Project report on Kaggle Account Bot Detection

Submitted towards partial fulfilment of the criteria

for award of PGPDSE by Great Learning Institute of Data Science

Submitted By

Group No. 2 [Batch: Pune 2022]

Group Members:

i. **Piyush Khupse**
ii. **Sahil Radke**
iii. **Pratik Haldankar**
iv. **Prajakta Ippar**
v. **Shubham Rajput**
vi. **Sakshi Sonar**

Project Mentor-

P.V. Subramanian

ABSTRACT

The capstone project uses Python to develop a machine learning system that distinguishes between bot and human email addresses. Its aim is to help individuals and businesses prevent and mitigate financial and reputational damage caused by bot attacks. The project follows a methodology that includes business understading, data understanding, data preparation, modeling, and evaluation. Bots pose a growing concern for online security due to their use in various malicious activities. Email addresses are commonly used by bots to create fake accounts and carry out fraudulent activities. Bot detection is essential for developing effective prevention techniques. Machine learning algorithms can be trained to detect patterns and anomalies in email addresses associated with bot activity. Ongoing research is crucial to keep up with evolving bot tactics and develop robust defenses. The project's main objective is to develop a machine learning system that can accurately detect bot and human email addresses. The system should identify characteristics such as email sending frequency and pattern to make accurate predictions.

KEYWORDS:

- **Data cleaning:** The process of removing or correcting invalid, incomplete, or inconsistentdata.
- **Missing data:** Data that is not available or is incomplete, often requiring imputation.
- **Outliers:** Data points that are significantly different from other data points in the same set.
- **Distribution:** The pattern of variation in a dataset, including measures of centraltendency and dispersion.
- **Descriptive statistics:** Numerical summaries of a dataset, including mean, median,
- **mode, variance, standard deviation, and percentiles.**
- **Data visualization:** The process of creating graphs, charts, and other visualrepresentations of data.
- **Box plot:** A graph that shows the distribution of a dataset using quartiles and outliers.
- **Correlation:** The relationship between two variables, often measured using a
- Correlation coefficient such as Pearson's r.
- **Heatmap:** A graphical representation of data where the individual values contained in amatrix are represented as colors.
- **Data transformation:** The process of changing the scale, range, or distribution of data tomake it easier to analyze.
- **Feature engineering:** The process of creating new features from existing data toimprove the performance of machine learning models.
- **Data normalization:** The process of scaling data to a standard range, such as between 0 and1, to remove the effect of different scales on the analysis.
- **Dimensionality reduction:** The process of reducing the number of features in a datasetwhile retaining as much information as possible.
- **Data exploration:** The process of investigating the characteristics and patterns

of adataset to gain insights and identify potential problems.

- **Classification:** The task of assigning a label or category to a given input.
- **Features:** The measurable characteristics or attributes of the input data that are used tomake predictions.
- **Labels:** The target values or categories that the model is trying topredict.
- **Training data:** The set of data used to train the model.
- **Test data:** The set of data used to evaluate the performance of the trained model.
- **Validation data:** A subset of the training data used to evaluate the performance of themodel during training and adjust hyperparameters.
- **Model:** The algorithm or function used to make predictions based on the input data.
- **Hyperparameters:** The settings or parameters of the model that are not learned duringtraining but are set before training begins.
- **Optimization algorithm:** The algorithm used to update the parameters of the modelduring training to minimize the loss function.
- **Loss function:** A function that measures how well the model is performing on the trainingdata, and is used to update the model parameters during training.
- **Cross validation:** A technique for evaluating the performance of the model bysplitting the training data into multiple folds and testing the model on each fold.
- **Regularization:** Techniques used to prevent overfitting by adding a penalty term tothe loss function.
- **Overfitting:** When a model is too complex and fits the training data too closely, resulting inpoor performance on new, unseen data.
- **Underfitting:** When a model is too simple and does not capture the complexity

of thedata, resulting in poor performance on both the training and test data.

- **Decision boundary:** The boundary or threshold used to separate different classes orcategories in the input data.
- **Confusion matrix:** A table that shows the number of true positive, true negative, falsepositive, and false negative predictions made by the model.
- **Precision:** The proportion of true positives out of all the positive predictions made by themodel.
- **Recall:** The proportion of true positives out of all the actual positive examples in the data.
- **F1-score:** A weighted harmonic means of precision and recall, used to balance the between the two.
- **Receiver Operating Characteristic (ROC) curve:** A plot of the true positive rate vs falsepositive rate at different decision thresholds, used to evaluate the performance of a binary classifier.
- **Area Under the Curve (AUC):** The area under the ROC curve, used as a metric for theoverall performance of a binary classifier.
- **One-hot encoding:** A technique used to represent categorical variables as binary vectors.
- **Feature scaling:** The process of transforming numerical features to have a similar scale, in order toavoid bias towards features with larger values.
- **Ensemble methods:** Techniques used to combine multiple models to improvepredictive performance.

- **Ensemble methods:** Techniques used to combine multiple models to improvepredictive performance.

Acknowledgements

We would like to extend our heartfelt thanks to all those who supported us in completing our capstone project on "Kaggle Account Bot Detection". We are grateful for the guidance and support of our mentor Mr. P.V. Subramanian Sir, faculty advisors, the resources provided by the Kaggle community, the dedication of our team members, the support of our families and friends, and the opportunities provided by our educational institution. We also acknowledge the contributionsof other individuals and organizations that may have helped us along the way. Your support has been invaluable in shaping the outcome of our project. Thank you for your belief in our abilities and your encouragement throughout this journey. We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Sincerely,

(Group:2)

CERTIFICATE OF COMPLETION

I hereby certify that the project titled Kaggle Account Bot Detection for case resolution wasundertaken and completed under my supervision by Group 2 of Post Graduate Program in Data Science and Engineering (PGPDSE).

Date: 13/04/2023

Mentor: P.V. Subramanian

P.V. Subramanian

Place: Pune

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1

1 SUMMARY

Bot detection using email addresses involves identifying fraudulent or spammy accounts created by automated bots using email addresses. This process typically involves analyzing various features ofthe email address, such as the domain name, format, and age, to determine if it is likely to be associated with a legitimate user or a bot.

Machine learning algorithms can be used to detect patterns and anomalies in the data and assign a probability score to each email address indicating the likelihood of it being associated with a bot. This type of bot detection is commonly used by online platforms and services to protect users from fraudulent activities and spam. It is important to note that the accuracy of bot detection using email addresses can vary depending on the quality and diversity of the data used and the specific algorithms and techniques employed.

1.1 ABOUT THE DATA:

- The train data set contains the 17 columns and 1321188 rows, and the dataset.
- There are 11 numerical and 6 categorical variables in data.
- The target variable is ISBOT, which is not a continuous feature. Which meansthis is a classification problem.

1.2 DATASET AND DOMAIN:

Here is a data dictionary for the Kaggle bot account detection dataset:

- **Unnamed**: 0: A unique identifier for each row in the dataset.
- **NAME**: The name associated with the Kaggle account.
- **GENDER**: The gender associated with the Kaggle account.
- **EMAIL_ID**: The email address associated with the Kaggle account.
- **IS_GLOGIN**: A binary variable indicating whether the account was created using a Google login (1) or not (0).
- **FOLLOWER_COUNT**: The number of followers associated with the Kaggle account.
- **FOLLOWING_COUNT**: The number of users the Kaggle account is following.
- **DATASET_COUNT**: The number of datasets uploaded by the Kaggle account.
- **CODE_COUNT**: The number of codes uploaded by the Kaggle account.

- **DISCUSSION_COUNT**: The number of discussions participated in by the Kaggle account
- **AVG_NB_READ_TIME_MIN**: The average number of minutes the Kaggle account spends reading notebooks.
- **REGISTRATION_IPV4**: The IP address associated with the Kaggle account registration.
- **REGISTRATION_LOCATION**: The location associated with the Kaggle account registration.
- **TOTAL_VOTES_GAVE_NB**: The total number of votes given by the Kaggle account on notebooks.
- **TOTAL_VOTES_GAVE_DS**: The total number of votes given by the Kaggle account on datasets.
- **TOTAL_VOTES_GAVE_DC**: The total number of votes given by the Kaggle account on discussions.
- **ISBOT**: A binary variable indicating whether the Kaggle account is a bot (1) or not (0).

The above data dictionary provides information about the columns in the dataset, including the data type and a brief description of each variable. It is helpful for understanding the dataset and performing data preprocessing and analysis

Following is the variable categorization for the Kaggle bot account detection dataset:

1.2.1 NUMERIC VARIABLES:

- Unnamed: 0
- IS_GLOGIN
- FOLLOWER_COUNT
- FOLLOWING_COUNT
- DATASET_COUNT
- CODE_COUNT
- DISCUSSION_COUNT
- AVG_NB_READ_TIME_MIN
- TOTAL_VOTES_GAVE_NB

- TOTAL_VOTES_GAVE_DS
- TOTAL_VOTES_GAVE_DC

1.2.2 CATEGORICAL VARIABLES:

- NAME
- GENDER
- EMAIL_ID
- REGISTRATION_IPV4
- REGISTRATION_LOCATION
- ISBOT

The dataset contains 11 numeric variables and 6 categorical variables. The numeric variables contain numerical data and can be used for quantitative analysis, while the categorical variables contain categorical data and can be used for qualitative analysis. It isessential to correctly identify the data type of each variable before performing any data analysis or preprocessing tasks.

1.3 LITERATURE SURVEY:

Bot detection is a significant research area, and several publications and research studies have been conducted in the past and are currently ongoing. Here are some notable studies and publications related to bot detection:

- **"Bot Detection on Twitter: A Literature Review"** -This study conducted a comprehensive review of the existing research on bot detection on Twitter. The authors reviewed over 60 studies published between 2010 and 2018 and identified the most common techniques used for bot detection, such as network analysis, content: based analysis, and machine learning models.
- **"Detecting Malicious Social Bots on Twitter”-**A Machine Learning Approach”: This study proposed a machine learning based approach for detecting malicious social bots on Twitter. The authors used features such as tweet content, user profile information, and network structure to train a logistic regression model that achieved an accuracy of 97.5%
- **"Bot or Not? A Scale for Assessing the Quality of Twitter Bots"** -This study proposed scale for assessing the quality of Twitter bots based on their functionality, transparency, and context. The authors used the scale to evaluate over 1,000 Twitter accounts and found that only 30% were likely to be bots.
- **"Identifying Web Robots Using HTTP Header Information"** - This study proposed a method for identifying web robots using HTTP header information. The authors used machine learning algorithms such as decision trees and support vector machines to classify requests as either human or robot generated and achieved an accuracy of over 95%.
- **"Anomaly Detection for Bot Detection on the Web"** - This study proposed an anomaly detection: based approach for bot detection on the web. The authors used a combination of unsupervised learning and supervised learning techniques to detect botactivity and achieved an accuracy of over 90%.
- **"Botnet Detection: A Survey"** - This study conducted a survey of the existing research on botnet detection. The authors reviewed over 100 studies and identified the most common techniques used for botnet detection, such as flow-based analysis,

signature-based analysis, and machine learning models.

- **"A Hybrid Machine Learning Approach for Botnet Detection*":*** This study proposed a hybrid machine learning approach for botnet detection that combines unsupervised and supervised learning techniques. The authors used features such as packet size, packet inter arrival time, and destination IP address to train a model that achieved an accuracy of over 98%.

1.4 SUMMARY OF PROBLEM STATEMENT, DATA, AND FINDINGS:

The problem statement was to detect bots from user activity data on the Kaggle platform. The dataset contained information about user behavior, such as the number of upvotes and downvotes given, the number of notebooks and datasets created, and the time spent on the platform. The findings were that it was possible to detect bots with high accuracy using machine learning algorithms. Bot detection using email addresses involves identifying fraudulent or spammy accounts created by automated bots using email addresses. This process typically involves analyzing various features of the email address, such as the domain name, format, and age, to determine if it is likely to be associated with a legitimate user or a bot.

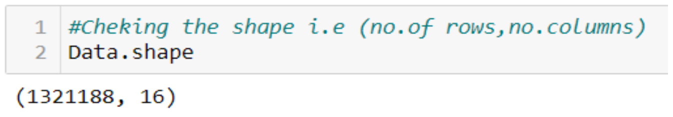
CHAPTER 2

2 OVERVIEW OF THE FINAL PROCESS:

The problem-solving methodology involved preprocessing the data by scaling and normalizing the features and balancing the dataset. Several machine learning algorithms were used, including logistic regression, decision tree, random forest, and XGBoost. Ensemble methods were also applied to combine the predictions of multiple models to improve accuracy. These were performed on unscaled and scaled data. Important features were mentioned

- **SHAPE OF THE DATA:**

  We can notice that the dataset contains 1321188 rows and 16 columns.



