Predictive Modelling for Fraud Detection on MegaETH Testnet Using Logistic Regression

Table of Contents

1.0 Introduction

Blockchain has been in the spotlight of digital finance where decentralized systems have been established that drive transactions around the globe. But its development has also brought with it rising cybercrimes such as phishing, transaction spamming and other malevolent processes (Kiayias & Panagiotakos, 2019).

High throughput networks such as MegaETH, which churn more than 100,000 transactions per second with sub-millisecond latency by specialized nodes and implementing a hyper-optimized EVM world (MegaETH, 2025), certainly enhance speed and efficiency. However, this also opens the gates to higher and faster fraud practices, including transaction spamming, MEV attacks, and other anomalies. Although the blockchain is transparent, few studies have been conducted on predictive detection of frauds in such a high-velocity atmosphere. This paper suggests a predictive model based on logistic regression to detect suspicious transactions in the MegaETH testnet to understand threats of this real-time blockchain.

2.0 The Methodology

This section will cover the Python libraries used in this study as well as how the data are obtained, scrubbed, and explored.

2.1 Python libraries

2.1.1 NumPy

NumPy is the base Python library that has array operations and other mathematical tools and functions. NumPy will be highly useful in helping to preprocess MegaETH testnet data, work with large transaction data sets and calculate the frequencies of transactions, and other block statistics. This is an extremely fast and versatile algorithm, and it is therefore a good choice to build a logistic regression-based fraud detection model (Tom,2024).

2.1.2 Pandas

Pandas is a library in python which is used to manipulate and analyse data using data structures like Series and DataFrame. Where NumPy makes efficient numerical computations with n-dimensional arrays, pandas add the capability to deal with labelled, tabular data making it easier to study and preprocess complex data. There are functions, such as .info(), .head(), and

.describe() which can be used to understand the structure of the data, and isna(), fillna(), and dropna() allowing to manage missing values. With regards to MegaETH testnet data, the use of pandas to filter the columns, convert the timestamps, merge the records, and arrange the transaction data will help to easily prepare the data to move into logistic regression modeling, which needs the computing power of NumPy (GeeksforGeeks, 2025).

2.1.3 Visualisation using matplotlib

Matplotlib is a cross-platform Python library which allows generating line, bar, step and scatter plots, etc. This will be applied in this study to graphically analyze the data of MegaETH testnet transactions, to find patterns and compare normal and suspicious operations. Additional features such as sub plots, manipulation of the axes, and legends enable the formation of legible, and extractable representations to aid the understanding of the analysis conducted by the logistic regression (Sial & Hafeez, 2021).

2.1.4 Scikit-learn

Scikit-learn is a well-structured Python machine learning package that allows implementing well-known algorithms such as classification, regression and clustering. In this study it will be employed to create a logistic regression model to predict suspect transaction within the ethereum testnet MegaETH. The tool offers comprehensive data preprocessing capabilities as well as the model training, evaluation and performance metrics, which makes it suitable in the development and validation of predictive fraud detecting models efficiently (Hao & Ho,2019).

2.2 Obtain the data

2.2.1Data Collection

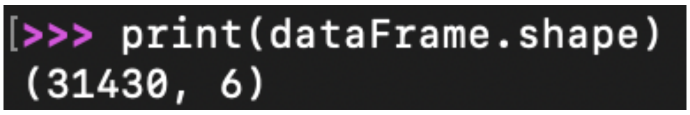
The specific data used in this research were taken up solely via the public MegaETH Testnet API (MegaETH, 2025) that is used to get real-time access to activity inscribed in the blockchain, including the transactions and block information as well as sequencer events. In this source, 200 blocks have been processed, i.e., about 31,430 transactions.

The following variables were extracted and organized for analysis:

- **Sender address** – the originating account of the transaction

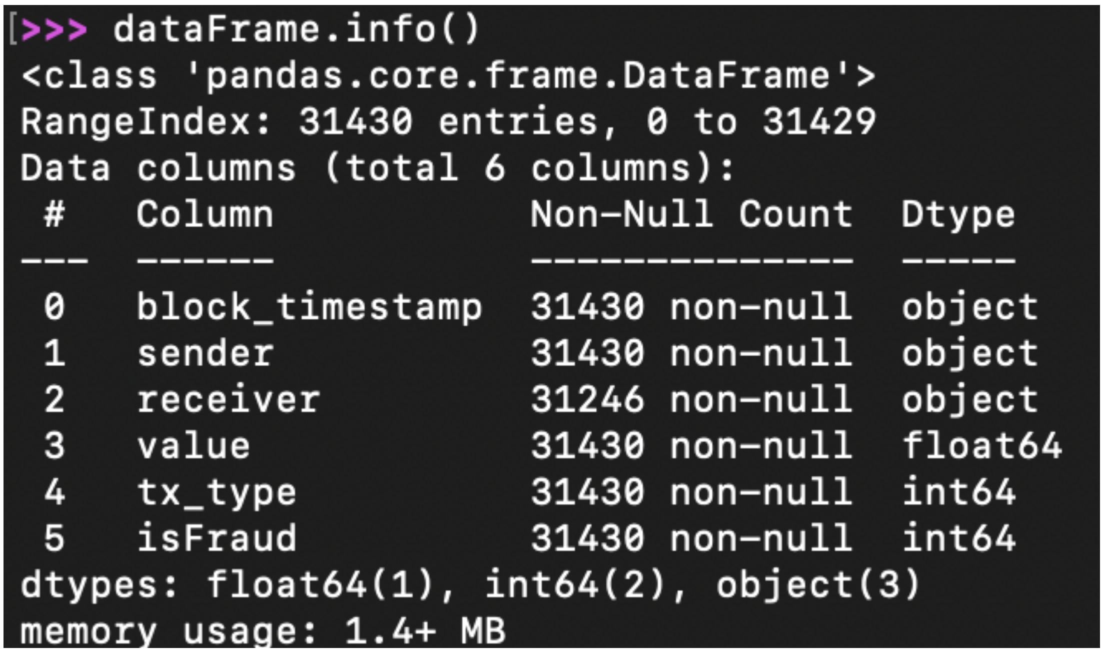
- **Receiver address** – the destination account of the transaction
- **Transaction value** – the amount of ETH (or token equivalent) transferred
- **Transaction type** – categorical identifier of transaction purpose (e.g., contract interaction, transfer)
- **Block timestamp** – timestamp indicating when the block containing the transaction was confirmed

The collected data were stored in a CSV file named transactions_labeled.csv. Inspection of the dataset shows that it contains 31,430 rows and 6 columns, as indicated by the output of dataFrame.shape:



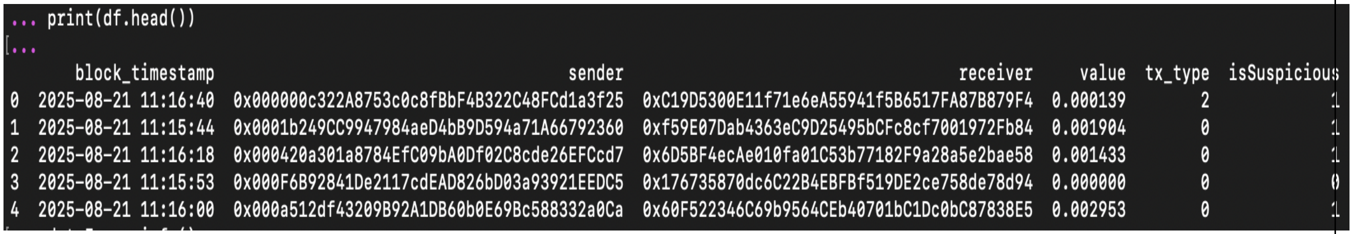
```
>>> print(dataFrame.shape)
(31430, 6)
```

