

#### SIES (NERUL) COLLEGE OF ARTS, SCIENCE AND COMMERCE

NAAC ACCREDITED ‘A’ GRADE COLLEGE (ISO 9001:2008 CERTIFIED INSTITUTION) NERUL, NAVI MUMBAI – 400706

PROJECT REPORT ON

## Ethereum Transaction Fraud Flagging

SUBMITTED BY

## Ashwin Krishnan

PROJECT GUIDE

## Asst. Professor Shweta K

SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF

**MSc. (COMPUTER SCIENCE)**

SEMESTER – IV, 2022 - 2023

# SIES (NERUL) COLLEGE OF ARTS SCIENCE AND COMMERECE

NAAC ACCERDITED ‘A’ GRADE COLLEGE (ISO 9001:2015 CERTIFIED INSTITUTION) NERUL, NAVI MUMBAI - 400706

THIS IS TO CERTIFY THAT THE PROJECT TITLED

#### Ethereum Transaction Fraud Flagging

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

IS UNDERTAKEN BY

#### Ashwin Krishnan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Seat No: 07

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In partial fulfilment of MSc - IT / CS Degree (Semester IV)

in the academic year **2022-2023** and has not been submitted for any other examination and does not form part of any other course undergone by the candidate. It is further certified that he/she has completed all the required phases of the project.

\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Project Guide | External Examiner |
| Head of Department | Principal |

**ACKNOWLEDGMENT**

I extend my heartfelt gratitude and thanks to Asst. Professor Shweta K ma’am for providing me excellent guidance to work on this project and for their understanding and assistance by providing all the necessary information needed for my project topic.

I would also like to acknowledge all the staffs for providing a helping hand to us in times of queries & problems. The project is a result of the efforts of all the peoples who are associated with the project directly or indirectly, who helped us to work to complete the project within the specified time frame. They motivated me in the project and gave feedback on it to improve my adroitness.

Thanks to all my teachers, who were a part of the project in numerous ways and for the help and inspiration they extended to me and for providing the needed motivation.

With all Respects & Gratitude, I would like to thanks to all the people, who have helped for the development of the Project.

**Ashwin Krishnan MSc. Computer Science (Part-II)**

**SIES (Nerul) College of Arts, Science, and Commerce**.

|  |  |
| --- | --- |
| **Table of Content** |  |
| 1) Title | 5 |
| 2) Implementation details | 6 |
| 3) Experimental set up and results | 9 |
| 4) Analysis of the results: | 14 |
| 5) Conclusion | 15 |
| 6) Future enhancement | 15 |
| 7) Program code | 17 |

PROJECT REPORT ON:

# Ethereum Transaction

# Fraud Flagging

INTRODUCTION:

Cryptocurrency is a type of digital currency that operates independently of a central bank or government. It utilizes blockchain technology, which is a decentralized ledger that records transactions across a network of computers. One of the most popular cryptocurrencies is Ethereum, which was launched in 2015 and has since become a major player in the crypto world.

Ethereum is unique because it allows developers to create decentralized applications (Dapps) on its blockchain. These Dapps can be used for a variety of purposes, from creating digital art to conducting financial transactions. Ethereum has its own cryptocurrency called Ether (ETH), which is used to pay for transactions and to incentivize miners to secure the network.

One of the biggest advantages of Ethereum is its ability to execute smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. This eliminates the need for intermediaries, making transactions faster and cheaper.

However, the decentralized nature of Ethereum and other cryptocurrencies also makes them vulnerable to fraud and other illegal activities. This is where fraud analysis comes in. Fraud analysis involves using various tools and techniques to identify and prevent fraudulent transactions.

In the case of Ethereum, fraud analysis can involve monitoring the blockchain for suspicious activity, analyzing transaction patterns, and identifying potential vulnerabilities in smart contracts. By conducting fraud analysis on Ethereum transactions, it is possible to detect and prevent fraud, which helps to maintain the integrity and security of the network.

In conclusion, cryptocurrency and Ethereum offer many exciting possibilities for innovation and financial transactions. However, it is important to remain vigilant and conduct fraud analysis to ensure that the network remains secure and free from illegal activities.

