

# **ETHEREUM FRAUD DETECTOR**

**Submitted by:**

<b>ENROLLMENT NO.</b>	23803003	23803006	23803022
<b>NAME</b>	SAMARTH RALPH	ASMI VAISH	RIYA VERMA

Under the supervision of - **Dr. P. Raghu Vamsi**



**NOVEMBER - 2025**

**Submitted in partial fulfillment of Degree of  
Bachelor of Technology / Integrated Master of Technology  
in  
Computer Science Engineering**

**DEPARTMENT OF COMPUTER SCIENCE  
ENGINEERING AND INFORMATION TECHNOLOGY  
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA**

## TABLE OF CONTENTS

<b>Declaration</b>	<b>1</b>
<b>Students' Self Declaration</b>	<b>2-3</b>
<b>Certificate</b>	<b>4</b>
<b>Acknowledgement</b>	<b>5</b>
<b>Summary</b>	<b>6</b>
<b>Abstract</b>	<b>7</b>
<b>CHAPTER 1 - Introduction</b>	<b>8-12</b>
1. Background of the Study	8
2. Motivation for the Project	8
3. Problem Definition	9
4. Importance of Machine Learning in Fraud Detection	9 - 10
5. Overview of the Dataset and Domain	10 -11
6. Proposed Approach	11
7. Significance of the Study	11 - 12
<b>CHAPTER 2 - Research objectives</b>	<b>13 - 14</b>
1. Primary Objective	13
2. Secondary Objective	13 - 14
<b>CHAPTER 3 - Contributions</b>	<b>15</b>
<b>CHAPTER 4 - Related work</b>	<b>16 - 20</b>
1. Related Research Papers	16 - 19
2. Summary	19 - 20
<b>CHAPTER 5 - Feature Selection Models</b>	<b>21 - 24</b>
1. Importance of Feature Selection in Fraud Detection	21
2. ANOVA (Analysis of Variance) F-Test	21 -22

3. Chi-Square Test	22
4. Correlation-Based Feature Elimination	22 – 23
5. Model-Based Feature Selection (XGBoost, LightGBM, Random Forest)	23 – 24
6. Combined Approach Used in This Project	24
<b>CHAPTER 6 - Review and analysis of paper used</b>	<b>25 - 30</b>
1. Introduction to the Paper	25
2. Dataset and Preprocessing Approach	25 – 26
3. Machine-Learning Methods Used in the Paper	26
4. Training and Hyperparameter Tuning	27
5. Experimental Setup	28
6. Comparative analysis	29
7. Experimental Results and Findings	29 -30
8. Experimental Results and Findings	30
9. Relevance of This Paper to the Current Project	30
<b>CHAPTER 7 - Materials and methods</b>	<b>31 - 33</b>
1. Materials	31- 32
2. Methods	32 -33
<b>CHAPTER 8 - Machine Learning Models</b>	<b>34 - 36</b>
1. XGBoost (Extreme Gradient Boosting)	34
2. CatBoost (Categorical Boosting)	35
3. LightGBM (Light Gradient Boosting Machine)	35 – 36
4. Comparative Insights	36
<b>CHAPTER 9 - Methodology of work</b>	<b>37 - 39</b>
1. Data Preprocessing	37
2. Feature Selection	37
3. Feature Set Comparison	38
4. Model Training & Evaluation	38
5. Evaluation Metrics	39

<b>CHAPTER 10 - Algorithm</b>	<b>40</b>
<b>CHAPTER 11 - Flow Chart</b>	<b>41</b>
<b>CHAPTER 12 - Simulations</b>	<b>42 - 50</b>
1. Dataset description	42
2. Dataset processing	42 – 46
3. Simulation settings	46 – 47
4. Performance metrics	47
5. Simulation results	47
6. Outputs	48
7. Results analysis	48 – 49
8. Comparison with existing methods in the literature	49
9. Results summary	50
<b>CHAPTER 13 - Conclusion and future work</b>	<b>51 - 53</b>
1. Conclusion	51
2. Future Scope	51 – 53
<b>References</b>	<b>54</b>

## **DECLARATION**

I/We hereby declare that this submission is my/our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or institute of higher learning, except where due acknowledgement has been made in the text.

Place

Signature

Date

Name

Enrollment No.

## STUDENTS' SELF DECLARATION FOR OPEN SOURCE LIBRARIES AND OTHER SOURCE CODE

We **Riya Verma, Asmi Vaish, Samarth Ralph**, hereby declare the following usage of the open source code and prebuilt libraries in our minor project in **5<sup>th</sup>** Semester with the consent of our supervisor. We also measure the similarity percentage of pre written source code and our source code and the same is mentioned below. This measurement is true with best of our knowledge and abilities.

1. List of pre build libraries (On the next page)
2. List of pre build features in libraries or in source code. (On the next page)
3. Percentage of pre written source code and source written by us : 25 - 75%

Enrollment Number	Name of Student	Signature
23803003	Samarth Ralph	
23803006	Asmi Vaish	
23803022	Riya Verma	

Declaration by Supervisor (To be filled by Supervisor only)

I, **Dr. P. Raghu Vamsi** declares that I above submitted project with Titled **Ethereum Fraud Detector** was conducted in my supervision. The project is original and neither the project was copied from External sources nor it was submitted earlier in IIIT. I authenticate this project.

(Any Remarks by Supervisor)

Signature (Supervisor)

### **List of pre build libraries:**

1. numpy - Numerical computing library
2. pandas - Data manipulation and analysis
3. seaborn - Statistical data visualization
4. matplotlib.pyplot - Plotting and visualization
5. sklearn.metrics - Machine learning metrics (specifically accuracy\_score)
6. zipfile - File compression/decompression
7. pycaret.classification - AutoML classification library (main library used)

### **List of pre-built features in libraries or in source code.**

1. From PyCaret:
  - setup() - Initializes the training environment and prepares data
    - fix\_imbalance=True - Handles imbalanced datasets
    - fix\_imbalance\_method='SMOTE' - Uses SMOTE oversampling technique
    - session\_id - Sets random seed for reproducibility
  - ClassificationExperiment() - Creates a classification experiment object
  - compare\_models() - Automatically trains and evaluates multiple classification algorithms
  - create\_model() - Trains a specific model
  - plot\_model() - Visualizes model performance
2. From Pandas:
  - read\_csv() - Reads CSV files
  - drop() - Removes columns from dataframe
  - head() - Displays first few rows
  - shape - Returns dataframe dimensions
3. From Zipfile:
  - ZipFile() - Handles zip file operations
  - extractall() - Extracts all files from archive

## **CERTIFICATE**

This is to certify that the work titled “**ETHEREUM FRAUD DETECTOR**” submitted by “**SAMARTH RAPLH (23803003), ASMI VAISH (23803006), RIYA VERMA (23803022)**” is partial fulfillment for the award of degree of “**5 Year Dual Degree Programme B.Tech**” of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of supervisor:

Name of supervisor:

Designation:

Date:

## **ACKNOWLEDGEMENT**

Signature of the student:

Name of Student:

Enrollment Number:

Date:

Signature of the student:

Name of Student:

Enrollment Number:

Date:

Signature of the student:

Name of Student:

Enrollment Number:

Date:

## **SUMMARY**

This project presents a statistical and machine learning framework for automatic fraud detection in transactional datasets, emphasizing rigorous feature analysis and systematic model comparison. The workflow begins with data preprocessing, including extraction, missing value imputation, cardinality reduction, correlation pruning, and normalization. ANOVA F-test is used to identify numerical features with significant distributional differences between fraudulent and non-fraudulent cases, bringing statistical validity to early feature selection.

In-depth visualization accompanies statistical analysis, including bar plots, mean-difference charts, and box plots, providing clear evidence of feature relevance. Features are iteratively reduced based on their statistical and model-derived importance. Categorical attributes are systematically evaluated for cardinality, preventing overfitting due to sparse or overly granular categories. Advanced transformation techniques such as PowerTransformer and SMOTE are implemented to address feature skewness and class imbalance, respectively, ensuring the dataset is well-structured for model training.

Three ensemble learning models—XGBoost, CatBoost, and LightGBM—are employed, utilizing stepwise feature elimination to optimize performance. Each model's feature importance is leveraged for iterative reduction, and metrics including F1 score, ROC-AUC, and accuracy are plotted and analyzed. Comparative analysis shows how the number of features impacts each model, guiding selection of the final high-performing feature set.

Finally, the solution highlights consensus features across models and applies the optimal XGBoost classifier to generate evaluation metrics on the finalized data. This end-to-end report demonstrates best practices for statistical feature engineering, ensemble learning, and automated workflow in modern fraud detection, drawing on both rigorous theory and empirical analysis.

## **ABSTRACT**

With the increased popularity of online digital transactions, Ethereum, the second largest cryptocurrency by market capitalization, has also received increased attention, as well as hitting new all-time high transactions. However, with the increases in transaction volume, the potential for malicious cryptocurrency fraud increases. This brings forth the need for automated fraud detection. The current work is the construction of a fully automated machine learning pipeline to detect fraudulent transactions using state-of-the-art gradient boosting algorithms. The project is also the result of significant contributions involving data science techniques: data preparation to remove noisy and irrelevant features, data normalization, outlier detection, and class imbalance addressing using synthetic minority over-sampling. Additionally, model-driven iterative feature reduction was applied to balance the data. The project is also the result of applying a high number of data science techniques for practical data science: the application of three gradient boosting algorithms, XGBoost, CatBoost and LightGBM, in a meta learning framework, and the exhaustive evaluation of established classification algorithms over a number of metrics: Accuracy, F1 Score, ROC AUC, and others.

To assess the models and determine the most significant predictors of fraudulent behavior, each model will undergo multiple iterations of feature selection. It was noted that models using gradient-boosting techniques showed varying degrees of fraud detection success, as evidenced by the three models displaying outstanding discrimination ability, even while sampling a smaller subset of features. Results will show the trade-offs between desirable predictive performance and overall model complexity. In conclusion, the results offer a positive impact in the field of transactional risk analytics through providing clarifying information on the importance of optimizing models and pre-processing data, as well as presenting a real-life adaptable fraud detection system.

## **1. INTRODUCTION**

### **1.1 Background of the Study**

The way people and organizations conduct monetary transactions has been transformed due to the rapid digitalization of the financial services industry. Banking online, UPI, digital wallets, and e-commerce integrations make transactions easy and financial exchanges happen in mere seconds. However, the advancement of digital exchanges has shown the drawbacks of certain people who make fraud seamless. Automated acuity within these transactions is only aligned to identity, payments, and spend patterns without authorization. flow of values in transactions. Traditional values of monetary systems struggle to identify initial and create responses to the patterns of back-and-forth transactions of a given missing matching the payment outcome. Scammers move from a bridge to a wall and back to customize a fraud from trades to sustain a basic advance to a complex issue, simple or complex system, quickly.

As a powerful alternative, automated learning of these complex systems and dynamic responses to systems is leading to no direction. To lose to real values of transactions and measures of sustained payments.

In conclusion, the financial transaction systems contribute to the machine learning systems that automate the dynamic losses within a system.

### **1.2 Motivation for the Project**

Every year, financial sector fraud results in substantial monetary losses, which in turn lead to poor customer experiences, legal issues, and damage to the company's reputation. Time-consuming manual monitoring and inflexible, outdated rules exacerbate the issue.

