FLIP ROBO

Project Report On

" <u>Malignant Comment Classification</u> "

Submitted By :

Arun Joshva Stephenson

ACKNOWLEDGMENT

I would like to express my sincere thanks of gratitude to present this report on “Malignant Comments Classification” 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 where in 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 for 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.

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.analyticsvidhya.com/
4. https://towardsdatascience.com/

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 other has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behavior.

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 lines miserable. People who are not well aware of mental health online hate or cyberbullying became 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 test 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 intership 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.

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 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, 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 contains 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	***Defination***
Id	It includes unique ids associated with each comment text given
loathe	It describes the comments which are hateful and loathing in nature
Comment_text	The comments extracted from various social media platforms
malignant	It denotes the comments are malignant or not
Highly_malignant	It denotes the 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

➢ *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 dataframe.
- 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 Punctuation 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 dat. 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 modeling. 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 Random over sampler method.

➢ *Data Inputs – Logic – Output Relationships*

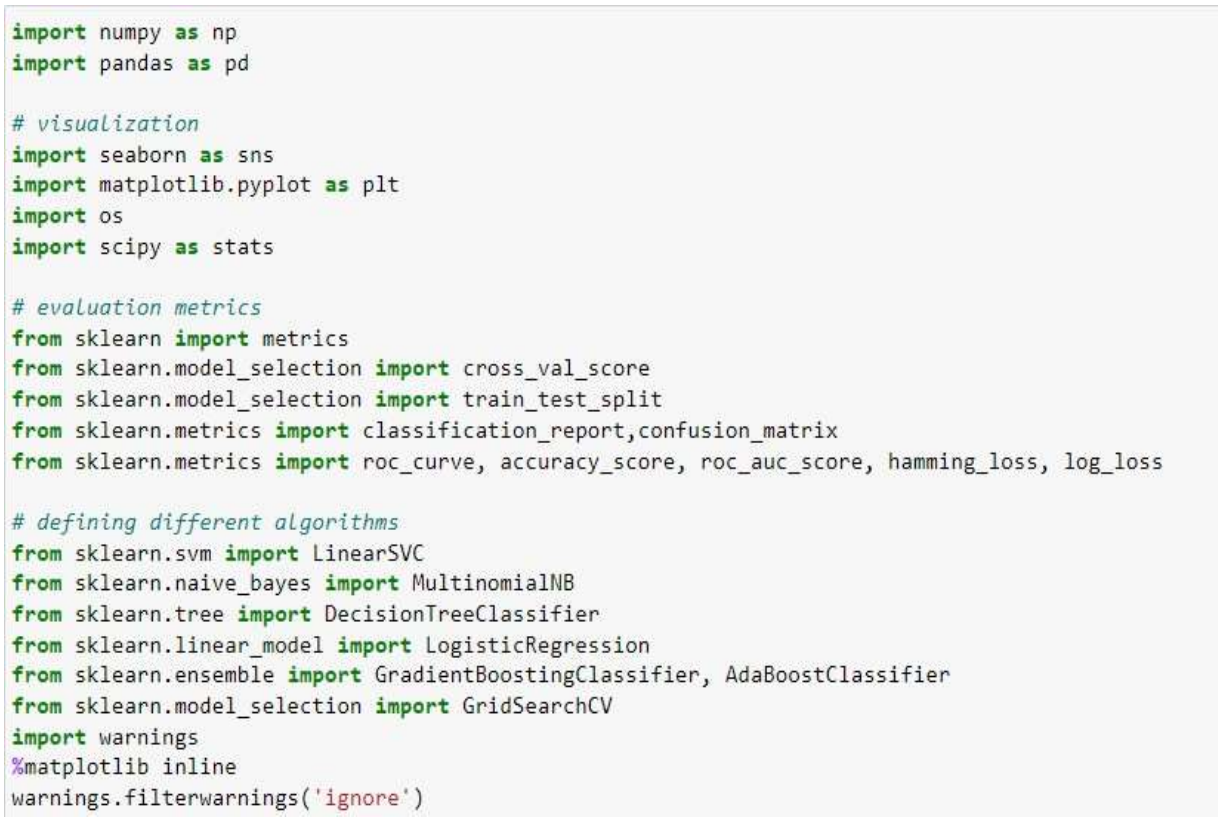
The dataset consists of multilabel and features. The features are independent and label is dependent as the values of our independent variables changes our label varies.

- I checked the distribution of skewness using dist plotsand 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.

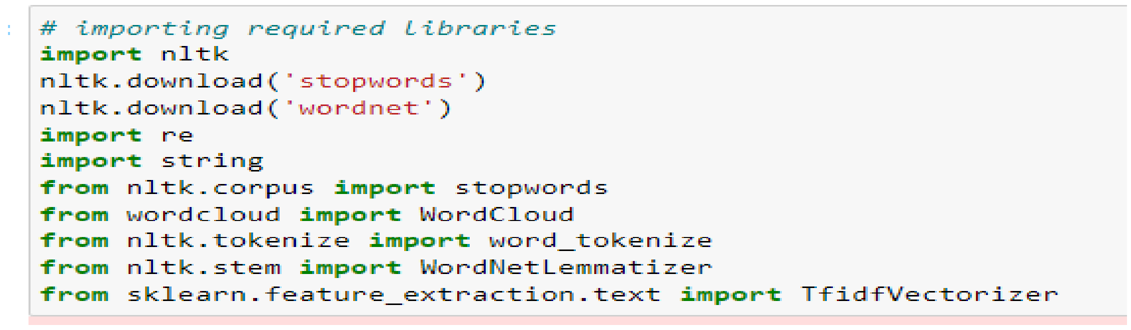
➢*Hardware & Software Requirements & Tools Used*

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

# 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 sklearn.svm import LinearSVC
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
warnings.filterwarnings('ignore')
```



