Deep Learning and Machine Learning based Ethereum Fraud Detection Framework

A Thesis Submitted
In Partial Fulfillment of the Requirements for the
Degree of

MASTER OF TECHNOLOGY
in
ARTIFICIAL INTELLIGENCE

by
Aditya Rastogi
(2K23/AFI/11)

Under the Supervision of
Dr. Pawan Singh Mehra
Assistant Professor
(Dept of Computer Science & Engineering)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

May 2025

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I, Aditya Rastogi, Roll No. 23/AFI/11 student of M.Tech (Artificial Intelligence), hereby certify that the work which is being presented in the thesis entitled "**Deep Learning and Machine Learning based Ethereum Fraud Detection Framework**" in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Artificial Intelligence in the Department of Computer Science and Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from August 2023 to Jun 2025 under the supervision of **Dr. Pawan Singh Mehra**, Asst. Professor,Dept of Computer Science and Engineering. The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Place: Delhi **Candidate's Signature**

This is to certify that the student has incorporated all the corrections suggested by the examiners in the thesis and the statement made by the candidate is correct to the best of our knowledge.

Signature of Supervisor (s) **Signature of External Examiner**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE

I hereby certify that the Project titled "**Deep Learning and Machine Learning based Ethereum Fraud Detection Framework**", submitted by **Aditya Rastogi**, Roll No. **2K23/AFI/11**, Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology (M.Tech) in Artificial Intelligence is a genuine record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in par or full for any Degree to this University or elsewhere.

Place: Delhi
Date: 31.05.2025

Dr. Pawan Singh Mehra
Assistant Professor
Delhi Technological University

ACKNOWLEDGEMENT

I am grateful to **Prof. Manoj Kumar**, HOD (Department of Computer Science and Engineering), Delhi Technological University (Formerly Delhi College of Engineering), New Delhi, and all other faculty members of our department for their astute guidance, constant encouragement, and sincere support for this project work.

I am writing to express our profound gratitude and deep regard to my project mentor **Dr. Pawan Singh Mehra**, for his exemplary guidance, valuable feedback, and constant encouragement throughout the project. His valuable suggestions were of immense help throughout the project work. His perspective criticism kept us working to make this project much better. Working under her was an extremely knowledgeable experience for us.

I would also like to thank all my friends for their help and support sincerely.

Aditya Rastogi

(2K23/AFI/11)

ABSTRACT

This thesis presents a comprehensive framework for detecting fraudulent Ethereum addresses using machine learning and deep learning techniques. Leveraging a rich set of behavioral and transactional features, the framework incorporates robust preprocessing (including log transformation and scaling), feature selection via Recursive Feature Elimination (RFE), and classification using an FT-Transformer model tailored for tabular data. To address the challenge of class imbalance inherent in fraud detection, a weighted binary cross-entropy loss is employed. The proposed system achieves high performance, with an accuracy of 97.09% , F1-score of 93.55%, precision of 91.76%, recall of 95.41%, and AUC of 0.9961. Even after reducing the feature space, the model maintains comparable accuracy, demonstrating its efficiency and generalizability. This work introduces a scalable and interpretable deep learning-based approach for Ethereum fraud detection, contributing to the growing field of blockchain security.

Keywords- Ethereum, Fraud Detection, Deep Learning, Machine Learning, FT-Transformer,Recursive Feature Elimination (RFE), Class Imbalance

Table of Content

List of Tables

LIST OF FIGURES

LIST OF ABBREVIATIONS

CLSTM	Convolutional Long Short-Term Memory
CNN	Convolutional Neural Network
EVM	Ethereum Virtual Machine
ETH	Ether (Ethereum cryptocurrency)
FT-Transformer	Feature Tokenizer Transformer
GNN	Graph Neural Network
LSTM	Long Short-Term Memory
ML	Machine Learning
RF	Random Forest
RFE	Recursive Feature Elimination
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting
F1	F1-Score
AUC	Area Under Curve

CHAPTER 1
INTRODUCTION

Back in 2015, Ethereum was introduced by Vitalik Buterin as a decentralized blockchain platform, aim to do more than just facilitate digital currency transfers. Bitcoin, which mainly focused on sending and receiving value, Ethereum brought the idea of smart contracts—small programs that run on the blockchain automatically. This made it possible to build a wide variety of decentralized applications (dApps), and has since led to the rise of entire ecosystems like decentralized finance (DeFi) and NFTs. These technologies have really changed how people interact digitally and manage value online. But at the same time, Ethereum's opened and anonymous nature also makes it a good target for fraud. Since transactions can't be reversed and there's no real-world identity linked to most addresses, bad actors can take advantage of users pretty easily. Traditional fraud detection methods, especially those that are rule-based or require manual work, often don't catch up with how quickly fraud techniques are evolving.

In this thesis, we try to build a complete fraud detection framework for Ethereum by using advanced machine learning and deep learning models. The approach we're proposing includes strong preprocessing step, smart feature selection, and also uses a transformer-based deep learning model (FT-Transformer) to help catch fraudulent addresses more accurately. It also looks at important issues like having too many features and class imbalance problems of not enough fraud examples in the data.

1.1 Architecture of Ethereum

1.1.1 Ethereum Virtual Machine (EVM)

Ethereum Virtual Machine or EVM is the main engine that runs smart contracts on Ethereum. It does the execution on all nodes in the network and make sure they all getting the same result. This is important for everyone to agree on what happen on blockchain. EVM handles lot of complex computing, and tries using resource in smart way. It also helps prevent bad codes or peoples from breaking the system or wasting computing power for nothing.

1.1.2 Smart Contracts

Smart contracts is like auto programs which run when some condition is met. It is mainly written in Solidity and then put on blockchain. After that, they just work by itself. You don’t need any middle person or someone to control. It does the task like transfer tokens, check some rules, or give access if allowed. It used for DeFi things, token launch, lending and borrow, exchange without a middleman and more others things too.

1.1.3 Ethereum Accounts

Ethereum have two type account. First is EOAs, or Externally Owned Accounts, which are controlled with private key by user. These account can send transactions, deploy contracts and also talk with other contracts. Other type is Contract Accounts. These ones not belong to anyone, they just working by logic written inside them. If they get transaction from other account, they just do what they are programmed to do. No need for someone control.

1.1.4 Transactions and Gas Mechanism

So, in Ethereum, everything mostly work through transactions, Deploying contract or calling a function in smart contract and transaction too. Every transaction has to be signed with crypto keys so it can be checked and not fake. But also, it not free — there's something called gas. Gas is like a fee you pay depending on how heavy the work is. If the transaction needs lot of computing, you pay increased gas. It helps protect the network, so people don't spam it. When there's not enough gas, the transaction just stuck or failed. So it keeps things from going out of control.

1.1.5 Consensus Mechanism

Ethereum before was using Proof-of-Work, like Bitcoin does. Miners had to solve these hard puzzles using lots of electric and hardware. But now it switch to Proof-of-Stake. It's way more eco friendly. Instead of solving stuff, now people stake Ether to become validator. If you got more Ether staked, you get more chance to be picked. And if you do something wrong, like trying to cheat the system, you can lose your stake. Hence , it is faster, cleaner, and safer than before.

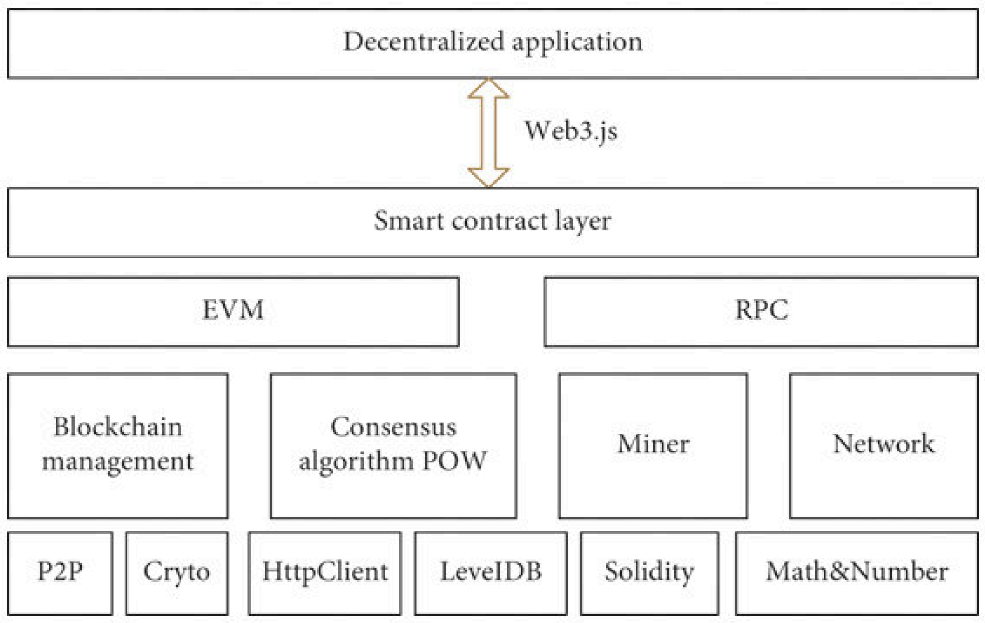


Fig1.1 Ethereum Architecture

1.2 Frauds in Ethereum

Although decentralized and transparent , frauds can still occur on the Ethereum blockchain. Among the most prevalent types of fraud in the Ethereum ecosystem are Ponzi schemes and Phishing attacks, both of which exploit the openness of the network and the anonymity of its users [3].