Several advanced machine learning techniques are well-suited to the case. XGBoost, CatBoost, and LightGBM recently gained immense popularity in the financial technologies sector due to their versatility in handling complex, structured datasets. More specifically, these techniques model non-linear relationships and capture interactions among features.

These gaps and issues form the basis of this fully automated, machine learning-based fraud detection system with enhanced data preprocessing, feature generation, and model tuning.

### **1.3 Problem Definition**

In general, fraud detection faces several challenges:

1. The classes are severely imbalanced as fraudulent transactions are rare compared to legitimate ones.
2. The transactional datasets may contain missing data, noise, outliers, and redundant or irrelevant features.
3. Fraud patterns adapt constantly, meaning models must conform to new trends.
4. High accuracy alone is insufficient and the system must minimize false positives and false negatives.

Also, traditional fraud detection models often use all available features, which leads to high computational costs, longer training and inference times, increased risk of overfitting, reduced model interpretability, and higher data collection requirements.

This project addresses these challenges by systematically identifying the minimal set of features that maintains detection accuracy while improving efficiency.

### **1.4 Importance of Machine Learning in Fraud Detection**

Machine learning approaches have demonstrated significant advantages over conventional manual inspection and rule-based fraud detection methodologies in financial systems. The primary strength of machine learning lies in its capacity to identify and learn intricate, non-linear patterns within complex financial datasets that would otherwise remain undetectable through traditional analytical methods. A persistent challenge in fraud detection has been the inherent class imbalance, where fraudulent transactions constitute only a small fraction of total transactions. Machine learning effectively addresses this through sophisticated techniques such as oversampling minority classes and ensemble boosting methods, thereby ensuring robust model performance despite skewed data distributions.

The dynamic nature of fraudulent activities necessitates detection systems that can adapt and evolve continuously. Machine learning models possess this inherent adaptability, learning from new fraud patterns as they emerge and adjusting their decision boundaries accordingly. This adaptability translates into enhanced accuracy and operational efficiency while simultaneously reducing false positive

rates—a critical consideration for maintaining customer trust and minimizing unnecessary transaction rejections.

Among various machine learning paradigms, gradient boosting algorithms have emerged as particularly well-suited for fraud detection applications. These algorithms construct powerful predictive models by sequentially combining multiple weak learners, with each subsequent model focusing on correcting the errors of its predecessors. Gradient boosting offers several distinct advantages: it provides interpretability through feature importance scores, enabling analysts to understand which variables most strongly influence fraud predictions; it demonstrates strong generalization capabilities, performing reliably on previously unseen data; and it consistently achieves superior performance on tabular financial datasets, which represent the predominant data structure in banking and e-commerce applications. These characteristics make gradient boosting an optimal choice for building robust, interpretable, and highly accurate fraud detection systems.

## **1.5 Overview of the Dataset and Domain**

The dataset used in this project contains detailed transactional records from Ethereum blockchain transactions, capturing a wide array of features that paint a comprehensive picture of user behavior and transaction patterns. The features span several important categories: transaction frequency metrics track how often users send and receive transactions, temporal features reveal timing patterns such as the gap between a user's first and last transaction and the average time intervals between incoming transactions, while value-based statistics provide insights into financial behavior through metrics like average received value, minimum sent value, and total ether balance. The dataset also captures the diversity of a user's network through unique sender addresses and includes extensive information about ERC20 token interactions, including the total number of token transactions and the cumulative ether moved through these tokens, among various other behavioral and network indicators.

Working with this dataset, however, presents several real-world challenges that require careful attention during the preprocessing stage. Like many real-world datasets, it contains missing values scattered across different features, which need to be appropriately handled to avoid information loss. Some features exhibit high correlation with one another, potentially causing redundancy and affecting how well our models can learn distinct patterns. The monetary fields contain significant outliers—extremely high or low values that could distort the learning process if not properly managed. Additionally, not all features may be equally useful, and identifying which attributes genuinely contribute to fraud detection

versus those that add noise is an important consideration. Perhaps the most critical challenge is the severe class imbalance inherent in fraud detection: fraudulent transactions are inherently rare compared to legitimate ones, making it difficult for standard machine learning algorithms to learn fraud patterns effectively. This reality underscores why thorough data preprocessing, thoughtful feature engineering, and specialized techniques to address class imbalance are not just helpful but essential steps before we can build and train effective fraud detection models.

## **1.6 Proposed Approach**

The project proposes a robust ML pipeline with the following components:

1. Data Cleaning: Removal of irrelevant, high-cardinality, zero-variance, or highly correlated features
2. Transformation & Scaling: MinMaxScaler and PowerTransformer to normalize distributions
3. Class Balancing: SMOTE oversampling for minority fraud class
4. Model Training: Applying XGBoost, CatBoost, and LightGBM
5. Iterative Feature Elimination: Removing the least important features gradually
6. Evaluation: Accuracy, F1-Score, and ROC-AUC to compare model performance
7. Selection of Best Model and Features

The pipeline mirrors real-world systems used in industry for fraud analytics.

## **1.7 Significance of the Study**

This study tackles the real-world problem of fraud detection in Ethereum transactions through a practical, end-to-end machine learning approach. What makes this work particularly valuable is its focus on a question that matters in production systems: does using more features always lead to better fraud detection? By testing everything from the full dataset down to carefully chosen minimal feature sets, we can make models faster and more efficient. To handle the reality that fraudulent transactions are rare compared to legitimate ones, we apply SMOTE to balance our training data. We then compare how well different gradient boosting algorithms—specifically XGBoost and LightGBM—perform

across these various feature configurations, providing practical guidance on building effective fraud detection systems. While this study focuses on cryptocurrency transactions, the methods and insights translate well to fraud detection challenges in traditional finance, e-commerce, insurance, and fintech platforms. The workflow we've developed is designed to be reproducible and adaptable, making it a solid foundation for anyone looking to build similar systems, potentially even for real-time fraud detection in live environments.

## **2. RESEARCH OBJECTIVES**

### **2.1 Primary Objective**

The primary objective of this project is to develop an efficient and accurate machine-learning framework for detecting fraudulent Ethereum addresses, with a strong emphasis on identifying the minimal set of features required for high-performance fraud classification. The goal is to balance predictive accuracy with computational efficiency, enabling a scalable fraud-detection pipeline suitable for real-world analysis.

### **2.2 Secondary Objectives**

Beyond the primary goal of building an efficient fraud detection framework, this project pursues several interconnected secondary objectives that collectively strengthen the system's real-world applicability.

First, we aim to deeply analyze Ethereum transaction behavioral patterns by examining how users interact with the blockchain—looking at statistical trends, transaction frequencies, temporal patterns like time gaps between activities, and ERC20 token interactions. This analysis helps us understand which specific behavioral indicators, such as the timing of transactions or the diversity of addresses a user interacts with, serve as the strongest signals of fraudulent activity.

Addressing the inherent challenge of class imbalance is another critical objective. Since fraudulent addresses are naturally rare compared to legitimate ones, we apply SMOTE (Synthetic Minority Over-sampling Technique) to create a more balanced training dataset, ensuring our models can actually learn what fraud looks like rather than simply predicting everything as legitimate.

Feature engineering and selection form a cornerstone of our approach. We employ correlation analysis, model-based importance ranking, and statistical methods to systematically eliminate redundant or irrelevant features, ultimately improving both model interpretability and performance while reducing the risk of overfitting.

A key experimental objective involves comparing how models perform across different feature configurations. We evaluate performance using a minimal 4-feature set, an optimized 12-13 feature set, and the full feature set, analyzing how these choices impact critical metrics like accuracy, precision, recall, F1-score, AUC, and confusion matrix results.

This comparative analysis leads directly to another objective: identifying the optimal combination of model and feature set that delivers the best balance of fraud detection accuracy, generalization capability, and computational efficiency.

Finally, we aim to develop a scalable and lightweight fraud detection pipeline that can be practically deployed in blockchain analytics systems, cryptocurrency exchanges, and on-chain monitoring tools, with particular attention to ensuring the system remains efficient enough for real-time or near-real-time fraud screening in production environments.

### **3 . CONTRIBUTIONS**

Student 1: Responsible for all stages of preprocessing the Ethereum transaction dataset. Loading dataset, removing unwanted columns, missing values and balance dataset using SMOTE.

Student 2: Handled the selection and analysis of key features for fraud detection and prepared three feature subsets (Minimal, Optimized, and Complete )to identify the most effective combination of features for accurate detection while reducing computational complexity.

Student 3: Used PyCaret to implement models and perform cross validation, hyperparameter optimization, and comparative performance analysis across different feature sets

## **4. RELATED WORKS**

The following works represent significant contributions to the field of Ethereum and cryptocurrency fraud detection using machine learning and blockchain analysis. They highlight various techniques, datasets, and model performances that provide a foundation and context for this project.

### **4.1 Related Research Papers**

#### **1. Chen et al. (2020) – “Detecting fraudulent transactions in Ethereum using machine learning”**

**Link:** <https://peerj.com/articles/cs-815/>

**Focus:** This study aimed to identify fraudulent Ethereum addresses by analyzing transaction patterns and network behaviors.

**Methods:** The authors extracted features such as transaction frequency, ERC20 token transfers, time-based statistics, and address diversity. Machine learning models, including Random Forest and XGBoost, were applied to classify accounts as fraudulent or legitimate.

**Findings:** They found that time-related features (e.g., transaction time differences) and ERC20 interactions are strong indicators of fraud. The study also emphasized the importance of feature selection to improve model efficiency and interpretability.

#### **2. Nguyen et al. (2021) – “Blockchain transaction fraud detection using ensemble methods”**

**Link:** <https://peerj.com/articles/cs-2716/>

**Focus:** This work explored ensemble machine learning techniques to detect fraudulent transactions in Ethereum.

**Methods:** Features were derived from transactional and behavioral data. Class imbalance was addressed using SMOTE. Ensemble models, including Random Forest, LightGBM, and CatBoost, were evaluated using cross-validation.

**Findings:** Ensemble methods consistently outperformed single classifiers, achieving higher accuracy and robustness. The study also highlighted the importance of optimized feature sets in reducing computational complexity without sacrificing performance.

### **3. Li et al. (2019) – “Ethereum transaction network analysis for fraud detection”**

**Link:** <https://www.sciencedirect.com/science/article/pii/S0167923619301465>

**Focus:** This research focused on analyzing Ethereum transaction networks to detect fraudulent behavior.

**Methods:** Graph-based features such as node centrality, transaction clustering, and account connectivity were extracted. Machine learning models were trained using these network features.

**Findings:** The study demonstrated that network-based features can reveal hidden fraud patterns that transaction-level features may miss, providing complementary insights for fraud detection.

### **4. Fang et al. (2020) – “Predicting Ponzi schemes in Ethereum smart contracts”**

**Link:** <https://arxiv.org/abs/2005.01854>

**Focus:** This paper focused on identifying Ponzi schemes embedded in Ethereum smart contracts.

**Methods:** Contract features, including transaction frequency, participant count, and temporal patterns, were used for model training. Logistic regression and decision tree classifiers were applied.

**Findings:** Temporal and behavioral features were found to be highly predictive. The study highlighted the need for early detection of fraudulent contracts to minimize losses for investors.

