BT4012 Fraud Analytics
Group Project Final Report

Name	Matriculation Number
DAI YANG	A0219907W
LUO HAONAN	A0262714H
XU LUOQI	A0221556J
YANG LEYUAN	A0211189H
ZHU XIAOXIAO	A0221538J

Link to Github Repo: https://github.com/cocoy02/BT4012_AML-Fraud-Detection

1. Problem Statement

In the wake of rapid technological advancements, the banking industry has undergone significant digital transformation. This evolution has notably led to the decentralisation of financial transactions, transitioning traditional currencies into digital forms. With this shift, however, comes an increased vulnerability to digital attacks such as fraud, anomalies, and privacy breaches. The financial sector, in particular, faces escalating challenges due to the rise in transaction volumes, leading to substantial economic losses globally. Anomalies in this context represent any suspicious network activity that deviates from the norm, critical in cybersecurity and digital financial exchange for detecting fraud and network intrusions.

The emergence of blockchain technology introduced a secure, decentralized ledger system, addressing many security issues of centralized systems. However, this technology, especially within the Ethereum network, still faces challenges from malicious activities like phishing scams. These scams pose a significant threat to the security and integrity of digital transactions, leading to considerable economic damages.

This research aims to address these challenges by leveraging advanced machine learning techniques, specifically XGBoost and Random Forest, to detect and classify phishing accounts within the Ethereum blockchain. This approach represents a novel application of machine learning in analyzing blockchain transaction data, with a focus on enhancing the security and integrity of digital currencies. Our work contributes to the field by integrating these models directly into the blockchain, using a smart contract system for real-time classification of transactions as malicious or legitimate, and implementing attacker models to bolster the system's resilience against blockchain-specific threats.

2. Research Gap

The goal of this research is to detect and classify phishing accounts within the Ethereum network, leveraging the interconnected nature of accounts and transactions represented as nodes and edges, respectively. By transforming this issue into a node classification problem, we aim to harness the inherent structural patterns of the transaction network to pinpoint suspicious or malicious nodes.

Given this extensive transactional network, the primary challenge is to:
- Develop a robust model that can effectively utilize both node and edge attributes to classify an Ethereum account (node) as phishing or non-phishing.
- Evaluate the model's ability to generalize across the vast number of unlabeled nodes in the dataset.
- Understand the inherent patterns or behaviors exhibited by phishing nodes within the Ethereum network to detect newer phishing schemes or variants possibly preemptively.

Expected Outcomes: A successful solution to this problem will not only detect existing phishing nodes with high accuracy but will also serve as a deterrent, making Ethereum's ecosystem more secure and trustworthy. Further, insights drawn from this analysis could potentially guide and inform anti-phishing strategies across other blockchain platforms.

3. Dataset Description

The original data source can be found on Kaggle: https://www.kaggle.com/datasets/xblock/ethereum-phishing-transaction-network. And our version with preprocessing and fundamental feature engineering can be found in Google Drive folder: https://drive.google.com/drive/u/0/folders/1KEAwhCMOPDwB5L9PJGSmPUlHHVQun0jJ.

The dataset has been meticulously curated from the Ethereum blockchain, with phishing nodes sourced from the Etherscan labeled cloud. Using a second-order Breadth-First Search (BFS) strategy centered around these phishing nodes.This approach ensures a comprehensive capture of transactions directly or indirectly linked to these phishing activities. We will apply preprocessing techniques to derive relevant features from these data to craft our models.

Key Features	Definitions and Attributes
Nodes	Representing individual Ethereum accounts, the dataset encompasses 2,973,489 nodes, of which 1,165 are labeled as phishing nodes. Each node denoting an Ethereum address, possesses an 'isp' attribute, marking its status as a phishing node. Phishing Nodes ('ISP' = 1): 1,165, explicitly labeled as involved in phishing activities. Non-Phishing Nodes ('ISP' = 0): Comprise the bulk of the dataset, labeled as non-fraudulent or status unknown. **Key Attributes:**

	Node Identifier: Unique Ethereum address identifying each node. ISP (Indicator of Suspicious Activity): Binary attribute serving as the target variable. It indicates whether a node is associated with phishing (1) or not (0), forming the basis for our classification model.
Edges	Symbolizing transactions between Ethereum accounts, a total of 13,551,303 edges have been documented. **Key Attributes:** Transaction Amount: Represents the monetary value of each transaction between nodes. Timestamp: The exact moment each transaction was executed, which can be critical for analyzing transaction patterns over time.