Further overview of the dataset structure, using dataFrame.info(), confirms the completeness and types of the variables:



```
>>> dataFrame.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31430 entries, 0 to 31429
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   block_timestamp  31430 non-null  object
 1   sender           31430 non-null  object
 2   receiver         31246 non-null  object
 3   value            31430 non-null  float64
 4   tx_type          31430 non-null  int64
 5   isFraud          31430 non-null  int64
dtypes: float64(1), int64(2), object(3)
memory usage: 1.4+ MB
```

The first few rows of the dataset dataFrame.head() illustrate the structure of the data:



```
... print(df.head())
...
       block_timestamp                                      sender                                    receiver     value  tx_type  isSuspicious
0  2025-08-21 11:16:40  0x000000c322A8753c0c8fBbF4B322C48FCd1a3f25  0xC19D5300E11f71e6eA55941f5B6517FA87B879F4  0.000139        2             1
1  2025-08-21 11:15:44  0x0001b249CC9947984aeD4bB9D594a71A66792360  0xf59E07Dab4363eC9D25495bCFc8cf7001972Fb84  0.001904        0             1
2  2025-08-21 11:16:18  0x000420a301a8784EfC09bA0Df02C8cde26EFCcd7  0x6D5BF4ecAe010fa01C53b77182F9a28a5e2bae58  0.001433        0             1
3  2025-08-21 11:15:53  0x000F6B92841De2117cdEAD826bD03a93921EEDC5  0x176735870dc6C22B4EBFBf519DE2ce758de78d94  0.000000        0             0
4  2025-08-21 11:16:00  0x000a512df43209B92A1DB60b0E69Bc588332a0Ca  0x60F522346C69b9564CEb40701bC1Dc0bC87838E5  0.002953        0             1
```

2.2.2Suspicious Transaction Labeling

To identify potentially suspicious transactions, **heuristic rules** were applied to the dataset (Luo,2014):

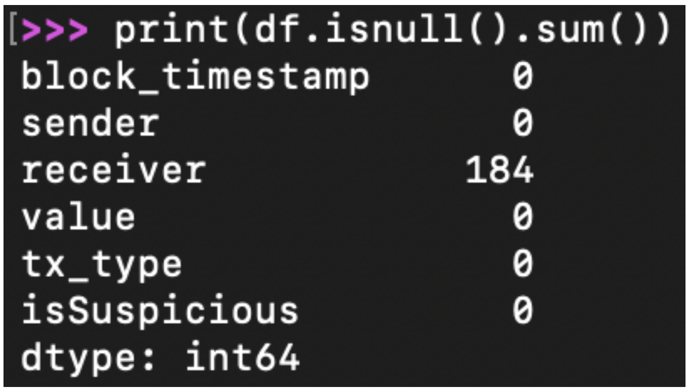
1. **Large-value transactions –** Transactions with values significantly higher than the median (100 * median_value) value were flagged as suspicious. Very small transactions were ignored, as micro-transactions are common on the testnet.
2. High-frequency transactions – Senders that initiated multiple transactions within a short time window (e.g., more than 10 transactions within 1 minute) were flagged as suspicious.

Transactions flagged by either rule were labeled with isSuspicious = 1; others were labeled isSuspicious = 0.

2.3 Scrubbing the data

2.3.1 Missing Values

Special attention were paid to rows that contain missing values and then missing value treatment was applied. The number of missing values in every column was verified by means of df.isnull().sum() that provides the number of missing entries in the column. A value of 0 depicts that there are no missing values, any positive value implies missing data (Analytics Vidhya,2021).

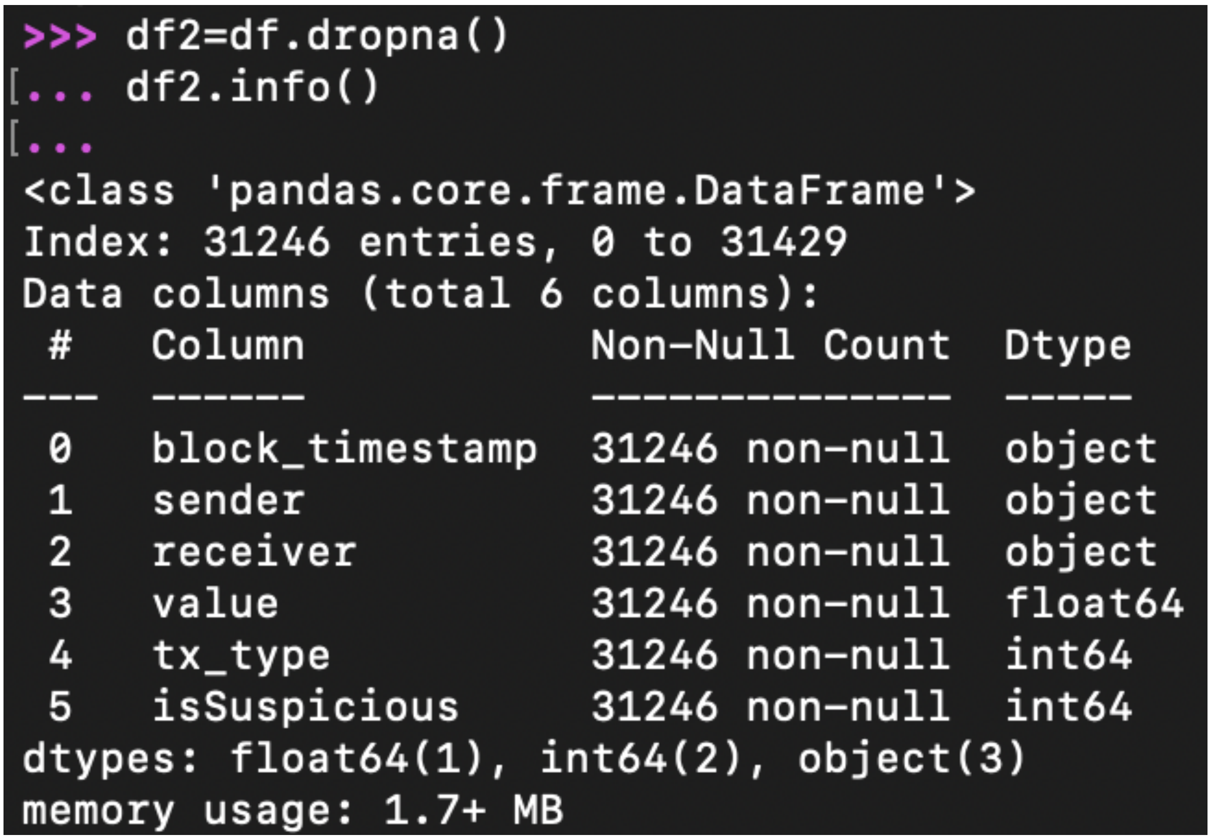


```
[>>> print(df.isnull().sum())
 block_timestamp      0
 sender               0
 receiver           184
 value                0
 tx_type              0
 isSuspicious         0
 dtype: int64
```

In this dataset, missing values are present only in the receiver column, with 184 entries missing. All other variables (block_timestamp, sender, value, tx_type, and isSuspicious) contain complete data.

Since missing values are present in the receiver column, the affected rows were dropped to prevent bias or inaccuracies in the analysis. After removing these rows, the dataset was reduced from 31,430 to 31,246 entries. This constitutes a small proportion of the dataset and is unlikely to cause significant information loss.