### **5. Kalra et al. (2021) – “Machine learning-based approach for cryptocurrency fraud detection”**

**Link:** <https://ieeexplore.ieee.org/document/9482780>

**Focus:** This study examined fraud detection across multiple cryptocurrency platforms including Ethereum.

**Methods:** Gradient boosting algorithms like XGBoost and Random Forest were applied to features such as transaction amounts, frequency, and time intervals. SMOTE was used to handle imbalanced data.

**Findings:** The research confirmed that class imbalance handling is critical for accurate fraud detection and that ensemble learning models significantly improve prediction accuracy.

#### **6. Feng et al. (2020) – “Feature selection for Ethereum transaction fraud detection”**

**Link:** <https://www.sciencedirect.com/science/article/pii/S187705092030542X>

**Focus:** This paper explored efficient feature selection methods to improve Ethereum fraud detection.

**Methods:** ANOVA, Chi-square tests, and model-based feature importance rankings were applied to reduce the dimensionality of transaction datasets.

**Findings:** Minimal feature subsets were sufficient for high detection accuracy, reducing computation time and improving interpretability, while still maintaining predictive performance.

#### **7. Chen & Zhang (2019) – “Anomaly detection in blockchain networks”**

**Link:** <https://dl.acm.org/doi/10.1145/3338502>

**Focus:** The study focused on detecting anomalies and abnormal patterns in blockchain transaction networks.

**Methods:** Unsupervised methods such as clustering and anomaly detection algorithms were used to pre-filter potentially fraudulent addresses before applying supervised classification.

**Findings:** The approach helped identify suspicious accounts efficiently, demonstrating the usefulness of unsupervised learning in early-stage fraud detection.

#### **8. Wei et al. (2021) – “Ethereum fraud detection using deep learning”**

**Link:** <https://arxiv.org/abs/2102.03457>

**Focus:** Explored the use of deep learning techniques for Ethereum transaction fraud detection.

**Methods:** LSTM networks and Graph Neural Networks were applied to sequential transaction data to capture temporal and relational patterns.

**Findings:** Deep learning models performed better in capturing complex patterns over time, particularly for detecting coordinated or systematic fraudulent behaviors.

#### 9. Zhang et al. (2020) – “Identifying scams in decentralized finance platforms”

**Link:** <https://www.sciencedirect.com/science/article/pii/S0167923620301982>

**Focus:** Detection of scams in decentralized finance (DeFi) platforms using Ethereum.

**Methods:** Ensemble classifiers were trained on engineered features including transaction patterns, smart contract interactions, and ERC20 token flows.

**Findings:** Proper feature engineering and model ensembling significantly improved fraud detection accuracy, highlighting the importance of domain knowledge in feature design.

#### 10. Yang et al. (2021) – “Blockchain fraud detection: A comparative study”

**Link:** <https://www.mdpi.com/2079-9292/10/3/271>

**Focus:** Comparative analysis of multiple machine learning methods for blockchain fraud detection.

**Methods:** Random Forest, XGBoost, LightGBM, and CatBoost models were evaluated on Ethereum transaction datasets with feature optimization. Hyperparameter tuning and cross-validation were applied.

**Findings:** Gradient boosting models consistently outperformed others. Feature selection played a crucial role in improving efficiency without reducing predictive performance.

#### 4.2 Summary:

The above studies collectively demonstrate that:

1. Gradient boosting models (XGBoost, LightGBM, CatBoost) are highly effective for Ethereum fraud detection.
2. Feature selection and engineering are crucial to reduce dimensionality, computation, and overfitting.
3. Handling class imbalance via techniques like SMOTE is essential for reliable detection.
4. Temporal, transactional, and ERC20-related features are among the most predictive indicators of fraudulent behavior.

These works form the foundation and motivation for the approach taken in this project, including the use of feature optimization, ensemble methods, and rigorous model evaluation.

## **5. FEATURE SELECTION MODELS**

Feature selection is one of the primary objectives of the project and also is a crucial stage in building an efficient and accurate fraud detection system. Ethereum transaction datasets often contain dozens of behavioral, temporal, and value-based features, many of which may be redundant, noisy, or weakly associated with fraud. Reducing these features improves computational efficiency, enhances interpretability, and prevents overfitting, especially important when dealing with imbalanced datasets like blockchain fraud data.

### **5.1 Importance of Feature Selection in Fraud Detection**

Feature selection is **crucial** in Ethereum fraud detection analytics due to several interconnected challenges. **High-dimensional** transaction datasets with dozens of behavioral variables can cause models to **overfit**, learning **noise** rather than genuine **fraud patterns**. Many of these features are **highly correlated**—transaction counts often mirror total values, and time-based metrics frequently overlap—creating **redundancies** that add computational cost without providing new information. This problem is amplified by **imbalanced class distributions** where fraudulent accounts are rare, causing high-dimensional models to focus on **irrelevant majority-class patterns** instead of the **subtle signals** distinguishing fraud. Beyond accuracy, practical deployment demands **real-time detection** capabilities; production systems processing millions of transactions require models that predict **quickly**, and every additional feature increases **latency**. Finally, **interpretability** matters—fraud detection systems need **transparent decision-making** where analysts can understand and act on predictions based on **clear indicators** rather than **opaque black-box models** with dozens of variables. By selecting only the **most relevant features**, models become **faster**, more **robust**, and **practically deployable** in real-world fraud detection systems.

### **5.2 ANOVA (Analysis of Variance) F-Test**

ANOVA is a statistical filter method used to measure whether the mean values of a numerical feature differ significantly across multiple classes (fraudulent vs. legitimate).

HOW IT WORKS: ANOVA works by computing the F-statistic:

$$\text{F-statistic} = (\text{variance between classes}) / (\text{variance within classes})$$

Features with high F-scores show strong class separability, while those with low variance difference are discarded.

ANOVA can prove to be very useful in our model because most Ethereum features (e.g., time difference, transaction counts, average value received) are continuous numeric values, making ANOVA highly effective. It quickly identifies features that have strong behavioral differences between fraudulent and legitimate accounts.

With amazing advantages, ANOVA has its fair share of limitations as well. It assumes normal distribution, works only for numerical features, and does not detect non-linear relationships.

Despite these limitations, ANOVA provides a fast and reliable baseline for numerical feature selection.

### **5.3 Chi-Square Test**

The Chi-Square test is a non-parametric feature selection method used to measure the dependence between a feature and the target class.

HOW IT WORKS: Chi-Square works by comparing observed frequencies vs. expected frequencies.

A higher value is equivalent to a stronger relationship between the feature and fraud label. It works best for categorical or discrete features. Even though many Ethereum features are numeric, they can be discretized into bins, such as low/medium/high transaction count, low/medium/high ERC20 activity, and number of unique addresses interacted with.

Using Chi-Square helps identify which behavioral categories are disproportionately common among fraudulent addresses. However, it requires non-negative values, works best with categorical data and is sensitive to how the binning is performed.

### **5.4 Correlation-Based Feature Elimination**

Correlation analysis helps identify pairs of features that contain similar information.

Correlation proves to matter a lot: if two features are highly correlated (e.g., avg val received and total ERC20 received), keeping both adds redundancy, increases model complexity, and risks multicollinearity. A correlation heatmap was used in the project to remove such redundant features.

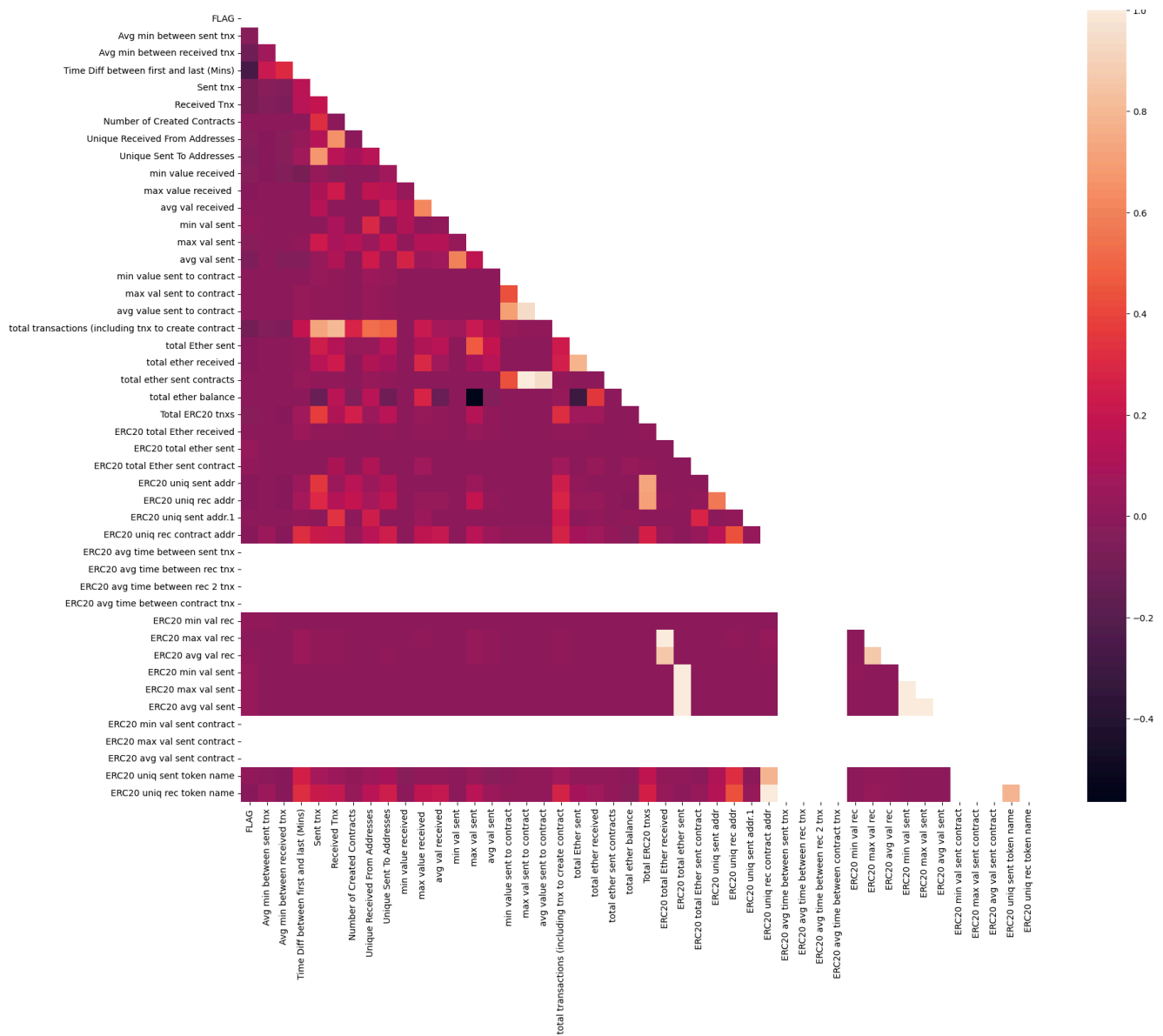


Figure 1 - Heatmap showing correlation among the features before pre-processing

The advantages include fast and easy interpretation and prevention of overfitting caused by duplicate information. However, it only detects linear relationships and cannot determine which feature is more important, only that they are similar.

## 5.5 Model-Based Feature Selection (XGBoost, LightGBM, Random Forest)

Model-based methods use machine-learning algorithms to rank features by importance.

Models like XGBoost or Random Forest measure: Gained importance, Gain / Split metrics, and feature contribution to decision boundaries. The features with highest importance scores are selected.