1.2.1 Ponzi scheme

It is a fraudulent investment model in which returns to previous investors are paid by using the capital from new inventors , rather than from any actual profit or legitimate business activity [19]. In the Ethereum blockchain , Ponzi schemes often present as high-yield smart contracts projects , which lures investors with a promise of fast gains. The transparent and immutable nature of smart contracts make them seem trustworthy to the investors . Due to the immutable nature of the Ethereum network, transactions are permanent once deployed on the blockchain , with no way for investors to recover their funds [19].

1.2.2 Phishing

In an Ethereum Network , users are targeted through social engineering with the aim to trick users into revealing their private information such as private keys or authorizing malicious transactions . Fake websites which imitate popular Ethereum services are used to trick users into sharing this information with the scammers . Some phishing attacks involve contracts that request unlimited token approvals, enabling attackers to drain user wallets even after a single interaction.

1.2.3 Rug Pulls

Rug pulls occur when malicious developers initiate seemingly legitimate decentralized finance projects, attract substantial investment, and then suddenly withdraw liquidity from decentralized exchanges. The abrupt removal of liquidity causes token values to plummet instantly, inflicting severe financial damage on unsuspecting investors and severely damaging market confidence.

1.3 Machine Learning Models

1.3.1 Logistic Regression

Logistic regression is used for binary classification problem by estimating probability from linear combination of input features. It gives results which are transparent and easy to understand, so it’s often used as a quick baseline when testing models. Because of its simplicity, it can be implemented fast and works well when the data have clear linear relationships. But logistic regression also have some limitations. It only capture linear patterns, which might be too simple when working with Ethereum transaction data that can be complex or non-linear. So, while it’s useful in early stages, it may miss deeper patterns in the fraud behaviour.

1.3.2 Random Forest

Random Forest is an ensemble machine learning algorithm that works by building multiple decision trees using random subsets of samples and combining their output — through majority voting for classification and average or mean for regression. RF is effective in detecting frauds in Ethereum network because of its ability to model complex , non-linear relationships in transaction data [5] , for example irregular gas usage or anomalous transfer patterns. RF is also robust to noise and can handle high-dimensional, imbalanced data which makes it ideal for Ethereum transaction dataset [18].

1.3.3 XGBoost (Extreme Gradient Boosting)

XGBoost is boosting method that builds decision trees one after another, where each new tree try to fix the mistake made by the previous ones. This process helps in making the predictions more accurate with each step. It works good even when the dataset have too many features, missing values, or lot of noise. One of the main strength of XGBoost is that it can handle large datasets quite efficiently, both in terms of memory and time. It also supports parallel processing and has regularization, which help to avoid overfitting sometimes. It able to capture hidden relationships and interactions between variables, which is useful in Ethereum data because fraud behaviour can be very complex and not easy to detect. Because of these things, XGBoost is used a lot for structured data and has shown strong results in many machine learning tasks and even real applications.

1.3.4 Support Vector Machines (SVM)

Support Vector Machines classify data by identifying hyperplanes maximizing margin between different classes. Highly effective in high-dimensional spaces, SVMs robustly handle complex datasets, though their performance significantly depend on proper kernel selection and parameter tuning, sometimes complicating practical implementation.

1.3.5 K-Nearest Neighbors (KNN)

KNN classifies samples based on proximity to labeled instances, providing intuitive and straightforward classifications. Despite simplicity, KNN can be computationally intensive with large datasets, limiting scalability in extensive Ethereum datasets.

1.4 Deep Learning Models

1.4.1 1D Convolutional Neural Network (1DCNN):

1DCNN is a deep learning model which is commonly used in pattern recognition and time series analysis. Originally CNNs were designed for image classification but 1DCNN has been proven useful in working with one-dimensional data like numerical time series . In Ethereum fraud detection localized patterns such as sudden spikes in transfer amount are identified by processing raw transaction data using 1DCNNs [8]. Features are extracted from the dataset using a convolutional layer , pooling layer to reduce dimensionality and to classify the pattern a fully connected layer is part of the architecture of 1 DCNN [8].

1.4.2 Long Short-Term Memory (LSTM)

LSTMs specialize in modeling sequential dependencies in transaction histories. They effectively capture evolving fraud strategies, recognizing gradual behavioral shifts in Ethereum address activities, making them particularly effective against slow-building fraud tactics.

1.4.3 CNN-Long Short-Term Memory (CLSTM):

CLSTM combines the strength of CNN and LSTM both to achieve a hybrid deep learning model [8]. CLSTM is effective in fraud detection because of its ability to capture both spatial and temporal patterns in transaction data. Local features such as transaction values are extracted from CNN while how the features evolve over time are modelled by the LSTM layer . The architecture consists of a convolution and pooling layer and a dropout between them which prevents overfitting. The result is given to the LSTM layer and finally to the dense layer which generates the final result [8]. It allows to detect complex fraud behaviours by learning both short-term and long term patterns [21].

1.4.4 Graph Neural Networks (GNN)

Graph neural networks is a type of deep learning technique in which nodes represent entities and relationships are represented by edges and it operates on graph structured data [9]. Most often there are unique patterns between Fraudulent accounts such as immediate transaction spikes and closely connected clusters and can be detected by GNNs. Embeddings are generated by combining information from neighboring nodes [9]. These embeddings carry complex individual and contextual behavior which results in more accurate fraud detection in Ethereum transactions.

1.4.5 Transformers for Tabular Data (FT-Transformer)

Transformers, initially developed for natural language processing, have been adapted for structured tabular data through models like the FT-Transformer. The FT-Transformer employs self-attention mechanisms to dynamically learn feature interactions without manual feature engineering, providing superior performance and interpretability in Ethereum fraud detection tasks.

CHAPTER 2
LITERATURE REVIEW

Several studies have investigated anomaly and fraud detection in blockchain environments.In this section we will be discussing relevant research and studies. Table 1. summarizes these key findings.

Kabla et al. provided a comprehensive review of the applicability of intrusion detection systems (IDS) on Ethereum-based attacks, categorizing vulnerabilities and emphasizing the need for IDS frameworks tailored to the Ethereum architecture [20]. However, they do not delve into the practical implementation or evaluation of machine learning models for fraud detection.

Mounnan et al. explored deep anomaly detection techniques for blockchain, offering a broad method of learning approaches like discriminative, generative, and hybrid models [4]. But their main focus was towards attacks at the protocol level and lacked transaction fraud specific to Ethereum network .

Saravanan et al. discussed federated learning models in enhancing anomaly detection on Ethereum networks [1] ,they discussed classic attacks and lacked analysis on evolved smart contract frauds or phishing scams .

Fouly et al. focuses on anomaly detection for blockchain using supervised, unsupervised and deep learning techniques[2], but offers limited insights into Ethereum specific implementations.

Tripathy et al. evaluated the performance of various supervised machine learning models such as Random Forest and XGBoost on Ethereum fraud detection dataset although achieving high performance [3], but solely focused on classification without exploring advanced approaches such as hybrid or graph-based models.

In this study we focus on Ethereum specific fraud detection strategies and also discusses challenges such as data imbalance , preprocessing techniques and discussion on various machine learning models as well as unique techniques like Graph Neural Networks (GNNs) and transformer-based language models providing a more holistic view in latest advancements.