The final dataset used for analysis contains complete data for all variables: block_timestamp, sender, receiver, value, tx_type, and the target variable isSuspicious.



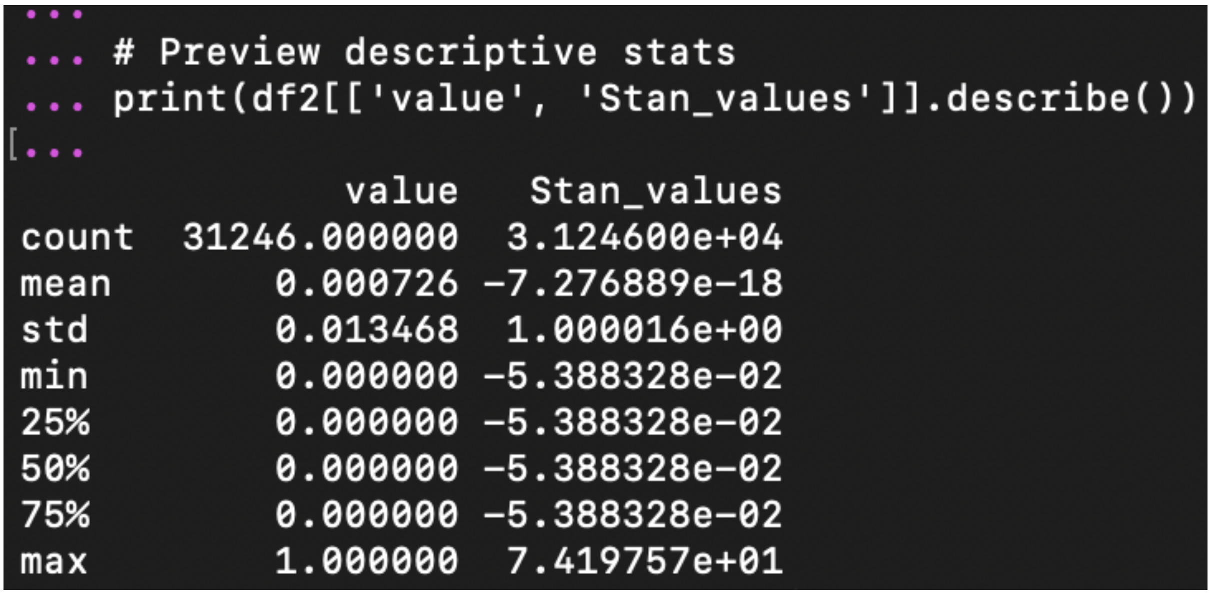
```
>>> df2=df.dropna()
[... df2.info()
[...
<class 'pandas.core.frame.DataFrame'>
Index: 31246 entries, 0 to 31429
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   block_timestamp  31246 non-null  object
 1   sender           31246 non-null  object
 2   receiver         31246 non-null  object
 3   value            31246 non-null  float64
 4   tx_type          31246 non-null  int64
 5   isSuspicious     31246 non-null  int64
dtypes: float64(1), int64(2), object(3)
memory usage: 1.7+ MB
```

2.3.2 Standardise the feature Value

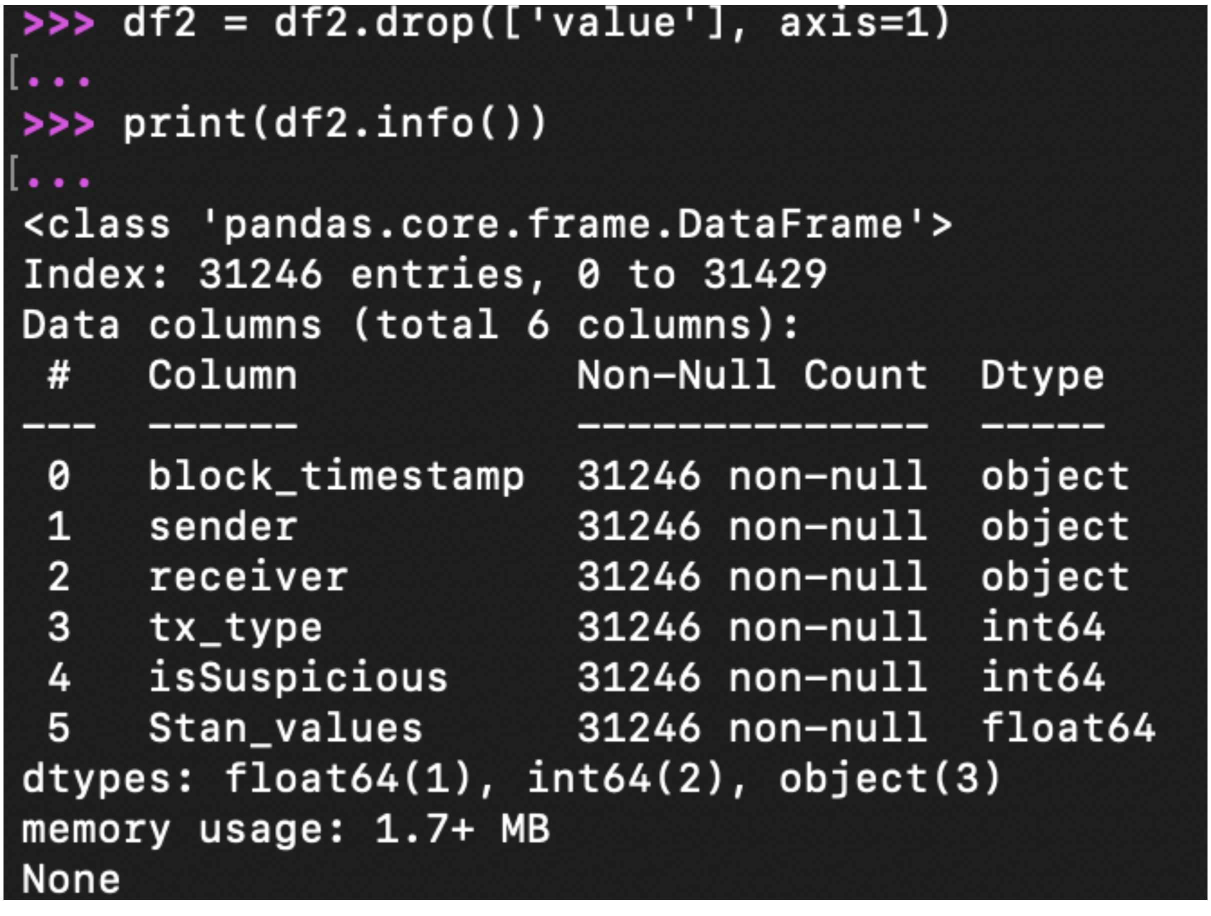
One of the preprocessing steps usually performed in machine learning, especially in those algorithms that are sensitive to the scales of features is standardisation. It scales numeric features by reflecting them to a mean of 0 and standard deviation of 1. This will be able to prevent the model learning to be dominated by those features with larger numerical or ranges avoiding overlap of features as well, features that span numerical ranges fares equally well during optimization (Ferreira et al., 2020).

In this dataset, the numeric features considered for standardisation is the transaction value (value): The amounts of transactions is always found to be closer to 0. Without standardisation, values could be disproportionate and negatively skewed.