```
# importing required libraries
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
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
```

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

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 labeled 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.

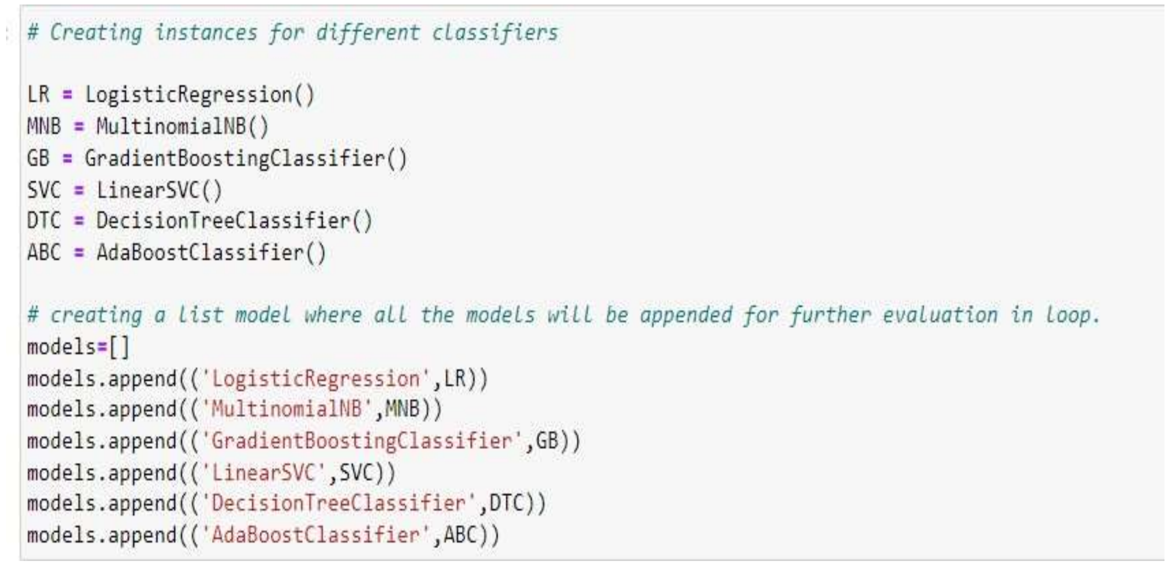
➢ *Testing of Identified Approaches (Algorithms)*

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:

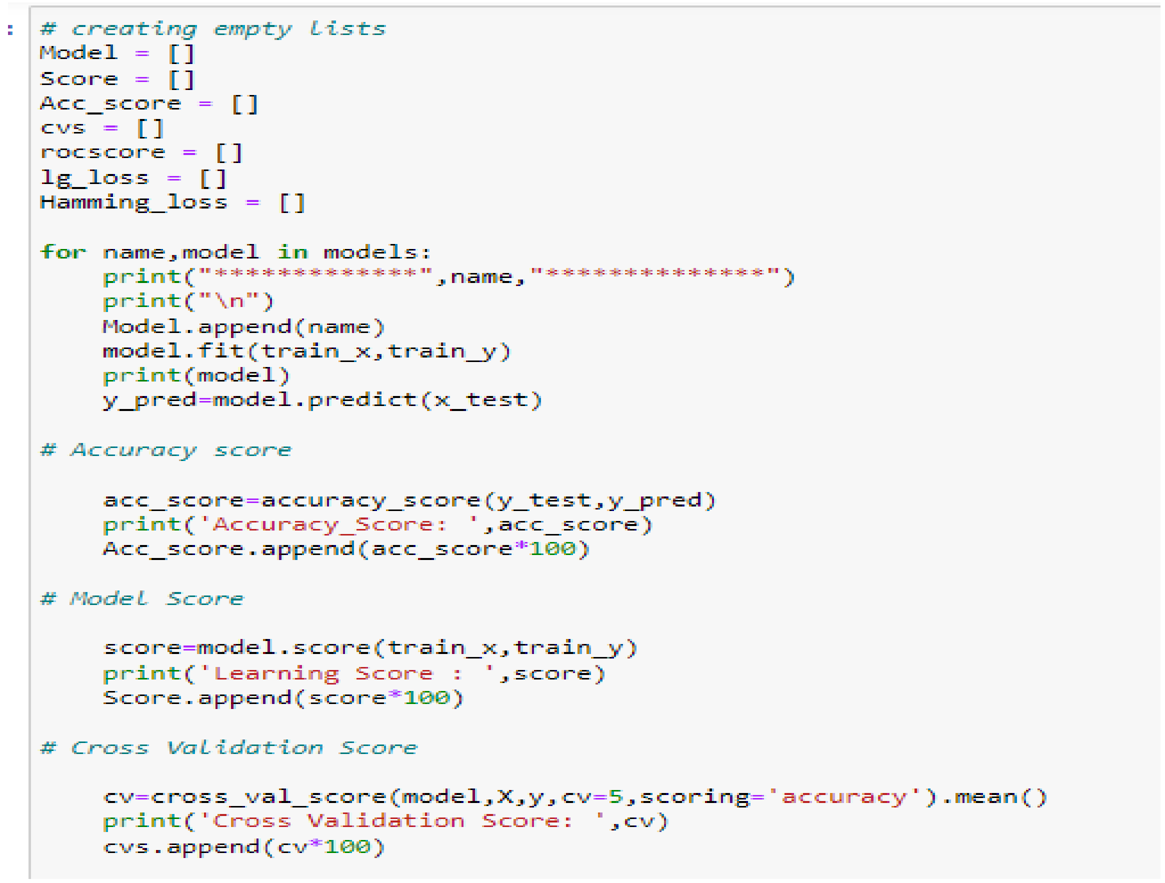
- ✓ Logistic Regression
- ✓ MultinomialNB
- ✓ LightGBM Classifier
- ✓ LinearSVC
- ✓ Gradient Boosting Classifier
- ✓ Decision Tree Classifier
- ✓ AdaBoost Classifier

➢ *Run and evaluate selected models*

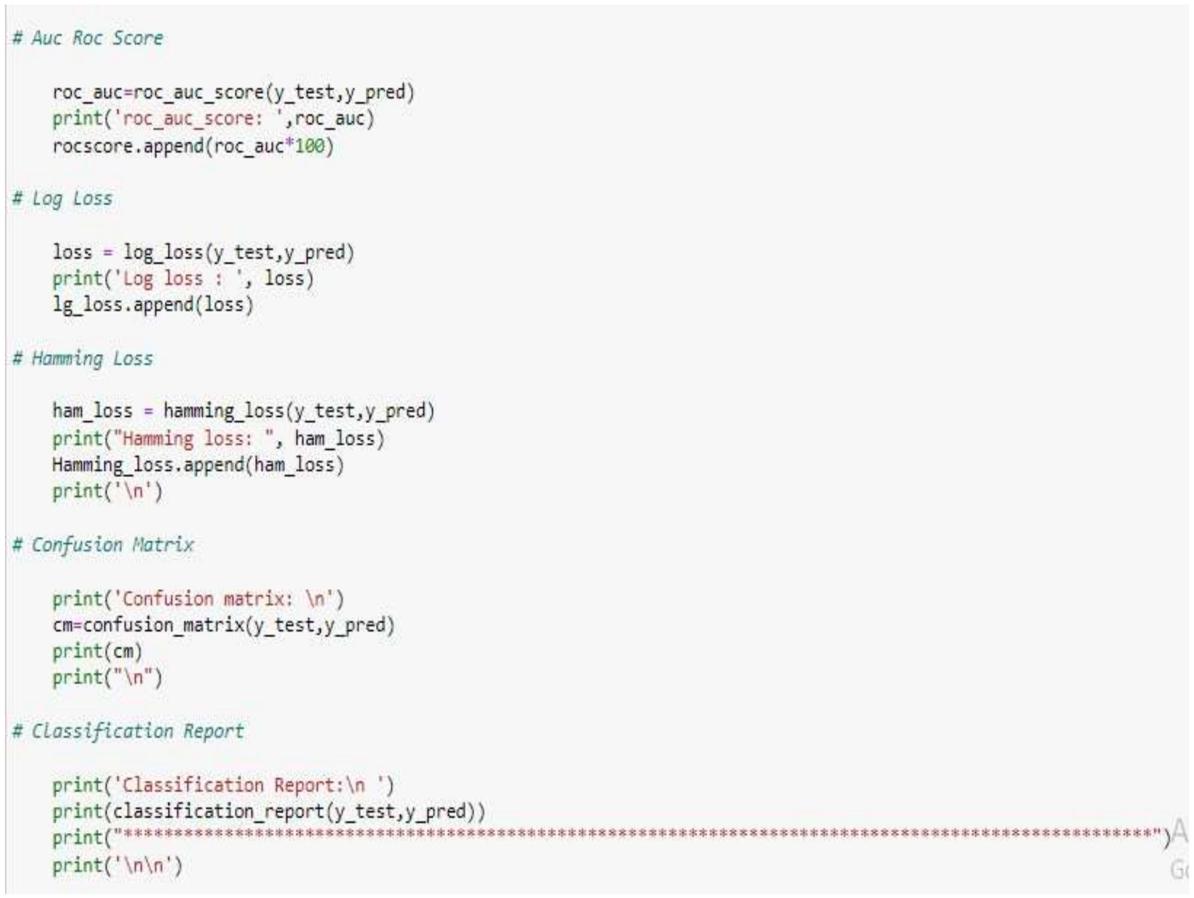
I have used different classification algorithms after choosing random state . First, I have created different classification algorithms and are appended in the variable models. Then, ran a for loop which contained the accuracy of the models along with different evaluation metrics.



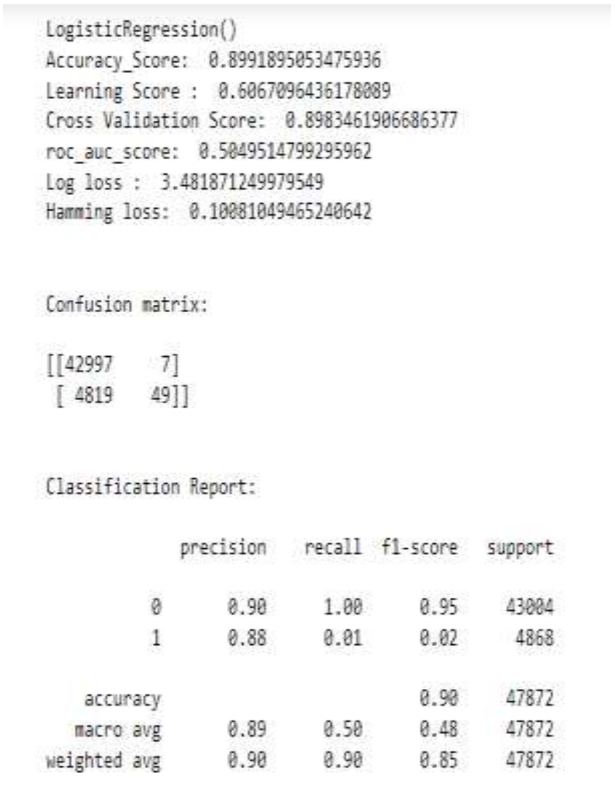
```
# Creating instances for different classifiers