Table 2.1 Summarizes various studies conducted in the field on fraud detection in blockchain networks.

TABLE 2.1 Summary of related work in Fraud Detection

Reference	**Year**	**Pros**	**Cons**
[20]	2022	Comprehensive review of Ethereum vulnerabilities and IDS strategies.	No implementation or evaluation of ML-based detection models.
[1]	2023	Proposes federated learning for blockchain anomaly detection using historical data.	Limited to Ethereum Classic and classic attack types.
[4]	2024	Provides deep anomaly detection taxonomy and method comparisons.	Focuses on low-level cyberattacks, not transactional fraud.
[2]	2024	Broad survey of ML techniques with practical applications across domains.	General overview lacking Ethereum-specific depth or new model proposals.
[3]	2024	Implements and evaluates multiple ML models (XGBoost, RF, etc.) on real Ethereum fraud data.	Focuses only on supervised learning; lacks hybrid or graph-based approaches.

Table 2.2 highlights the various research questions and Table 2.3 highlights whether various studies addressed these questions.

TABLE 2.2 Research Questions and areas of Focus in Ethereum Fraud Detection

Q.No	**Research Questions**
1	What are the different DL techniques employed in Ethereum fraud detection ?
2	Are challenges such as data imbalance, evolving fraud behavior, or real-time detection addressed?
3	What preprocessing techniques were described in the studies?
4	Is a multi field survey conducted in the Ethereum fraud detection studies ?

TABLE 2.3 Comparison of Research Questions Across Various Studies

Research Questions	**[1]**	**[2]**	**[3]**	**[4]**	**[20]**
RQ1	No	No	No	Yes	No
RQ2	Yes	Yes	Yes	Yes	No
RQ3	Yes	No	No	Yes	No
RQ4	Yes	No	No	Yes	Yes

Fig 2.1 Compares the frequency of various Blockchain fraud detection studies published between 2022 and 2024

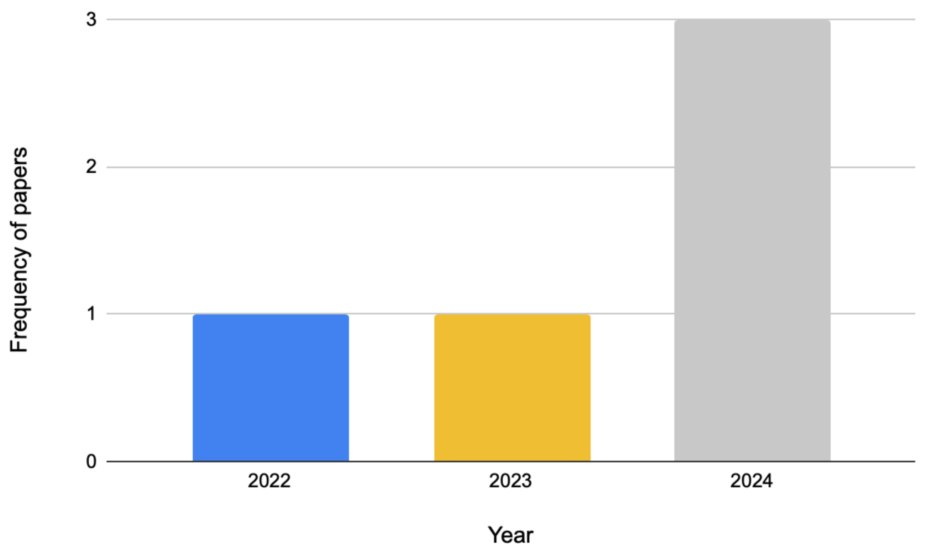


Fig 2.1 High Level flowchart of the process

Table 2.4 provides key information of various datasets used in the studies .

TABLE 2.4 Brief Summary of Publically Available Ethereum Datasets

Dataset names	**Used By**	**Link**	**Records**	**Description**
Ethereum Fraud Detection Dataset	[6],[11],[16]	https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset	9840	Contains records of fraudulent and legitimate transactions on Ethereum. Dataset is imbalanced.
Fraud Detection: ETHEREUM transactions	[7],[8],[12]	https://www.kaggle.com/code/sukantokumardas/fraud-detection-ethereum-transactions	9841	The dataset consists of 47 columns , including a ‘flag’ column indicating the nature of the transaction.
Fraud Detection in Ethereum Transactions	[17]	https://www.kaggle.com/code/unnatkmistry/fraud-detection-in-ethereum-transactions	12146	Dataset contains Ethereum transaction records with features such as timestamps, sender and receiver addresses, transaction amounts, and labels for fraudulent or non-fraudulent transactions.
Transaction Graph Dataset for the Ethereum Blockchain [18]	[18]	https://zenodo.org/records/3669937	2000000	Dataset contains ether and ERC20 token transfer transactions from the Ethereum Mainnet blockchain.

In table 2.5 we compare the performance of various DL and ML models used along with their pros and cons in this section.

TABLE 2.5 Comparison of performance of ML and DL used for Ethereum Fraud Detection

Reference	Year	Pros	Cons	Models Used	Result
[5]	2023	Reduced Feature Set	Dependent on transaction network data	Random Forest (RF) , Neural Network (NN) and K-nearest neighbor (KNN) [5]	Accuracy: Random Forest : 94% Neural Network: 92% KNN: 86%
[10]	2024	Addresses data imbalance	Complex architecture	Meta-IFD (Generative + Contrastive + GNN) Self-supervised Dual View Learning with GNN	Macro-F1: +14.7% for Ponzi; +2.02% for phishing over best ML baseline
[6]	2023	Handles Complex Patterns	Complex Training	Artificial Neural Network (ANN)	Accuracy: 97.1%
[7]	2024	Improved Optimization	Computationally intensive	Deep Neural Network with Genetic Algorithm metaheuristic optimization.	Accuracy: 98.6 F1-Score: 94.86
[13]	2024	ML techniques showed improved performance over naive rule-based methods	Unable to handle complex patterns due to using simple ML techniques.	Logistic Regression, Random Forest , KNN Classifier, AdaBoost Classifier	Accuracy: Logistic Regression: 61% Random Forest : 97% KNN Classifier : 90% AdaBoost Classifier: 96%
[14]	2024	Handles imbalanced datasets	Complex preprocessing.	Topological Filtering + VGG16 Transfer Learning	F1- Score: .9891
[11]	2023	Improved Accuracy	Limited to one Fraud	Ensemble CNN + LSTM (Bagging & Boosting)	Accuracy: 96.4% (Bagged LSTM)

[8]	2024	Detailed architecture tuning	High Training time and complexity	TabPFN, CLSTM, 1DCNN, LSTM, MLP	TabPFN: 99.2% Accuracy
[9]	2024	Improved performance	High resource consumption and training complexity.	Transaction Language Model + Multi-head Attention + GNN	F1-score: +10–20% over baselines Balanced Accuracy: +8–15%
[12]	2022	Fast and Efficient	Sensitive to Hyperparameters	LGBM	Accuracy LGBM: 98.60% Optimized LGBM: 99.03%
[16]		Handles class imbalance efficiently	Complex GAN training.	CTGAN + Cost-sensitive Learning + Ensemble Models (RF, LGBM, XGBoost, AdaBoost, DT)	Accuracy: 98%
[17]		highly efficient	computationally intensive	XGBoost (final), compared with CART, RF, LGBM	Accuracy: 96%
[18]		Improved prediction	Complex preprocessing on blockchain graphs	MLP, RF, DT, GB, KNN, SVM	Accuracy:.97%

CHAPTER 3
METHODOLOGY

To solve the problem of Ethereum fraud detection, this work proposes an end-to-end approach that combines data preprocessing, feature selection, and a deep learning model. The full framework, starting from input data and ending in fraud prediction, is shown in Figure 4.2. Each part of the system is explained more in the following subsections.

3.1 Dataset and Feature Description

We used the publicly available Ethereum fraud detection dataset provided by Vagifa Abilova on Kaggle [source]. It is a tabular collection of Ethereum addresses, with a wide set of behavioral and statistical features. It contains 9,841 addresses, where each address is labeled with a binary FLAG value. If address was involved in fraud, the label is 1, otherwise it is 0. After removing unnecessary columns, there are around 46 features for each address. These features are basis on transaction history and interaction with smart contracts. They covered different types of behavior, which are grouped as follows:

Transaction Time Features: These include values like average time between sent transactions, average time between received transactions, and the time difference between first and last transaction.How often the addresses were active can be understood by these features.Such features helps to understand when and how often the address been active. Fraudulent addresses sometimes showing strange or unusual pattern of activity, which these features are able to point out.