IMPLEMENTATION DETAIL

#### Data Collection:

For fraud flagging a Ethereum transaction dataset was used from Kaggle

#### Data Variables:

#### Index: the index number of a row

#### Address: the address of the Ethereum account

#### FLAG: whether the transaction is fraud or not

#### Avg min between sent tnx: Average time between sent transactions for account in minutes

#### Avg min between received tnx: Average time between received transactions for account in minutes

#### Time Diff between first and\_last (Mins): Time difference between the first and last transaction

#### Sent\_tnx: Total number of sent normal transactions

#### Received\_tnx: Total number of received normal transactions

#### NumberofCreated\_Contracts: Total Number of created contract transactions

#### UniqueReceivedFrom\_Addresses: Total Unique addresses from which account received transaction

#### UniqueSentTo\_Addresses20: Total Unique addresses from which account sent transactions

#### MinValueReceived: Minimum value in Ether ever received

#### MaxValueReceived: Maximum value in Ether ever received

#### AvgValueReceived5Average value in Ether ever received

#### MinValSent: Minimum value of Ether ever sent

#### MaxValSent: Maximum value of Ether ever sent

#### AvgValSent: Average value of Ether ever sent

#### MinValueSentToContract: Minimum value of Ether sent to a contract

#### MaxValueSentToContract: Maximum value of Ether sent to a contract

#### AvgValueSentToContract: Average value of Ether sent to contracts

#### TotalTransactions(IncludingTnxtoCreate\_Contract): Total number of transactions

#### TotalEtherSent:Total Ether sent for account address

#### TotalEtherReceived: Total Ether received for account address

#### TotalEtherSent\_Contracts: Total Ether sent to Contract addresses

#### TotalEtherBalance: Total Ether Balance following enacted transactions

#### TotalERC20Tnxs: Total number of ERC20 token transfer transactions

#### ERC20TotalEther\_Received: Total ERC20 token received transactions in Ether

#### ERC20TotalEther\_Sent: Total ERC20token sent transactions in Ether

#### ERC20TotalEtherSentContract: Total ERC20 token transfer to other contracts in Ether

#### ERC20UniqSent\_Addr: Number of ERC20 token transactions sent to Unique account addresses

#### ERC20UniqRec\_Addr: Number of ERC20 token transactions received from Unique addresses

#### ERC20UniqRecContractAddr: Number of ERC20token transactions received from Unique contract addresses

#### ERC20AvgTimeBetweenSent\_Tnx: Average time between ERC20 token sent transactions in minutes

#### ERC20AvgTimeBetweenRec\_Tnx: Average time between ERC20 token received transactions in minutes

#### ERC20AvgTimeBetweenContract\_Tnx: Average time ERC20 token between sent token transactions

#### ERC20MinVal\_Rec: Minimum value in Ether received from ERC20 token transactions for account

#### ERC20MaxVal\_Rec: Maximum value in Ether received from ERC20 token transactions for account

#### ERC20AvgVal\_Rec: Average value in Ether received from ERC20 token transactions for account

#### ERC20MinVal\_Sent: Minimum value in Ether sent from ERC20 token transactions for account

#### ERC20MaxVal\_Sent: Maximum value in Ether sent from ERC20 token transactions for account

#### ERC20AvgVal\_Sent: Average value in Ether sent from ERC20 token transactions for account

#### ERC20UniqSentTokenName: Number of Unique ERC20 tokens transferred

#### RC20UniqRecTokenName: Number of Unique ERC20 tokens received

#### ERC20MostSentTokenType: Most sent token for account via ERC20 transaction

#### 46. ERC20MostRecTokenType: Most received token for account via ERC20 transaction

#### Algorithm Selection:

XGBoost (Extreme Gradient Boosting) and Logistic Regression are both machine learning algorithms used for classification tasks, such as fraud detection. While logistic regression is a traditional algorithm that has been widely used for decades, XGBoost is a relatively new algorithm that has gained popularity due to its high accuracy and speed in handling large datasets.

Logistic regression is a linear model that is used to predict the probability of a binary outcome (e.g., fraud or non-fraud). It works by fitting a curve to the data that separates the two classes based on a set of input features. The algorithm then uses this curve to predict the probability of fraud for new transactions based on their features.