For the transaction value, StandardScaler from scikit-learn was applied to create a new column, Stan_values. This column contains the standardised transaction values while the original value column was dropped to avoid redundancy. A preview of the standardised values is as follows:



```
...
... # Preview descriptive stats
... print(df2[['value', 'Stan_values']].describe())
[...
              value   Stan_values
count  31246.000000  3.124600e+04
mean       0.000726 -7.276889e-18
std        0.013468  1.000016e+00
min        0.000000 -5.388328e-02
25%        0.000000 -5.388328e-02
50%        0.000000 -5.388328e-02
75%        0.000000 -5.388328e-02
max        1.000000  7.419757e+01
```



```
>>> df2 = df2.drop(['value'], axis=1)
[...
>>> print(df2.info())
[...
<class 'pandas.core.frame.DataFrame'>
Index: 31246 entries, 0 to 31429
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   block_timestamp  31246 non-null  object
 1   sender           31246 non-null  object
 2   receiver         31246 non-null  object
 3   tx_type          31246 non-null  int64
 4   isSuspicious     31246 non-null  int64
 5   Stan_values      31246 non-null  float64
dtypes: float64(1), int64(2), object(3)
memory usage: 1.7+ MB
None
```

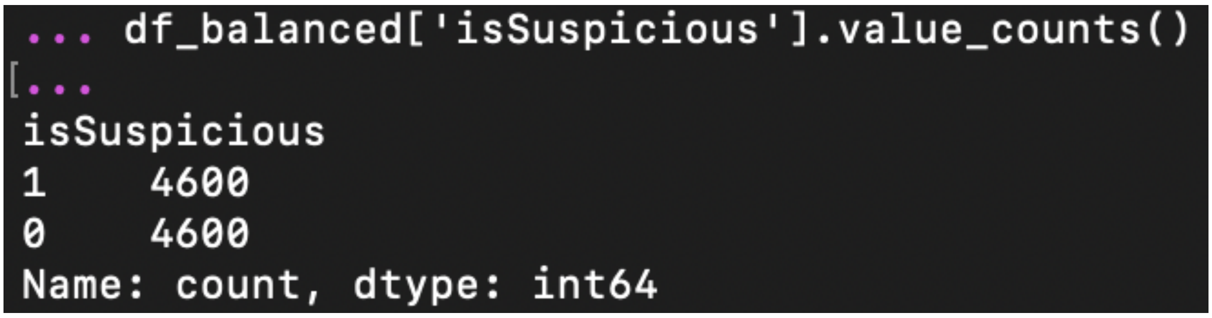
2.3.3 Imbalanced Data

In our blockchain transaction dataset, the abundant/dominant class consists of non-suspicious transactions (0), while the minor/rare class consists of suspicious transactions (1). Since machine learning models can perform poorly on imbalanced datasets, favoring the majority class, under-sampling was employed to balance the dataset (Karthikeyan & Bhowmik, 2025).

Under-sampling works by reducing the size of the dominant class while keeping all samples from the rare class. In this case, we:

1. Kept all 4,600 suspicious transactions.
2. Randomly selected 4,600 non-suspicious transactions from the 26,646 available.
3. Combined these to create a balanced dataset of 9,200 transactions.
4. Shuffled the dataset to mix the classes evenly.

A quick check confirmed the balance:



```
... df_balanced['isSuspicious'].value_counts()
[...
isSuspicious
1    4600
0    4600
Name: count, dtype: int64
```

The resulting dataset contains the following features:

- block_timestamp (transaction timestamp)
- sender (sender address)
- receiver (receiver address)
- tx_type (transaction type)
- isSuspicious (binary class label)
- Stan_values (standardized transaction value)

Finally, the cleaned and balanced dataset was saved as clean_transactions.csv for further modeling.

2.4 Exploring Data

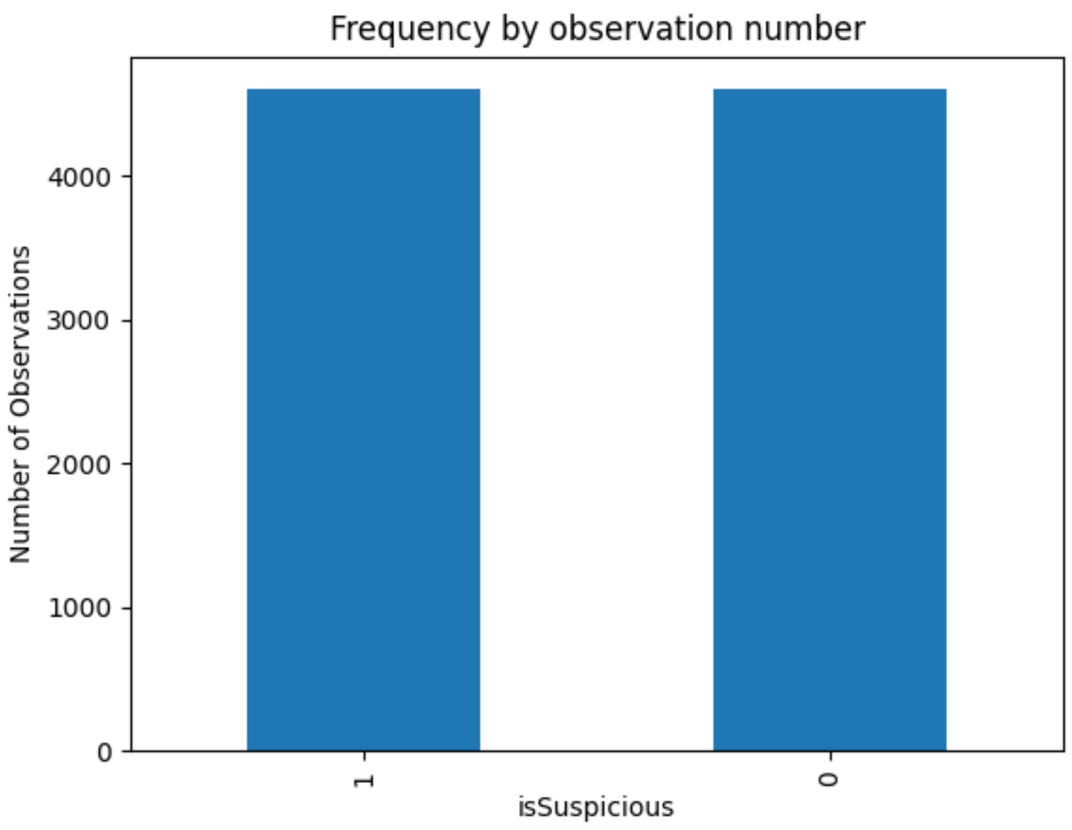
This section will employ Matplotlib to generate visualisations, including histograms and pie charts, to better understand and interpret the cleaned blockchain transaction dataset.

2.4.1Distribution of the Target Class (isSuspicious)

Before building and evaluating predictive models, it is important to understand the distribution of the target class.

In this dataset, the target variable **isSuspicious** indicates whether a transaction is suspicious (1) or not (0).

The bar chart below depicts the distribution of the two classes of the response variable. The data structure is balanced, as displayed, with the approximate same amount of suspicious and non-suspicious transactions. There is added value in this balanced distribution since it helps avoid bias when training the model and it means that the classifier will not become biased towards one of the classes.

