Hochschule Fresenius University of Applied Sciences
Faculty of Economics & Media
International Business School
Industrial Engineering and International Management
Cologne Campus

Report on

Detecting Fraud Transactions on Ethereum Blockchain

Avishkar Kanade
Student ID No.: 400287820
3.Semester

Module: Technical Applications and Data Management
Lecturer: Mrs. Barbara Lampl
Due Date: 06 Feb 2023

1. Introduction

Cryptocurrency is a digital currency that operates on decentralized systems using blockchain technology independently of a central bank.
In the decentralized nature of cryptocurrencies, no single entity can manage their command, unlike traditional currencies like the US dollar, Euro, and many more. One such cryptocurrency is Ethereum which Vitalik Buterin developed in 2015. Ethereum is a blockchain system that enables one to send cryptocurrencies to anyone for a nominal charge. All the cryptocurrencies on the Ethereum blockchain follow the Ethereum Request for Comment(ERC20) standard to transact. Ethereum provides everyone, regardless of background or location, with an uncluttered usage of digital transactions at a very nominal rate in a very secure manner. Due to the decentralized framework of Ethereum, no organization or institution can be convicted for the blockchain's acts. As a result, It becomes challenging to identify users who misuse the platform to conduct fraudulent transactions. These factors have led to fraudulent conduct and cyber crimes like money laundering, phishing, and dark web weapons purchases using various cryptocurrencies.

Due to the lack of adequate supervision on the blockchain market, Various fraud technologies have also begun to point to the blockchain, especially in the field of financial investment; there have been some scams that induce investors with high returns. Because many investors do not understand the blockchain technology and are tempted by the appreciation of various cryptocurrencies, they are readily induced by some criminals, leading to severe economic losses. This has led to doubts regarding the long-term sustainability of the technology in the minds of investors and stakeholders.

One idea is to detect fraud manually by viewing the source code, but the smart contract implementation requires only bytecode, and the source code is hidden. As a result, Data Analysts have started to study chunks of data to discover underlying patterns and factors contributing to fraud on Ethereum. In this regard, the Ethereum Fraud Detection Dataset has become an essential resource for studying the problem of fraud on the Ethereum network.

The Ethereum Fraud Detection Dataset is a collection of 9841 instances and 51 attributes that provides valuable information on the characteristics of both fraudulent and legitimate transactions, including the transaction amount, the number of inputs and outputs, the block height, and the time of the transaction, among other attributes. This information can be used to develop models and algorithms for detecting and preventing fraud on the Ethereum network.

In our study, we aim to study various statistical models such as decision trees, random forest, gradient boosted trees to detect fraudulent transactions on the Ethereum blockchain. We look for abnormalities in the transactional dataset. Transactions that differ from the norm are defined as aberrant or suspicious. In addition, these transactions could be legitimate or fraudulent, but they are worth investigating. Our goal is to analyze each model's accuracy and understand the different combinations of attributes they use to investigate whether a particular transaction is legitimate or fraudulent.

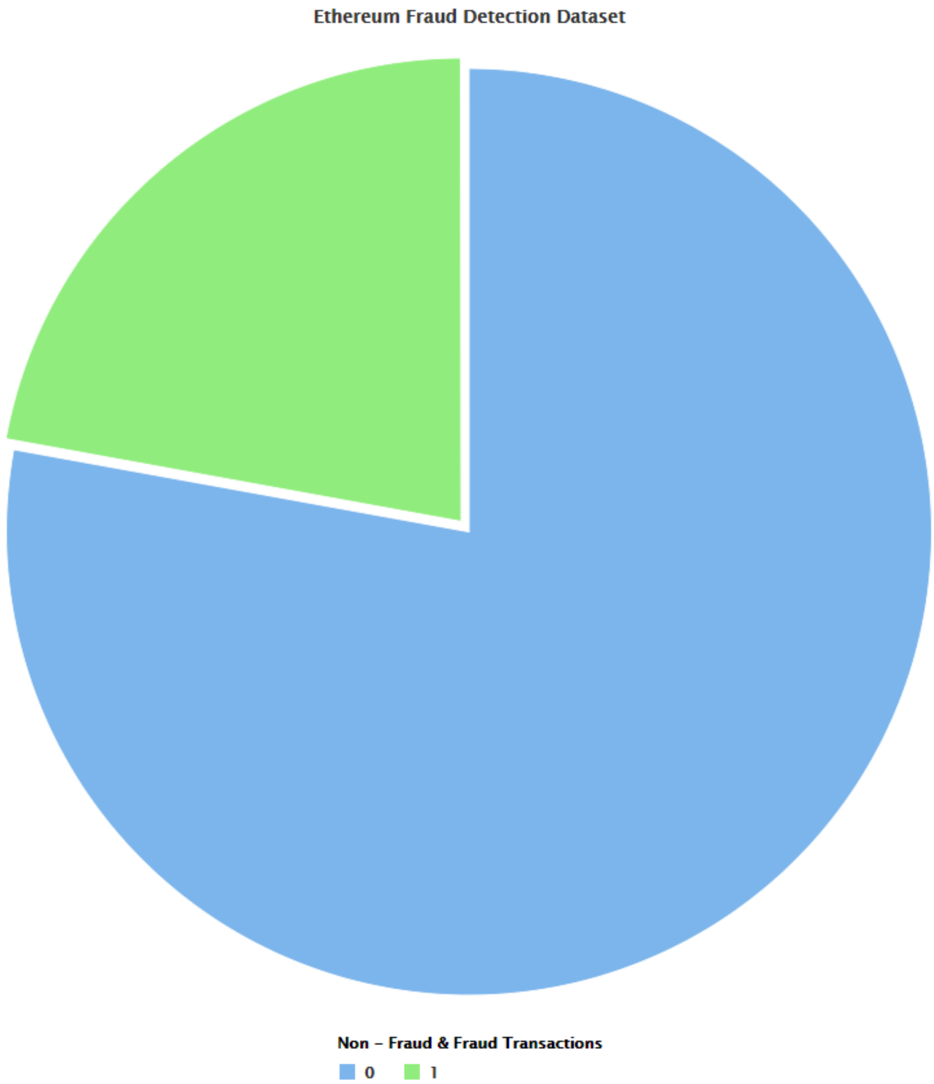


Figure. 1: Fraud and non-fraud examples in the Ethereum dataset.

2. Business Understanding

The increasing prevalence of fraudulent transactions on the Ethereum blockchain has become a cause for concern for its Chief Technology Officer (CTO). The CTO and the risk management team at Ethereum recognize the potential damage that these transactions can have on Ethereum's reputation and brand image, which is a crucial factor in the success and continued adoption of any technology or platform. In addition, a large number of fraudulent transactions can damage trust among users and stakeholders, harming Ethereum's profitability and ability to draw new users and investors.

The risk management team at Ethereum has engaged our services as a Data Analyst to address the issue of fraudulent transactions on its blockchain. The main objective of our project is to evaluate the dataset of Ethereum transactions in order to gain helpful knowledge about fraud trends and their underlying causes. This will include identifying the significant attributes that need to be focused on in order to prevent such fraudulent transactions in the future.

In addition, Ethereum expects us to identify and implement the best statistical models, such as the Random Forest and Gradient Boosted Trees, to classify transactions accurately as either legitimate or fraudulent. The selection of appropriate models will be based on analyzing the Ethereum Fraud Detection Dataset and considering relevant performance metrics such as accuracy, precision, and recall. The deployment of these models will be an essential step in detecting and preventing fraudulent transactions on the Ethereum blockchain, which will help to maintain its reputation and ensure its continued success.

As Data Analysis experts, we must do our due diligence on the dataset provided and come up with the best possible fraud detection models.

3. Dataset Preparation

The present study utilizes a dataset containing 9841 instances and 51 attributes pertaining to known instances of fraudulent and valid transactions conducted over the Ethereum blockchain. Upon initial examination, it was noted that 14 of the 51 attributes contained null values, and an additional attribute comprised unique addresses that were deemed irrelevant for the current analysis. As a result, these attributes were dropped using Microsoft Excel, resulting in a final dataset consisting of 9841 instances and 36 attributes. We are then using Rapidminer to further process the altered dataset.