Table 1 Data Description

4. Data Preprocessing

4.1. Subgraph creation

Among the 2,973,489 nodes, only 1,165 nodes are identified as fraudulent, leaving the remaining 2,972,324 nodes classified as normal or unknown. Our decision is to concentrate on the fraudulent nodes. Specifically, we aim to extract a subgraph by examining the last incoming nodes and next outgoing nodes associated with all identified fraudulent nodes. Within the subgraph, there are 29,461 nodes connected by 1,243,550 edges. Among these nodes, 1,165 are classified as fraudulent, while the remaining nodes are labeled as normal or unknown.

4.2. Node Characteristics

From this set of nodes, we have derived the following fundamental features, which flag anomalies, identifies fraud hubs, and enhances fraud mitigation strategies within networks:

Features	Definition & Importance
In_degree	Incoming degrees of the nodes. Identify nodes' transactional relationships.
Out_degree	Outgoing degrees of the nodes. Identify nodes' transactional relationships.
PageRank	PageRank Score of the nodes. Evaluates importance based on transactional connections.
Weights_out	Sum of outgoing transactions amount of the nodes. Sum of transaction amounts for analysis.
Weights_in	Sum of incoming transactions amount of the nodes. Sum of transaction amounts for analysis.
Num_out	Total number of outgoing transactions of the nodes.
Num_in	Total number of incoming transactions of the nodes.
clustering_coefficient	Measures connections among a node's neighbors. Detects clusters or unusual node connections.
closeness_centrality	Proximity of a node to all others. Identify potential anomalies.
betweenness_centrality	Node's influence on information flow. Detects nodes crucial for transaction flow.
eigenvector_centrality	Importance based on connected high-scoring nodes. Highlights influential nodes in the network.

Table 2 Nodes Features generated from the graph

Utilizing the mentioned fundamental features, we applied a Random Forest model to evaluate the quality and efficacy of these features. Unfortunately, the results indicate suboptimal performance in detecting fraudulent nodes. Consequently, a decision has been made to undertake additional feature engineering to enhance the model's capabilities.

4.3. Network Characteristics

Network characteristics offer insights into structure, anomalies, and efficiency. They identify irregular patterns, aid decision-making, and inform security measures. By analyzing connectivity and behavior, they enable predictive modeling and are crucial for managing, securing, and detecting fraud within networks.

Features	Definition & Importance
num_connected_components	Identifies isolated clusters, potentially highlighting segregated fraudulent activities.

network_density	Indicates overall interconnectivity, revealing potential areas of dense or sparse interactions that might signal fraud.
avg_path_length	Average distance between nodes reveals network efficiency and connections.

Table 3 Network Features

4.4. Component Characteristics

By analyzing component-specific features and associations, irregularities, unusual patterns, or segregated clusters in the network can be identified, contributing significantly to fraud detection and enhancing the precision of fraud mitigation strategies.

Features	**Definition & Importance**
component_size component_diameter component_eccentricity component_average_degree component_clustering_coefficient	Provide a nuanced understanding of component structures. Anomalously large diameters, high clustering coefficients in smaller components, or irregularly high average degrees could suggest potential fraudulent activities within those specific segments.

Table 4 Component Features

4.5. Edges Features

Initially, we converted the edges relationship between nodes into tabular format including fromnode, tonode, timestamp, and the amount.

Features	**Definition & Importance**
fromnode	The identifier of the account initiating the transaction
tonode	The identifier of the account receiving the transaction
timestamp	The date and time when the transaction was executed
amount	The numerical value of the transaction

Table 5 Edges Features

5. Exploratory Data Analysis

5.1. Graph

A total of 15,704 strongly connected components, where each node is reachable from any other node via a directed path, and 16 weakly connected components have been identified in the graph comprising 29,461 nodes. Notably, weakly connected components exhibit varying degrees of proximity, with one particularly large component containing 29,411 nodes, while the remaining components display a sparser distribution.

