National College of Ireland

Exploring the Application of Neural Network with Back Propagation in Detecting Fraudulent Transactions within the Ethereum Network

MSc Research Project
MSc Fintech

Alexandra Oluwaseun Olaitan Scott
Student ID: 21211809

School of Computing
National College of Ireland

Supervisor: Theo Mendonca

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	…ALEXANDRA OLUWASEUN OLAITAN SCOTT……
Student ID:	…21211809…….…….
Programme	…MSc FINTECH…… **Year:** 2022/ 2023.
Module:	…RESEARCH PROJECT……..……
Supervisor:	…THEO MENDONCA……..……
Submission Due Date:	…14 AUGUST 2023……..……
Project Title:	…EXPLORING THE APPLICATION OF NEURAL NETWORK WITH BACK PROPAGATION IN DETECTING FRAUDULENT TRANSACTIONS WITHIN THE ETHEREUM NETWORK ……..……
Word Count:	……6697……**Page Count**……26……..……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……ALEXANDRA SCOTT……

Date: ……14 AUGUST 2023……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

TABLE OF CONTENTS

ABSTRACT

Blockchain technology, exemplified by the Ethereum network has witnessed widespread adoption in its use. Relying on its robust security features of decentralisation, immutability and cryptography, individuals and establishments have over time gravitated towards cryptocurrencies for financial transactions. However, as with many financial systems, the cryptocurrency domain is not immune to fraudulent activities. With reports of illicit transactions peaking to around USD 14 million in 2021, several researchers have developed robust algorithms all attempting to detect and consequently, prevent the occurrence of these fraudulent transactions.

This study seeks to add to research in this domain by utilising a machine learning algorithm – Neural Network with back propagation in detecting these fraudulent transactions within the Ethereum framework. By leveraging historical Ethereum data from the Kaggle repository which contained known valid and illicit transactions, three supervised machine learning algorithms – Support Vector Machine, K-Nearest Neighbour and XGBoost were built as benchmark models for comparison. This paper goes on to build the neural network with back propagation algorithm and evaluates its performance using three performance metrics – recall, accuracy and specificity.

Overall, this study finds that the three benchmark models had a high performance with regards sensitivity / recall. The proposed model saw a high sensitivity of 0.989 but with mixed results based on other metrics.

Keywords - Ethereum, cryptocurrency, fraud detection, Neural Network, Back Propagation, Support Vector Machine, K-Nearest Neighbour, XGBoost.

INTRODUCTION

1.1. Background of the study

Blockchains are a form of distributed ledger where transactions are recorded into an immutable chain and are verified cryptographically. By using cryptography, its users are able to communicate through encoded messages and every user possesses a copy of the ledger which ensures transparency across board. The blocks contained within the blockchain framework are bound together by a hash. Additionally, once a transaction has been added, it cannot be altered resulting in a tamper proof ledger (Miah et al, 2019). The distributed ledger distinguishes itself from other ledgers as it implies there is no central authority such as governments or any establishments. This eliminates the risk of a single point of failure.

Cryptocurrencies are a form of digital asset which are created and secured using cryptographic techniques (Judmayer et al 2017). Some cryptocurrencies rely on blockchain technology and by doing so, are able to attain the security backing needed to withstand potential attacks on the system. One form of cryptocurrency that has utilised this robust feature of the blockchain framework is Ethereum which is the primary focus of this study.

With the aim of developing a cryptocurrency superior to Bitcoin and capable of transferring not just cryptocurrencies but other complex assets such as smart properties and smart contracts, Vitalik Buterin put forth in his 2013 white paper, the Ethereum framework (Buterin 2013). The Ethereum network is an open-source blockchain platform which utilises the Proof-of-Work consensus (Friedhelm and Luders, 2017)

Made up of several components, the Ethereum framework consists of the following (ibid):

- Ether which is the cryptocurrency of the Ethereum network

- Ethereum accounts which are broadly separated into two types – Externally Owned Accounts (EOA) and Contract Accounts. EOAs are typically controlled externally and require a private key to verify and authorise transactions. Contrarily, contract accounts are not controlled externally but on codes written within the smart contract.
- Gas: All transactions carried out on the Ethereum network require a transaction fee which is referred to as gas and this fee is expressed in Ether.

Furthermore, Ethereum uses smart contracts which are self-executing contracts where the use of blockchain technology is applied in enforcing, verifying and negotiating contracts digitally (Nzuva 2019). This implementation of smart contracts solves one of Bitcoins concerns – its limited scalability. Given the adaptability resulting from the use of smart contracts, developers are able to build general purpose smart contracts unlike in the Bitcoin network.(Aziz et al 2022).

