UTILIZING MACHINE LEARNING AND NEURAL NETWORK MODELS FOR DETECTING ILLICIT TRANSACTIONS IN ETHEREUM BLOCKCHAIN

MADHUMITHA RAMAJEYATHILAGAM

Final Thesis Report

March 2023

TABLE OF CONTENTS

DEDICATION

I dedicate this thesis to my family and friends, who have supported and motivated me through all of my life's challenges and encouraged me to accomplish my goals.

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to Dr. Nilam Upasani, who oversaw my thesis project, for her unwavering support and helpful advice. I would like to express my sincere gratitude to the professors at IIIT-Bangalore and Liverpool John Moores University as well as the teaching staff at UpGrad, UpGrad student mentors, and all UpGrad support personnel for their unending assistance. Last but not least, I'd like to thank everyone who has supported me in finishing this project.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

Acronym	Abbreviations
AdaBoost	Adaptive Boosting
Al-SPSD	anti-leakage smart Ponzi schemes detection
AUC	Area under the ROC curve
CART	Classification and Regression Tree
CNN	Convolutional Neural Network
CPU	Central Processing Unit
EVM	Ethereum Virtual Machine
FN	False Negatives
FP	False Positives
FPR	False Positive Rate
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GPU	Graphics Processing Unit
GRU/LSTM	Gated Recurrent Unit/Long Short-Term Memory
KNN	K-Nearest Neighbour
lightGBM	Light Gradient Boosting Method
LOF	Local Outlier Factor
NFT	Non-fungible token
PCA	Principal Component Analysis
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
RCNN	Region-Based Convolutional Neural Network
RF	Random Forest
RFE	Recursive Feature Elimination
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency
TN	True Negative
TP	True Positive
XGBoost	Extreme Gradient Boosting
ROC	Receiver Operating characteristic curve

Abstract

People now conduct most of their business online, but fiat currencies still require the involvement of third parties, such as banks, to process transactions. By contrast, block chain, a platform technology widely used by cryptocurrencies like Bitcoin and Ethereum, removes the need for third parties. It enables peer-to-peer transfers by performing transaction verification based on a consensus algorithm, removing dependency. However, even Ethereum blockchains are vulnerable to security issues like phishing attacks, fraudulent transactions, and majority attacks. Ethereum frauds provide significant profits and seriously jeopardize the network's ability to maintain its financial stability. In this study, we use machine learning models to identify illicit accounts on the Ethereum Blockchain. The classification algorithms are applied to data downloaded from the Kaggle website. When compared to other models, Gradient descent model has better precision, Recall and execution time is low hence we choose the Gradient Descent model with Over sampling data balancing technique as the best model. On the training set data, it has an Accuracy of 99.6%, Precision of 99.45%, Recall of 99.7% and F1-score of 99.6% and on validation set, it has Accuracy of 98.78%, Precision of 96.4%, Recall of 98.17% and F1-score of 97.27%. When the Gradient descent model is applied on the testing data set it has an Accuracy of 99.08%, Precision of 97.29%, Recall of 98.62%, F1-score of 97.95%. The best performing Gradient Descent model is proposed to identify the fraud accounts on the Blockchain platform, thereby enhancing security and gaining the confidence of its end users.

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

Blockchain is a decentralized, distributed, immutable database or ledger that enables the recording of transactions and the tracking of assets. By lowering the associated cost and risk, it makes it possible to practically track and trade anything of value. Cryptocurrency systems like Bitcoin, Ethereum, etc., NFTs, and Smart Contracts all depend on blockchains to keep a secure and decentralized record of transactions.

In Blockchain, the data or transaction is added to a new block as it becomes available. The block is then connected to the preceding block, and any further transactions are connected as new blocks, one after the other, to create a chain of data. A specific block of data cannot be changed since it is securely linked to its neighboring blocks. Depending on the type of blockchain being used, these transactions are recorded in the ledger with the aid of consensus algorithms like Proof of Stake (PoS), Proof of Work (PoW), or Proof of Authority (PoA)(Mingxiao et al., 2017; Zheng et al., 2017).The functional diagram of the blockchain network (Monrat et al., 2019)is given in the Figure 1.1.1 .

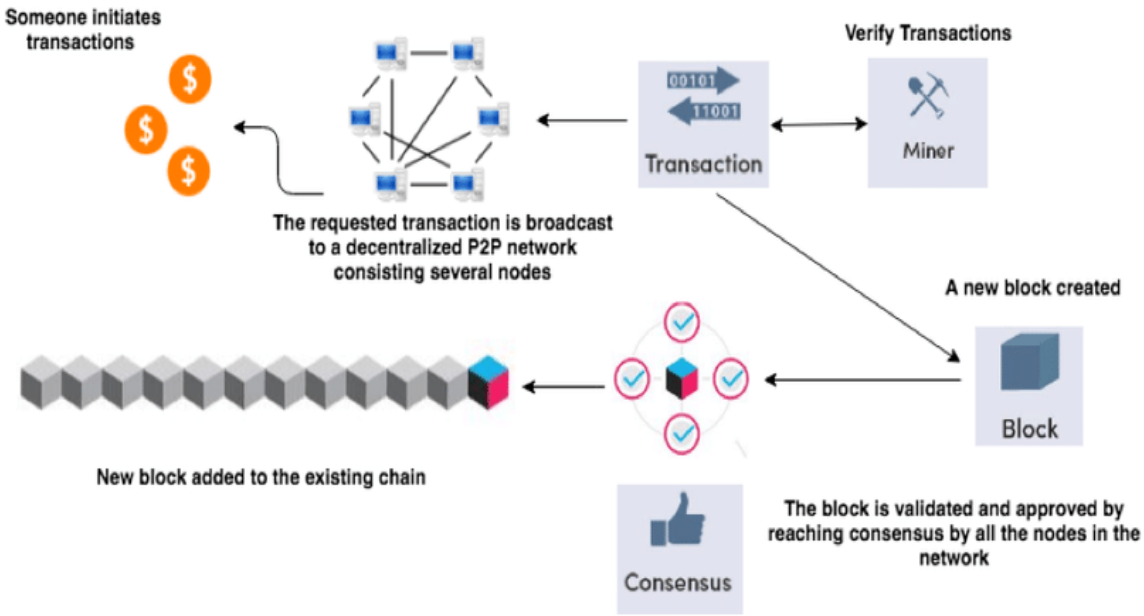

Figure 1.1.1 Functional Diagram of the Block chain Network

A person or group of persons using the alias Satoshi Nakamoto launched Bitcoin in November 2008, an electronic peer-to-peer system for electronic money transfer(Nakamoto, n.d.). Bitcoin is the most well-known and valuable cryptocurrency in the world despite the emergence of thousands of new ones and frequent price changes. The unique feature of Bitcoin coins is that they eliminate the issue of double spending without the need for a single, authoritative source.

Vitalik Buterin first suggested the Ethereum blockchain in 2014 (Buterin, n.d.), and the project was later launched in 2015. It is a Blockchain-based open-source decentralized network that generates its own cryptocurrency, called "ether." Next to Bitcoin, Ethereum is the most popular cryptocurrency on the market. It is used to build and execute Smart contracts without any fraud, downtime, or outside interference. It is also exchanged as digital money.

Compared to bitcoin's nearly consistent 10 minutes, buying ether takes only approximately 14 or 15 seconds, and there are a lot more ether units available than bitcoin. The Ethereum platform has two types of accounts: contract and externally held. Each account has a distinct 20-byte address that points to four fields required to guarantee the uniqueness of each state.

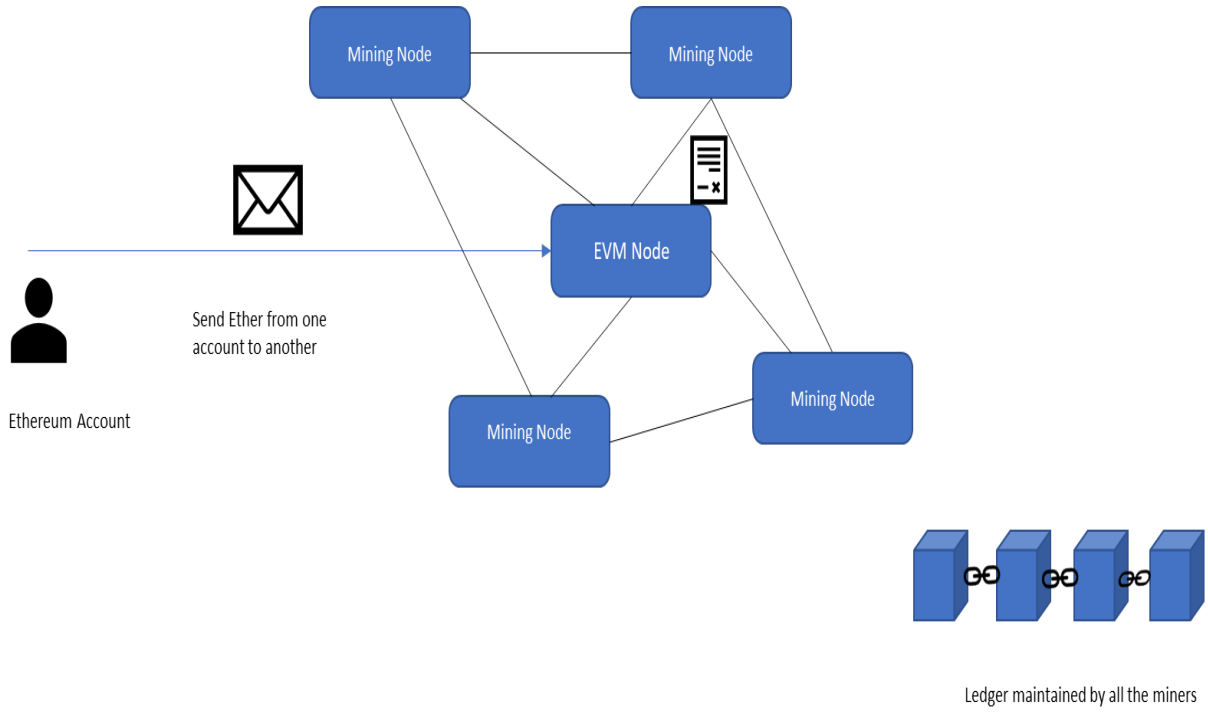

Figure 1.1.2 Architecture Diagram of the Ethereum.

Every node, or computer, in the peer-to-peer (P2P) network receives a transaction with a digital signature from the Ethereum blockchain, and after receiving it, the nodes use a consensus method to verify the transaction. As a result, if the transaction is valid, the new block is added

to the chain and the miner receives fees. By manipulating with gas cap and price of gas, these fees are created. The miner who has the greater quantity of money in his wallet has the right to validate the transaction and add it to the chain. The architecture of Ethereum is provided in the Figure 1.1.2.

The Ethereum Virtual Machine (EVM) merely assists in the transaction execution; it does not engage in mining. A copy of the ledger is kept by each miner. Every block in the chain is included in the ledger. The ledger instance maintained by the miner is constantly synchronized with the others.

In Ethereum, every block has a connection to every other block as shown in Figure 1.1.3. Two blocks have a parent-child relationship with one another. A parent is only permitted to have one child, and a child may have only one parent. In a blockchain, this aids in chain formation. Three blocks—Block 1, Block 2, and Block 3—are depicted in the diagram that follows. The parent of Block 1 is Block 2, while the parent of Block 2 is Block 3. The link is made by including the hash of the parent block in the block header of the child block. The hash of Block 1 is kept in Block 2's header, while the hash of Block 2 is stored in Block 3's header. The genesis block, often called the first block, is a concept used by Ethereum. When the chain is started for the first time, this block is automatically constructed.

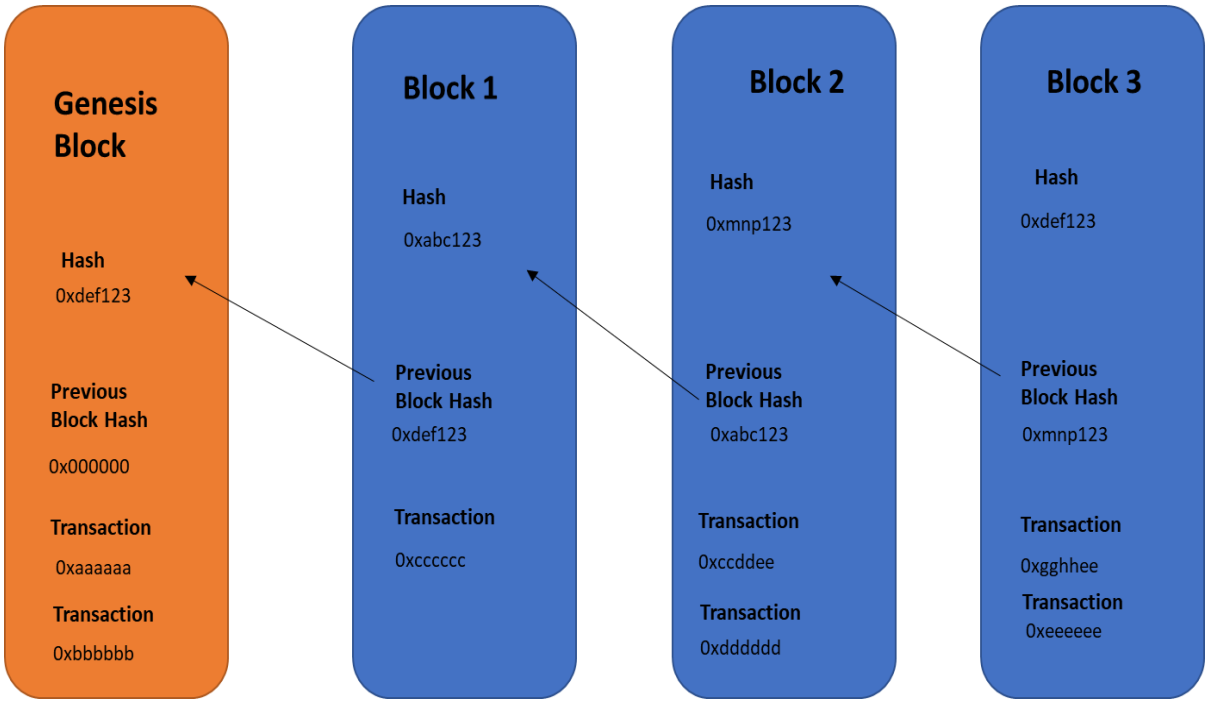

Figure 1.1.3 Blocks relationship between each other.

In blocks, Ethereum stores transactions. Each transaction requires a specific amount of gas to be used as part of its execution, and each block has an upper gas limit. The block gas limit cannot be exceeded by the total gas from all transactions that have not yet been recorded in a ledger. This makes sure that not every transaction is put in a single block. Other transactions are removed from the block as soon as the gas cap is met, and mining then starts.

As reported in the news channel CNBC (MacKenzie Sigalos, n.d.) ,a record $14 billion in cryptocurrencies was earned by scammers worldwide in 2021, mostly as a result of the growth of DeFi.

From the article(Figure 1.1.4) published by Federal Trade Commission on June 3,2022 it is clearly evident that around 46000 people have been scammed in the past 1 year reporting a loss of over $1billions in crypto. As per the article about 70% of the Bitcoin,9% Ether was used to pay to scammers(‘Fletcher’, 2022).

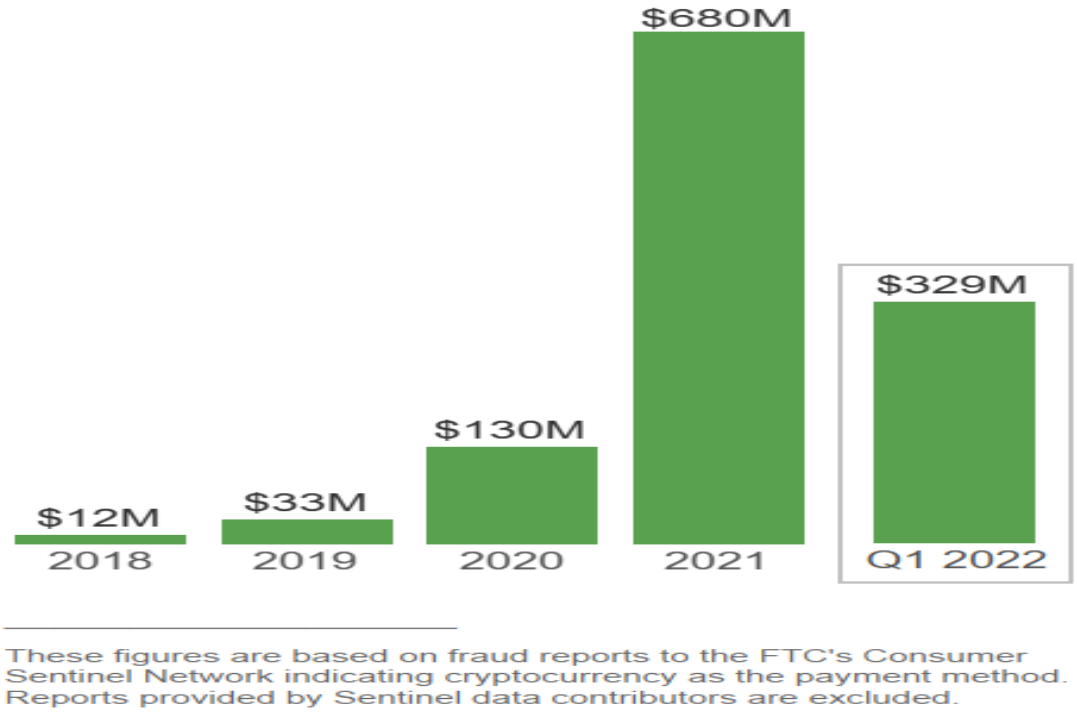

Figure 1.1.4 Cryptocurrency fraud loss w.r.t to years

1.2 Problem Statement

Blockchain technology is the foundation of Ethereum, a decentralized, international software platform. Ether serves as its native cryptocurrency (ETH). Ethereum now employs the Proof

of Stake (PoS) process, in which a network of individuals known as validators creates new blocks and collaborates to confirm the data they contain. All of these things do not, however, imply that the Ethereum block chain is faultless. Public keys are used to identify users of the Ethereum network. Since identity of the user is concealed via a public key, it has caught the attention of lawbreakers. Blockchain technology has also faced significant security challenges, including majority assaults, fraudulent transactions, forking, Initial coin offerings, Ponzi schemes, pump and dumps, ransomware, wallet scam, and phishing scams.

1.3 Aim and Objectives

The main aim of this research is to propose a model to detect the illicit transactions in the Ethereum Blockchain and in the process helps to stop the frauds. The identification of the fraudulent transactions can be done using machine learning classifiers. Various feature selection techniques and classifiers are used to compare and analyzed using various performance measures.

Based on the goals of this study, which are as follows, the research objectives have been developed:

- To utilize feature selection techniques to find the most important features of fraudulent transactions and visualization of the most important input features to examine the pattern and relationships between them.
- To make a reasonable balancing technique recommendation that can be used on the unbalanced dataset.
- To compare the predicted models and determine which one is more accurate to predict the illicit transaction accounts based on the important features selected.
- To assess how well proposed machine learning or neural network models perform.

1.4 Research Questions

For each of the study goals outlined below, the following research questions are suggested.

- How can we generate a generalized model to detect the illicit transactions in the Ethereum Blockchain?
- How is the dataset available for research structured. Does it further need any restructuring before implementing the models?

- What are the various attributes influencing in determining the illicit transaction and how well can we define them?
- What are the parameters tuning needed to improve the model performance ?
- Does the performance metrics effectively contribute in determining the illicit transaction?

1.5 Significance of the Study

Blockchain usage is now widespread among financial and non-financial areas. Financial security is essential for the healthy development of Blockchain technology. The proliferations of scams in the Ethereum blockchain will hinder the use of the technology and also reduces its acceptance among users. Thus, it is necessary to identify all these scams in the Ethereum blockchain. To safeguard the Ethereum platform against these frauds and their negative effects on its reputation, a thorough investigation must be conducted to ascertain the many reasons that allow these illicit transactions to occur, their core cause, and how they can be identified, reduced, and avoided. This study employs machine learning techniques to identify fraudulent transactions on the Ethereum Blockchain. Once the model proposed has been integrated with the platform itself, end users can use the unlawful transaction information obtained to avoid doing business with scammers based on the provided address. Additionally, interested authorities may take the appropriate measures against these addresses.

1.6 Scope of the Study

Due to the limited time available, some limitations must be made on this research in order for the project to be finished on schedule. The information that will be covered in this research study and that will be excluded from this study is listed in the following parts.

1.6.1 In-Scope

The results of this study are expected to deliver the following:

- Determining the illicit transaction and corresponding addresses of the users who are performing this transaction based on the machine learning approach.
- Able to identify the illicit transactions even in the imbalanced dataset.
- Lists out the main attributes that are useful in determining the illicit transactions.

- Comparison of various machine learning models pros and cons to determine the best model suitable for determining the illicit transaction.

1.6.2 Out-of-Scope

The following are not planned to be included in this study:

- The Ethereum platform is not directly scraped for the dataset. Instead, we make use of a transaction dataset found on the Kaggle website(vagifa, 2022).
- The model performance is evaluated based on metrics like F1-score, Accuracy, Precision Recall. The cost of implementing the model is not taken into account while deciding the best model.
- This study shall not include developing an end-to-end system or prototype to be integrated with the Ethereum system to work on the huge live transactions that keeps updating continuously. Instead it shall be focusing on the modelling techniques only needed to detect the illicit transactions.
- This research work is also limited with Ethereum blockchain. For fraud transactions in other cryptocurrencies the models used are applicable but the performance of models is not expected to be similar to Ethereum.

1.7 Structure of the Study

An overview of the Blockchain network and cryptocurrencies like bitcoin and ether is provided in Chapter 1. Additionally, the effect of these digital currencies on the financial market is covered. a description of the fraud detection procedure, the scope of the study, and the suggested method for developing a fraud detection system to find the scam in the Ethereum block chain.

The focus of chapter 2 will be on some of the earlier research projects on various cryptocurrency fraud detection.

In chapter 3, we will describe the methodology like preprocessing techniques(Data cleaning, Data Sampling, Data Transformation, Machine learning algorithms) that has been used for the analysis.

In chapter 4, We'll implement several models for imbalanced streams of data of training set.

In chapter 5,We will examine the performance of various models based on various metrics of model evaluation as well as performance fine-tuning and finalize the best model.

Chapter 6 will conclude with a summary of the findings and some recommendations for further research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

By reading and analyzing published works, the goal of this review is to comprehend the fraud in the Blockchain network in relation to the field of machine learning. This knowledge can provide other scholars and practitioners with perceptions on the present status and future directions for this area of research Given this motive, I have examined and confirmed the distribution of research papers based on the type of fraud, coin and categorize the research papers by the machine learning algorithms utilized. The following electronic research databases are used to give a thorough analysis of research papers: IEEE Xplore, arXiv, Science Direct , Google Scholar, Springer Link.

Eight keywords—"Ethereum", "Cryptocurrency" ,"Bitcoin", "Blockchain", "Fraud", "Machine learning", "Neural network", "Artificial intelligence" —and their variations were used in the search. Each paper's abstract was examined, and studies that were unquestionably unrelated to both machine learning and blockchain (Bitcoin, Ethereum) were eliminated. If, after reading the abstract, it was not possible to determine a paper's relevance with confidence the whole text of the paper was examined.

Because blockchain research is still in its infancy, there aren't enough relevant peer-reviewed journal papers to confine the scope of this survey to them. Consequently, we decided to broaden the inclusion criteria in this review study by including journal publications, conference working papers, excellent research reports, and articles. Each reviewed study's source is identified in this review paper so that scholars can choose whether to incorporate or reject it.

We chose 30 articles in total, classifying them according to the sort of fraud, the type of cryptocurrency, and machine learning methods. In the sections that follow, the details and outcomes of this classification are covered.

2.2 Detection Based on Type of Scam

One of the earliest methods of financial theft is through phishing scams. In this fraud, a false ID that remarkably resembles an authentic is presented in an attempt to take money from victims' wallets. Some people may ask us to install software that is malicious and infects the computer, allowing others to view our files. In the case of Ethereum, the conventional content-based detection methods are ineffective since phishing accounts are our detection targets .The blockchain system's openness and decentralization allow us to monitor all transaction records, including those pertaining to phishing accounts. Numerous studies are conducted to find phishing scams on the Ethereum network.

To identify phishing scams on Ethereum transactions, the authors (Yuan et al., 2020)proposed a three-step process. First, we get the tagged phishing accounts and associated transaction data from two reliable sources. Node2vec network embedding is used to extract latent features. Additionally, the phishing scams are categorized using a single-class support vector machine (SVM). This phishing detection technique has an F-score of 0.846. Similar to this, the authors (Chen et al., 2020)suggested 3-step approaches for phishing scam detection. First, they gathered the data using the Parity client and by crawling etherscan.io, gathering all Ethereum blockchain transactions and the phishing addresses. Then, with this data, a transaction graph was built and a graph-based cascade feature extraction method was presented, aiding in the extraction of numerous valuable characteristics. Next, a Dual-sampling Ensemble model to identify phishing suspects was proposed using the extracted features and lightGBM. The proposed model (i.e., DElightGBM) performs well across all metrics, including F-1 score, AUC, Precision, and Recall (i.e., all larger than 0.8).