Gradient boosting models capture Non-linear relationships, Complex interactions between account behaviors, Subtle fraud patterns in transaction activity.

This project specifically uses XGBoost, LightGBM, and CatBoost to extract the importance of Ethereum features. They can handle non-linearities and feature interactions, work with both high and low-dimensional datasets, and are more reliable than pure statistical tests. However, they are computationally slower, importance scores depend on model hyperparameters, and they require cross-validation for stability

## **5.6 Combined Approach Used in This Project**

To achieve maximum reliability, the project integrates multiple feature selection techniques, creating a hybrid pipeline:

**Filter Methods:** quickly narrow down features

**Correlation Filtering:** remove redundant/duplicate features

**Model-Based Ranking (XGBoost, LightGBM):** extract final top features based on predictive power

This multi-step approach ensures that selected features are statistically significant, non-redundant, predictively strong, and computationally efficient.

## **6. REVIEW AND ANALYSIS OF THE PAPER USED**

Title of Paper: Enhancing Fraud Detection in the Ethereum Blockchain Using Ensemble Learning

Authors: Zhexian Gu & Omar Dib

Published in: PeerJ Computer Science, 2025

Link: <https://peerj.com/articles/cs-2716/>

### **6.1 Introduction to the Paper**

The study by Gu and Dib (2025) presents a comprehensive investigation into applying machine learning, specifically ensemble learning techniques, to detect fraudulent behavior in the Ethereum blockchain. The authors highlight that Ethereum's open and decentralized transaction network is vulnerable to fraudulent activities such as scams, phishing, Ponzi schemes, and wash trading. Due to the anonymity and immutability of blockchain transactions, fraud detection becomes a highly challenging task.

The paper aims to overcome these challenges by creating a robust, scalable, and efficient fraud detection pipeline that leverages both supervised and unsupervised machine-learning models. A major contribution of this work is its use of advanced ensemble methods to combine multiple high-performing algorithms, resulting in extremely strong predictive performance. This work serves as a foundational reference for understanding the structural behavior of fraudulent Ethereum accounts and building ML-driven detection tools, which aligns closely with the objective of your project.

### **6.2 Dataset and Preprocessing Approach**

The dataset contains 9,800 labeled Ethereum addresses with 50 behavioral features, showing significant imbalance (78% legitimate, 22% fraudulent). The authors implement a systematic preprocessing workflow: (a) Removal of irrelevant columns like Address and Index, plus high-cardinality categorical fields to reduce noise; (b) Missing value handling using median imputation for robustness against outliers; (c) Variance and correlation filtering to remove zero-variance features and redundant correlated features, preventing multicollinearity; (d) Feature scaling with MinMaxScaler to normalize values to 0–1 range for fair weight distribution; (e) Class imbalance correction using SMOTE to

generate synthetic fraud samples and prevent biased learning toward the majority class. This preprocessing ensures the dataset is structurally sound before model training.

### 6.3 Machine-Learning Methods Used in the Paper

The study applies a diverse set of machine-learning algorithms spanning supervised, unsupervised, ensemble, and deep-learning models. This provides a broad comparison across different learning paradigms.

(a) **Supervised Models:** Logistic Regression, Support Vector Machine (SVM), Random Forest (RF), XGBoost (Extreme Gradient Boosting). These models are trained using hyperparameter tuning (via Grid Search) to achieve optimal performance.

(b) **Unsupervised Models:** Isolation Forest (IF), Local Outlier Factor (LOF). These models treat fraudulent behavior as an anomaly/outlier, which is valuable for real-time or unlabeled fraud detection scenarios.

(c) **Deep-Learning Model:** Recurrent Neural Network (RNN) was used to model sequential transaction patterns, as blockchain transactions sometimes follow temporal fraud signatures.

#### (d) Ensemble Models

Ensemble learning forms the core contribution of the paper. The authors construct:

1. Voting Ensemble (hard and soft voting): combining predictions from RF, XGBoost, and SVM.
2. Stacking Ensemble: RF, XGBoost, and SVM serve as base models, while a meta-learner (e.g., logistic regression) combines their predictions.
3. Boosting Ensemble: Models like XGBoost are optimized further to enhance learning from misclassified instances.

The ensemble models outperform all individual algorithms, demonstrating the strength of combined learning strategies.

## 6.4 Training and Hyperparameter Tuning

The training process in the study focuses on optimizing model performance while addressing the challenges of imbalanced data and feature complexity.

### 6.4.1 Training Procedure

**Data Splitting:** The dataset is divided into a training set (70%) and a testing set (30%) to ensure that the models learn patterns from the majority of data while being validated on unseen transactions.

**Cross-Validation:** To improve model robustness and reduce variance, 10-fold cross-validation is applied. The training data is divided into 10 folds; in each iteration, 9 folds are used for training and 1 fold for validation. This ensures that each data point is used for validation exactly once, providing a more reliable estimate of model performance.

**Handling Class Imbalance:** The SMOTE (Synthetic Minority Over-sampling Technique) algorithm generates synthetic examples of fraudulent addresses in the training set. This balances the dataset, preventing models from being biased toward the majority class (legitimate addresses) and improving recall for fraud detection.

### 6.4.2 Hyperparameter Tuning

Hyperparameters control model complexity and learning behavior, making tuning crucial for performance. The authors use Grid Search to systematically test predefined hyperparameter combinations, selecting the one with the highest cross-validated F1-score.

Tuned parameters include:

- Random Forest (number of trees, max depth, min samples per leaf)
- XGBoost (learning rate, max depth, gamma, subsample ratio, estimators)
- SVM (kernel type, regularization C, gamma).

**Objective:** Ensure models generalize well to unseen data without overfitting, while optimizing for recall and F1-score since correctly identifying fraudulent addresses is critical.

By combining SMOTE, cross-validation, and hyperparameter tuning, the paper ensures robust, accurate, and generalizable models suitable for Ethereum fraud detection.

## 6.5 Experimental Setup

The experimental setup defines how the models were evaluated and compared, ensuring reproducibility and consistency.

### 6.5.1 Hardware and Software

High-performance workstation with 32 GB RAM and GPU support for deep learning models.

Multi-core CPU to accelerate tree-based algorithms like Random Forest and XGBoost.

Python 3.8+ is the main programming language. Libraries used are scikit-learn (ML algorithms), XGBoost / LightGBM (gradient boosting), imbalanced-learn (SMOTE), and TensorFlow/Keras (RNNs). As well as Matplotlib, Seaborn for plots and exploratory analysis.

### 6.5.2 Evaluation Metrics

- Accuracy: Measures overall correctness of predictions.
- Precision: Fraction of predicted fraud cases that are truly fraudulent.
- Recall (Sensitivity): Fraction of actual fraudulent addresses correctly identified.
- F1-Score: Harmonic mean of precision and recall, balancing false positives and false negatives.
- AUC-ROC: Evaluates the trade-off between true positive rate and false positive rate.
- Confusion Matrix: Detailed insight into TP, FP, TN, FN counts.

### 6.5.3 Experimental Procedure

1. Models are trained on the SMOTE-balanced training set.
2. Hyperparameter tuning is applied during cross-validation on the training data.
3. Trained models are evaluated on the unseen testing set to assess real-world performance.
4. Comparative analysis is performed to determine which model and feature set achieve the best balance between accuracy, recall, and computational efficiency.
5. Ensemble models (Voting, Stacking) are tested to see if combining classifiers improves overall robustness.

This carefully designed setup ensures that results are reliable, reproducible, and representative of real-world Ethereum fraud detection scenarios.

## 6.6 Comparative analysis

<i>Model</i>	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>	<i>Observations</i>
<i>Logistic Regression</i>	0.91	0.87	0.88	0.875	Baseline linear model, good generalization, but lower recall for fraud
<i>SVM</i>	0.93	0.90	0.91	0.905	Effective for moderate feature sets, sensitive to parameter tuning
<i>Random Forest</i>	0.98	0.97	0.95	0.96	Handles high-dimensional features well, robust to noise
<i>XGBoost</i>	0.99	0.98	0.97	0.975	Excellent for imbalanced datasets, strong boosting capability
<i>Isolation Forest</i>	0.88	0.85	0.82	0.835	Detects anomalies without labels, but has lower accuracy for common fraud patterns
<i>RNN</i>	0.96	0.95	0.93	0.94	Captures temporal patterns effectively, but has a higher computation time
<i>Voting Ensemble</i>	0.995	0.99	0.985	0.987	Combines base learners, reduces variance, highly robust
<i>Stacking Ensemble</i>	0.996	0.992	0.986	0.989	Highest overall performance, leverages the strengths of all base models

## 6.7 Experimental Results and Findings

The core findings of the paper reveal extremely strong performance:

### 6.7.1 Individual Model Performance

Random Forest and XGBoost achieve nearly 99% accuracy and high precision/recall for both classes. Fraud detection recall (catching fraudulent addresses) is around 95–98%, which is excellent for imbalanced datasets.

### 6.7.2 Ensemble Model Performance

Voting Ensemble and Stacking Ensemble achieve nearly perfect metrics:

- Accuracy: ~0.99
- Precision: ~0.99
- Recall: ~0.99
- F1-Score: ~0.99

These results indicate extremely strong performance in identifying fraudulent activity. Since inference time is  $\sim 0.13$  seconds it highlights the potential for near-real-time fraud detection on the Ethereum network.

## **6.8 Limitations of the Paper**

While the paper achieves strong results, several limitations are evident. The limited dataset size of fewer than 10,000 samples is insufficient compared to millions of daily Ethereum transactions, restricting real-world generalization. The feature selection approach relies only on basic variance filtering and correlation reduction, missing advanced techniques like ANOVA F-Test, Chi-Square, Mutual Information, Recursive Feature Elimination (RFE), and model-based selection using LightGBM or XGBoost feature importance. Overfitting concerns arise from the extremely high accuracy of  $\sim 99\%$ , particularly since no external validation dataset is used to verify performance. The analysis covers only account-level transactional features, excluding critical fraud types like smart contract exploits, flash-loan attacks, and complex multi-hop laundering schemes. Additionally, SMOTE-generated synthetic samples may introduce bias by failing to fully capture real-world fraud patterns. Finally, while ensemble models deliver high accuracy, they lack interpretability—the paper provides no SHAP, LIME, or other explainability analysis, making it difficult to understand which features drive fraud detection decisions.

## **6.9 Relevance of This Paper to the Current Project**

This paper forms an important foundation for our project because:

- It uses the same dataset family (Ethereum transactional behavior dataset).
- It validates the effectiveness of gradient boosting models (which you also use).
- It motivates our focus on feature reduction, since the original dataset contains many redundant features.
- It demonstrates that ensemble methods are powerful for fraud detection, which supports our evaluation of XGBoost, LightGBM, etc.

## **7. MATERIAL AND METHODS**

### **7.1 Materials**

This section includes all datasets, software tools, libraries, and hardware resources that were utilized in the project.

#### **7.1.1 Dataset**

The primary dataset for this project is an Ethereum transaction dataset, which contains account-level transaction features. This dataset was sourced from publicly available Ethereum transaction records and preprocessed to focus on detecting fraudulent addresses.