Since its launch, the Ethereum network has seen exponential growth. It has grown to become the second largest cryptocurrency after Bitcoin and the largest public blockchain platform where the use of smart contracts is supported (Wu et al 2023). At the time of this study, Ethereum holds a significantly large market capitalization of $226.05 billion and a circulating supply of USD 120.18million (Yahoo Finance 2023).
Despite its architectural structure which boasts of great security features of immutability, decentralization and cryptography, studies have shown an increase in financial fraud being committed through the Ethereum network as its adoption increases. The network has gone on to see various scams such as Ponzi schemes, pump and dump and false Initial Coin Offerings . All of which are capable of destabilising the stability of the financial ecosystem.
In 2021, Chainanalysis reported an all-time high of illicit transactions within the Blockchain framework amounting to $14 billion - an amount that is expected to rise alongside an increase in blockchain's scalability (ibid). The pseudo anonymity associated with cryptocurrencies alongside its lack of regulation has made this medium attractive to malicious individuals as a channel through which financial fraud can be committed.
Within the context of this paper, fraud would be defined as any misrepresentation committed with the intent to obtain financial advantage.
Given this issue of fraud highlighted in preceding text, vast methods of fraud detection have been applied over time, all aimed at detecting and ultimately preventing fraudulent transactions from occurring. Some of these methods are referred to as traditional fraud detection techniques. These Traditional methods entail auditing and manually flagging abnormal transactions by a trained personnel – a process which is not only time consuming but also costly due to the human resources involved (Sanchez et al, 2023)
However, as technology has rapidly influenced the financial system, establishments have turned to computational and automated fraud detection processes thereby rendering these traditional methods redundant. One of which is the application of machine learning.
Fraud detection is mainly seen as a binary classification problem as transactions are broadly classified into two main groups - fraud and valid transactions. Some studies have gone on to classify fraud detection as an anomaly detection given the rarity of fraud in most datasets.
This study adds to research in the use of machine learning algorithms as a means of detecting fraudulent transactions within the Ethereum network. It develops three supervised machine learning algorithms as benchmark models for comparison - Support Vector Machine ("SVM"), K-Nearest Neighbour ("kNN") and XGBoost. These machine learning algorithms have been tested over time for their performance in detecting fraudulent transactions. This paper goes on to develop the Neural Network with back propagation model as a novel approach in Ethereum fraud detection and compares its performance against the aforementioned benchmark models.

1.2. Research Question and Objectives

This study aims to answer the following research question:

"How well does Neural Network with back propagation model perform in predicting fraudulent transactions within the Ethereum network?"

Given the above research question, the objectives of this study are thus:

- To assess the performance of the Neural network with Back Propagation algorithm in identifying fraudulent transactions within the Ethereum network
- To perform three other supervised machine learning – Support Vector Machine K-Nearest Neighbour and XGBoost as benchmark models for comparison.
- Compare results of the Neural Network with Back Propagation algorithm against these benchmark supervised machine learning algorithms
- Highlight ways which these results and findings can be deployed in preventing and detecting fraudulent Ethereum transactions across the network.

1.3. Structure

Following the introduction into the study and its research objectives presented in section 1, this paper progresses to a literature review in section two where past research into fraud within the cryptocurrency and blockchain domain, the use of supervised ML algorithms in fraud detection and the implementation of hybrid ML algorithms would be discussed. Section three highlights the methodology followed, outlining all techniques applied in this study. In section 4, all design specifications for the machine learning algorithms are explained. Section 5 details implementation of the proposed models which is followed by section 6 where the obtained results are evaluated. Finally, section 7 concludes the study, summarizing all key findings whilst offering recommendations for future work in this domain.

By following this structure, this paper provides an in-depth exploration into utilising machine learning algorithms in detecting fraudulent transactions within the Ethereum network.

1.4. Justification of the study

Despite the increase in research focusing on the Blockchain domain aimed at boosting security and scalability, malicious individuals have remained persistent in using the network as a channel to conduct fraudulent transactions.

Multiple research has been carried out within the area of fraud detection in the cryptocurrency domain, the focus has continuously remained on the use of supervised machine learning algorithms widely used in classification problems. This study seeks to determine if the application of the Neural Network with back propagation algorithm can yield higher performance when compared to these previously deployed algorithms.