Ponzi schemes are financial frauds that demand that investors receive their money back through the involvement of new investors. By doing a thorough examination of unusual contracts on the Ethereum based on the advertisements, manually analyzing the contract source code and analyzing transaction logs, the authors (Bartoletti et al., 2017) developed one of the first approaches to detect Ponzi scams on the Ethereum Smart contract. The authors (Ibba et al., 2021)suggested a machine learning approach that employs textual categorization methods to identify contracts that mimic a Ponzi scheme's behavior. They developed models that could accurately recognize Ponzi scheme contracts starting from a contracts dataset that only contained Ponzi schemes uploaded between 2016 and 2018. The linear Support Vector Machine and Multinomial Naive Bayes models, which offer the best results in terms of metrics

evaluation, ended up being the top models. Similarly the authors (Chen et al., 2019)also proposed a model based on the Random Forest which was very efficient in detecting the Smart Ponzi Schemes running on the Ethereum.

The authors(Chen et al., 2018) used XGBoost to detect the Ponzi scheme Smart contracts fraud in Ethereum. They obtained a precision and recall of 0.94 and 0.81 respectively. Similar study was performed by(Jung et al., 2019) and they obtained a precision of 0.99 and recall of 0.97 using SGD and J48 algorithms respectively.

The authors(Aljofey et al., 2022) put forth a model to detect Ethereum abnormal smart contract that extracts three different aspects from contract source code, transaction history, and opcodes. It has a hybrid feature set that consists of n-gram opcodes, distinct transaction features from contract account transaction data, and TF-IDF character level features from contract account source code. To improve the effectiveness of detecting fraud for atypical contract accounts, these features are combined and input into a classifier ensemble consisting of Extra-Trees and Gradient Boosting Machines. The model achieved an F-1 score of 0.90, an accuracy of 0.99, and a sensitivity of 0.82.

Contracts were categorized by (Fan et al., 2021) based on their operating codes (opcodes).They put forth an ordered boosting-based anti-leakage smart Ponzi schemes detection (Al-SPSD) approach. It outperformed other methods with a F-score of 96%.

A smart contract known as a "honeypot" appears to have a design flaw that permits any user to drain its Ether (Ethereum's native currency) by sending a specific amount of Ether to the contract beforehand.(Surin Murugiah, n.d.).(Ferreira Torres et al., n.d.) created a tool called HONEYBADGER to uncover smart contract honeypots using symbolic execution and well-defined criteria.

Based on the research of (Ferreira Torres et al., n.d.) and (Tsikerdekis et al., 2018) data pertaining to the Ethereum smart contract and its transactions were examined, using a combination of the data science technique and the machine learning method to identify honeypots.

2.3 Detection Based on the Type of Coin

By spotting anomalies in the Ethereum and Bitcoin transaction networks, the authors (Elmougy and Manzi, 2021) concentrated on identifying fake accounts and transactions. They employed Support Vector machine (SVM), Random Forest, and Logistic Regression on the Bitcoin and Ethereum datasets. The normalization and standardization transformations techniques are used because the online-scraped dataset showed imbalance and was right-skewed. With an accuracy rate of 96.9% on the Bitcoin dataset, the SVM method performed better than Random Forest, which had an accuracy rate of 80.2% on the Ethereum dataset.

The authors (Pham and Lee, 2016a) used three unsupervised learning methods including k-means clustering, Mahalanobis distance, and Unsupervised Support Vector Machine (SVM) on two graphs generated by the Bitcoin transaction network: one graph has users as nodes, and the other has transactions as nodes. The Unsupervised SVM method had better anomaly detection compared to the other models. Similarly, (Pham and Lee, 2016b) also proposed another technique for anomaly identification in the Bitcoin Transaction network. They used the laws of power degree & densification and the local outlier factor (LOF) method (which was accompanied by k-means clustering method). To validate the methods employed and determine the correctness, a sufficient evaluation method was not currently available.

In order to identify fraudulent activity in a bitcoin transaction, the authors (Monamo et al., 2016a)looked into the use of k-means, which aids in clustering related objects and identifying fraud in a multivariate setup. It is then run on two related algorithms following feature extraction and pre-processing. The first is trimmed k-means clustering, and the second is k-means clustering. Trimmed k-means have demonstrated considerable and promising result in identifying fraud with an improvement in detection rate. Similar techniques were employed in another study (Monamo et al., 2016b) to detect fraud across the Bitcoin network using trimmed k-means for global outliers and kd-trees for local outliers. Even without the use of class imbalance techniques, random forest was proven to be the best at classifying frauds based on the eight important parameter.

In order to conduct comparison experiments, the authors(Wang et al., 2021b) used three supervised ensemble classification algorithms—XGBoost, AdaBoost, and RF—as well as two basic algorithms—SVM and KNN—as well as two sampling techniques—SMOTETomek and SMOTE. They utilized ContractWard because it has high precision and recall in vulnerability

identification in smart contracts and obtained predictive micro-f1 and macro-f1 over 96% with XGBoost classifier and SMOTETomek as the sampling approach.

2.4 Detection Based on the Methodology

In this section the research works are classified based on the algorithms and the machine learning techniques used.

The Random Forest Algorithm has been suggested by the authors (Ibrahim et al., 2021) as a method for identifying fraudulent transactions on the Ethereum blockchain. They have used the machine learning algorithms like decision tree(j48),Random Forest ,K-nearest neighbor(KNN) on the Ethereum dataset obtained from the Kaggle website. They identified six features out of 42 that were the most successful in the prediction model using the Correlation Attribute Evaluator in the Weka software tool. Their findings reveal that the Random Forest algorithm significantly improved their time measurements and the F measure. Similarly the authors(Ostapowicz and Żbikowski, 2019) also proposed Random Forest model. The dataset that was taken from the Ethereum platform was subjected to 10-fold cross validation. On the dataset, they used Random Forest, XGBoost, and SVM models. But recall and false positive rate were suggested by the as crucial metrics for finding frauds in the Ethereum blockchain by these authors. In comparison to the other models, the Random Forest offered a lower false positive rate.

The authors proposed the Boosting algorithms to detect the Ethereum transaction fraud after performing the sampling techniques on the dataset to adjust the data imbalance. The authors(Poursafaei et al., 2020) proposed a solution for detecting the illicit entities using ensemble methods like stacking and ADABoost classifier. They have used dataset obtained by extracting them from the Ethereum website. The PCA method is used for feature extraction and various sampling techniques like Under sampling, over sampling and SMOTE was used. The modelling techniques Logistic regression, Support vector machine, Random Forest and ensemble methods were applied on the sampled dataset.A LGBM-based method for spotting Ethereum fraud transaction was suggested by the authors (Aziz et al., 2022). By up sampling the minority class to the same frequency as the majority class, they were able to balance the classes using the SMOTE technique. The power transform function is used to normalize the training characteristics. The model assists in accurately identifying unusual transactions while

avoiding overfitting. With excellent efficiency and minimal memory consumption, it anticipates Ethereum transaction frauds.

The authors proposed that Extreme Gradient Boosting performed efficiently in detection of the fraud cases. The authors (Farrugia et al., 2020) suggested a technique to spot illicit accounts on the Ethereum network and also highlighted the most crucial input features that affect the choice of fraudulent accounts. The "XGBoost" classification model was employed to categorize these fraudulent accounts. Similarly the authors(Maurya and Kumar, 2022) used Logistic Regression, Random Forest, XGBoost, and Decision tree to detect the fraud cases. In comparison to all the models, they detected that the XGBoost has attained the highest accuracy of 99.21% for the stated dataset.

With a classification accuracy of 0.91, the authors(Nerurkar et al., 2021) developed a supervised model based on the ensemble model based on decision trees to identify fraudulent activity in Bitcoin. On a dataset of 1216 real-world items, it outperformed numerous other models, including SVM, Logistic Regression, XGBoost, and Random Forest. The authors (Nerurkar et al., 2020) used an ensemble model of decision trees, decision trees and random forests in a related study. It was discovered that the Ensemble model outperformed the other models on 13 of the 14 performance parameters.

The graph neural network classification method was suggested by the authors (Tan et al., 2021) to identify fraudulent accounts on the Ethereum platform. The information was gathered by extracting fraudulent transactions from etherscamdb.info and legitimate transactions from etherscan.io. An algorithm based on random walks called network embedding is employed for the automatic feature extraction process. This method just considers how much information is contained in the Ethereum transactions, not how the Ethereum-based transaction network is structured. It has been discovered that other graph neural network models, including GAT and GraphSAGE, perform better than the GCN model in many applications; as a result, the authors advise using these techniques in future efforts. In addition, they advise using their detection methods for additional unlawful Ethereum-based activities including gambling and money laundering. The EvolveGCN approach was suggested by (Weber et al., 2019) to identify bitcoin's behaviour associated with money laundering. This approach creates a unique GCN model for each time step using the GCN as the feature extractor of the graph. The RNN (GRU/LSTM) is then used to connect these GCN models in order to capture the dynamic changes in the graph. This strategy, however, only works with the supervised learning mode

and ignores the network's overall features, using the most powerful Top-k node to represent all network data at a given point in time. To gather the Ethereum dataset required for the investigation, the authors(Hu et al., 2021) used a data-slicing method. Then a smart contract was built using its first 14 components. In addition, they used LSTM network on the train and test set our smart contract datasets. Similarly, the authors(Wang et al., 2021a) also used LSTM model to detect the Ponzi schemes on the Ethereum smart contracts. But they utilized over-sampling techniques to adjust the class-imbalance.

In order to detect illegal transactions, the authors (Baek et al., 2019) employed an unsupervised learning expectation maximisation (EM) approach to cluster the data set. They employed Random Forest to do an anomaly detection based on the features created by the unsupervised learning (RF). All of the clusters that served the same function were combined into one, while the remaining cluster contained a wallet that handled anomalous.

2.5 Discussion

The Pre-processing, ML algorithms, Evaluation methods used by the research scholars in the cryptocurrency fraud detection are represented tabular format below for selected 15 references. Since Ethereum was founded in 2016, it is still in its infancy. The majority of research efforts have been conducted in the last three years. And most of the researches are done related to Ponzi schemes, anomaly transaction, money laundering activities. It helps in mitigating the illicit activity over the blockchain but does not help in the account level detection of the same. The year the journal article was published, the kind of issue the research addresses, and the specifics of the dataset utilised by the authors are all summarized in the table (Table 2.5.1) below.

Table 2.5.1 Summary of Year Type Dataset of few citations from Literature Review

	Reference	**Year**	**Type**	**Dataset**
1.	(Ibrahim et al., 2021)	2021	Ethereum Illicit account detection	Kaggle dataset (Ethereum Transaction dataset). – It contains 7809 transactions 7651 normal transactions 159 fraudulent transactions. It

				has 42 features related to Ethereum transaction.
2.	(Aziz et al., 2022)	2022	Ethereum Fraudulent transaction detection	Kaggle dataset (Ethereum Transaction dataset). Total 9841 transactions .7662 normal 2179 fraudulent
3.	(Tan et al., 2021)	2021	Ethereum transaction records fraud	Dataset obtained by scraping it from two websites Ether-ScamDb and Etherscan. 4196 addresses are fraudulent and 5649 are legal transaction addresses
4.	(Farrugia et al., 2020)	2020	Illicit account detection	Illicit account is obtained from Etherscamdb. Legal transaction was obtained by scraping functional tool developed by Sokolowska (2018).2179 accounts are illegal accounts and 2502 are normal accounts.
5.	(Jung et al., 2019)	2019	Ethereum Ponzi scheme smart contract detection	Web scrapper to collect 3203 addresses of smart contract from Etherscan.184 Ponzi scheme address from (M. Bartoletti, n.d.)
6.	(Yuan et al., 2020)	2020	Ethereum Phishing scam	Dataset obtained by scraping it from two websites Ether-ScamDb and Etherscan.
7.	(Poursafaei et al., 2020)	2020	Detecting malicious entities in Ethereum	Manually crawling from Ethereum block explorer Etherscan.io

8.	(Ostapowicz and Żbikowski, 2019)	2019	Detecting fraudulent account in Ethereum Blockchain	Dataset downloaded using Etherscan API
9.	(Baek et al., 2019)	2019	Detecting cryptocurrency illegal transactions in Ethereum blockchain	By scraping metadata from wallets of Etherscan.io
10.	(Chen et al., 2020)	2020	Ethereum Phishing scam	Downloading the Ethereum ledger using Ethereum parity client and crawled etherscan.io
11.	(Chen et al., 2019)	2019	Detect Smart Ponzi scheme on Ethereum	Obtained Smart Ponzi scheme contracts and other contacts from etherscan.io. 3,780 open-source contracts and 200 smart Ponzi contracts
12.	(Ibba et al., 2021)	2021	Detect Smart Ponzi scheme on Ethereum	171 Smart Ponzi contracts and 1500 contracts that are not Ponzi schemes
13.	(Maurya and Kumar, 2022)	2022	Ethereum fraudulent transaction dataset detection	Kaggle dataset (Ethereum transaction dataset) -9841 transaction records
14.	(Liu et al., 2022)	2022.	Ethereum abnormal Smart contract detection	Dataset obtained from etherscan.io .1382 total contract of which 1251 normal contracts and 131 are fraud contracts
15.	(Elmougy and Manzi, 2021)	2021	Anomaly detection on bitcoin and Ethereum	Bitcoin dataset obtained from public ledger. It has a total of 30,294,698 transactions.

				30,290,045nonfraudulent transactions, 4653 of which were fraudulent transactions. Ethereum dataset obtained from Ethereum blockchain browser Etherscan.io.

The Pre-processing, ML algorithms, Evaluation methods used by the research scholars in the cryptocurrency fraud detection are represented tabular format below for selected 15 references. The pre-processing methods, Machine learning techniques implemented, Evaluation methods and the best model proposed by the authors are all summarized in the table (Table 2.5.2) below. Most of the authors have utilized Accuracy, F1-score, precision and recall as the evaluation metrics. From the research done by most of authors Random Forest (Chen et al., 2019b; Ostapowicz and Żbikowski, 2019; Poursafaei et al., 2020; Elmougy and Manzi, 2021; Ibrahim et al., 2021),SVM (Yuan et al., 2020; Elmougy and Manzi, 2021; Ibba et al., 2021), XGBoost(Farrugia et al., 2020; Maurya and Kumar, 2022) are well performing models.

Table 2.5.2 Summary of Pre-processing techniques, Evaluation methods of few citations from Literature Review

	Reference	**Pre-Processing Methods**	**Machine Learning Techniques**	**Evaluation Methods**	**Proposed Model**
1.	(Ibrahim et al., 2021)	– Feature selection based on Correlation Attribute Evaluator in Weka Software Tool	– Decision Tree(j48) – Random Forest – KNN	– Accuracy – Precision – Recall – F-Measure	Random Forest

2.	(Aziz et al., 2022)	– SMOTE is used on imbalanced dataset	– LRR – RF – LGBM – MLP – KNN – XGB – SVM	– Accuracy – F1 Score – Precision	LGBM
3.	(Tan et al., 2021)	– Node2vec network embedding method is used for the automatic feature extraction	– Logistic Regression – Naïve Bayes – GCN Classifier	– Accuracy – Precision – Recall – F1-score	GCN
4.	(Farrugia et al., 2020)	10-fold cross validation	XGBoost	– Accuracy – F1-score – AUC Execution time	XGBoost
5.	(Jung et al., 2019)	– Feature Selection based on code features for Day -0 classification model and behaviour-based features for behaviour-based models. – Full feature model was	– Decision Tree(J48) – Random Forest – Stochastic Gradient Descent	– Precision – Recall – F1-score	SGD using precision J48 using Recall

		implemented for day 0 to 248			
6.	(Yuan et al., 2020)	– Latent feature selection using node2vec method outperforms Deepwalk, time features only, Amount features only	– One-class support vector machine (SVM)	– Precision – Recall – F1-score	One-class support vector machine (SVM)
7.	(Poursafaei et al., 2020)	– Feature selection using PCA – Data imbalance techniques like Over sampling, Under sampling and SMOTE	– Logistic Regression – Support vector machine – Random Forest – AdaBoost Stacking classifier	– Precision – Recall – F1 score – Accuracy	Stacking classifier consisting of RF and LR
8.	(Ostapowicz and Żbikowski, 2019)	Grid search using 10-fold cross validation for finding parameters	– Random Forest – Support vector machine XGBoost	– False positive rate – Precision – Recall – F1 score Specificity	Random Forest
	(Baek et al., 2019)	– EM algorithm for the Gaussian mix model	– Unsupervised learning to cluster dataset and	– Precision – Recall – F1-score –	Unsupervised learning to cluster

			then RF to identify the cluster containing anomalous records		dataset and then RF to identify the cluster containing anomalous records .
10.	(Chen et al., 2020)	– Transaction graph construction – Cascade feature extraction	– SVM – DT – lightGBM – DESVM – DEDT – DElightGBM	– Precision – Recall – F1-score – AUC –	Dual Sampling Ensemble model-DelightGBM
11.	(Chen et al., 2019)	– Obtain the Account feature from transaction – Code feature are obtained from the opcode generated from Bytecode	– DT – SVM – XGBoost – IF – One-class SVM – OCSVM+DT – OCSVM+SVM	– Precision – Recall – F1-score	Random Forest
12.	(Ibba et al., 2021)	– Obtain the Account feature from transaction Code feature are obtained from the opcode	– Decision Tree – Support vector machine Multinomial Naïve Bayes	– Precision – Recall – F1-score – Accuracy	Linear SVM and Multinomial Naïve Bayes

		generated from Bytecode			
13.	(Maurya and Kumar, 2022)	– Normalization using Standard Scalar – SMOTE for dataset Imbalance	– XGBoost	– Precision – Recall – F1-score Support	XGBoost
14.	(Liu et al., 2022)	– account features and code features as node attributes – code is pre-processed and converted into vectors using Doc2Vec.	– SVM – CNN – RCNN – GCN – GAT – S_HGTNs	– F1-score – Micro-F1 – Macro-F1	Heterogeneous Graph Transformer Network
15.	(Elmougy and Manzi, 2021)	– Normalization and Standardization transformation are applied	– SVM – Random Forest – Logistic Regression	– Precision – Recall – Accuracy – Confusion matrix – F1-score	SVM for bitcoin dataset Random Forest for Ethereum

2.6 Summary

Blockchain and machine learning technologies have both captured the interest of academics and industry professionals, and everyone is starting to see how they are being used in the real world. In order to comprehend the trend of machine learning techniques employed in detection of the scams in blockchain technology, this chapter has compiled 30 research papers that have been published during the last five years. It has been observed from above analysis that most of the methods the features are extracted through manual analysis and the dataset imbalance are ignored while implementing the ML models. Unlike the majority of past studies, our

method involves managing the data imbalance using a variety of methods such SMOTE, ADASYN, and oversampling before using classification algorithms. Also feature extraction are to be finalised without much manual intervention using RFE, Information gain etc. Metrics like accuracy, F-score, precision, recall and time and speed at which it executes are used to determine which model is the best. The further details of these feature extraction, balancing the dataset, modelling is comprehensively discussed in the next section 3.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

The methodology used entails crucial steps like choosing the target data, pre-processing the selected data, transforming the data into an organized and comprehensible format, adjusting the dataset imbalance, putting machine learning techniques into practice, and evaluating the effectiveness of machine learning with evaluation metrics.

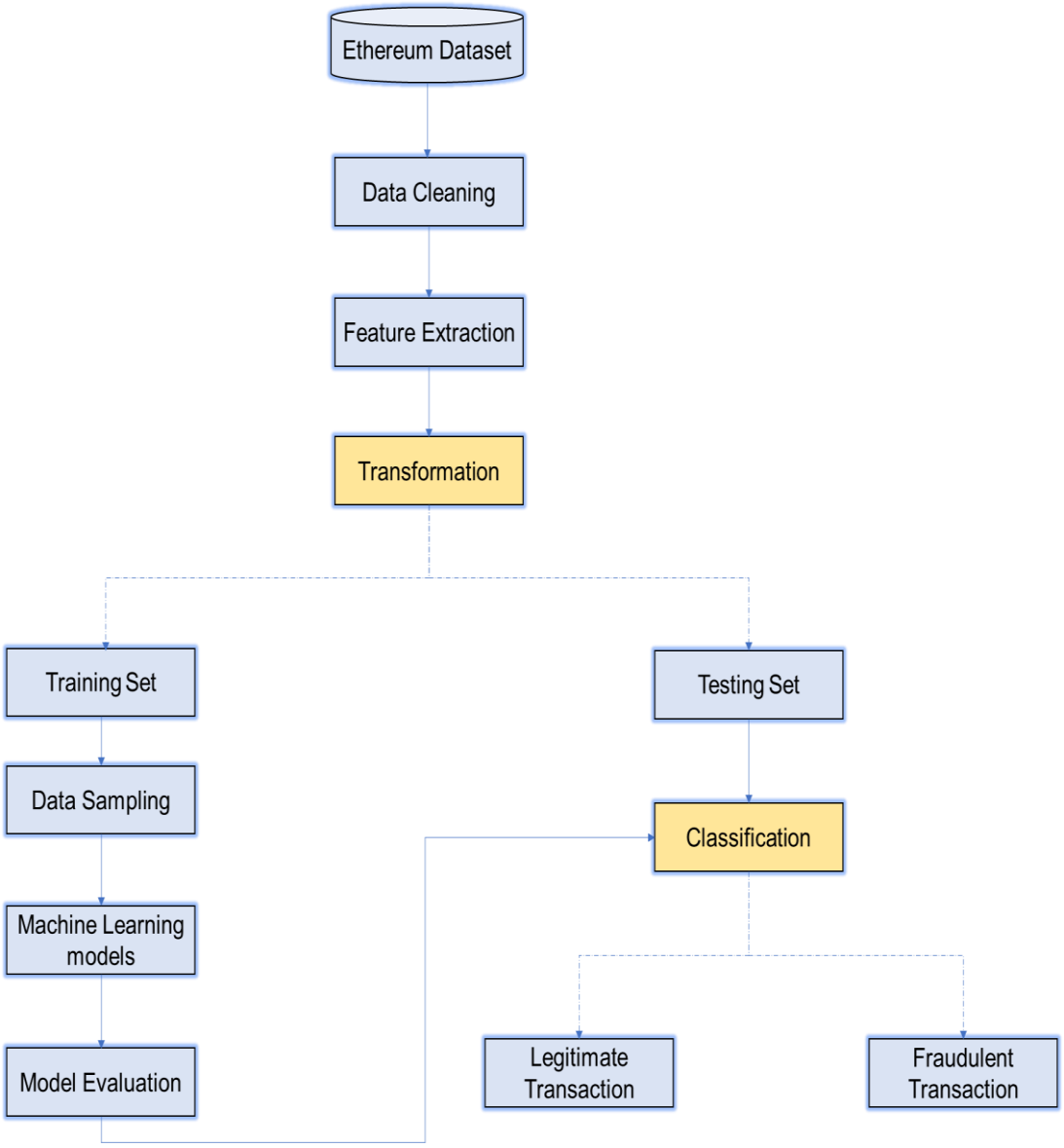

Figure 3.1.1 Flowchart for determining the Ethereum illicit Transactions

3.2 Research Methodology

The working of the model for the Ethereum illicit transaction has been proposed in the Figure 3.1.1. All the procedures required to process the Ethereum dataset are shown in the flowchart. Data cleaning is the initial step in the process, which entails eliminating any missing data, controlling outliers, deleting erroneous data, and removing duplicates. The most crucial features are then obtained. After that, the data is transformed. Afterwards, the data is divided into training and testing sets. On the training set, the data sampling techniques are used to correct the imbalance. After applying the classification models to the training set, the models are assessed. The most effective model is selected, and it is used to categorise the data on the testing set.

3.2.1 Dataset Selection

On the website etherscan.io, Ethereum transactions are visible. With every new transaction, the webpage is updated. You can process the data by scraping it off from the webpage. However, the data is constantly changing and is very large in size. Therefore, for this study, we take a sample of the dataset(Ibrahim et al., 2021; Aziz et al., 2022) that has been scraped, processed, and made available on the Kaggle website (vagifa, 2022). This dataset contains rows of Ethereum transactions, a kind of cryptocurrency, that have been found to be valid and fraudulent.

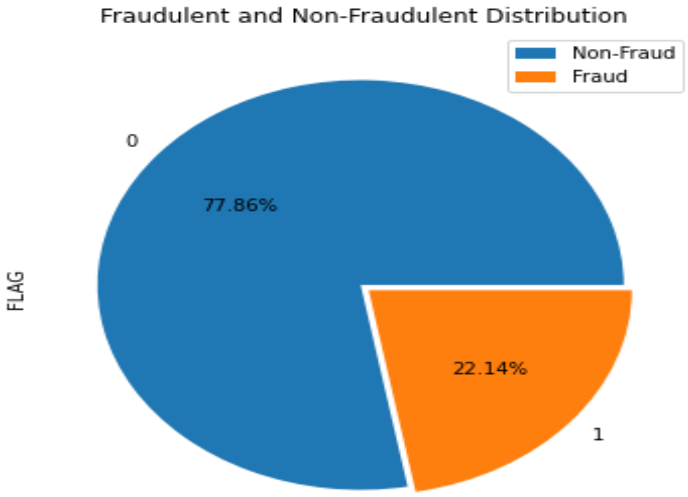

Figure 3.2.1.1 Distribution of fraud and non-fraud cases

The dataset has 9841 records and 51 columns. But the dataset is highly imbalanced which may affect the performance of model. The Flag attribute can be used as target variable. In the dataset, there are 7662 valid transactions (which account for almost 77.9%) and 2179 fraudulent transactions (which account for 22.1%).