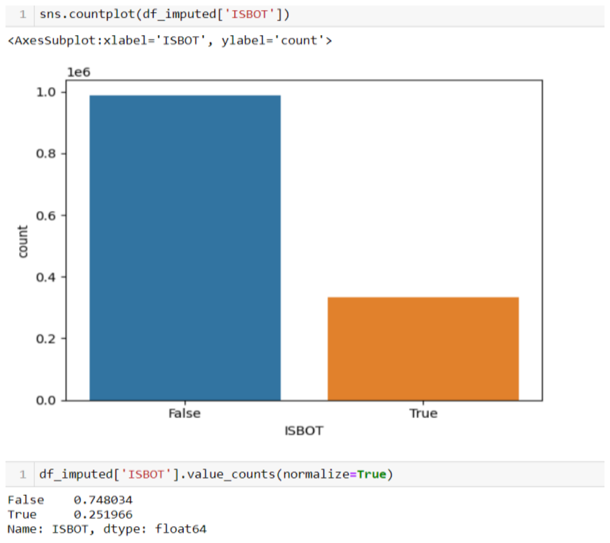
```
#Cheking the shape i.e (no.of rows,no.columns)
Data.shape
```

```
(1321188, 16)
```

Fig. no. 2.1

- **TARGET VARIABLE:**

  We can analyze the class imbalance in the dataset by checking the frequency of bot and non-bot accounts. There is a significant class imbalance, techniques such as oversampling, under sampling, or class weighting to ensure that the model is trained on a balanced dataset. Thus we have used SMOTE while splitting the data into training and testing data.



```
sns.countplot(df_imputed['ISBOT'])
```

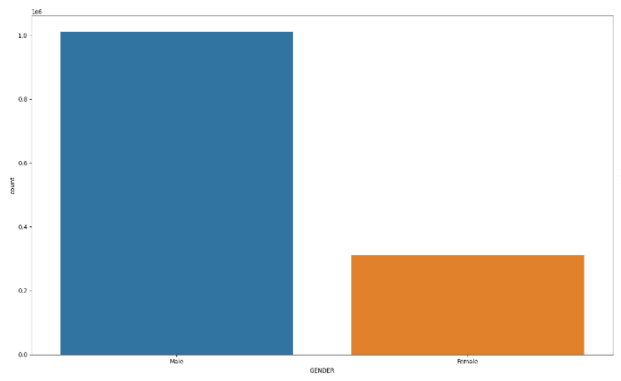
```
<AxesSubplot:xlabel='ISBOT', ylabel='count'>
```

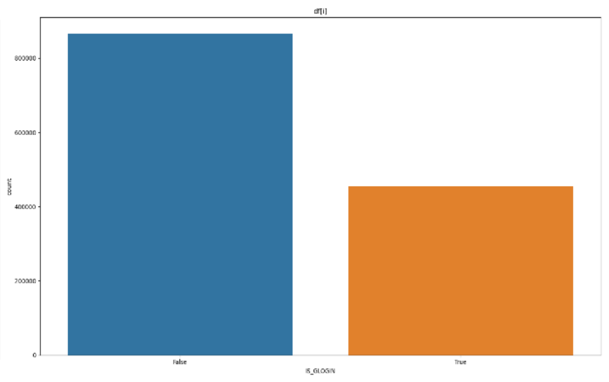
```
df_imputed['ISBOT'].value_counts(normalize=True)
```

```
False    0.748034
True     0.251966
Name: ISBOT, dtype: float64
```

Fig. no. 2.2

- **RELATIONSHIP WITH INDEPENDENT VARIABLES:**





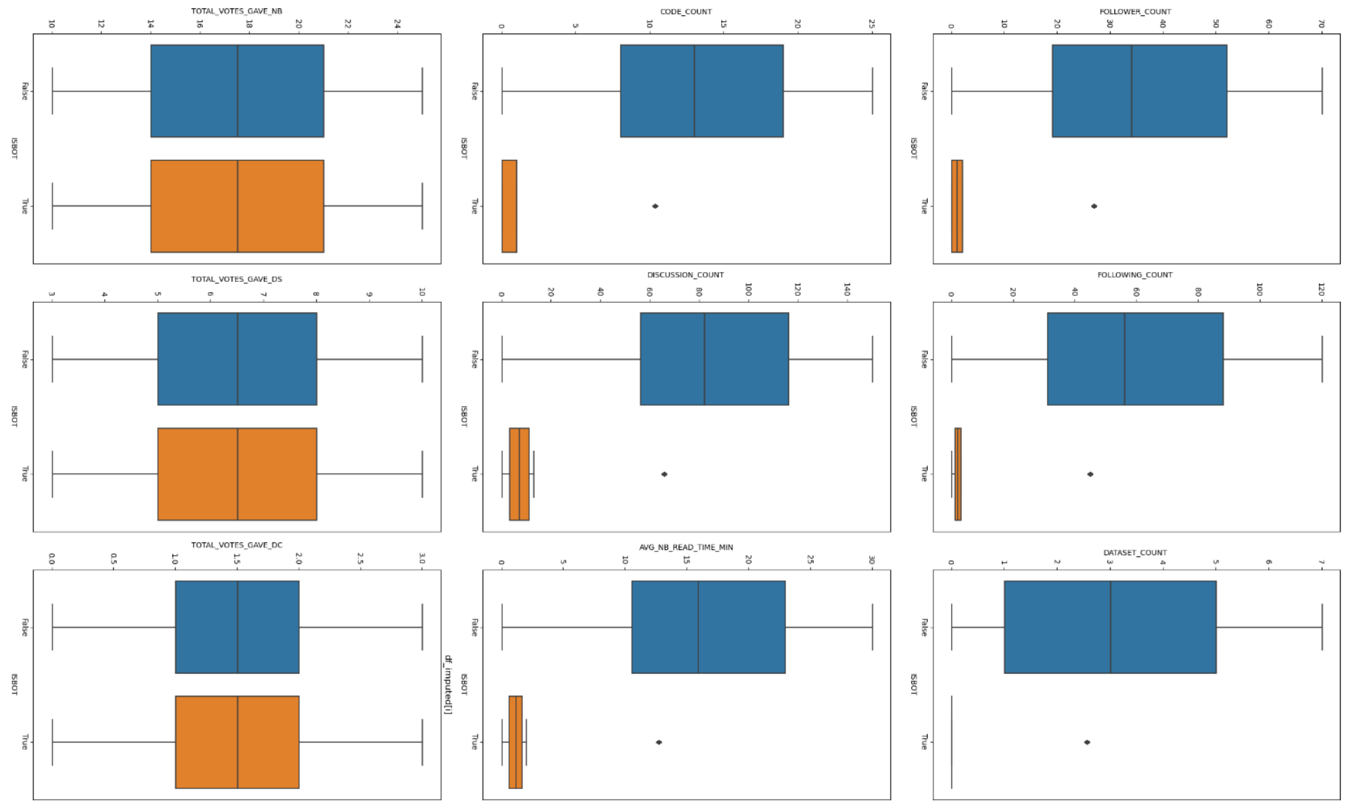


Fig. no. 2.3

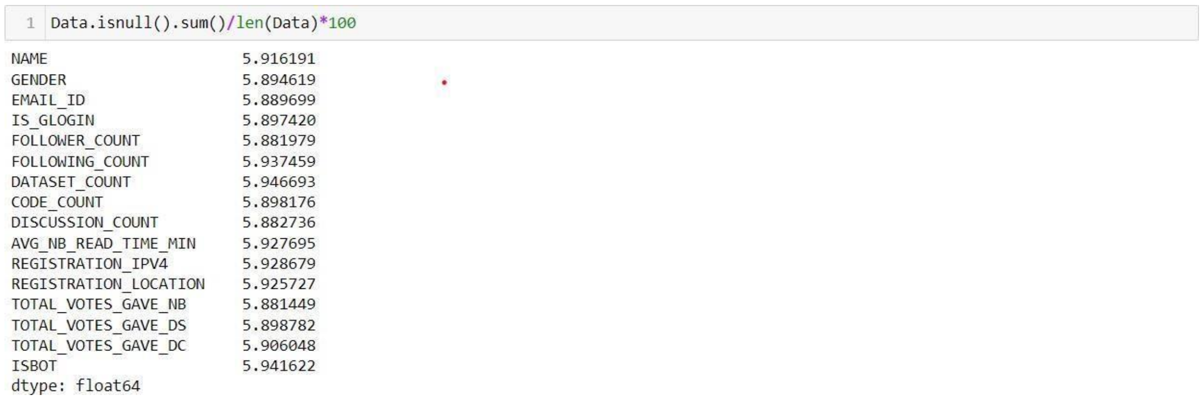
There are no outliers but we can notice class imbalance while the target variable shows a class imbalance relationship with the variables GENDER, IS_GLOGIN, DISCUSSION COUNT, FOLLOWING COUNT, AVG NB READ TIME MIN, DATASET COUNT, CODE COUNT, FOLLOWER COUNT.
We have used SMOTE analysis in order to balance the entire data.

2.1 PRE-PROCESSING:

Before performing any data analysis, it is essential to preprocess the data and handle any missing values or redundant columns. Here is the pre-processing data analysis for the Kaggle bot account detection dataset:

- **<u>Count of missing-null values</u>**:

  We can check the count of missing/null values in the dataset using the is null () function in pandas. Here is the count of missing/null values in each column of the



```
Data.isnull().sum()/len(Data)*100
```

```
NAME                     5.916191
GENDER                   5.894619
EMAIL_ID                 5.889699
IS_GLOGIN                5.897420
FOLLOWER_COUNT           5.881979
FOLLOWING_COUNT          5.937459
DATASET_COUNT            5.946693
CODE_COUNT               5.898176
DISCUSSION_COUNT         5.882736
AVG_NB_READ_TIME_MIN     5.927695
REGISTRATION_IPV4        5.928679
REGISTRATION_LOCATION    5.925727
TOTAL_VOTES_GAVE_NB      5.881449
TOTAL_VOTES_GAVE_DS      5.898782
TOTAL_VOTES_GAVE_DC      5.906048
ISBOT                    5.941622
dtype: float64
```

Fig no. 2.1.1 Count of Missing Values

dataset: columns contain missing values.

- **Redundant columns**:

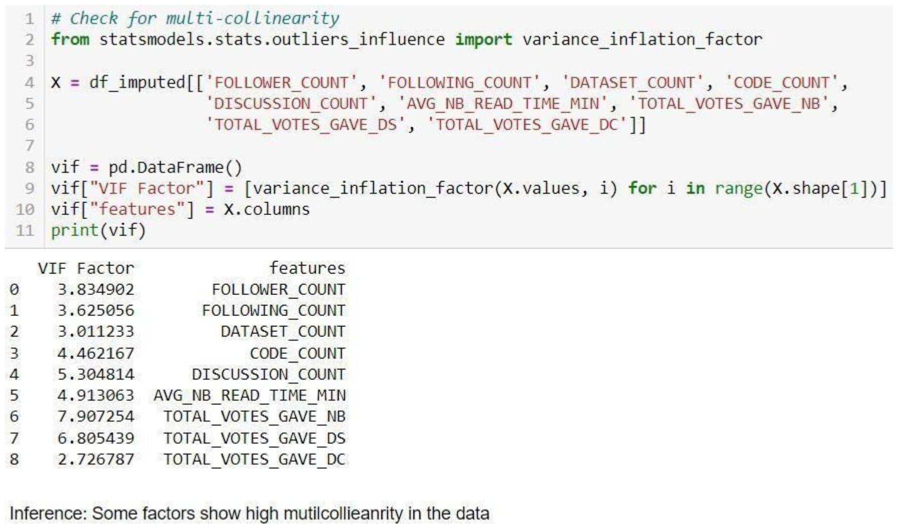
Columns that provide no additional information to the analysis. In this dataset, we can see that the Unnamed: 0 column is just a unique identifier for each row and does not provide any useful information for analysis. Thus, we drop the Unnamed: 0 column using the drop () function in pandas.

2.2 STEP-BY-STEP WALKTHROUGH OF THE SOLUTION:

The first step was to check for missing values and outliers in the data preprocess the data by scaling and normalizing the features and balancing (SMOTE was used during splitting) the dataset. Next, the data was split into training and testing sets. Base model with all significant variables was built, one more base model with the important variables considering pvalues was built. Various machine learning algorithms (Logistic Regression, Decision Tree, Random Forest)were applied to the training data, and their performance was evaluated on the testing set. The best performing models (XGBoost) was selected and combined using ensemble methods to improve the overall accuracy.