XGBoost, on the other hand, is an ensemble model that combines multiple decision trees to make predictions. It works by iteratively building decision trees that focus on the areas where the model is performing poorly, allowing it to learn from its mistakes and improve over time. XGBoost is particularly useful for fraud detection because it can handle large datasets with high dimensionality, where traditional models like logistic regression may struggle.

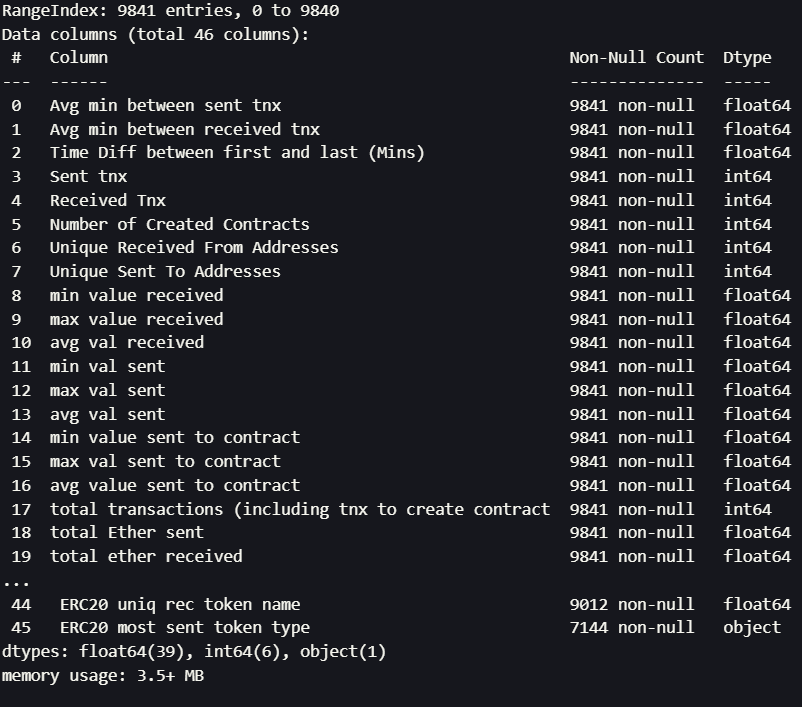
In fraud flagging, both logistic regression and XGBoost can be used to predict the probability of fraud for each transaction, based on features such as transaction amount, location, and previous transaction history. Once the model has made its predictions, a threshold can be set to flag transactions with a high probability of fraud for manual review.

In practice, XGBoost has shown to outperform logistic regression in many cases due to its ability to handle complex datasets with high dimensionality. However, logistic regression may still be useful in cases where the dataset is relatively small and the features are linearly separable.

Logistic regression and XGBoost are powerful machine learning algorithms that can be used for fraud detection in financial transactions. The choice of algorithm will depend on the specific characteristics of the dataset.

### EXPERIMENTAL SET UP AND RESULTS DATASET

The data set consist of individual transaction level details used to sign a smart contract.



The transaction dataset consists of 9841 rows and 46 columns.

Memory Usage of dataset: 3.5 MB

**Data Cleaning and Preprocessing:**

1. Hypothesis Setup:

H0: A transaction is not fraudulent

H1: A transaction is fraudulent.