2. LITERATURE REVIEW

2.1. Fraud within the Blockchain and Cryptocurrency domain

The use of cryptocurrency particularly within the blockchain framework is renowned for its robust security features stemming from its decentralised framework and immutability discussed in section 1. However, despite these features, it is important to note that cryptocurrencies are still being used with malicious intent such as to evade tax and commit fraud. Various researchers have analysed the different ways fraud is committed using cryptocurrencies from diverse standpoints and these concerns have been grouped into the

following distinct categories by (Linh et al, 2019) which include – Security concerns related to the blockchain architecture, fraudulent activities such as Ponzi schemes or scams, and studies exploring the intersection of the aforementioned concerns .
Major fraud related incidents such as the Distributed Autonomous Organization (DAO) attack have raised scrutiny regarding the prevalence of fraud within the Ethereum network. The DAO – set up as an self-directed venture capital fund for digital assets launched on the Ethereum network in 2016. Gaining USD 150 million in cryptocurrency investments, it was soon infiltrated by an anonymous hacker who utilised a flaw in the smart contract resulting in the loss of USD 50 million Ethers from the total invested sum (Meher et al, 2019).
In its 2022 cryptocurrency crime report, Chainanalysis – a blockchain analysis firm providing research insights, reported that the number of illicit transactions within the blockchain network has continuously peaked over time and in 2021, reached an all-time high of $14 billion (Chainanalysis 2021).
A common and well-known scam within the Ethereum network highlighted by (Linh et al) is the Ponzi Scheme scam.
In their study, (Vasek and Moore, 2015) explored this category of scams by focusing on fraud associated with the Bitcoin Ponzi schemes. By scouring 11,424 threads across three subforums on the bitcointalk.org platform, the authors were able to obtain information relevant to these scams. The discussions on various scams within these forums included gambling (games and rounds), investment games (Ponzi schemes) and scam accusations. The objective was to extract information which contained the supply and demand for scams within the cryptocurrency network. The research was able to identify 1780 scams from 2625 posts. Interestingly the research went further to establish that the average crypto scam lasted about a week, however, by the scammer posting more frequently, the scam was enlivened, lasting approximately three weeks..
A distinct attribute of financial scams which utilise cryptocurrencies when compared to traditional financial scams is the public availability of these fraudulent transactions which has proven useful for research purposes. (Jung et al, 2019)

2.2. Detecting fraudulent transactions with supervised machine learning algorithms

Supervised Machine Learning ("ML") approaches in fraud detection entail building classification based models that are able to predict fraudulent or valid transactions given a dataset. The use of supervised algorithms in this domain has gained popularity due to its high performance in solving classification problems particularly the Support Vector Machine and Neural Networks algorithms (Osisanwo et al 2017).
Supervised ML algorithms aim to derive a map between the input and output and move on to subsequently predict outputs given new inputs. (Liu and Wu 2012). The output here is referred to as the supervision or the label of the input data. (ibid). Figure 1 is a visual representation of the supervised learning process where 'x' and 'y' represent the input and output respectively, making up the training sample (x,y) and 'i' refers to the training index.
Given its reliance on labels, supervised machine learning works well with only labelled dataset. This can however be perceived as a con of the supervised ML algorithm given the high costs associated with labelling large volumes of data (ibid)

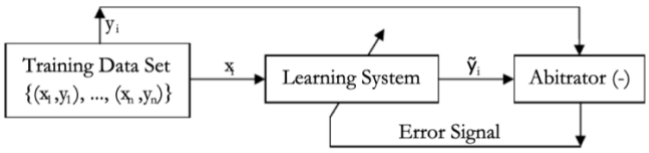


Figure 1: The supervised machine learning process (Liu and Wu 2012)

However, despite its high performance in various studies, two main issues are often attributed to the implementation of supervised machine learning methods – the overfitting and generalization error

The overfitting error, also known as the parametric variance, occurs when the model is unable to generalise accurately from the inputted data into unseen data. This results in poor fitting on the testing data despite high performance on the training set. (Ying, 2019)

Taking a supervised ML approach and implementing the XGBoost model, (Farrugia et al, 2020), created a dataset by combining 2502 normal accounts and 2179 accounts flagged by the Ethereum community for illegal activity. Based on the transaction history, the research sought to detect these illegal accounts. These accounts were flagged for a number of reasons which included phishing, website mirroring, Ponzi schemes and deceptive initial coin offerings amongst others. The study implemented grid search cross-validation tuning consisting of 10 folds to determine the optimal parameter values and then implemented the XGBoost model which achieved an accuracy of 0.963. By implementing the cross validation prior to modelling, the author was able to identify the optimal values for the three parameters of interest – max tree depth, learning rate and number of trees and in turn, were able to build a more accurate model.