Transaction Count Features: This part tells how many actions the address do. Like, total number of transaction, how many time it send or receive money, and how many smart contracts it did create. These values can shows if the address was too active and doing things in weird ways, like sending too many transactions or creating lots of contracts repeatedly.

Value Features (Ether and Tokens): These include total Ether sent or received, Ether sent to contracts, and the current Ether balance. Similar features are included for ERC-20 token activity, like how much tokens were sent or received, and their minimum, maximum, and average values. These are important to measure the volume of money going through the address. In fraud cases, you might see extreme values—like very large token movements in Ponzi-type scams or many small token transfers in phishing attempts.

Unique Interaction Features: This includes things like how many unique addresses send Ether to this address, or how many other addresses it has sent Ether to. Also, it looks at how many different types of tokens were sent or received. These help measure how wide or diverse the address interactions are. For example, an address used in phishing might receive Ether from many different victims, show high number of unique senders. In case of Ponzi, the address might have unusual fan-in and fan-out behavior, which stands out from normal ones.

All features in the dataset are numeric or converted into numeric format. Identifiers that are not numeric, like the wallet address string or token names, are either dropped or changed into numbers. For example, the ERC20 most sent token type was removed because it's a category, but the number of unique token names sent was kept, since it's a count value and can be used as numeric.By including so many different kinds of features, the model can look at both big-picture behaviors (like totals and counts) and smaller patterns (like timing or min/max values). But because the feature space becomes very large, it's important to do good preprocessing and feature selection, which is explained in the next part.

3.2 Data Preprocessing

Doing proper data preprocessing is important to make sure the features are in right shape before training the model. In this work, we applied several steps to clean and prepare the raw dataset before it goes into the learning algorithm.

Data Cleaning: First step is to remove columns which doesn't really add anything useful for the learning. In this dataset, things like Unnamed: 0, Index, and Address columns are removed. These are mostly just identifiers, and they don't help in predicting anything. They don't show any behaviour or activity, so keeping them only adds extra data that model don't need.Also, textual fields such as the most frequent token type names are dropped, and we keep focus only on features which are numeric.

Type Conversion: All the feature values are forced into numeric format. In cases where values are non-numeric or missing (for example, strings like "None"), they are turn into NaN (Not a Number). This help to make sure that the model receives clean and consistent numerical inputs for every feature.

Missing Value Imputation: In some cases, few addresses might not have done certain kinds of activities at all. Like, if one address never received any ERC20 tokens, then features such as ERC20 max value received will just stay empty — that is, as NaN. To handle this, we followed a two-step method. First step is to go through the data and replace all infinite values (both +Inf and -Inf) with NaN as well. Then in second step, we take each feature one by one and calculate the median — but only using the training set, so test data doesn't leak into it. All NaN values in that feature are filled with this median. Median was selected because it is less sensitive to outliers and usually makes more sense for skewed data. If some feature still has NaN values even after that (which only happens when the whole column is empty in training), then we just use 0 to fill those. That way, there's no missing data left, and at the same time, we don't introduce very large or unusual values into the model.

Feature Transformation: Many of the features exhibit highly skewed distributions. For instance, features like total Ether received or max value sent can span several orders of magnitude (some addresses have transacted tiny amounts, while others have handled massive Ether volumes). To reduce skewness and stabilize variance, we apply a logarithmic transformation to high-magnitude features. Specifically, for any feature whose maximum value is greater than 100 (a heuristic threshold indicating a wide value range). A log(1 + x) transformation is applied to reduce the effect of extreme values, where log denotes the natural logarithm.

Feature Scaling: After the above steps, we perform feature scaling to normalize the feature ranges. We use standardization, i.e., z-score scaling, which shift feature to mean 0 and standard deviation 1. This is done using StandardScaler from scikit-learn, fitted on the training data and apply to all sets. Scaling is particularly important for our chosen model (FT-Transformer) to ensure that the optimization process is well-behaved and that features are on comparable scales when computing distances or dot-products in the neural network. It also benefits any distance-based interpretation and accelerates convergence during training.

After preprocessing, the processed dataset is split into training, validation, and test sets. We use stratified split to maintained the same fraud ratio in each subset. 70% of the data is used for train, and remaining 30% is further split evenly into 15% validation and 15% test. The training set is used to fit the model, the validation set is used for tuning and feature selection, and the test set is reserved for final performance evaluation.

3.3 Feature Selection with Recursive Feature Elimination (RFE)

When dataset has too many features, like in our case, not all of them help much in finding fraud. Some features say kind of the same thing again, or just bring noise, which makes it hard for model to learn properly or make it slow and complex. So, to find which features actually useful, we used something called as Recursive Feature Elimination (RFE).

RFE is a method where a model is trained again and again, and each time it removes the feature that seems least useful — based on things like weight or score from the model. This keeps happening step by step, until only a fixed number of features are left. In our case, we used RFE with training data, and picked a basic model like logistic regression or sometimes random forest to do this. After ranking the features, we kept the top 'k' ones that gave better results when we checked with validation set. During our tests, we found that using only around 20 features was enough to keep most of the useful info. These important features included things like total ERC20 token transactions, the time gap between first and last transaction, total Ether received, number of different addresses that sent tokens, and total number of transactions. This kind of makes sense — for example, scam addresses usually show very high number of token transfers in phishing, or have long time gaps in Ponzi-like activity where funds keep flowing in over time.

By cutting down the feature list, the model becomes simpler, runs faster, and sometimes even performs better. It also helps us understand things more easily, since we only look at the most meaningful features. Later in this work, we also compare how the model performs when we use all features versus when we only use the ones selected by RFE, to see how much difference it makes.

3.4 Model Architecture: FT-Transformer for Tabular Data

The main part of the proposed Ethereum fraud detection system is a deep learning model based on transformer structure, inspired by FT-Transformer. This model is specially made to work well with tabular data. The architecture is designed in a way that it can capture complex relationships between different transaction features, while still staying efficient and not too hard to understand. It includes different parts like feature preprocessing, tokenizing and embedding of features, transformer encoder blocks, and finally a classification head. The model is trained using a loss function that is aware of class imbalance, which helps in detecting frauds more accurately, especially when fraud examples are much less than normal ones.

3.4.1 Input Layer

The model takes in 20 numeric features as input, which are selected using Recursive Feature Elimination (RFE). These features picked from Ethereum transaction dataset and found to be most important in fraud identification. Each feature about some kind of behavior or transaction activity of an Ethereum address, like how much Ether was received, how often the address was active, or how many ERC20 tokens were moved.

3.4.2 Feature Tokenization and Embedding

Every one of the 20 input values is passed through a learnable embedding layer. This converts the raw number into a dense vector (for example, 64 dimensions). So now, instead of simple numbers, the model works with richer vector representations. This helps it understand not just individual feature values, but also how they relate with each other in deeper ways. Along with that, we add positional embeddings to each feature token, even if the order of features doesn't mean anything directly. This way, the model can still tell features apart during training.

3.4.3 Transformer Encoder Blocks

After embedding, the feature tokens are passed into a stack of transformer encoder layers. Each of these blocks contains multiple parts — like multi-head self-attention, feedforward neural networks (FFN), dropout, and layer normalization. The self-attention helps the model to understand how features relate to each other in context. For example, when there is a high number of ERC20 transfers together with short time gaps between transactions, it may point toward fraud activity. These encoder blocks allows the model to learn such patterns through deep and non-linear relationships in efficient way.

3.4.4 Attention Pooling

Once the data goes through all the transformer layers, the output from all the tokens are combined. In our case, we do this by flattening the transformer output directly into the classifier. This is called attention pooling in some works, but here it's done in a simple way. The idea is to summarize everything the model has learned from all the features into one compact vector that reflects the behavior of the address overall.

3.4.5 Classification Head

This final step is where the model makes the actual prediction. The pooled vector is sent through a two-layer feedforward neural net. First layer takes the 64-dimensional input and reduces it to 32, with a ReLU activation and dropout applied to avoid overfitting. The second layer gives out a single number (logit), which then goes through sigmoid to make it a probability. That final value tells how likely the address is doing fraud.