LR = LogisticRegression()
MNB = MultinomialNB()
GB = GradientBoostingClassifier()
SVC = LinearSVC()
DTC = DecisionTreeClassifier()
ABC = AdaBoostClassifier()

# 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(('GradientBoostingClassifier',GB))
models.append(('LinearSVC',SVC))
models.append(('DecisionTreeClassifier',DTC))
models.append(('AdaBoostClassifier',ABC))
```



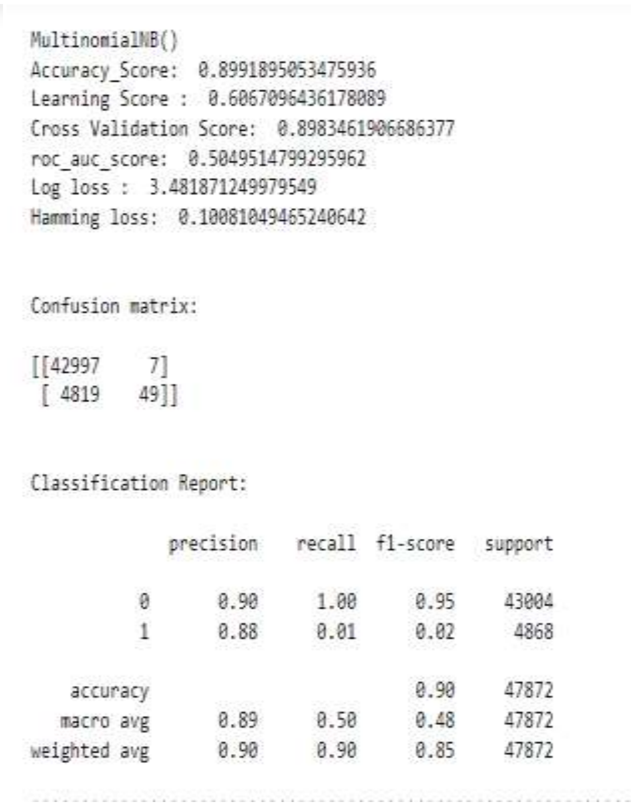
```
# creating empty Lists
Model = []
Score = []
Acc_score = []
cvs = []
rocscore = []
lg_loss = []
Hamming_loss = []

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)
```



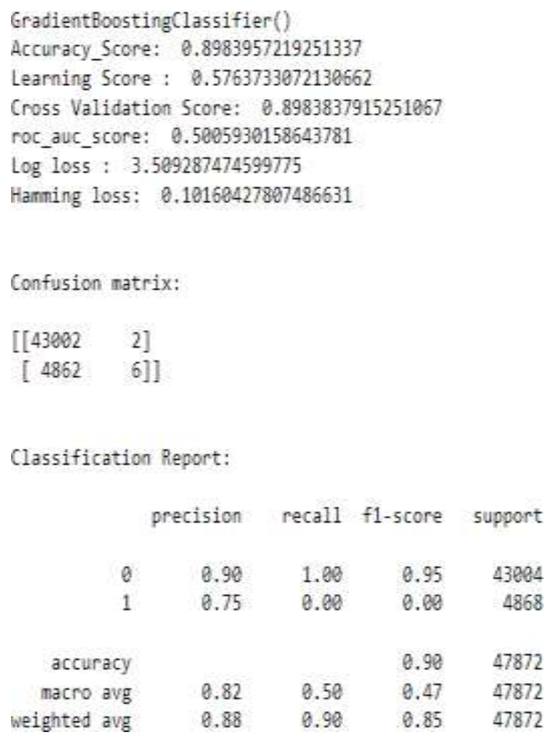
```
# Auc Roc Score