Table 3.2.1.1 Definition of the variables used in the dataset.

Variable	Type	Description
Index	Int	the index number of a row
Address	Object	the address of the ethereum account
FLAG	Int	whether the transaction is fraud or not
Avg min between sent tnx	Float	Average time between sent transactions for account in minutes
Avgminbetweenreceivedtnx	Float	Average time between received transactions for account in minutes
TimeDiffbetweenfirstand_last(Mins)	Float	Time difference between the first and last transaction
Sent_tnx	Int	Total number of sent normal transactions
Received_tnx	Int	Total number of received normal transactions
NumberofCreated_Contracts	Int	Total Number of created contract transactions
UniqueReceivedFrom_Addresses	Int	Total Unique addresses from which account received transactions
UniqueSentTo_Addresses20	Int	Total Unique addresses from which account sent transactions
MinValueReceived	Float	Minimum value in Ether ever received
MaxValueReceived	Float	Maximum value in Ether ever received
AvgValueReceived5Average value in Ether ever received	Float	
MinValSent	Float	Minimum value of Ether ever sent
MaxValSent	Float	Maximum value of Ether ever sent
AvgValSent	Float	Average value of Ether ever sent
MinValueSentToContract	Float	Minimum value of Ether sent to a contract
MaxValueSentToContract	Float	Maximum value of Ether sent to a contract
AvgValueSentToContract	Float	Average value of Ether sent to contracts
TotalTransactions(IncludingTnxtoCreate_Contract)	Int	Total number of transactions
TotalEtherSent	Float	Total Ether sent for account address
TotalEtherReceived	Float	Total Ether received for account address
TotalEtherSent_Contracts	Float	Total Ether sent to Contract addresses
TotalEtherBalance	Float	Total Ether Balance following enacted transactions
TotalERC20Tnxs	Float	Total number of ERC20 token transfer transactions
ERC20TotalEther_Received	Float	Total ERC20 token received transactions in Ether
ERC20TotalEther_Sent	Float	Total ERC20token sent transactions in Ether
ERC20TotalEtherSentContract	Float	Total ERC20 token transfer to other contracts in Ether
ERC20UniqSent_Addr	Float	Number of ERC20 token transactions sent to Unique account addresses
ERC20UniqRec_Addr	Float	Number of ERC20 token transactions received from Unique addresses
ERC20UniqRecContractAddr	Float	Number of ERC20token transactions received from Unique contract addresses
ERC20AvgTimeBetweenSent_Tnx	Float	Average time between ERC20 token sent transactions in minutes
ERC20AvgTimeBetweenRec_Tnx	Float	Average time between ERC20 token received transactions in minutes
ERC20AvgTimeBetweenContract_Tnx	Float	Average time ERC20 token between sent token transactions
ERC20MinVal_Rec	Float	Minimum value in Ether received from ERC20 token transactions for account
ERC20MaxVal_Rec	Float	Maximum value in Ether received from ERC20 token transactions for account
ERC20AvgVal_Rec	Float	Average value in Ether received from ERC20 token transactions for account
ERC20MinVal_Sent	Float	Minimum value in Ether sent from ERC20 token transactions for account
ERC20MaxVal_Sent	Float	Maximum value in Ether sent from ERC20 token transactions for account
ERC20AvgVal_Sent	Float	Average value in Ether sent from ERC20 token transactions for account
ERC20UniqSentTokenName	Float	Number of Unique ERC20 tokens transferred
ERC20UniqRecTokenName	Float	Number of Unique ERC20 tokens received
ERC20MostSentTokenType	Object	Most sent token for account via ERC20 transaction
ERC20MostRecTokenType	Object	Most received token for account via ERC20 transactions

The Table 3.2.1.1 shows the definition of the variables used in the dataset.

3.2.2 Data Pre-processing

The Ethereum dataset and all the necessary libraries are imported. The dataset is imbalanced and may contain duplicate data so the first step is to perform the Data Cleaning. One of the most crucial processes in the model-building is data cleaning. It involves handling the missing values, removing any duplicates or irrelevant data, fixing structural errors, handling outliers.

There are no duplicates in the dataset we used. Additionally, because there aren't any categorical columns, label encoding is not necessary. The median of the related column is used to replace null values in numerical columns.

The important features can be selected using the methods like correlation matrix, Recursive feature elimination or Regularization methods like Lasso Regression. The RFE takes the machine learning model and number of features as input. It then recursively reduces the number of features by ranking them based on the accuracy of model. We can then learn the names of the features that have been determined to be the most crucial using the RFE support method. In the correlation matrix method based on the heatmap/correlation coefficient if the selected variables having high correlation values between them, then we need to keep just one of the correlated ones and drop the other. If the input features are not helping to train our machine learning model in a good way, the coefficients of those features are reduced when using Lasso Regression. By assigning them coefficients equal to zero, some of the traits might be automatically eliminated in this way. The Feature Exaction technique like Principal Component Analysis can also be used to reduce the original dimensions of the dataset. It is achieved by maximizing the variances and minimizing the reconstruction error. The entropy decrease caused by a dataset modification is calculated using information gain. By assessing each variable's information gain in relation to the target variable, it can be used for feature selection.

3.2.3 Dataset Transformation

Data Transformation is the process of converting the data to a format suitable for model building. It is performed since the algorithm might become biased if the data is skewed. And converting all the attribute to the same scale allows the algorithm to compare the features better. We can proceed with using Min Max scaler, Standard scaler or Power Transformer based on Exploratory Data Analysis performed on the dataset. The dataset has the features right skewed. Therefore, we can use log scaling to transform the right skewed distribution into normal distribution. We can use the np.log () function to achieve the same. After the log transformation

the features which are right skewed tends to be normally distributed.

3.2.4 Class Balancing

The data is divided into a training set and a testing set in a ratio of 70:30. Since the data is imbalanced, we can proceed using Over sampling, SMOTE or ADASYN on the training set to balance the majority and minority class (Poursafaei et al., 2020; Aziz et al., 2022; Maurya and Kumar, 2022).

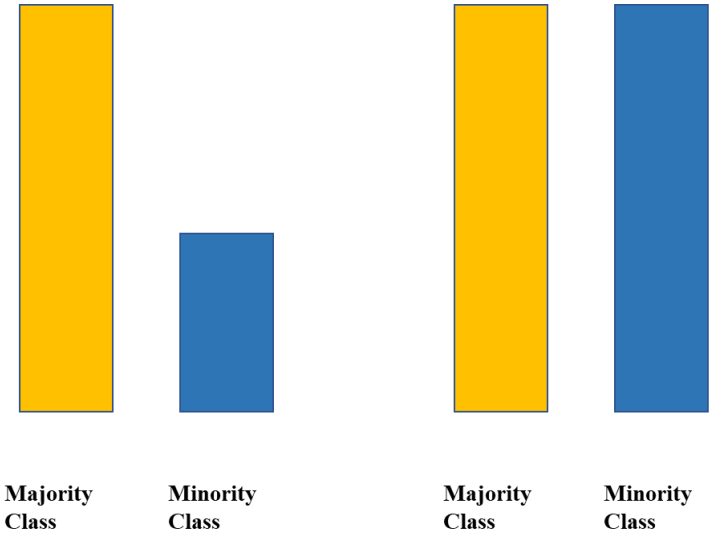

Figure 3.2.4.1 Over Sampling Techniques

In order to present a better representation of the minority class in the sample, oversampling (shown in Figure 3.2.4.1) increases the number of instances in the minority class by randomly repeating them.

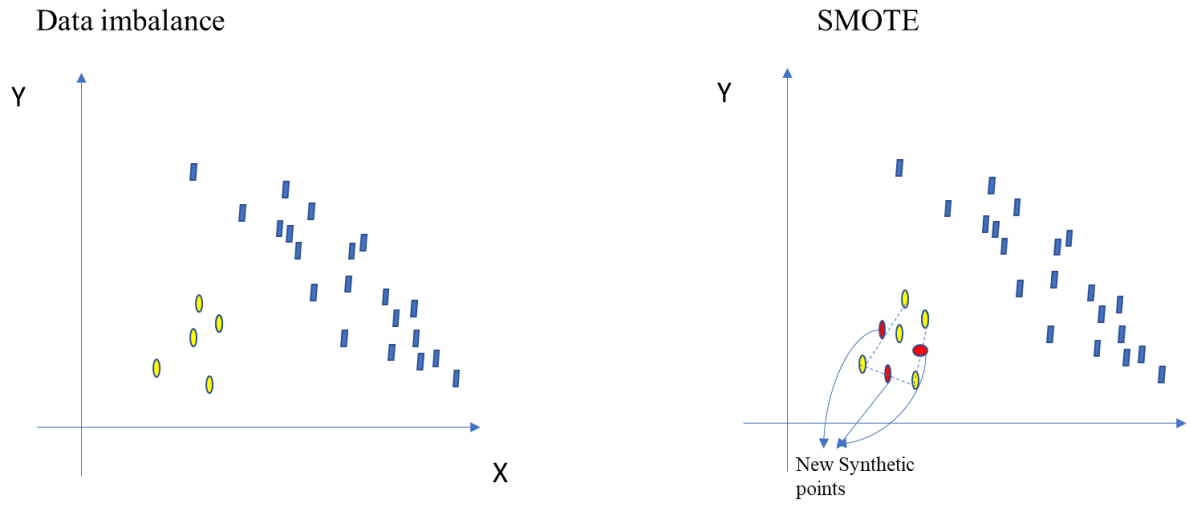

Figure 3.2.4.2 SMOTE Technique

In the SMOTE technique (shown in Figure 3.2.4.2) a slice of data from the minority class is used as an illustration before new synthetic cases that are comparable to it are constructed. The original dataset is then supplemented with these synthetic cases. This technique has been used by authors (Aziz et al., 2022; Maurya and Kumar, 2022).

ADASYN technique also induces synthetic data based on an algorithm. We can use all these three techniques and employ the Machine Learning model and determine the best combination.

3.2.5 Data Mining

A number of machine learning techniques, including Logistic Regression, Decision Tree, KNN, Naive Bayes, Ensemble models like Random Forest, XGBoost, ADABoost, Light GBM, GBM, CatBoost, Neural Network etc. can be used on training dataset.

Based on independent variables, logistic regression calculates the likelihood of an event. Given that the result is a probability, the dependent variable's value ranges from 0 to 1. Certain behaviours or characteristics may have a higher association with fraudulent activities which can be used by financial organisations to protect their clients.

The output of Decision Tree Classification (shown in Figure 3.2.5.1) is produced as a binary tree-like structure, making it relatively simple for financial organisations to understand and identify the fraud. The target variable can be predicted using rules in a decision tree model. The underlying distribution of the data is explained simply by the Tree Classification technique.

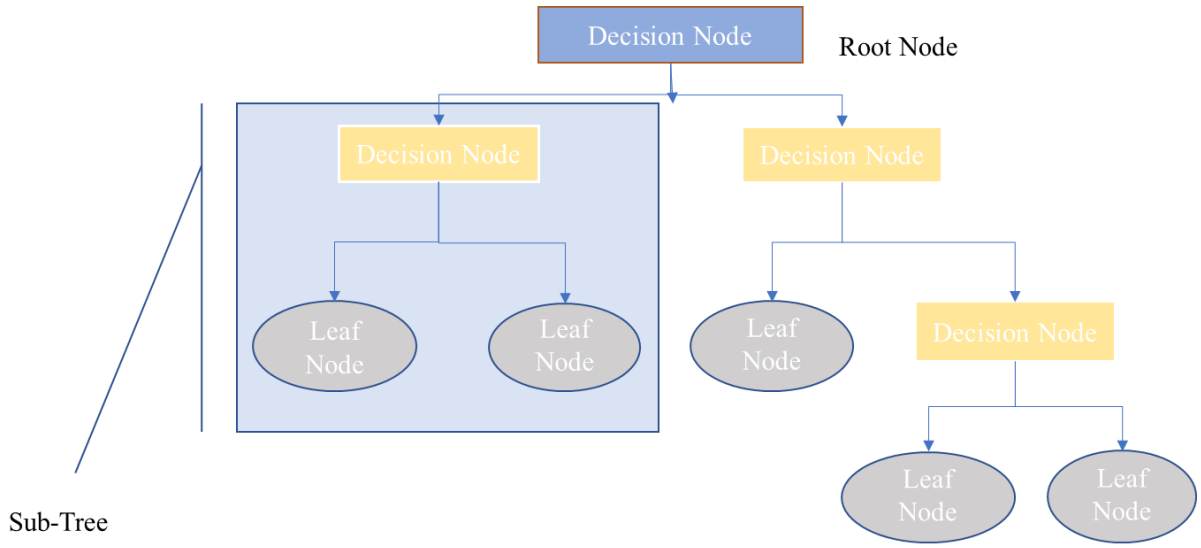

Figure 3.2.5.1 Decision Tree Classification

The KNN algorithm (shown in Figure 3.2.5.2) simply saves the dataset during the training phase, and when it receives new data, it categorises it into a category that is very similar to the new data based on Euclidean distance.

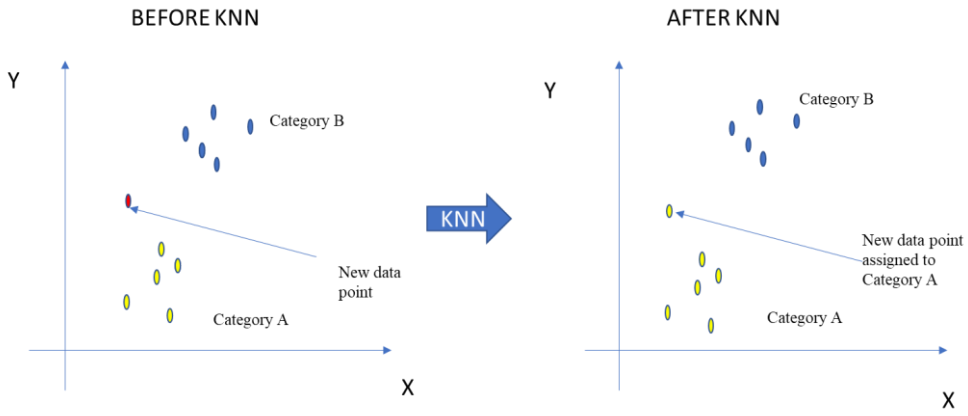

Figure 3.2.5.2 KNN Classification

The Naive Bayes Classifier is one of the most straightforward and efficient classification algorithms available today. It aids in the development of quick machine learning models capable of making accurate predictions. Being a probabilistic classifier, it makes predictions based on the likelihood that an object will occur.

Random Forest is an ensemble technique (shown in Figure 3.2.5.3) which combines multiple decision tree to solve a complex problem. The accuracy of the model increases with increase in number of trees. It is one efficient model in detecting the fraud and many authors(Ostapowicz and Żbikowski, 2019; Ibrahim et al., 2021) have used the same.

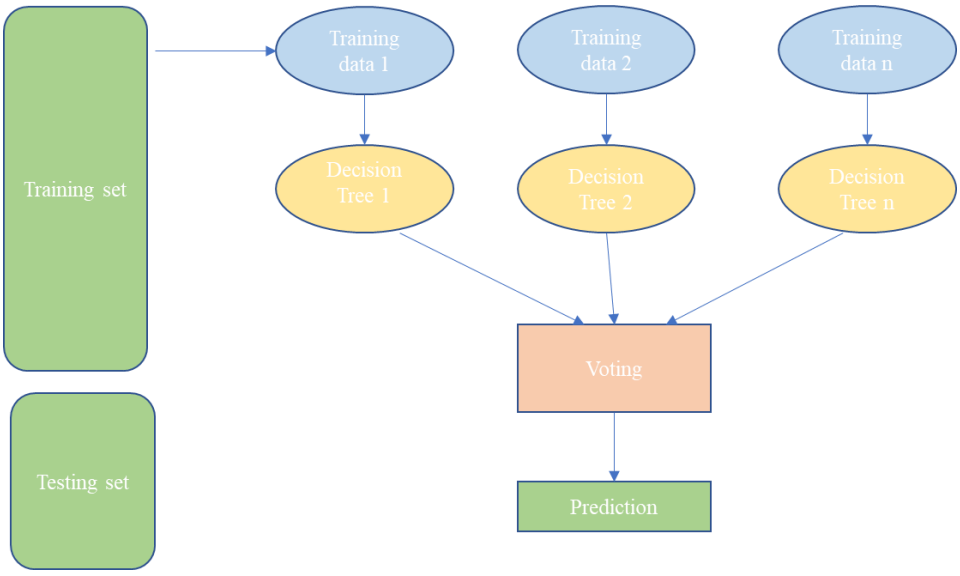

Figure 3.2.5.3 Random Forest

Boosting Algorithms work by combining weak learner models to form a strong rule. The base learning algorithm creates a new weak prediction rule each time it is used. This procedure is iterative. The boosting algorithm combines these numerous weak prediction rules into a single strong prediction rule after many rounds. The four boosting algorithms to be used in the Ethereum illicit transaction detection are listed below.

Adaptive Boosting begins by projecting the original data set and gives each observation the same weight. When the first learner makes an erroneous prediction, the observation that was incorrectly forecasted is given more weight. Due to the iterative nature of the process, it keeps adding learners until the accuracy or quantity of models can no longer be increased.

Both XGBoost and Gradient Boosting Machines (GBMs), ensemble tree approaches, use the gradient descent architecture to boost weak learners (CARTs in general). But XGBoost enhances the fundamental GBM architecture with system optimization and algorithmic improvements.

Compared to many other machine learning methods, CatBoost is exceptionally quick. On both GPU and CPU, the splitting, tree construction, and training processes have been sped up. Training on the GPU is 40 times quicker than training on the CPU, twice as fast as training on LightGBM, and 20 times faster than training on XGBoost.

LightGBM is a gradient boosting framework that employs algorithms for tree-based learning. It has the following benefits since it is spread effectively: faster training pace and greater effectiveness. reduced memory utilisation.

SVM is a supervised machine learning algorithm that is most widely used in classification. When using the SVM algorithm, each data point is represented as a point in n-dimensional space (where n is the number of features you have), with each feature's value being the value of a certain coordinate. Then, we carry out classification by locating the hyper-plane that effectively distinguishes the two classes.

The learning process that occurs in human brains serves as an inspiration for neural networks. They are made up of an artificial network of parameters that the computer may learn from and fine-tune by examining fresh data.

Such a network is represented visually in Figure 3.2.5.4. The first layer of neurons takes the input x and then process it through three functions before producing an output. The second layer is then given that output. Based on the results of the first layer, additional output is calculated. The model's ultimate output is created by combining these secondary outputs.

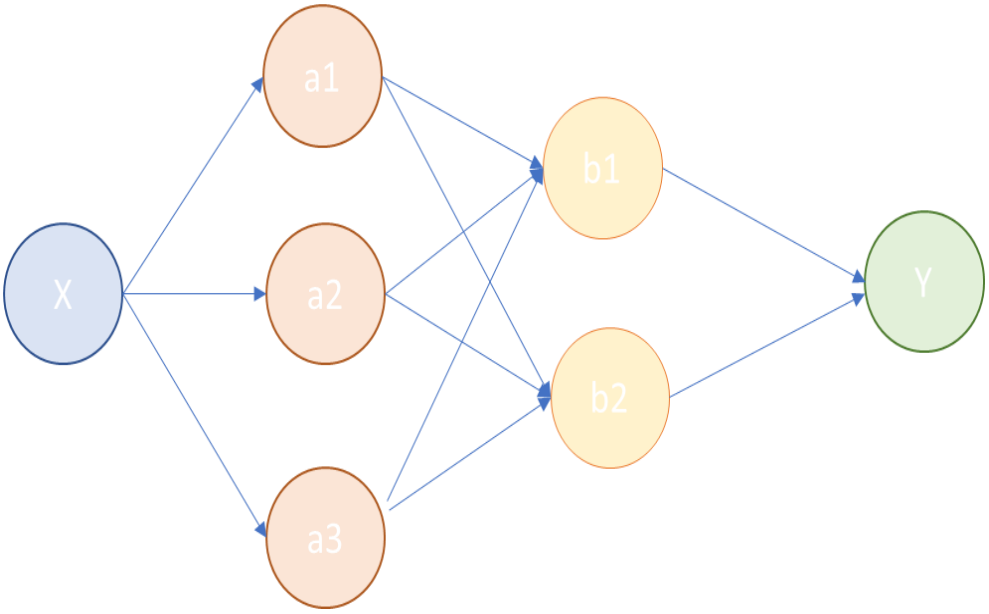

Figure 3.2.5.4 Neural Network Layers

The architecture of the model is frequently defined by a hyperparameter, which is a parameter whose value is utilised to regulate the learning process. A procedure of looking for the best model architecture is required in this situation since several classification algorithms will be employed to build the model. A step that can accomplish this is hyperparameter tuning.

3.2.6 Evaluation Metrics

The performance of the above algorithms on the training set is determined based on metrics such as Confusion matrix, Precision, Recall, F1-score, Accuracy and other metrics. Based on comparison for the above metrics for different models the best model is chosen. As soon as the best model has been chosen, it is tested on testing dataset to determine its performance on the unseen dataset.

The performance of the classification model is assessed using a NxN matrix called the confusion matrix, where N is the number of the target class. A successful model has high TP and TN and low FN and FP.

In the confusion matrix, there are four crucial values.

True Positives: When both Actual and Predicted values are Positive.

True Negatives: When both the Actual and the Predicted values are Negative.

False Positives: When the Actual is Negative but the Prediction is Positive. Also known as Type I error.

False Negatives: When the Actual is Positive but the Prediction is Negative. Also known as Type II error.

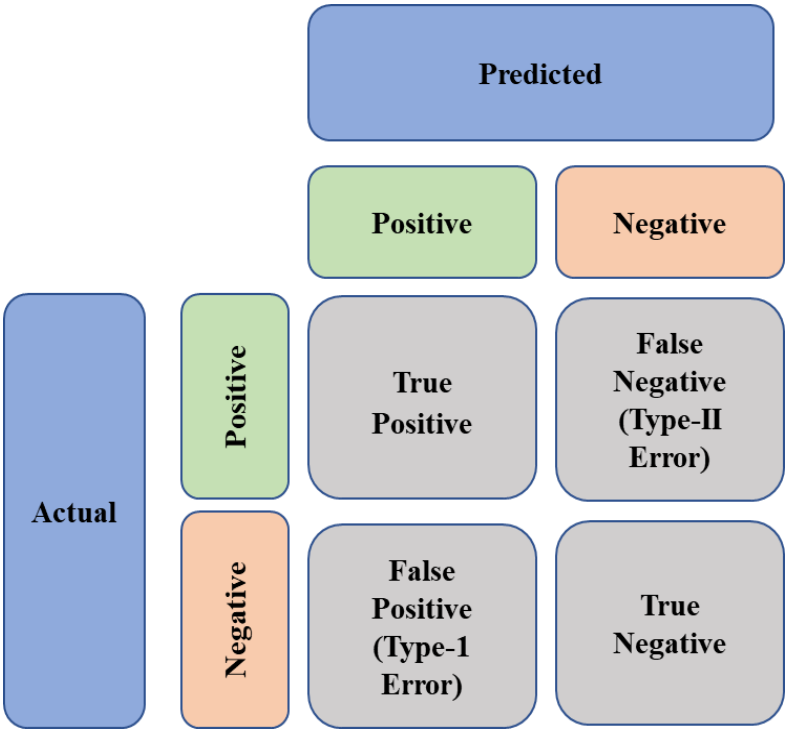

Figure 3.2.6.1 Confusion matrix of 2x2

Accuracy quantifies how frequently the classifier predicts correctly. It is the proportion of forecasts that were accurate to those that were made overall.

But since our dataset is imbalanced this metric is not suited. Even if the model predicts that each transaction belongs to the majority class the accuracy of model will be high.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

The ratio of the total number of accurately classified positive classes to the total number of predicted positive classes is known as precision. The precision value has to be high in our study. We should focus on reducing the FP (reducing the number of legal accounts classified wrongly as illicit accounts).

$$Precision = \frac{TP}{TP + FP}$$

The measure of the positive class predicted correctly among all the positive class. Recall should be ideally 1. The FN values should be low as it implies maximum illicit transactions are correctly predicted. In our study we shall focus on model having high recall value.

$$Recall = \frac{TP}{TP + FN}$$

The harmonic mean of precision and recall is represented by the F1 score, which ranges from 0 to 1. The F1 score sort of keeps the precision and recall of your classifier in balance. In our study we aim in having a high F1-score.

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

The proportion of negative cases that were mistakenly classified as positive cases in the data is known as False Positive Rate (FPR). FPR should be very low in our study.

$$FPR = \frac{FP}{FP + TN}$$

The receiver operating characteristic curve (ROC curve) is a graph that displays how well a classification model performs across all categorization levels. The True Positive Rate and False Positive Rate are plotted on this graph. For each classifier employed here, it shows both of these parameters against one another at different threshold values. This is done purely for the purpose of developing prediction models for fraud detection in the Ethereum Blockchain.