Row No.	FLAG	Avg min bet...	Avg min bet...	Time Diff bet...	Sent tnx	Received Tnx
1	0	844.260	1093.710	704785.630	721	89
2	0	12709.070	2958.440	1218216.730	94	8
3	0	246194.540	2434.020	516729.300	2	10
4	0	10219.600	15785.090	397555.900	25	9
5	0	36.610	10707.770	382472.420	4598	20
6	0	9900.120	375.480	20926.680	2	3
7	0	69.460	629.440	8660.350	25	11
8	0	1497.390	176.840	319828.050	213	5
9	0	0	0	496.620	1	1
10	0	2570.590	3336.010	30572.700	8	3
11	0	32.450	12921.570	129540.150	10	10
12	0	3716.410	1448.090	385961.980	8	246
13	0	0	12431.270	198900.250	0	16
14	0	9520.700	5776.320	78197.580	7	2
15	0	14106.660	3742.820	540061.900	32	24
16	0	757.910	11.080	25802.320	34	3
17	0	3.130	4923.240	280803.430	57	57
18	0	27681.450	11171.030	842599.050	26	11
19	0	770.290	3.820	2318.500	3	2
20	0	163.780	1.110	329.780	2	2
21	0	0	6324.450	113840.020	0	18
22	0	725.770	41108.660	292841.020	7	7
23	0	91.140	64.300	712.880	5	4
24	0	2477.340	6928.280	892782.200	8	126
25	0	0	13330.900	226778.270	0	17

ExampleSet (9,841 examples, 0 special attributes, 36 regular attributes)

Figure.2: Ethereum dataset loaded in Rapidminer.

It is important to note that 17 out of the 36 attributes in the dataset contain missing values, which will be addressed in the subsequent data modeling stage. Furthermore, most of these missing values are found in attributes related to ERC20. Hence, it is important to understand the difference between Ether (ETH) and ERC20 tokens.

3.1 Differences between Ether(ETH) and ERC20

The native cryptocurrency of the Ethereum network is called Ether (ETH), which was developed to speed up transactions on the Ethereum blockchain. ERC20, on the other hand, is the Ethereum blockchain-based fungible token standard used to regulate the creation of new tokens on the Ethereum blockchain. Hence ERC20 is the accepted framework for creating Ethereum-based tokens that may be used and implemented in the Ethereum network.

Subsequently, this refined dataset was utilized in the data modeling phase using the RapidMiner platform.

Attribute	Description
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
Number of Created Contracts	Total Number of created contract transactions
Unique Received From Addresses	Total Unique addresses from which account received transaction
Unique Sent To Addresses	Total Unique addresses from which account sent transactions
min value received	Minimum value in Ether ever received
max value received	Maximum value in Ether ever received
avg val received	Average value in Ether ever received
min val sent	Minimum value of Ether ever sent
max val sent	Maximum value of Ether ever sent
avg val sent	Average value of Ether ever sent
total transactions (including tnx to create contrac	Total number of transactions
total Ether sent	Total Ether sent for account address
total ether received	Total Ether received for account address
total ether balance	Total Ether Balance following enacted transactions
Total ERC20 tnxs	Total number of ERC20 token transfer transactions
ERC20 total Ether received	Total ERC20 token received transactions in Ether
ERC20 total ether sent	Total ERC20token sent transactions in Ether
ERC20 total Ether sent contract	Total ERC20 token transfer to other contracts in Ether
ERC20 uniq sent addr	Number of ERC20 token transactions sent to Unique account addresses
ERC20 uniq rec addr	Number of ERC20 token transactions received from Unique addresses
ERC20 uniq rec contract addr	Number of ERC20token transactions received from Unique contract addresses
ERC20 min val rec	Minimum value in Ether received from ERC20 token transactions for account
ERC20 max val rec	Maximum value in Ether received from ERC20 token transactions for account
ERC20 avg val rec	Average value in Ether received from ERC20 token transactions for account
ERC20 min val sent	Minimum value in Ether sent from ERC20 token transactions for account
ERC20 max val sent	Maximum value in Ether sent from ERC20 token transactions for account
ERC20 avg val sent	Average value in Ether sent from ERC20 token transactions for account
ERC20 uniq sent token name	Number of Unique ERC20 tokens transferred
ERC20 uniq rec token name	Number of Unique ERC20 tokens received
ERC20 most sent token type	Most sent token for account via ERC20 transaction
ERC20_most_rec_token_type	Most received token for account via ERC20 transactions

Figure.3: List of attributes.

4. Data Modelling

In order to conduct a comprehensive analysis of the Ethereum fraud detection dataset, three main statistical models will be employed in this study. We are using Rapidminer to deploy these models. The models are Decision Tree, Random Forest, and Gradient Boosted Trees. The Decision Tree algorithm will build a tree-based model that makes predictions by sorting data points into categories based on their features. The Random Forest algorithm, on the other hand, is an ensemble model that combines multiple Decision Trees to produce a more robust and accurate prediction. Finally, the Gradient Boosted algorithm is a machine learning model that uses boosting to improve the prediction accuracy of a weak learner iteratively. The utilization of these three models will allow for a thorough examination of the Ethereum dataset and identify key factors and patterns that distinguish fraudulent transactions from valid ones.

Before moving further with the processes, it is essential to understand the use of various operators common to all the statistical models.
These operators include Retrieve, Set Role, Numerical to Binomial, Select Attributes, Replacing Missing Values, and Filter Examples.
They are standard operators used in most of the processes in Rapidminer and are fundamental for the functionality of the process. Therefore, knowing and effectively utilizing them is crucial for implementing the statistical models to analyze the Ethereum fraud detection dataset.

4.1 Operators

4.1.1 Retrieve

The Retrieve Operator loads a RapidMiner dataset into the Process. In our study, we will load the Ethereum fraud detection dataset in the retrieve operator. Retrieving data this way also provides the metadata of the RapidMiner Object.

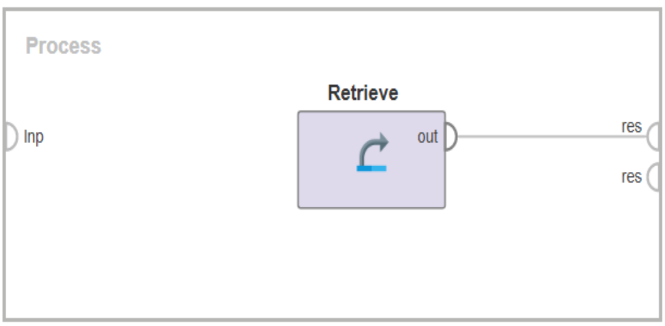


Figure.4: Retrieve operator process flow.

4.1.2 Set Role

The Set Role attribute defines the importance of a specific attribute in the dataset. The default role set is regular, and the other roles are special. An ExampleSet can have many unique Attributes, but each special role can only appear once. If a particular role is assigned to more than one Attribute, all roles will be changed to regular except for the last Attribute. For our dataset, we will set the FLAG attribute to 'label' as FLAG contains the information if a particular transaction is a fraud or not, and we want to test our models to predict fraud.

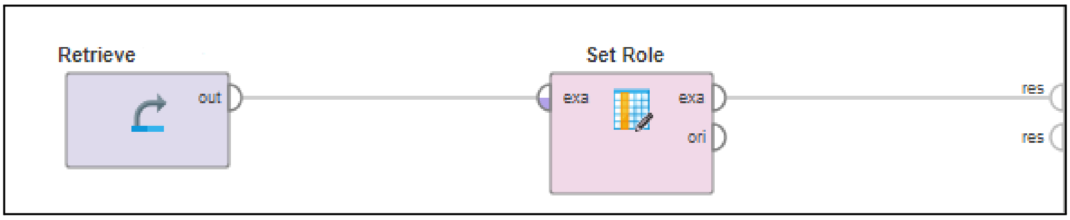


Figure.5: Set Role operator process flow.

4.1.3 Numerical to Binominal

The Numerical to Binominal operator is used to change the numeric values of an attribute to binary values. This operator maps all the numeric values and sets binominal values to them. Binominal attributes have only two possible values - 'true' and 'false.' The binominal output for a particular numeric input is based on the min and max parameters that the users can set.

For the Ethereum dataset, the FLAG attribute values are changed to binominal. Hence, all examples with '0' become 'false,' representing non-fraud transactions, and all '1' become 'true,' which denotes the transactions are fraud.

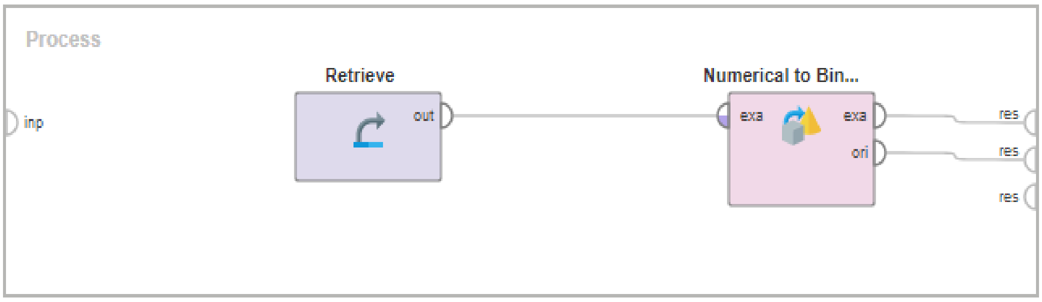


Figure.6: Numerical to Binominal operator process flow.

4.1.4 Select Attributes

The Operator provides different filter types to make Attribute selection easy. Possibilities are, for example, Direct selection of Attributes. Selection by a regular expression or selecting only Attributes without missing values. See the parameter attribute filter type for a detailed description of the different filter types.

