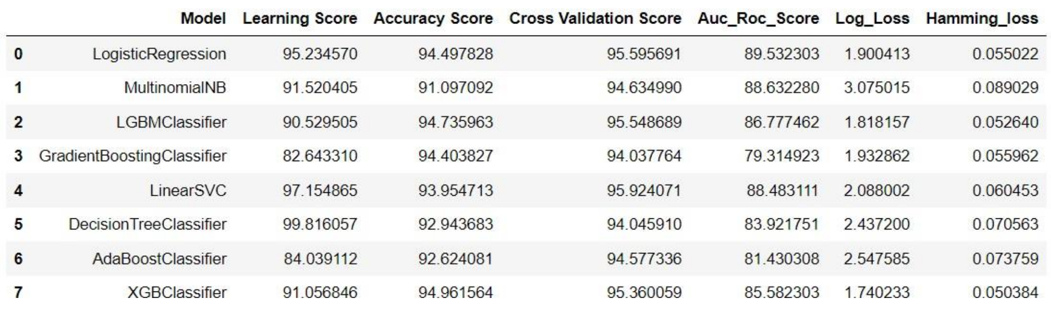


```
Accuracy_Score:  0.9496156417112299
Learning Score :  0.910568457499516
Cross Validation Score:  0.9536005912254529
roc_auc_score:  0.8558230275267432
Log loss :  1.7402330758439741
Hamming loss:  0.05038435828877005 4


Confusion matrix:

[[41867  1137]
 [ 1275  3593]]


Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.76      0.74      0.75      4868

    accuracy                           0.95     47872
   macro avg       0.87      0.86      0.86     47872
weighted avg       0.95      0.95      0.95     47872

*********************************************************
```

Model Selection:



```
# Displaying Scores and metrics:
Results=pd.DataFrame({'Model': Model,'Learning Score': Score,'Accuracy Score': Acc_score,'Cross Validation Score':cvs,
                     'Auc_Roc_Score':rocscore,'Log_Loss':lg_loss,'Hamming_loss':Hamming_loss})
Results
```



	Model	Learning Score	Accuracy Score	Cross Validation Score	Auc_Roc_Score	Log_Loss	Hamming_loss
0	LogisticRegression	95.234570	94.497828	95.595691	89.532303	1.900413	0.055022
1	MultinomialNB	91.520405	91.097092	94.634990	88.632280	3.075015	0.089029
2	LGBMClassifier	90.529505	94.735963	95.548689	86.777462	1.818157	0.052640
3	GradientBoostingClassifier	82.643310	94.403827	94.037764	79.314923	1.932862	0.055962
4	LinearSVC	97.154865	93.954713	95.924071	88.483111	2.088002	0.060453
5	DecisionTreeClassifier	99.816057	92.943683	94.045910	83.921751	2.437200	0.070563
6	AdaBoostClassifier	84.039112	92.624081	94.577336	81.430308	2.547585	0.073759
7	XGBClassifier	91.056846	94.961564	95.360059	85.582303	1.740233	0.050384

After creating and training different classification algorithms, we can see that the difference between accuracy and cross validation score is less for "Extreme Gradient Boosting Classifier (XGBClassifier)" and "Gradient Boosting Classifier". But, "XGBClassifier" giving less loss values, high auc roc score and accuracy score compared to Gradient Boosting Classifier. On this basis I can conclude that "XGBClassifier" as the best fitting model. Now, we will try Hyperparameter Tuning to find out the best parameters and using them to improve the scores and metrics values.