3.3 Proposed Method

The suggested solution in this research requires the use of high-performance machine learning algorithms. Before using the Power Transformation technique, the data must first undergo basic clean-up. To choose the key features, one may also employ the RFE or information gain. The SMOTE, ADASYN, and Oversampling methodologies will also be put to the test in search of the best balancing approach. N-fold cross validation will also be carried out with the training set. Models like Ensemble techniques LGBM, XGBoost, AdaBoost, CatBoost, and Random Forest are anticipated to be more effective at spotting unauthorised accounts.

3.4 Summary

In order to lower the danger of Ethereum blockchain fraud, the proposed model should offer some quick and cost-effective predictions. In order to identify frauds in the Ethereum blockchain, there should be a reliable, efficient, and accurate method. The technique must also be suitable for real-world applications for cryptocurrency applications to exploit it and maximise customer satisfaction by preventing fraud in user-made transactions. The further

details of best modelling techniques and the results are comprehensively discussed in the next chapter 4.

CHAPTER 4

ANALYSIS

4.1 Introduction

The analysis chapter provides information about the dataset as well as critical pre-processing procedures such variable elimination, data transformation, handling missing values, univariate and bivariate analysis, dataset splitting, and data visualisation.

4.2 Data Description

On the website etherscan.io, Ethereum transactions are visible. With every new transaction, the webpage is updated. You can process the data by scraping it off from the webpage. However, the data is constantly changing and is very large in size. Therefore, for this study, we take a sample of the dataset(Ibrahim et al., 2021; Aziz et al., 2022) that has been scraped, processed, and made available on the Kaggle website (vagifa, 2022). This dataset contains rows of Ethereum transactions, a kind of cryptocurrency, that have been found to be valid and fraudulent. The fraud and non-fraud cases from the dataset are represented in a bar chart in Figure 4.2.1.

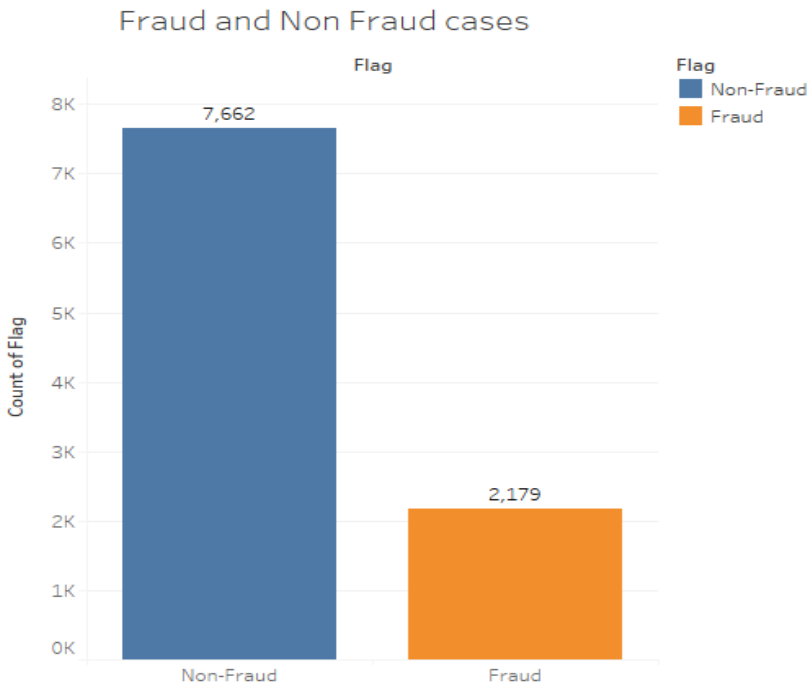

Figure 4.2.1 Bar Chart showing count of fraud and non-fraud cases

The dataset has 9841 records and 51 columns. But the dataset is highly imbalanced which may affect the performance of model. The Flag attribute can be used as target variable. In the dataset, there are 7662 valid transactions (which account for almost 77.9%) and 2179 fraudulent transactions (which account for 22.1%). The distribution of fraud and non-fraud instances is shown in Figure 4.2.2.

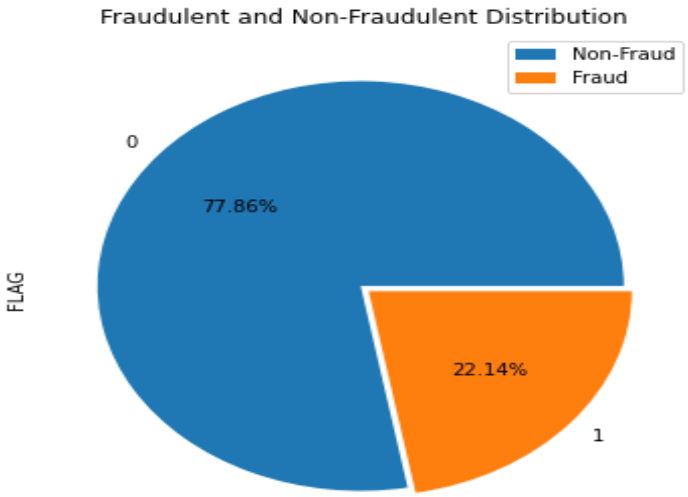

Figure 4.2.2 Distribution of fraud and non-fraud cases

The 51 columns that make up the dataset are listed in the Table 4.2.1 below, along with information about each column's datatype and description.

In the context of a challenge to detect Ethereum fraud transaction, we investigate feature correlations and feature importance. This feature selection approach aims to increase computing speed without sacrificing performance. Data transformation is used in feature engineering as well since it makes it easier to find insights and transforms raw data into a format that is better suited for model construction.

Table 4.2.1 Attributes Type and Description

Variable	Type	Description
Index	Int	the index number of a row
Address	Object	the address of the ethereum account
FLAG	Int	whether the transaction is fraud or not
Avg min between sent tnx	Float	Average time between sent transactions for account in minutes
Avgminbetweenreceivedtnx	Float	Average time between received transactions for account in minutes
TimeDiffbetweenfirstand_last(Mins)	Float	Time difference between the first and last transaction
Sent_tnx	Int	Total number of sent normal transactions
Received_tnx	Int	Total number of received normal transactions
NumberofCreated_Contracts	Int	Total Number of created contract transactions
UniqueReceivedFrom_Addresses	Int	Total Unique addresses from which account received transactions
UniqueSentTo_Addresses20	Int	Total Unique addresses from which account sent transactions
MinValueReceived	Float	Minimum value in Ether ever received
MaxValueReceived	Float	Maximum value in Ether ever received
AvgValueReceived5Average value in Ether ever received	Float	
MinValSent	Float	Minimum value of Ether ever sent
MaxValSent	Float	Maximum value of Ether ever sent
AvgValSent	Float	Average value of Ether ever sent
MinValueSentToContract	Float	Minimum value of Ether sent to a contract
MaxValueSentToContract	Float	Maximum value of Ether sent to a contract
AvgValueSentToContract	Float	Average value of Ether sent to contracts
TotalTransactions(IncludingTnxtoCreate_Contract)	Int	Total number of transactions
TotalEtherSent	Float	Total Ether sent for account address
TotalEtherReceived	Float	Total Ether received for account address
TotalEtherSent_Contracts	Float	Total Ether sent to Contract addresses
TotalEtherBalance	Float	Total Ether Balance following enacted transactions
TotalERC20Tnxs	Float	Total number of ERC20 token transfer transactions
ERC20TotalEther_Received	Float	Total ERC20 token received transactions in Ether
ERC20TotalEther_Sent	Float	Total ERC20token sent transactions in Ether
ERC20TotalEtherSentContract	Float	Total ERC20 token transfer to other contracts in Ether
ERC20UniqSent_Addr	Float	Number of ERC20 token transactions sent to Unique account addresses
ERC20UniqRec_Addr	Float	Number of ERC20 token transactions received from Unique addresses
ERC20UniqRecContractAddr	Float	Number of ERC20token transactions received from Unique contract addresses
ERC20AvgTimeBetweenSent_Tnx	Float	Average time between ERC20 token sent transactions in minutes
ERC20AvgTimeBetweenRec_Tnx	Float	Average time between ERC20 token received transactions in minutes
ERC20AvgTimeBetweenContract_Tnx	Float	Average time ERC20 token between sent token transactions
ERC20MinVal_Rec	Float	Minimum value in Ether received from ERC20 token transactions for account
ERC20MaxVal_Rec	Float	Maximum value in Ether received from ERC20 token transactions for account
ERC20AvgVal_Rec	Float	Average value in Ether received from ERC20 token transactions for account
ERC20MinVal_Sent	Float	Minimum value in Ether sent from ERC20 token transactions for account
ERC20MaxVal_Sent	Float	Maximum value in Ether sent from ERC20 token transactions for account
ERC20AvgVal_Sent	Float	Average value in Ether sent from ERC20 token transactions for account
ERC20UniqSentTokenName	Float	Number of Unique ERC20 tokens transferred
ERC20UniqRecTokenName	Float	Number of Unique ERC20 tokens received
ERC20MostSentTokenType	Object	Most sent token for account via ERC20 transaction
ERC20MostRecTokenType	Object	Most received token for account via ERC20 transactions

4.2.1 Elimination of Variables

The 21 columns listed below in the Table 4.2.1.1 are excluded from the dataframe for additional analysis because they had higher correlation coefficients with other variables, larger null values, and also no values.

Table 4.2.1.1 Reason for Elimination of attributes

Column	Reason
Index	Does not affect further analysis
ERC20_most_rec_token_type	Have 53% null values
ERC20 most sent token type	Have 53% null values
ERC20 avg time between sent tnx	as they have no values corresponding to each record
ERC20 avg time between rec tnx	as they have no values corresponding to each record
ERC20 avg time between rec 2 tnx	as they have no values corresponding to each record
ERC20 avg time between contract tnx	as they have no values corresponding to each record
ERC20 min val sent contract	as they have no values corresponding to each record
ERC20 max val sent contract	as they have no values corresponding to each record
ERC20 avg val sent contract	as they have no values corresponding to each record
total ether received	high correlation coefficient value
total ether sent contracts	high correlation coefficient value
ERC20 uniq sent addr	high correlation coefficient value
ERC20 uniq rec addr	high correlation coefficient value
ERC20 max val rec	high correlation coefficient value
ERC20 avg val rec	high correlation coefficient value
ERC20 min val sent	high correlation coefficient value
ERC20 max val sent	high correlation coefficient value
ERC20 avg val sent	high correlation coefficient value
ERC20 uniq sent token name	high correlation coefficient value
ERC20 uniq rec token name	high correlation coefficient value
avg value sent to contract'	high correlation coefficient value
total transactions (including tnx to create contract	high correlation coefficient value

4.2.2 Identification of missing values

There are several null values in the 23 columns shown in Table 4.2.2.1 below. Moreover, the data are right skewed. Thus, median is used to fill in for missing numbers. Furthermore, the 841 and 851 entries in the categorical columns "ERC20 most send token type" and "ERC20 most rec token type" have a value of 0. The category values are changed to null because they cannot be zero.

Table 4.2.2.1 Handling of missing values

Column	Null count	Method to handle	Reason
Total ERC20 tnxs	829	replace missing numerical variables with median	data is skewed
ERC20 total Ether received	829	replace missing numerical variables with median	data is skewed
ERC20 total ether sent	829	replace missing numerical variables with median	data is skewed
ERC20 total Ether sent contract	829	replace missing numerical variables with median	data is skewed
ERC20 uniq sent addr	829	replace missing numerical variables with median	data is skewed
ERC20 uniq rec addr	829	replace missing numerical variables with median	data is skewed
ERC20 uniq sent addr.1	829	replace missing numerical variables with median	data is skewed
ERC20 uniq rec contract addr	829	replace missing numerical variables with median	data is skewed
ERC20 avg time between sent tnx	829	replace missing numerical variables with median	data is skewed
ERC20 avg time between rec tnx	829	replace missing numerical variables with median	data is skewed
ERC20 avg time between rec 2 tnx	829	replace missing numerical variables with median	data is skewed
ERC20 avg time between contract tnx	829	replace missing numerical variables with median	data is skewed
ERC20 min val rec	829	replace missing numerical variables with median	data is skewed
ERC20 max val rec	829	replace missing numerical variables with median	data is skewed
ERC20 avg val rec	829	replace missing numerical variables with median	data is skewed
ERC20 min val sent	829	replace missing numerical variables with median	data is skewed
ERC20 max val sent	829	replace missing numerical variables with median	data is skewed
ERC20 avg val sent	829	replace missing numerical variables with median	data is skewed
ERC20 min val sent contract	829	replace missing numerical variables with median	data is skewed
ERC20 max val sent contract	829	replace missing numerical variables with median	data is skewed
ERC20 avg val sent contract	829	replace missing numerical variables with median	data is skewed
ERC20 uniq sent token name	829	replace missing numerical variables with median	data is skewed
ERC20 uniq rec token name	829	replace missing numerical variables with median	data is skewed
ERC20 most sent token type	841	replace missing numerical variables with median	data is skewed
ERC20_most_rec_token_type	851	replace missing numerical variables with median	data is skewed
ERC20 most sent token type	841	changing 0 values to null	cause a 0 value doesnt mean anything in categorical features
ERC20_most_rec_token_type	851	changing 0 values to null	cause a 0 value doesnt mean anything in categorical features

4.2.3 Data Normalization

The attribute distribution prior to transformation demonstrates that the data is asymmetric and skewed. Since the data is skewed (Figure 4.2.3.1), power transformation is applied on the attributes and all the values in the dataset are in same scale.

Figure 4.2.3.1 Histogram of attributes before Transformation

The Figure 4.2.34.2.3.2 below shows the distribution of attributes after the Transformation is applied.

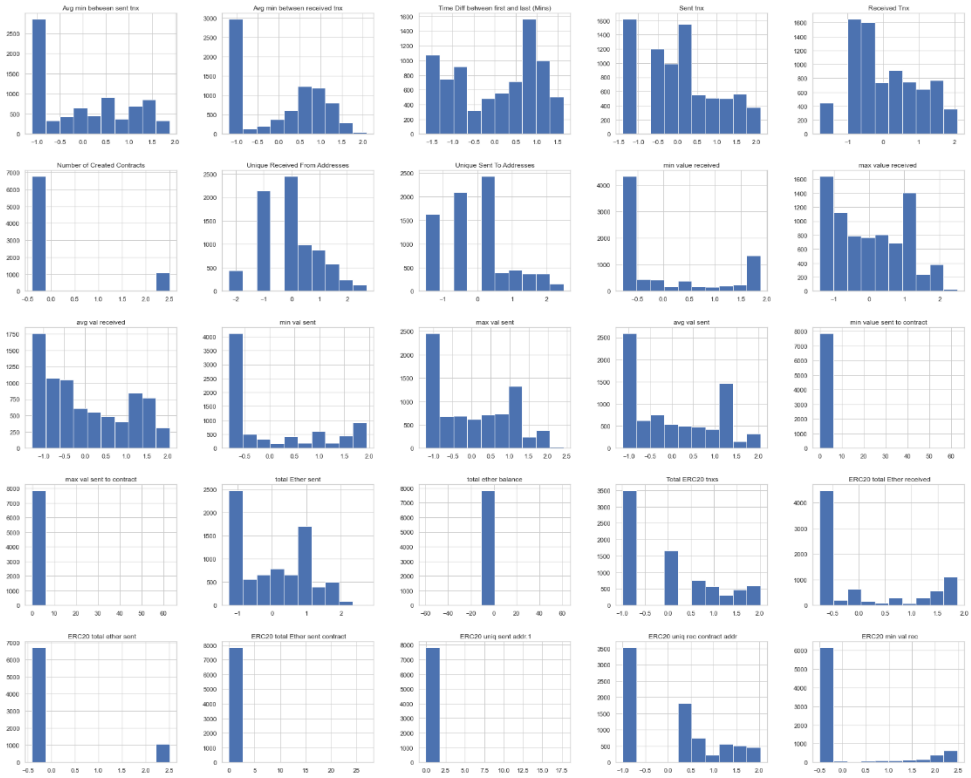

Figure 4.2.3.2 Histogram of attributes after Power Transformation

4.2.4 Splitting of Original Dataset

The Flag attribute is used as target variable The dataset is divided into a training set, validatio n set and testing set, each with a size of 0.8 ,0.1, 0.1 respectively. The training set has 7872 re cords, validation set has 984 records while the testing set contains 985 records.

```
]:  # Putting response variable to y  and Putting feature variable to X
    y=dataset['FLAG']
    X=dataset.drop(['FLAG','Address'],axis=1)
```

```
]:  def train_val_test_split(X, y, train_size, val_size, test_size):
        X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size = test_size)
        relative_train_size = train_size / (val_size + train_size)
        X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val,
                                                          train_size = relative_train_size, test_size = 1-relative_train_size)
        return X_train, X_val, X_test, y_train, y_val, y_test
```

```
]:  # splitting into  train, test ,validation datasets
    X_train, X_val, X_test, y_train, y_val, y_test=train_val_test_split(X,y,0.8,0.1,0.1)
```

Figure 4.2.4.1 Splitting of dataset into training, validation and testing set

4.2.5 Handling Imbalanced Dataset

The training set is now being further processed for data sampling. Oversampling and under-sampling is carried out because the data collection is unbalanced. It lessens the disparity in class counts. Under sampling results in loss of data since some samples are removed from majority class. Therefore, we do not consider under sampling in our further analysis. The Oversampling, SMOTE, ADASYN technique is used for sampling in the models. The total count and individual dataset count for each class after handling imbalance is provide in the table below. Table 4.2.5.1 displays the record counts after applying the data imbalance techniques.

Table 4.2.5.1 Handling Imbalance Techniques along with record counts

Handling Imbalance	Total records	Individual Dataset counts
Under Sampling	3486	1 1743 0 1743
Over Sampling	12258	1 6129 0 6129
SMOTE	12258	1 6129 0 6129
ADASYN	12299	1 6170 0 6129

4.3 Exploratory Data Analysis

In this section, the trends and patterns are displayed as graphs, which makes it much simpler to comprehend the trends and patterns in the data.

4.3.1 Correlation Coefficient and Heat Map

To determine which variables are related to one another and the strength of this association, correlation coefficient and heatmap are utilised. The attributes that have a correlation with other variables of greater or equal to 0.7 are eliminated.

The 13 attributes removed are 'avg value sent to contract', 'total transactions (including tnx to create contract','total ether received', 'total ether sent contracts', ' ERC20 uniq sent addr', ' ERC20 uniq rec addr', ' ERC20 max val rec', ' ERC20 avg val rec', ' ERC20 min val sent', ' ERC20 max val sent', ' ERC20 avg val sent', ' ERC20 uniq sent token name', ' ERC20 uniq rec token name'.

The correlation coefficient between various numerical attributes is displayed in the heatmap below in Figure 4.3.1.1.

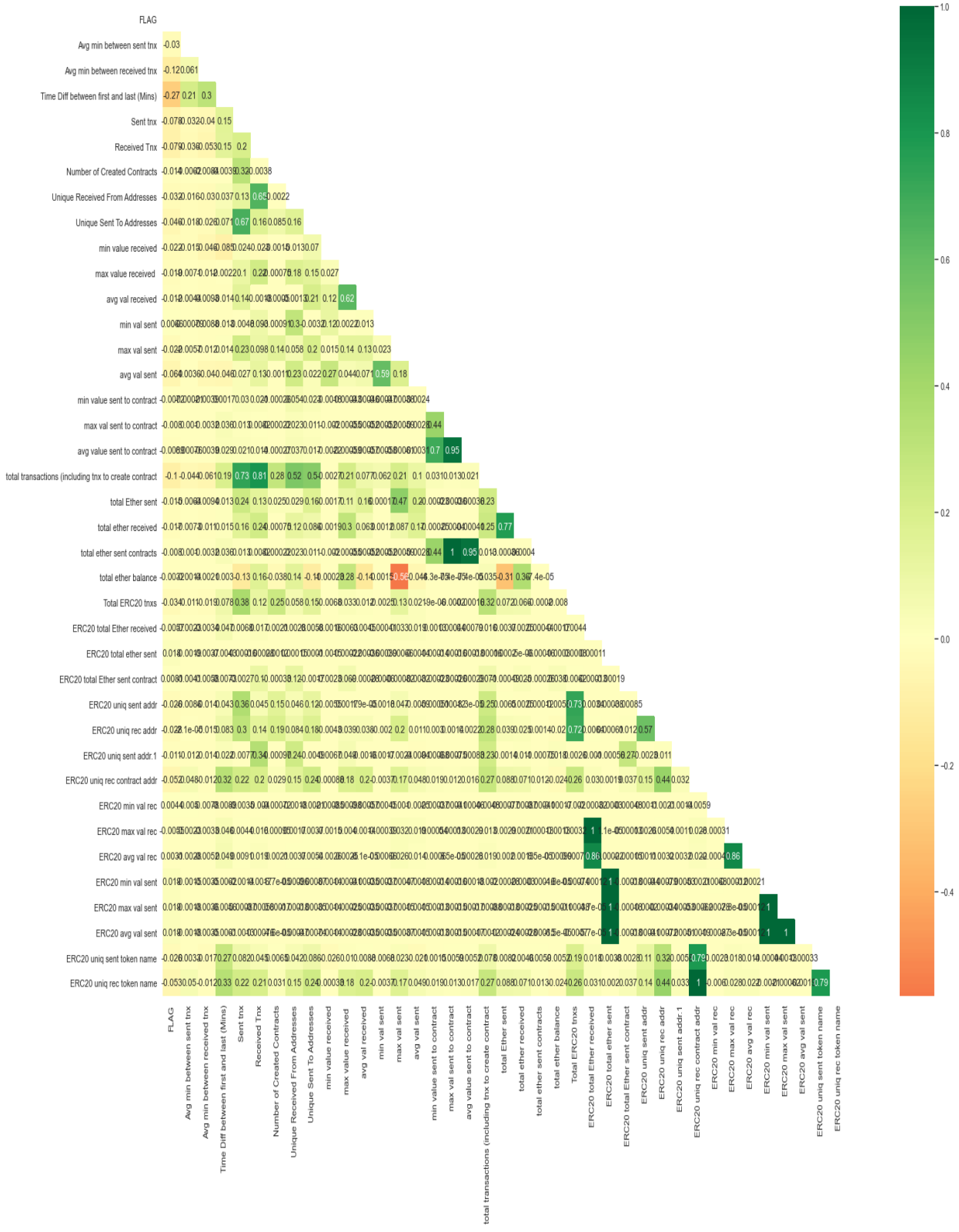

Figure 4.3.1.1 Heatmap of all attributes

The heat map after removing the variables with high coefficient is shown below in Figure 4.3.1.2.

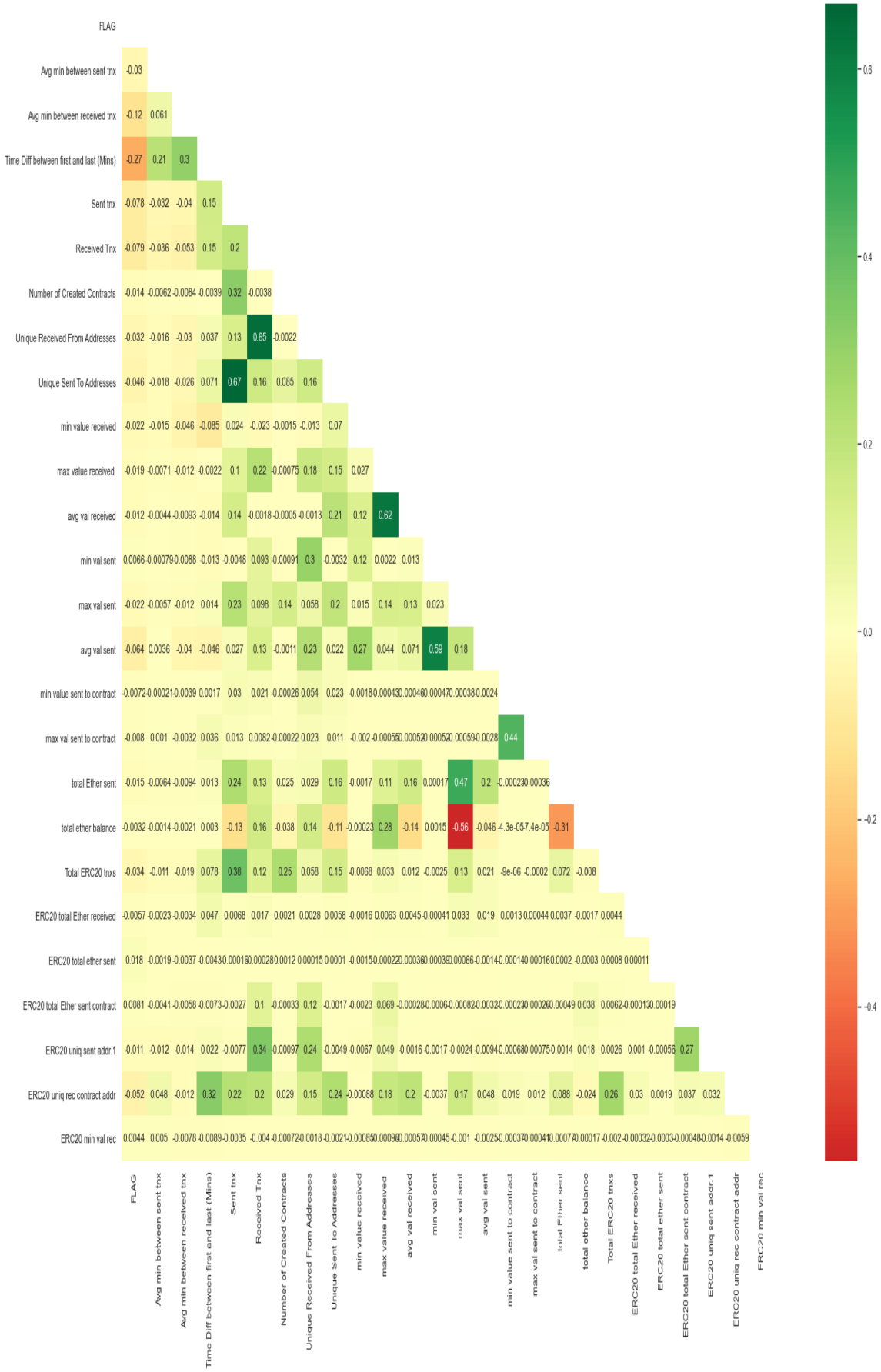

Figure 4.3.1.2 Heat map after removing the variables with high coefficient

4.3.2 Information Gain

The entropy decrease caused by a dataset modification is calculated using information gain. By assessing each variable's information gain in relation to the target variable, it can be used for feature selection.