    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')
```



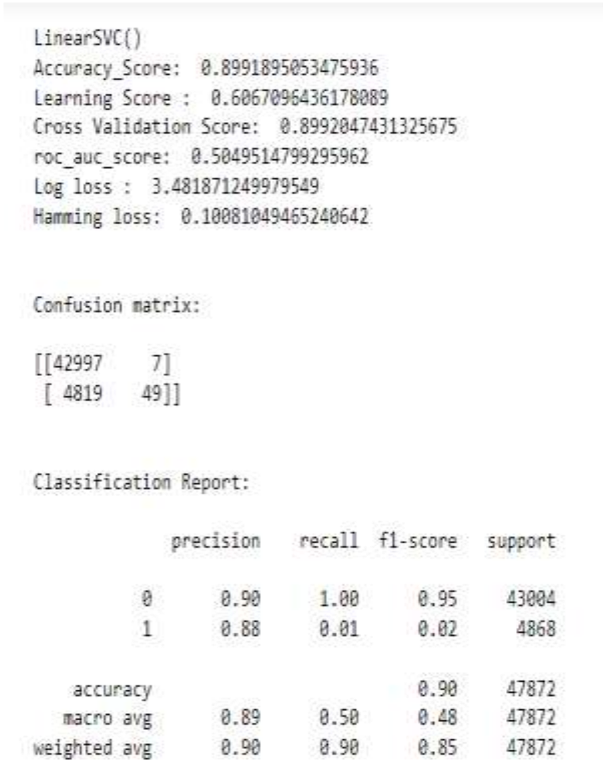
```
LogisticRegression()
Accuracy_Score:  0.8991895053475936
Learning Score :  0.6067096436178089
Cross Validation Score:  0.8983461906686377
roc_auc_score:  0.5049514799295962
Log loss :  3.481871249979549
Hamming loss:  0.10081049465240642


Confusion matrix:

[[42997     7]
 [ 4819    49]]


Classification Report:

              precision    recall  f1-score   support

           0       0.90      1.00      0.95     43004
           1       0.88      0.01      0.02      4868

    accuracy                           0.90     47872
   macro avg       0.89      0.50      0.48     47872
weighted avg       0.90      0.90      0.85     47872
```



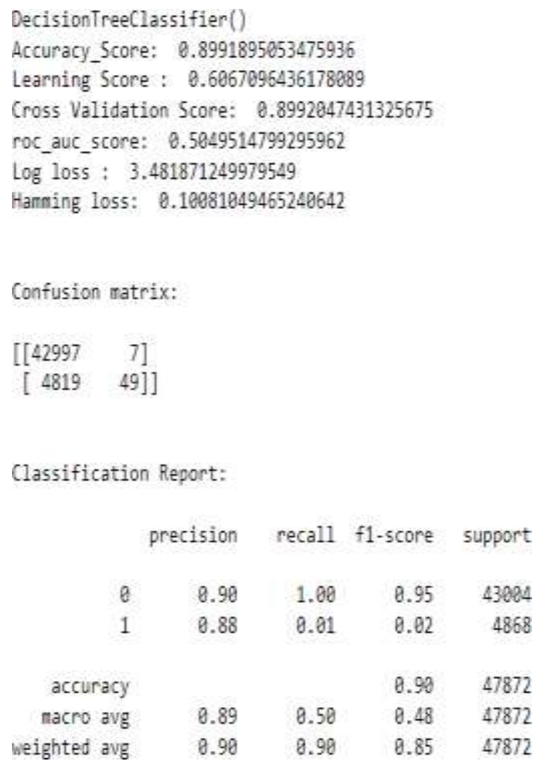
```
MultinomialNB()
Accuracy_Score:  0.8991895053475936
Learning Score :  0.6067096436178089
Cross Validation Score:  0.8983461906686377
roc_auc_score:  0.5049514799295962
Log loss :  3.481871249979549
Hamming loss:  0.10081049465240642


Confusion matrix:

[[42997     7]
 [ 4819    49]]


Classification Report:

              precision    recall  f1-score   support

           0       0.90      1.00      0.95     43004
           1       0.88      0.01      0.02      4868

    accuracy                           0.90     47872
   macro avg       0.89      0.50      0.48     47872
weighted avg       0.90      0.90      0.85     47872
```



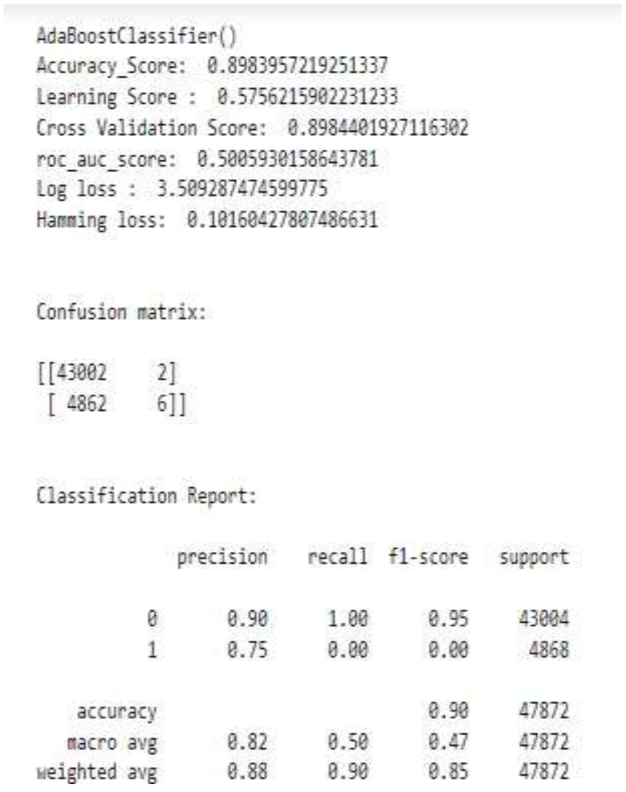
```
GradientBoostingClassifier()
Accuracy_Score:  0.8983957219251337
Learning Score :  0.5763733072130662
Cross Validation Score:  0.8983837915251067
roc_auc_score:  0.5005930158643781
Log loss :  3.509287474599775
Hamming loss:  0.10160427807486631


Confusion matrix:

[[43002     2]
 [ 4862     6]]