The inverted selection parameter reverses the selection. Special Attributes (Attributes with Roles, like id, label, and weight) are, by default, ignored in the selection. They will always remain in the resulting output ExampleSet. The parameter includes special attributes that change this.

Only the selected Attributes are delivered to the output port. The rest is removed from the ExampleSet.

In the Ethereum dataset, we leave out three attributes as they have more than 90% null values as examples.

The attributes that are left out are:

ERC20 max val sent

ERC20 total Ether sent

ERC20 total Ether sent contract

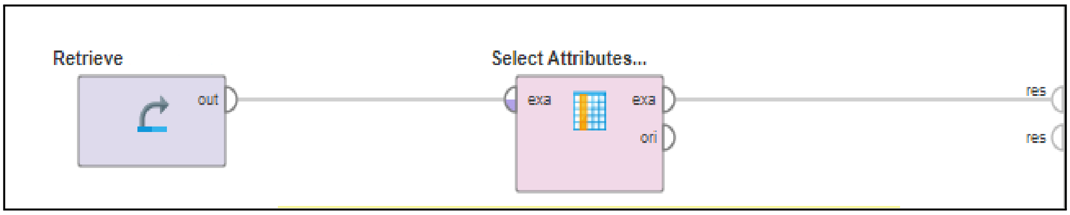


Figure.7: Select Attribute operator process flow

4.1.5 Replace Missing Values

The Replace Missing Values operator can replace the missing values in the example dataset. Missing values are generally replaced by the attribute's minimum, average, and maximum values. Missing values of all attributes or the selected group of attributes can be replaced by the 'attribute filter type' option.

The Replace Missing Values operator replaces missing values of 14 attributes in the Ethereum fraud detection dataset.

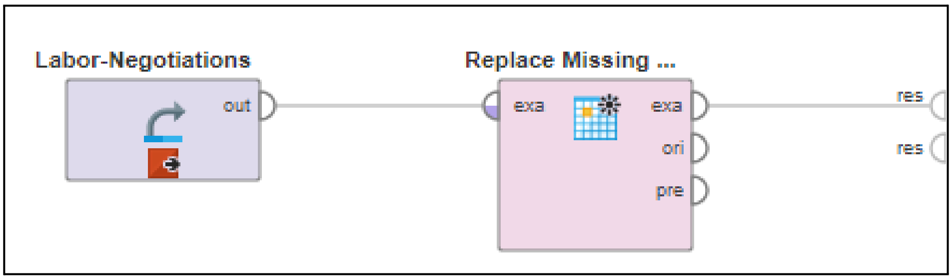


Figure.8: Replace Missing Value Operator process flow.

4.1.6 Standard Operators Process and Output Dataset

The following process depicts the use of the standard operators in the Ethereum fraud detection dataset.

This process will be common for all statistical models and follows the steps given in figure.

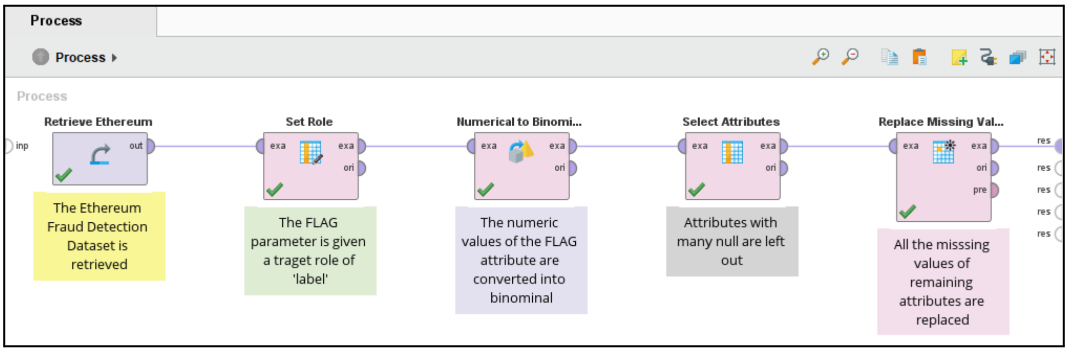
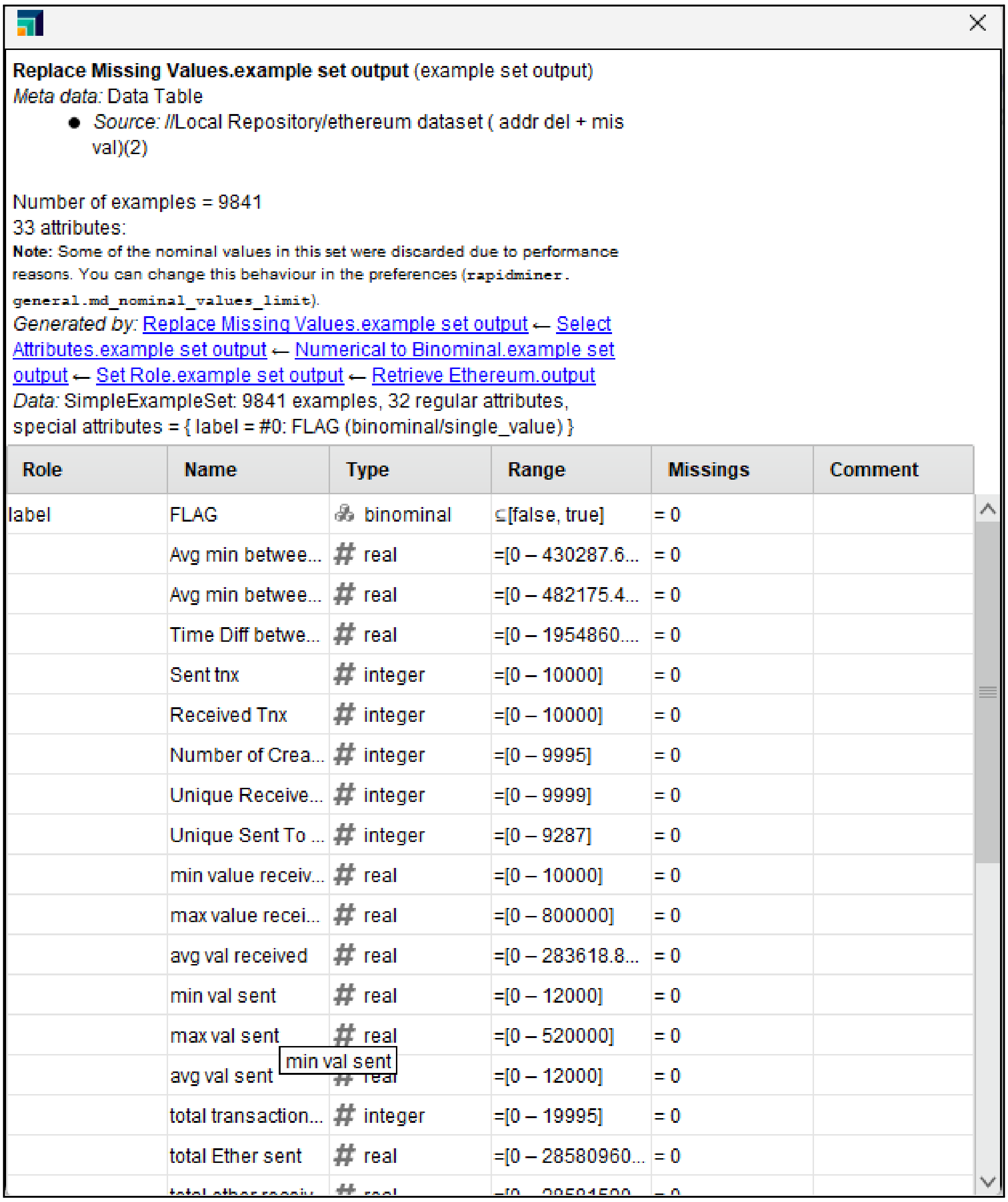


Figure.9: Standard operators process flow in Ethereum dataset.

The Retrieve operator loads the Ethereum dataset in the process. Set Role operator is used to label the FLAG attribute since that's the value we want to predict. The Numerical to Binominal operator changes the values of the FLAG attribute. The Select Attribute then selects the attributes we want to analyze. Finally, the Replace Missing Values attribute handles all the missing values in the Ethereum dataset.