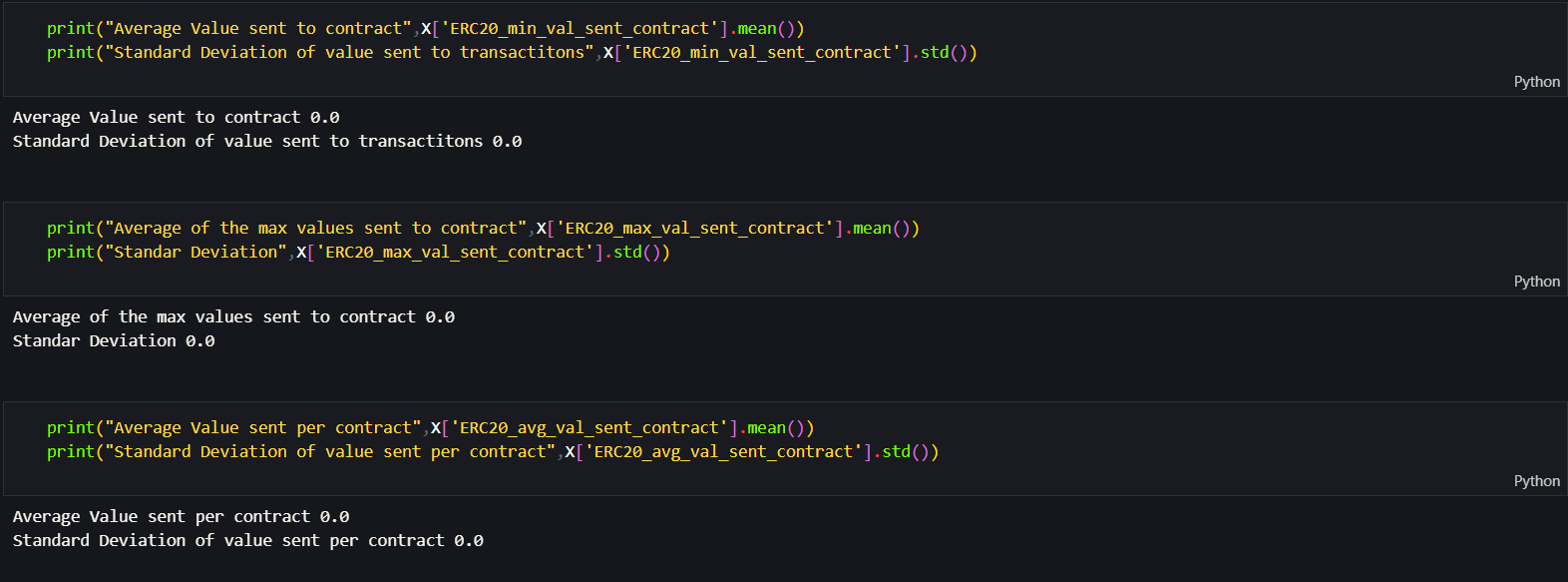
1. Heatmap & Multicollinearity Check:

The correlation yields 'NULL' values for certain values. This is due to division by zero.

The mean & standard deviation of these columns yield zero. This is due to all the values being null.

Since there is no data, these variables will not be included in the model.





1. Mutual Information Index:

The equation for mutual information (MI) is:

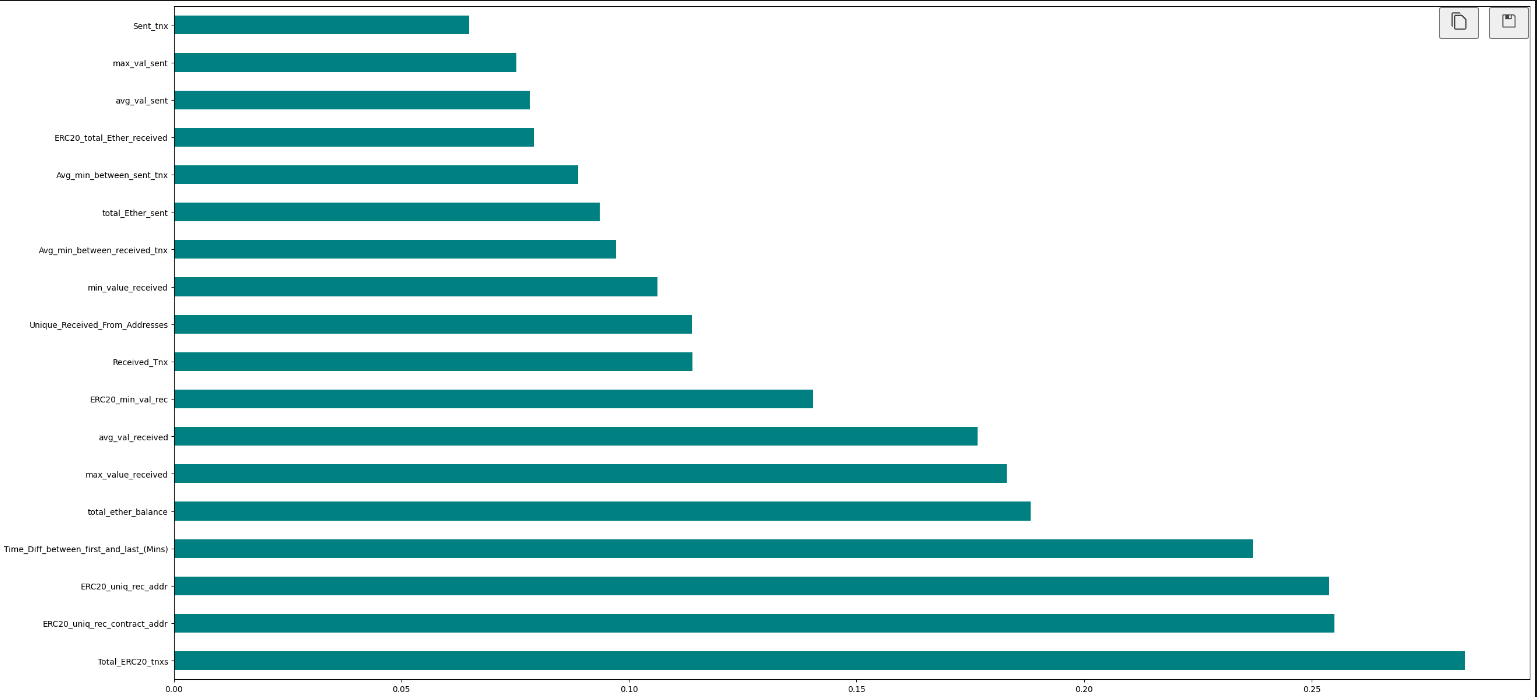
MI(X, Y) = ∑x∈X ∑y∈Y p(x,y) log [p(x,y) / (p(x) \* p(y))]

