Context

In this semester, the focus is on the dataset. The work on dataset can be separated into dataset selection, feature expansion, feature selection, and data cleaning. Also, I spent some time reading papers related to why tree-based models are suitable for this topic, and how other scholars train their models for Ethereum fraud detection. Each of these steps will be explained in depth in their respective sections below.

Dataset Selection

5 datasets are shortlisted as our dataset. They are:

* Dataset 1: Ethereum Fraud Detection by Kaggle user Vagiv Aliyev.
* Dataset 2: Ethereum Fraud Detection Aggregated Dataset by Kaggle users Saket P. and Suresh Siliveru.
* Dataset 3: Ethereum Fraud Dataset by Kaggle user Guille Escobero.
* Dataset 4: Ethereum Fraud Detection Models by GitHub user sepandhaghighi.
* Dataset 5: Labelled Transaction-based Dataset of Ethereum Network by Dr. Salam Al-E’Mari of University of Petra, Jordan.

The reference of the datasets are located in the reference section of this report.

Here is the analysis on why the datasets are chosen/not chosen:

Dataset 1 is a list of Ethereum addresses that are flagged as either fraud or not. It contains 9840 rows with 51 columns. This is the most popular dataset in Kaggle of this topic, and it was used as a dataset for other scholars to train models for Ethereum fraud detection, such as Aziz et. al (2022). This dataset also contains the most features (columns) out of all in the shortlist. However, it is not chosen due to its lack in rows, a dataset that is large enough is preferred.

Dataset 2 is also a list of Ethereum addresses that are flagged as fraud or not. It contains 19682 rows and 20 columns. The row count is basically the double of dataset 1, but its lack of features (columns) deters me from choosing it.

Dataset 3 is another list of flagged Ethereum addresses. It has 12146 rows and 34 columns. The number of rows and columns are in between that of datasets 1 and 2, and its features are similar to them. Since I did not choose either of them, there is no reason to choose dataset 3.

Dataset 4 is the combined set of datasets 1 and 2. The rows are increased to 20302 but the column count is only 17. It is still not ideal to use due to lack of columns.

Dataset 5 is a list of Ethereum transactions that are flagged as fraud or not by matching fraud addresses. It has 71250 rows and 18 columns. This is the largest I have obtained. After consultation, this dataset is selected.

Dataset 5 has these columns:

|  |  |
| --- | --- |
| Field name | Description |
| hash | A transaction hash |
| nonce | A scalar value indicating the number of transactions sent FROM an Ethereum address |
| transaction\_index | Transaction index of a block |
| from\_address | The address of the sender of the transaction |
| to\_address | The address of the receiver of the transaction |
| value | Transaction amount in Wei |
| gas | Gas used by the sender for the transaction |
| gas\_price | Gas price in Wei |
| input | Data transferred from the sender of the transaction |
| receipt\_cumulative\_gas\_used | Gas used in the transaction as a block |
| receipt\_gas\_used | Total gas used for the transaction |
| block\_timestamp | Timestamp of the transaction |
| block\_number | The transaction block number |
| block\_hash | The transaction block hash |
| from\_scam | Whether the sender is fraudulent or not |
| to\_scam | Whether the receiver is fraudulent or not |
| from\_category | The reason of the sender of the transaction being marked as scam |
| to\_category | The reason of the receiver of the transaction being marked as scam |

Feature Expansion

The dataset selected is a transaction list. To expand the columns, I attempted to retrieve the transaction information and address information from the sender and receiver. To get these information, I retrieved them via Blockscout{6] with their free API. Here is the schema of this APIs, which some of them are included into the expanded dataset.