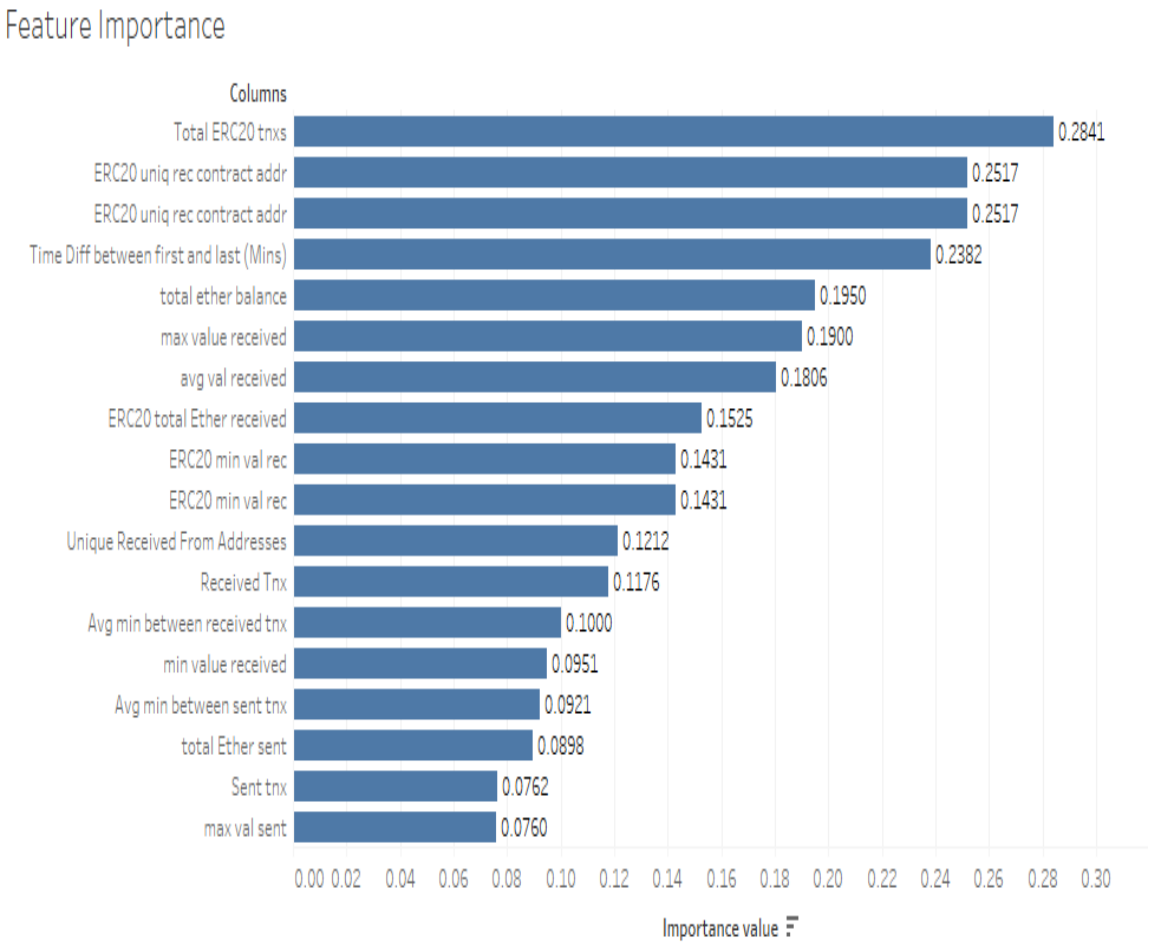

Figure 4.3.2.1 Importance value of the attributes after using Information Gain

The top 18 features having good importance value are selected and is shown in the Figure 4.3.2.1 above.

The Table 4.3.2.2 has the details of attributes along with Importance value after using the Information for the best feature selection.

Table 4.3.2.1 Attributes along with Importance value

Columns	Importance value
Avg min between sent tnx	0.092077
Avg min between received tnx	0.10003
Time Diff between first and last (Mins)	0.238181
Sent tnx	0.076205
Received Tnx	0.117618
Number of Created Contracts	0.006151
Unique Received From Addresses	0.121176
Unique Sent To Addresses	0.043216
min value received	0.095065
max value received	0.190043
avg val received	0.180622
min val sent	0.037793
max val sent	0.075986
avg val sent	0.075954
min value sent to contract	0
max val sent to contract	0
total Ether sent	0.089761
total ether balance	0.194996
Total ERC20 tnxs	0.284062
ERC20 total Ether received	0.152453
ERC20 total ether sent	0.003219
ERC20 total Ether sent contract	0
ERC20 uniq sent addr.1	0
ERC20 uniq rec contract addr	0.251734
ERC20 min val rec	0.143092
ERC20 uniq sent addr.1	0
ERC20 uniq rec contract addr	0.251734
ERC20 min val rec	0.143092

4.3.3 Univariate Analysis

From the Figure 4.3.3.1 Boxplot for Time between first and last (Mins) it is inferred that the data is right skewed. But there are also many outliers in the boxplot for Time Diff between first and last (mins).

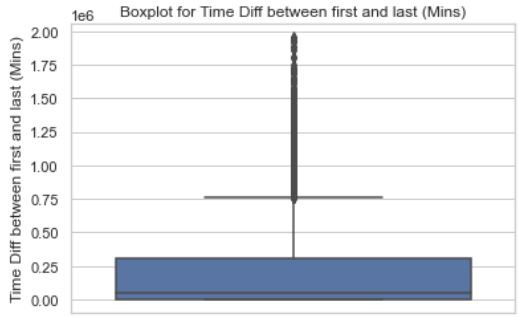

Figure 4.3.3.1 Boxplot for Time between first and last (Mins)

From the Figure 4.3.3.2 Boxplot for Number of Created Contracts is also skewed and has many outliers in range of 2000 to 4000 which can be accepted.

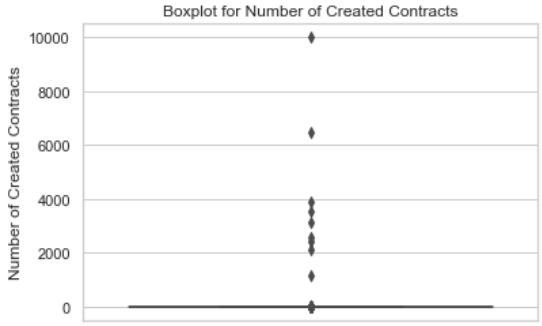

Figure 4.3.3.2 Boxplot for Number of Created Contracts

From the Figure 4.3.3.3 Boxplot for total ether balance is distributed evenly at the lower end and higher end.

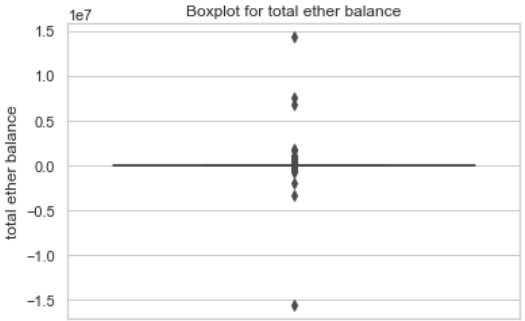

Figure 4.3.3.3 Boxplot for total ether balance

4.3.4 Bivariate Analysis

From the Figure 4.3.4.1 Boxplot for Total ERC20 tnxs the Inter Quartile range lies close to 0. But there are outliers in case of Flag 0. But in case of fraud scenario the value is close to 0 since the fraud transactions done are relatively low.

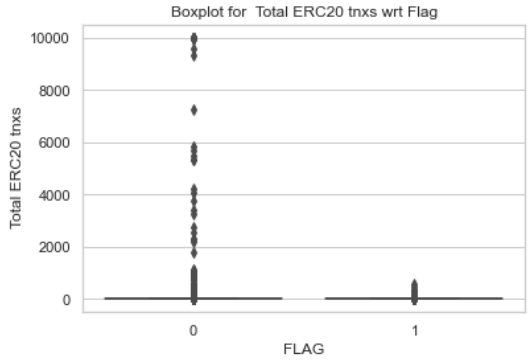

Figure 4.3.4.1 Boxplot for Total ERC20 tnxs wrt Flag

From the Figure 4.3.4.2 Boxplot for Average min between received tnx the Inter Quartile range lies close to 0. But there are outliers in case of Flag 0. But in case of fraud scenario the value is close to 0 since the fraud transactions done are relatively low.

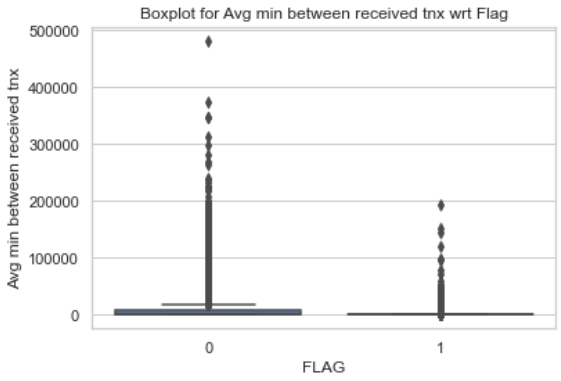

Figure 4.3.4.2 Boxplot for Average min between received tnx wrt Flag

From the Figure 4.3.4.3 Boxplot for Time diff between first and last (Mins) the Inter Quartile range is very less in case of fraud scenario but for the non-fraud cases the IQR is relatively more compared to the fraud and is right skewed. There are outliers in both fraud and non-fraud cases but the Time diff between first and last (Mins) in case of the fraud scenario is low compared to non-fraud.

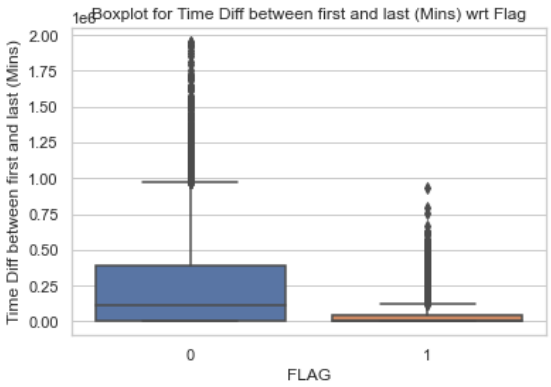

Figure 4.3.4.3 Boxplot for Time diff between first and last (Mins) w.r.t Flag

The Received tnx and total transactions (including tnx to create contract) attributes follow a linear relationship as shown in Figure 4.3.4.4. As the Recived tnx value increases the Total transactions (including tnx to create contract) value increases up to a limit of 10000.But Total transactions value appears to be mostly capped at a value of 10000.But in case of the fraud scenario Received tnx and total transactions value as mostly low compared to non-fraud (total transactions is mostly less than 2500).

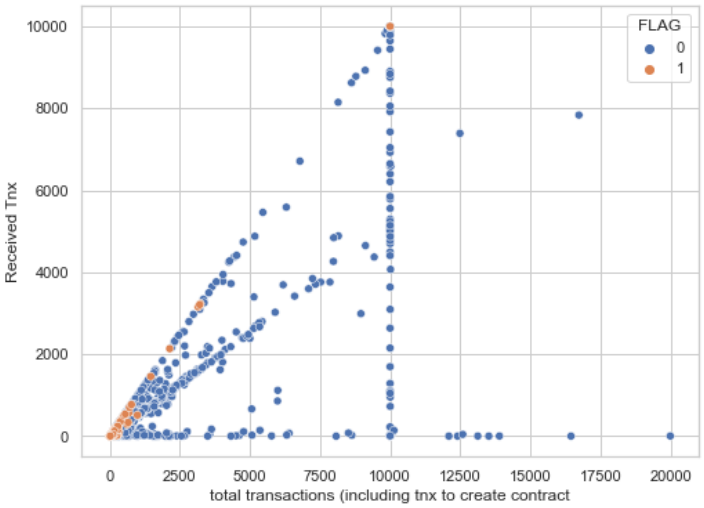

Figure 4.3.4.4 Scatter Plot of Received tnx and total transactions (including tnx to create contract)

The Sent tnx and Unique Sent to Addresses attributes follow a linear relationship as shown in Figure 4.3.4.5. As the Sent tnx value increases Unique Sent to Addresses value increases. But sent tnx value appears to be mostly capped at a value of 10000.But in case of the fraud scenario Sent tnx and Unique Sent to Addresses value as mostly low compared to non-fraud (Sent tnx transactions is mostly less than 500).

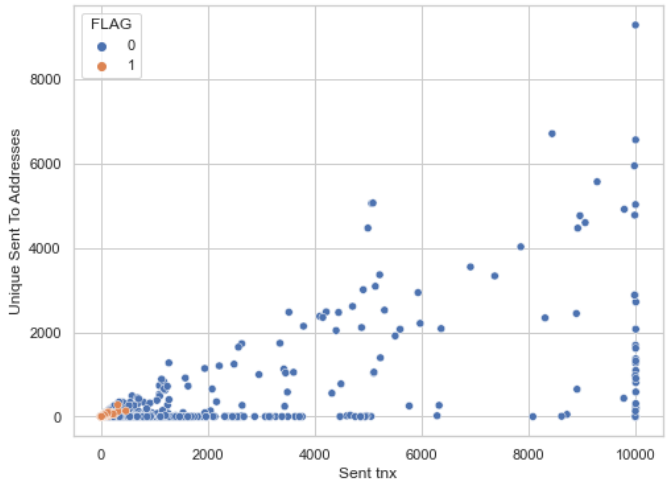

Figure 4.3.4.5 Scatter Plot of Sent tnx and Unique Sent to Addresses

The Total ERC20 tnxs and ERC20 uniq sent addr attributes follow a logarithmic relationship as shown in Figure 4.3.4.6. In case of the fraud scenario Total ERC20 tnxs and ERC20 uniq sent addr attribute value as mostly low compared to non-fraud (Total ERC20 tnxs transactions is mostly less than 500).

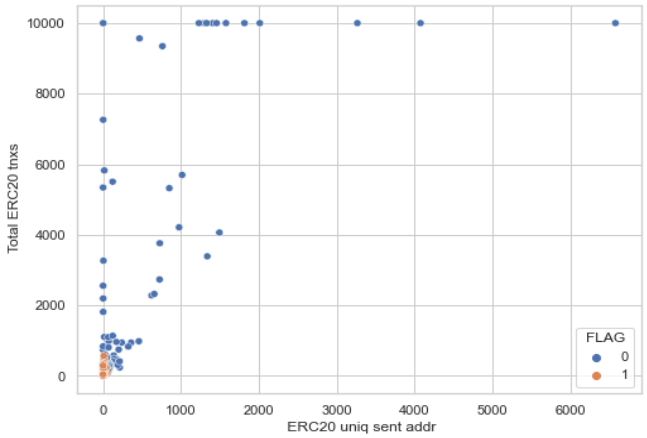

Figure 4.3.4.6 Scatter plot of Total ERC20 tnxs and ERC20 uniq sent addr

From the Figure 4.3.4.7 Boxplot for total ether balance is distributed evenly at the lower end and higher end for flag 0. But in case of fraud the Ether balance is low close to 0.

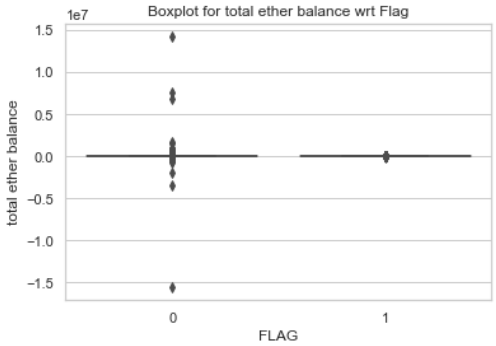

Figure 4.3.4.7 Boxplot for Total Ether balance w.r.t Flag

4.4 Implementation using Machine Learning

The Jupyter IDE is used to create and run code for various models. The dataset is added to the Jupyter folder. The notebook version is 6.1.4 and the Jupyter core version is 4.6.3. The code is written in the Python programming language.

The necessary libraries have all been imported and installed in the Jupyter notebook. The data is cleaned, and the 18 key attributes are decided upon based on features selection methods, correlation coefficient, and information gain. Next, the data is divided into a training set and a testing set. To bring attribute values to the same scale, the dataset is subjected to the power

transformation. On the training dataset, the imbalance approaches Over Sampling, SMOTE, and ADASYN are used.

The 10-fold randomized search cross validation technique is used to do hyper parameter tuning for various models. The best parameters and the best scores are obtained for the training set for different models.

4.4.1 Model 1-Logistic Regression

A key machine learning approach for predicting the binary result from various independent dataset features is logistic regression. The Logistic Regression using Randomized search Cross-validation is employed on the training set data for different parameters as shown in Figure 4.4.1.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
#Logistic regression parameters
params_LR = {'C':np.logspace(-1, 5, 10), 'class_weight':[None,'balanced'], 'penalty':['l1','l2']}
```

Figure 4.4.1.1 Hyper parameter Tuning parameters for Logistic Regression

Model obtained for Oversampling:

The best parameters for the Over Sampling Logistic Regression obtained after hyper-parameter tuning is:

'penalty': 'l2'

'class_weight': None

'C': 21544.346900318822

Model obtained for SMOTE:

The best parameters for the SMOTE Logistic Regression obtained after hyper-parameter tuning is:

'penalty': 'l2'

'class_weight': 'balanced'

'C': 215.44346900318823

Model obtained for ADASYN:

The best parameters for the ADASYN Logistic Regression obtained after hyper-parameter tuning is:

'penalty': 'l2'

'class_weight': 'balanced'

'C': 215.44346900318823

4.4.2 Model 2-Decision Tree

The Decision tree using Randomized search Cross-validation is employed on the training set data for different parameters as shown in Figure 4.4.2.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
#Decision tree parameters
params_DT = {
    'max_depth': [10, 20, 50, 100, 200],
    'min_samples_leaf': [10, 20, 50, 100, 200],
    'min_samples_split' : [10, 20, 50, 100, 200],
    'criterion': ["gini", "entropy"]
}
```

Figure 4.4.2.1 Hyper parameter Tuning parameters for Decision Tree

Model obtained for Oversampling:

The best parameters for the Oversampling Decision Tree obtained after hyper-parameter tuning is:

'min_samples_split': 100

'min_samples_leaf': 20

'max_depth': 100

'criterion': 'entropy'

Model obtained for SMOTE:

The best parameters for the SMOTE Decision Tree obtained after hyper-parameter tuning is:

'min_samples_split': 100

'min_samples_leaf': 20

'max_depth': 100

'criterion': 'entropy'

Model obtained for ADASYN:

The best parameters for the ADASYN Decision Tree obtained after hyper-parameter tuning is:

'min_samples_split': 100

'min_samples_leaf': 20

'max_depth': 100

'criterion': 'entropy'

4.4.3 Model 3-Random Forest

The Random Forest using Randomized search Cross-validation is employed on the training set data for different parameters as shown in Figure 4.4.3.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
#Random forest parameters
params_RF = {
    'n_estimators': [10,12,15],
    'max_features':['sqrt',0.3],
    'max_depth': [10,50],
    'min_samples_leaf': [50,200],
    'min_samples_split' : [50,100],
    'criterion': ["gini"]

}
```

Figure 4.4.3.1 Hyper parameter Tuning parameters for Random Forest

Model obtained for Oversampling:

The best parameters for the Oversampling Random Forest obtained after hyper-parameter tuning is:

'n_estimators': 15

'min_samples_split': 50

'min_samples_leaf': 50

'max_features': 0.3

'max_depth': 10

'criteron': 'gini'

Model obtained for SMOTE:

The best parameters for the SMOTE Random Forest obtained after hyper-parameter tuning is:

'n_estimators': 12

'min_samples_split': 50

'min_samples_leaf': 50

'max_features': 0.3

'max_depth': 10

'criteron': 'gini'

Model obtained for ADASYN:

The best parameters for the ADASYN Random Forest obtained after hyper-parameter tuning is:

'n_estimators': 15

'min_samples_split': 50

'min_samples_leaf': 50

'max_features': 0.3

'max_depth': 10

'criteron': 'gini'

4.4.4 Model 4-XGBoost

The XGBoost using Randomized search Cross-validation is employed on the training set data for different parameters shown in the Figure 4.4.4.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
#XGB parameters
params_XGB={
   'learning_rate':[0.01,0.1,0.3,0.5,0.7],
   'max_depth':[2,3,4,10],
   'n_estimators':[10,15,20,50,100,200],
   'subsample':[0.3, 0.5, 0.9],
   'colsample_bytree':[0.3,0.5,0.7],
   'max_features':[8,10,14,16]
}
```

Figure 4.4.4.1 Hyper parameter Tuning parameters for XGBoost

Model obtained for Oversampling:

The best parameters for the Oversampling XGBoost model obtained after hyper-parameter tuning is:

'subsample': 0.9

'n_estimators': 200

'max_features': 14

'max_depth': 10

'learning_rate': 0.3

'colsample_bytree': 0.5}

Model obtained for SMOTE:

The best parameters for the SMOTE XGBoost model obtained after hyper-parameter tuning is:

'subsample': 0.9

'n_estimators': 200

'max_features': 14

'max_depth': 10

'learning_rate': 0.3

'colsample_bytree': 0.5

Model obtained for ADASYN:

The best parameters for the ADASYN XGBoost model obtained after hyper-parameter tuning is:

'subsample': 0.9

'n_estimators': 200

'max_features': 14

'max_depth': 10

'learning_rate': 0.3

'colsample_bytree': 0.5

4.4.5 Model 5-AdaBoost

The AdaBoost model using Randomized search Cross-validation is employed on the training set data for different parameters as shown in Figure 4.4.5.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
#Adaboost parameters
params_ada={
  'learning_rate':[0.0001, 0.01, 0.1, 1.0, 1.1, 1.2,0.3,0.5,0.7],
  'n_estimators':[2,5,8,10,15,20,50]
}
```

Figure 4.4.5.1 Hyper parameter Tuning parameters for AdaBoost

Model obtained for Oversampling:

The best parameters for the Over Sampling AdaBoost model obtained after hyper-parameter tuning is:

'n_estimators': 50

'learning_rate': 0.3

Model obtained for SMOTE:

The best parameters for the SMOTE AdaBoost model obtained after hyper-parameter tuning is:

'n_estimators': 50

'learning_rate': 0.3

Model obtained for ADASYN:

The best parameters for the ADASYN AdaBoost model obtained after hyper-parameter tuning is:

'n_estimators': 50

'learning_rate': 0.3

4.4.6 Model 6-Gradient Boosting

The Gradient Boosting algorithm using Randomized search Cross-validation is employed on the training set data for different parameters is shown in Figure 4.4.6.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
#Gradient boosting parameters
params_gb={
  'learning_rate':[0.0001, 0.01, 0.1, 1.0, 1.1, 1.2,0.3,0.5,0.7],
  'n_estimators':[2,5,8,10,15,20,50,100]
}
```

Figure 4.4.6.1 Hyper parameter Tuning parameters for Gradient Boosting

Model obtained for OverSampling:

The best parameters for the Oversampling Gradient Boosting model obtained after hyper-parameter tuning is:

'n_estimators': 50

'learning_rate': 0.5

Model obtained for SMOTE:

The best parameters for the SMOTE Gradient Boosting model obtained after hyper-parameter tuning is:

'n_estimators': 50

'learning_rate': 0.5

Model obtained for ADASYN:

The best parameters for the ADSYN Gradient Boosting model obtained after hyper-parameter tuning is:

'n_estimators': 50

'learning_rate': 0.5

4.4.7 Model 7-Light Gradient Boosting Machine

The Light Gradient Boosting Machine using Randomized search Cross-validation is employed on the training set data for different parameters is shown in Figure 4.4.7.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
#Light Gradient boosting parameters
params_lgbm={
  'boosting_type':['gbdt','dart','rf'],
  'learning_rate':[0.0001, 0.01, 0.1, 1.0, 1.1, 1.2,0.3,0.5,0.7],
  'n_estimators':[2,5,8,10,15,20,50,100,200],
  'subsample':[0.3, 0.5, 0.9],
  'max_depth':[-1,2,3,4,5,10],
  'colsample_bytree':[0.3,0.5,0.7,1.]
}
```

Figure 4.4.7.1 Hyper parameter Tuning parameters for LGBM

Model obtained for OverSampling:

The best parameters for the Over Sampling Light Gradient Boosting Machine model obtained after hyper-parameter tuning is:

'subsample': 0.5
'n_estimators': 50
'max_depth': -1
'learning_rate': 0.1
'colsample_bytree': 0.3
'boosting_type': 'dart'

Model obtained for SMOTE:

The best parameters for the SMOTE Light Gradient Boosting Machine model obtained after hyper-parameter tuning is:

'subsample': 0.5
'n_estimators': 50
'max_depth': -1
'learning_rate': 0.1
'colsample_bytree': 0.3

'boosting_type': 'dart'

Model obtained for ADASYN:

The best parameters for the ADASYN Light Gradient Boosting Machine model obtained after hyper-parameter tuning is:

'subsample': 0.5

'n_estimators': 50

'max_depth': -1

'learning_rate': 0.1

'colsample_bytree': 0.3

'boosting_type': 'dart'

4.4.8 Model 8-CatBoost

The CatBoost using Randomized search Cross-validation is employed on the training set data for different parameters is shown in the Figure 4.4.8.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
55
56 #Catboost parameters
57 params_cat={
58   'boosting_type':["Ordered","Plain"],
59   'iterations':[100,200] ,
60   'learning_rate':[0.0001, 0.01, 0.1, 1.0, 1.1, 1.2,0.3,0.5,0.7],
61   'loss_function':['RMSE','Logloss','MAE','CrossEntropy','MAPE'],
62   'subsample':[0.3, 0.5, 0.9],
63   'depth':[-1,2,3,4,5,10]
64 }
65
```

Figure 4.4.8.1 Hyper parameter Tuning parameters for CatBoost

Model obtained for OverSampling:

The best parameters for the Over Sampling CatBoost model obtained after hyper-parameter tuning is:

'subsample': 0.3

'loss_function': CrossEntropy

'learning_rate': 1.0

'iterations': 200

'depth': 2

'boosting_type': 'Plain'

Model obtained for SMOTE:

The best parameters for the SMOTE CatBoost model obtained after hyper-parameter tuning is:

'subsample': 0.9

'loss_function': Logloss

'learning_rate': 0.7

'iterations': 100

'depth': 3

'boosting_type': 'Plain'

Model obtained for ADASYN:

The best parameters for the ADASYN CatBoost model obtained after hyper-parameter tuning is:

'subsample': 0.9

'loss_function': Logloss

'learning_rate': 0.7

'iterations': 100

'depth': 3

'boosting_type': 'Plain'

4.4.9 Model 9-Support Vector Machine

The Support Vector Machine using Randomized search Cross-validation is employed on the training set data for different parameters as shown in Figure 4.4.9.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
65
66 #SVM parameters
67 params_svc = {'C':[i for i in range(1,10,1)],'kernel':['linear','rbf','poly']}
68
```

Figure 4.4.9.1 Hyper parameter Tuning parameters for SVM

Model obtained for OverSampling:

The best parameters for the Over Sampling SVM model obtained after hyper-parameter tuning is:

'kernel': 'rbf

'C': 6

Model obtained for SMOTE:

The best parameters for the SMOTE SVM model obtained after hyper-parameter tuning is:

'kernel': 'rbf

'C': 6

Model obtained for ADASYN:

The best parameters for the ADASYN SVM model obtained after hyper-parameter tuning is:

'kernel': 'rbf

'C': 6

4.4.10 Model 10-KNN

The KNN using Randomized search Cross-validation is employed on the training set data for different parameters as shown in Figure 4.4.10.1 and data balancing techniques like SMOTE, ADASYN, Oversampling and the best parameter is obtained.