Hyper Parameter Tuning:

```
# Let's Use the GridSearchCV to find the best paarameters in XGBClassifier

# Extreme XGBClassifier
parameters = {'n_estimators':[100,1000],
              'booster':['gbtree'],
              'max_depth':[2,6],
              'eta':[0,0.2,0.3],
              'colsample_bytree':[1,0.8]}

# Running GridSearchCV for the model Bagging Regressor.
GCV=GridSearchCV(XGBClassifier(),parameters,cv=5,scoring='accuracy')
```

```
# Training the best model
GCV.fit(train_x,train_y)
```

```
GridSearchCV(cv=5,
             estimator=XGBClassifier(base_score=None, booster=None,
                                     colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None, gamma=None,
                                     gpu_id=None, importance_type='gain',
                                     interaction_constraints=None,
                                     learning_rate=None, max_delta_step=None,
                                     max_depth=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     n_estimators=100, n_jobs=None,
                                     num_parallel_tree=None, random_state=None,
                                     reg_alpha=None, reg_lambda=None,
                                     scale_pos_weight=None, subsample=None,
                                     tree_method=None, validate_parameters=None,
                                     verbosity=None),
             param_grid={'booster': ['gbtree'], 'colsample_bytree': [1, 0.8],
                         'eta': [0, 0.2, 0.3], 'max_depth': [2, 6],
                         'n_estimators': [100, 1000]},
             scoring='accuracy')
```

```
#Getting best parameters
GCV.best_params_
```

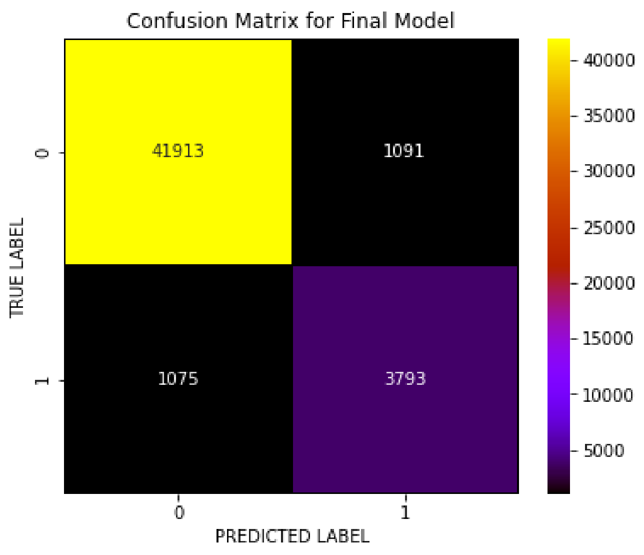
```
{'booster': 'gbtree',
 'colsample_bytree': 1,
 'eta': 0.3,
 'max_depth': 6,
 'n_estimators': 1000}
```

I Have used 5 XGBClassifier parameters to be saved under the variable "parameters" that will be used in GridSearchCV for finding the best output. Assigned a variable to the GridSearchCV function after entering all the necessary inputs. And we used our training data set to make the GridSearchCV aware of all the hyper parameters that needs to be applied on our best model.

Creating Final Model:

```
# Creating final model
comment_model = XGBClassifier(n_estimators=1000, max_depth=6, eta=0.3, colsample_bytree=1, booster='gbtree')
comment_model.fit(train_x, train_y)
pred = comment_model.predict(x_test)
acc_score = accuracy_score(y_test,pred)
print("Accuracy score:", acc_score*100)
roc_auc = roc_auc_score(y_test,y_pred)
print('roc_auc_score: ',roc_auc*100)
print('Log loss : ', log_loss(y_test,pred))
print("Hamming loss: ", hamming_loss(y_test,pred))
print("\n")
print('Confusion Matrix: \n',confusion_matrix(y_test,pred))
print('\n')
print('Classification Report:','\n',classification_report(y_test,pred))
```

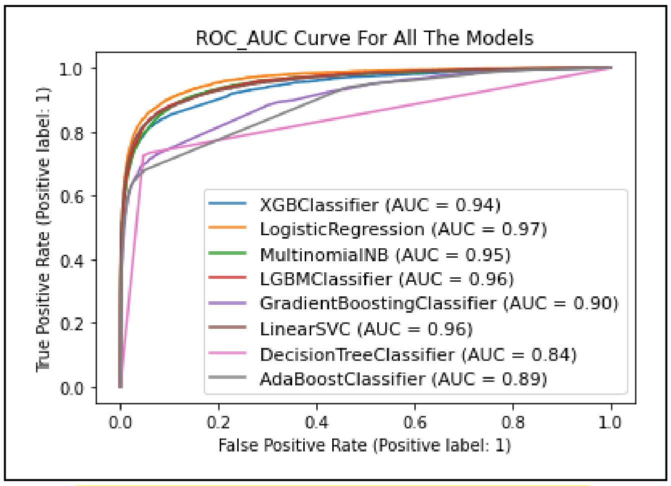
```
Accuracy score: 95.47543449197886
roc_auc_score:  85.58230275267432
Log loss :  1.5627477864340933
Hamming loss:  0.0452456550802139


Confusion Matrix:
 [[41913  1091]
 [ 1075  3793]]


Classification Report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97     43004
           1       0.78      0.78      0.78      4868

    accuracy                           0.95     47872
   macro avg       0.88      0.88      0.88     47872
weighted avg       0.95      0.95      0.95     47872
```