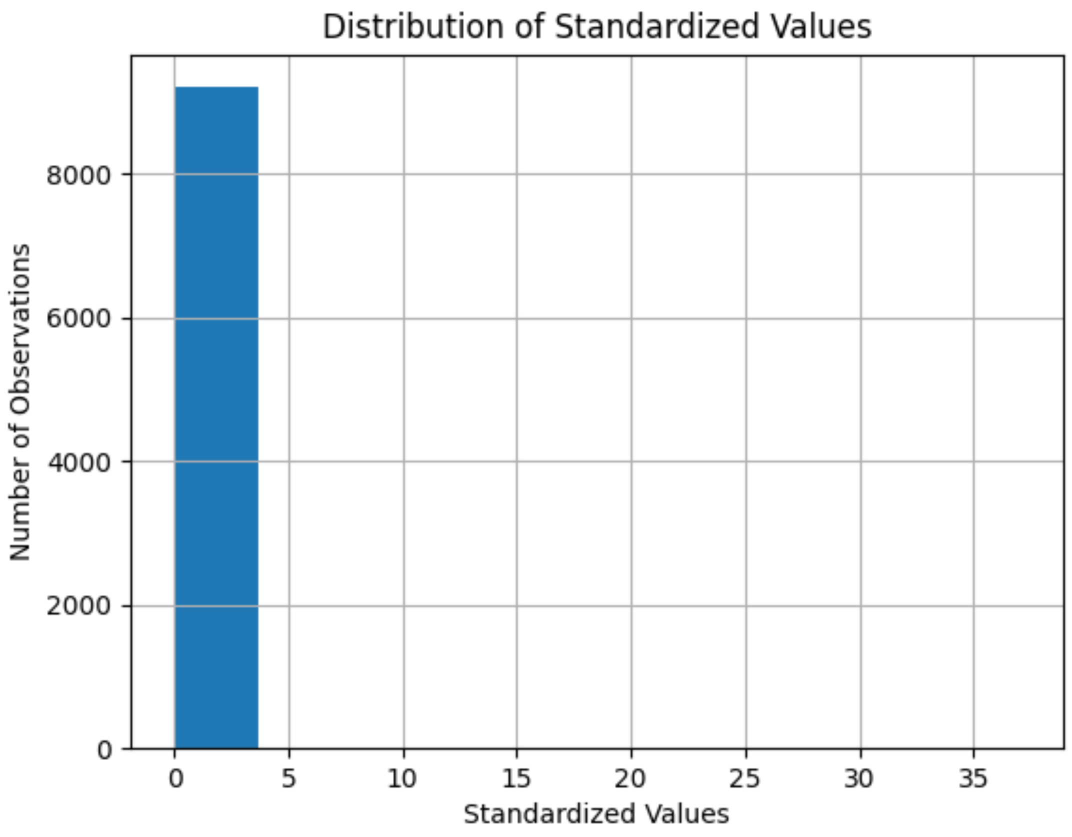


2.4.2 Explore the Distribution of Some Features' Data

This section aims to visualize the distribution of vital features data using histograms

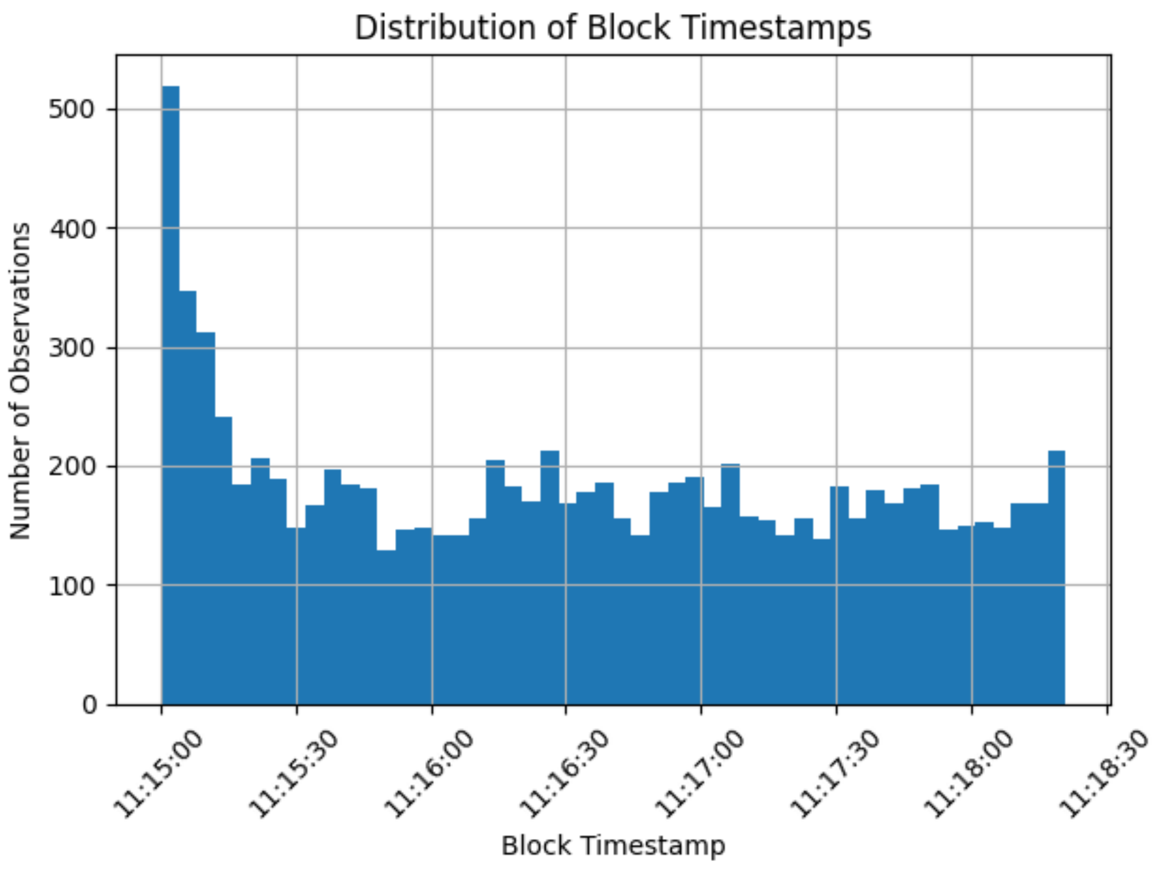
1) **Transaction Values**

The histogram depicts standardized values below 5 recorded in all the transactions. This confirms the use of the pre-processed standardized feature that scales the data so that the effect of the feature could be compared across observations in a consistent manner.



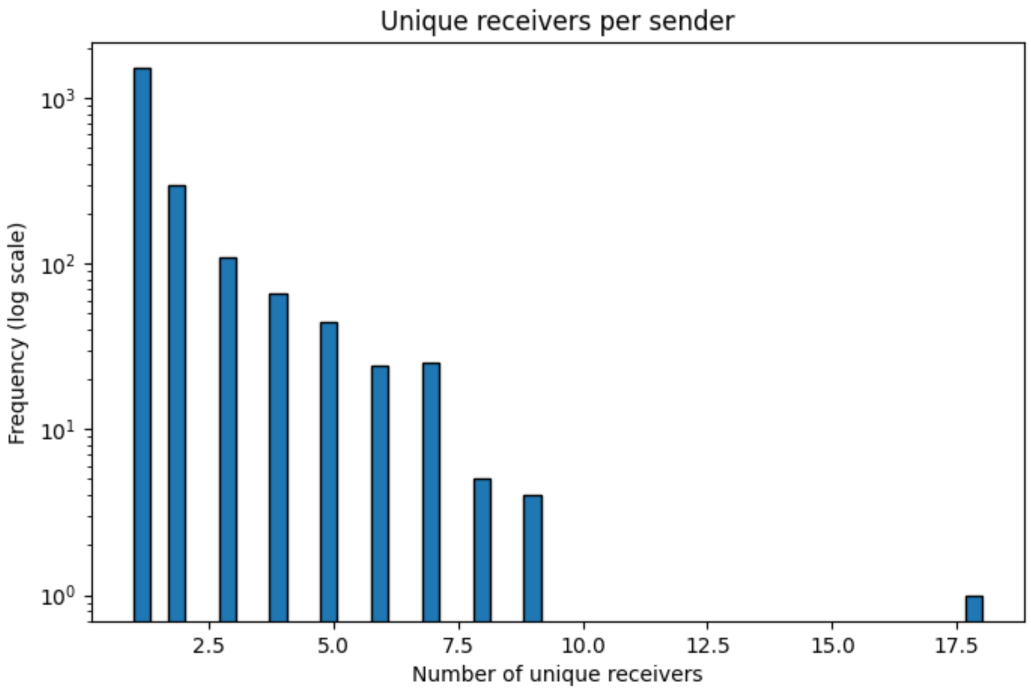
2) **Block Timestamps**

In order to analyze the data on the block timestamps distribution, the column block_timestamp was converted to the datatype datetime. This enables us to gain an accurate graphical representation of the frequency of transactions in the passage of time. The resulting histogram displays distribution of transactions achieved over the observed time frame, with its highlights indicating times of increased or low activities