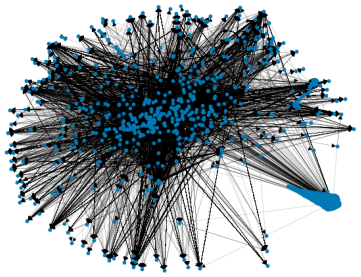

Figure 1 Transaction Graph on 2018-02-12

5.2. Temporal Features

To investigate the distinction in the average number of transactions per account between fraudulent and non-fraudulent activities, we employed a bar plot for visualization. The first graph in Figure 2 reveals that accounts associated with non-fraudulent transactions (denoted as class 0) exhibit a higher average transaction count when compared to accounts linked with fraudulent transactions (denoted as class 1).

Following our analysis of the average number of transactions per account, we delved deeper to examine the average daily transactions per account. This inquiry allows for a more granular understanding of transaction behaviors. The subsequent bar plot in Figure 2 facilitates a comparison that supports our initial findings; we observe a consistent pattern where the average number of daily transactions per account mirrors the overall average transaction frequency for each account category.

We sought to determine if fraudulent transactions exhibit a higher frequency of identical transactions occurring on the same day. Nevertheless, the visualization we generated indicates that there is no significant difference in the frequency of such transactions between the two classes.

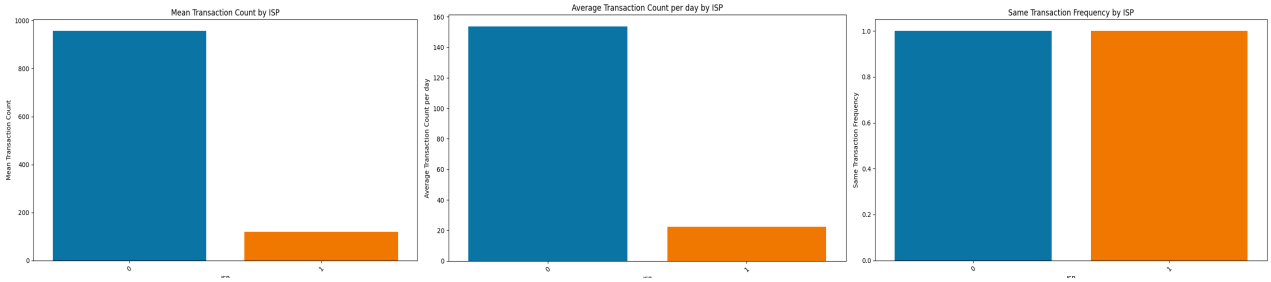

Figure 2 EDA for fraud and non-fraud group: blue represents non-fraud and orange represents fraud

6. Data Balancing

The initial dataset exhibits a substantial imbalance, with a ratio of 25 to 1 between fraud and non-fraud instances, with fraud data significantly outnumbered, as shown in Fig 4. To address this issue, our team employed synthetic data generation techniques for both oversampling and undersampling.

In the oversampling approach, the algorithm equalized the number of fraud instances with non-fraud instances. Following the oversampling, we observed a slight enhancement in the recall of fraud instances, signifying an improvement in fraud detection.

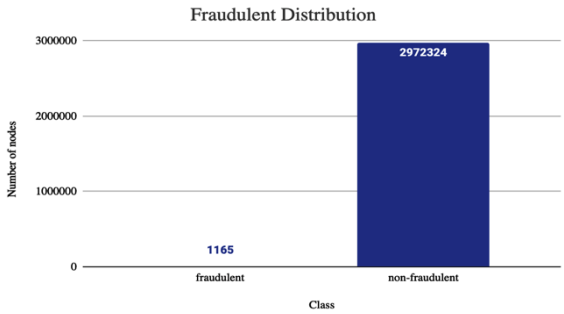

Figure 3 Fraud Class Distribution

Regarding undersampling, three distinct methods were tested, each exhibiting unique characteristics and yielding different outputs:

(1) NearMiss-1: Majority class examples with a minimum average distance to the three closest minority class examples.
(2) NearMiss-2: Majority class examples with a minimum average distance to the three furthest minority class examples.
(3) NearMiss-3: Majority class examples with minimum distance to each minority class example.