3.4.6 Loss Function and Class Imbalance Handling

To deal with the class imbalance in this fraud detection task (since only around 22% of the addresses are marked as fraud), the model is trained using Binary Cross-Entropy with Logits Loss. We also include a pos_weight parameter to give more focus to the fraud class during training. This helps the model to pay more attention catching fraudulent samples, which improve recall while keeping precision at decent level.

To sum up, the model brings together the powerful capability of transformers with smart feature selection and a method to handle class imbalance. Each tabular feature is treated as a separate token, and with self-attention, the FT-Transformer can learn deep and complex connections between different features. These kinds of relationships are often key in detecting fraud. The model is not only able to scale well with data but also gives good performance while still being somewhat interpretable. This makes it a suitable choice for detecting fraud in Ethereum addresses with high precision and recall.

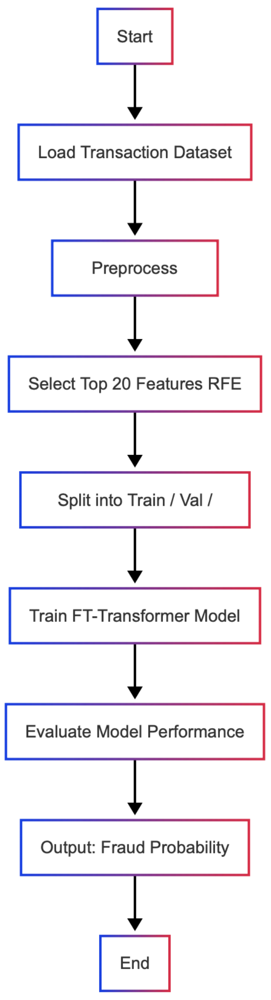
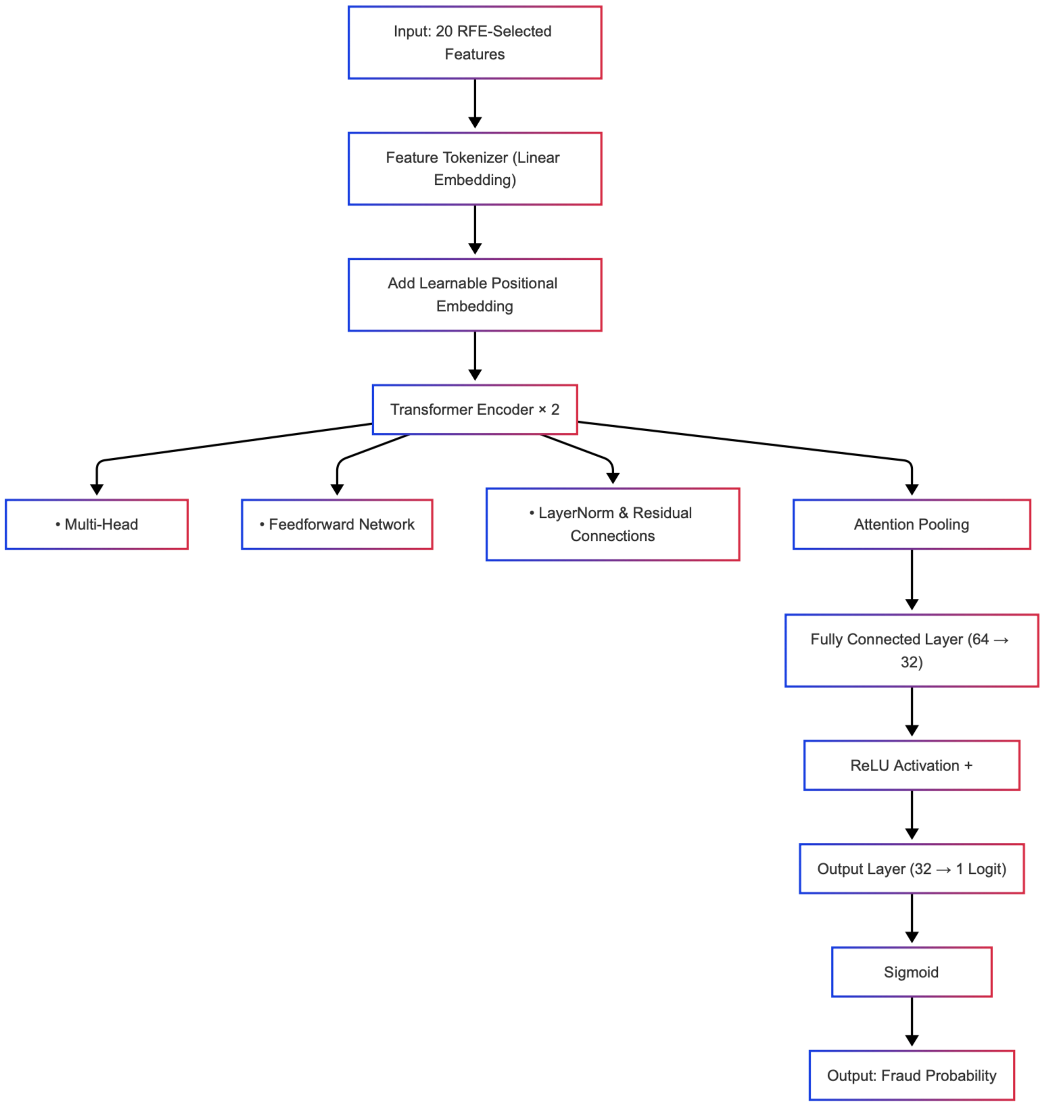

Start
Load Transaction Dataset
Preprocess
Select Top 20 Features RFE
Split into Train / Val /
Train FT-Transformer Model
Evaluate Model Performance
Output: Fraud Probability
End

Fig 3.1 High Level flowchart of the process



Input: 20 RFE-Selected Features
Feature Tokenizer (Linear Embedding)
Add Learnable Positional Embedding
Transformer Encoder × 2
• Multi-Head
• Feedforward Network
• LayerNorm & Residual Connections
Attention Pooling
Fully Connected Layer (64 → 32)
ReLU Activation +
Output Layer (32 → 1 Logit)
Sigmoid
Output: Fraud Probability

Fig 3.2 FT-Transformer Architecture for Ethereum Fraud Detection.

3.5 Training Strategy and Imbalance Handling

3.5.1 Training Procedure:

We trained the FT-Transformer model using the cleaned training data and the training was carried out for a fixed number of epochs — 20 in our case — and we make use of the Adam optimizer with an initial learning rate of 0.001 for updating the model weights during training. The data was divided into batches of size 64, and after processing each batch, the model parameters got updated to reduce the loss. While training, we also kept checking how the model was doing on the validation set, so we can avoid overfitting. If the validation loss started increasing or not improving, it would give us a signal to stop early or adjust the learning settings. If needed, we apply early stopping or reduce the learning rate depending on how validation loss behaves. The choice of hyperparameters like learning rate, number of epochs, and batch size was based on initial experiments done using the validation data. After training is finished, we pick the model version (or checkpoint) that gave the best results on validation, and then we use that one to test final performance.

3.5.2 Loss Function and Class Imbalance:

As mentioned earlier, class imbalance is a serious challenge in this task, since only around 22% of the addresses are marked as fraudulent. If no correction is applied, then the normal binary cross-entropy loss becomes biased toward the majority class, and the model ends up predicting "non-fraud" most of the time just to reduce the total loss. This kind of behavior makes the model less useful for identifying actual fraud cases.

To fix this, we use a weighted version of the Binary Cross Entropy loss, known as BCEWithLogitsLoss, where we set the pos_weight parameter to increase the penalty for misclassifying fraudulent cases.

In our dataset, since fraud samples make up about 22%, we use a pos_weight value of roughly 3.5. This means the model treats a mistake on a fraud address as 3.5 times more serious than a mistake on a benign one. This forces the model to pay more attention to the minority class during training. With this setting, we observed the model performed better in terms of balancing precision and recall, compared to when the loss was unweighted. This is a cost-sensitive learning approach, which works better than resampling methods in many cases — because it keeps the training data as-is, while only changing how the model learns from mistakes.

3.5.3 Evaluation during Training:

We evaluates the model on the validation set after each epoch (or in some cases, after few epochs) using metrics like validation loss and F1-score, which considers both precision and recall. Selecting the best version of model — usually the one having highest validation F1 or lowest loss — and also helps deciding when to stop the training early if validation metrics not improving anymore. While doing evaluation, the model's output (which is a probability after applying sigmoid) is by default thresholded at 0.5 for calculating classification metrics. But in some cases, we changes this threshold depending on precision-recall curve, especially if getting higher recall is more important than precision.