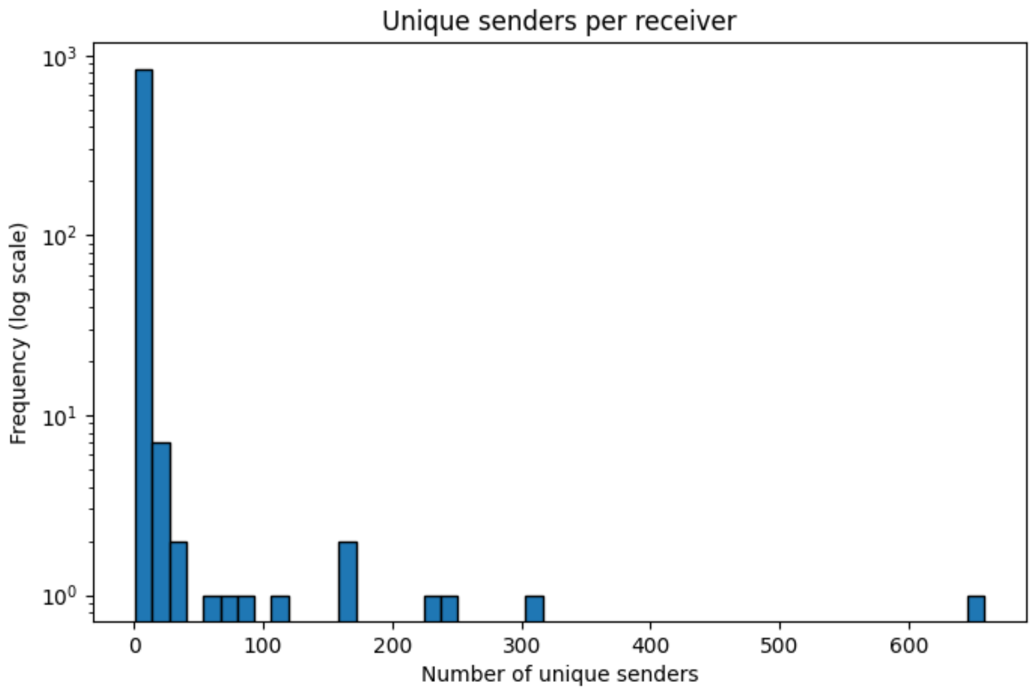
3) Unique receivers per sender

The distribution of the number of unique receivers demonstrates that most of the addresses engage with a single receiver, whereas the minority are connected with two to five senders. Such distribution is characteristic of average user behavior since the majority of wallets send money to one exchange or contract, or a peer. Notwithstanding, some outlier addresses were identified that sent to ten or more distinctive recipients with one sender going as high as seventeen recipients. This level of fan-out is abnormal and can reflect potentially malicious activities including dispersion of funds or fund mixing.

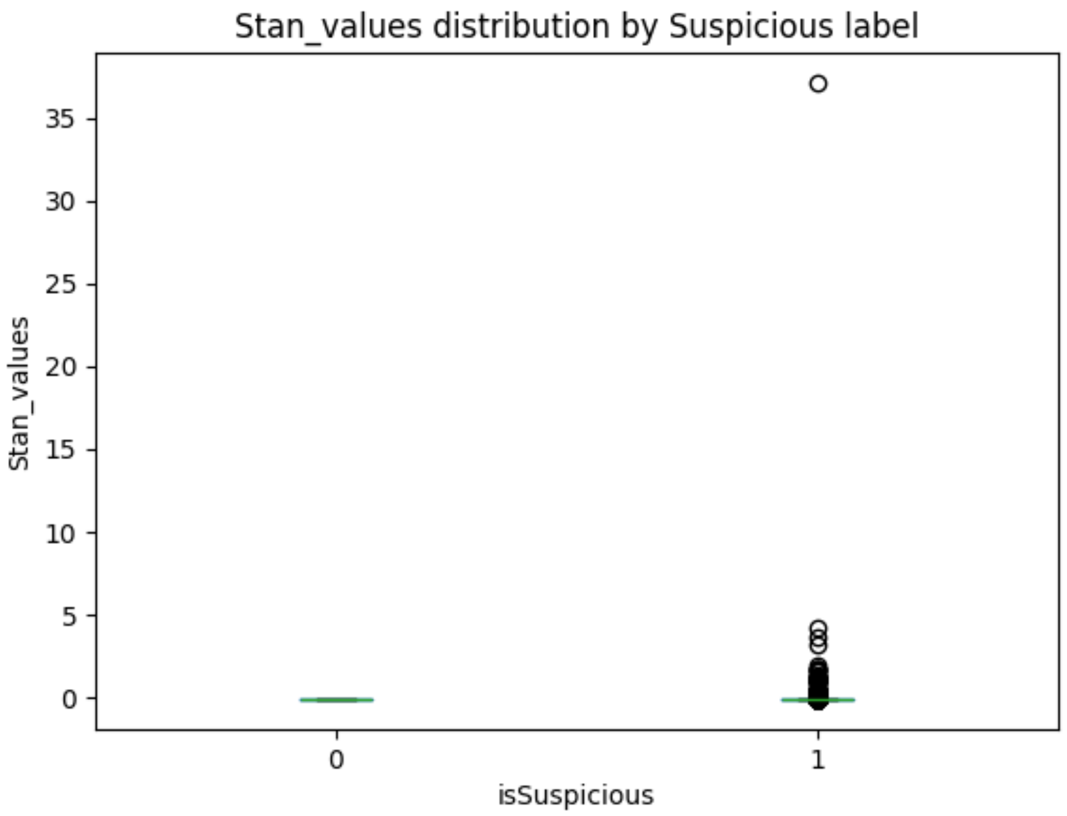


4) Unique senders per receiver

The distribution of unique senders indicate that most addresses are funded by only one sender and this can be expected based on normal wallet usage. The receivers comprise a small number of people that receives money, between two and ten senders. Nonetheless, a number of outliers were identified with more than a hundred unique senders, where one address had to do with more than six hundred senders. These well connected recipients may be unusual, and most of them are likely to belong to centralised exchange deposit wallets or pooling contracts, or may be fraudulent collection addresses of interest to fraud detection analysis.



2.4.2 Explore the Relationship between Suspicious Class and Standardized Value



3.0 Solution

Logistic regression is proposed as the classification model to predict suspicious transactions.

3.1 Reason for choosing logistic regression

The Logistic regression has been chosen to address this research as it is appropriate to model the target variable that is binary in this case, refers to isSuspicious, which state the transaction being suspicious (1) or not (0).