These methods will only keep the non-fraud samples as many as the fraud samples. After implementing undersampling, we noticed that the recall for fraud class has significantly improved, while the F1 score is not greatly compromised. Through our testing, we found that NearMiss-3 is the best-performing among the three.

7. Feature Engineering

Feature engineering within the Ethereum network context for phishing account detection involves extracting, creating, and aggregating features that capture transactional behavior, structural patterns, and temporal dynamics. These engineered features are crucial inputs for training machine learning models to identify and classify suspicious nodes effectively. Evaluating the performance of these features in distinguishing between legitimate and phishing accounts will be vital for model effectiveness and accuracy.

7.1. Additional Temporal Features

Creating additional features like timestamp-related details, transaction counts, frequency metrics, and interaction patterns enriches data, enabling nuanced temporal and behavioral analysis. This depth aids in detecting irregularities, identifying potential fraud patterns, and distinguishing suspicious behavior, enhancing the precision of fraud detection within the network.

Features	**Definition & Importance**
Timestamp Related	year, hour-of-day, date, and day_of_week aiding in a more nuanced analysis of the timestamp's impact on fraudulent transactions.

transaction_count	The frequency of transactions for each account.
transaction_per_day	Records the number of identical transactions, exploring the influence of the frequency of identical transactions occurring on the same day.
same_transaction_frequency	The numerical value of the transaction.
active_days	The number of unique days each node interacted on.
business_hours_interactions_count	Business hour, defined as between 9 am and 5 pm. The count of interactions during business hours.
hour-by-hour interaction	The count of interactions that node experienced during each respective hour.

Table 6 Additional Temporal Features

7.2. Generate Aggregated Features

Features Techniques	**Feature name**	**Rationale**
Node Pair Features	in_degree_difference	Identify differences between incoming and outgoing transaction counts, potentially highlighting unusual behavior or patterns associated with phishing activities.
	weighted_difference	Explore differences in transaction weights (if available), revealing potential discrepancies in transaction volumes associated with suspicious accounts.
Time-Dependent Aggregations	Temporal Patterns	Derive rolling metrics (e.g., means, standard deviations) over time for transaction volumes or interactions to capture temporal variations that might indicate sudden bursts of activity associated with phishing attempts.
Ratio Features	In-Out Degree Ratio	Compute the ratio between incoming and outgoing transactions, uncovering anomalies where phishing accounts might exhibit unusual transaction behavior compared to regular accounts.
	Weighted Ratio	Explore ratios between transaction weights to identify abnormalities in the volume or value of transactions associated with specific accounts.
Node Importance	Centrality Measures	Aggregate various centrality measures (e.g., closeness, betweenness, eigenvector centrality) to represent the relative importance of nodes within the network, potentially highlighting nodes with anomalous behavior.
Historical Features	Rolling Averages	Compute rolling averages of transaction-related metrics to capture changes in behavior over time, potentially indicating sudden shifts in account activity.
	Cumulative Interaction Count	Track the cumulative count of transactions for each node, identifying accounts with unusual transaction behavior compared to their historical norms.
Statistical Anomalies	Z-Scores	Compute z-scores for transaction-related metrics like pagerank to identify nodes deviating significantly from the mean, potentially signaling suspicious or fraudulent activities.
Change-Based Features	Pagerank Changes	Track changes in pagerank values over time, highlighting nodes experiencing significant shifts in importance within the network structure.
Cross-Feature Interactions	Pagerank-Centrality Interactions	Examine interactions between pagerank and different centrality measures, potentially revealing combined effects or correlations indicative of anomalous behavior.

Table 7 Aggregated Features

7.3. Feature Scaling and Normalization

Feature scaling and normalization are crucial preprocessing steps to standardize data across various scales, ensuring fair comparisons and consistent model performance.