Utilising synthetic data in their recent study, (Afriyie et al 2023) tested three different supervised machine learning algorithms – logistic regression, decision trees and random forest in terms of their capabilities in accurately predicting and detecting fraudulent credit card transactions. With data ranging over a 12 month span – January to December 2020, this simulated data set comprising over 500,000 observations was used. Made up of various nodes, leaf nodes and edges which graphically represent information in patterns similar to a tree, the decision tree model yielded an accuracy, F1-score, recall, precision and specificity of 0.92, 0.09, 0.93, 0.05 and 0.92 respectively whilst the logistic regression model followed in a similar pattern with 0.92, 0.08, 0.76, 0.04 and 0.92. Overall, the random forest model outweighed its counterparts in this study with an accuracy of 0.96, and an F1 score, recall, precision and specificity of 0.17, 0.97, 0.09 and 0.96 respectively. Similar to other fraud detection studies, the data used was highly imbalanced and in order to reduce the potential of overfitting, the dataset was balanced using an under sampling technique.

(Dubey, 2020), implemented the Artificial Neural Network (ANN) with Back propagation in detecting credit card fraud by utilising a credit card customer dataset which contained 30 attributes of information related to the customer. The results of the ANN with Back Propagation model performed well with an accuracy of 99.92%, precision of 99.96% and an F1-score of 99.96% as well. This author additionally highlights that implementing this algorithm is particularly relevant in the financial industry due to its capability of real-time fraud detection

2.3. Exploring hybrid and alternative machine learning approaches in fraud detection

Hybrid machine learning techniques entail using ensemble learning. Ensemble learning refers to algorithms that classify new data points by taking the weighted vote of the predictions of a set of classifiers. They are known for their performance in generating highly accurate classifiers through the combination of less accurate classifiers (Dietterich, 2000). Various methods have been identified for constructing ensembles which include Bayesian voting, manipulating the training data to generate multiple hypotheses and manipulating the input features or output targets (ibid.)

Combining both supervised and unsupervised machine learning algorithms in detecting credit card fraud, (Fabrizio et al 2019) highlighted a flaw in using only supervised machine learning – it's inability to adapt to changes in behavioural patterns. By introducing unsupervised machine learning algorithms, these anomalies can be detected. Using three approaches – global, local and cluster, the results of the study however yielded unpromising results in terms of the

global and local approach. The cluster approach showed more promising results in terms of the Area Under Precision Recall Curve (AUC-PR) performance metric.

Similarly, AdaBoost and LGBM were combined as a hybrid approach in detecting fraud by (Malik et al 2022). As baseline models, the authors developed several ML algorithms – Linear regression, SVM, Naives Bayes, XGBoost, Random Forest, AdaBoost, extreme Gradient Boosting and Decision Trees. When tested as standalone models, these baseline algorithms yielded relatively similar results with an Area Under the Receiver Operating Characteristic (AUROC) value between 0.66 and 0.71 with the exception of Naives Bayes with a 0.56 score. Developing several hybrid models at the second stage of the study, the authors saw an increase in performance through the combination of Adaboost + LGBM model yielding an AUROC score of 0.82.

A final hybrid algorithm being discussed was proposed by (Tekkali and Natarajan 2022) as a novel approach in detecting digital transactional fraud. This paper combines deep reinforcement learning – an algorithm where prediction of fraudulent transaction is done by an agent through activation of the reward function with rough set theory. The proposed hybrid model showed an accuracy of 96.09% making it a suitable option for solving binary classification problems.

In the face of unlabelled datasets, unsupervised machine learning has become the de facto approach. A dataset is said to unlabelled when there is no indicator field which explains why a data occurs in a certain manner. Unlabelled data is typically naturally occurring and can be seen in data such as audio recordings, pictures or tweets.

Due to inconclusive results in the field of unsupervised ML in fraud detection, research is limited.

Despite this, in the presence of label scarcity, (Lorenz et al 2020) sought to detect money laundering - a form of financial fraud within the Bitcoin network. Highlighting that implementing this fraud detection algorithm with no labels might seem impossible, the authors combined unsupervised learning with active learning. Active learning here mimics a real world scenario where there is limited human resources for data labelling. The findings from the study revealed that using unsupervised ML as a standalone model yielded poor results. However, by implementing Active Learning, the results were similar to supervised ML algorithms.

3. RESEARCH METHODOLOGY

All research methods and techniques applied in this study are detailed in this section. Following the Knowledge Discovery in Database ("KDD") methodology, this section of the study details the process of data selection, the research procedure and all techniques applied towards obtaining the results. It follows a process similar to figure 2.