where X and Y are the two random variables being compared, p(x,y) is the joint probability distribution of X and Y, p(x) and p(y) are the marginal probability distributions of X and Y, respectively.

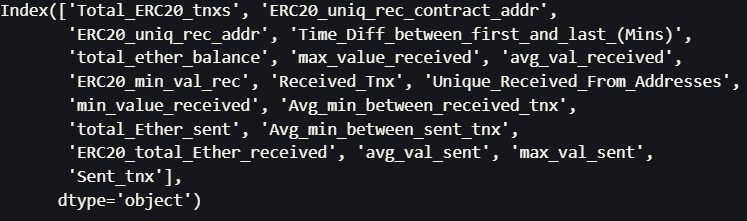
The mutual information between two variables measures the amount of information one variable provides about the other variable. It is a non-negative value, where a higher value indicates a greater mutual dependence between the variables.

The equation can be interpreted as follows: the joint probability of observing both X and Y together is compared to the product of the marginal probabilities of observing X and Y separately. The ratio of these probabilities is then multiplied by its logarithm to obtain a scalar value that measures the degree of mutual dependence between the two variables.

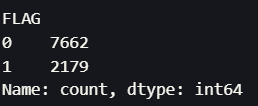
In feature selection tasks, the mutual information index is used to compare the relationship between each feature and the target variable. Features with higher mutual information scores are considered more informative for predicting the target variable, and are therefore selected for further analysis. Features with lower mutual information scores may be discarded as they may not provide much information about the target variable.



Using the 18 most important features derived using mutual information index.



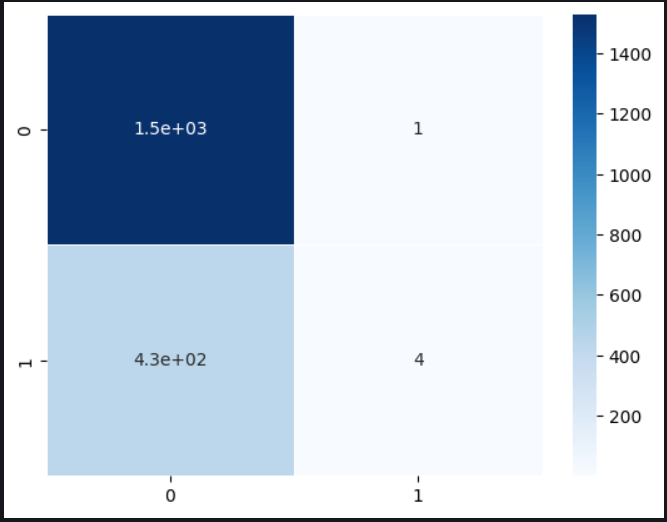
1. Random Oversampling:



Since there is a large imbalance between the classes {1: Fraudulent Transaction, 0: Non-Fraudulent Transaction} Random Oversampling is used to balance the classes.