Transaction:

|  |  |
| --- | --- |
| Field name | Description |
| timestamp | Duplicated from “block\_timestamp” above |
| fee | Total fee of the transaction in Wei |
| gas\_limit | Upper limit of gas used in the transaction |
| block | Block number |
| status | The status of the block, it must be “ok” since it is recorded onto Ethereum and its immutable |
| method | Method of transfer |
| confirmations |  |
| type | Unidentified |
| exchange\_rate | Ethereum exchange rate at the time of transaction |
| to | Object containing receiver address detail |
| tx\_burnt\_fee | [all blank] |
| max\_fee\_per\_gas | [all blank] |
| result | The result of the transaction, it must be “success” [1 value] |
| hash | Transaction hash |
| gas\_price | Gas price in Wei |
| priority\_fee | [all blank] |
| base\_fee\_per\_gas | [all blank] |
| from | Object containing sender address detail |
| token\_transfers | Object containing fields found in address detail and this table [redundant] |
| tx\_types | Where this transaction is from: coin/token transfer, contract call |
| gas\_used | Amount of gas used in the transaction |
| created\_contract | [all blank] |
| position | Transaction index of the block [redundant] |
| nonce | Nonce of the transaction [redundant] |
| has\_error\_in\_internal\_txs | [all blank] |
| actions | [all blank] |
| decoded\_input | The transaction method call if it exists |
| token\_transfers\_overflow | Whether the transaction token overflows |
| raw\_input | Hex value transaction input |
| value | The amount transferred in Wei |
| max\_priority\_fee\_per\_gas | [all blank] |
| revert\_reason | [all blank] |
| confirmation\_duration | Period to confirm the transaction |
| tx\_tag | [all blank] |

Address:

|  |  |
| --- | --- |
| Field name | Description |
| block\_number | Block number of the address |
| coin\_balance | Current balance of the address at the time of fetching the block [redundant] |
| creation\_tx\_hash | Transaction hash on address creation |
| creator\_address\_hash | Address hash on address creation |
| exchange\_rate | Exchange rate on address creation |
| has\_beacon\_chain\_withdrawals | Self-explanatory |
| has\_custom\_methods\_read | [all false] |
| has\_custom\_methods\_write | [all false] |
| has\_decompiled\_code | Self-explanatory |
| has\_logs | Self-explanatory |
| has\_methods\_read | Whether the address has any read methods |
| has\_methods\_read\_proxy | [too few rows] |
| has\_methods\_write | Whether the address has any write methods |
| has\_methods\_write\_proxy | [too few rows] |
| has\_token\_transfers | Whether the address has transferred tokens to another address |
| has\_tokens | Whether the address has tokens when fetched |
| has\_validated\_blocks | Whether the block is validated |
| hash | The address hash |
| implementation\_address | [all blank] |
| implementation\_name | [all blank] |
| is\_contract | [all false] |
| is\_verified | [all false] |
| name | [all blank] |
| private\_tags | [all blank] |
| public\_tags | [all blank] |
| token | [all blank] |
| watchlist\_address\_id | [all blank] |
| watchlist\_names | [all blank] |

Feature Selection

Selecting the feature for the model, all the columns have to meet these conditions: they must have 2 or more values, the value distribution cannot be overly dominant (1:20000), they cannot be blank.

In addition, many of the columns shall be converted into Boolean or numerical values if they original value is a string/object/array value from the API or the dataset. Also, the block timestamp is converted into time elapsed. After the selection and conversion, these columns are present:

|  |  |
| --- | --- |
| Field name | Data type |
| hash | String |
| nonce | Int |
| transaction\_index | Int |
| from\_address | String |
| to\_address | String |
| value | Int |
| gas | Int |
| gas\_price\_txn | Int |
| input | String |
| receipt\_cumulative\_gas\_used | Int |
| receipt\_gas\_used | Int |
| block\_number | Int |
| block\_hash | String |
| confirmations | Int |
| exchange\_rate\_txn | Float |
| token\_transfers\_overflow | Boolean |
| block\_number\_balance\_updated\_at\_from | Int |
| coin\_balance\_from | Int |
| exchange\_rate\_from | Float |
| has\_beacon\_chain\_withdrawals\_from | Boolean |
| has\_token\_transfers\_from | Boolean |
| has\_tokens\_from | Boolean |
| has\_validated\_blocks\_from | Boolean |
| block\_number\_balance\_updated\_at\_to | Int |
| coin\_balance\_to | Int |
| exchange\_rate\_to | Float |
| has\_beacon\_chain\_withdrawals\_to | Boolean |
| has\_token\_transfers\_to | Boolean |
| has\_tokens\_to | Boolean |
| has\_validated\_blocks\_to | Boolean |
| fee\_value | Int |
| confirmation\_duration\_0 | Int |
| confirmation\_duration\_1 | Int |
| is\_scam | Boolean |
| elapsed\_time | Int |
| tx\_token\_transfer | Boolean |
| tx\_coin\_transfer | Boolean |
| tx\_contract\_call | Boolean |