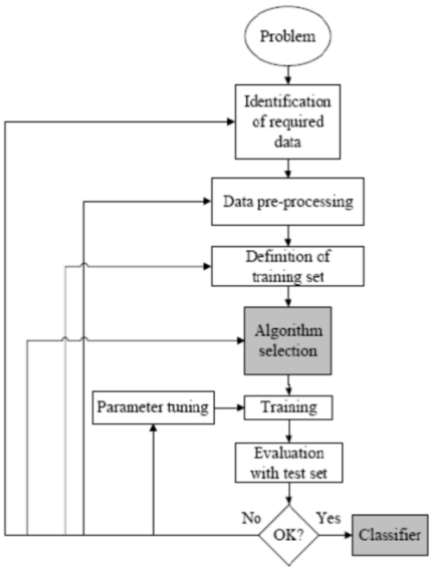


Figure 2: Graphical representation of research methodology

3.1. Data Selection

Utilising secondary data, the Ethereum dataset was obtained from Kaggle (Kaggle 2023). Kaggle, founded in 2010 and later acquired by Google provides a digital platform which facilitates collaboration between machine learning experts and data scientists by offering a vast repository of meaningful datasets relevant to their work as well as a platform where datasets can be published. The platform supports multiple data formats, including Comma Separated List (or "CSV"), JSON files, SQLite amongst other data formats. As at the time if this paper, Kaggle hosted 242,689 datasets, making it a valuable resource for identifying relevant data for research purposes.

From this extensive database, the 'Ethereum Fraud Detection dataset' was chosen which consists of 9461 observations of known valid and fraudulent transactions carried out over the Ethereum network. The classification of fraudulent transactions is seen under the 'FLAG' variable. The dataset contained 51 variables associated with each transaction which are described in the appendix of this document – Appendix 1
The dataset is imbalanced with the larger portion of the data skewed towards valid transactions visually represented in figure 6.

Further analysing the dataset, the dataset is split into two distinct categories. The first half of the dataset containing variables related to the transaction whilst the second half contains variables associated with Ethereum Request for Comment – ERC20.

3.2. Data Pre-processing and transformation

The data preparation stage is crucial in building robust machine learning algorithms. It involves critical tasks such as Exploratory Data Analysis (or "EDA"), data cleaning, variable transformation and all preparations carried out which are essential in building effective models. Upon importation of the dataset, all pre-processing steps were carried out using R Studio. Employing Exploratory Data Analysis (or "EDA") techniques are useful in gaining comprehensive insights into the dataset. Through the creation of summary statistics and plots, trends and patterns within the dataset are identified thereby providing a solid foundation for subsequent analysis.

Furthermore, as part of the data cleaning process, addressing all missing values and outliers within the dataset is essential in this stage. Handling these aspects of the dataset is crucial as most machine learning algorithms are not equipped to handle them directly.
Several techniques have been proposed as solutions to handling missing attributes (Tlamelo et al 2021) which include:
- Deleting the missing values which is often considered the simplest approach.
- Imputation where missing values are replaced with predicted values. Using simple imputation, missing values are replaced using quantitative attributes of non-missing values by calculating the mean, median or mode. Another imputation technique is regression imputation which relies on creating a regression model based on the presumption that the data is missing at random (Tlamelo et al, 2021)
- K-Nearest Neighbour algorithm is another technique that uses a distance measure to evaluate the similarity between instances in a dataset. Once these closest neighbours are identified, they are imputed, thereby addressing any missing values present (ibid).

In addition to handling any missing values, the pre-processing stage uses different techniques to address the class imbalance as this step is crucial in building bias-free models. Several techniques have been proposed to address this class imbalance problem (Kotsiantis et al, 2006) which include:
- Resampling techniques which aim to balance class distribution through reducing or increasing the number of instances contained within the minority of majority class. These techniques include: random oversampling or under sampling, direct oversampling or under sampling, and a combination of these techniques (Kotsiantis et al, 2006)
- Algorithmic level techniques such as modifying the estimated probability of a tree leaf , modifying the various class costs in order to counter any class imbalance or a combination of these techniques (ibid).

4. DESIGN SPECIFICATION

This section discusses the design specification utilised in this study. This entails all four machine learning algorithms which were developed for testing purposes.

This research utilises four different machine learning algorithms. Similar to work carried out by (Malik et al, 2022) in section 2.3, this study creates an initial benchmark consisting of three supervised algorithms – Support Vector Machine (or "SVM"), K-Nearest Neighbours (or "KNN") and XG Boost. These models have previously been evaluated by other authors for their effectiveness in detecting fraudulent transactions.
As a novel approach, the use of Neural Network with back propagation is introduced in this study. By comparing Neural network with BP against these benchmark algorithms, this paper

seeks to examine its performance in predicting fraudulent transactions within the Ethereum network.