**Model Selection & Analysis:**

* **Logistic Regression Prior Oversampling:**

****

1. The confusion matrix shows a clear sign of overfitting.

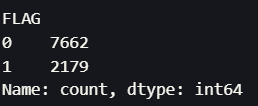
2. This is due to the highly imbalanced target variable.

3. The "0" in the target variable defines non-Fraudulent transactions. The "1" denotes

fraudulent transactions.

4. Due to the imbalance the fraudulent transactions are being classified into non fraudulent.

5. The Type II error is high (false-negative), meaning "1" is classified as "0".





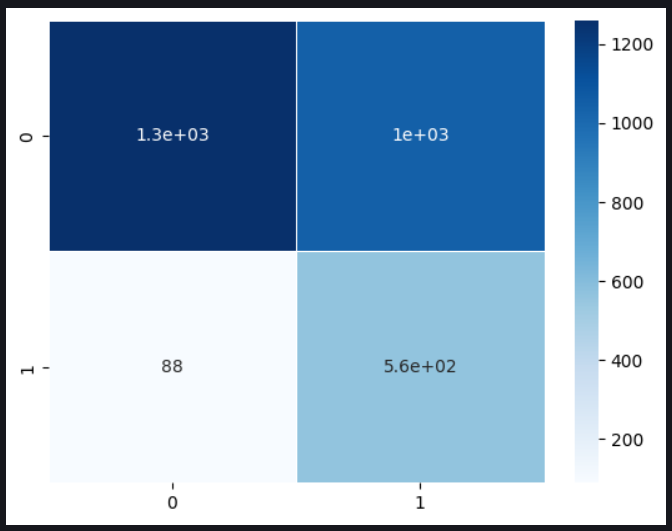
1. The precision of the model is high but the recall is really low, this displays the inability of the

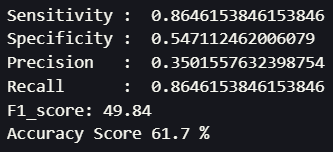
model to classify fraudulent transactions.

2. For this case we want precision & recall to both be high.

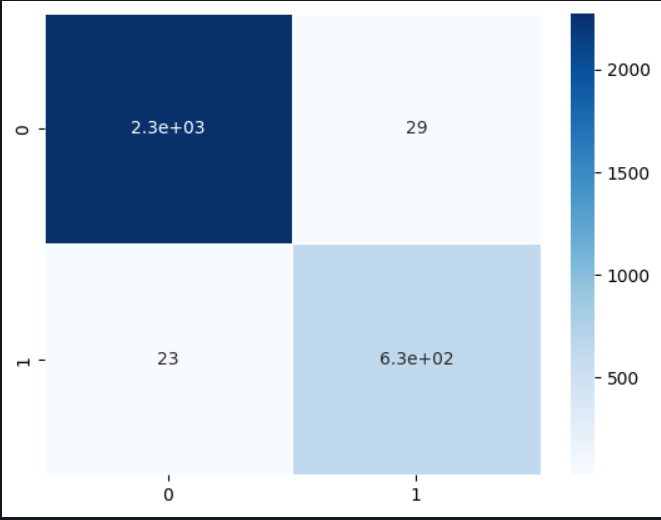
3. The Model is not good at classifying this is backed up by the F1 score

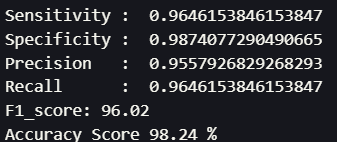
* Logistic Regression Post Over Sampling:





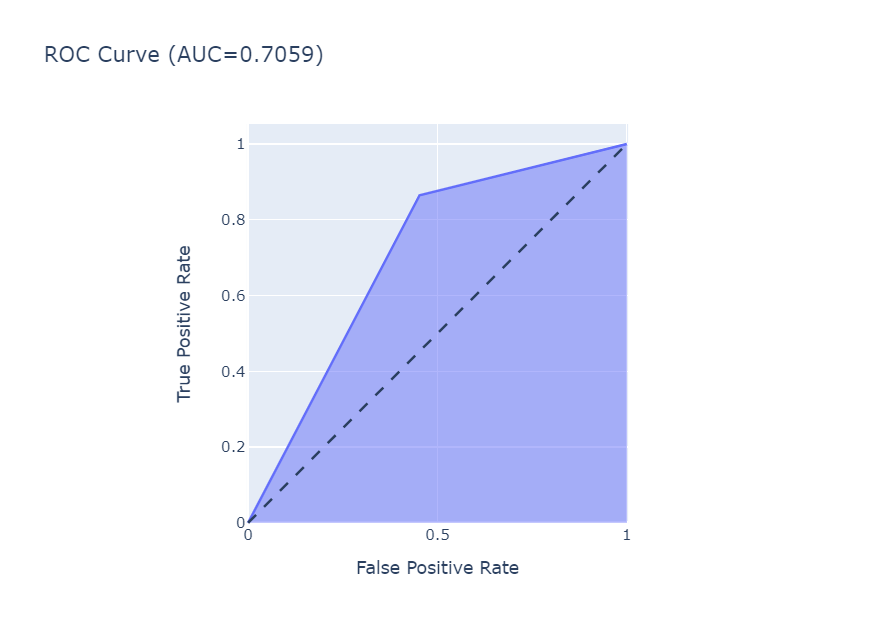
* XGBoost Algorithm:



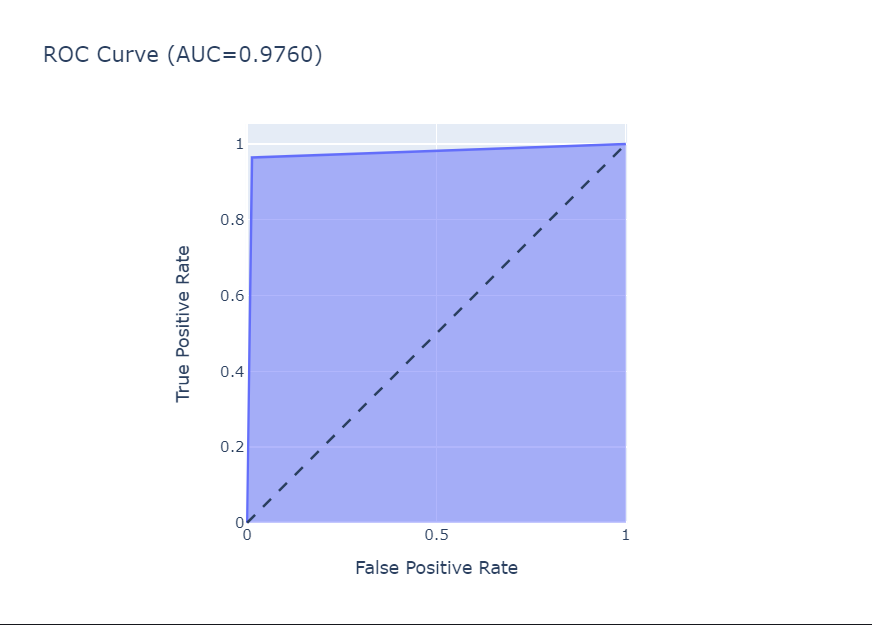


ROC & AUC Analysis:

1. Logistic Regression



1. XGBoost



CICD (Continuous Integration & Continuous Deployment):

Fast API that uses a post endpoint to re-train a model is a powerful tool that allows for real-time updates and improvements to machine learning models. By utilizing the post endpoint, users can provide trigger a re-training process, which can improve the accuracy and relevance of the model's predictions. Once The Model is above quality threshold is then used to classify fraudulent transactions.

This type of API is especially useful in applications where the underlying data is constantly changing or where new data becomes available at regular intervals. For example, in a fraud detection system, the model can be retrained on a regular basis to improve its accuracy in detecting new types of fraudulent transactions.