With the whole methodology now explained — from preprocessing and selecting features to model architecture and training steps — we proceeds to applying this framework on the Ethereum fraud dataset. The next section presenting the results of our implementation, with comparison between models with and without feature selection, and some discussion about how well the deep learning approach worked for fraud detection.

comparing performance with and without feature selection and analyzing the effectiveness of our deep learning based approach.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Experimental Setup

The proposed framework was done using Python, with commonly used libraries like pandas for managing data, scikit-learn for preprocessing and selecting features, and PyTorch to build and train the FT-Transformer model. All the experiments were carried out on a system that had GPU support — in this case, an NVIDIA Tesla T4 GPU on Google Colab — which helped in making the training of the neural net faster. The system was also having an Intel(R) Xeon(R) CPU @ 2.20GHz and 12 GB RAM, which was enough to load the full dataset into memory for smooth processing. Using the GPU helped a lot in making the transformer model train faster, especially because attention operations involves large matrix computations that can run in parallel.

During the training phase, we was monitoring performance on both training and validation sets in every epoch. Early stopping were applied based on validation loss — if it didn't improve for few epochs, then training gets stopped early to avoid overfitting. In practice, the model was usually converging somewhere between 15 to 20 epochs, as the dataset wasn't too large to begin with.

4.2 Evaluation Metrics

The effective evaluation of models for Ethereum fraud detection models need effective performance metrics . They should take in consideration both the predictive power and class imbalance. We will briefly discuss the following metrics used to assess the models performance:

Accuracy: It measures the models ability to correctly predict the true labels out of all the predictions. Ratio of the correct predictions to the total number of predictions is known as accuracy [25]. In the context of Ethereum fraud detection , accuracy may be misleading due to the imbalanced nature of the dataset.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ negatives + False\ Positives + False\ Negatives}$$

Precision: the proportion of predicted frauds that are actually frauds. This metric tells us how often the model is correct when it raises an alarm. High precision means few false positives (innocent addresses flagged incorrectly).

Recall: It is the proportion of actual frauds that the model correctly identified. This measures the model's ability to catch the frauds. High recall means few false negatives (fraudulent addresses missed by the model).

F1-Score: It is the harmonic mean of the precision and recall where precision measures how many of the "positive" predictions made were actually correct and recall is the measure of how many actual positive classes were correctly identified by the model. This evaluation metric is preferred when the data is imbalanced in nature as it provides a balance between the precision and recall [25].

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

ROC AUC (Area Under the Receiver Operating Characteristic Curve): AUC measures the model's ability to discriminate between the classes across all classification thresholds. It is threshold-independent and summarizes the trade-off between true positive rate and false positive rate. An AUC of 1.0 represents a perfect classifier, while 0.5 is no better than chance. AUC is particularly informative in imbalanced settings as well, since it considers the ranking of predictions.

In addition to these, we also examined the **confusion matrix** of the final model to see the breakdown of true positives, true negatives, false positives, and false negatives, and **balanced accuracy** (which is the average of recall on fraud and recall on non-fraud) . However, for brevity, we focus on the main metrics listed above in our discussion.

All metrics are reported on the **test set**, which was held out from the training process and not used for model tuning, to give an unbiased evaluation of how the model would perform on new, unseen data.

4.3 Results with RFE-Selected Features

Using 20 features selected through Recursive Feature Elimination (RFE), the model was able to reach strong performance across all major evaluation metrics. This shows that the selected features are quite informative for detecting fraud in Ethereum addresses.

Accuracy of 96.89% means the model classified nearly 97 out of every 100 addresses correctly, which reflects overall high prediction quality.

Precision is around 90.03%, which tells that whenever model predicts an address is fraud, it's right almost 9 out of 10 times. That helps reduce false positives. Because if it keeps saying something is fraud when it's not, that would create lot of noise and make the system less usable.

Recall came to be 96.64%, so it means the model was able to catch most of the fraud addresses. Missing fraud is something you don't want in such tasks, especially in real systems, because even few slipping through can be a problem.

F1 score was 93.22% — which is kind of the balance between precision and recall. So, it shows that model isn't just being correct sometimes, but also not missing much. Both things are working together here.

Lastly, the **AUC score** reached 0.9970, which is almost near perfect. That means the model is able to tell difference between fraud and non-fraud addresses very well, no matter what cutoff value you use to make the final decision.

In **Table 6** we summarize the results of our fraud detection model.

TABLE 4.1 Summary of performance of fraud detection model

Metric	**Value**
Accuracy	96.89%
Precision	90.03%
Recall	96.64%
F1 Score	93.22%
AUC (ROC)	99.70%

To sum up, the model which was trained using 20 features selected by Recursive Feature Elimination (RFE) were giving very strong results across all of the evaluation metrics. These features was proving to be very useful for catching frauds on the Ethereum network. The accuracy come out as 96.89%, and precision was 90.03%, meaning that model was right most of times when it predict a fraudulent address. Recall was 96.64%, which shows it didn't miss much of actual fraud cases. The F1-score came 93.22%, showing model had a decent balance between catching frauds and not giving too many false alarms.Also, the AUC was extremely high at 0.9970, which proves the model is very good at separating fraud and non-fraud addresses.

These numbers show that the model can work really well even with fewer features — less than half of the original ones — without losing much in terms of how well it detects fraud. Having a smaller feature set makes the model simpler, faster, and easier to understand. This is especially useful in real-time fraud monitoring systems, where fast and clear results are important. It also makes it easier for analysts to look at why the model made a decision, since they only have to focus on fewer, more meaningful features.Some of the most useful features included in the final set are things like ERC20 transaction counts, how many different tokens were used, how much Ether was received, and the average time between received transactions. These match up with known fraud patterns — attackers often deal with lots of addresses, use tokens to hide movements, and show timing behavior that doesn't look normal. So overall, the RFE-based model not only performs well but also makes the whole system more efficient and practical for real-world fraud detection.