4.1. Support Vector Machine ("SVM")

Initially proposed by Vladimir Vapnik , SVM was developed from the theory of structural risk minimization and provides a supervised learning algorithm typically used in regression and classification problems (Dheepa and Dhanapal, 2012). SVM classifier is often well suited for binary classification problems where it predicts patterns into two categories.

As a general intuition, the SVM model works by finding the optimal hyperplane where the data points associated with different classes are separated in a high dimensional space

The input vector is mapped into a higher dimensional space where the maximal separating hyperplane is then constructed. On either side of this maximal hyperplane which separates the data, two parallel hyperplanes are further constructed. (Srivastava and Bhambhu 2010).

An assumption is made regarding the SVM model which indicates that a larger distance or margin between the two parallel hyperplanes constructed, the better the classifier's generalisation error (ibid).

Therefore, the high success attributed to using SVM in classification problems is as a result of its ability in tuning multiple parameters to minimise the generalization error highlighted in section 4.1.1.

In instances of a binary classification problem, SVM's decision function is described in eqn.(1). where b is a constant and x is the input vector containing weights (w)

$$f(x) = sgn(x.w) + b \tag{1}$$

4.2. K-Nearest Neighbours (kNN)

The second model built in this study – kNN provides a clustering algorithm widely adopted in the field of data science in its application towards categorizing datasets. It is often seen as the a simplified method of solving the classification problem (Zhang, 2016).

kNN is commonly used in instances where information pertaining to the distribution of the data is limited. It performs by identifying patterns in a dataset and proceeds to classify these objects based on the Euclidean distance between the training and test samples.

The kNN algorithm works by classifying unlabelled observations within a dataset through assigning said observation to its most similar example.

There are two important concepts in the kNN algorithm – (i) The K Parameter and (ii)The Euclidean distance. Both of which are linked to the performance of the kNN classifier.

The K parameters determines the number of neighbours which would selected in building the kNN algorithm. By choosing a large k, the variance attributed to random error is reduced. However, this poses a risk of ignoring the small but important patterns in the data.

For the Euclidean distance, Taking an example n = the total number of input samples (i = 1, 2…, n), x_i = the input sample containing p features (x_{i1} , x_{i2},…, x_{ip}) and p is the total number of features (j = 1, 2, …,p)

The Euclidean distance is calculated using the following equation (2): (Peterson, 2009)

$$d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \cdots + (x_{ip} - x_{lp})^2}. \tag{2}$$

Another method utilised in calculating this distance is the Manhattan Distance (Zhang, 2016).

4.3. XG Boost Model

The use of gradient tree boosting models has become a widely adopted machine learning method. One of which is the XGBoost model described as a scalable end-to-end tree boosting model (Chen and Guestrin, 2016). Studies over the years have shown that tree boosting algorithms have particularly performed well in standard classification tasks (ibid.)

Scaling to billions of examples in a distributed setting, XGBoost is said to run approximately ten times faster than other algorithms. This scalability XGBoost presents has made it appealing to ML experts.

XGBoost works by generating multiple sequential trees. Each of the successive trees generated aims at reducing the error of the previous tree and in turn updates the residual error (Ashfaq et al, 2022).

XGBoost is known to predict incoming transactions within the blockchain framework by connecting to the blockchain smart contract.

4.4. Neural Network with Back Propagation

Neural Networks are known for their intricate structure composed of many interconnecting layers which mirrors the human brain's neural connections.

Neural Networks are composed of several elements which are interconnected, referred to as neurons, similar to the ones found in human brain. These neurons work in parallel to solve specific problems (Ahamed and Akthar, 2016) . Despite their interconnectedness, these neurons do not touch each other and are separated by a small gap called a Synapse (ibid).

The architectural structure of the Neural Network is composed of various layers, each, with a specific role. These layers include the Input, Output and one or multiple Hidden Layers illustrated graphically in figure 3. Each of the nodes present in the input layer are all connected to a node in the hidden layer and each node in the hidden layer, connected to the output.

- Input layer: Present to receive data from external sources
- Hidden Layer performs all computations based on the function provided
- Output layer returns the output based on all inputs fed into the algorithm.