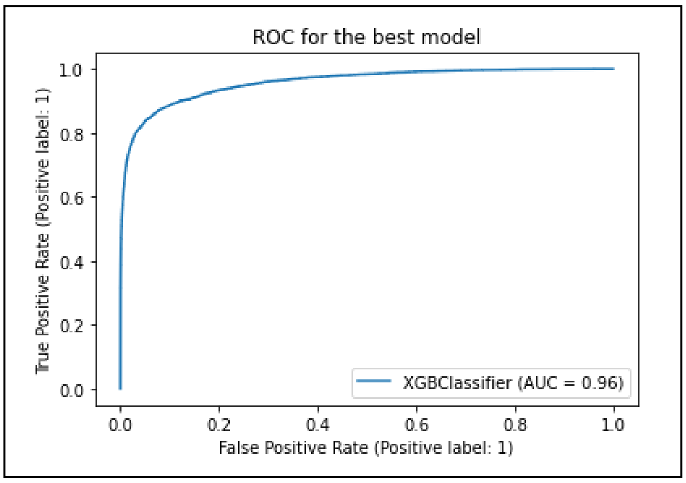
I have successfully incorporated the hyper parameter tuning using best parameters of XGB Classifier and the accuracy of the model has been increased after hyperparameter tuning and received the accuracy score as 95.47% which is very good. With the help of confusion matrix, we can able to see actual and predicted values for the final model.

And also, we can understand the number of times we got the correct outputs and the number of times my model missed to provide the correct prediction

ROC-AUC Curve for all the models used and for best model:



ROC-AUC Curve for all the models



ROC-AUC Curve for final model

I have generated the ROC Curve for all the models and for the best model and it shows the AUC score for the models. The AUC score for my final model is 96% which is increased after tuning the model.

Saving the final model and predicting the results

```
# Saving the model using .pkl
import joblib
joblib.dump(comment_model,"Malignant_Comments_Classification(IP6).pkl")
```

```
['Malignant_Comments_Classification(IP6).pkl']
```