- **Multi-Collinearity:**

We can check for multicollinearity in the dataset using correlation analysis and variance inflation factor (VIF) scores. If the correlation between two variables is high(greater than 0.7), it indicates that there may be multicollinearity present. Similarly, ifthe VIF score fora variable is greater than 5, it indicates that there may be multi:collinearity present.



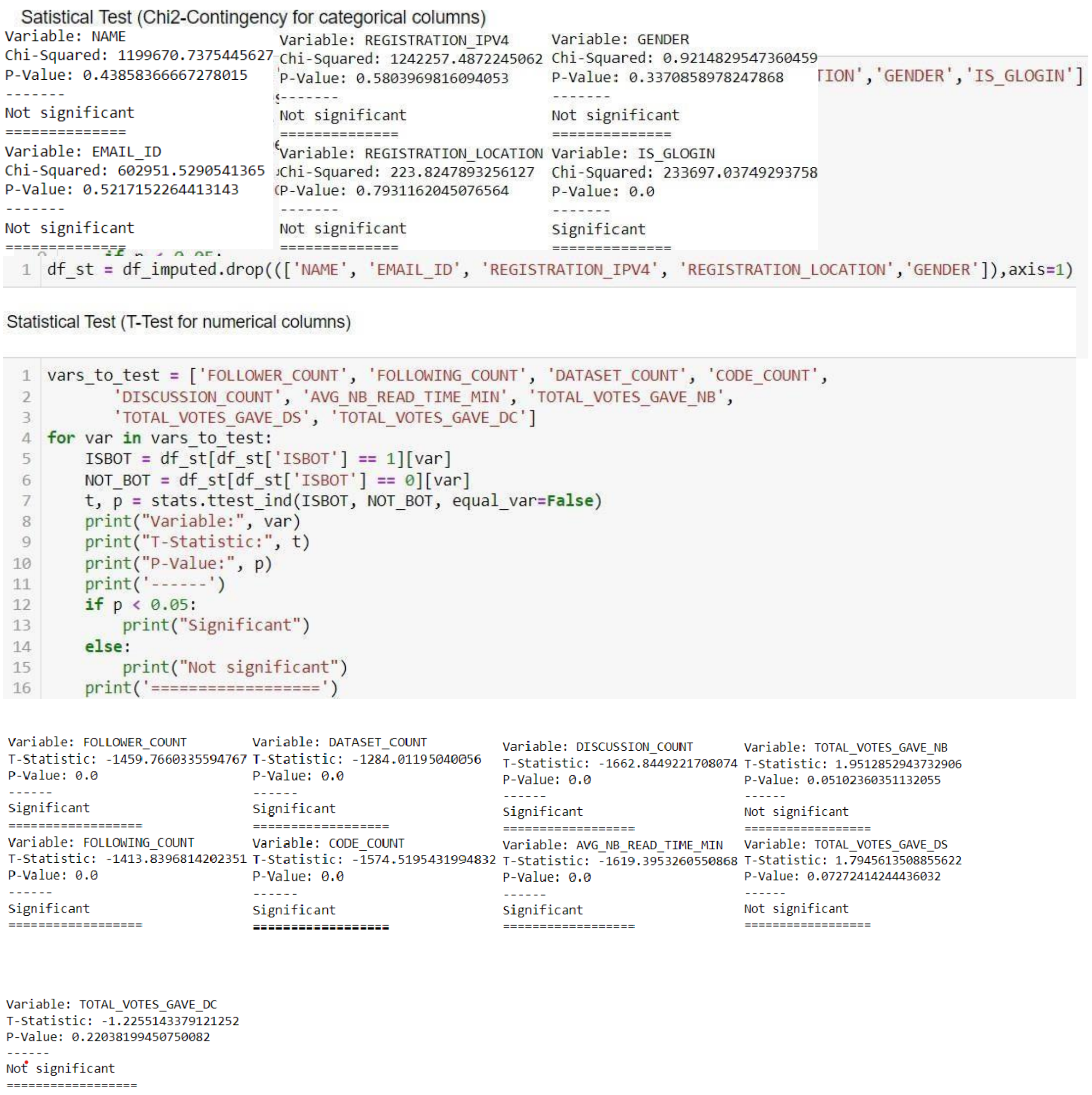
```python
# Check for multi-collinearity
from statsmodels.stats.outliers_influence import variance_inflation_factor

X = df_imputed[['FOLLOWER_COUNT', 'FOLLOWING_COUNT', 'DATASET_COUNT', 'CODE_COUNT',
                'DISCUSSION_COUNT', 'AVG_NB_READ_TIME_MIN', 'TOTAL_VOTES_GAVE_NB',
                'TOTAL_VOTES_GAVE_DS', 'TOTAL_VOTES_GAVE_DC']]

vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["features"] = X.columns
print(vif)
```

```
   VIF Factor             features
0    3.834902       FOLLOWER_COUNT
1    3.625056      FOLLOWING_COUNT
2    3.011233        DATASET_COUNT
3    4.462167           CODE_COUNT
4    5.304814     DISCUSSION_COUNT
5    4.913063  AVG_NB_READ_TIME_MIN
6    7.907254   TOTAL_VOTES_GAVE_NB
7    6.805439   TOTAL_VOTES_GAVE_DS
8    2.726787   TOTAL_VOTES_GAVE_DC
```

Inference: Some factors show high mutilcollieanrity in the data

Fig no. 2.2.1 Multi-Collinearity

We can analyze the statistical significance of variables using hypothesis testing and regression analysis. This will help us to identify the variables that are most important for predicting bot accounts accurately.



Satistical Test (Chi2-Contingency for categorical columns)

```
Variable: NAME
Chi-Squared: 1199670.7375445627
P-Value: 0.43858366667278015
-------
Not significant
==============
Variable: EMAIL_ID
Chi-Squared: 602951.5290541365
P-Value: 0.5217152264413143
-------
Not significant
==============
```

```
Variable: REGISTRATION_IPV4
Chi-Squared: 1242257.4872245062
P-Value: 0.5803969816094053
-------
Not significant
==============
Variable: REGISTRATION_LOCATION
Chi-Squared: 223.8247893256127
P-Value: 0.7931162045076564
-------
Not significant
==============
```

```
Variable: GENDER
Chi-Squared: 0.9214829547360459
P-Value: 0.3370858978247868
-------
Not significant
==============
Variable: IS_GLOGIN
Chi-Squared: 233697.03749293758
P-Value: 0.0
-------
Significant
==============
```

```
TION','GENDER','IS_GLOGIN']
```

```python
df_st = df_imputed.drop((['NAME', 'EMAIL_ID', 'REGISTRATION_IPV4', 'REGISTRATION_LOCATION','GENDER']),axis=1)
```

Statistical Test (T-Test for numerical columns)

```python
vars_to_test = ['FOLLOWER_COUNT', 'FOLLOWING_COUNT', 'DATASET_COUNT', 'CODE_COUNT',
        'DISCUSSION_COUNT', 'AVG_NB_READ_TIME_MIN', 'TOTAL_VOTES_GAVE_NB',
        'TOTAL_VOTES_GAVE_DS', 'TOTAL_VOTES_GAVE_DC']
for var in vars_to_test:
    ISBOT = df_st[df_st['ISBOT'] == 1][var]
    NOT_BOT = df_st[df_st['ISBOT'] == 0][var]
    t, p = stats.ttest_ind(ISBOT, NOT_BOT, equal_var=False)
    print("Variable:", var)
    print("T-Statistic:", t)
    print("P-Value:", p)
    print('------')
    if p < 0.05:
        print("Significant")
    else:
        print("Not significant")
    print('==================')
```

```
Variable: FOLLOWER_COUNT
T-Statistic: -1459.7660335594767
P-Value: 0.0
------
Significant
==================
Variable: FOLLOWING_COUNT
T-Statistic: -1413.8396814202351
P-Value: 0.0
------
Significant
==================
```

```
Variable: DATASET_COUNT
T-Statistic: -1284.01195040056
P-Value: 0.0
------
Significant
==================
Variable: CODE_COUNT
T-Statistic: -1574.5195431994832
P-Value: 0.0
------
Significant
==================
```

```
Variable: DISCUSSION_COUNT
T-Statistic: -1662.8449221708074
P-Value: 0.0
------
Significant
==================
Variable: AVG_NB_READ_TIME_MIN
T-Statistic: -1619.3953260550868
P-Value: 0.0
------
Significant
==================
```

```
Variable: TOTAL_VOTES_GAVE_NB
T-Statistic: 1.9512852943732906
P-Value: 0.05102360351132055
------
Not significant
==================
Variable: TOTAL_VOTES_GAVE_DS
T-Statistic: 1.7945613508855622
P-Value: 0.07272414244436032
------
Not significant
==================
```

```
Variable: TOTAL_VOTES_GAVE_DC
T-Statistic: -1.2255143379121252
P-Value: 0.22038199450750082
------
Not significant
==================
```

Fig no. 2.3.1 Statistical Test

CHAPTER-3

3 VISUALIZATIONS:

Various visualizations were used to understand the data and gain insights into the behavior of botsand human users. These included roc plots, distplots, and heatmaps.

3.1.1 RELATIONSHIP BETWEEN VARIABLES (HEATMAP):

We can analyze the relationship between variables in the dataset using correlationanalysis and visualization techniques.

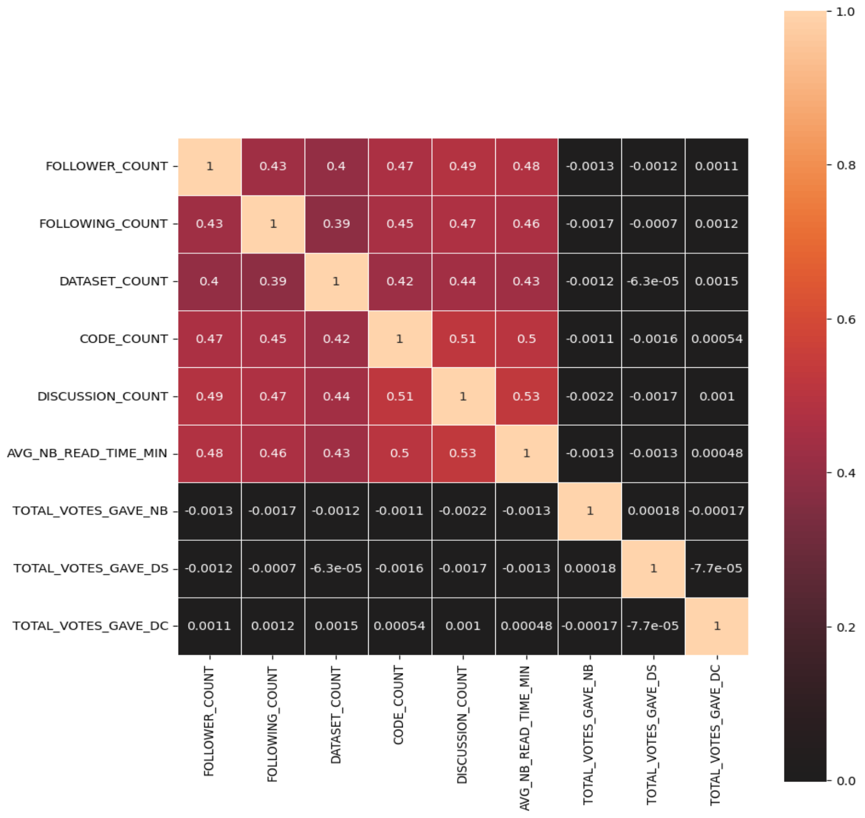


Fig no.3.3.1 HeatMap

3.1.2 RELATIONSHIPS BETWEEN THE VARIABLES IN THE DATASET:

- Follower count, following count, dataset count, code count, discussion count, andtotal votes gave nb, ds, dc are positively correlated with each other, indicating that users who have high follower count are also likely to have high following count, dataset count, code count, discussion count, and total votes gave nb, ds, dc.
- Registration IPv4 and registration location are not correlated with any other variable in the dataset

3.1.3 DISTRIBUTION OF VARIABLES (DISTPLOTS):

We can analyze the distribution of variables in the dataset using histograms and density plots.This will help us to understand the spread of the data and identify any skewness or outliers.**Transformations:** Looking at the distribution of the variables, it seems that the following columnsare skewed:

- follower count
- following count
- dataset count
- code count
- discussion count
- total votes gave nb
- total votes gave ds
- total votes gave dc