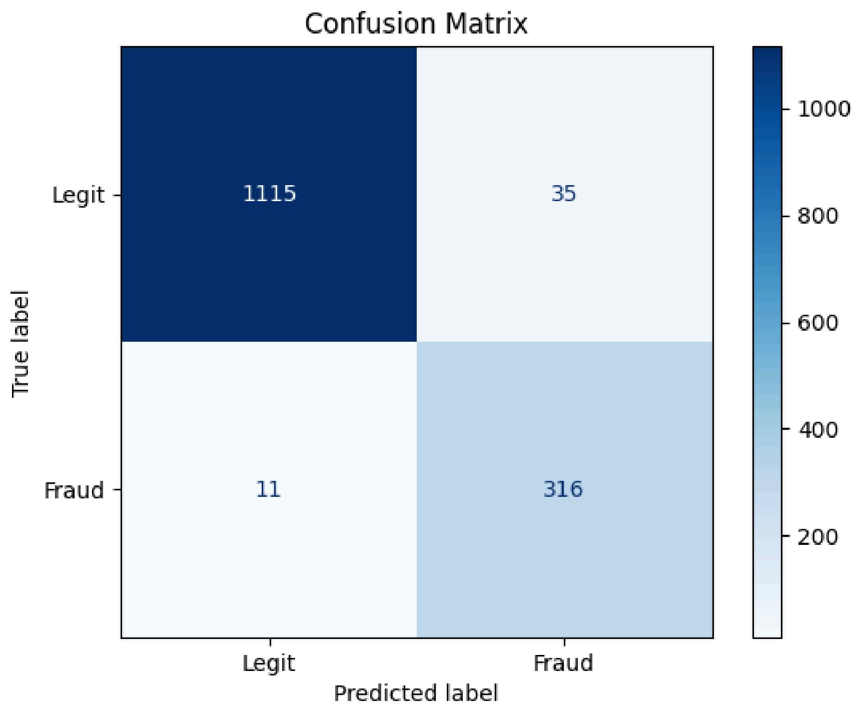


Fig 4.1 Confusion Matrix for the fraud detection model

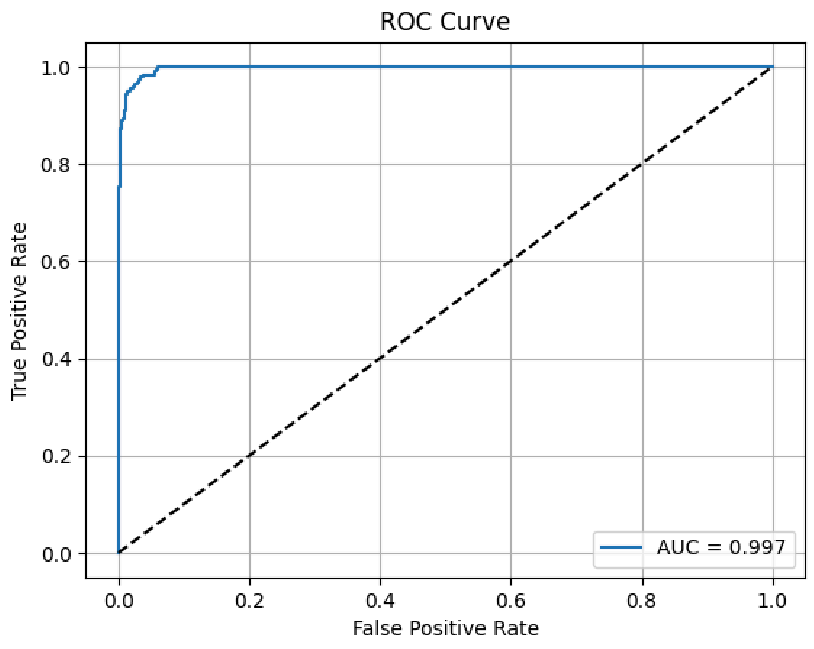


Fig 4.2 ROC curve for the fraud detection model

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

The proposed Ethereum fraud detection framework, which is based on deep learning and machine learning, shows how transformer-based models can work effectively in identifying fraudulent addresses on blockchain. By combining strong data preprocessing, recursive feature selection, and the FT-Transformer architecture, the model was able to capture complex interactions between features that are present in transactional data. The results showed that FT-Transformer gave good performance in terms of accuracy, precision, recall, and AUC, even when trained using a smaller number of selected features. This proves the robustness of the model and makes it suitable for high-dimensional fraud detection tasks on Ethereum. Overall, the work supports the idea that modern deep learning models can work well for tabular blockchain data and gives a scalable and flexible solution that helps improve current fraud detection systems.

Looking forward, there are still few improvements and directions that can help to expand and make the method more effective. One is to add graph-based learning like Graph Neural Networks (GNNs), which can help to learn the relationship patterns among addresses in the network. This can increase the model's ability to detect frauds that are done in a coordinated way. Also, time-based behavior could be modeled more directly using LSTM or Transformer-based sequence models. This would help the model understand how fraud activity happens step-by-step over time. Instead of just binary classification, multi-class fraud detection can also be useful, so the system can tell different types of frauds apart and maybe apply different rules or actions for each type. Another idea is to use unsupervised learning, like autoencoders or variational methods, to detect frauds that were not labeled or seen before. Class imbalance can be further improved by doing data augmentation — like using SMOTE or by generating synthetic transactions. Lastly, deploying the model in real time, along with explainability tools such as SHAP values or visualizing attention weights, would make it more usable in real-world environments. These steps can help build a stronger and more practical Ethereum fraud detection system that works better in different situations.

REFERENCES

[1] R. Saravanan, S. Santhiya, K. Shalini and V. S. Sreeparvathy, "Comparative Study Analysis of MachineLearning Algorithms for Anomaly Detection in Blockchain," *2023 International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, Ballar, India, 2023, pp. 1-6, doi: 10.1109/ICDCECE57866.2023.10150785.

[2] Fouly, Mohamed & Soliman, Taysir & Taloba, Ahmed. (2024). Machine Learning Techniques for Detecting Abnormal Behaviors in Blockchain Technologies: A Methodological Review. 1-6. 10.1109/ICCA62237.2024.10927796.

[3] N. Tripathy, K. S. Chaudhury, A. Nibedita, S. K. Nayak, M. Behera and B. Jena, "Ethereum Fraud Detection: A comparative analysis of supervised learning approach," *2024 IEEE 1st International Conference on Advances in Signal Processing, Power, Communication, and Computing (ASPCC)*, Bhubaneswar, India, 2024, pp. 1-6, doi: 10.1109/ASPCC62191.2024.10881143.

[4] Mounnan, O., Manad, O., Boubchir, L., Mouatasim, A. E., & Daachi, B. (2024). A review on deep anomaly detection in Blockchain. *Blockchain Research and Applications*, 100227. https://doi.org/10.1016/j.bcra.2024.100227

[5] Onu, I.J., Omolara, A.E., Alawida, M. *et al.* Detection of Ponzi scheme on Ethereum using machine learning algorithms. *Sci Rep* **13**, 18403 (2023). https://doi.org/10.1038/s41598-023-45275-0

[6] M. Dahiya, N. Mishra, R. Singh and Pavitra, "Neural network based approach for Ethereum fraud detection," *2023 4th International Conference on Intelligent Engineering and Management (ICIEM)*, London, United Kingdom, 2023, pp. 1-4, doi: 10.1109/ICIEM59379.2023.10166745.

[7] A. Srivastava, A. Kumar, A. Pillai and V. S. Sharma, "Enhanced Fraud Detection in Ethereum Transactions: Fusion of Modified Genetic Algorithms and Deep Learning with Limited Attributes," *2024 International Conference on Signal Processing and Advance Research in Computing (SPARC)*, LUCKNOW, India, 2024, pp. 1-5, doi: 10.1109/SPARC61891.2024.10829277.

[8] Olusegun, Ruth & Yang, Bo. (2024). Improved Ethereum Fraud Detection Mechanism with Explainable Tabular Transformer Model. 59-68. 10.1109/TPS-ISA62245.2024.00017.

[9] Jia, Yifan & Wang, Yanbin & Sun, Jianguo & Liu, Yiwei & Sheng, Zhang & Tian, Ye. (2024). Ethereum Fraud Detection via Joint Transaction Language Model and Graph Representation Learning. 10.48550/arXiv.2409.07494.

[10] C. Jin, J. Zhou, C. Xie, S. Yu, Q. Xuan and X. Yang, "Enhancing Ethereum Fraud Detection via Generative and Contrastive Self-Supervision," in *IEEE*

Transactions on Information Forensics and Security, vol. 20, pp. 839-853, 2025, doi: 10.1109/TIFS.2024.3521611.

[11] Q. Umer, J. -W. Li, M. R. Ashraf, R. N. Bashir and H. Ghous, "Ensemble Deep Learning-Based Prediction of Fraudulent Cryptocurrency Transactions," in *IEEE Access*, vol. 11, pp. 95213-95224, 2023, doi: 10.1109/ACCESS.2023.3310576.

[12] Aziz, Rabia & Baluch, Mohammed Farhan & Patel, Sarthak & Ganie, Abdul. (2022). LGBM: a machine learning approach for Ethereum fraud detection. International Journal of Information Technology. 14. 10.1007/s41870-022-00864-6.

[13] Tripathy, Nrusingha & Balabantaray, Sidhanta & Parida, Surabi & Nayak, Subrat. (2024). Cryptocurrency fraud detection through classification techniques. International Journal of Electrical and Computer Engineering (IJECE). Vol. 14, No. 3, June 2024, pp. 2918~2926 ISSN: 2088-8708, DOI: 10.11591/ijece.v14i3. 2918-2926. 10.11591/ijece.v14i3.pp2918-2926.

[14] Nakatani, Shunsuke & Kuzuno, Hiroki & Takita, Makoto & Mohri, Masami & Shiraishi, Yoshiaki. (2024). Can We Determine Whether a Set of Ethereum Transaction Data Contains Fraudulent Transactions?. 108-114. 10.1109/DSC63325.2024.00023.

[15] Nadella, Geeta & Meduri, Karthik & Gonaygunta, Hari & Satish, Snehal & e Vadakkethil Somanathan Pillai, Sanjaikanth. (2024). Blockchain Fraud Detection Using Unsupervised Learning: Anomalous Transaction Patterns Detection Using K-Means Clustering. 407-412. 10.1145/3675888.3676080.

[16] P. Saket, P. Jyothi, A. B. Venkata Ayush Patnaik, N. Chaithanya Vardhan Reddy and S. Suresh, "Cost Sensitive Approach to Ethereum Transactions Fraud Detection using Machine Learning," *2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Bhilai, India, 2024, pp. 1-8, doi: 10.1109/ICAECT60202.2024.10469665.