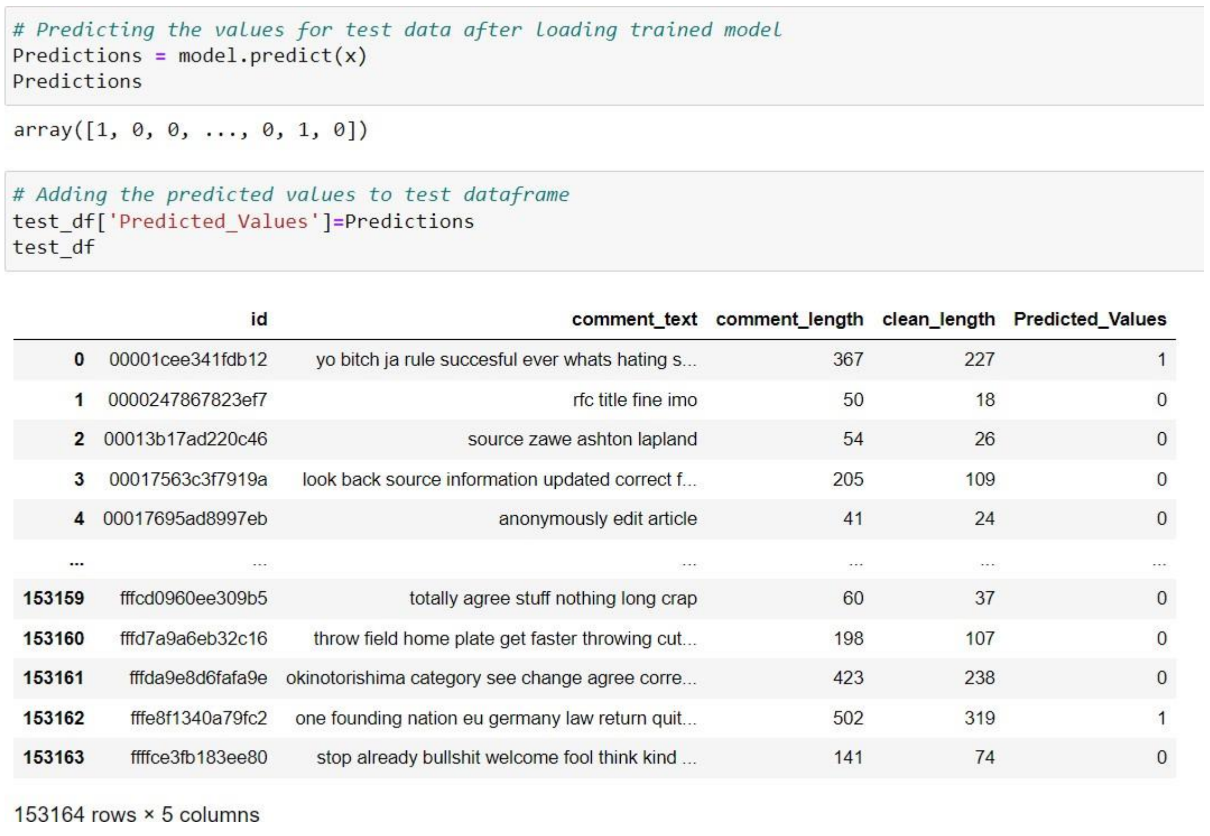
I am using the joblib option to save the final classification model in the form of .pkl.

```
# Predicting the trained final model
comment_model.predict(X)
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
# Loading the final model
model = joblib.load('Malignant_Comments_Classification(IP6).pkl')
```

I have loaded my saved model to use further and to get the predictions for test data.



```python
# Predicting the values for test data after loading trained model
Predictions = model.predict(x)
Predictions
```

```
array([1, 0, 0, ..., 0, 1, 0])
```

```python
# Adding the predicted values to test dataframe
test_df['Predicted_Values']=Predictions
test_df
```

	id	comment_text	comment_length	clean_length	Predicted_Values
0	00001cee341fdb12	yo bitch ja rule succesful ever whats hating s...	367	227	1
1	0000247867823ef7	rfc title fine imo	50	18	0
2	00013b17ad220c46	source zawe ashton lapland	54	26	0
3	00017563c3f7919a	look back source information updated correct f...	205	109	0
4	00017695ad8997eb	anonymously edit article	41	24	0
...	...	...	...	...	...
153159	fffcd0960ee309b5	totally agree stuff nothing long crap	60	37	0
153160	fffd7a9a6eb32c16	throw field home plate get faster throwing cut...	198	107	0
153161	fffda9e8d6fafa9e	okinotorishima category see change agree corre...	423	238	0
153162	fffe8f1340a79fc2	one founding nation eu germany law return quit...	502	319	1
153163	ffffce3fb183ee80	stop already bullshit welcome fool think kind ...	141	74	0

153164 rows × 5 columns

Key Metrics for success in solving problem under consideration

In order to evaluate the performance of each algorithm, several metrics are defined accordingly, and are discussed briefly below.

- **Accuracy score**: This metric measures how many of the comments are labelled correctly. However, in our dataset, where most of comments are not toxic, regardless of performance of model, a high accuracy was achieved. Accuracy is the ratio of number of correct predictions into number of predictions. In binary classification problem, accuracy can be calculated as below,

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

  Where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.
- **Precision and Recall:** Precision and recall were designed to measure the model performance in its ability to correctly classify the malignant

comments. Precision explains what fraction of malignant classified comments are truly malignant, and Recall measures what fraction of