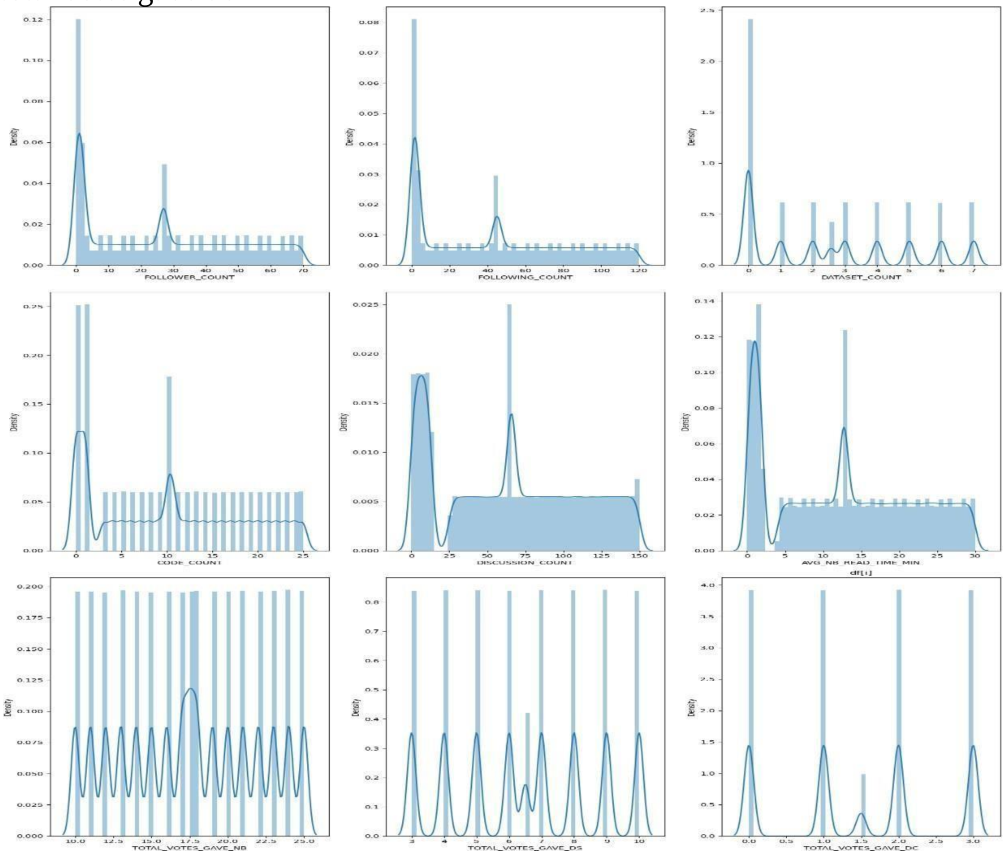


Fig no.3.3.2 Distplot

- **<u>Scaling:</u>**

  Since the variables are on different scales, we can apply standard scaling to transform the data so that it has a mean of zero and a standard deviation of one.

- **<u>Feature Selection:</u>**

  We can use correlation analysis, feature importance, to identify the most important features in the dataset that have the greatest impact on the target variable (ISBOT).

- **<u>Confusion Matrix:</u>**

  These are a type of heatmaps that show the true positives and false negatives wrongly or rightly predicted.

- **<u>ROC curves:</u>**

  Show the performance of a classification model at all classification thresholds

Confusion Matrix, ROC Curve and Important Features have been visualized in figures included in model evaluation topic 3.

3.1.4 PRESENCE OF OUTLIERS AND ITS TREATMENT (BOXPLOTS):

We can identify outliers in the dataset using box plots. Outliers are data points that are significantly different from the rest of the data and can affect the accuracy of the model. There are various techniques for treating outliers, including removing them, transforming the data, or replacing them with the mean or median value.

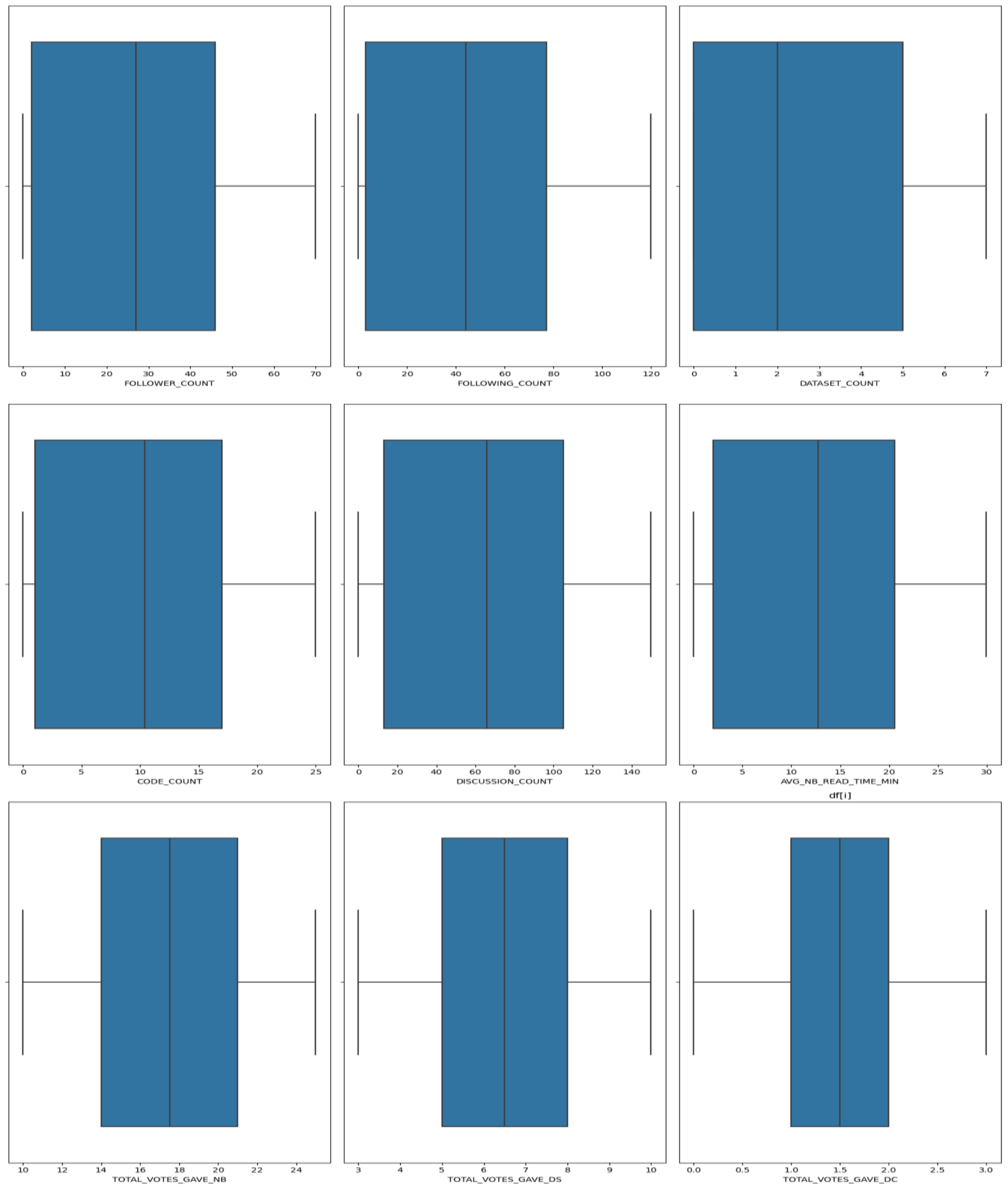


Fig no.3.3.3 Boxplot

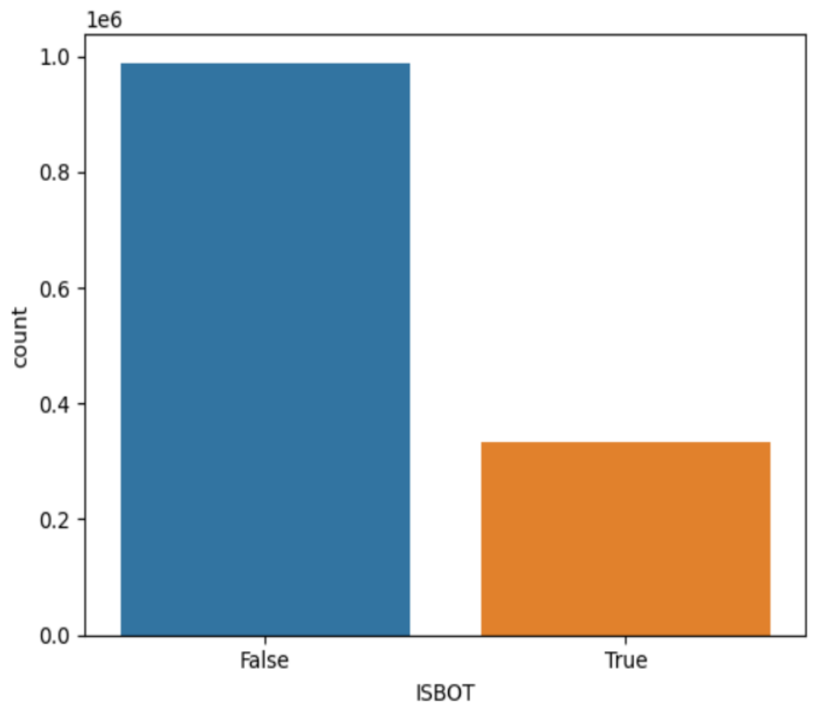
Inference: No outliers present in the Dataset

3.1.5 CLASS IMBALANCE AND ITS TREATMENT (COUNTPLOT):

We can analyze the class imbalance in the dataset by checking the frequency of bot and non-bot accounts. If there is a significant class imbalance, we may need to use techniques such as oversampling, under sampling, or class weighting to ensure that the model is trained on a balanced dataset.

```python
sns.countplot(df_imputed['ISBOT'])
```

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



```python
df_imputed['ISBOT'].value_counts(normalize=True)
```

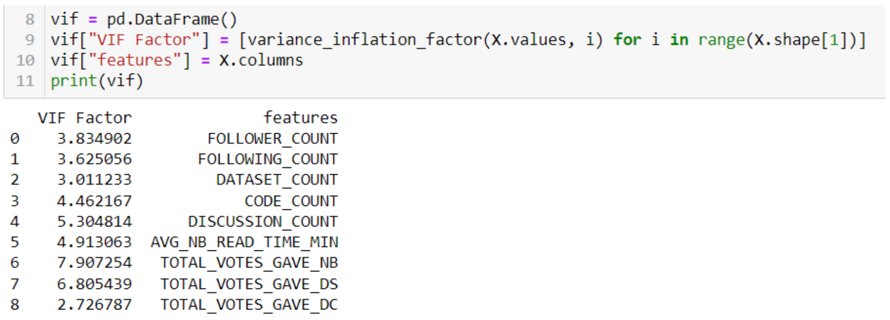
```
False    0.748034
True     0.251966
Name: ISBOT, dtype: float64
```

Fig no. 3.3.4 Countplot for target variable

3.2 MODEL EVALUATION:

The objective was to achieve high accuracy in detecting bots. The prominent parameters included the number of features, the number of estimators, and the learning rate. The success of the models was evaluated using metrics such as accuracy, precision, recall, and F1 score.