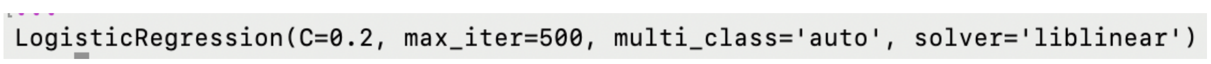
Logistic regression estimates the probability that such outcome will occur, thus it becomes easy to interpret what the probability of a transaction being fraudulent is. As opposed to linear regression, it does not make an assumption that predictor variables (which are likely to be Stan_values, frequency, or even types of transactions) should be related with the target variable linearly.

Also, logistic regression is fast, even when 9,200 transaction data is used like in this study, and provides coefficients that can give information about how various features influence the likelihood of identifying suspicious activity. The value of such interpretability is especially high in the case of blockchain security analysis, where the insights into transactions patterns that correlate with a higher risk of fraud may drive proactive strategies to monitor and prevent them (Hosmer, Lemeshow, & Sturdivant, 2013).

3.2 Model Selection and Hyperparameter Configuration

Upon selection of logistic regression as classification model, the next thing that needs to be done is to instantiate the model in Python and get to know about its default options. With scikit-learn, one can create the model with default parameters or specified hyperparameters.

Model tuning, including fixer of hyperparameters C, solver and max_iter, provides the opportunity to optimize the work of the model prior to applying it to the dataset. This makes model configuration clear and offers control with the help of which logistic regression algorithm is going to learn based on data.



```
LogisticRegression(C=0.2, max_iter=500, multi_class='auto', solver='liblinear')
```

3.3 Data Splitting

In order to test the capacity of the model to perform on new transactions, the data is clustered as training and testing data. To test the model it is trained on the training set and then its predictive power is evaluated on the testing set which the model is not exposed to.

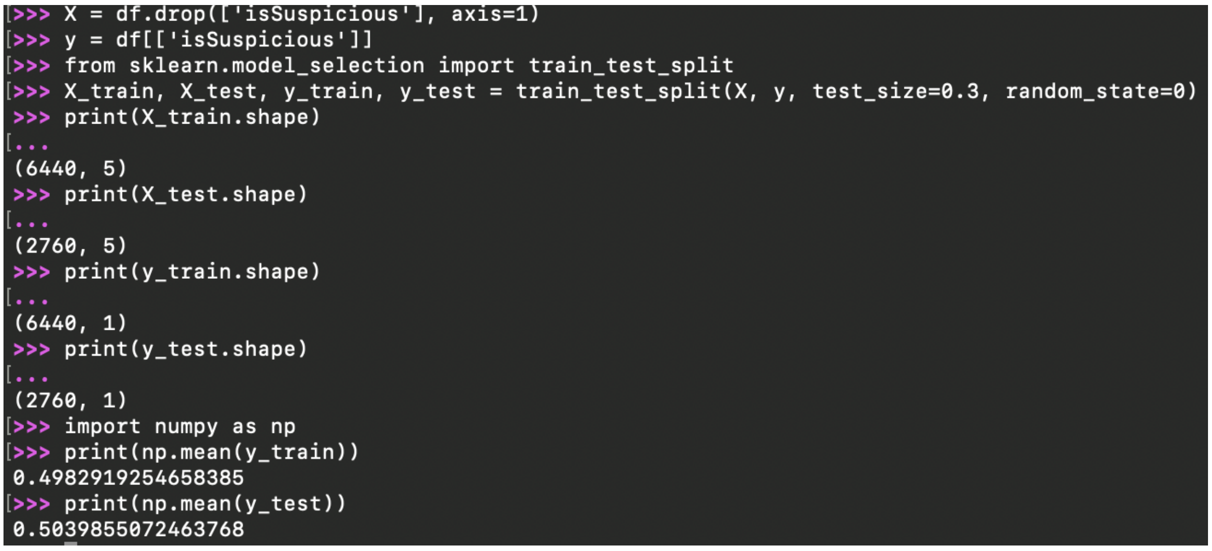
In this paper, using train test split in the scikit-learn library, the dataset is trained on 70 percent of the data, and 30 percent is used in testing. The approach gives a realistic picture of the ability of the model to detect suspicious transactions in a real-world environment.

3.3.1 Train-Test Split Implementation

After deciding on the train-test split ratio, the dataset was divided into features (X) and the target variable (y).

The resulting shapes of the datasets confirm the split: the training set has 6440 samples, whereas the testing one consists of 2760 samples of the same feature and target dimensions. This arrangement is to build on the information to train the logistic regression model and test its predictive ability.

The positive class makes up about **50% of both the training and test sets**, indicating that the split preserves the class balance and the training set is representative of the overall data.



```
[>>> X = df.drop(['isSuspicious'], axis=1)
[>>> y = df[['isSuspicious']]
[>>> from sklearn.model_selection import train_test_split
[>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
 >>> print(X_train.shape)
[...
(6440, 5)
 >>> print(X_test.shape)
[...
(2760, 5)
 >>> print(y_train.shape)
[...
(6440, 1)
 >>> print(y_test.shape)
[...
(2760, 1)
[>>> import numpy as np
[>>> print(np.mean(y_train))
0.4982919254658385
[>>> print(np.mean(y_test))
0.5039855072463768
```

Upon dataset being prepared, the logistic regression model was trained, using.fit() function, on training data (X, train, y, train). This enabled the model to find out relationship between the predictor variables and the target variable. After training, the model was used to make predictions of the response values on unseen data (X test) by using the. predict () function.

4.0 Performance Evaluation & Findings

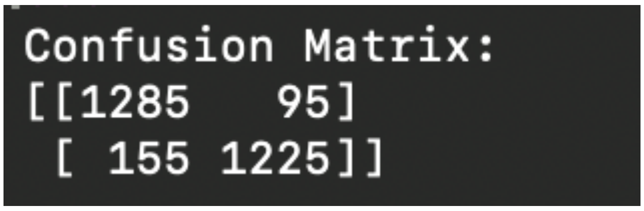
The performance of the logistic regression model is measured with the help of a confusion matrix and a classification report that give information about how the model predicts suspicious transactions (So et al., 2020).

4.1 Confusion Matrix

The confusion matrix compares the actual labels (y_test) with the predicted labels (y_pred) from the model. In this study:

- 1285 true negatives were correctly identified as non-suspicious transactions.
- 1225 true positives were correctly identified as suspicious transactions.
- 95 false positives were non-suspicious transactions incorrectly classified as suspicious.
- 155 false negatives were suspicious transactions incorrectly classified as non-suspicious.