Replace Missing Values.example set output (example set output)
Meta data: Data Table

- *Source:* //Local Repository/ethereum dataset (addr del + mis val)(2)

Number of examples = 9841
33 attributes:
Note: Some of the nominal values in this set were discarded due to performance reasons. You can change this behaviour in the preferences (`rapidminer.general.md_nominal_values_limit`).
Generated by: Replace Missing Values.example set output ← Select Attributes.example set output ← Numerical to Binominal.example set output ← Set Role.example set output ← Retrieve Ethereum.output
Data: SimpleExampleSet: 9841 examples, 32 regular attributes, special attributes = { label = #0: FLAG (binominal/single_value) }

Role	Name	Type	Range	Missings	Comment
label	FLAG	binominal	⊆[false, true]	= 0	
	Avg min betwee...	real	=[0 – 430287.6...	= 0	
	Avg min betwee...	real	=[0 – 482175.4...	= 0	
	Time Diff betwe...	real	=[0 – 1954860....	= 0	
	Sent tnx	integer	=[0 – 10000]	= 0	
	Received Tnx	integer	=[0 – 10000]	= 0	
	Number of Crea...	integer	=[0 – 9995]	= 0	
	Unique Receive...	integer	=[0 – 9999]	= 0	
	Unique Sent To ...	integer	=[0 – 9287]	= 0	
	min value receiv...	real	=[0 – 10000]	= 0	
	max value recei...	real	=[0 – 800000]	= 0	
	avg val received	real	=[0 – 283618.8...	= 0	
	min val sent	real	=[0 – 12000]	= 0	
	max val sent	real	=[0 – 520000]	= 0	
	avg val sent	real	=[0 – 12000]	= 0	
	total transaction...	integer	=[0 – 19995]	= 0	
	total Ether sent	real	=[0 – 28580960...	= 0	

Figure.10: The Ethereum fraud detection dataset containing no missing values.

4.1.7 Cross Validation

It is mainly used to predict how well a model (trained by a specific learning Operator) would function in actual application. The nested operator is the cross-validation operator. A Training subprocess and a Testing subprocess are its two subprocesses. First, a model is trained using the Training subprocess. The Testing subprocess then uses the learned model. During the Testing phase, the model's effectiveness is evaluated.

The input ExampleSet has been divided into k equal-sized subsets. One subset is kept as the test data set out of the k subsets (i.e., input of the Testing subprocess). As a training data set, the remaining k - 1 subsets are employed (i.e., input of the Training subprocess). The cross-validation procedure is then carried out k times, using a single instance of the test data from each k subgroup. Finally, the k outcomes from the k iterations are averaged (or otherwise combined) to create a single estimation. The number of fold parameters can be used to change the value of k.

The Ethereum fraud detection dataset is cross-validated for several statistical models: Decision Tree, Random Forest, and Gradient Boosted.

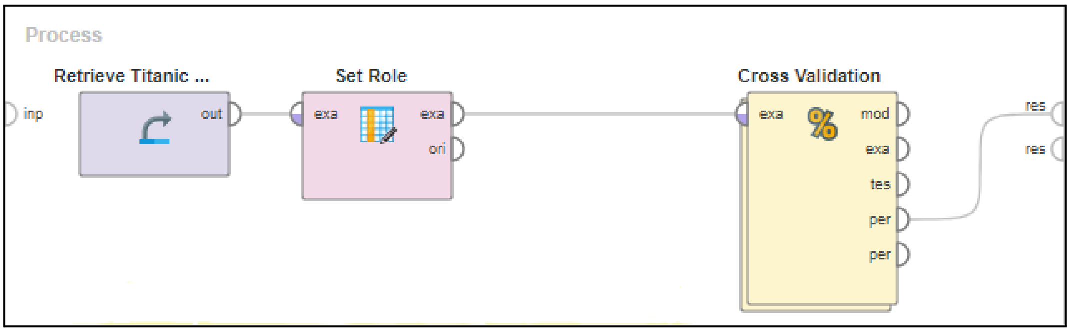


Figure.11: Cross Validation operator process flow

4.2 Processes

4.2.1 Decision Tree

In RapidMiner, Decision Tree is a machine-learning algorithm for predictive modeling and classification tasks. It works by building a tree-like structure of decisions and their corresponding outcomes based on the input features of the data, which follows a top-down approach. Then, the algorithm splits the data into branches based on the feature that provides the most information gain. The process continues until a stopping criterion is met, such as a maximum tree depth. The resulting tree can then predict new data points by following the decision trail from the root node to a leaf node(top-down). In

RapidMiner, the Decision Tree algorithm is used as an operator that can be easily implemented into a predictive modeling and analysis process.

We implemented the decision tree operator on our dataset to understand how frauds can be detected on the Ethereum blockchain.

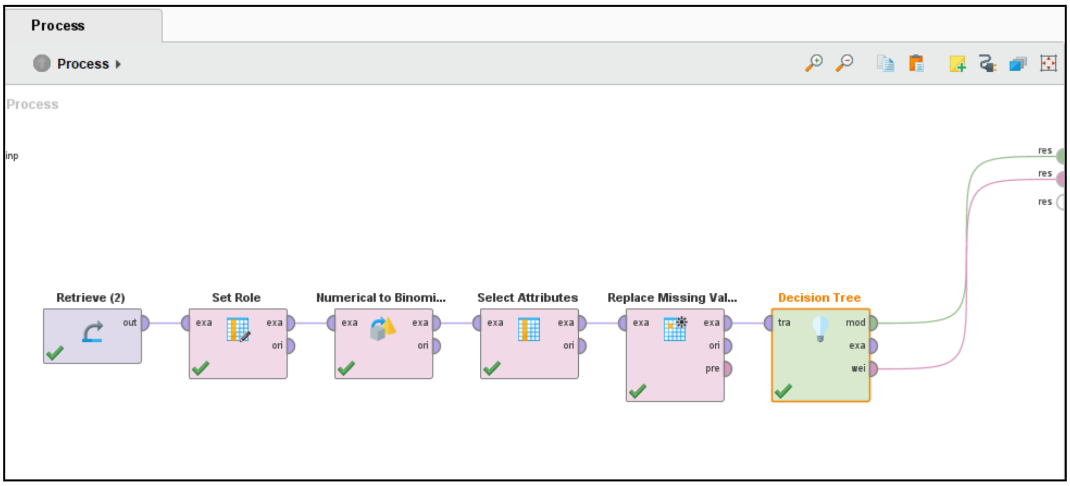


Figure.12: Decision tree process flow.

The Decision Tree Cross Validation operator can perform cross-validation on the Ethereum fraud detection dataset. The operator splits the dataset into a specified number of folds and uses each as a validation set while training the Decision Tree model on the remaining data. The model's performance is evaluated on the validation set, and the process is repeated for each fold. The average performance across all folds provides an estimate of the generalization ability of the Decision Tree model for the Ethereum fraud detection dataset and helps to identify overfitting.

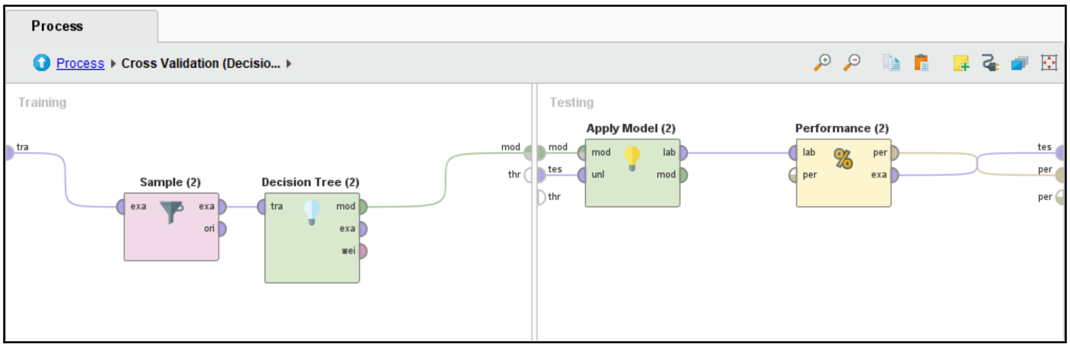


Figure.13: Decision tree model in cross validation operator.

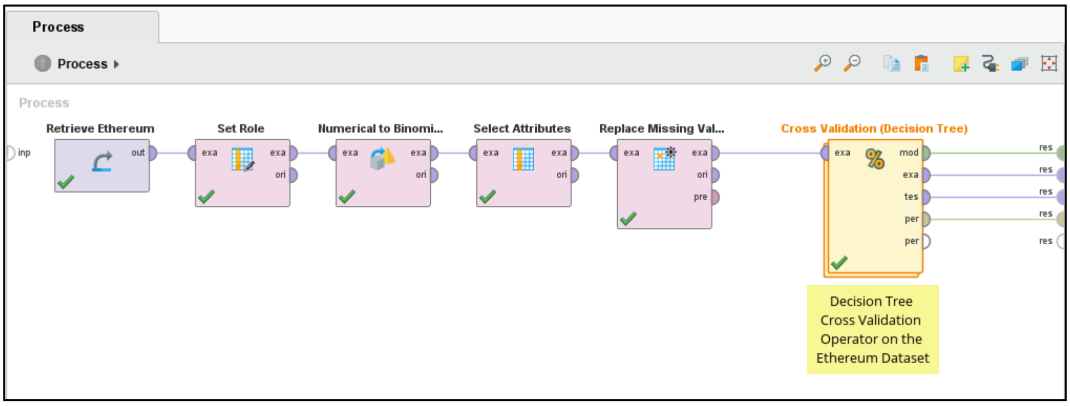


Figure.14: Decision tree cross validation model for Ethereum fraud detection dataset.