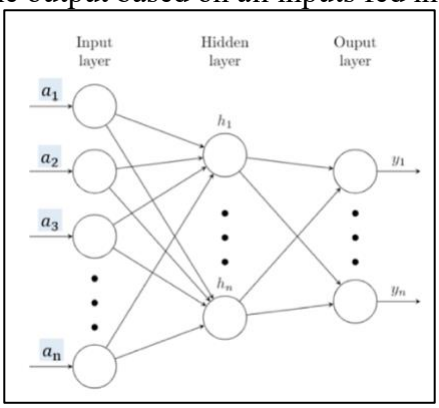


Figure 3: Simple Neural Network

Another key component of the Neural Network is the Activation function. The activation function consist of either linear, sigmoid or threshold functions necessary to enable the hidden layer of the neural network introduce nonlinearity,

Back Propagation
The Back Propagation (BP) algorithm, unlike typical neural network algorithms, works by minimising the disparity between the predicted and actual outputs by adjusting the connection weights (Buscema, 1998) and can be divided into the following

1. Feed Forward back propagation
2. Back propagation to the hidden layer
3. Weight updates
4. Back propagation to the output layer

Similar to work carried out by Dubey, 2020 on credit card transactions discussed in section 2.3, this research employs this final algorithm – Neural Network with Back Propagation ("BP") as a novel approach in Ethereum fraud detection.

5. IMPLEMENTATION

The section details the implementation of the design specifications outlined in preceding texts using the Ethereum Fraud detection Data set.

Experimental Setup
Further detailed in the corresponding configuration manual of this study, the following experimental setup was utilised to build each of the highlighted algorithms.

HARDWARE		SOFTWARE	
System	MacBook Pro (Retina, 13-inch, Early 2015)	Operating System	macOS Monterey Version 12.6.5
Processor	2.7 GHz Dual-Core Intel Core i5	Programming Environment	RStudio Version 2023.06.1+524

Table 1:Experimental setup

5.1. Data Pre-processing and Data Transformation

5.1.1. Handling missing values

The first step in the data pre-processing stage was in identifying any missing values present. The result showed 4% of the dataset returned null values indicated in figure 4. The missing values were handled using simple imputation methods which (Tlamelo et al 2021) proposed in section 3.2. The median was computed for the numerical variables and then inputted into the Ethereum dataset to replace all missing observations illustrated in figure 5.