[17] Rathore, Muhammad Mazhar & Chaurasia, Sushil & Shukla, Dhirendra & Anand, P.. (2023). Detection of Fraudulent Entities in Ethereum Cryptocurrency: A Boosting-based Machine Learning Approach. 6444-6449. 10.1109/GLOBECOM54140.2023.10437184.

[18] B. Kılıç, A. Sen and C. Özturan, "Fraud Detection in Blockchains using Machine Learning," *2022 Fourth International Conference on Blockchain Computing and Applications (BCCA)*, San Antonio, TX, USA, 2022, pp. 214-218, doi: 10.1109/BCCA55292.2022.9922045.

[19] Y. Zhang, S. Kang, W. Dai, S. Chen and J. Zhu, "Code Will Speak: Early detection of Ponzi Smart Contracts on Ethereum," *2021 IEEE International*

Conference on Services Computing (SCC), Chicago, IL, USA, 2021, pp. 301-308, doi: 10.1109/SCC53864.2021.00043.

[20] A. H. H. Kabla *et al*., "Applicability of Intrusion Detection System on Ethereum Attacks: A Comprehensive Review," in *IEEE Access*, vol. 10, pp. 71632-71655, 2022, doi: 10.1109/ACCESS.2022.3188637.

[21] R. Olusegun, T. Oladunni, H. Audu, Y. Houkpati and S. Bengesi, "Text Mining and Emotion Classification on Monkeypox Twitter Dataset: A Deep Learning-Natural Language Processing (NLP) Approach," in *IEEE Access*, vol. 11, pp. 49882-49894, 2023, doi: 10.1109/ACCESS.2023.3277868

[22] Goodfellow, Ian & Pouget-Abadie, Jean & Mirza, Mehdi & Xu, Bing & Warde-Farley, David & Ozair, Sherjil & Courville, Aaron & Bengio, Y.. (2014). Generative Adversarial Networks. Advances in Neural Information Processing Systems. 3. 10.1145/3422622.

[23] Mohammed, Roweida & Rawashdeh, Jumanah & Abdullah, Malak. (2020). Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. 243-248. 10.1109/ICICS49469.2020.239556.

[24] Ma, W., Gou, C., & Hou, Y. (2023). Research on Adaptive 1DCNN Network Intrusion Detection Technology Based on BSGM Mixed Sampling. *Sensors*, *23*(13), 6206. https://doi.org/10.3390/s23136206

[25] Rainio, O., Teuho, J. & Klén, R. Evaluation metrics and statistical tests for machine learning. *Sci Rep* 14, 6086 (2024). https://doi.org/10.1038/s41598-024-56706-x

Thesis_Aditya.docx

Delhi Technological University

Document Details

Submission ID	trn:oid:::27535:97822831
Submission Date	May 26, 2025, 5:47 PM GMT+5:30
Download Date	May 26, 2025, 5:50 PM GMT+5:30
File Name	Thesis_Aditya.docx
File Size	413.9 KB

35 Pages

7,930 Words

44,373 Characters

12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Cited Text

Match Groups

92 Not Cited or Quoted 12%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

7%	Internet sources
7%	Publications
9%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

92 Not Cited or Quoted 12%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

7%	Internet sources
7%	Publications
9%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	**www.mdpi.com**	<1%
2	Publication	**Poonam Nandal, Mamta Dahiya, Meeta Singh, Arvind Dagur, Brijesh Kumar. "Pro...**	<1%
3	Publication	**D. Jeya Mala, Anto Cordelia Tanislaus Antony Dhanapal, Saurav Sthapit, Anita Kha...**	<1%
4	Internet	**peerj.com**	<1%
5	Internet	**pdffox.com**	<1%
6	Internet	**doctorpenguin.com**	<1%
7	Submitted works	**University of Reading on 2024-04-22**	<1%
8	Internet	**fastercapital.com**	<1%
9	Internet	**www.ubishops.ca**	<1%
10	Internet	**vbmv.org**	<1%

11 Submitted works

University of Southampton on 2023-09-14 <1%

12 Submitted works

Indus International School on 2024-03-05 <1%

13 Submitted works

University of Auckland on 2024-08-20 <1%

14 Submitted works

University of Surrey on 2024-09-19 <1%

15 Submitted works

Glyndwr University on 2025-04-18 <1%

16 Submitted works

Academy of Information Technology on 2025-05-20 <1%

17 Submitted works

Associatie K.U.Leuven on 2025-05-21 <1%

18 Publication

Ruth Olusegun, Timothy Oladunni, Halima Audu, Yao Houkpati, Staphord Bengesi... <1%

19 Internet

blockchaindose.com <1%

20 Internet

www.geeksforgeeks.org <1%

21 Publication

Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelli... <1%

22 Publication

Thangaprakash Sengodan, Sanjay Misra, M Murugappan. "Advances in Electrical ... <1%

23 Internet

genomemedicine.biomedcentral.com <1%

24 Internet

arxiv.org <1%

25	Internet	
www.oecd.org		<1%
26	Submitted works	
University of South Florida on 2024-10-31		<1%
27	Internet	
dspace.daffodilvarsity.edu.bd:8080		<1%
28	Internet	
repository.riteh.uniri.hr		<1%
29	Submitted works	
Liverpool John Moores University on 2021-02-01		<1%
30	Submitted works	
UCL on 2025-04-25		<1%
31	Submitted works	
University of Stellenbosch, South Africa on 2025-02-11		<1%
32	Publication	
"Blockchain and Trustworthy Systems", Springer Science and Business Media LLC,...		<1%
33	Submitted works	
Southampton Solent University on 2025-01-10		<1%
34	Submitted works	
VIT University on 2025-04-22		<1%
35	Submitted works	
Canadian University of Dubai on 2023-11-12		<1%
36	Submitted works	
Leeds Beckett University on 2024-05-04		<1%
37	Publication	
M. Sabari Ramachandran, S. Sajithabanu, A. Ponmalar, M. Mohamed Sithik, A. Jos...		<1%
38	Submitted works	
Morgan State University on 2025-05-12		<1%

39 Submitted works
University of Duhok on 2024-11-27 <1%

40 Submitted works
University of Sheffield on 2017-04-19 <1%

41 Publication
V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challeng... <1%

42 Internet
dspace.bracu.ac.bd <1%

43 Publication
"Innovations and Advances in Cognitive Systems", Springer Science and Business ... <1%

44 Submitted works
Cardiff University on 2019-10-10 <1%

45 Submitted works
Liverpool John Moores University on 2023-02-22 <1%

46 Submitted works
The University of Manchester on 2025-05-05 <1%

47 Submitted works
Tilburg University on 2025-05-19 <1%

48 Submitted works
University of Hertfordshire on 2025-04-26 <1%

49 Submitted works
University of Lincoln on 2022-05-19 <1%

50 Submitted works
University of Stirling on 2024-09-09 <1%

51 Submitted works
University of Stirling on 2024-12-09 <1%

52 Submitted works
University of the Pacific on 2025-05-07 <1%

53	Internet	
acikerisim.ikcu.edu.tr		<1%
54	Internet	
bmcmedimaging.biomedcentral.com		<1%
55	Internet	
dokumen.pub		<1%
56	Internet	
dr.ntu.edu.sg		<1%
57	Internet	
ijece.iaescore.com		<1%
58	Internet	
jis-eurasipjournals.springeropen.com		<1%
59	Internet	
mdpi-res.com		<1%
60	Submitted works	
Indian Institute of Technology Guwahati on 2023-04-30		<1%
61	Publication	
Salwa Belaqziz, Salma El Hajjami, Hicham Amellal, Redouan Lahmyed, Lahcen Kou...		<1%
62	Submitted works	
University of Essex on 2023-08-25		<1%
63	Submitted works	
UCL on 2025-05-02		<1%

Thesis_Aditya.docx

Delhi Technological University

Document Details

Submission ID	trn:oid:::27535:97822831
Submission Date	May 26, 2025, 5:47 PM GMT+5:30
Download Date	May 26, 2025, 5:49 PM GMT+5:30
File Name	Thesis_Aditya.docx
File Size	413.9 KB

35 Pages

7,930 Words

44,373 Characters

*% detected as AI

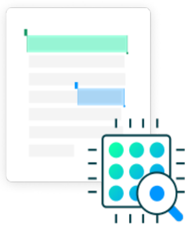
AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.



False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.