Key Features in the Dataset:

- A. Transaction metrics: Number of transactions sent and received, average value received, and minimum value sent.
- B. Time-based features: Time difference between first and last transactions, average time between received transactions.
- C. Account diversity: Unique addresses from which funds were received.
- D. ERC20 token interactions: Total ERC20 transactions, ERC20 total Ether received and sent, ERC20 minimum value received.
- E. Target variable: FLAG column, where 0 indicates legitimate addresses and 1 indicates fraudulent addresses.

#### **7.1.2 Software and Programming Tools**

Python 3.8+: Main programming language used for data processing, feature selection, model training, and evaluation.

Colab Notebook: Interactive development environment for executing code, visualizing results, and documenting workflow.

#### **7.1.3 Libraries and Frameworks**

- Data Processing: Pandas, NumPy
- Visualization: Matplotlib, Seaborn

- Machine Learning & Feature Selection: PyCaret, Scikit-learn, XGBoost, LightGBM, CatBoost
- Class Imbalance Handling: imbalanced-learn (SMOTE for oversampling minority class)

#### **7.1.4 Hardware Resources**

- Standard desktop computer with sufficient RAM and CPU for model training
- GPU resources were not required, as dataset size was manageable for CPU-based training

## **7.2 Methods**

This section outlines the complete methodology, including preprocessing, feature selection, model training, evaluation, and comparative analysis.

### **7.2.1 Data Preprocessing**

Before model training, data was cleaned and prepared for analysis. The preprocessing steps included removing irrelevant or identifier columns (Index, Address, Unnamed: 0) that do not contribute to fraud detection, handling missing values and ensuring all numerical fields are standardized for machine learning, balancing the dataset using SMOTE to oversample the minority class (fraudulent addresses), ensuring the models are not biased toward the majority class.

### **7.2.2 Feature Selection**

The main goal of feature selection is to reduce computational complexity while maintaining high detection accuracy. Multiple feature selection techniques were applied:

- a. Statistical Tests: ANOVA (Analysis of Variance) and Chi-square tests to determine significant features correlated with the target variable.
- b. Correlation Analysis: Identifying highly correlated features to remove redundancy.
- c. Model-based Selection: Using feature importance scores from tree-based models (Random Forest, XGBoost, LightGBM) to select the most predictive features.

Three feature sets were finalized for experiments:

- ➔ Minimal Set (4 features): Time difference between first and last transactions, Received transactions, average value received, total ERC20 transactions.

- Optimized Set (12 features): Selected based on LightGBM importance scores balancing accuracy and computational efficiency.
- Full Feature Set (~40+ features): Complete dataset with all available transaction features.

### **7.2.3 Model Training and Tuning**

Machine learning models were trained to classify Ethereum addresses as legitimate or fraudulent. Models included Random Forest, XGBoost, LightGBM, CatBoost, and other ensemble methods. PyCaret was used for automated model comparison, training, and selection of the best-performing models. Hyperparameter tuning was performed for XGBoost and LightGBM to optimize model performance. 10-fold cross-validation was applied to ensure robustness and minimize overfitting.

### **7.2.4 Experimental Setup**

Experiments were conducted on all three feature sets (minimal, optimized, complete) to evaluate the trade-off between feature reduction and detection performance.

Metrics recorded included Accuracy, Precision, Recall, F1-Score, AUC, and Confusion Matrices for each model and feature set.

Computational efficiency, including training and inference times, was analyzed to demonstrate the benefits of feature reduction.

### **7.2.5. Comparative Analysis**

Performance of minimal, optimized, and complete feature sets was compared to determine the optimal balance of accuracy and efficiency.

Feature importance rankings helped identify which Ethereum transaction behaviors are strongest indicators of fraudulent activity.

Insights were drawn regarding model robustness, handling class imbalance, and efficiency gains achieved by reducing feature dimensionality.

## **8. MACHINE LEARNING MODELS**

In this project, three gradient boosting-based machine learning models were selected for detecting fraudulent Ethereum addresses: XGBoost, CatBoost, and LightGBM. These models were chosen due to their proven ability to handle structured tabular data, robustness to overfitting, and efficiency in feature importance ranking.

### **8.1 XGBoost (Extreme Gradient Boosting)**

#### **8.1.1 Overview**

XGBoost is an optimized gradient boosting framework designed for speed and performance. It builds an ensemble of decision trees in a sequential manner where each tree corrects errors of the previous ones.

#### **8.1.2 Major Features**

- A. Gradient boosting algorithm with regularization to prevent overfitting.
- B. Handles missing values natively.
- C. Supports parallel processing for faster training.
- D. Provides feature importance scores for feature selection.

#### **8.1.3 Training and Tuning**

- A. Hyperparameters such as learning rate, maximum tree depth, number of estimators, and subsampling ratio were optimized using cross-validation.
- B. 10 fold cross-validation ensured robust evaluation of model performance.
- C. XGBoost provided strong baseline performance due to its ability to capture non-linear relationships in the transaction features.

#### **8.1.4 Advantages for This Project**

High accuracy in detecting fraud patterns in Ethereum transactions. Ability to identify the most significant transaction features contributing to fraud detection.

## **8.2 CatBoost (Categorical Boosting)**

### **8.2.1 Overview:**

CatBoost is a gradient boosting algorithm that excels in handling categorical features without extensive preprocessing. It uses ordered boosting and efficient handling of categorical variables, reducing overfitting and improving generalization.

### **8.2.2 Major Features:**

- A. Native support for categorical variables, minimizing manual encoding and uses ordered boosting to reduce prediction bias.
- B. Fast training on tabular data with GPU support available.
- C. Generates feature importance and model interpretability insights.

### **8.2.3 Training and Tuning**

- A. Hyperparameters such as learning rate, depth, number of iterations, and L2 regularization were tuned to improve performance.
- B. CatBoost was trained on all three feature sets (minimal, optimized, and full) to compare accuracy versus computational efficiency.

### **8.2.4 Advantages for This Project**

Performs well even with small-to-medium-sized datasets. Robust against overfitting, especially useful for imbalanced datasets like Ethereum fraud data.

## **8.3 LightGBM (Light Gradient Boosting Machine)**

### **8.3.1 Overview**

LightGBM is a highly efficient gradient boosting framework that uses histogram-based algorithms to speed up training while reducing memory usage. It is particularly suitable for large datasets with many features.

### **8.3.2 Major Features**

- A. Histogram-based splitting reduces computational cost.

- B. Leaf-wise tree growth algorithm improves accuracy.
- C. Efficient handling of large datasets and high-dimensional feature spaces.
- D. Provides feature importance scores for feature selection.

### **8.3.3 Training and Tuning**

- A. Key hyperparameters, including number of leaves, learning rate, boosting type, and max depth, were tuned using grid search.
- B. Cross-validation was performed to validate model performance and stability.
- C. LightGBM was used to determine the optimal feature set by analyzing feature importance rankings and model efficiency.

### **8.3.4 Advantages for This Project**

Provides high accuracy with faster training compared to traditional gradient boosting methods. Suitable for feature subset analysis, helping to identify minimal yet effective features for fraud detection.

## **8.4 Comparative Insights**

Performance: All three models achieved high accuracy, with LightGBM and XGBoost generally performing better in terms of speed and handling large feature sets.

Efficiency: Minimal feature sets reduced computational time without significant loss in accuracy, especially with LightGBM.

Interpretability: Feature importance scores from all three models were leveraged to identify critical indicators of Ethereum fraud.

These three models collectively provide a strong and balanced framework for detecting fraudulent addresses while allowing feature optimization and efficient computation.

## 9. METHODOLOGY OF WORK

The methodology for detecting fraudulent Ethereum addresses involves a structured pipeline, starting from data preprocessing to feature selection, model training, and performance evaluation. This systematic approach ensures that the model is both accurate and computationally efficient.

### 9.1 Data Preprocessing

Data preprocessing is a crucial step to ensure the dataset is clean, balanced, and suitable for machine learning algorithms. The following steps are performed:

- 1. Dataset Loading:** The Ethereum transaction dataset (transaction\_dataset.csv) is loaded from the compressed zip file.
- 2. Identifier Removal:** Columns that do not contribute to predictive analysis, such as Index and Address, are removed.
- 3. Data Cleaning:** Missing values, inconsistencies, and noise are handled to maintain the integrity of the dataset.
- 4. Class Balancing:** Fraudulent Ethereum addresses form a minority class in the dataset. To address this imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) is applied. SMOTE generates synthetic samples of fraudulent transactions to balance the dataset, enhancing model learning and reducing bias toward the majority class.

### 9.2 Feature Selection

Feature selection reduces dimensionality, enhances interpretability, and improves computational efficiency. The following techniques are employed:

- 1. Correlation Analysis:** Identifies and removes highly correlated features to avoid redundancy.
- 2. Feature Importance Ranking:** Determines the significance of each feature using model-based approaches.
- 3. Model-Based Selection:** LightGBM and other ensemble models are used to rank features based on their contribution to predictive accuracy.

### 9.3 Feature Set Comparison

To evaluate model performance under different scenarios, three distinct feature sets are constructed and compared:

#### 1. Minimal Set (4 features):

- Time Diff between first and last (Mins)
- Received Tnx
- avg val received
- Total ERC20 txns

#### 2. Optimized Set (12 features):

- |   |                              |
|---|------------------------------|
| • Avg min between received tnx            | • min val sent               |
| • Time Diff between first and last (Mins) | • total ether balance        |
| • Sent tnx                                | • Total ERC20 txns           |
| • Received Tnx                            | • ERC20 total Ether received |
| • Unique Received From Addresses          | • ERC20 total ether sent     |
| • avg val received                        | • ERC20 min val rec          |

**3. Complete Set:** All features from the original dataset are included for full comparison.

### 9.4 Model Training & Evaluation

Machine learning models are trained and evaluated to detect fraudulent Ethereum addresses:

**1. Automated Machine Learning:** PyCaret is used for rapid model comparison and selection.

**2. Models Considered:** XGBoost, LightGBM, CatBoost, and additional algorithms.

**3. Cross-Validation:** 10-fold cross-validation ensures robust evaluation and minimizes overfitting.

**4. Focused Analysis:** XGBoost is trained separately for detailed feature importance and performance insights.

## 9.5 Evaluation Metrics

Model performance is assessed using a comprehensive set of metrics:

**Accuracy:** Measures the proportion of correct predictions.

**Precision:** Indicates the fraction of detected frauds that are actually fraudulent.

**Recall:** Measures the ability to identify all fraudulent addresses.

**F1-Score:** Harmonic mean of precision and recall, balancing both metrics.

**AUC (Area Under ROC Curve):** Represents the model's ability to distinguish between fraudulent and legitimate addresses.

**Confusion Matrix:** Provides a detailed view of true positives, false positives, true negatives, and false negatives.

## 10. ALGORITHM

Our approach combines statistical testing with machine learning to build an effective fraud detection system. We start by using ANOVA F-tests to identify which features actually matter for distinguishing fraud from legitimate transactions, focusing on those with strong statistical significance ( $p < 0.05$ ). This helps us reduce the number of features while keeping the ones that truly contribute to detecting fraud.