Classification Report:

              precision    recall  f1-score   support

           0       0.90      1.00      0.95     43004
           1       0.75      0.00      0.00      4868

    accuracy                           0.90     47872
   macro avg       0.82      0.50      0.47     47872
weighted avg       0.88      0.90      0.85     47872
```



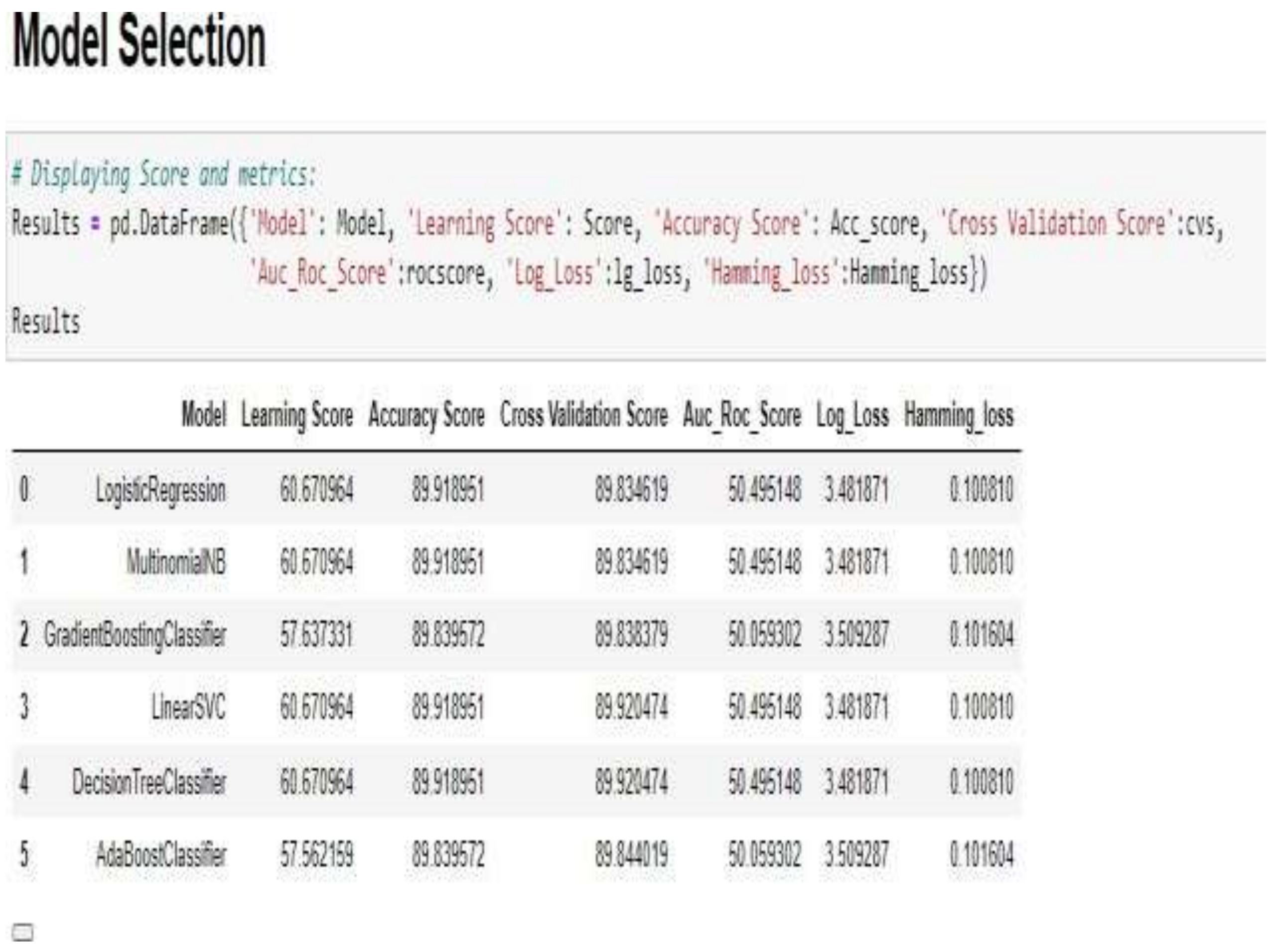
```
LinearSVC()
Accuracy_Score:  0.8991895053475936
Learning Score :  0.6067096436178089
Cross Validation Score:  0.8992047431325675
roc_auc_score:  0.5049514799295962
Log loss :  3.481871249979549
Hamming loss:  0.10081049465240642


Confusion matrix:

[[42997     7]
 [ 4819    49]]


Classification Report:

              precision    recall  f1-score   support

           0       0.90      1.00      0.95     43004
           1       0.88      0.01      0.02      4868

    accuracy                           0.90     47872
   macro avg       0.89      0.50      0.48     47872
weighted avg       0.90      0.90      0.85     47872
```



```
DecisionTreeClassifier()
Accuracy_Score:  0.8991895053475936
Learning Score :  0.6067096436178089
Cross Validation Score:  0.8992047431325675
roc_auc_score:  0.5049514799295962
Log loss :  3.481871249979549
Hamming loss:  0.10081049465240642


Confusion matrix:

[[42997     7]
 [ 4819    49]]


Classification Report:

              precision    recall  f1-score   support

           0       0.90      1.00      0.95     43004
           1       0.88      0.01      0.02      4868

    accuracy                           0.90     47872
   macro avg       0.89      0.50      0.48     47872
weighted avg       0.90      0.90      0.85     47872
```



```
AdaBoostClassifier()
Accuracy_Score:  0.8983957219251337
Learning Score :  0.5756215902231233
Cross Validation Score:  0.8984401927116302
roc_auc_score:  0.5005930158643781
Log loss :  3.509287474599775
Hamming loss:  0.10160427807486631


Confusion matrix:

[[43002     2]
 [ 4862     6]]


Classification Report:

              precision    recall  f1-score   support

           0       0.90      1.00      0.95     43004
           1       0.75      0.00      0.00      4868

    accuracy                           0.90     47872
   macro avg       0.82      0.50      0.47     47872
weighted avg       0.88      0.90      0.85     47872
```



Model Selection

```
# Displaying Score 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	60.670964	89.918951	89.834619	50.495148	3.481871	0.100810
1	MultinomialNB	60.670964	89.918951	89.834619	50.495148	3.481871	0.100810
2	GradientBoostingClassifier	57.637331	89.839572	89.838379	50.059302	3.509287	0.101604
3	LinearSVC	60.670964	89.918951	89.920474	50.495148	3.481871	0.100810
4	DecisionTreeClassifier	60.670964	89.918951	89.920474	50.495148	3.481871	0.100810
5	AdaBoostClassifier	57.562159	89.839572	89.844019	50.059302	3.509287	0.101604