4.2.2 Random Forest

As an ensemble learning system for classification, regression, prediction, and other tasks, random forests build many decision trees during training and then predict the class that is the mean predictor of all the individual trees or the mode of the categories.

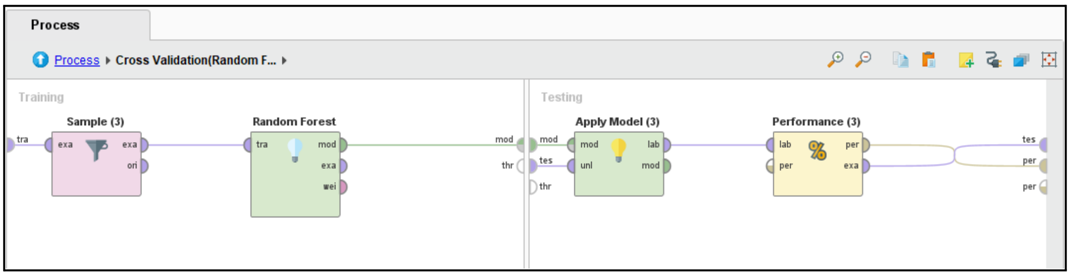


Figure.15. Random forest operator for cross validation.

In RapidMiner, the Random Forest operator can be used to evaluate the performance of a model using cross-validation. The operator splits the data into a specified number of folds and uses each as a testing set while training the model on the remaining data. The performance of the model is then evaluated on the testing set, and the process is repeated for each fold.

In this study, The dataset has been divided into two parts: the training dataset used by the model to adapt to the dataset and the testing dataset, which validates and justifies the validity of the trained model. The dataset split is of the ratio 4:1 or 80–20% for training and testing, respectively.

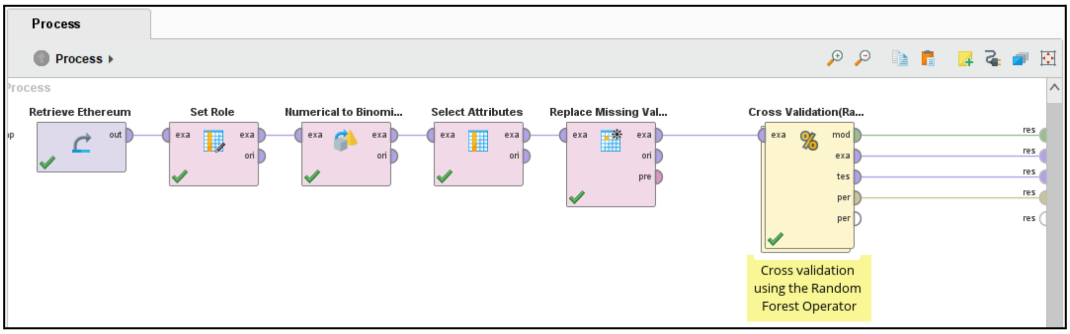


Figure.16. Random forest cross validation process flow for Ethereum fraud detection dataset.

4.2.3 Gradient Boosted Trees

Gradient Boosted Trees is the fastest computing tree-based algorithm than other algorithms, because it progresses vertically. Since it is a tree-based algorithm, it has a root and leaf that can grow vertically or horizontally. It can be easily understood from the given diagram. Here consider that we are at the left leaf; instead of going to the rightmost leaf, Gradient Boosted Trees expand from the leaf with significant loss vertically, i.e., growing leaf-wise. In contrast, other algorithms grow horizontally or level-wise. This model is beneficial if we are computing results on a large dataset; otherwise, it may overfit the small dataset. The primary advantage of this algorithm is that it is very lightweight. It consumes extremely low memory to compute thousands of rows by providing accurate results.

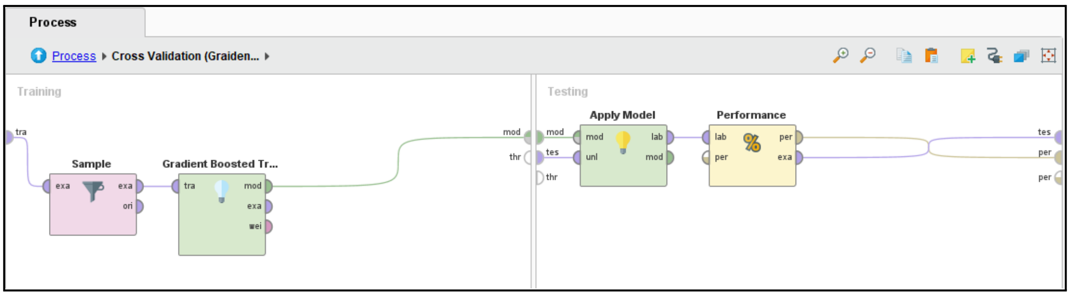


Figure.17: Gradient boosted trees operator for cross validation.

Similar to Decision Trees and Random Forests, Gradient Boosted Trees can also be used to test the fraud detection accuracy on the Ethereum blockchain. Gradient Boosted Trees, when used with the cross-validation operator, also follow the same working principle of training and testing the dataset in several folds for predictive modeling.

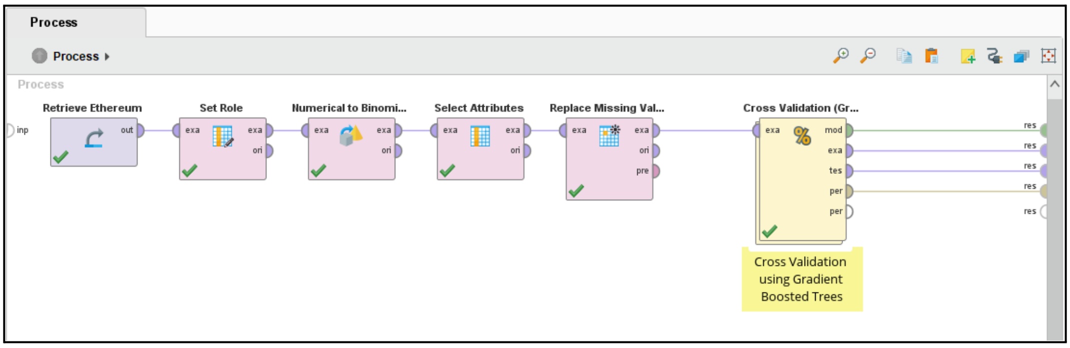


Figure.18: Gradient boosted trees cross validation process flow.

5. Results

The decision tree, random forest, and gradient-boosted trees models were used to analyze the Ethereum fraud detection dataset. The models were compared based on performance vectors such as accuracy, precision, and area under the curve(AUC).

The results obtained are as follows:

Before comparing the results of various performance vectors, it is crucial first to review the decision trees of different models and the level of importance assigned to each attribute within the decision trees. The decision tree is a crucial component of predictive modeling, and understanding it can provide valuable insights into how each algorithm is making its predictions.

5.1 Decision Tree

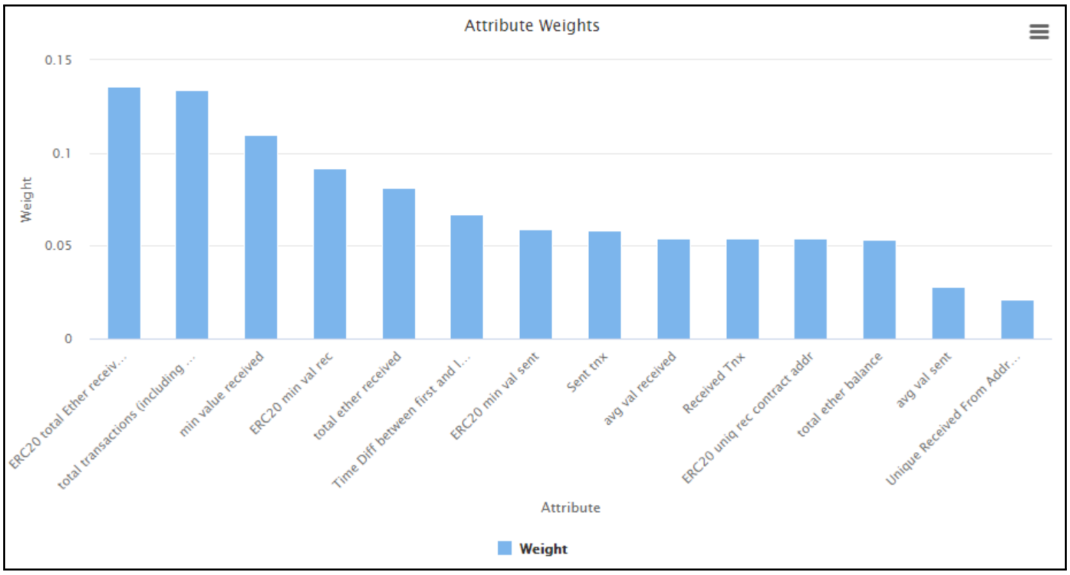


Figure.19: Decision tree attribute weights

The decision tree operator gives attribute weights. For example, the ECR20 total Ether received attribute has the highest weight, and the Unique received from address attribute has the least weight.