```
69 #KNN parameters
70 params_knn = {'n_neighbors':[i for i in range(1,25,1)],'algorithm':['kd_tree','auto'],'n_jobs':[-1]}
```

Figure 4.4.10.1 Hyper parameter Tuning parameters for KNN

Best model obtained for OverSampling:

The best parameters for the Over Sampling KNN model obtained after hyper-parameter tuning is:lka

'n_neighbors': 11

'n_jobs': -1

'algorithm': 'auto'

Model obtained for SMOTE:

The best parameters for the SMOTE KNN model obtained after hyper-parameter tuning is:

'n_neighbors': 11

'n_jobs': -1

'algorithm': 'auto'

Best model obtained for ADASYN:

The best parameters for the ADASYN KNN model obtained after hyper-parameter tuning is:

'n_neighbors': 11

'n_jobs': -1

'algorithm': 'auto'

4.4.11 Model 11-Naïve Bayes

One of the quick and simple machine learning techniques to predict a class of datasets is naive Bayes. The Naïve Bayes is employed on the training set data and data balancing techniques like SMOTE, ADASYN, Oversampling is utilized.

4.4.12 Artificial Neural Network

The ANN model is built with 5 hidden layers and an output layer using keras tensor flow on the SMOTE dataset with an epoch value of 10.

4.5 Summary

The suggested model should be able to make some accurate predictions quickly in order to reduce the risk of Ethereum blockchain fraud. There should be a trustworthy, effective, and precise technique for detecting scams in the Ethereum blockchain. To maximise consumer satisfaction through eliminating fraud in user-made transactions, the technique must also be appropriate for real-world applications. The dataset is used to apply modelling approaches, and the training set's findings are obtained. Additionally, the top model is picked and used on the testing dataset, which is covered in detail in chapter 5 that follows.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Introduction

This chapter compares the outcomes of every machine learning model used on the training data from the previous chapter. The most effective model is chosen based on performance indicators including Precision, Recall, F1-score, and execution time and put into use on the data from the testing set. The most important features that affect the detection of fraud in Ethereum platform are also listed in this chapter.

5.2 Evaluation of various models and Results

The following chart depicts the attributes that have a significant impact on the detection of illegitimate accounts during Ethereum transactions. "Total ERC20 tnxs," "ERC20 uniq rec contract addr," "Time Diff between first and last (Mins")," "total ether balance," and "max value received" are the top 5 attributes.

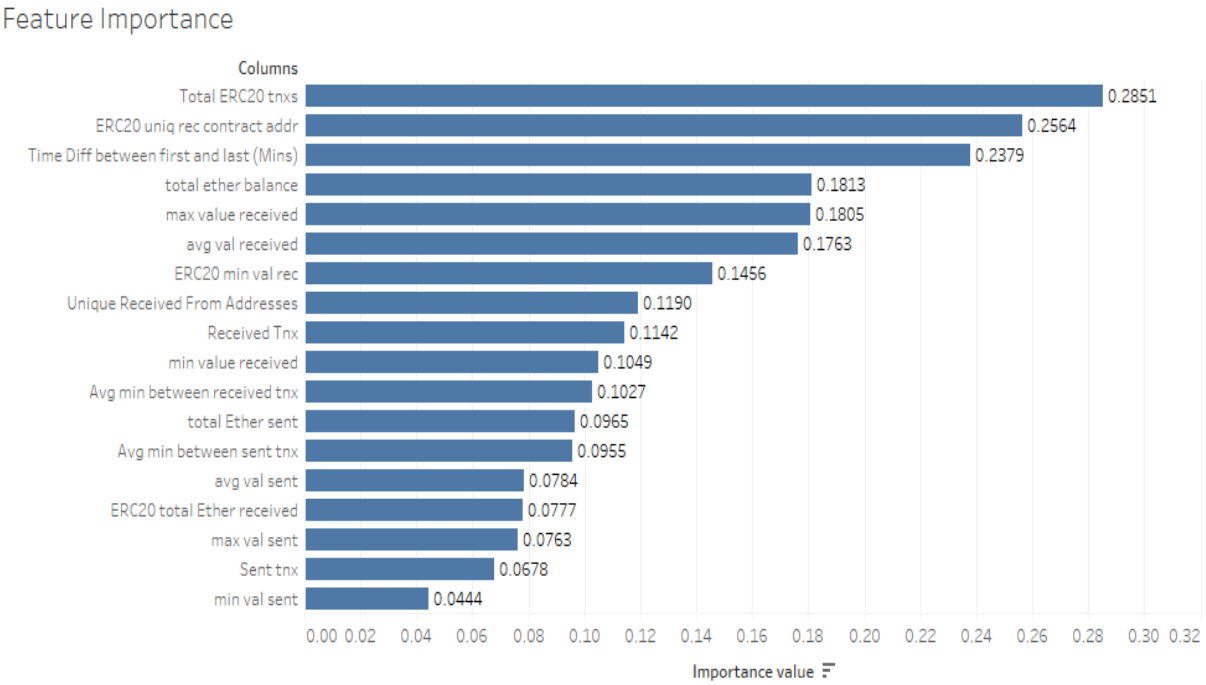

Figure 5.2.1 Top attributes along with importance value

As mentioned in the chapter 4 we have applied the following machine learning models on the training dataset: Logistic Regression, Decision Tree, Random Forest, XGBoost, AdaBoost,

Gradient Boosting, Light Gradient Boosting Machine, CatBoost, SVM, KNN, Naïve Bayes and ANN.

5.2.1 Model 1-Logistic Regression

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and Validation set for logistic regression are compared in the Table 5.2.1.1 below.

Table 5.2.1.1 Performance metrics of Logistic Regression for Training and Validation set

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.942241801	0.942241801	0.947299723	0.93718388	0.93781	0.9473	0.94253
	SMOTE	0.94770762	0.94770762	0.95154185	0.943873389	0.9443	0.95154	0.94791
	ADASYN	0.902959864	0.903000244	0.915072934	0.890846794	0.89406	0.91507	0.90445
Validation set	Over Sampling	0.95196062	0.943089431	0.967889908	0.936031332	0.81154	0.96789	0.88285
	SMOTE	0.948996335	0.946138211	0.95412844	0.94386423	0.82869	0.95413	0.88699
	ADASYN	0.928815244	0.899390244	0.981651376	0.875979112	0.69256	0.98165	0.81214

5.2.2 Model 2-Decision Tree

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for Decision Tree model are compared in the Table 5.2.2.1 below.

Table 5.2.2.1 Performance metrics of Decision Tree for Training and Validation set

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.963615598	0.963615598	0.974220917	0.953010279	0.95399	0.97422	0.964
	SMOTE	0.972018274	0.972018274	0.966226138	0.97781041	0.97755	0.96623	0.97186
	ADASYN	0.959066946	0.959102366	0.969692058	0.948441834	0.94983	0.96969	0.95966
Validation set	Over Sampling	0.944744533	0.942073171	0.949541284	0.939947781	0.81818	0.94954	0.87898
	SMOTE	0.955505785	0.961382114	0.944954128	0.966057441	0.88793	0.94495	0.91556
	ADASYN	0.950337749	0.93800813	0.972477064	0.928198433	0.79401	0.97248	0.87423

5.2.3 Model 3-Random Forest

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training and validation set for Random Forest model is compared in the Table 5.2.3.1 below.

Table 5.2.3.1 Performance metrics for the training and validation set for Random Forest

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.96467613	0.96467613	0.966226138	0.963126122	0.96324	0.96623	0.96473
	SMOTE	0.968999837	0.968999837	0.969978789	0.968020884	0.96802	0.96998	0.96998
	ADASYN	0.958928988	0.959021059	0.986547812	0.931310165	0.93531	0.98655	0.96025
Validation set	Over Sampling	0.966320933	0.965447154	0.967889908	0.964751958	0.88655	0.96789	0.92544
	SMOTE	0.970255348	0.966463415	0.97706422	0.963446475	0.88382	0.97706	0.9281
	ADASYN	0.948062136	0.931910569	0.97706422	0.919060052	0.77455	0.97706	0.8641

5.2.4 Model 4 -XGBoost

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for XGBoost model are compared in the Table 5.2.4.1 below.

Table 5.2.4.1 Performance metrics for the training and validation set for XGBoost

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	1	1	1	1	1	1	1
	SMOTE	1	1	1	1	1	1	1
	ADASYN	1	1	1	1	1	1	1
Validation set	Over Sampling	0.986256497	0.988821138	0.981651376	0.990861619	0.96833	0.98165	0.97494
	SMOTE	0.985603756	0.987804878	0.981651376	0.989556136	0.96396	0.96396	0.97273
	ADASYN	0.985286368	0.984756098	0.986238532	0.984334204	0.94714	0.98624	0.96629

5.2.5 Model 5-AdaBoost

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for AdaBoost model are compared in the Table 5.2.5.1 below.

Table 5.2.5.1 Performance metrics of Training set and validation set for AdaBoost

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.957905042	0.957905042	0.965573503	0.95023658	0.95099	0.96557	0.95823
	SMOTE	0.96459455	0.96459455	0.972263012	0.956926089	0.95758	0.97226	0.96486
	ADASYN	0.942307937	0.942434344	0.980226904	0.90438897	0.91167	0.98023	0.9447
Validation set	Over Sampling	0.955877069	0.949186992	0.967889908	0.94386423	0.83071	0.96789	0.89407
	SMOTE	0.958152682	0.955284553	0.963302752	0.953002611	0.85366	0.9633	0.90517
	ADASYN	0.933384435	0.906504065	0.981651376	0.885117493	0.70861	0.98165	0.82308

5.2.6 Model 6-Gradient Boosting

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for Gradient Boosting model are compared in the Table 5.2.6.1 below.

Table 5.2 6.1 Performance metrics of the Training and Validation set of the Gradient Boosting model

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.99608419	0.99608419	0.997715777	0.994452602	0.99447	0.99772	0.99609
	SMOTE	0.995431555	0.995431555	0.99608419	0.99477892	0.99479	0.99608	0.99543
	ADASYN	0.993968547	0.993983251	0.998379254	0.98955784	0.98972	0.99838	0.99403
Validation set	Over Sampling	0.985603756	0.987804878	0.981651376	0.989556136	0.96396	0.98165	0.97273
	SMOTE	0.984633626	0.983739837	0.986238532	0.983028721	0.94298	0.98624	0.96413
	ADASYN	0.978106211	0.973577236	0.986238532	0.96997389	0.90336	0.98624	0.94298

5.2.7 Model 7-Light Gradient Boosting Machine

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for Light Gradient B oosting Machine model are compared in the Table 5.2.7.1 below.

Table 5.2 7.1 Performance metrics of the Training and Validation set for LGBM

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.991434165	0.991434165	0.994126285	0.988742046	0.9888	0.99413	0.99146
	SMOTE	0.991434165	0.991434165	0.991678904	0.991189427	0.99119	0.99168	0.99168
	ADASYN	0.984261685	0.984307667	0.998055105	0.970468266	0.97145	0.99806	0.98457
Validation set	Over Sampling	0.978088246	0.978658537	0.97706422	0.979112272	0.93013	0.97706	0.95302
	SMOTE	0.976782763	0.976626016	0.97706422	0.976501305	0.92208	0.97706	0.94878
	ADASYN	0.9679977	0.955284553	0.990825688	0.945169713	0.83721	0.99083	0.90756

5.2.8 Model 8-CatBoost

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for CatBoost model are compared in the Table 5.2.8.1 below.

Table 5.2.8.1 Performance metrics of Training and Validation set for the CatBoost model

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.998776309	0.998776309	0.999836841	0.997715777	0.99772	0.99984	0.99878
	SMOTE	0.994615761	0.994615761	0.994942079	0.994289444	0.99429	0.99494	0.99462
	ADASYN	0.993233791	0.993233791	0.998541329	0.987926252	0.98813	0.99854	0.99331
Validation set	Over Sampling	0.983645531	0.984756098	0.981651376	0.985639687	0.95111	0.98165	0.96614
	SMOTE	0.984633626	0.983739837	0.986238532	0.983028721	0.94298	0.98624	0.96413
	ADASYN	0.978423599	0.976626016	0.981651376	0.975195822	0.91845	0.98165	0.949

5.2.9 Model 9-SVM

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for SVM model are compared in the Table 5.2.9.1 below.

Table 5.2.9.1 Performance metrics of the Training and validation set for the SVM Model

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.987028879	0.987028879	0.993147332	0.980910426	0.98114	0.99315	0.98711
	SMOTE	0.989149943	0.989149943	0.992005221	0.986294665	0.98629	0.99201	0.98918
	ADASYN	0.982075305	0.982112367	0.993192869	0.970957742	0.97177	0.99319	0.98237
Validation set	Over Sampling	0.983328143	0.981707317	0.986238532	0.980417755	0.93478	0.98624	0.95982
	SMOTE	0.981351953	0.983739837	0.97706422	0.985639687	0.95089	0.97706	0.9638
	ADASYN	0.96303926	0.965447154	0.958715596	0.967362924	0.89316	0.95872	0.92478

5.2.10 Model 10-KNN

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for KNN model are compared in the Table 5.2.10.1 below.

Table 5.2.10.1 Performance metric of the Training and Validation set for the KNN model

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.973486703	0.973486703	0.982215696	0.964757709	0.96536	0.98222	0.97372
	SMOTE	0.978218306	0.978218306	0.98955784	0.966878773	0.96761	0.98956	0.97846
	ADASYN	0.964431392	0.964549963	1	0.928862783	0.934	1	0.96587
Validation set	Over Sampling	0.970255348	0.966463415	0.97706422	0.963446475	0.88382	0.97706	0.9281
	SMOTE	0.972213572	0.969512195	0.97706422	0.967362924	0.89496	0.97706	0.93421
	ADASYN	0.957553836	0.93902439	0.990825688	0.924281984	0.78832	0.99083	0.87805

5.2.11 Model 11-Naïve Bayes

AUC, accuracy, sensitivity, specificity, precision, recall, and F1-score performance measures for different data imbalance approaches on training set and validation set for Naïve Bayes mo del are compared in the Table 5.2.11.1 below.

Table 5.2.11.1 Performance metric of the Training and Validation set for Naive Bayes model

Dataset	Sampling	AUC	Accuracy	Sensitivity	Specificity	Precision	Recall	F1_score
Training set	Over Sampling	0.829662261	0.829662261	0.918420623	0.740903899	0.77996	0.91842	0.84355
	SMOTE	0.837167564	0.837167564	0.92804699	0.746288138	0.78531	0.92805	0.85073
	ADASYN	0.779790448	0.780307342	0.934846029	0.624734867	0.71492	0.93485	0.81023
Validation set	Over Sampling	0.83929983	0.785569106	0.935779817	0.742819843	0.50873	0.93578	0.65913
	SMOTE	0.841910796	0.789634146	0.935779817	0.748041775	0.51385	0.93578	0.66341
	ADASYN	0.809716866	0.706300813	0.995412844	0.624020888	0.4297	0.99541	0.60028

5.2.12 Model 12-Artificial Neural Network

The accuracy of the model on Training set is obtained to be 97.62%. And when implemented the same on the Testing set the accuracy obtained is 97.30%.

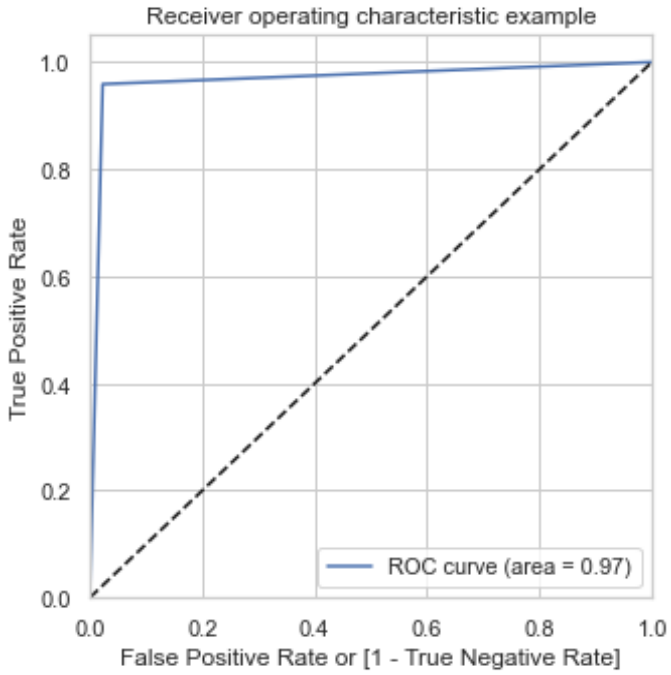

Figure 5.2.12.1 ROC curve for ANN Testing set