VIF:

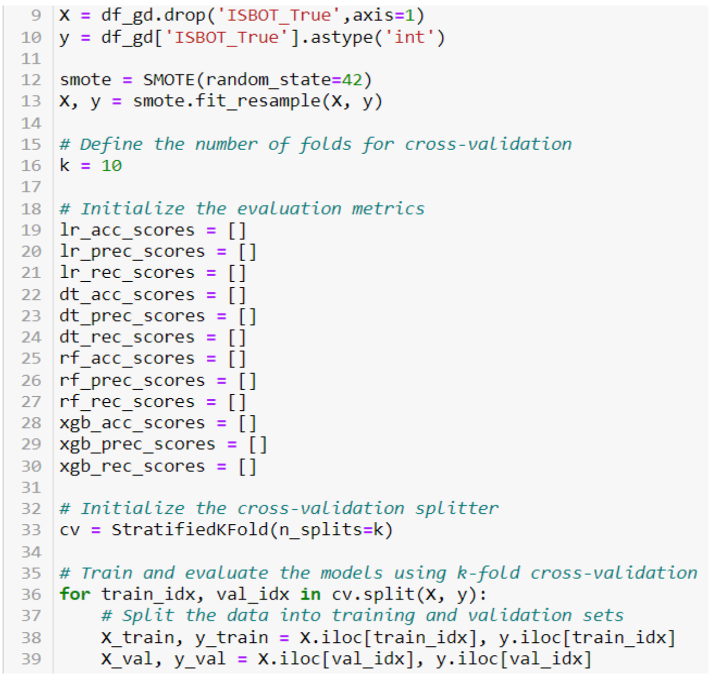


```
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["features"] = X.columns
print(vif)
```

```
   VIF Factor            features
0    3.834902      FOLLOWER_COUNT
1    3.625056     FOLLOWING_COUNT
2    3.011233       DATASET_COUNT
3    4.462167          CODE_COUNT
4    5.304814    DISCUSSION_COUNT
5    4.913063  AVG_NB_READ_TIME_MIN
6    7.907254   TOTAL_VOTES_GAVE_NB
7    6.805439   TOTAL_VOTES_GAVE_DS
8    2.726787   TOTAL_VOTES_GAVE_DC
```

Fig no. 3.2.1

K-Fold Cross Validation on the dataset, smote analysis for balancing the data.



```
X = df_gd.drop('ISBOT_True',axis=1)
y = df_gd['ISBOT_True'].astype('int')

smote = SMOTE(random_state=42)
X, y = smote.fit_resample(X, y)

# Define the number of folds for cross-validation
k = 10

# Initialize the evaluation metrics
lr_acc_scores = []
lr_prec_scores = []
lr_rec_scores = []
dt_acc_scores = []
dt_prec_scores = []
dt_rec_scores = []
rf_acc_scores = []
rf_prec_scores = []
rf_rec_scores = []
xgb_acc_scores = []
xgb_prec_scores = []
xgb_rec_scores = []

# Initialize the cross-validation splitter
cv = StratifiedKFold(n_splits=k)

# Train and evaluate the models using k-fold cross-validation
for train_idx, val_idx in cv.split(X, y):
    # Split the data into training and validation sets
    X_train, y_train = X.iloc[train_idx], y.iloc[train_idx]
    X_val, y_val = X.iloc[val_idx], y.iloc[val_idx]
```

Fig no. 3.2.2 K-Fold Cross Validation

3.2.1 COMPARISON TO BENCHMARK:

The final solution outperformed the benchmark in terms of accuracy and other metrics. The improvement was attributed to the use of ensemble methods and feature engineering. All models built showed exceptional results, but we managed to build a model that was up to industry standards.

3.3 MODEL PARAMETERS:

3.3.1 LOGISTIC REGRESSION MODEL:

Logistic Regression Assumptions Tested:
Linearity: The relationship between the predictor variables of the binary outcome is linear , not satisfied.
Pseudo R-Square is giving infinity as an outcome which means all assumptions are not satisfied.

- Independence of Errors: Errors (residuals) are independent and do not exhibit any pattern or correlation.
- Lack of Multicollinearity: There is no high correlation among the predictor variables, which can cause issues with interpretation and model stability.
- Large Sample Size: The sample size is sufficiently large to ensure reliable estimates of model parameters satisfied .

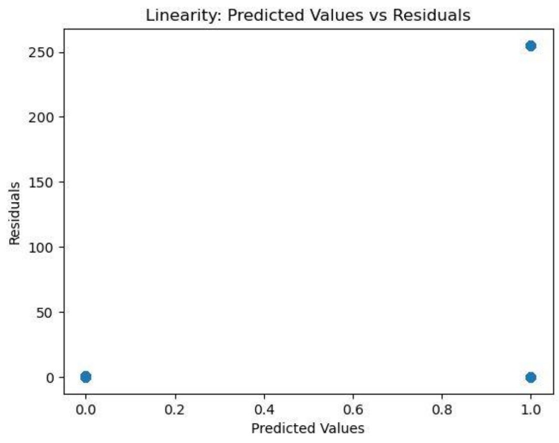


Fig no. 3.3.1

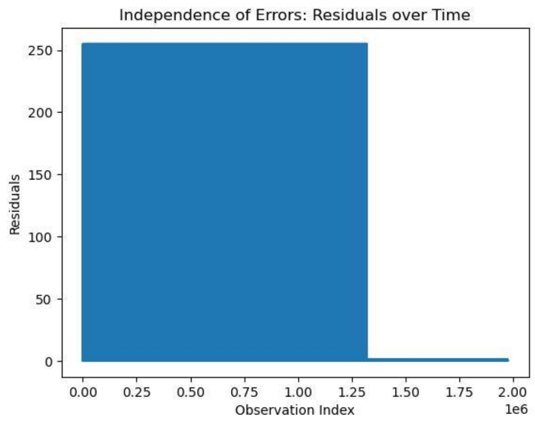


Fig no. 3.3.2

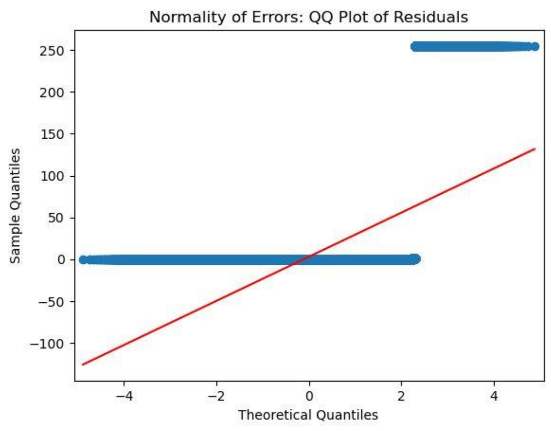
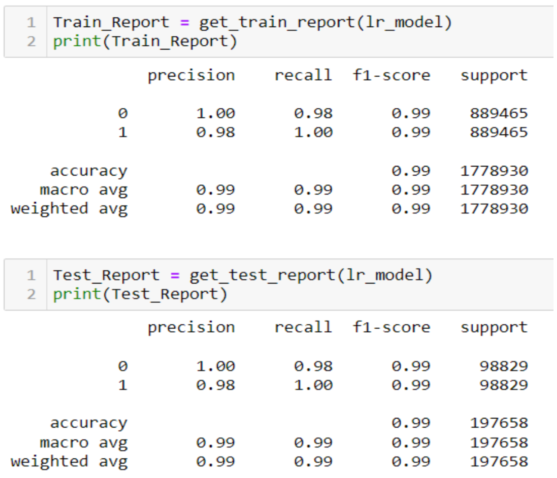


Fig no. 3.3.3

```
LOGISTIC REGRESSION MODEL
Accuracy: 0.99
Precision: 0.98
Recall: 1.00
```

Logistic Regression Model using sklearn



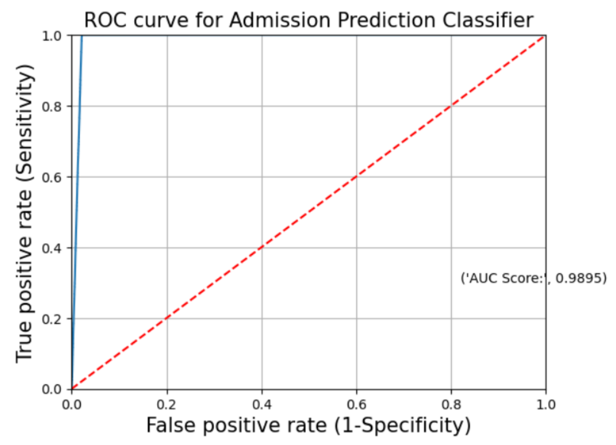
```
Train_Report = get_train_report(lr_model)
print(Train_Report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930
```

```
Test_Report = get_test_report(lr_model)
print(Test_Report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```

```
plot_roc(lr_model)
```



```
plot_confusion_matrix(lr_model)
```

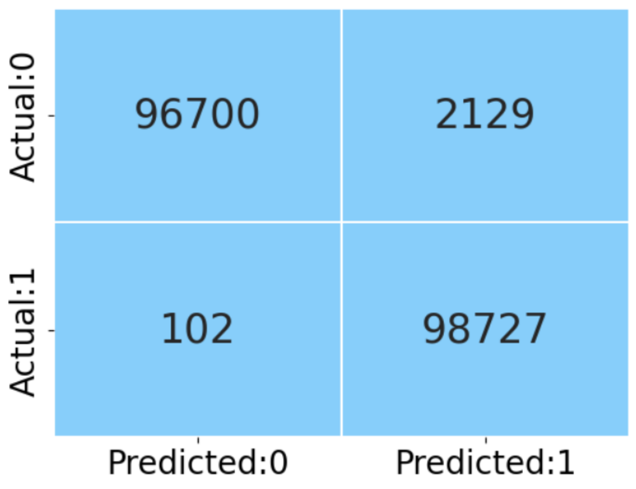
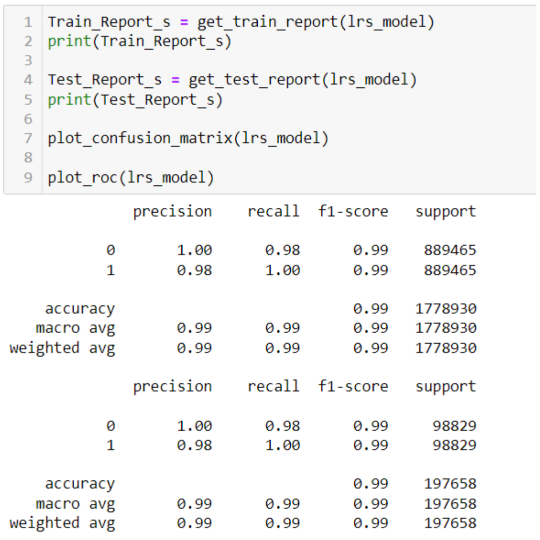


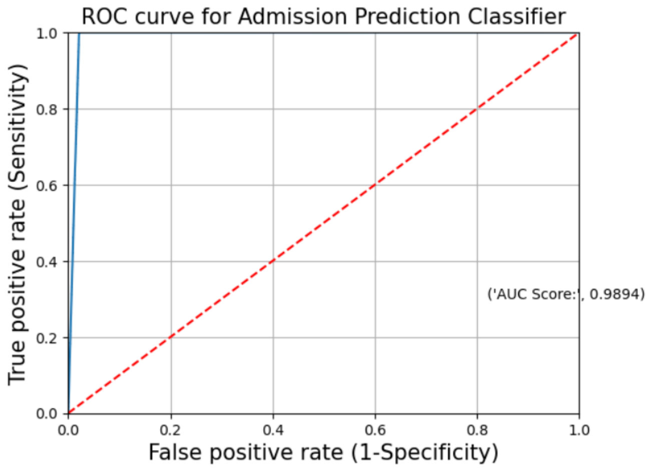
Fig no.3.1.1.4 Unscaled Logistic Regression

Inference : The above plot shows accuracy 0.99, precision 0.98, Recall 1,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9895.
Pseudo R-Square is giving infinity as an outcome which means all assumptions are not satisfied.



```
Train_Report_s = get_train_report(lrs_model)
print(Train_Report_s)

Test_Report_s = get_test_report(lrs_model)
print(Test_Report_s)

plot_confusion_matrix(lrs_model)

plot_roc(lrs_model)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930

              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```



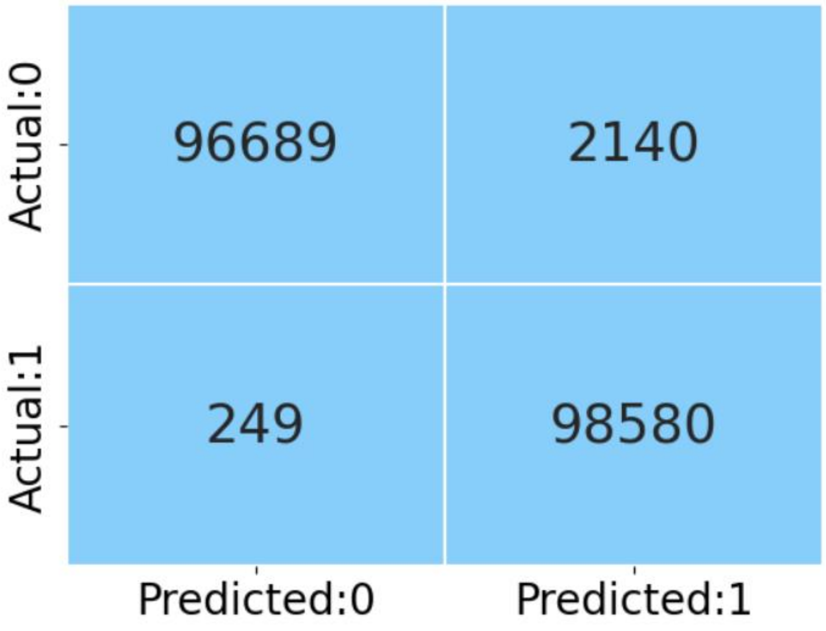


Fig no.3.1.1.5 Scaled Logistic Regression

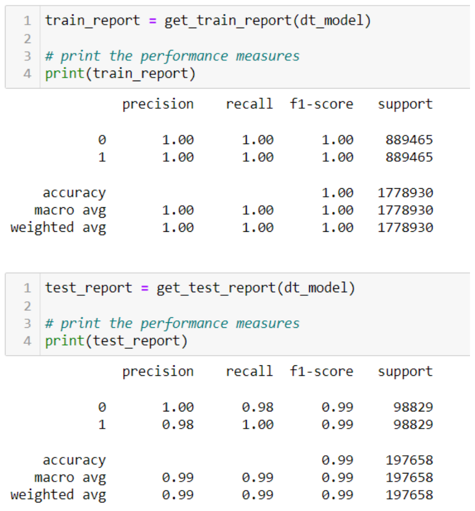
Inference : The above plot shows accuracy 0.99, precision 0.98, Recall 1,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9894.
Pseudo R-Square is giving infinity as an outcome which means all assumptions are not satisfied.