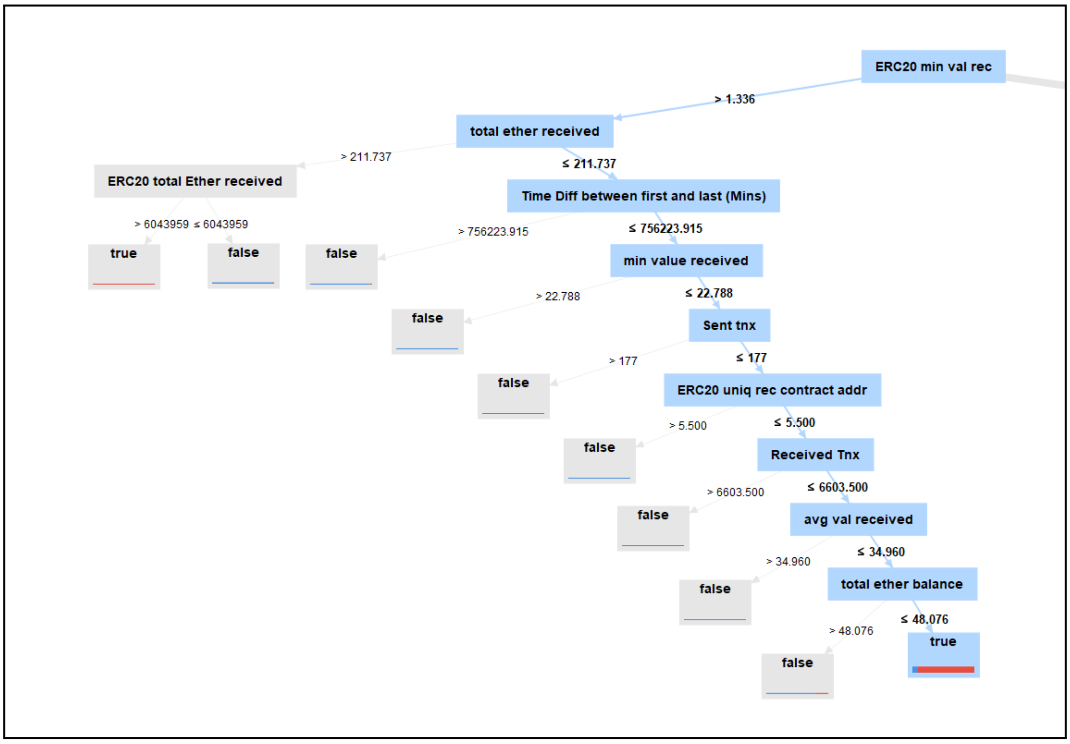


Figure.20: Decision tree

The representation in figure depicts the hierarchical flow of decision-making based on various attributes of the data instances. The tree structure provides a visual representation of how different attributes interact, leading to a specific outcome, in this case, the prediction of whether a transaction is fraudulent or legitimate. This figure provides an in-depth view of the working of the decision tree algorithm, showing the relationship between each attribute and the decision made at the terminal nodes of the tree.

Tree

```
ERC20 min val rec > 1.336
|   total ether received > 211.737
|   |   ERC20 total Ether received > 6043959: true {false=0, true=5}
|   |   ERC20 total Ether received ≤ 6043959: false {false=137, true=4}
|   total ether received ≤ 211.737
|   |   Time Diff between first and last (Mins) > 756223.915: false {false=25, true=1}
|   |   Time Diff between first and last (Mins) ≤ 756223.915
|   |   |   min value received > 22.788: false {false=6, true=0}
|   |   |   min value received ≤ 22.788
|   |   |   |   Sent tnx > 177: false {false=4, true=0}
|   |   |   |   Sent tnx ≤ 177
|   |   |   |   |   ERC20 uniq rec contract addr > 5.500: false {false=2, true=0}
|   |   |   |   |   ERC20 uniq rec contract addr ≤ 5.500
|   |   |   |   |   |   Received Tnx > 6603.500: false {false=2, true=0}
|   |   |   |   |   |   Received Tnx ≤ 6603.500
|   |   |   |   |   |   |   avg val received > 34.960: false {false=2, true=0}
|   |   |   |   |   |   |   avg val received ≤ 34.960
|   |   |   |   |   |   |   |   total ether balance > 48.076: false {false=36, true=10}
|   |   |   |   |   |   |   |   total ether balance ≤ 48.076: true {false=168, true=1703}
ERC20 min val rec ≤ 1.336
|   total transactions (including tnx to create contract > 0.500
|   |   ERC20 min val sent > 871250: true {false=0, true=2}
|   |   ERC20 min val sent ≤ 871250
|   |   |   min value received > 8166.508: true {false=1, true=6}
|   |   |   min value received ≤ 8166.508
|   |   |   |   avg val sent > 4688.153: true {false=1, true=2}
|   |   |   |   avg val sent ≤ 4688.153
|   |   |   |   |   Unique Received From Addresses > 12.500
|   |   |   |   |   |   total transactions (including tnx to create contract > 24.500: false {false=553, true=166}
|   |   |   |   |   |   total transactions (including tnx to create contract ≤ 24.500: true {false=2, true=66}
|   |   |   |   |   Unique Received From Addresses ≤ 12.500: false {false=6723, true=206}
|   total transactions (including tnx to create contract ≤ 0.500: true {false=0, true=8}
```

Figure.21: Decision tree description.

The figure above represents the decision tree levels in the decision tree cross-validation operator for the Ethereum fraud detection dataset.

5.2 Random Forest

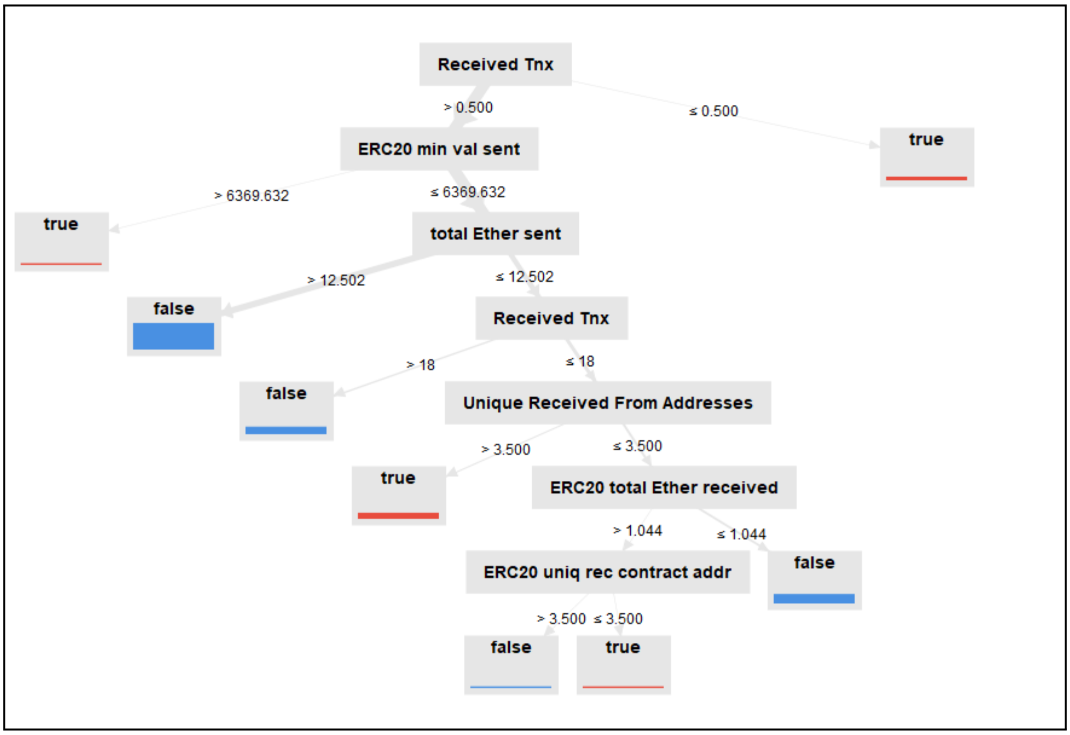


Figure.22: Random Forest Cross Validation decision tree

One of the decision trees generated utilizing the Random Forest operator is depicted in Figure. In this representation, the True values signify fraudulent transactions, while False values indicate legitimate transactions. The decision tree shows the series of decisions and the associated criteria used by the Random Forest algorithm to classify transactions as either fraudulent or legitimate.