After creating and training different classification algorithms, we can see that the difference between accuracy and cross validation score is less for " Gradient Boosting Classifier" and AdaBoost Classifier. But GradientBoosting Classifier giving less loss vales, auc roc score and high accuracy score compared to AdaBoost Classifier. On this basis I can conclude that "Gradient Boosting Classifier" 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 parameters in       Classifier

# Extreme       Classifier
parameters = {'max_depth': [3,6,9],
             'max_features': ['auto','sqrt','log2'],
             'learning_rate': [0.1,0.25,0.5],
             'min_samples_leaf': [1,50,100]}



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

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

```
GridSearchCV(cv=5, estimator=GradientBoostingClassifier(),
             param_grid={'learning_rate': [0.1, 0.25, 0.5],
                         'max_depth': [3, 6, 9],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 50, 100]},
             scoring='accuracy')
```

I have used 4 GradientBoosting classifier 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

```
# getting best parameters
GCV.best_params_
```

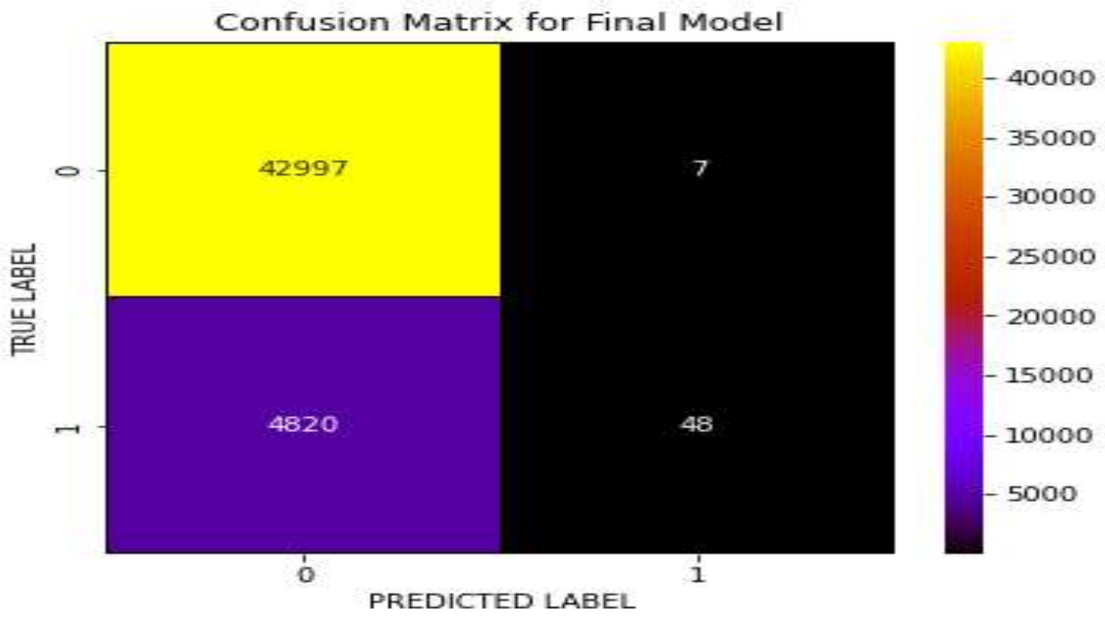
```
{'learning_rate': 0.5,
 'max_depth': 9,
 'max_features': 'auto',
 'min_samples_leaf': 1}
```

These are the best parameters obtained after running GridSearchCV.

Creating final model

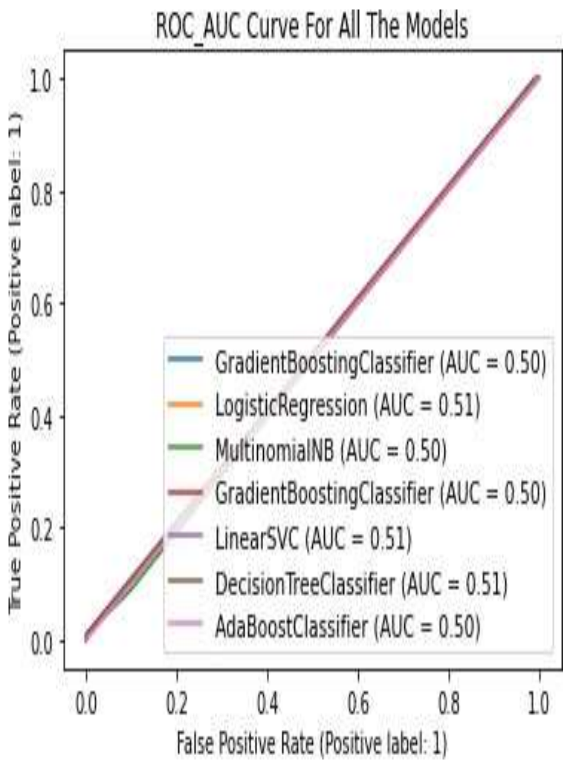
```
# Creating final model
comment_model = GradientBoostingClassifier(learning_rate= 0.5,max_depth= 9,max_features= 'auto',min_samples_leaf =1)
comment_model.fit(train_x, train_y)
pred = comment_model.predict(x_test)
acc_score = accuracy_score(y_test,pred)
print("Accuracy score: ", acc_score*100)
roc_auc = 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))
```

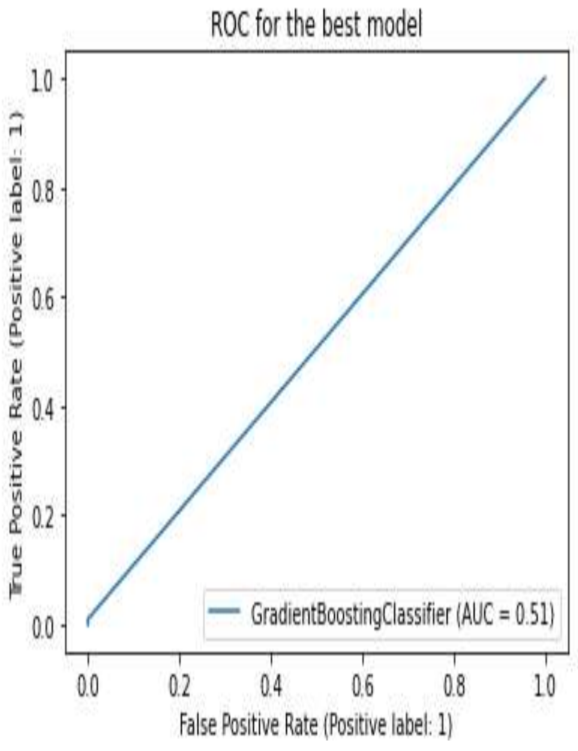
```
Accuracy score:  89.91686163101605
roc_auc_score:  50.05930158643781
Log loss :  3.482592731772558
Hamming loss:  0.10083138368983957


Confusion Matrix:
 [[42997     7]
 [ 4820    48]]


Classification Report:
               precision    recall  f1-score   support

           0       0.90      1.00      0.95     43004
           1       0.87      0.01      0.02      4868

    accuracy                           0.90     47872
   macro avg       0.89      0.50      0.48     47872
weighted avg       0.90      0.90      0.85     47872
```



I have successfully incorporated the hyper parameter tuning using best parameters of GradientBoosting Classifier and the accuracy of the model has been increased after hyperparameter tuning and received the accuracy score as 89.91% 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..