8. Feature Selection

Feature selection serves to enhance model performance by identifying and utilizing the most influential features. Multiple techniques, such as Exploratory Data Analysis (EDA), correlation analysis, and feature importance via algorithms like Random Forest, aid in this process.

8.1. Using Exploratory Data Analysis

EDA helps identify potential features by comparing distributions between classes. Out of 56 features, 12 exhibit substantial distribution differences between download and non-download classes, indicating their significance. More information can be found in Appendix A.

8.2. Using Feature vs Target Correlation

Correlation analysis measures the relationship between each feature and the target variable. Utilizing Pearson correlation in Figure 4, 14 features surpassing a threshold of 0.006 are chosen due to their strong association with the target.

8.3. Using Feature Importance

Employing Random Forest to gauge feature importance aids in selecting the most predictive features. Figure 4 illustrates the 14 selected features based on their importance scores derived from the model training process.

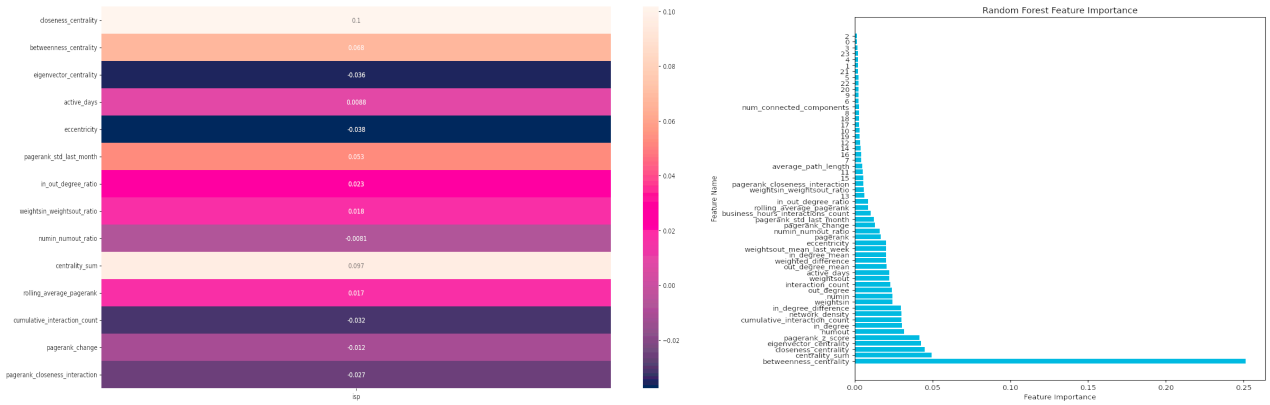

Figure 4 (left) Person Correlation Analysis between selected features and target; (right) Feature importance from XGBoost

9. Machine Learning Models

9.1. K Nearest Neighbours (KNN)

The KNN model can function in both supervised and unsupervised modes, with its objective being the identification of centers with the shortest distance to partition the data into distinct groups. In unsupervised scenarios, the algorithm clusters data points according to their inherent characteristics, making it applicable when data lacks labels, a common occurrence in real-life situations. In this project, both supervised and unsupervised approaches were tested on extracted features. However, it was observed that supervised learning yielded superior performance compared to unsupervised learning. Fine-tuning was conducted for parameters like leaf_size, p, and n_neighbors.

Hyperparameter	**Definition & Importance**
leaf_size: *10*	Number of points the algorithm uses when searching for neighbors.
p: *1*	Power parameters for Minkowzi distance. When p = 1 it actually represents Manhaten Distance.
n_neighbors: *5*	Number of neighbors been considered when making new predictions for the incoming data.

Table 8 KNN Model

9.2. Random Forest Classifier

Random Forest is a powerful and versatile machine learning algorithm widely favored for its robustness and effectiveness across various domains. Its strength lies in ensemble learning, where multiple decision trees collectively contribute to more accurate predictions. By aggregating the outputs of numerous trees, it minimizes overfitting and increases generalizability, making it resilient to noisy data. Additionally, Random Forest can handle large datasets efficiently, accommodate diverse types of features, and provide insights into feature importance, aiding in feature selection. Its ability to manage unbalanced data and maintain high accuracy, even in the presence of missing values or outliers, makes it a go-to choice for tasks like fraud detection, classification, and regression, ensuring reliable and robust predictive performance.