5.3 Gradient Boosted Trees

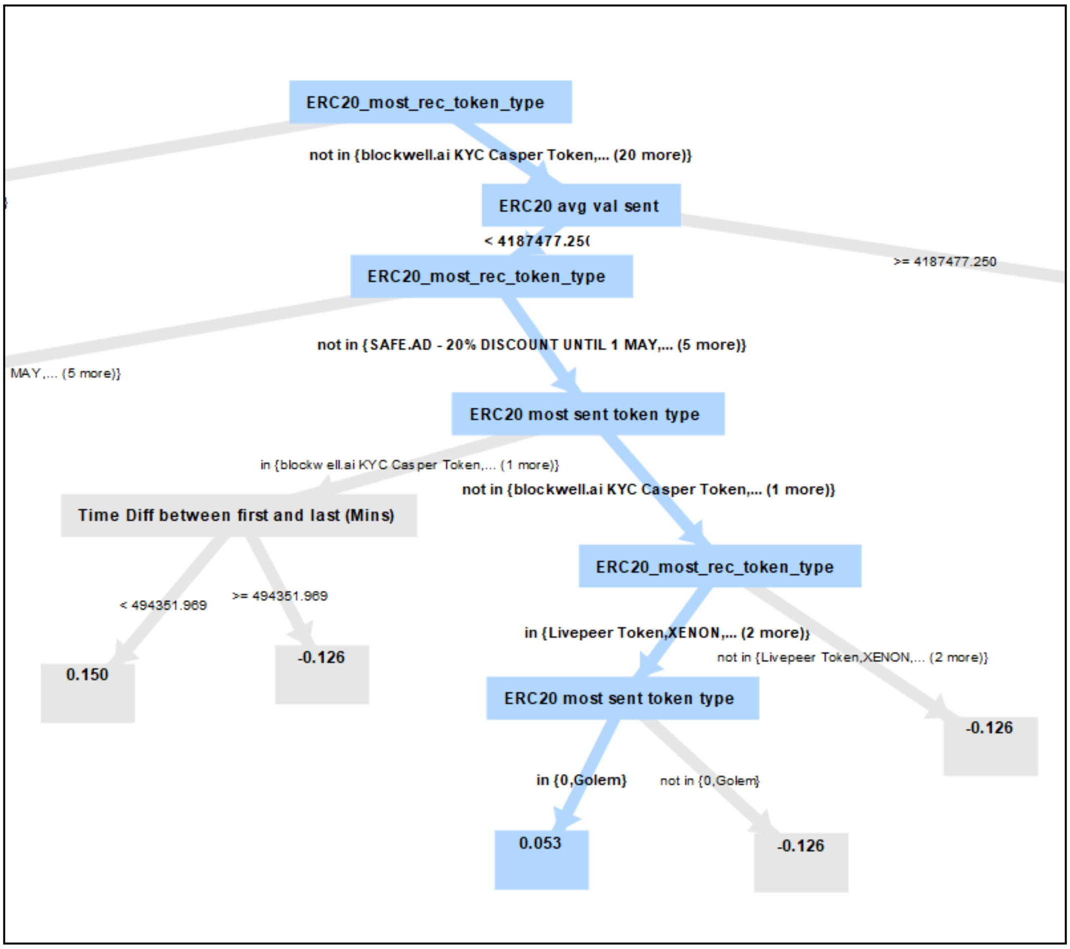


Figure.23: Gradient Boosted Trees Cross Validation decision tree

Gradient boosted trees represented in the figure have several advantages over traditional decision tree models, including improved accuracy, better handling of imbalanced data, and the ability to deal with missing values and noisy data. This is because gradient-boosted trees combine weak models, such as decision trees, and improve them through each iteration.

5.4 Accuracy

5.4.1 Decision tree

accuracy: 89.48% +/- 2.82% (micro average: 89.48%)

	true false	true true	class precision
pred. false	7155	528	93.13%
pred. true	507	1651	76.51%
class recall	93.38%	75.77%	

Figure.24: Accuracy values for Decision Tree Cross Validation Model.

5.4.2 Random forest

accuracy: 92.95% +/- 1.14% (micro average: 92.95%)

	true false	true true	class precision
pred. false	7360	392	94.94%
pred. true	302	1787	85.54%
class recall	96.06%	82.01%	

Figure.25: Accuracy values for Random Forest Cross Validation Model.

5.4.3 Gradient Boosted Trees

accuracy: 96.70% +/- 0.88% (micro average: 96.70%)

	true false	true true	class precision
pred. false	7591	254	96.76%
pred. true	71	1925	96.44%
class recall	99.07%	88.34%	

Figure.26: Accuracy values for Gradient Boosted Cross Validation Model.

Model	Accuracy
Decision Tree	89.48% +/- 2.82%
Random Forest	92.95% +/- 1.14%
Gradient Boosted Trees	96.70% +/- 0.88%

Table.1: Accuracy comparison of various models.

Therefore, the accuracy of

Gradient Boosted Trees > Random Forest > Decision Tree.

5.5 Precision

Similar to accuracy, the precision values of the described models are represented in the table.

Model	Accuracy
Decision Tree	78.91% +/- 10.05%
Random Forest	86.09% +/- 5.71%
Gradient Boosted Trees	96.53% +/- 2.96%

Table.2: Precision comparison of various models.

Out of all the statistical models used in the Ethereum fraud detection data set, gradient boosted trees have the highest precision.

Gradient Boosted Trees > Random Forest > Decision Tree

It is crucial to understand that accuracy refers to the degree to which measurement, calculation, or prediction results match the actual values. Precision, on the other hand, refers to the degree of consistency and reproducibility of the results.

5.6 Area Under the Curve(AUC)

AUC (Area Under the Curve) metric is used to evaluate the performance of binary classification models such as 'fraud' and 'non-fraud' instances in our Ethereum fraud detection dataset. The AUC metric measures the ability of a model to distinguish between positive and negative classes by computing the area under the receiver operating characteristic (ROC) curve. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for different classification thresholds. The AUC value ranges from 0 to 1, with a value of 1 indicating perfect discrimination and a value of 0.5 indicating random classification.

5.6.1 Decision Tree AUC

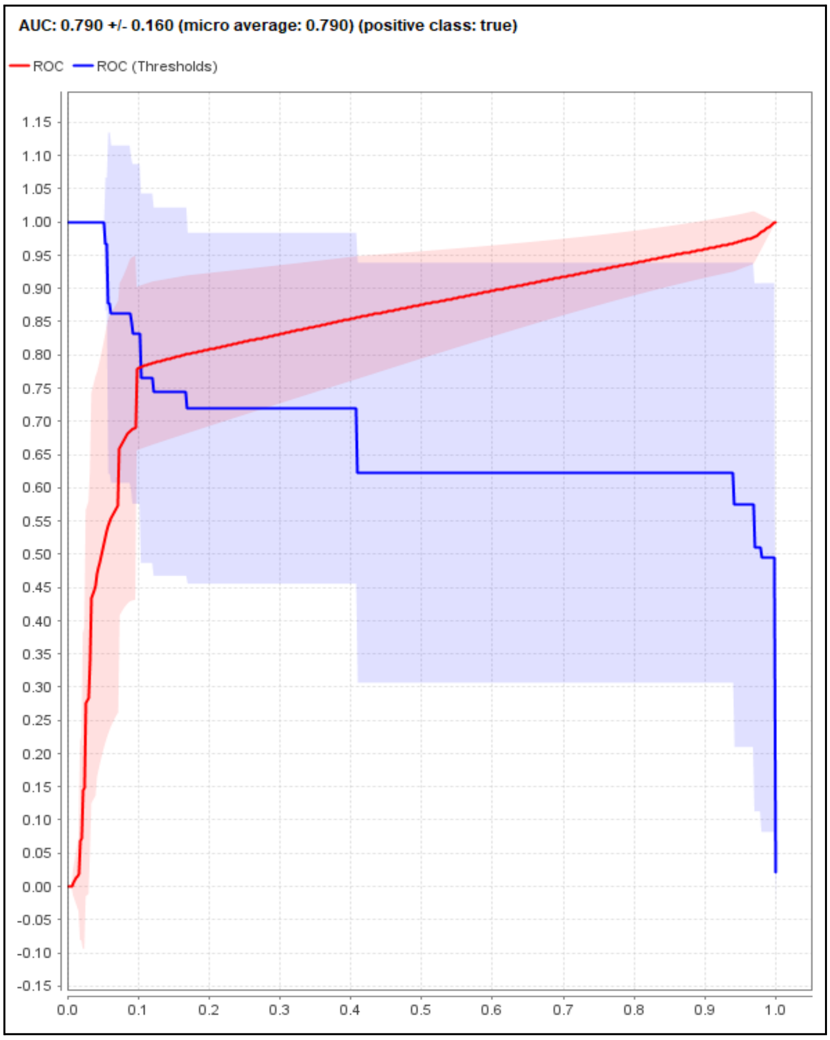


Figure.27: AUC for Decision Tree Cross Validation Model.

The AUC value for decision tree cross validation model is 0.79 +/- 0.16 this suggests that the decision tree operator is not very efficient in classification of examples in fraud and non-fraud categories.