To use this API, users can simply send a post request with the updated training data to the API endpoint. The API will then process the data and use it to retrain the model. Once the retraining process is complete, the updated model can be used to make predictions on new data.

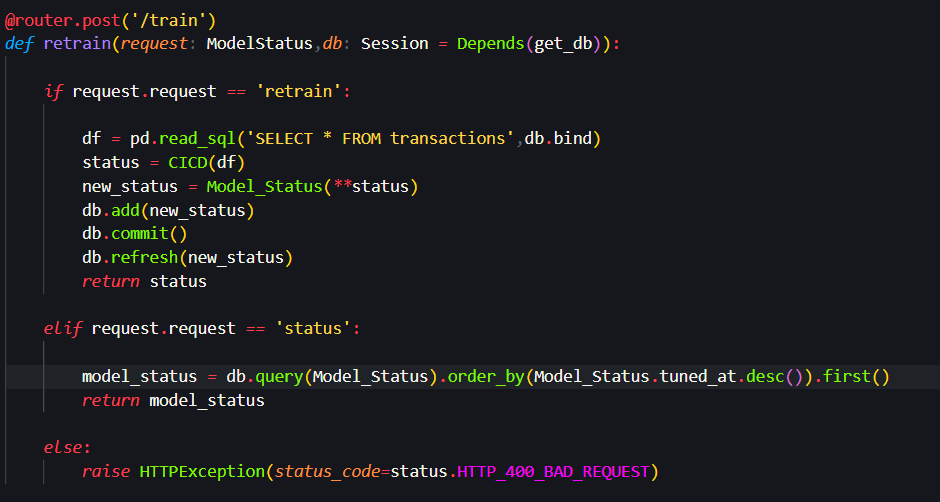
Overall, a fast API with a post endpoint for retraining machine learning models is a powerful tool that can improve the accuracy and relevance of predictions in real-time.

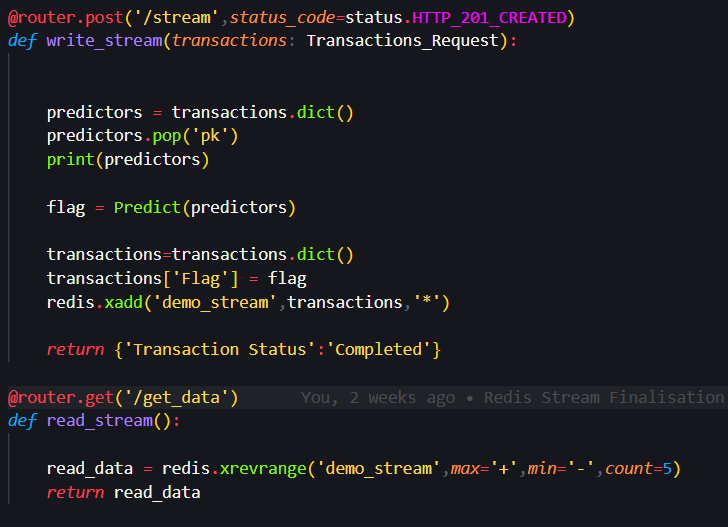
Redis Streams

Redis Streams is a powerful data structure that can be used for storing and processing transaction data in real-time. Redis Streams is essentially a log-like data structure that allows you to store, query, and process data in a highly scalable and efficient manner.

One of the main benefits of using Redis Streams for storing transaction data is its ability to process data in real-time. As new transactions are added to the stream, they can be immediately processed and analyzed using various Redis commands and functions. For example, you can use the XREAD command to read messages from the stream and process them in real-time.

Another benefit of using Redis Streams for storing transaction data is its scalability. Redis Streams can handle high volumes of data and can be easily distributed across multiple Redis nodes to ensure high availability and fault tolerance.





CONCLUSION:

1. The logistic regression model is unable to capture the separability in the transactions.
2. XGBoost model significantly outperforms the logit model hence we will use XGBoost as the base model for classifying an Ethereum transaction as fraud.
3. Serving the Fraud Classification model using a continuous integration pipeline helps reduce model downtime during retraining.

### FUTURE ENHANCEMENT:

* Distributed Node setup for Redis DB.
* Docker Swarm Setup for API along with NGINX reverse proxy & SSL.