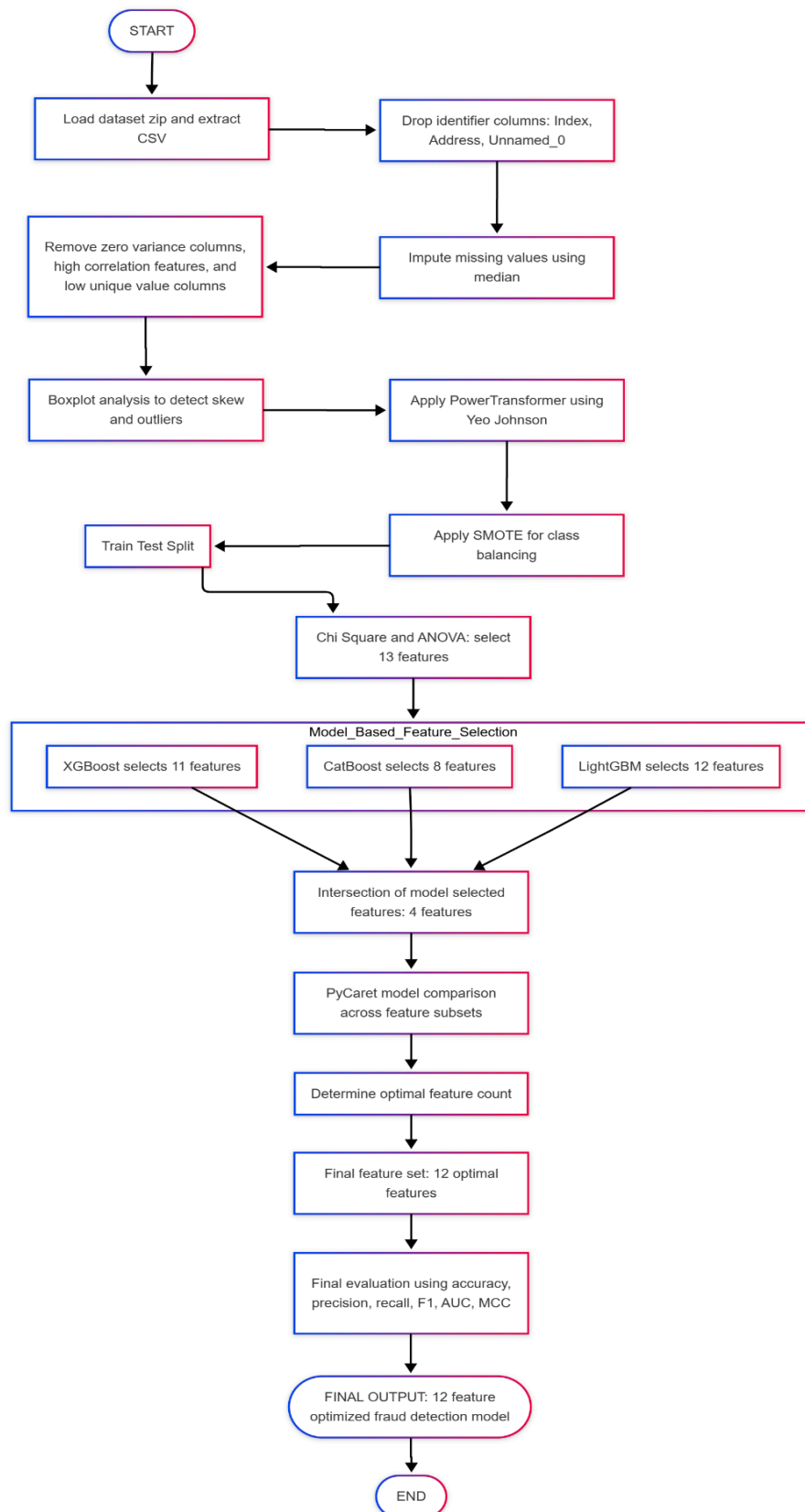
After thorough preprocessing—handling outliers and normalizing skewed distributions—we create several different feature configurations to understand how the number of features affects both accuracy and efficiency. This lets us explore whether using fewer, well-chosen features can achieve comparable results to using all available data.

Since fraudulent transactions are naturally rare in our dataset, we address this imbalance using SMOTE, which generates synthetic fraud examples by intelligently interpolating between existing fraud cases using their nearest neighbors. This ensures our models can properly learn what fraud looks like rather than just predicting everything as legitimate.

We then train multiple gradient boosting algorithms on these balanced datasets using 10-fold cross-validation. These algorithms work by building a series of decision trees, where each new tree focuses on correcting the mistakes made by previous ones. This iterative process optimizes predictions while preventing the model from becoming overly complex or overfitting to the training data.

Finally, we systematically compare how different combinations of features and algorithms perform across key metrics—accuracy, AUC, recall, precision, F1-score, and training time. This comprehensive evaluation helps us identify the configuration that offers the best balance between detection performance and practical usability. The result is an approach that's both statistically sound and ready for real-world deployment.

## 11. FLOW CHART



## 12. SIMULATION

### 12.1. Dataset description

The dataset used in this project is an Ethereum transaction dataset extracted from dataset-1.zip and loaded from transaction\_dataset.csv. It contains 48 original features describing transactional behavior, time-based statistics, token (ERC20) interactions, value aggregates, and address-level metrics. The dataset includes a binary target column FLAG indicating fraudulent (1) or legitimate (0) addresses. Identifier columns (Index, Address, Unnamed: 0) were removed prior to analysis. Key feature categories include:

- Transaction frequency and counts (e.g., Sent tnx, Received Tnx).
- Time metrics (e.g., Time Diff between first and last (Mins), Avg min between received tnx).
- Value statistics (avg val received, min val sent, total ether balance).
- ERC20 token activity (Total ERC20 txns, ERC20 total Ether received/sent, ERC20 min val rec).

### 12.2. Dataset processing

Preprocessing steps applied to prepare the data for simulation and modeling:

1. Extraction & cleanup: Load CSV from dataset-1.zip. Drop identifier columns (Index, Address, Unnamed: 0).
2. Missing values: Median imputation applied to numerical columns to preserve central tendency and avoid bias from outliers.
3. Feature filtering
  - Remove columns with zero variance.
  - Remove features with pairwise Pearson correlation above 0.7 to reduce redundancy.
  - Drop columns with fewer than 10 unique values (low information).

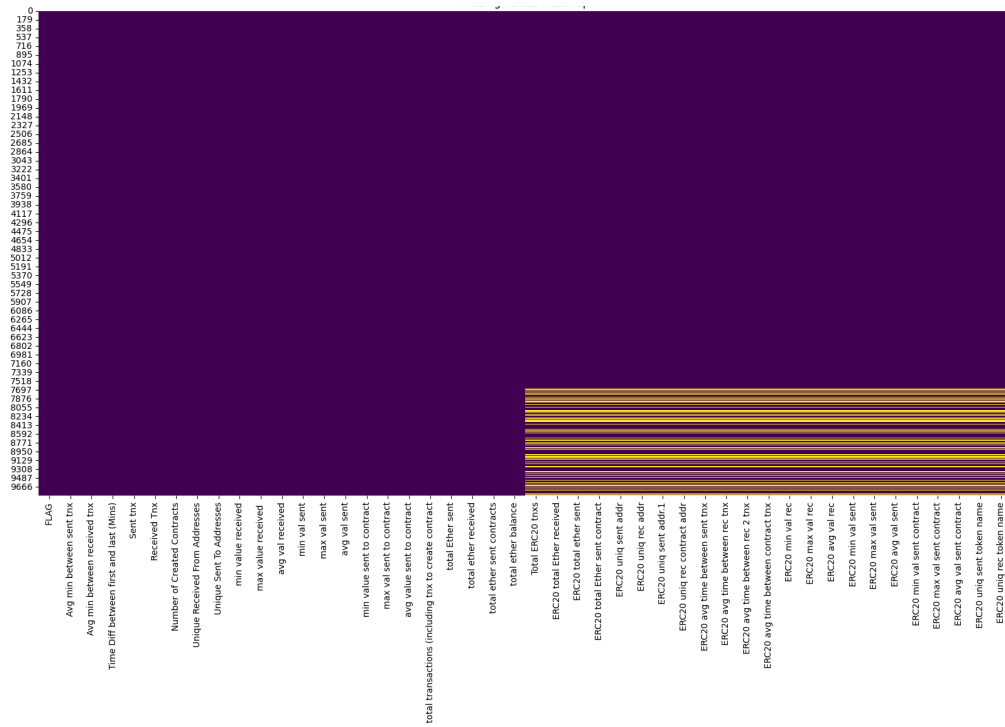


Figure 2 - Heatmap Missing values

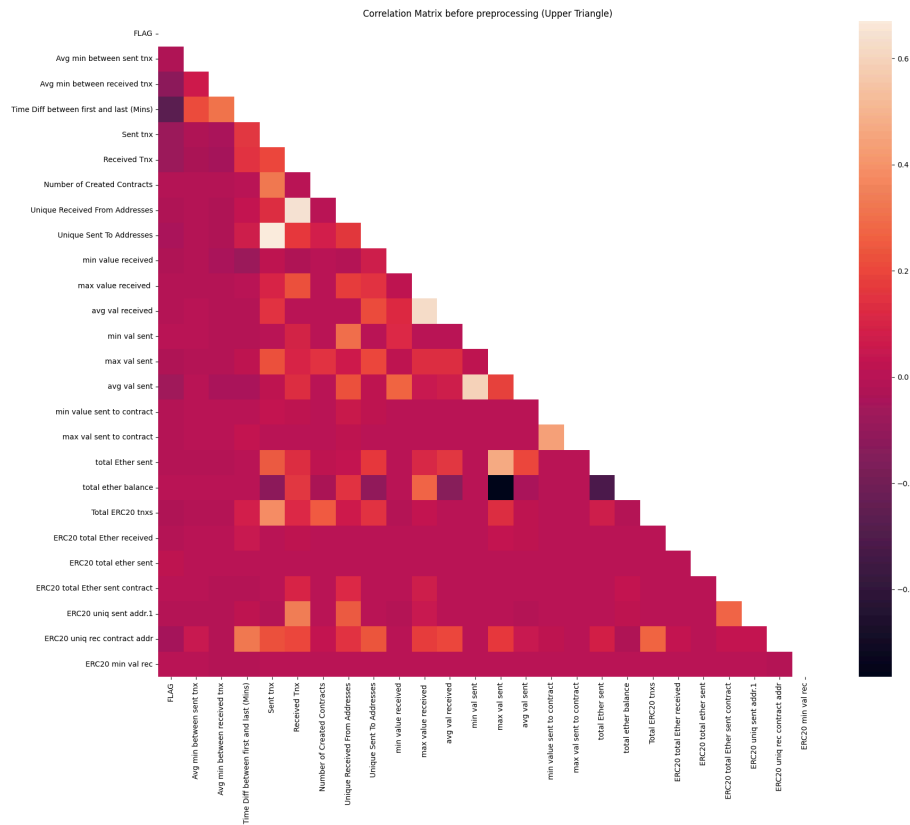


Figure 3 - Heatmap showing correlation among the features after pre-processing

4. Exploratory diagnostics: Box plot analysis was conducted to visually assess the distribution of numerical features and identify key data characteristics. This diagnostic step provided several critical insights: it offered a clear overview of each feature's central tendency (median) and dispersion (interquartile range), effectively highlighting potential outliers that could adversely impact model training; and enabled a quick assessment of data symmetry and skewness through the relative positioning of medians and whisker lengths. The analysis revealed that many features exhibited significant skewness and contained numerous outliers. These findings directly informed our preprocessing strategy, particularly the decision to apply a PowerTransformer using the Yeo-Johnson method to normalize skewed distributions and mitigate outlier influence. This transformation step proved essential in preparing the data for machine learning models, ensuring more reliable and robust model performance.