5.6.2 Random Forest AUC

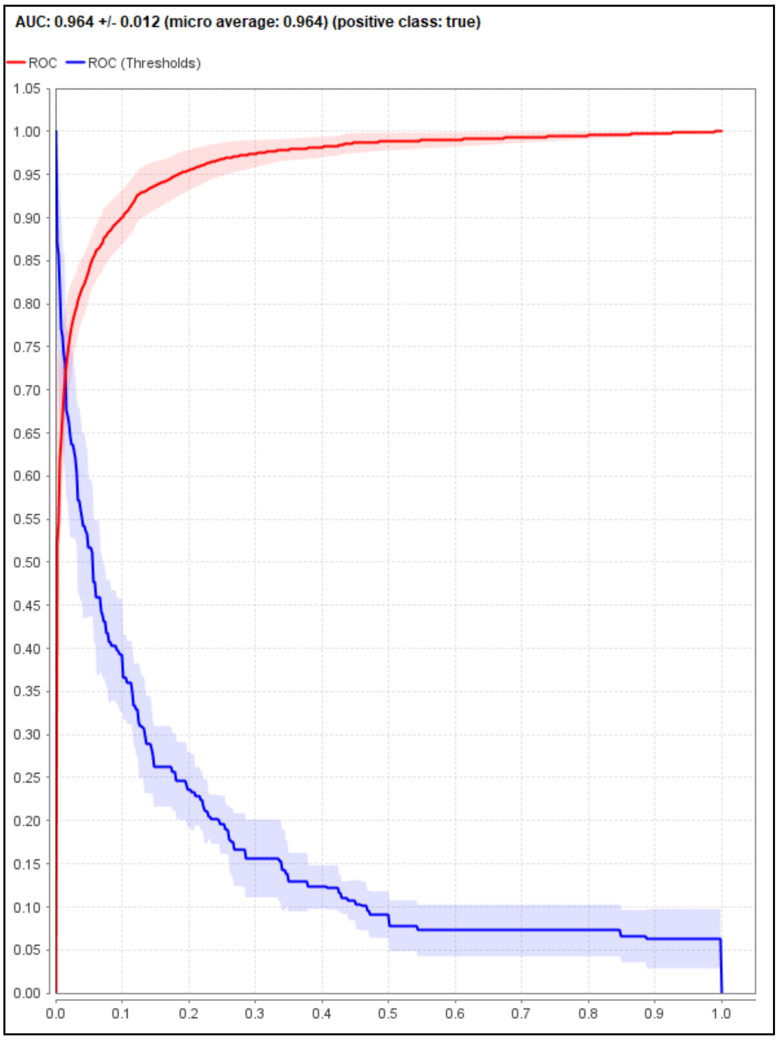


Figure.28: AUC for Random Forest Cross Validation Model.

The AUC value for the random forest cross validation model is 0.964 +/- 0.012 which suggests that model is able to clearly differentiate between fraud and non-fraud examples.

5.6.3 Gradient Boosted Trees AUC

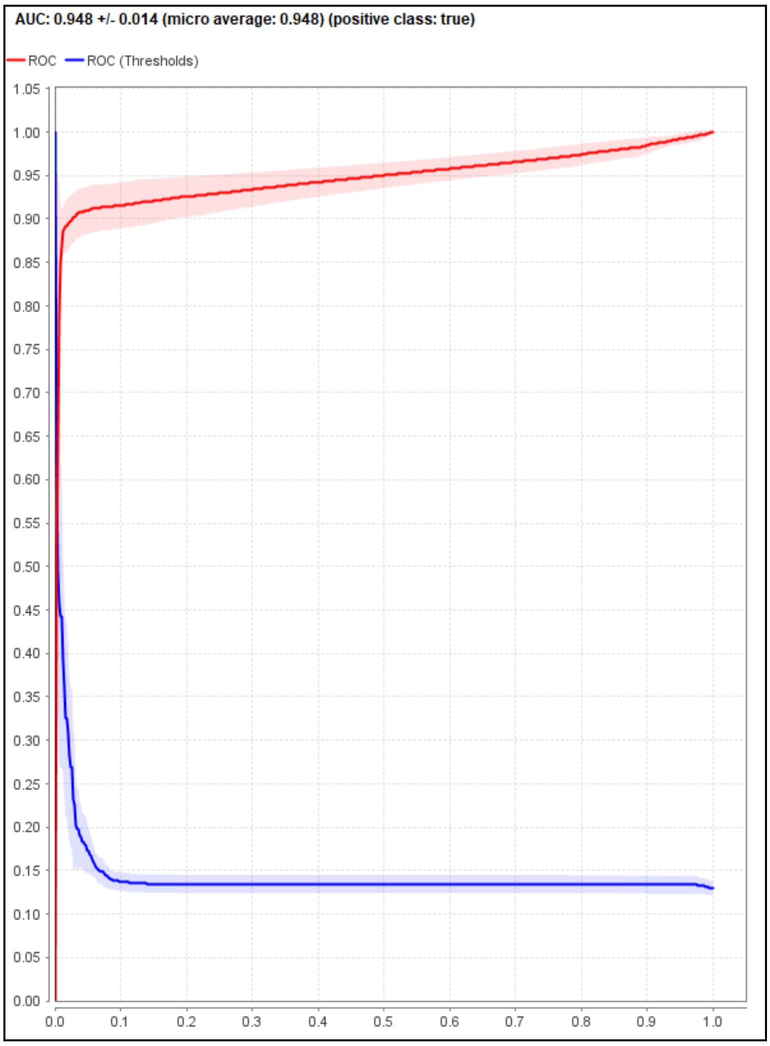


Figure.29: AUC for Gradient Boosted Cross Validation Model.

The AUC value of the gradient boosted cross validation model is 0.948 +/- 0.014 is very close to 1 which represents perfect discrimination on fraud and non-fraud examples in the Ethereum fraud detection dataset.

Model	AUC
Decision Tree	0.79 +/- 0.16
Random Forest	0.964 +/- 0.012
Gradient Boosted Trees	0.94 +/- 0.014

Table.3: AUC comparison of various models.

As a result, it is clear from the figures that the AUC of Random forest is the most accurate.

6. Conclusion

The study aimed to detect Ethereum fraud transactions in Rapidminer using the algorithms of the decision tree, random forest, and gradient-boosted trees. These models were compared on various performance vectors, such as accuracy and precision, and are under the curve(AUC). The dataset used for training and testing the models contained 9841 examples and 32 attributes.

Most of the algorithms performed well after the preprocessing & the selected attributes. However, from the obtained results, it is empirical that the gradient-boosted trees and random forest had the highest accuracy of 96.70% and 92.95%, respectively. The precision values also followed the same trend as accuracy. The random forest was the best-performing model when measured for AUC with 0.964 precision, followed by gradient-boosted trees and decision trees. The decision tree model was the worst-performing one of the three models across all the performance parameters. Our study concludes that the gradient-boosted decision tree is the best-performing statistical model to detect fraud on the Ethereum blockchain. Lastly, as data analysts, we recommend that the Ethereum risk management team implement gradient-boosted trees on the Ethereum blockchain to detect fraudulent transactions.

7. Future Work

For future research, it is possible to detect patterns in Ethereum network transactions. It is also possible to build upon this research and implement more statistical models such as logistic regression, neural networks, and others and compare their results on various performance parameters for the Ethereum fraud detection dataset.

8. References

1. Aziz, R.M. *et al.* (2022) "LGBM: A machine learning approach for ethereum fraud detection," *International Journal of Information Technology*, 14(7), pp. 3321–3331. Available at: https://doi.org/10.1007/s41870-022-00864-6.

2. Jung, E. *et al.* (2019) "Data Mining-based Ethereum Fraud Detection," *2019 IEEE International Conference on Blockchain (Blockchain)* [Preprint]. Available at: https://doi.org/10.1109/blockchain.2019.00042.

3. Liu, L. *et al.* (2022) "Blockchain-enabled fraud discovery through abnormal smart contract detection on ethereum," *Future Generation Computer Systems*, 128, pp. 158–166. Available at: https://doi.org/10.1016/j.future.2021.08.023.

4. SoFi (2022) *What is erc20? A guide to the ethereum token standard*, *SoFi*. SoFi. Available at: https://www.sofi.com/learn/content/what-is-erc20-token-standard/ (Accessed: February 6, 2023).

5. Reiff, N. (2023) *What are ERC-20 tokens on the Ethereum Network?*, *Investopedia*. Investopedia. Available at: https://www.investopedia.com/news/what-erc20-and-what-does-it-mean-ethereum/ (Accessed: February 6, 2023).

6. GmbH, R.M. (no date) *Retrieve (rapidminer studio core)*, *Retrieve - RapidMiner Documentation*. Available at: https://docs.rapidminer.com/9.9/studio/operators/data_access/retrieve.html (Accessed: February 6, 2023).