Hyperparameter	**Definition & Importance**

n_estimators: *300*	Number of trees in the forest. Tuned to balance between model complexity and performance.
max_depth: *20*	Limits the depth of each tree. Capture more intricate patterns in the data, potentially leading to overfitting.
min_samples_split: *2*	Minimum number of samples required at an internal node before splitting. Control the tree's complexity by avoiding splits in cases where the sample size is small.
min_samples_leaf: *1*	Minimum number of samples required at leaf node. Reduces overfitting by preventing the creation of nodes that have too few samples.

Table 9 Random Forest Model

Random Forest, sensitive to varying feature magnitudes, can exhibit stark performance disparities with and without scaling. Unnormalized data might skew the tree-building process, causing it to disproportionately favor features with larger scales. Scaling mitigates this, enabling the algorithm to assess features more evenly, enhancing model convergence and accuracy. Hence, adopting scaling practices is pivotal for optimizing Random Forest's predictive capabilities and ensuring robustness across diverse datasets as shown in Table__

9.3. EXtreme Gradient Boosting (XGBoost)

XGBoost is an advanced implementation of gradient-boosting decision trees. It is designed for speed, user-friendliness, and is notably efficient for large datasets (Kelley, 2023). Given these attributes, XGBoost is an apt choice for analyzing our Ethereum dataset. Furthermore, XGBoost includes L1 and L2 regularization, which can help in preventing overfitting. Since our data is very imbalanced, the model may easily overfit the majority non-fraudulent class. Moreover, XGBoost provides insights into feature importance, which can be crucial in understanding which features are most predictive of the minority fraudulent class. This capability is instrumental in helping us to enhance the model's performance and interpretability.

Hyperparameter	**Definition & Importance**
n_estimators: 56	# of boosting rounds or trees to build (balance model performance & complexity); We have a moderate number of trees (56) in our model. Having too many trees can lead to overfitting, especially in imbalanced datasets.
max_depth: 17	max depth of trees (too high may lead to overfitting)
learning_rate: 0.036	the step size at which the optimizer makes updates to the weights (A smaller learning rate (e.g 0.036) requires more boosting rounds but can lead to a more robust model.)
min_child_weight: 1	min sum of weights required in a child node; Setting to 1, which allows XGBoost to create children nodes even if they have a small number of instances, which is critical for detecting the minority fraudulent cases.
gamma: 0.72	regularization parameter; To make the algorithm more conservative, less likely to overfit to the majority non-fraudulent class.
subsample: 0.58	the fraction of observations used for each tree (balance model performance & complexity)
colsample_bytree: 0.54	the fraction of features to be randomly sampled for each tree (balance model performance & complexity)

Table 10 XGBoost Model

9.4. Support Vector Machine (SVM)

Firstly, SVM excels in high-dimensional spaces. In our scenario, even though fraudulent data points are few, they could be distributed across numerous features, making this attribute particularly helpful. Additionally, SVM prioritizes decision boundaries between classes, focusing on the most challenging points to classify. This attribute is particularly advantageous for our imbalanced dataset, where distinguishing the minority class can be difficult. Furthermore, SVM is recognized for its robustness, making it highly effective in scenarios where the data is not linearly separable.

Hyperparameter	**Definition & Importance**
C: 10	regularization parameter; This helps in controlling the trade-off between achieving a low bias and a high variance.
kernel: rbf	the kernel type to be used in the algorithm. (control the decision boundary); rbf kernel can help in capturing intricate patterns that might not be apparent with a linear kernel. This can lead to better identification of fraudulent cases.

Table 11 SVM Model

9.5. Ensemble Learning

To enhance the overall predictive performance, experiments were conducted using ensemble learning. The stacking algorithm was employed to aggregate individual predictions from base models, capturing diverse perspectives of underlying patterns. Subsequently, a meta-model combines the outcomes from base models to make the final prediction. This approach offers the potential for the model to generalize effectively to unseen data while simultaneously mitigating overfitting. In this project, the base model contains KNN, SVM, XGBoost, while the meta model is another Random Forest model.