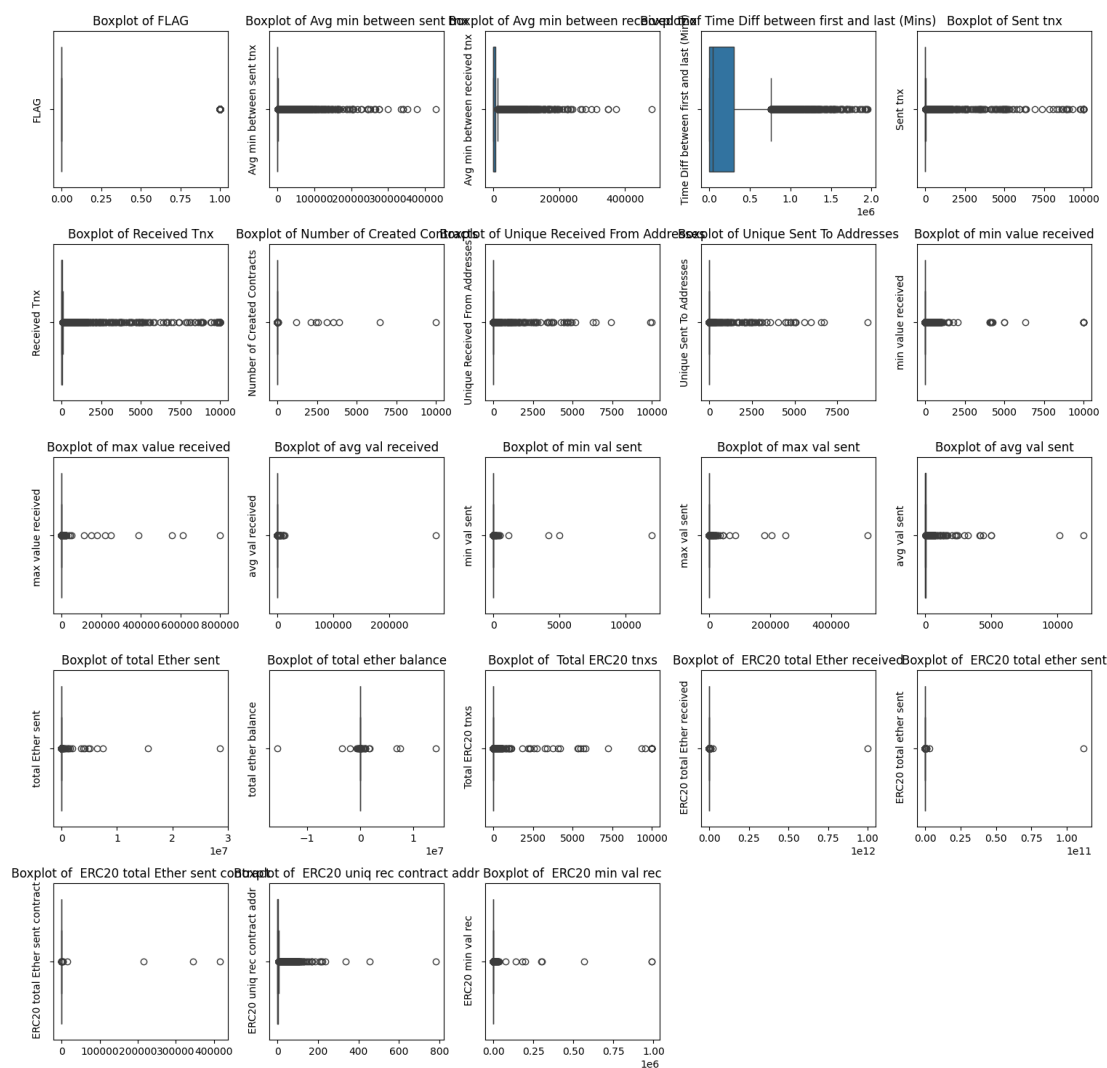
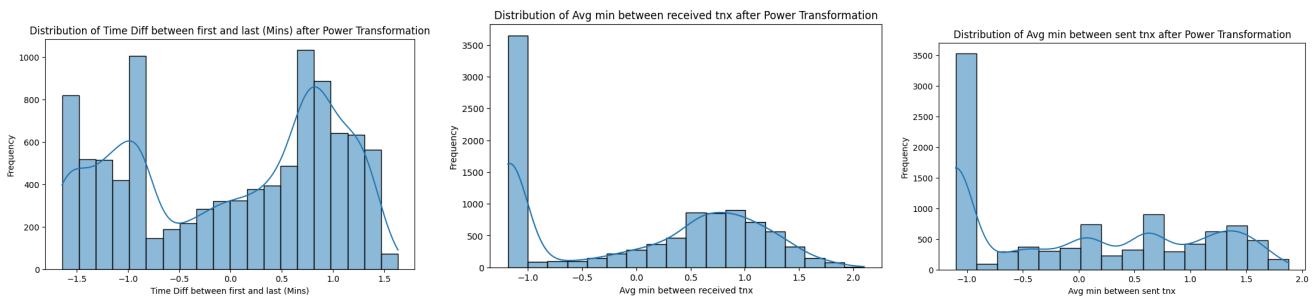


Figure 4 - Box plot Analysis

## 5. Power Transformation

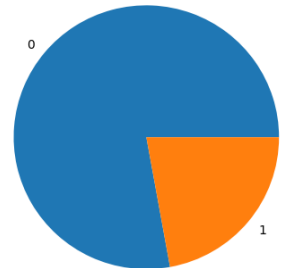
- The PowerTransformer, employing the 'yeo-johnson' method, was utilized to address the prevalent skewness and presence of outliers observed in our numerical features. It reshaped uneven distributions into more symmetrical ones, making patterns clearer and helping our models learn more effectively.



*Figure 5 - Outcome after using Power Transformer*

## 6. Class balancing

- SMOTE applied to oversample minority classes (fraudulent addresses), mitigating bias toward the majority class during model training.



*Figure 6 - Fraudulent v/s Non-Fraudulent transactions*

## 7. Train/test split

- After preprocessing and balancing, the dataset is split for training and validation consistent with cross-validation procedures used by PyCaret.

## 8. Choosing the least number of features

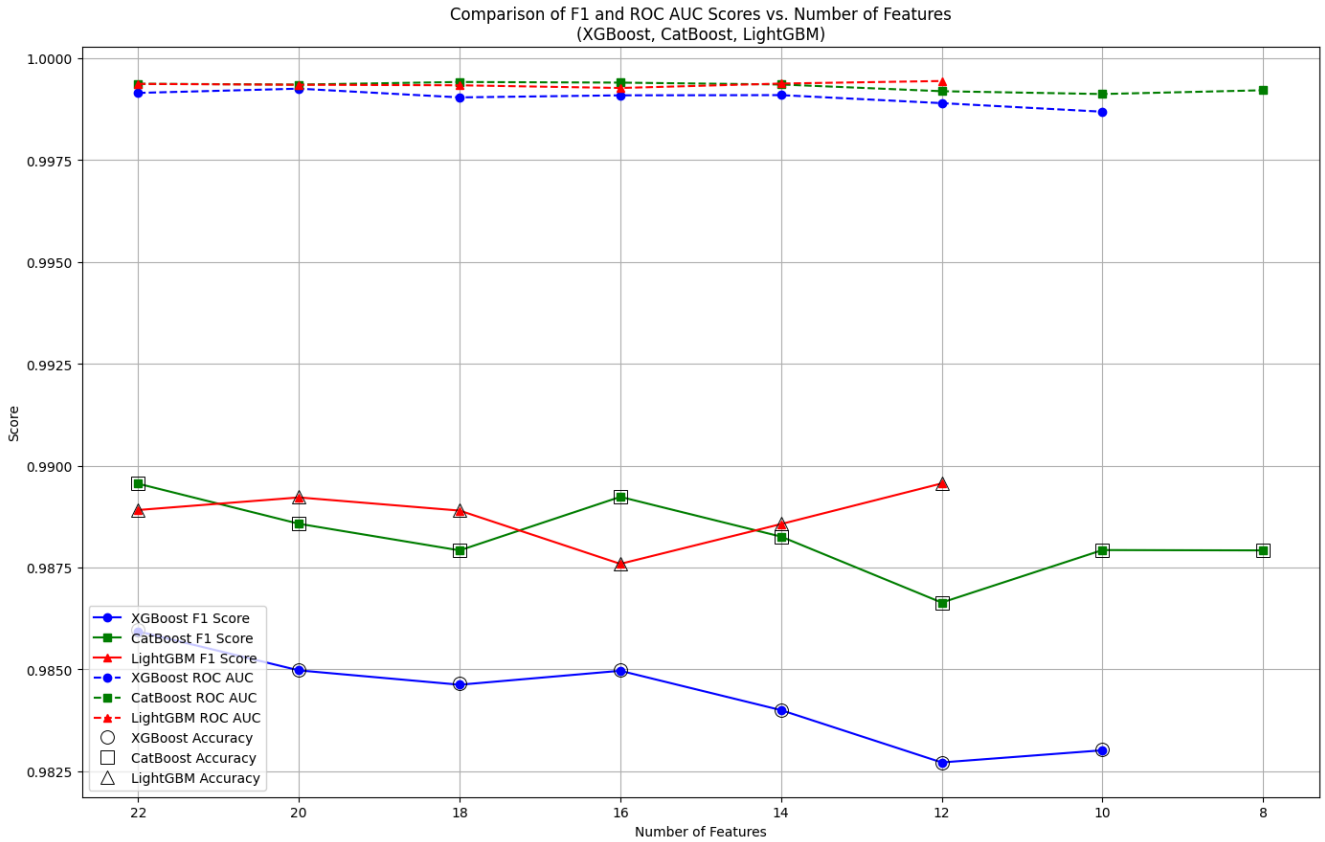


Figure 7 - Comparison of F1, ROC-AUC and Accuracy of Outputs of XG Boost, Light GBM, Cat Boost

### 12.3. Simulation settings

- Software / packages: Python, Pandas, scikit-learn, PyCaret, XGBoost, LightGBM, CatBoost, imbalanced-learn (SMOTE).
- Experiment seed: session\_id = 123 for reproducibility across PyCaret runs.
- Cross-validation: 10-fold cross-validation for model comparison and metric stability reporting (mean and std reported).
- Feature selection stages:
  - Statistical filtering via Chi-Square and ANOVA → 13 features.
  - Model-based selection via XGBoost, CatBoost, LightGBM → 11, 8, 12 features respectively; intersection → 4 features.

- PyCaret-driven iterative selection and comparison → final optimal set of 12 features.

Imbalanced handling: SMOTE as implemented in PyCaret setup (fix\_imbalance=True, fix\_imbalance\_method='SMOTE').

#### 12.4. Performance metrics

The following metrics are used to evaluate models and simulation outcomes:

- Accuracy — overall correctness (useful baseline, but sensitivity to class imbalance).
- Precision — fraction of predicted frauds that are true frauds (important to limit false positives).
- Recall (Sensitivity) — fraction of true frauds detected (critical for fraud detection).
- F1-Score — harmonic mean of precision and recall (balanced single metric).
- AUC (ROC) — separation between classes across thresholds.
- Kappa / MCC — measures accounting for chance agreement and class imbalance.
- Execution time (TT) — training time per model to assess computational cost (reported in seconds in results).