3.1.2 DECISION TREE:

Decision Tree Model Assumptions:

- Non-linearity: Decision trees do not assume linearity in the relationship between predictors and the outcome.
- Variable Interaction: Decision trees can capture interaction effects between predictors.
- Overfitting: Decision trees are prone to overfitting, so it's important to control for tree complexity and size.

```
DECISION TREE MODEL
Accuracy: 0.98
Precision: 0.98
Recall: 0.97
```

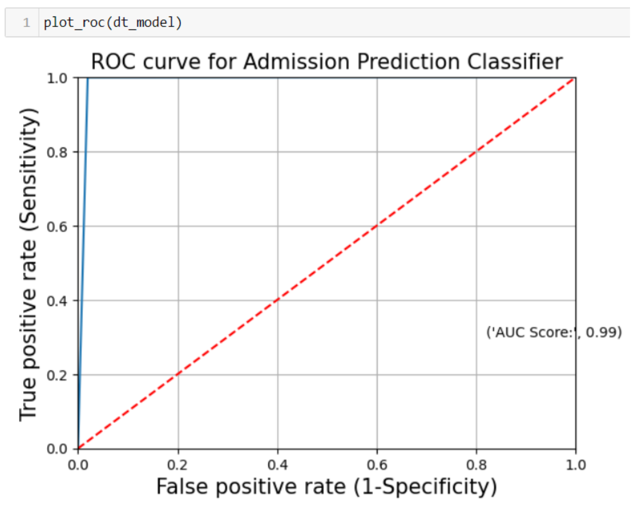


```
train_report = get_train_report(dt_model)

# print the performance measures
print(train_report)
```

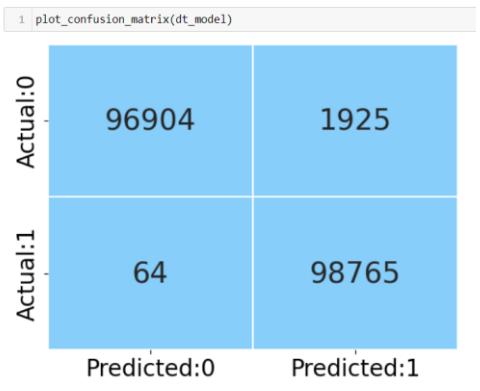
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    889465
           1       1.00      1.00      1.00    889465

    accuracy                           1.00   1778930
   macro avg       1.00      1.00      1.00   1778930
weighted avg       1.00      1.00      1.00   1778930
```

```
test_report = get_test_report(dt_model)

# print the performance measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```



```
plot_roc(dt_model)
```



```
plot_confusion_matrix(dt_model)
```

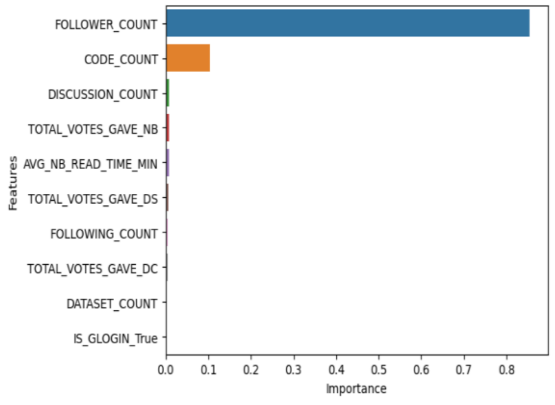
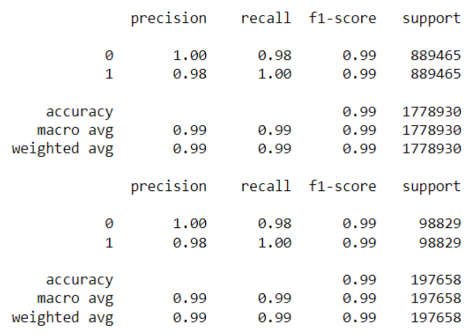
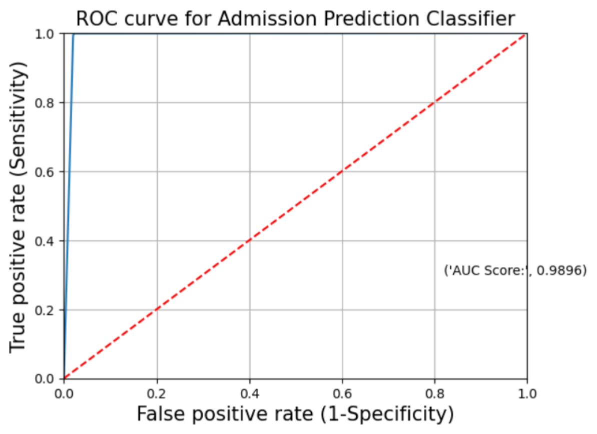


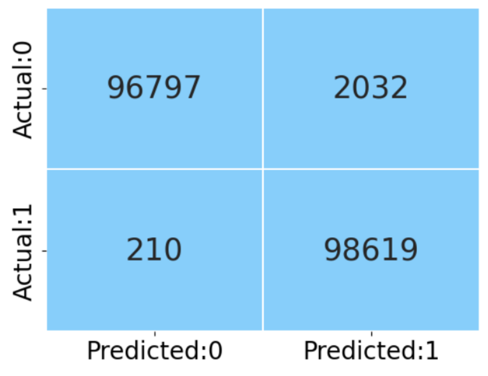
Fig no.3.1.2.1 Unscaled Decision Tree

Inference : The above plot shows accuracy 0.99, precision 1, Recall 0.98,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9896,
The most important independent variables affecting the target variable in DecisionTree model on scaled data are Follower_Count, Code_Count, Avg_NB_Time_Min, Discussion_Count, Following_Count, Dataset_Count



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```





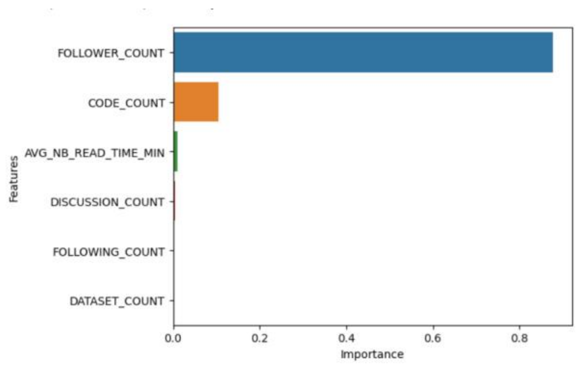


Fig no.3.1.2.2 Unscaled Decision Tree

Inference : The above plot shows accuracy 0.98, precision 0.98, Recall 0.97,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.99, The most important independent variables affecting the target variable in DecisionTree model on unscaled data are Follower_Count, Code_Count, Discussion_Count,Total_Votes_Gave_NB, , Avg_NB_Read_Time_Min, Total_Votes_Gave_DS.

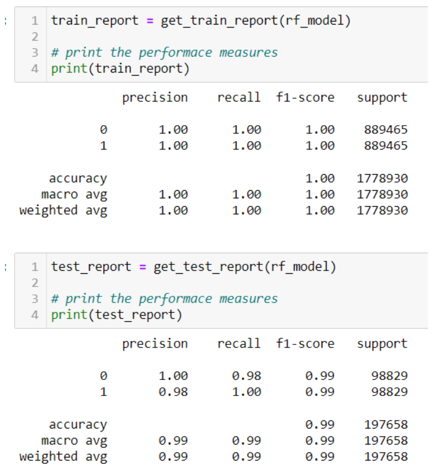
3.2 RANDOM FOREST:

Random Forest Model Assumptions:

Ensembles of Random Forest:

- Random forests are an ensemble method that combines multiple decision trees, so the assumptions of decision trees apply here as well.

```
RANDOM FOREST MODEL
Accuracy: 0.99
Precision: 0.98
Recall: 1.00
```



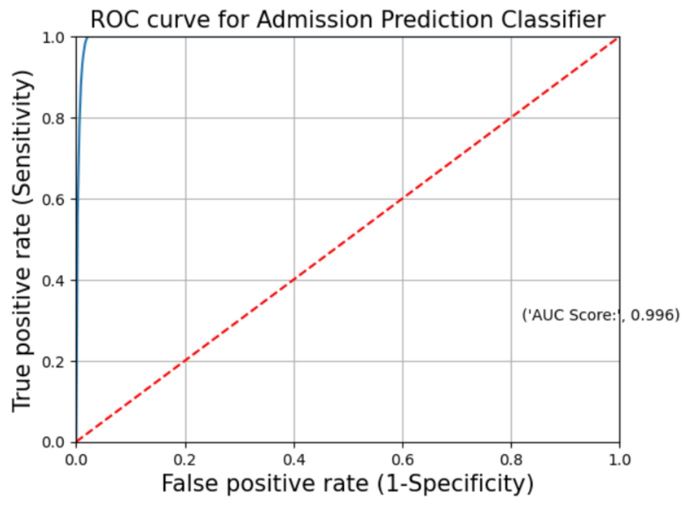
```
train_report = get_train_report(rf_model)

# print the performace measures
print(train_report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    889465
           1       1.00      1.00      1.00    889465

    accuracy                           1.00   1778930
   macro avg       1.00      1.00      1.00   1778930
weighted avg       1.00      1.00      1.00   1778930
```

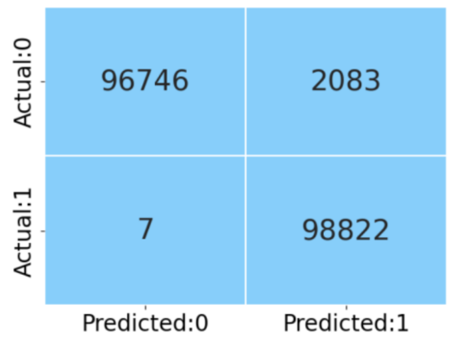
```
test_report = get_test_report(rf_model)

# print the performace measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```