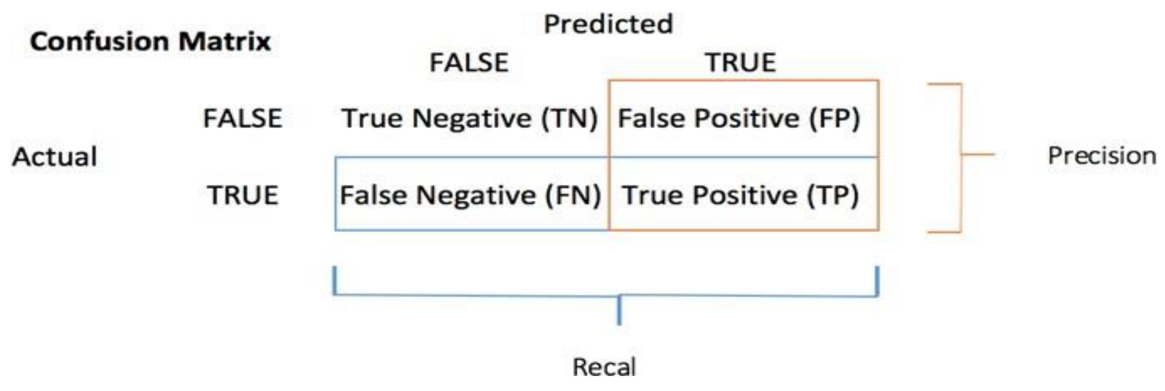
$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad \text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

malignant comments are labelled correctly.

- **F1 Score** is used to express the performance of the machine learning model (or classifier). It gives the combined information about the precision and recall of a model. This means a high F1-score indicates a high value for both recall and precision.

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

- **Confusion Matrix** is one of the evaluation metrics for machine learning classification problems, where a trained model is being evaluated for accuracy and other performance measures. And this matrix is called the confusion matrix since it results in an output that shows how the system is confused between the two classes.



- **Cross Validation Score** is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited. In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate. It is used to estimate the performance of ML models.

- **Roc Auc Score:** The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values.
  The **Area Under Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.
- **Log Loss** is the most important classification metric based on probabilities. Log Loss is the negative average of the log of corrected predicted probabilities for each instance. Log loss for binary classification is:

$$\text{Log loss} = \frac{1}{N} \sum_{i=1}^{N} -\left(y_i * \log(p_i) + (1-y_i) * \log(1-p_i)\right)$$

Where pi is the probability of class 1, and (1-pi) is the probability of class 0.

Log loss for multi-class classification is:

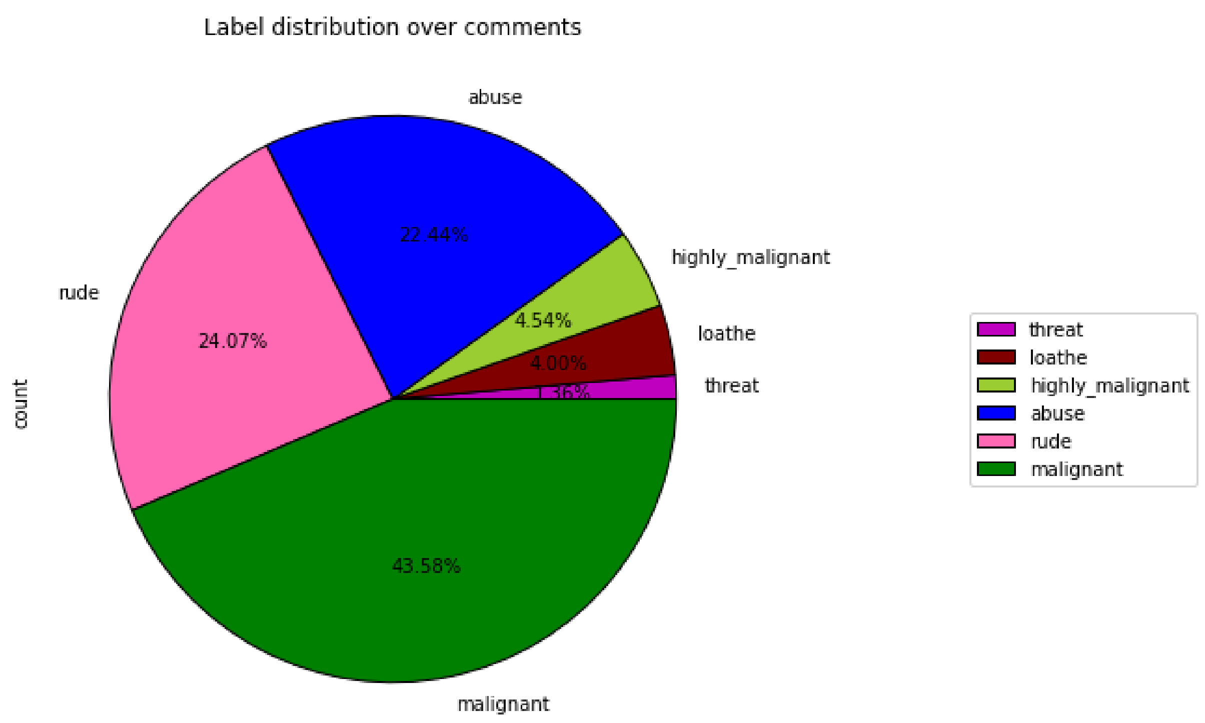
$$logloss = -\frac{1}{N} \sum_{i}^{N} \sum_{j}^{M} y_{ij} \log(p_{ij})$$