I have generated the ROC Curve for all the models used here and it shows the AUC score for the models.I have generated the ROC Curve for my final model and it shows the AUC score for my final model to be of 51% which is increased after tuning the model.

Saving The Model

```
# 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

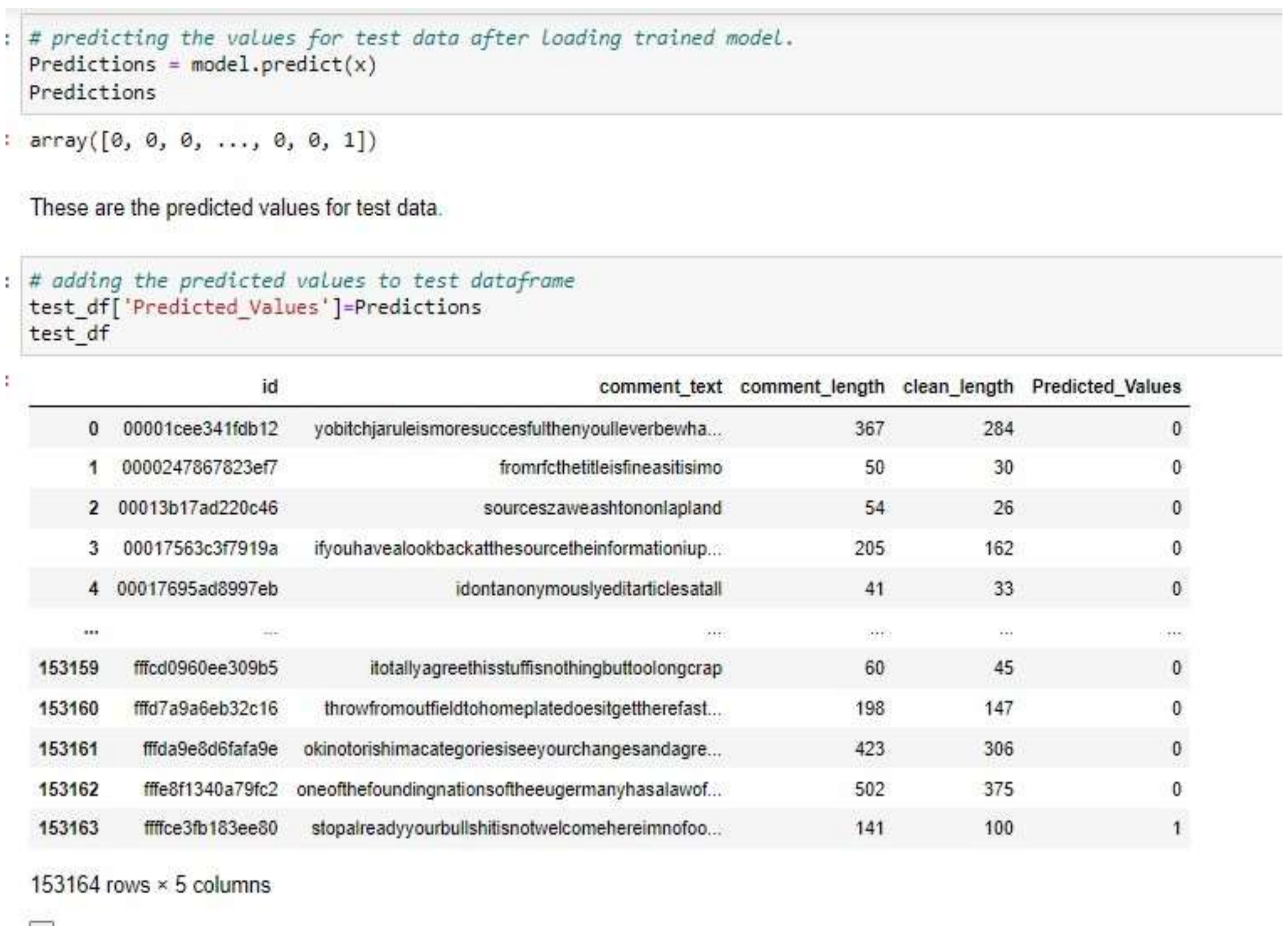
```
# predicting the trained final model
comment_model.predict(X)
```

```
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.

Prediction for test dataset using final model



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

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

These are the predicted values for test data.

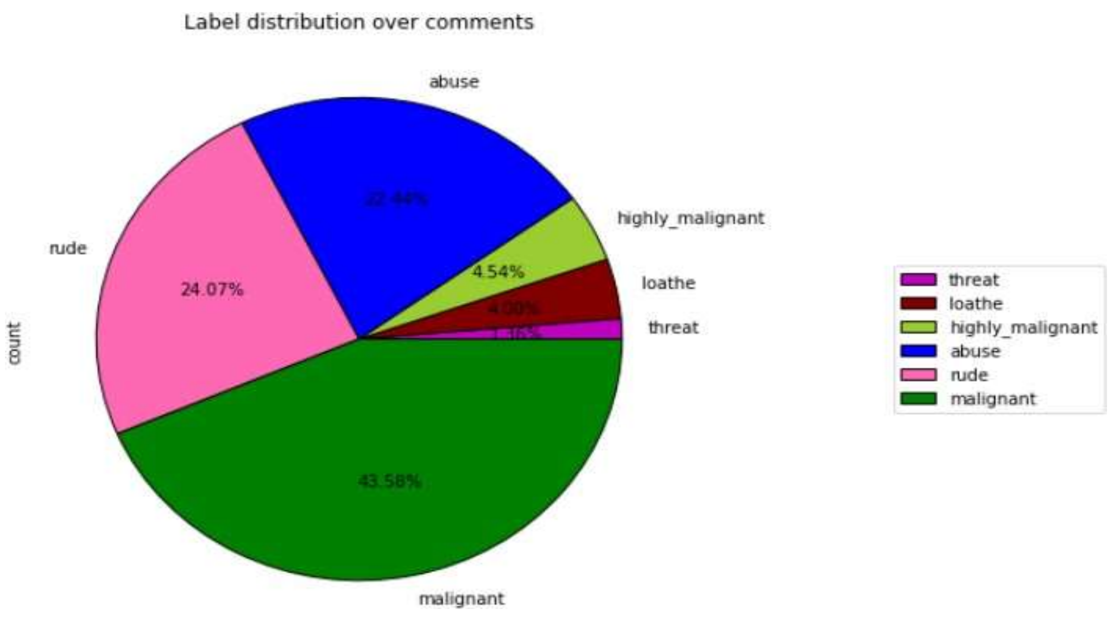
```
# adding the predicted values to test dataframe
test_df['Predicted_Values']=Predictions
test_df
```