```
Accuracy    :  0.973082783138649
Sensitivity :  0.958904109589041
Specificity :  0.977139124755062
Precision   :  0.9230769230769231
Recall      :  0.958904109589041
F1_score: 0.940649496080627
[[1496   35]
 [  18  420]]
```

Figure 5.2.12.2 Performance metrics of Testing set using ANN

5.3 Comparison of various models and their Results

In this section, the results from our experiment when we applied the above-mentioned algorithms on the training data sets (Table 5.3.1) according to AUC, Accuracy, Sensitivity, Specificity, Precision, Recall, F-measure, time is illustrated.

Table 5. 3.1 Training set performance metrics

Training set									
Sampling	**Model**	**AUC**	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**Recall**	**F1_score**	**Time(seconds)**
Over Sampling	Logistic Regression	94.22%	94.22%	94.73%	93.72%	93.78%	94.73%	94.25%	6.435789108
SMOTE	Logistic Regression	94.77%	94.77%	95.15%	94.39%	94.43%	95.15%	94.79%	1.866598845
ADASYN	Logistic Regression	90.30%	90.30%	91.51%	89.08%	89.41%	91.51%	90.44%	2.059737682
Over Sampling	Decision Tree	96.36%	96.36%	97.42%	95.30%	95.40%	97.42%	96.40%	3.207585335
SMOTE	Decision Tree	97.20%	97.20%	96.62%	97.78%	97.76%	96.62%	97.19%	4.290286779
ADASYN	Decision Tree	95.91%	95.91%	96.97%	94.84%	94.98%	96.97%	95.97%	5.297066689
Over Sampling	Random Forest	96.47%	96.47%	96.62%	96.31%	96.32%	96.62%	96.47%	8.736284733
SMOTE	Random Forest	96.90%	96.90%	97.00%	96.80%	96.80%	97.00%	97.00%	10.54772305
ADASYN	Random Forest	95.89%	95.90%	98.65%	93.13%	93.53%	98.65%	96.02%	13.17775106
Over Sampling	Xgboosting	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	27.92805576
SMOTE	Xgboosting	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	32.03076625
ADASYN	Xgboosting	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	36.46115685
Over Sampling	AdaBoost	95.79%	95.79%	96.56%	95.02%	95.10%	96.56%	95.82%	12.6754396
SMOTE	AdaBoost	96.46%	96.46%	97.23%	95.69%	95.76%	97.23%	96.49%	0.823076923
ADASYN	AdaBoost	94.23%	94.24%	98.02%	90.44%	91.17%	98.02%	94.47%	17.83045125
Over Sampling	GradientBoosting	99.61%	99.61%	99.77%	99.45%	99.45%	99.77%	99.61%	36.70209265
SMOTE	GradientBoosting	99.54%	99.54%	99.61%	99.48%	99.48%	99.61%	99.54%	48.51211214
ADASYN	GradientBoosting	99.40%	99.40%	99.84%	98.96%	98.97%	99.84%	99.40%	66.73106313
Over Sampling	LightGradientBoosting	99.14%	99.14%	99.41%	98.87%	98.88%	99.41%	99.15%	5.009220839
SMOTE	LightGradientBoosting	99.14%	99.14%	99.17%	99.12%	99.12%	99.17%	99.17%	7.583759785
ADASYN	LightGradientBoosting	98.43%	98.43%	99.81%	97.05%	97.14%	99.81%	98.46%	5.233744621
Over Sampling	CatBoost	99.88%	99.88%	99.98%	99.77%	99.77%	99.98%	99.88%	14.88918948
SMOTE	CatBoost	99.46%	99.46%	99.49%	99.43%	99.43%	99.49%	99.46%	16.32202172
ADASYN	CatBoost	99.32%	99.32%	99.85%	98.79%	98.81%	99.85%	99.33%	15.30503607
Over Sampling	SVM	98.70%	98.70%	99.31%	98.09%	98.11%	99.31%	98.71%	147.8752818
SMOTE	SVM	98.91%	98.91%	99.20%	98.63%	98.63%	99.20%	98.92%	118.2155435
ADASYN	SVM	98.21%	98.21%	99.32%	97.10%	97.18%	99.32%	98.24%	202.0254724
Over Sampling	KNN	97.35%	97.35%	98.22%	96.48%	96.54%	98.22%	97.37%	9.265317917
SMOTE	KNN	97.82%	97.82%	98.96%	96.69%	96.76%	98.96%	97.85%	10.17277813
ADASYN	KNN	96.44%	96.45%	100.00%	92.89%	93.40%	100.00%	96.59%	9.66738987
Over Sampling	Naïve bayes	82.97%	82.97%	91.84%	74.09%	78.00%	91.84%	84.35%	0.288635254
SMOTE	Naïve bayes	83.72%	83.72%	92.80%	74.63%	78.53%	92.80%	85.07%	0.218610287
ADASYN	Naïve bayes	77.98%	78.03%	93.48%	62.47%	71.49%	93.48%	81.02%	0.5149436

The Table makes it very clear that the Gradient Boosting, CatBoost, Light Gradient Boosting, and SVM models outperform all others in terms of performance. The XGB model seems to be over fitting. Despite having superior precision and recall, the SVM model's execution time is relatively long. Since speed and time are equally important factors in fraud detection, we would prefer to do away with the SVM model.

It is abundantly clear when comparing the remaining three models—Gradient Boosting, CatBoost, and Light Gradient Boosting—that oversampling and SMOTE approaches outperform ADASYN in handling the data imbalance. For oversampling and SMOTE, the precision and recall values for the three models are nearly identical. In contrast to over-sampling, however, SMOTE requires more time. As a result, we determine that over sampling data imbalance handling technique is ideal for the fraud detection in the Ethereum transaction.

The Table 5.3.1 below shows the performance metrics of the model on the validation set.

Table 5.3.2 Validation set performance metrics

Validation set								
Sampling	**Model**	**AUC**	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**Recall**	**F1_score**
Over Sampling	Logistic Regression	95.20%	94.31%	96.79%	93.60%	81.15%	96.79%	88.28%
SMOTE	Logistic Regression	94.90%	94.61%	95.41%	94.39%	82.87%	95.41%	88.70%
ADASYN	Logistic Regression	92.88%	89.94%	98.17%	87.60%	69.26%	98.17%	81.21%
Over Sampling	Decision Tree	94.47%	94.21%	94.95%	93.99%	81.82%	94.95%	87.90%
SMOTE	Decision Tree	95.55%	96.14%	94.50%	96.61%	88.79%	94.50%	91.56%
ADASYN	Decision Tree	95.03%	93.80%	97.25%	92.82%	79.40%	97.25%	87.42%
Over Sampling	Random Forest	96.63%	96.54%	96.79%	96.48%	88.66%	96.79%	92.54%
SMOTE	Random Forest	97.03%	96.65%	97.71%	96.34%	88.38%	97.71%	92.81%
ADASYN	Random Forest	94.81%	93.19%	97.71%	91.91%	77.45%	97.71%	86.41%
Over Sampling	Xgboosting	98.63%	98.88%	98.17%	99.09%	96.83%	98.17%	97.49%
SMOTE	Xgboosting	98.56%	98.78%	98.17%	98.96%	96.40%	96.40%	97.27%
ADASYN	Xgboosting	98.53%	98.48%	98.62%	98.43%	94.71%	98.62%	96.63%
Over Sampling	AdaBoost	95.59%	94.92%	96.79%	94.39%	83.07%	96.79%	89.41%
SMOTE	AdaBoost	95.82%	95.53%	96.33%	95.30%	85.37%	96.33%	90.52%
ADASYN	AdaBoost	93.34%	90.65%	98.17%	88.51%	70.86%	98.17%	82.31%
Over Sampling	GradientBoosting	98.56%	98.78%	98.17%	98.96%	96.40%	98.17%	97.27%
SMOTE	GradientBoosting	98.46%	98.37%	98.62%	98.30%	94.30%	98.62%	96.41%
ADASYN	GradientBoosting	97.81%	97.36%	98.62%	97.00%	90.34%	98.62%	94.30%
Over Sampling	LightGradientBoosting	97.81%	97.87%	97.71%	97.91%	93.01%	97.71%	95.30%
SMOTE	LightGradientBoosting	97.68%	97.66%	97.71%	97.65%	92.21%	97.71%	94.88%
ADASYN	LightGradientBoosting	96.80%	95.53%	99.08%	94.52%	83.72%	99.08%	90.76%
Over Sampling	CatBoost	98.36%	98.48%	98.17%	98.56%	95.11%	98.17%	96.61%
SMOTE	CatBoost	98.46%	98.37%	98.62%	98.30%	94.30%	98.62%	96.41%
ADASYN	CatBoost	97.84%	97.66%	98.17%	97.52%	91.85%	98.17%	94.90%
Over Sampling	SVM	98.33%	98.17%	98.62%	98.04%	93.48%	98.62%	95.98%
SMOTE	SVM	98.14%	98.37%	97.71%	98.56%	95.09%	97.71%	96.38%
ADASYN	SVM	96.30%	96.54%	95.87%	96.74%	89.32%	95.87%	92.48%
Over Sampling	KNN	97.03%	96.65%	97.71%	96.34%	88.38%	97.71%	92.81%
SMOTE	KNN	97.22%	96.95%	97.71%	96.74%	89.50%	97.71%	93.42%
ADASYN	KNN	95.76%	93.90%	99.08%	92.43%	78.83%	99.08%	87.80%
Over Sampling	Naïve bayes	83.93%	78.56%	93.58%	74.28%	50.87%	93.58%	65.91%
SMOTE	Naïve bayes	84.19%	78.96%	93.58%	74.80%	51.39%	93.58%	66.34%
ADASYN	Naïve bayes	80.97%	70.63%	99.54%	62.40%	42.97%	99.54%	60.03%

From the table it is clearly evident that the Gradient Boost, CatBoost, SVM algorithms Over Sampling and SMOTE models have performed well. But based on the execution time the Gradient Boost and CatBoost Over sampling models have better performance.

5.4 Gradient Boost model

Figure below shows the confusion matrix for Gradient Boosting Model. Out of the total Over sampled training dataset, the gradient boost model has identified 48 cases wrongly. There are 14 False Negative values and 34 False Positive values in the training set. The same model when

applied on the validation set has identified 12 cases wrongly. There are 4 False Negative values and 8 False Positive values in the validation set.

Training set metrics

Gradient Boost Classifiers has curves closer to the top-left corner for Receiver operating char acteristic curve for the training set which indicates a better performance.

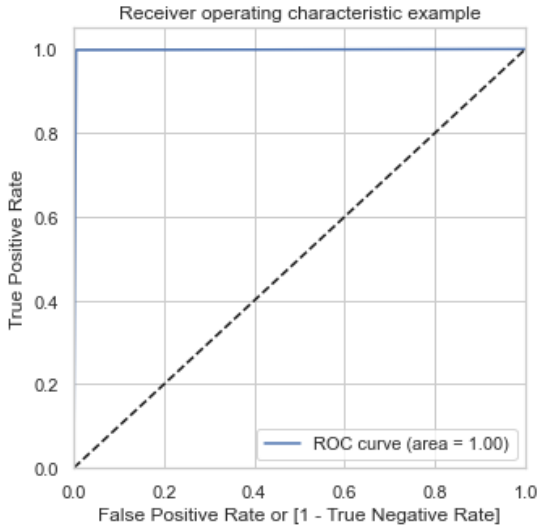

Figure 5.4.1 ROC curve of Training set for Over sampling Gradient Boost Model

AUC for the Gradient Boosting Model over sampling technique: 0.9960841899

Accuracy: 0.996084189916789

Sensitivity: 0.9977157774514602

Specificity: 0.9944526023821177

Precision: 0.9944706456334363

Recall: 0.9977157774514602

F1_score: 0.9960905684964978

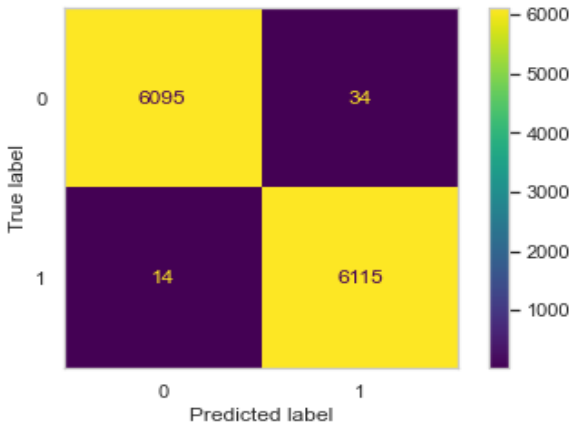

Figure 5.4.2 Confusion Matrix of Training set for Over sampling Gradient Boost Model

Validation set metrics

Gradient Boost Classifiers has curves closer to the top-left corner for Receiver operating char acteristic curve for the validation set which indicates a better performance.

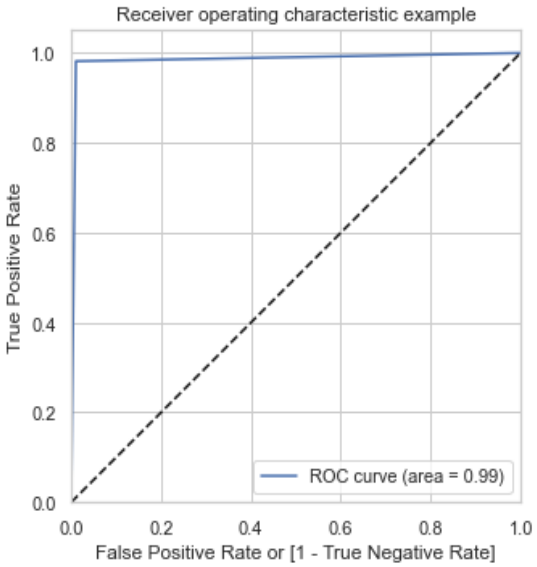

Figure 5.4.3 ROC curve of Validation set for Over sampling Gradient Boost Model

AUC for the Gradient Boosting Model over sampling technique 0.9856037559585119

Accuracy: 0.9878048780487805

Sensitivity: 0.981651376146789

Specificity: 0.9895561357702349

Precision: 0.963963963963964

Recall: 0.981651376146789

F1_score: 0.9727272727272728

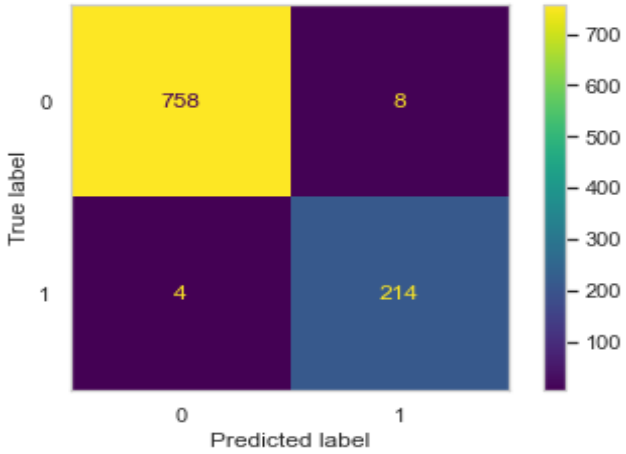

Figure 5.4.4 Confusion Matrix of Validation set for Over sampling Gradient Boost Model

5.5 Performance of the model on Testing set

The False Negative value has to be comparatively low as the illicit transaction accounts has been detected correctly and flagged/blocked. Also, since Ethereum platform is pseudo-anonymous we cannot afford to have many False Positives as this will affect our customer experience in case a valid transaction is flagged as fraud. Therefore, it necessary to keep both FN and FP low. When compared to other models, Gradient descent model has better precision, Recall and relatively good timing, hence we choose the Gradient Descent model with Over sampling data balancing technique as the best model.

Gradient Boosting Classifier has curves closer to the top-left corner for Receiver operating characteristic curve for the testing set data (Figure 5.5.1) which indicates a better performance.

best estimator: GradientBoostingClassifier(learning_rate=0.5, n_estimators=50)

Gradient boosting involves sequentially building and adding 50 trees to the model (n_estimators =50). The gradient boosting model uses a technique called the learning rate (0.5) to slow down the learning.

When this Gradient Boost model is applied on the Testing set the following results are obtained.

AUC for the Gradient Boosting Model over sampling technique 0.9892079231606521

Accuracy: 0.9908629441624366

Sensitivity: 0.9862385321100917

Specificity: 0.9921773142112125

Precision: 0.9728506787330317

Recall: 0.9862385321100917

F1_score: 0.979498861047836

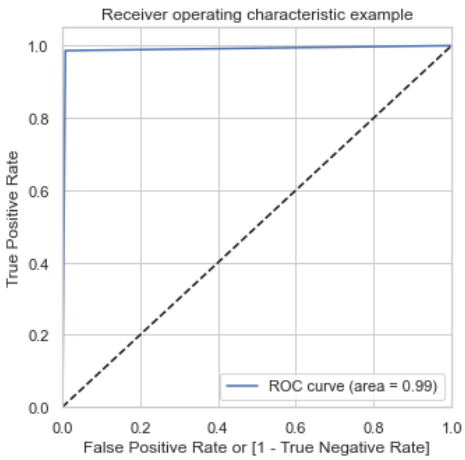

Figure 5.5.1 ROC curve of Testing set for Over sampling Gradient Boost Model

Figure 5.5.2 below shows the confusion matrix for Gradient Boosting Model of the Testing set. Out of the total testing dataset, the gradient boost model has identified 9 cases wrongly. There are 3 False Negative values and 6 False Positive values.

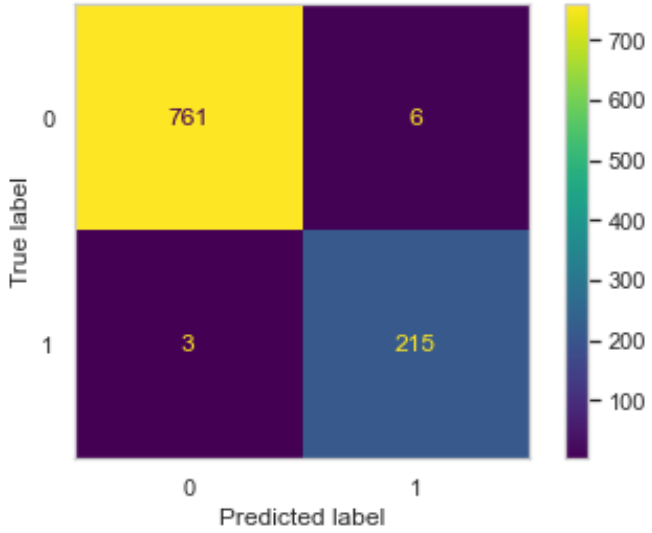

Figure 5.5.2 Confusion Matrix of Testing set for Over sampling Gradient Boost Model

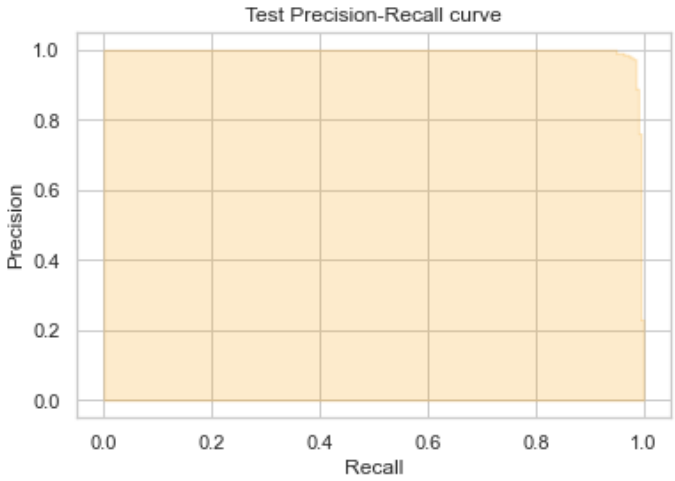

Figure 5.5.3 Precision-Recall curve of Testing set for Over sampling Gradient Boost Model

The Precision-Recall curve for the Testing set is shown in Figure 5.5.3 above. A high area under the curve represents both high recall and high precision,

5.6 Summary

The Over-sampling data balancing technique performs better when compared to other techniques in case of Ethereum fraud. The False Negative value has to be comparatively low as the illicit transaction accounts has been detected correctly and flagged/blocked. Also, since Ethereum platform is pseudo-anonymous we cannot afford to have many False Positives as this will affect our customer experience in case a valid transaction is flagged as fraud. Therefore, it necessary to keep both FN and FP low. When compared to other models, Gradient descent model has better precision, Recall and execution time is low hence we choose the Gradient Descent model with Over sampling data balancing technique as the best model. On the training set data, it has an Accuracy of 99.6%, Precision of 99.45%, Recall of 99.7% and F1-score of 99.6% and on validation set, it has Accuracy of 98.78%, Precision of 96.4%, Recall of 98.17% and F1-score of 97.27%. When the Gradient descent model is applied on the testing data set it has an Accuracy of 99.08%, Precision of 97.29%, Recall of 98.62%, F1-score of 97.95%.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Introduction

In this study, we explored the detection of unauthorized accounts on the Ethereum blockchain using a variety of models, including Logistic Regression, Decision Tree, Random Forest, XGBoost, Gradient Boost, LGBM, CatBoost, KNN, SVM, Naive Bayes, and Artificial Neural Networks. We then proposed the best fraud detection model, which combines Gradient Boost Classifier with Over Sampling data balancing.

6.2 Limitation

The dataset is not directly scraped from the Ethereum platform. We utilize a sampled transaction dataset obtained from the Kaggle website (vagifa, 2022). However, since the Ethereum Live dataset is enormous and updates every minute, the required technology (GPU) should be powerful and highly effective. Metrics like F1-score, Accuracy, and Precision Recall are used to assess the model performance. When choosing the best model, the cost of implementation is not taken into account.

6.3 Recommendations and Future Work

Despite the positive outcomes of our research and that of others in automating the identification of fraudulent accounts and transactions in blockchain frameworks, we still need to increase end users' awareness of these accounts and transactions and improve their knowledge of them. Also, should ensure that the high model performance is true while employing the same algorithms in subsequent works on a larger data set. On the Ethereum dataset, a basic artificial neural network model was used in this study. Although employing ANN for Ethereum fraud detection proved effective, machine learning models still outperformed it. To improve ANN's performance, we can try applying hyper parameter tuning. To see if different neural network models perform better than machine learning models, try experimenting with convolution neural network, recurrent neural network, MLPs, etc in the future.

6.4 Access to Resource

The Ethereum dataset used for modelling, model building and EDA python codes, the results are uploaded in the GitHub. Link to my GitHub account: (https://github.com/Madhu2409/Master-Thesis).

6.5 Conclusion

In this research, a model is developed to identify the illicit transaction in the Ethereum blockchain. The important attributes in the transaction dataset are selected using the correlation coefficient and information gain. The Over Sampling data balance technique is used to adjust the imbalance in classes then the Gradient Boost model accurately and quickly detects fraud and non-fraud cases. The Gradient descent model applied on the testing data set it has an Accuracy of 99.08%, Precision of 97.29%, Recall of 98.62%, F1-score of 97.95%. This shows that our model is robust enough to generalize to real-life problems. The best model can be used to identify the fraud accounts on the Blockchain platform, thereby enhancing security and gaining the confidence of its end users.

REFERENCES

Aljofey, A., Rasool, A., Jiang, Q. and Qu, Q., (2022) A Feature-Based Robust Method for Abnormal Contracts Detection in Ethereum Blockchain. *Electronics*, [online] 1118, p.2937. Available at: https://www.mdpi.com/2079-9292/11/18/2937 [Accessed 11 Dec. 2022].

Aziz, R.M., Baluch, M.F., Patel, S. and Ganie, A.H., (2022) LGBM: a machine learning approach for Ethereum fraud detection. International Journal of Information Technology. [online] Available at: https://link.springer.com/10.1007/s41870-022-00864-6.

Baek, H., Oh, J., Kim, C.Y. and Lee, K., (2019) A Model for Detecting Cryptocurrency Transactions with Discernible Purpose. In: *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*. [online] IEEE, pp.713–717. Available at: https://ieeexplore.ieee.org/document/8806126/.

Bartoletti, M., Carta, S., Cimoli, T. and Saia, R., (2017) Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact. *Future Generation Computer Systems*, [online] 102, pp.259–277. Available at: http://arxiv.org/abs/1703.03779 [Accessed 19 Dec. 2022].

Chen, W., Guo, X., Chen, Z., Zheng, Z. and Lu, Y., (2020) *Phishing Scam Detection on Ethereum: Towards Financial Security for Blockchain Ecosystem*.

Chen, W., Zheng, Z., Cui, J., Ngai, E., Zheng, P. and Zhou, Y., (2018) Detecting Ponzi Schemes on Ethereum. In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. [online] New York, New York, USA: ACM Press, pp.1409–1418. Available at: http://dl.acm.org/citation.cfm?doid=3178876.3186046.

Chen, W., Zheng, Z., Ngai, E.C.-H., Zheng, P. and Zhou, Y., (2019a) Exploiting Blockchain Data to Detect Smart Ponzi Schemes on Ethereum. *IEEE Access*, [online] 7, pp.37575–37586. Available at: https://ieeexplore.ieee.org/document/8668768/.

Chen, W., Zheng, Z., Ngai, E.C.-H., Zheng, P. and Zhou, Y., (2019b) Exploiting Blockchain Data to Detect Smart Ponzi Schemes on Ethereum. *IEEE Access*, [online] 7, pp.37575–37586. Available at: https://ieeexplore.ieee.org/document/8668768/ [Accessed 16 Dec. 2022].

Elmougy, Y. and Manzi, O., (2021) Anomaly Detection on Bitcoin, Ethereum Networks Using GPU-accelerated Machine Learning Methods. In: *2021 31st International Conference on*

Computer Theory and Applications (ICCTA). [online] IEEE, pp.166–171. Available at: https://ieeexplore.ieee.org/document/9916625/.

Fan, S., Fu, S., Xu, H. and Cheng, X., (2021) Al-SPSD: Anti-leakage smart Ponzi schemes detection in blockchain. *Information Processing & Management*, [online] 584, p.102587. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0306457321000856 [Accessed 19 Dec. 2022].

Farrugia, S., Ellul, J. and Azzopardi, G., (2020) Detection of illicit accounts over the Ethereum blockchain. *Expert Systems with Applications*, [online] 150, p.113318. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0957417420301433.

Ferreira Torres, C., Steichen, M. and State, R., (n.d.) *The Art of The Scam: Demystifying Honeypots in Ethereum Smart Contracts*. [online] Available at: https://www.usenix.org/conference/usenixsecurity19/presentation/ferreira.

('Fletcher', 'Emma '), (2022) *Reports show scammers cashing in on crypto craze*. [online] *FEDERAL TRADE COMMISSION*. Available at: https://www.ftc.gov/news-events/data-visualizations/data-spotlight/2022/06/reports-show-scammers-cashing-crypto-craze [Accessed 26 Dec. 2022].

Hu, T., Liu, X., Chen, T., Zhang, X., Huang, X., Niu, W., Lu, J., Zhou, K. and Liu, Y., (2021) Transaction-based classification and detection approach for Ethereum smart contract. *Information Processing & Management*, [online] 582, p.102462. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0306457320309547.

Ibba, G., Pierro, G.A. and di Francesco, M., (2021) Evaluating Machine-Learning Techniques for Detecting Smart Ponzi Schemes. In: *2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. [online] IEEE, pp.34–40. Available at: https://ieeexplore.ieee.org/document/9474794/.

Ibrahim, R.F., Mohammad Elian, A. and Ababneh, M., (2021) Illicit Account Detection in the Ethereum Blockchain Using Machine Learning. In: *2021 International Conference on Information Technology (ICIT)*. [online] IEEE, pp.488–493. Available at: https://ieeexplore.ieee.org/document/9491653/.

Jung, E., le Tilly, M., Gehani, A. and Ge, Y., (2019) Data Mining-Based Ethereum Fraud Detection. In: *2019 IEEE International Conference on Blockchain (Blockchain)*. [online] IEEE, pp.266–273. Available at: https://ieeexplore.ieee.org/document/8946232/.

Liu, L., Tsai, W.-T., Bhuiyan, Md.Z.A., Peng, H. and Liu, M., (2022) Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Generation Computer Systems*, [online] 128, pp.158–166. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0167739X21003319 [Accessed 11 Dec. 2022].

M. Bartoletti, S.C.T.C. and R.S., (n.d.) *Dataset of ponzi schemes, 2017*. Available at: https://goo.gl/CvdxBp. [Accessed 20 Dec. 2022].

MacKenzie Sigalos, (n.d.) CNBC. [online] Available at: https://www.cnbc.com/2022/01/06/crypto-scammers-took-a-record-14-billion-in-2021-chainalysis.html [Accessed 26 Dec. 2022].

Maurya, A. and Kumar, A., (2022) Credit card fraud detection system using machine learning technique. In: *2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*. [online] IEEE, pp.500–504. Available at: https://ieeexplore.ieee.org/document/9865466/.

Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W. and Qijun, C., (2017) *A Review on Consensus Algorithm of Blockchain*.

Monamo, P., Marivate, V. and Twala, B., (2016a) Unsupervised learning for robust Bitcoin fraud detection. In: *2016 Information Security for South Africa (ISSA)*. [online] IEEE, pp.129–134. Available at: https://ieeexplore.ieee.org/document/7802939/.

Monamo, P.M., Marivate, V. and Twala, B., (2016b) A Multifaceted Approach to Bitcoin Fraud Detection: Global and Local Outliers. In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [online] IEEE, pp.188–194. Available at: http://ieeexplore.ieee.org/document/7838143/.

Monrat, A., Andersson, K. and Schelén, O., (2019) A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. [online] Available at: https://ieeexplore.ieee.org/document/8805074 [Accessed 26 Dec. 2022].

Nakamoto, S., (n.d.) *Bitcoin: A Peer-to-Peer Electronic Cash System*. [online] Available at: www.bitcoin.org.

Nerurkar, P., Bhirud, S., Patel, D., Ludinard, R., Busnel, Y. and Kumari, S., (2021) Supervised learning model for identifying illegal activities in Bitcoin. *Applied Intelligence*, [online] 516, pp.3824–3843. Available at: https://link.springer.com/10.1007/s10489-020-02048-w.

Nerurkar, P., Busnel, Y., Ludinard, R., Shah, K., Bhirud, S. and Patel, D., (2020) Detecting Illicit Entities in Bitcoin using Supervised Learning of Ensemble Decision Trees. In: *Proceedings of the 2020 10th International Conference on Information Communication and Management*. [online] New York, NY, USA: ACM, pp.25–30. Available at: https://dl.acm.org/doi/10.1145/3418981.3418984.

Ostapowicz, M. and Żbikowski, K., (2019) Detecting Fraudulent Accounts on Blockchain: A Supervised Approach. [online] Available at: http://arxiv.org/abs/1908.07886.

Pham, T. and Lee, S., (2016a) Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods. [online] Available at: http://arxiv.org/abs/1611.03941.

Pham, T. and Lee, S., (2016b) Anomaly Detection in the Bitcoin System - A Network Perspective. [online] Available at: https://arxiv.org/abs/1611.03942 [Accessed 25 Nov. 2022].

Poursafaei, F., Hamad, G.B. and Zilic, Z., (2020) Detecting Malicious Ethereum Entities via Application of Machine Learning Classification. In: *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. [online] IEEE, pp.120–127. Available at: https://ieeexplore.ieee.org/document/9223304/.

Surin Murugiah, (n.d.) https://www.theedgemarkets.com/article/what-honeypot-crypto-scam.

Tan, R., Tan, Q., Zhang, P. and Li, Z., (2021) Graph Neural Network for Ethereum Fraud Detection. In: *2021 IEEE International Conference on Big Knowledge (ICBK)*. [online] IEEE, pp.78–85. Available at: https://ieeexplore.ieee.org/document/9667674/.

Tsikerdekis, M., Zeadally, S., Schlesener, A. and Sklavos, N., (2018) Approaches for Preventing Honeypot Detection and Compromise. In: *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. [online] IEEE, pp.1–6. Available at: https://ieeexplore.ieee.org/document/8635603/.