N is the number of rows and **M** is the number of classes.

- **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. In multi-label classification, hamming loss penalizes only the individual labels.

Visualizations

I used pandas profiling to get the over viewed visualization on the pre-processed data. Pandas is an open-source Python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable. I have used wordcloud to get the sense of loud words in the labels.



Observations:

- From the pie chart we can notice approximately 43% of the comments are malignant, 24% of the comments are rude and 22% are abuse. The count of malignant comment is high compared to other type of comments and the count of threat comments are very less.

Interpretation of the Results

Visualizations: I have used distribution plot to visualize how the data has been distributed. Used count plots and pie charts to check the count of particular category for each feature. The heat map helped me to understand the correlation between dependent and independent features. Also, heat map helped to detect the multicollinearity problem and feature importance. With the help of WordClouds I would able to sense the loud words in each label. AUC-ROC curve helped to select the best model.

Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed few NLP text processing steps which I have already mentioned in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model building: After cleaning and processing data, I performed train test split to build the model. I have built multiple classification models to get the accurate accuracy score, and evaluation metrics like precision, recall, confusion matrix, f1 score, log loss, hamming loss. I got Extreme Gradient Boosting Classifier (XGB Classifier) as the best model which gives 94.96% accuracy score. I checked the cross-validation score ensuring there will be no overfitting. After tuning the best model XGB Classifier, I got 95.47% accuracy score and also got increment in AUC-ROC curve. Finally, I saved my final model and got the good predictions results for test dataset.

CONCLUSION

Key Findings and Conclusions of the Study

From the above analysis the below mentioned results were achieved which depicts the chances and conditions of a comment being a hateful comment or a normal comment;

- ✓ With the increasing popularity of social media more and more people consume feeds from social media and due differences they spread hate comments to instead of love and harmony. It has strong negative impacts on individual users and broader society.

The conclusion for our study:

- In training dataset, we have only 10% of data which is spreading hate on social media.
- In this 10% data most of the comments are malignant, rude or abuse.
- After using the wordcloud we find that there are so many abusive words present in the negative comments. While in positive comments there is no use of such comments.
- Some of the comments are very long while some are very short

Learning Outcomes of the Study in respect of Data Science

While working on this project we learned many things and gains new techniques and ways to deal with uncleaned text data. Found how to deal with multiple target features. Tools used for visualizations gives a better understanding of dataset. We have used a lot of algorithms and find that in the classification problem where we have only two labels, XGB Classifier gives better results compared to others.

It is possible to classify the comments content into the required categories of authentic and however, using this kind of project an awareness can be created to know what is fake and authentic.

Limitations of this work and scope for future work

Limitations: This project was amazing to work on, it creates new ideas to think about but there were some limitations in this project like unbalanced dataset**.** Every effort has been put on it for perfection but nothing is perfect and this project is of no exception. There are certain areas which can be enhanced**.**

Future work: In future work, we can focus on performance and error analysis of the model as lots of comments are misclassified into the hate category. Previous work has achieved success using various algorithms on data in English language but in future, we can consider having data in regional languages. We can also work on after work of the detection of the malignant comments like automatic blocking of the user, auto-deletion of harmful comments on social media platforms. Comment detection is an emerging research area with few public datasets. So, a lot of works need to be done on this field