	id	comment_text	comment_length	clean_length	Predicted_Values
0	00001cee341fdb12	yobitchjaruleismoresuccesfulthenyoulleverbewha...	367	284	0
1	0000247867823ef7	fromrfcthetitleisfineasitisimo	50	30	0
2	00013b17ad220c46	sourceszaweashtononlapland	54	26	0
3	00017563c3f7919a	ifyouhavealookbackatthesourcetheinformationiup...	205	162	0
4	00017695ad8997eb	idontanonymouslyeditarticlesatall	41	33	0
...	...	...	...	...	...
153159	fffcd0960ee309b5	itotallyagreethisstuffisnothingbuttoolongcrap	60	45	0
153160	fffd7a9a6eb32c16	throwfromoutfieldtohomeplatedoesitgettherefast...	198	147	0
153161	fffda9e8d6fafa9e	okinotorishimacategoriesiseeyourchangesandagre...	423	306	0
153162	fffe8f1340a79fc2	oneofthefoundingnationsoftheeugermanyhasalawof...	502	375	0
153163	ffffce3fb183ee80	stopalreadyyourbullshitisnotwelcomehereimnofoo...	141	100	1

153164 rows × 5 columns

➢ *Visualizations*

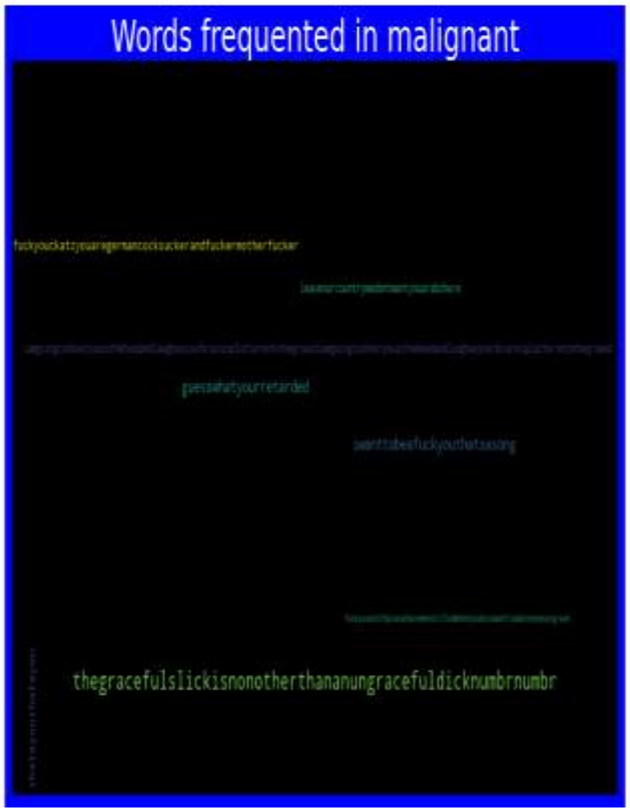
I used pandas profiling to get the over viewed visualization on the pre-processed data. Pandas is an open-source Python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable.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

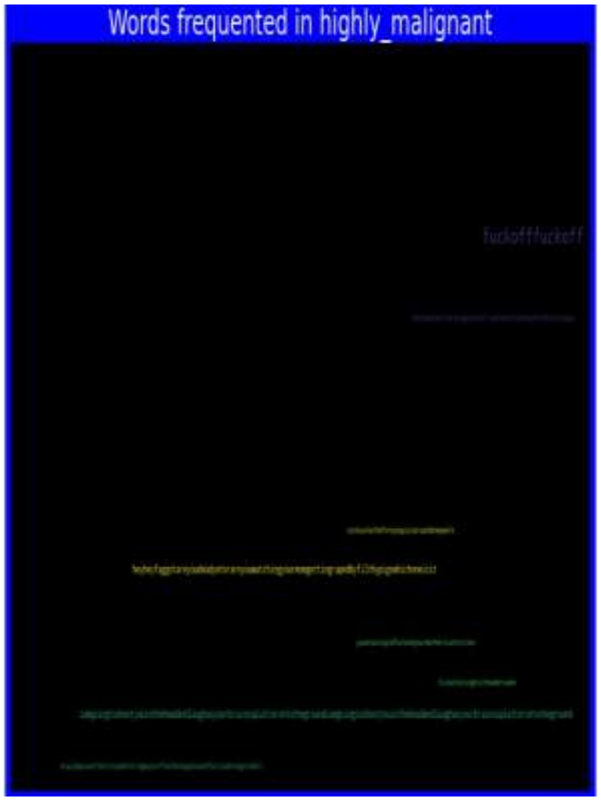


Observation:

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 comments are high compared to other type of comments and the count of threat comments are very less.

Plotting WordCloud for each label :





Words frequented in rude
thegracefulslickisnonotherthananungracefuldicknumbrnumbr
fuckyouckatzyouaregermancocksuckerandfuckermotherfucker
iwanttobeafuckyouthatsasong

Words frequented in threat
iamgoingtokillyouiamgoingtomurderyou
iwillkillyouandwearyourskinlikearobe
iwillcastrateyouandroastyourtesticlestheniwillfeedthemtoyou

Words frequented in abuse
guesswhatyourretarded
userabdismuslimscum
thegracefulslickisnonotherthananungracefuldicknumbrnumbr
fuckyouckatzyouaregermancocksuckerandfuckermotherfucker

Words frequented in loathe
fuckyouckatzyouaregermancocksuckerandfuckermotherfucker
userabdismuslimscum

Observations:

- From the above plots we can clearly see the toxic words which are indication of malignant, highly malignant, rude, threat, abuse and loathe words.
- Here most frequently words used for each label is displayed in the word cloud based on different label and also when all the values are present.

➢ *Interpretation of the Results*

<u>Visualization :</u> 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 WorldClouds I would able to sense the loud words in each label. AUC-ROC curve helped to select the best model.

<u>Pre-processing :</u> 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.

<u>Model Building :</u> 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 Gradient Boosting Classifier(GBC) as the best model which gives 89.83% accuracy score. I checked the cross-validation score ensuring there will be no overfitting. Agter tuning the best model GB Classifier, I got 89.91% accuracy score and also got increment in AUC-ROC curve. 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 trining 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, GB 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 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.

****