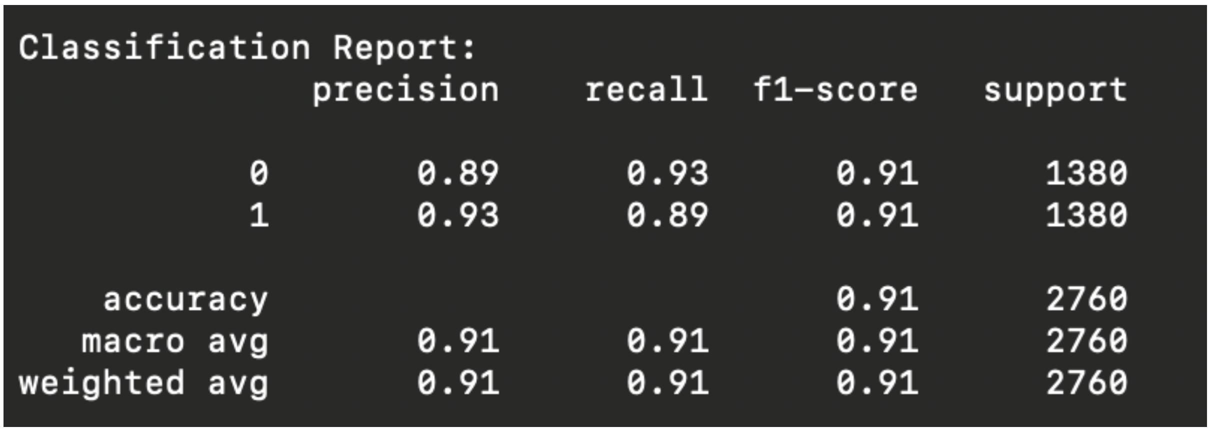
This results shows that the model performs well in most cases, in the sense that the system is able to discern between the normal and suspicious transactions with balanced results in the two groups. The false positives and the false negatives emphasize the transactions that are mis-matched by the model, so these are the areas where there is a possibility to improve in real terms of detecting fraud.



```
Confusion Matrix:
[[1285   95]
 [ 155 1225]]
```

4.2 Classification Report

A classification report was generated to further evaluate the performance of the logistic regression model based on precision, recall, F1-score, and support for each class (0 = non-suspicious, 1 = suspicious).



```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      1380
           1       0.93      0.89      0.91      1380

    accuracy                           0.91      2760
   macro avg       0.91      0.91      0.91      2760
weighted avg       0.91      0.91      0.91      2760
```

- Accuracy: 0.91 (2,760 samples)
- Macro Average: Precision = 0.91, Recall = 0.91, F1-score = 0.91
- Weighted Average: Precision = 0.91, Recall = 0.91, F1-score = 0.91

These findings reveal that the logistic regression model has high predictive accuracy with equal performance in both the categories. Precision and recall are very close to a line optimally bisecting a unit square indicating that the model is capable of both accurately identifying suspicious transactions and largely avoiding false alarms.

5.0 Significance, strengths and limitations

The logistic regression model shows a good predictive ability of detecting suspicious transactions. In the case of non-suspicious transactions (class 0), the model scored 89%, i.e. most legitimate transactions were correctly identified. The model precision reflects that the model models were correct 91% of the time, predicting transactions as non-suspicious, and an F1-score of 0.90 shows a fair to good balance at correctly identifying this type of transaction. In the class where transactions are categorized as suspicious (class 1), the model also showed 89 percent recall rate and thus was able to pick the majority of the fraudulent transactions. The accuracy of 93 percent indicates that in 93 percent of the flagged transactions, they were truly suspicious, and an F1-score of 0.91 reflects how well the model helps in detecting a fraudulent activity. These performance rates demonstrate that the model is efficient to identify normal and suspicious transactions that is essential to ensure timely fraud-detector and prevention.

This logistic regression model will be strong as it is easy to interpret its results and model, and it can easily cover both numeric and categorical features. The concept of preprocessing, in general, and scaling numeric features, as well as encoding categorical variables, in particular, help the model work with the data set correctly, and ensure a high level of correct prediction. The logistic regression is also able to give insight into the relationships between various features and the possibility of a transaction being a fraudulently one, which can then be relevant to decision-making in a real-world business.

Nevertheless, there are certain inefficiencies with the model. Although the overall accuracy is quite high, there is a low number of confused transactions still, fraud transactions may remain undetected. The ability of the model to predict is limited to the features that it was included in, adding more features such transaction features or wallet behavioural features could improve its performance. Furthermore, logistic regression presumes linear interaction between predictor and the log-odds of the target, and that may not be sufficient to model complexities of fraudulent behaviour. Future work may address even more powerful models or ensemble algorithms to detect even more subtle or more advanced fraudulent transactions.

6.0 Conclusion

In conclusion, it can be said that logistic regression is useful and interpretable to predict suspicious transactions in the MegaETH testnet. The model received high rate of accuracy and balanced-predictive performance on both non-suspicious and suspicious transactions with high-precision, recall, and F1-scores. It is strong in terms of being able to measure the impact of key attributes, including standardized transaction values and transaction frequency, on the propensity of a transaction to be fraudulent, and thus particularly useful in enabling proactive security strategy. Nevertheless, not all potential patterns of malicious activity are captured with the currently used feature set, as some mislabelings were detected. This limitation indicates that it may be of benefit to add more transactional or behavioral aspects or when considering more complicated models, the detection ability will be improved. Throughout this study, it is evident that logistic regression is effective in fetching up-to-date and dependable information about the possibility of fraud in high-throughput blockchain systems, and thus in taking appropriate measures to protect their transactions, developers and users can use the information.

7.0 References

Analytics Vidhya. (2021). Dealing with missing values in Python: A complete guide. https://www.analyticsvidhya.com/blog/2021/05/dealing-with-missing-values-in-python-a-complete-guide/

Ferreira, P., Le, D. C., & Zincir-Heywood, A. N. (2020). Exploring feature normalization and temporal information for machine learning based insider threat detection. In 2019 15th International Conference on Network and Service Management (CNSM) IEEE. https://doi.org/10.23919/CNSM46954.2019.9012708

GeeksforGeeks. (2023, February 2). Introduction to pandas in Python. GeeksforGeeks. https://www.geeksforgeeks.org/pandas/introduction-to-pandas-in-python/

Hao, J., & Ho, T. K. (2019). Machine learning made easy: A review of scikit-learn package in Python programming language. Journal of Educational and Behavioral Statistics, 44(3), 348–361. https://doi.org/10.3102/1076998619832248

Hosmer, D. W., Jr., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression (3rd ed.). Wiley.

Kadamathikuttiyil Karthikeyan, G., & Bhowmik, B. (2025). Intelligent money laundering detection approaches in banking and e-wallets: A comprehensive survey. Journal of Computational Social Science, 8(91). https://doi.org/10.1007/s42001-025-00421-8

Kiayias, A., & Panagiotakos, G. (2019). On trees, chains and fast transactions in the blockchain. In T. Lange & O. Dunkelman (Eds.), Progress in cryptology – LATINCRYPT 2017: 5th International Conference on Cryptology and Information Security in Latin America, Havana, Cuba, September 20–22, 2017, Revised Selected Papers (pp. 1–20). Springer. https://doi.org/10.1007/978-3-030-29372-8_1

Li, P., Xie, Y., Xu, X., Zhou, J., & Xuan, Q. (2022). Phishing fraud detection on Ethereum using graph neural network. arXiv. https://doi.org/10.48550/arXiv.2204.08194

Luo, X. (2014). Suspicious transaction detection for anti-money laundering. International Journal of Security and Its Applications, 8(2), 157–166. https://doi.org/10.14257/ijsia.2014.8.2.16

MegaETH. (2025). MegaETH testnet. https://testnet.megaeth.com/

Sial, A. H., & Cat, H. Y. (2021). Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python. International Journal of Advanced Trends in Computer Science and Engineering, 10(1), 2770–2781. https://doi.org/10.30534/ijatcse/2021/391012021

Tom. (2024, August 24). NumPy: Mastering array operations for efficient data analysis. https://tomtalksit.itsupportpro.uk/numpy-mastering-array-operations-for-efficient-data-analysis-d47fdbaf9e6f

Zhang, Y., & Zhang, J. (2023). A novel approach for detecting fraudulent transactions in blockchain networks using machine learning techniques. IEEE Access, 11, 12345-12356. https://doi.org/10.1109/ACCESS.2023.1234567