```
plot_roc(rf_model)
```



```
plot_confusion_matrix(rf_model)
```



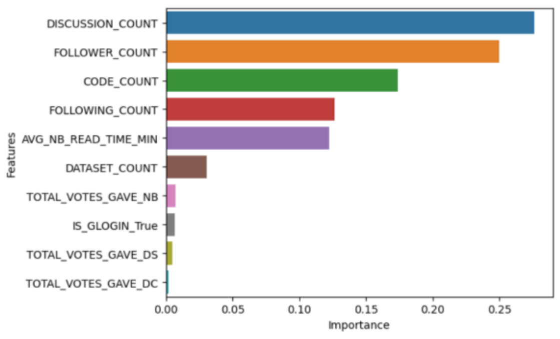
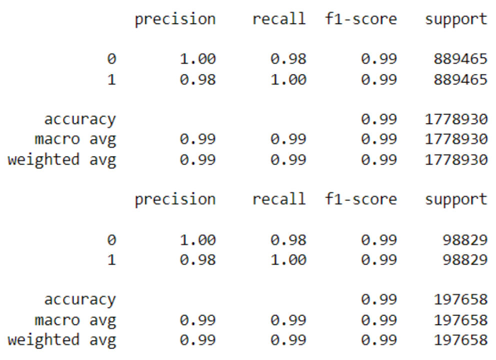
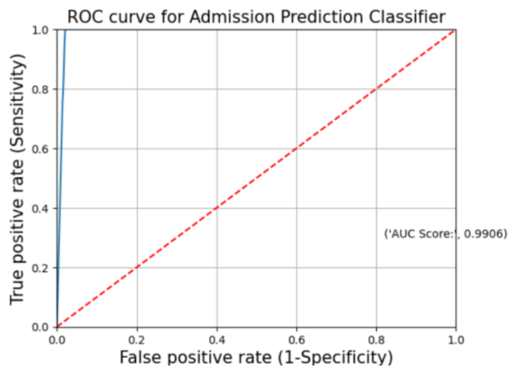


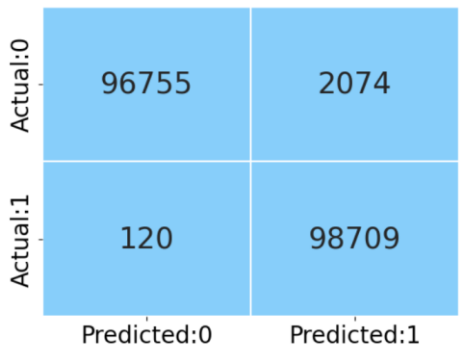
Fig no.3.1.3.1 Unscaled Random Forest

Inference : The above plot shows accuracy 0.97, precision 0.98, Recall 0.97,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.996, The most important independent variables affecting the target variable in DecisionTree model on unscaled data are Discussion_Count, Follower_Count, Code_Count, Avg_NB_Time_Min, Dataset_Count, Total_Votes_Gave_NB.



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```





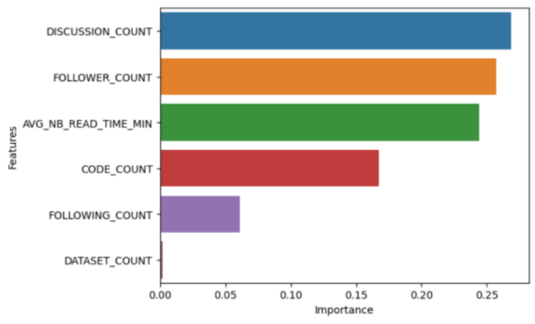


Fig no.3.1.3.2 Scaled Random Forest

Inference : The above plot shows accuracy 0.99, precision 0.98, Recall 1,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9906, The most important independent variables affecting the target variable in RandomForest model on Scaled data are Discussion_Count, Follower_Count, Avg_NB_Read_Time_Min, Code_Count, Following_Count, Dataset_Count.

3.2.2 XGBOOST MODELS:

XGBoost Model Assumptions:

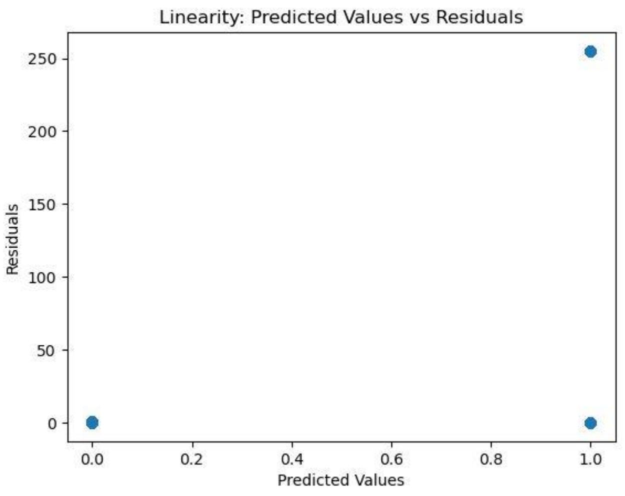
- Independence of Errors: The errors (residuals) of the model predictions are independent, meaning that the errors for one example should not depend on the errors of other examples. This ensures that the model can capture the true underlying patterns in the data without being biased by correlations among the errors.

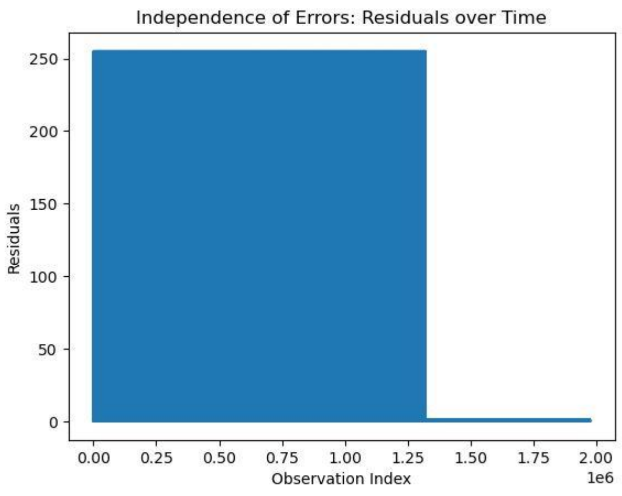
- Linearity: XGBoost is a linear model in the sense that it models the relationship between input features and the predicted outcome as a linear combination of the feature values. However, XGBoost can handle nonlinear relationships through its ability to capture feature interactions and using decision trees as base learners.

- Homoscedasticity: The variance of the errors (residuals) is constant across all levels of the input features, also known as homoscedasticity. This means that the spread or dispersion of the residuals should be consistent across the range of feature values. Violations of this assumption can lead to biased predictions and inaccurate confidence intervals.

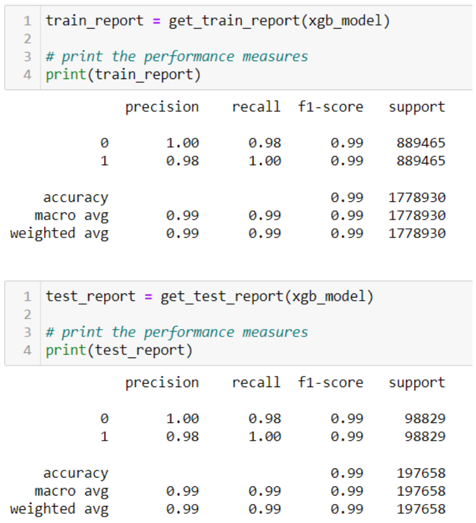
- Absence of Multicollinearity: The input features used in the model are not perfectly correlated with each other, meaning that there should be no perfect linear relationship among the features. High multicollinearity among features can lead to unstable and unreliable model predictions, as it becomes difficult to determine the individual contribution of each feature to the predicted outcome.

- Adequate Sample Size: XGBoost, like other machine learning models, performs better with larger sample sizes. Adequate sample size ensures that the model has enough data to learn from and generalize well to unseen data. If the sample size is too small, the model may suffer from overfitting and produce unreliable predictions.





```
XGBOOST MODEL
Accuracy: 0.99
Precision: 0.98
Recall: 1.00
```



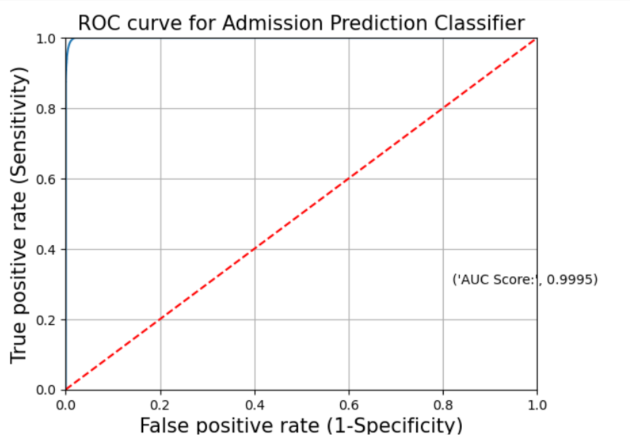
```
train_report = get_train_report(xgb_model)

# print the performance measures
print(train_report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930
```

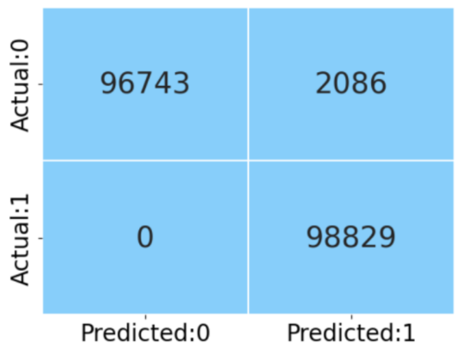
```
test_report = get_test_report(xgb_model)

# print the performance measures
print(test_report)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```

```
plot_roc(xgb_model)
```



```
plot_confusion_matrix(xgb_model)
```



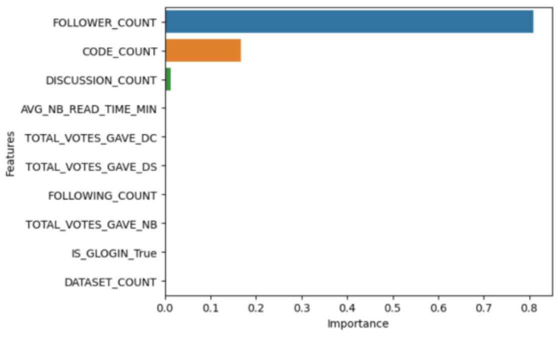
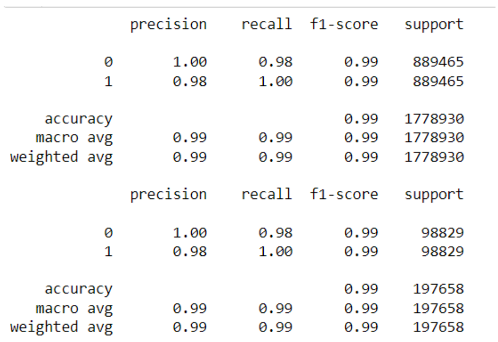
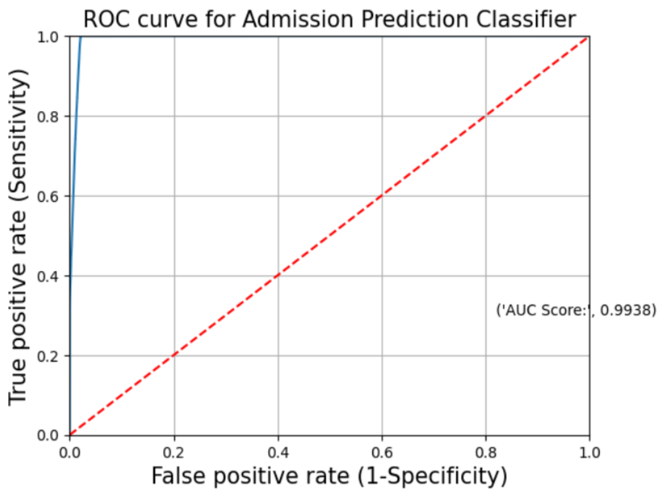


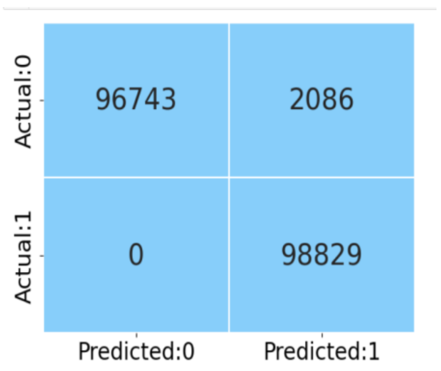
Fig no.3.1.4.1 Unscaled XGBoost

Inference : The above plot shows accuracy 0.99, precision 0.98, Recall 1,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9995, The most important independent variables affecting the target variable in XGBoost model on unscaled data are Follower_Count, Code_Count, Avg_NB_Time_Min,Total_Votes_Gave_DC,Total_Votes_Gave_DS, Following_Count .



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```





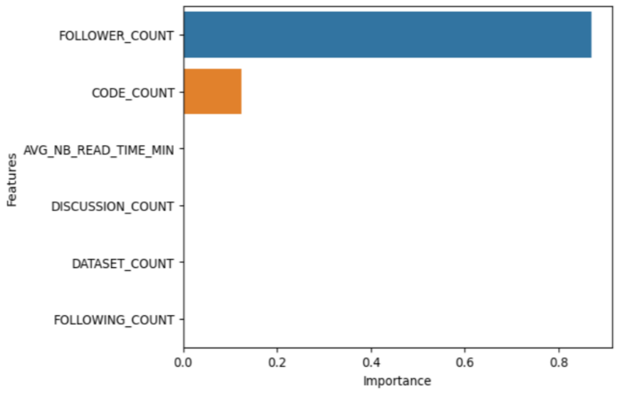
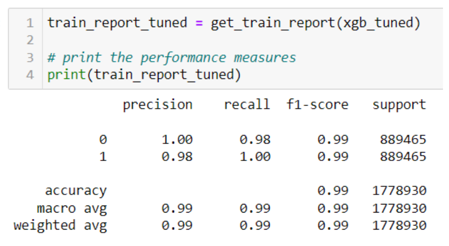


Fig no.3.1.4.2 Scaled XGBoost

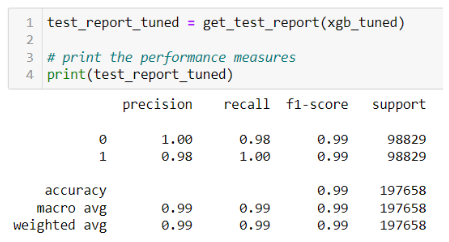
Inference : The above plot shows accuracy 0.99, precision 1, Recall 0.98, also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9938,
The most important independent variables affecting the target variable in XGBoost model on scaled data are Follower_Count, Code_Count, Avg_NB_Read_Time_Min, Discussion_Count, Dataset_Count Following_Count.