9.6. Graph Models

Graph models, notably Graph Convolutional Networks (GCNs) and Graph Neural Networks (GNNs), are an optimal choice for Ethereum datasets due to their adeptness in harnessing interconnected data structures. In Ethereum, transactions, smart contracts, and addresses form a complex web of relationships, where graph models shine in capturing these intricate connections. By leveraging the network's structure, graph models like GCNs efficiently identify fraudulent patterns and anomalies, offering a robust framework for comprehensive analysis and fraud detection within the Ethereum blockchain.

Hyperparameter	**Definition & Importance**
class_weights: *[1,20]*	Assigns weight to classes, giving more importance to minority class (20) over majority class (1). Adjusts model focus, prioritizing rare class for better learning in imbalanced datasets.
epoch:45	Epochs determine model convergence and training duration, affecting learning depth and potential overfitting.

Table 12 Graph Model

10. Experiment Results

10.1. Model Performance

10.1.1. Non-graph Models

Models	Accuracy	Class 1 Recall	**Class 1 F1 Score**	ROC AUC
Logistic Regression	0.96	0.00	**0.00**	0.50
K Nearest Neighbours	0.96	0.04	**0.07**	0.51
Random Forest without scaling	0.99	0.72	**0.81**	0.86
Random Forest with scaling	0.99	0.70	**0.80**	0.85
XGBoost	0.99	0.27	**0.41**	0.95
Support Vector Machine	0.99	0.10	**0.17**	0.86
Ensemble Learning	0.99	0.45	**0.54**	0.72

Table 13 Without data balancing

Given the significant imbalance in our dataset, which contains only 1.45% fraudulent data and 98.55% non-fraudulent data. It's evident from the table that most models exhibit high accuracy (around 99%), but demonstrate very low recall for class 1, with the exception of the random forest model. This suggests that these models predominantly predict instances as non-fraudulent. In such a scenario, relying solely on accuracy as the evaluation metric would not accurately reflect the models' true performance. Therefore, we have decided to select a model that demonstrates robust and balanced performance, not just in terms of accuracy, but also considering recall for class 1, the f1 score for class 1, and the roc_auc score. Additionally, we explored various tuning methods to address the issue of imbalanced data, including oversampling and undersampling techniques. These approaches will be discussed in more detail in the subsequent sections.

Models	Accuracy	Class 1 Recall	**Class 1 F1 Score**	ROC_AUC
Logistic Regression	0.93	0.08	**0.08**	0.52

K Nearest Neighbours	0.89	0.22	**0.13**	0.57
Random Forest	0.99	0.76	**0.82**	0.88
XGBoost	0.98	0.62	**0.54**	0.97
Support Vector Machine	0.93	0.72	**0.23**	0.92
Ensemble Learning	0.99	0.17	**0.26**	0.58

Table 14 Results with oversampling

Models	Accuracy	Class 1 Recall	**Class 1 F1 Score**	ROC_AUC
Logistic Regression	0.93	0.08	**0.08**	0.52
K Nearest Neighbours	0.66	0.60	**0.13**	0.63
Random Forest	0.94	0.91	**0.55**	0.93
XGBoost	0.96	0.56	**0.28**	0.93
Support Vector Machine	0.81	0.62	**0.09**	0.81
Ensemble Learning	0.86	0.82	**0.15**	0.84

Table 15 Results with undersampling

From the comparisons between oversampling and undersampling, it was observed that oversampling yields better results than undersampling, particularly evident in SVM and Random Forest models based on the enhancement in Class 1 F1 scores. However, a tradeoff between overall accuracy and Class 1 F1 has emerged, potentially causing normal nodes to be misidentified as fraud nodes. Despite this, it may be deemed acceptable, considering that, in real-life scenarios, the losses incurred from fraud nodes outweigh those from normal nodes.