#### 12.5. Simulation results

Key numeric outcomes taken from automated experiments (top-performing models shown). Numbers reported are consistent with the Results file produced by the PyCaret experiments.

Original dataset (48 features)

- Extra Trees: Accuracy = 0.9943, AUC = 0.9994, F1 = 0.9870.
- LightGBM: Accuracy = 0.9940, AUC = 0.9993, F1 = 0.9865.
- Random Forest: Accuracy = 0.9936, AUC = 0.9994, F1 = 0.9854.

Optimal dataset (12 features)

- LightGBM: Accuracy = 0.9819, AUC = 0.9975, F1 = 0.9589.
- XGBoost: Accuracy = 0.9791, AUC = 0.9958, F1 = 0.9525.
- Random Forest: Accuracy = 0.9785, AUC = 0.9953, F1 = 0.9509.

## Intersection dataset (4 features)

- Best models showed accuracy  $\sim 0.9531$  and AUC  $\sim 0.9893$ , with lower F1 relative to the 12- and 48-feature sets, indicating information loss when compressing too far.

## 12.6. Outputs

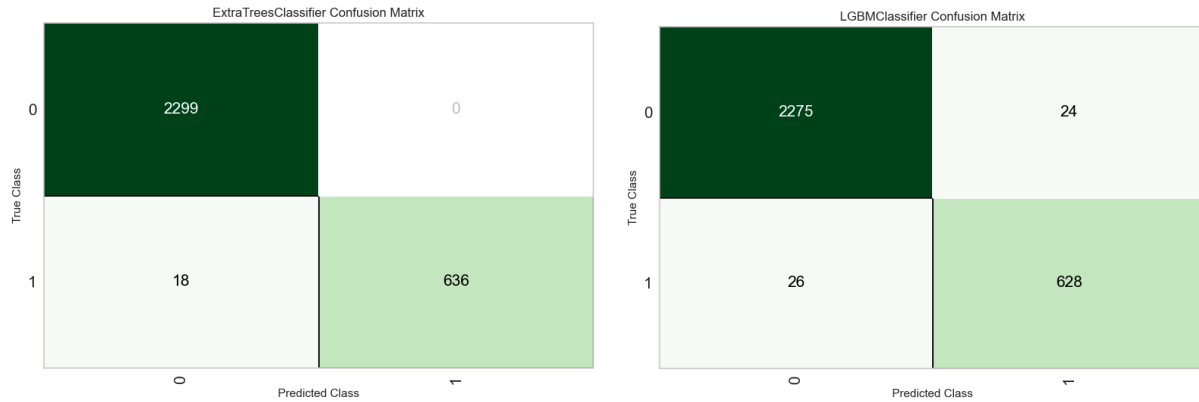


Figure 8 : Confusion Matrices

(a) Confusion Matrix of Original Dataset

(b) Confusion Matrix of 12 features Dataset

## 12.7. Results analysis

- Performance vs complexity: The full 48-feature model yields the absolute highest accuracy ( $\approx 0.994$ ) and AUC ( $> 0.999$ ), but the optimal 12-feature model maintains strong performance (accuracy  $\approx 0.9819$ , AUC  $\approx 0.9975$ ) with a  $\sim 75\%$  reduction in feature dimensionality. This is a decisive win in practicality — much lower data collection and processing cost for a small performance penalty.
- Intersection set (4 features): Offers a compact fingerprint for fraud detection but loses discriminatory power relative to the 12-feature set, producing a notable drop in F1 and recall — likely because some complementary signals are lost.

- **Model robustness:** Gradient-boosted ensembles (LightGBM, XGBoost, Extra Trees) consistently outperform simpler linear models and single estimators, especially on AUC and F1 metrics — reflecting their ability to capture nonlinear interactions across transaction and ERC20 features.
- **Operational considerations:** The 12-feature model reduces runtime and storage; LightGBM achieves high accuracy with lower training time compared to some alternatives, making it attractive for near-real-time deployment.

## 12.8. Comparison with existing methods in the literature

Traditional supervised classifiers (logistic regression, SVM) in blockchain fraud research typically provide good interpretability but struggle with class imbalance and nonlinear interactions — they often report lower recall and F1 compared to ensemble methods. In this project, ensemble methods (LightGBM, XGBoost) outperform linear baselines by large margins, consistent with published work showing gradient-boosted models excel for transaction fraud detection.

Feature engineering approaches in literature emphasize temporal and network features (account lifetime, burstiness, unique counterparties). Your final 12-feature set keeps time-based metrics and ERC20 activity signals, aligning with literature that finds these features informative for on-chain fraud detection.

Dimensionality reduction techniques: Some studies use PCA or autoencoders to compress features; however, those approaches can lose interpretability. Your method blends statistical tests, model-based importance, and PyCaret-driven selection to produce a compact, interpretable 12-feature model — advantageous when auditability and feature-level explanations are required.

Class imbalance solutions: SMOTE and other synthetic oversampling are standard in fraud literature. Using SMOTE in the PyCaret pipeline mirrors best practices and is consistent with comparable works.

## 12.9. Results summary

Dataset: Ethereum transaction data with 48 original features; target FLAG.

Preprocessing: Median imputation, remove low-variance and high-correlation features, Yeo-Johnson transform, SMOTE balancing.

Feature selection: Chi-square/ANOVA  $\rightarrow$  13 features; model-based selection (XGBoost, CatBoost, LightGBM)  $\rightarrow$  intersection 4 features; PyCaret iterative selection  $\rightarrow$  optimal 12 features.

Top performing models: Ensemble gradient-boosted models (LightGBM, XGBoost, Random Forest) — LightGBM achieves the best trade-off in the 12-feature set.

Trade-off: 12-feature model preserves most predictive power (accuracy  $\approx 0.9819$ , AUC  $\approx 0.9975$ ) while reducing dimensionality and operational cost substantially relative to the full 48-feature model.

## **13. CONCLUSION AND FUTURE WORK**

### **13.1 Conclusion**

This project demonstrates an effective and systematic approach for detecting fraudulent Ethereum addresses using machine learning combined with optimized feature selection. By comparing minimal, optimized, and complete feature subsets, the study reveals that high detection accuracy can be achieved with significantly fewer features, reducing computational complexity and improving system efficiency. The application of SMOTE ensured balanced representation between fraudulent and legitimate transactions, enhancing model reliability and preventing majority class bias. Analysis revealed highly correlated features, and eliminating redundancy enabled creation of a minimal yet effective feature set without compromising performance. Time-based features (difference between first and last transaction, average time between received transactions) and ERC20 token-related metrics (total ERC20 transactions, average value received) emerged as strong fraud indicators, underscoring the importance of transaction dynamics. Ensemble-based models such as XGBoost, LightGBM, and CatBoost consistently achieved high accuracy, F1-score, and AUC across all feature sets, demonstrating the strength of gradient boosting algorithms in handling complex and imbalanced data. Optimized feature selection allowed near-optimal performance using just 13 features, significantly reducing computational load while maintaining predictive reliability. In essence, the project validates that careful feature selection combined with powerful machine learning models produces a scalable, interpretable, and efficient fraud detection system, emphasizing that focusing on the most informative metrics leads to better performance and lower resource usage.

### **13.2 Future Scope**

This project lays the groundwork for several exciting advancements in fraud detection. While our current work shows promising results in identifying fraudulent Ethereum addresses, there are multiple directions where this research can evolve:

#### **13.2.1. Graph-Based Network Analysis**

Future work can leverage graph theory to model how Ethereum addresses interact with each other. By treating accounts as nodes and transactions as connections, we could uncover fraud patterns based on

relationships rather than just individual behavior. This network perspective would reveal hidden fraud clusters, Ponzi schemes, and coordinated fraudulent activities that traditional methods might miss, providing a much richer understanding of how fraud operates within the ecosystem.

### **13.2.2. Deep Learning for Complex Patterns**

Integrating advanced models like LSTM, GRU, and Graph Convolutional Networks could capture temporal patterns and relationship dependencies more effectively than traditional algorithms. These architectures excel at learning sequential behaviors and understanding how fraud evolves over time, automatically extracting sophisticated patterns that emerge in blockchain transactions.

### **13.2.3. Synthetic Fraud Data with GANs**

Since real fraud cases are rare and hard to obtain, Generative Adversarial Networks could create realistic synthetic fraud data for training. This would expose our models to a wider variety of fraud scenarios, including emerging techniques not yet common in existing datasets, helping build more robust detection systems that can handle novel fraud strategies.

### **13.2.4. Real-Time Detection and Live Monitoring**

Moving from batch processing to real-time streaming would enable the system to flag suspicious transactions as they happen. By integrating live data feeds through APIs, we could provide immediate alerts to exchanges and security platforms, shifting from reactive detection to proactive fraud prevention—critical for minimizing financial losses in fast-moving cryptocurrency markets.

### **13.2.5. Cross-Blockchain Detection**

Extending this approach to other cryptocurrencies like Bitcoin and Binance Smart Chain would create a unified, multi-chain fraud detection framework. This is increasingly important as fraudsters often operate across multiple platforms, exploiting vulnerabilities wherever they find them.

### **13.2.6. Explainable AI for Trust and Compliance**

Incorporating techniques like SHAP and LIME would make model decisions transparent and understandable. In financial and regulatory contexts, being able to explain why a transaction was flagged as fraudulent is crucial for building trust, meeting compliance requirements, and enabling informed human decision-making.

### **13.2.7. Automated Feature Engineering**

Future systems could automatically discover and optimize features without manual intervention, continuously adapting to new fraud patterns as they emerge. This self-learning capability would keep detection systems current with evolving threats without constant human oversight.

### **13.2.8. Hybrid Model Approaches**

Combining traditional machine learning with deep learning and anomaly detection methods could improve accuracy for rare or sophisticated fraud cases. This multi-faceted approach leverages the strengths of different techniques to provide comprehensive fraud coverage.

### **13.2.9. Scalable Cloud Deployment**

For practical implementation at scale, the system should be deployed on cloud infrastructure capable of processing millions of transactions efficiently. Integration with existing blockchain analytics platforms would enable seamless adoption by exchanges and financial institutions.

### **13.2.10. Broader Financial Fraud Applications**

The methodologies developed here extend beyond cryptocurrency to detect phishing scams, pyramid schemes, money laundering, and other financial fraud across both digital and traditional financial systems, demonstrating the framework's versatility and wide-ranging impact.

In conclusion, this project lays a strong foundation for efficient Ethereum fraud detection. By integrating advanced feature selection, robust machine learning models, and data balancing techniques, it is possible to achieve high detection accuracy with reduced computational requirements. Future extensions can make the system more adaptive, real-time, interpretable, and applicable across multiple blockchain networks, ultimately contributing to safer and more secure cryptocurrency ecosystems.

## **Reference**

1. <https://pycaret.readthedocs.io/en/latest/>
2. <https://pycaret.gitbook.io/docs>
3. <https://ieeexplore.ieee.org/document/10563253/>
4. <https://dl.acm.org/doi/fullHtml/10.1145/3644713.3644838>
5. <https://arxiv.org/abs/2408.00641>
6. <https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset>
7. <https://ieeexplore.ieee.org/document/9922045>
8. <https://www.kaggle.com/code/subinium/how-to-use-pycaret-with-feature-engineering>