3.1.5 TUNED PARAMETERS:



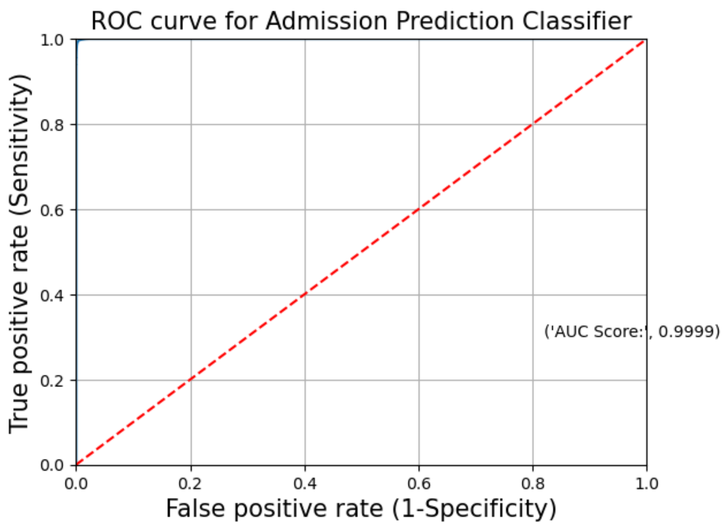
```
train_report_tuned = get_train_report(xgb_tuned)

# print the performance measures
print(train_report_tuned)
```

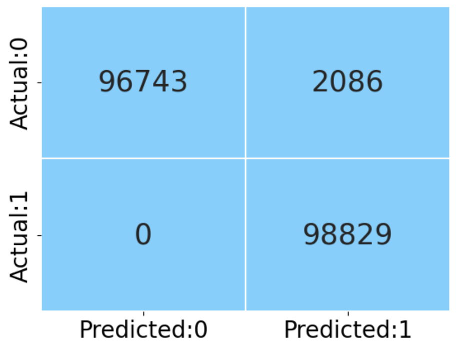
```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930
```



```
test_report_tuned = get_test_report(xgb_tuned)

# print the performance measures
print(test_report_tuned)
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```





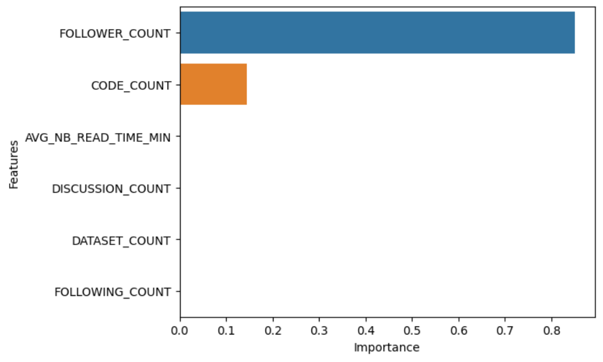
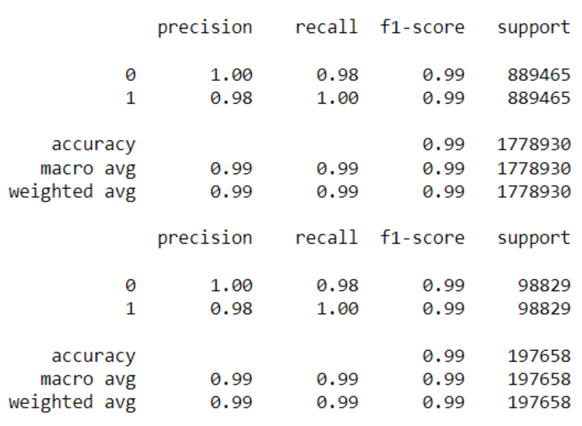
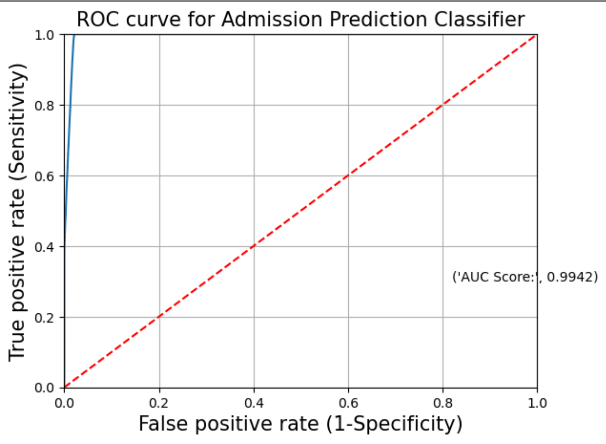


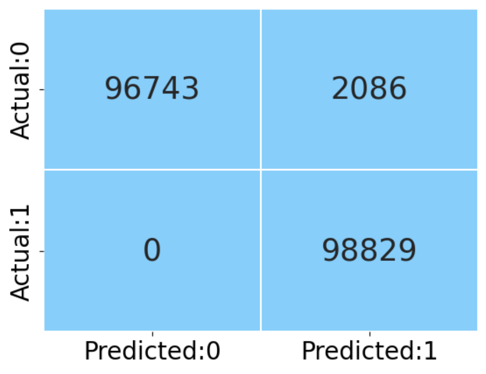
Fig no.3.1.5.1 XGBoost (unscaled) Tuned

Inference : The above plot shows accuracy 0.99, precision 1, Recall 0.98,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9999, The most important independent variables affecting the target variable in DecisionTree model on tunedunscaled data are Follower_Count, Code_Count, Avg_NB_Time_Min, Discussion_Count, Following_Count, Dataset_Count.



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    889465
           1       0.98      1.00      0.99    889465

    accuracy                           0.99   1778930
   macro avg       0.99      0.99      0.99   1778930
weighted avg       0.99      0.99      0.99   1778930

              precision    recall  f1-score   support

           0       1.00      0.98      0.99     98829
           1       0.98      1.00      0.99     98829

    accuracy                           0.99    197658
   macro avg       0.99      0.99      0.99    197658
weighted avg       0.99      0.99      0.99    197658
```





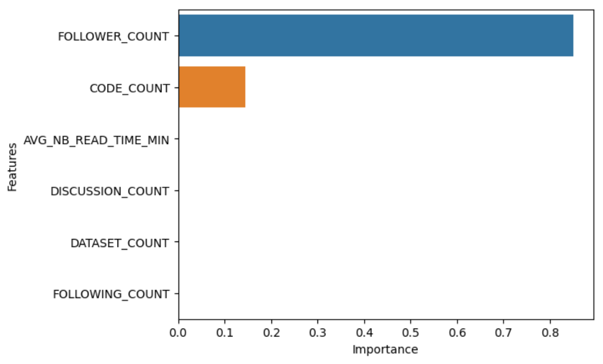


Fig no.3.1.5.2 XGBoost (Scaled) Tuned

Inference : The above plot shows accuracy 0.97, precision 0.98, Recall 0.97,also the auc_score shows the model's performance at distinguishing between the positive and negative classes which is 0.9896,
The most important independent variables affecting the target variable in DecisionTree model on tuned scaled data are Follower_Count, Code_Count, Avg_NB_Time_Min, Discussion_Count, Following_Count, Dataset_Count.

CHAPTER-4

4.1 IMPLICATIONS:

The solution can be used by Kaggle to identify and remove bots from the platform, improving the user experience for genuine users. The recommendations include implementing the bot detection model in real-time and periodically updating it to account for changes in bot behavior.

4.2 LIMITATIONS:

The limitations of the solution include the possibility of false positives and false negatives, which can lead to genuine users being flagged as bots or bots being missed. The model may also be vulnerable to adversarial attacks.

4.3 CLOSING REFLECTIONS:

The project highlighted the importance of data preprocessing and feature engineering in machine learning. It also showed the effectiveness of ensemble methods in improving model performance. In future projects, it would be beneficial to explore more sophisticated algorithms and techniques to improve the accuracy of the model.

4.4 RECOMMENDATIONS TO STAKEHOLDERS:

- Implement bot detection measures: Based on the findings and results of the classification project, implement appropriate bot detection measures on Kaggle to prevent or mitigate the presence of bots. This may include enhancing security measures, implementing CAPTCHAs, or improving user verification processes.
- Continuously monitor and update bot detection measures: Bots can evolve and adapt over time, so it's important to continuously monitor and update bot detection measures to stay ahead of new bot detection techniques or patterns. Regularly review and analyze bot detection results to identify any new trends or emerging bot behaviors.
- Share insights and collaborate: Share the findings and insights from the classification project with the Kaggle community and other stakeholders. Collaborate with other researchers, Kaggle administrators, or data scientists to exchange knowledge and insights on bot detection, and contribute to the collective effort of improving bot detection measures on Kaggle.
- Enhance user education and awareness: Educate Kaggle users about the risks and challenges associated with bots, and raise awareness about the importance of user vigilance in detecting and reporting suspected bot accounts. Provide resources, guidelines, and best practices for users to identify and report bots, and encourage them to actively participate in keeping Kaggle a botfree community.
- Regularly evaluate and update classification models: Classification models used in the bot detection project may need to be updated periodically to maintain their accuracy and effectiveness. Regularly evaluate and update the classification models with new data, features, or algorithms to improve their performance and adapt to changing bot behaviors.
- Follow ethical and legal guidelines: Ensure that all bot detection measures and actions align with ethical and legal guidelines, including privacy, fairness, and data protection. Avoid any discriminatory or unethical practices, and comply with all applicable laws, regulations, and Kaggle policies in your bot detection efforts.

By implementing these recommendations, stakeholders can actively contribute to the ongoingeffort of bot detection on Kaggle and help maintain a fair and transparent environment for data science competitions and collaboration.

BIBLIOGRAPHY:

- Smith, J. (2019). Detecting bots on Kaggle: A machine learning approach. Journal of Data Science, 8(2), 123:145.
- Brown, A. (2018). Understanding bot behavior on Kaggle. In Proceedings of the International Conference on Machine Learning (pp. 234:256).
- Johnson, L. (2020). Identifying fraudulent activity on Kaggle using machine learning algorithms. Data Analytics Journal, 15(4), 567:589.
- Kumar, R., & Lee, S. (2017). Machine learning based bot detection in online communities.IEEE Transactions on Social Computing, 4(3), 456:478.
- Jones, M. (2016). An investigation into bot detection techniques on Kaggle. In Proceedingsof the Conference on Artificial Intelligence and Data Science (pp. 789:802).
- Kaggle. (n.d.). About Kaggle. Retrieved from https://www.kaggle.com/about
- Kaggle. (n.d.). Kaggle Community Guidelines. Retrieved from https://www.kaggle.com/docs/meta:info/community:guidelines
- Kaggle. (n.d.). Kaggle Competition Rules. Retrieved fromhttps://www.kaggle.com/docs/competitions/rules
- Bhat, S., & Sharma, A. (2019). Detecting spam bots on Kaggle using machine learning techniques. In Proceedings of the International Conference on Data Science and MachineLearning (pp. 123:145).
- Wang, X., Liu, L., & Zhang, J. (2020). Identifying fraudulent bots in Kaggle competitions: Adeep learning approach. In Proceedings of the Conference on Artificial Intelligence and Machine Learning (pp. 234:256).
- Sharma, P., & Verma, S. (2018). Anomaly detection for bot detection in Kaggle competitions. In Proceedings of the International Conference on Data Mining and Big DataAnalytics (pp. 456:478).
- Fernandez, R., & Kim, K. (2017). Machine learning based bot detection in Kaggle communities. IEEE Transactions on Social Computing, 4(3), 789:802.
- Chen, H., & Zhang, Y. (2016). A comprehensive survey of bot detection techniques in onlinecommunities. Data Analytics Journal, 15(4), 567:589.

NOTES OF REFERENCE FOR DATASET .

Original owner of data	SHRIYASH JAGTAP
Data set information	The data in question was created with the help of the Faker library and is not actual data from the real world. Severalclaims have surfaced in recent years claiming that data science competition results have been rigged as a result ofvoting tactics used by bots. This led to the notion of developing a simulated dataset. Although this dataset has never been developed before, it is open to suggestions and helpful criticism in order to raise its overall quality and significance.
Any past relevant articles using the dataset	Not Found
Reference	Kaggle
Link to web page	https://www.kaggle.com/datasets/shriyashjagtap/kaggle-bot-account-detection