10.1.2. Graph models

The evaluation of various models, particularly the GCN in comparison to Random Forest and XGBoost, reveals significant differences in performance metrics. While Random Forest and XGBoost showcase commendable accuracy levels around 0.95-0.96, the GCN model notably lags behind with an accuracy of merely 0.50. Remarkably, all models, including GCN, exhibit a consistent ROC AUC of 0.50, implying an inability to discern between classes better than random chance. The underperformance of GCN compared to ensemble methods like Random Forest and XGBoost might be attributed to its struggle to capture intricate relationships within the data or inadequate representation of features within the graph structure. Furthermore, the imbalanced nature of the dataset could have hindered GCN's ability to generalize effectively, resulting in suboptimal predictions compared to the ensemble methods that might better handle non-linear relationships and class imbalances in this specific context.

Models	Accuracy	Class 1 Recall	**Class 1 F1 Score**	ROC_AUC
GNN	0.95	0.01	**0.01**	0.50
GNN with class weights	0.95	0.01	**0.01**	0.50
GNN with oversampling	0.96	0.00	**0.00**	0.50
GNN with undersampling	0.96	0.00	**0.00**	0.50
GCN	0.96	0.00	**0.00**	0.50

Table 16 Graph Models' Performance

10.2. Business Usage

The proposed models for detecting and classifying phishing accounts in the Ethereum network has several significant business applications and benefits:

1. **Enhanced Security for Cryptocurrency Exchanges and Wallets:** By integrating this model, cryptocurrency exchanges and wallet services can significantly bolster their security measures. The ability

to accurately identify phishing activities will protect user assets, fostering trust and enhancing the reputation of these services.

2. **Risk Management for Financial Institutions:** Banks and financial institutions increasingly engaging with cryptocurrencies can use this model to assess and mitigate risks associated with Ethereum transactions. This will be crucial in compliance and due diligence processes.Regulatory Compliance and Monitoring:
3. **Anti-Money Laundering for Regulators**: Regulatory bodies can adopt this model to monitor the Ethereum network for illicit activities. This aids in compliance with anti-money laundering (AML) and combating the financing of terrorism (CFT) regulations.
4. **Preventive Measures for Businesses:** Businesses that operate using Ethereum for smart contracts, payments, or other applications can use this model to pre-emptively screen and avoid interactions with malicious entities, safeguarding their operations.

In summary, the model for detecting phishing accounts in the Ethereum network offers crucial benefits, including bolstered security for crypto exchanges and wallets, enhanced risk management for financial institutions, and strengthened compliance with regulatory standards. It serves as a vital tool for businesses in safeguarding their operations, thereby reinforcing the overall stability and trustworthiness of the financial ecosystem.

10.3. Limitation

The effectiveness of features derived from the dataset is contingent upon the availability of a substantial number of data samples for validation. For example, the centrality feature, which necessitates that a fraud data point possesses a significant number of neighbors to be valid. Consequently, this approach may fail to capture fraudulent transactions when novel data points are introduced, resulting in a lag of information.
The lag in information is noteworthy as some of the features within the dataset may require an increased volume of data points to render them suitable for predictive purposes.

Moreover, it's crucial to recognize that our investigation in this project is confined to only a subset of the dataset. This selective exploration may lead to the loss of valuable insights, as certain patterns or trends within the dataset might remain unexplored. Therefore, an expanded analysis encompassing a more comprehensive portion of the dataset is warranted to ensure a more thorough understanding and effective detection of fraudulent activities.

11. Conclusion

The objective of this project is to detect fraudulent transactions within the Ethereum network. Features are extracted from the dataset graph, and multiple machine-learning algorithms are employed to predict fraudulent activities. While overall accuracy remains high across most models, the recall for fraud instances is notably low. Certain models, such as GNN and Neural Networks, tend to categorize all instances as non-fraudulent. Overall, the Random Forest algorithm yields the best performance.

To make this project applicable in real business scenarios, the need to enhance recall for fraudulent transactions is considered. Given the highly imbalanced nature of the data, we explore undersampling techniques, resulting in an augmented recall for fraud instances reaching 0.89. The findings of this project offer valuable insights into fraudulent transactions within the Ethereum network, and we believe that the developed model holds practical benefits and applicability for business use.

12. Appendix

Appendix A EDA results for distributions for two class