vagifa, (2022) *Kaggle Ethereum Transaction dataset*. [online] Available at: https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset [Accessed 26 Dec. 2022].

Wang, L., Cheng, H., Zheng, Z., Yang, A. and Zhu, X., (2021a) Ponzi scheme detection via oversampling-based Long Short-Term Memory for smart contracts. *Knowledge-Based Systems*, [online] 228, p.107312. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0950705121005748 [Accessed 16 Dec. 2022].

Wang, W., Song, J., Xu, G., Li, Y., Wang, H. and Su, C., (2021b) ContractWard: Automated Vulnerability Detection Models for Ethereum Smart Contracts. *IEEE Transactions on Network Science and Engineering*, [online] 82, pp.1133–1144. Available at: https://ieeexplore.ieee.org/document/8967006/.

Weber, M., Domeniconi, G., Chen, J., Weidele, D.K.I., Bellei, C., Robinson, T. and Leiserson, C.E., (2019) Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. [online] Available at: http://arxiv.org/abs/1908.02591.

Yuan, Q., Huang, B., Zhang, J., Wu, J., Zhang, H. and Zhang, X., (2020) Detecting Phishing Scams on Ethereum Based on Transaction Records. In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. [online] IEEE, pp.1–5. Available at: https://ieeexplore.ieee.org/document/9180815/.

Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H., (2017) An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: *2017 IEEE International Congress on Big Data (BigData Congress)*. [online] IEEE, pp.557–564. Available at: http://ieeexplore.ieee.org/document/8029379/.

APPENDIX A

It clearly specifies our focus, goals and methods to be followed in the research from start to end along with the timeline and status. Also, the risks anticipated during the research phase andits contingency plan are also included.

The schedule or timeline for the completion of the dissertation is shown below in the form of a Gantt chart.

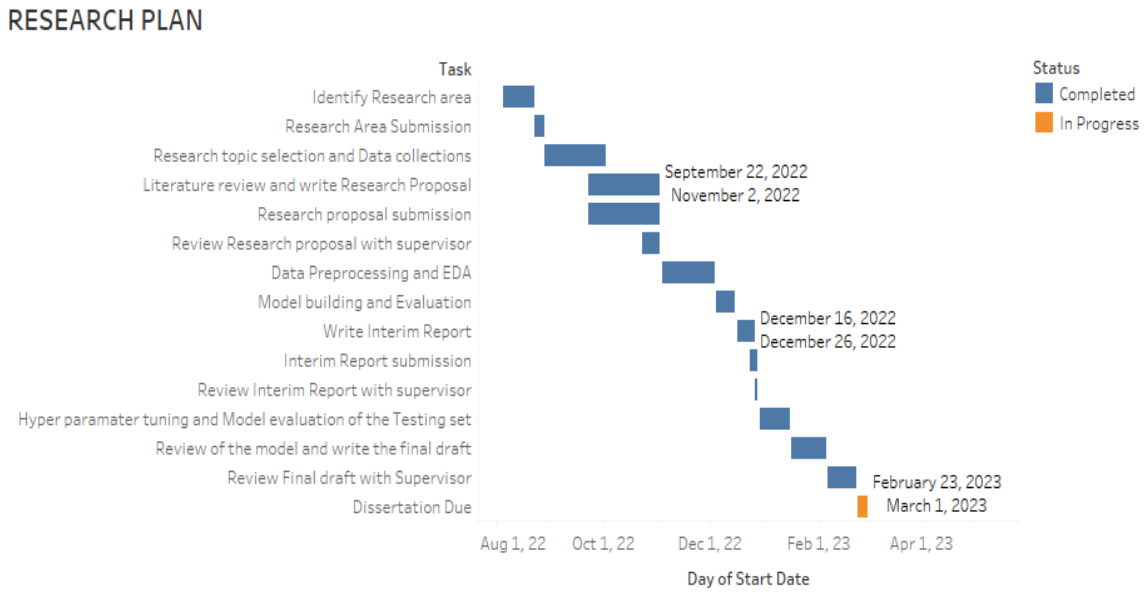

Figure 1: Gantt chart showing the schedule for completion of the dissertation

APPENDIX B

RESEARCH PROPOSAL

UTILIZING MACHINE LEARNING AND NEURAL NETWORK MODELS FOR DETECTING ILLICIT TRANSACTIONS IN ETHEREUM BLOCKCHAIN

MADHUMITHA RAMAJEYATHILAGAM

Research Proposal Report

October 2022

Abstract

People now conduct most of their business online, but fiat currencies still require the involvement of third parties, such as banks, to process transactions. By contrast, blockchain, a platform technology widely used by cryptocurrencies like Bitcoin and Ethereum, removes the need for third parties. It enables peer-to-peer transfers by performing transaction verification based on a consensus algorithm, removing a dependency. However, even Ethereum blockchains are vulnerable to security issues like phishing attacks, fraudulent transactions, and majority attacks. Ethereum frauds provide significant profits and seriously jeopardize the network's ability to maintain its financial stability. In this study, we use machine learning models to identify illicit accounts on the Ethereum Blockchain. The classification algorithms are applied to data downloaded from the Kaggle website. The best performing model is then proposed to identify the fraud accounts on the Blockchain platform, thereby enhancing security and gaining the confidence of its end users.

Table of Contents

1. Background

Blockchain is a decentralized, distributed, immutable database or ledger that enables the recording of transactions and the tracking of assets. By lowering the associated cost and risk, it makes it possible to practically track and trade anything of value. Cryptocurrency systems like Bitcoin, Ethereum, etc., NFTs, and Smart Contracts all depend on blockchains to keep a secure and decentralized record of transactions. In Blockchain, the data or transaction is added to a new block as it becomes available. The block is then connected to the preceding block, and any further transactions are connected as new blocks, one after the other, to create a chain of data. A specific block of data cannot be changed since it is securely linked to its neighboring blocks. Depending on the type of blockchain being used, these transactions are recorded in the ledger with the aid of consensus algorithms like Proof of Stake (PoS), Proof of Work (PoW), or Proof of Authority (PoA)(Mingxiao et al., 2017; Zheng et al., 2017).

Vitalik Buterin first suggested the Ethereum blockchain in 2014 (Buterin, n.d.), and the project was later launched in 2015. It is a Blockchain-based open-source decentralized network that generates its own cryptocurrency, called "ether." Next to Bitcoin, Ethereum is the most popular cryptocurrency on the market. It is used to build and execute Smart contracts without any fraud, downtime, or outside interference. It is also exchanged as digital money.

Every node, or computer, in the peer-to-peer (P2P) network receives a transaction with a digital signature from the Ethereum blockchain, and after receiving it, the nodes use a consensus method to verify the transaction. As a result, if the transaction is valid, the new block is added to the chain and the miner receives fees. By manipulating with gas cap and price of gas, these fees are created. The miner who has the greater quantity of money in his wallet has the right to validate the transaction and add it to the chain.

From the article published by Federal Trade Commission on June 3, 2022 it is clearly evident that around 46000 people have been scammed in the past 1 year reporting a loss of over $1billions in crypto. As per the article about 70% of the Bitcoin,9% of the Ether was used to pay to scammers(‘Fletcher’, 2022).

2. Problem Statement

Blockchain technology is the foundation of Ethereum, a decentralized, international software platform. Ether serves as its native cryptocurrency (ETH). Ethereum now employs the Proof

of Stake (PoS) process, in which a network of individuals known as validators creates new blocks and collaborates to confirm the data they contain. All of these things do not, however, imply that the Ethereum blockchain is faultless. Public keys are used to identify users of the Ethereum network. Since the identity of the user is concealed via a public key, it has caught the attention of lawbreakers. Blockchain technology has also faced significant security challenges, including majority assaults, fraudulent transactions, forking, Initial coin offerings, Ponzi schemes, pump and dumps, ransomware, wallet scam, and phishing scams. There are many studies performed to detect the illicit transactions in the Ethereum Blockchain and the process helps to stop the fraud.

1.1 Related Works

Numerous studies are conducted to find scams on the Ethereum network.

The Random Forest Algorithm has been suggested by the authors (Ibrahim et al., 2021) as a method for identifying fraudulent transactions on the Ethereum blockchain. They identified six features out of 42 that were the most successful in the prediction model using the Correlation Attribute Evaluator in the Weka software tool. Their findings reveal that the Random Forest algorithm significantly improved their time measurements and the F measure.

The authors (Farrugia et al., 2020) suggested a technique to spot illicit accounts on the Ethereum network and also highlighted the most crucial input features that affect the choice of fraudulent accounts. The "XGBoost" classification model was employed to categorize these fraudulent accounts.

The authors(Poursafaei et al., 2020) proposed a solution for detecting the illicit entities using ensemble methods like stacking and ADABoost classifier. They have used dataset obtained by extracting them from the Ethereum website. The PCA method is used for feature extraction and various sampling techniques like Under sampling, over-sampling and SMOTE was used. The modeling techniques Logistic regression, Support vector machine, Random Forest and ensemble methods were applied on the sampled dataset.

The graph neural network classification method was suggested by the authors (Tan et al., 2021) to identify fraudulent accounts on the Ethereum platform. The information was gathered by extracting fraudulent transactions from etherscamdb.info and legitimate transactions from etherscan.io. An algorithm based on random walks called network embedding is employed for the automatic feature extraction process. This method just considers how much information is

contained in the Ethereum transactions, not how the Ethereum-based transaction network is structured. It has been discovered that other graph neural network models, including GAT and GraphSAGE, perform better than the GCN model in many applications; as a result, the authors advise using these techniques in future efforts. In addition, they advise using their detection methods for additional unlawful Ethereum-based activities including gambling and money laundering.

A LGBM-based method for spotting Ethereum fraud transaction was suggested by the authors (Aziz et al., 2022). By up sampling the minority class to the same frequency as the majority class, they were able to balance the classes using the SMOTE technique. The power transform function is used to normalize the training characteristics. The model assists in accurately identifying unusual transactions while avoiding overfitting. With excellent efficiency and minimal memory consumption, it anticipates Ethereum transaction frauds.

In order to detect illegal transactions, the authors(Baek et al., 2019) employed an unsupervised learning expectation maximization (EM) approach to cluster the data set. They employed Random Forest to do an anomaly detection based on the features created by the unsupervised learning (RF). All of the clusters that served the same function were combined into one, while the remaining cluster contained a wallet that handled anomalous transactions.

Recall and false positive rate were suggested by the authors(Ostapowicz and Żbikowski, 2019) as crucial metrics for finding frauds in the Ethereum blockchain. The dataset that was taken from the Ethereum platform was subjected to 10-fold cross validation. On the dataset, they used Random Forest, XGBoost, and SVM models. In comparison to the other models, the Random Forest offered a lower false positive rate.

The authors (Yuan et al., 2020) suggested a three-step method to spot phishing scams on Ethereum transactions. Latent features are extracted using the node2vec network embedding technique. And one-class support vector machine (SVM) is used to classify the phishing scams.

1. Research Questions

For each of the study goals outlined below, the following research questions are suggested.

- How can we generate a generalized model to detect the illicit transactions in the Ethereum Blockchain.
- How is the dataset available for research structured. Does it further need any restructuring before implementing the models.

\

- What are the various attributes influencing in determining the illicit transaction and how well can we define them.
- What are the parameters tuning needed to improve the model performance .
- Does the performance metrics effectively contribute in determining the illicit transaction.

1. Aim and Objectives

The main aim of this research is to propose a model to detect the illicit transactions in the Ethereum Blockchain and in the process helps to stop the frauds. The identification of the fraudulent transactions can be done using machine learning classifiers. Various feature selection techniques and classifiers are used to compare and analyzed using various performance measures.

Based on the goals of this study, which are as follows, the research objectives have been developed:

- To utilize feature selection techniques to find the most important features of fraudulent transactions and visualization of the most important input features to examine the pattern and relationships between them.
- To make a reasonable balancing technique recommendation that can be used on the unbalanced dataset.
- To compare the predicted models and determine which one is more accurate to predict the illicit transaction accounts based on the important features selected.
- To assess how well proposed machine learning or neural network models perform.

2. Significance of the Study

Blockchain usage is now widespread among financial and non-financial areas. Financial security is essential for the healthy development of Blockchain technology. The proliferation of scams in the Ethereum blockchain will hinder the use of the technology and also reduces itsacceptance among users. Thus, it is necessary to identify all these scams in the Ethereum blockchain. To safeguard the Ethereum platform against these frauds and their negative effectson its reputation, a thorough investigation must be conducted to ascertain the many reasons that allow these illicit transactions to occur, their core cause, and how they can be identified, reduced, and avoided. This study employs machine learning techniques to identify fraudulent transactions on the Ethereum Blockchain. Once the model proposed has been integrated with the platform itself, end users can use the unlawful transaction information obtained to avoid

doing business with scammers based on the provided address. Additionally, interested authorities may take the appropriate measures against these addresses.

1. Scope of the Study

Due to the limited time available, some limitations must be made on this research in order for the project to be finished on schedule. The information that will be covered in this research study and that will be excluded from this study is listed in the following parts.

1.1 In-Scope

The results of this study are expected to deliver the following:

- Determining the illicit transaction and corresponding addresses of the users who are performing this transaction based on the machine learning approach.
- Able to identify the illicit transactions even in the imbalanced dataset.
- Lists out the main attributes that are useful in determining the illicit transactions.
- Comparison of various machine learning models pros and cons to determine the best model suitable for determining the illicit transaction.

1.2 Out-of-Scope

The following are not planned to be included in this study:

- The Ethereum platform is not directly scraped for the dataset. Instead, we make use of a transaction dataset found on the Kaggle website. (https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset)
- The cost of implementing the model is not taken into account while deciding the best model.
- This study shall not include developing an end-to-end system or prototype to be integrated with the Ethereum system to work on the huge live transactions that keeps updating continuously. Instead it shall be focusing on the modelling techniques only needed to detect the illicit transactions.
- This research work is also limited with Ethereum blockchain. For fraud transactions in other cryptocurrencies the models used are applicable but the performance of models is not expected to be similar to Ethereum.

2. Research Methodology

The methodology used entails crucial steps like choosing the target data, pre-processing the selected data, transforming the data into an organized and comprehensible format, adjusting

the dataset imbalance, putting machine learning techniques into practice, and evaluating the effectiveness of machine learning with evaluation metrics.

1.1 Workflow

The working of the model for the Ethereum illicit transaction has been proposed in this section.

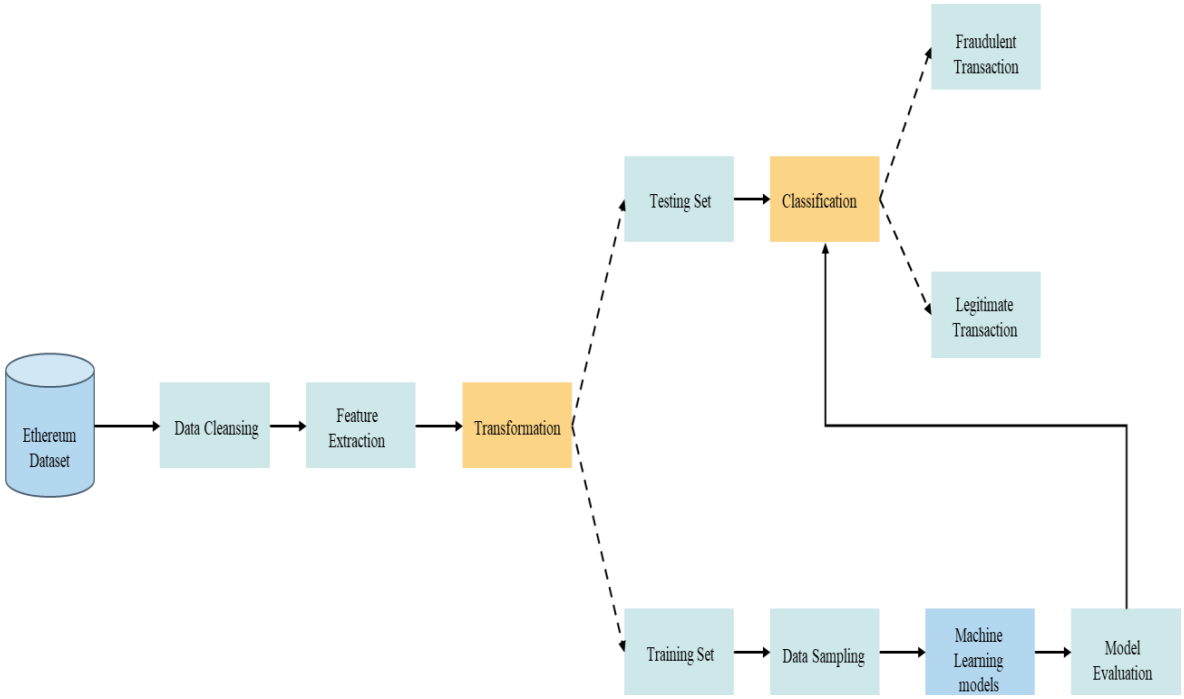

Figure 7.1: Flowchart for determining the Ethereum illicit Transactions

1.2 Dataset Description

The dataset to be used for processing is taken from the Kaggle site [https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset]. This dataset contains rows of Ethereum transactions, a kind of cryptocurrency, that have been found to be valid and fraudulent. The dataset has 9841 records and 51 columns. But the dataset is highly imbalanced which may affect the performance of model. The Flag attribute can be used as target variable.

1.3 Data Pre-processing

The Ethereum dataset and all the necessary libraries are imported. The dataset is imbalanced and may contain duplicate data so the first step is to perform the Data Cleaning. One of the most crucial processes in the model-building is data cleaning. It involves handling the missing values, removing any duplicates or irrelevant data, fixing structural errors, handling outliers. The important features can be selected using the methods like correlation matrix, Recursive feature elimination or Regularization methods like Lasso Regression.

The RFE takes the machine learning model and number of features as input. It then recursively reduces the number of features by ranking them based on the accuracy of model. We can then learn the names of the features that have been determined to be the most crucial using the RFE support method.

In the correlation matrix method based on the heatmap/correlation coefficient if the selected variables having high correlation values between them, then we need to keep just one of the correlated ones and drop the other.

If the input features are not helping to train our machine learning model in a good way, the coefficients of those features are reduced when using Lasso Regression. By assigning them coefficients equal to zero, some of the traits might be automatically eliminated in this way.

The Feature Exaction technique like Principal Component Analysis can also be used to reduce the original dimensions of the dataset. It is achieved by maximizing the variances and minimizing the reconstruction error.

Data Transformation is the process of converting the data to a format suitable for model building. It is performed since the algorithm might become biased if the data is skewed. And converting all the attribute to the same scale allows the algorithm to compare the features better. We can proceed with using Min Max scaler, Standard scaler or Power Transformer based on Exploratory Data Analysis performed on the dataset.

Then, the data is divided into a training set and a testing set in a ratio of 70:30. Since the data is imbalanced, we can proceed using Over sampling, SMOTE or ADASYN on the training set to balance the majority and minority class.

1.1 Modelling Techniques and Evaluation Metrics

A number of machine learning techniques, including Logistic Regression, Decision Tree, KNN, Naive Bayes, Ensemble models like Random Forest, XGBoosting, ADABoost, Light GBM, GBM, CatBoost, Neural Network etc. can be used on training dataset.

Then determine how the above algorithms have performed on the training set based on metrics such as Confusion matrix, Precision, Recall, F1-score, Accuracy and other metrics. Based on comparison for the above metrics for different models the best model is chosen. As soon as the best model has been chosen, it is tested on testing dataset to determine its performance on the unseen dataset.

2. Required Resources

Hardware Requirements:

Processor: Core i3 / i5

RAM: 4 GB.

Hard Disk: 500 GB.

We can use any desktop or laptop computer with the configuration listed above or a higher level.

Software Requirements:

Operating system: Windows 8 / 10

Programming Language: Python version 3.7 and above

IDE: Jupyter Notebook/Google colaboratory

Libraries: Numpy, Pandas, Matplotlib, Seaborn, sklearn, imblearn etc.

Visualization Tools: Tableau version 2021.4/Power BI

1. Research Plan

It clearly specifies our focus, goals and methods to be followed in the research from start to end along with the timeline and status. Also, the risks anticipated during the research phase and its contingency plan are also included.

1.1 Gantt Chart

The schedule or timeline for the completion of the dissertation is shown below in the form of a Gantt chart.

Figure 9.1: Gantt chart showing the schedule for completion of the dissertation

1.1 Risk Mitigation and Contingency Plan

Despite having a clear outline, the research strategy is not infallible. Risks are associated with the same. However, a backup plan has also been created for the risks that are expected.

The GPU offered by the IDE (Google Colaboratory, for instance) is K80, which is inadequate. There is a potential that the instances will frequently disconnect, halting the execution and the RAM is also insufficient when the dataset is large and numerous models are applied. In this situation, we can acquire Google Colaboratory Pro, which offers faster GPUs, longer sessions, and more RAM.

The other risk that is expected is that the activity will be delayed or the milestone won't be reached because of health problems, workplace problems, or any other reason. The research plan is created for these kinds of problems in a way that it always includes a buffer week before all deadlines and has an additional 2 weeks before submitting the final document of the dissertation.

The following risk is that of low-quality deliverables. To prevent this, it is planned to communicate with the mentor every two weeks to ensure that the research is progressing as expected, to receive their approval on the deliverables, and to make necessary modifications in response to their comments.

References

Aziz, R.M., Baluch, M.F., Patel, S. and Ganie, A.H., (2022) LGBM: a machine learning approach for Ethereum fraud detection. *International Journal of Information Technology*. [online] Available at: https://link.springer.com/10.1007/s41870-022-00864-6.

Baek, H., Oh, J., Kim, C.Y. and Lee, K., (2019) A Model for Detecting Cryptocurrency Transactions with Discernible Purpose. In: *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*. [online] IEEE, pp.713–717. Available at: https://ieeexplore.ieee.org/document/8806126/.

Farrugia, S., Ellul, J. and Azzopardi, G., (2020) Detection of illicit accounts over the Ethereum blockchain. *Expert Systems with Applications*, [online] 150, p.113318. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0957417420301433.

(‘Fletcher’, ’Emma ’), (2022) *Reports show scammers cashing in on crypto craze*. FEDERAL TRADE COMMISSION.

Ibrahim, R.F., Mohammad Elian, A. and Ababneh, M., (2021) Illicit Account Detection in the Ethereum Blockchain Using Machine Learning. In: *2021 International Conference on*

Information Technology (ICIT). [online] IEEE, pp.488–493. Available at: https://ieeexplore.ieee.org/document/9491653/.

Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W. and Qijun, C., (2017) *A Review on Consensus Algorithm of Blockchain*.

Ostapowicz, M. and Żbikowski, K., (2019) Detecting Fraudulent Accounts on Blockchain: A Supervised Approach. [online] Available at: http://arxiv.org/abs/1908.07886.

Poursafaei, F., Hamad, G.B. and Zilic, Z., (2020) Detecting Malicious Ethereum Entities via Application of Machine Learning Classification. In: *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. [online] IEEE, pp.120–127. Available at: https://ieeexplore.ieee.org/document/9223304/.

Tan, R., Tan, Q., Zhang, P. and Li, Z., (2021) Graph Neural Network for Ethereum Fraud Detection. In: *2021 IEEE International Conference on Big Knowledge (ICBK)*. [online] IEEE, pp.78–85. Available at: https://ieeexplore.ieee.org/document/9667674/.

Yuan, Q., Huang, B., Zhang, J., Wu, J., Zhang, H. and Zhang, X., (2020) Detecting Phishing Scams on Ethereum Based on Transaction Records. In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. [online] IEEE, pp.1–5. Available at: https://ieeexplore.ieee.org/document/9180815/.

Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H., (2017) An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: *2017 IEEE International Congress on Big Data (BigData Congress)*. [online] IEEE, pp.557–564. Available at: http://ieeexplore.ieee.org/document/8029379/.