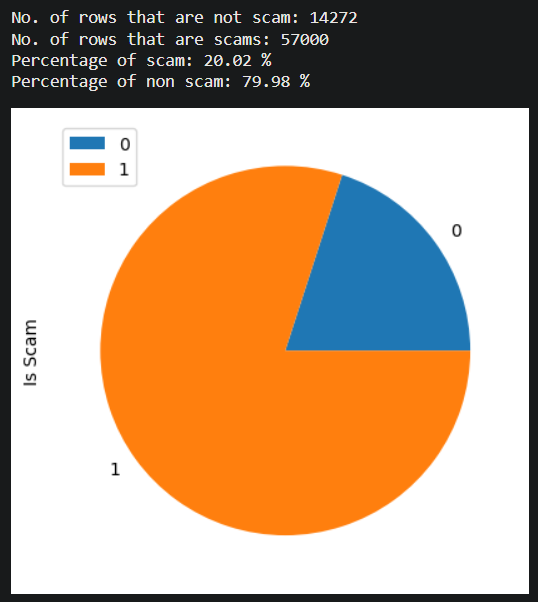
Note that in the original dataset (not the final one), the transaction is defined as it is from/to a scam address. In the final dataset, both cases are defined as scam transactions, and the reason of the transaction being a scam is removed.

Data Cleaning

Duplicated rows are determined by having the same transaction hash. In the selected dataset, there is only 1 duplicated row, which is removed from the final dataset.

The only columns with blank values after the above steps are the sender’s and receiver’s coin balance. To deal with those blank values, they are replaced with the balance value 0. The other columns are not altered for now.

The scam vs non-scam proportion is shown in this figure:



Papers

Regarding research on this topic, several papers have been read.

Using Grinsztajn’s research on comparing the performance of tree-based models versus neural networks (NN) on tabular data (2022), the paper concluded that tree-based models outperform NNs on tabular data due to 2 main reasons: 1. Models based on decision trees are not biased towards low-frequency functions, while NNs are. 2. Uninformative features affect NNs, which they should not be affected, lowering their accuracy.

Regarding the neural network model for this project, this paper by Ivanov (2021) illustrates the accuracy of the combination of Gradient Boosting Decision Trees (GBDT) and Graph Neural Networks (GNN) called Boosted GNN (BGNN). It carries the benefits of both GBDT and GNNs: it uses the GBDT model to build hyperplane decision boundaries, which are common in heterogeneous data, and utilizes GNN which uses relational information to refine predictions.

Projection

More plots shall be illustrated afterwards on each feature of the dataset, if possible, it should be done within 1.5 weeks. Afterwards, work shall be started to build the model.

References

Aziz, R.M., Baluch, M.F., Patel, S. et al. LGBM: a machine learning approach for Ethereum fraud detection. Int. j. inf. tecnol. 14, 3321–3331 (2022). <https://doi.org/10.1007/s41870-022-00864-6>

Ivanov, S., Prokorenkova, L. Boost Then Convolve: Gradient Boosting Meets Graph Neural Networks. ICLR Conference Paper (2021). <https://doi.org/10.48550/arXiv.2101.08543>

Grinsztajn, L., Oyallon, E., Varoquaux, G., Why do tree-based models still outperform deep learning on tabular data? (2022) [arXiv:2207.08815](https://arxiv.org/abs/2207.08815) [cs.LG].

Shortlisted Datasets:

[1]: <https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset>

[2]: <https://www.kaggle.com/datasets/saket03p/ethereum-fraud-detection-aggregated-dataset>

[3]: <https://www.kaggle.com/datasets/gescobero/ethereum-fraud-dataset?rvi=1>

[4]: <https://github.com/sepandhaghighi/Ethereum-Fraud-Detection-Models>

[5]: <https://github.com/salam-ammari/Labeled-Transactions-based-Dataset-of-Ethereum-Network/tree/master>