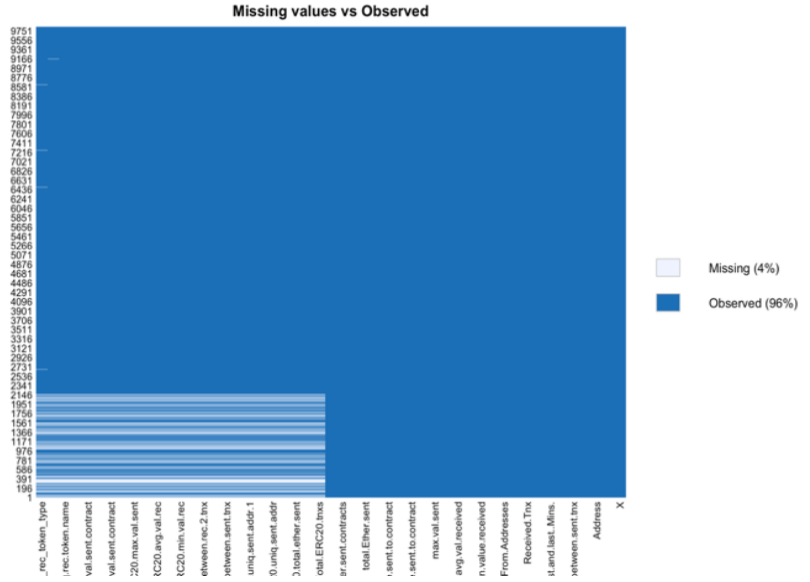


Figure 4: Dataset with missing values

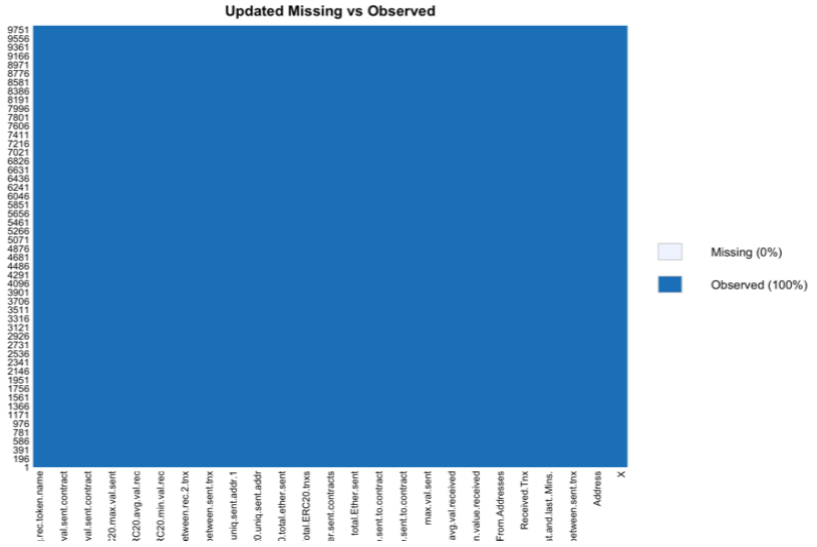


Figure 5: Data set following missing value computation

5.1.2. Data visualisation

Data visualisation techniques were applied to gain insight into trends and patterns contained within the dataset. Figure 6 presents a bar chart showing the number of fraudulent transactions in comparison to the valid / legit transactions. From this chart, it is observed that the transactions in the dataset were skewed towards being valid which indicated the need for class balancing whilst figure 7 provides insight into the correlation of the various variables.

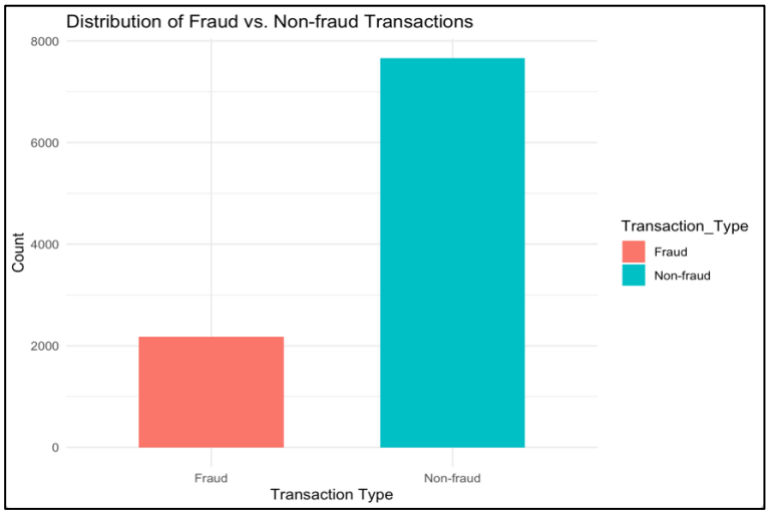


Figure 6: Distribution of fraudulent and valid transactions

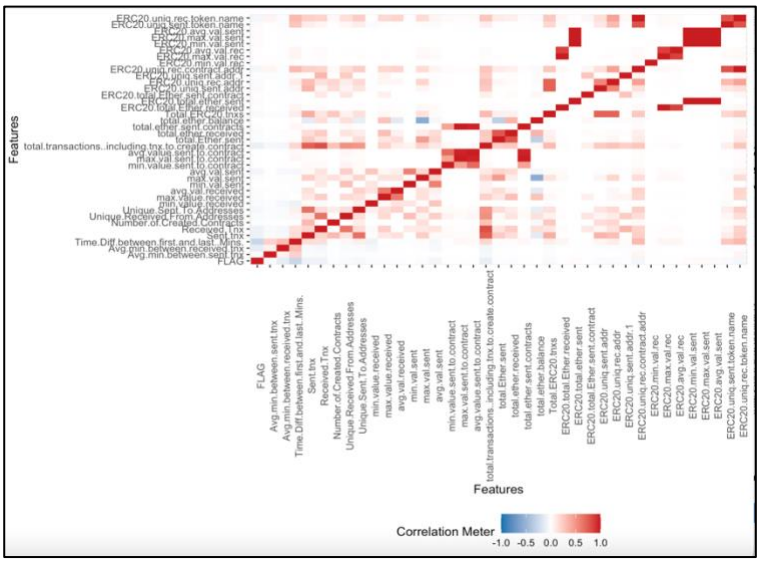


Figure 7: Heatmap of Correlation matrix

5.1.3. Removing columns with zero variance

All variables that have a variance of 0 were selected and removed from the dataset prior to training. These variables have no impact on the dataset as they hold the same value for all samples . By removing these variables, the risk of overfitting is minimised.

```
[1] "ERC20.avg.time.between.sent.tnx"      "ERC20.avg.time.between.rec.tnx"
[3] "ERC20.avg.time.between.rec.2.tnx"     "ERC20.avg.time.between.contract.tnx"
[5] "ERC20.min.val.sent.contract"          "ERC20.max.val.sent.contract"
[7] "ERC20.avg.val.sent.contract"
```

Figure 8: Cross-section of variables with zero variance

5.1.4. Splitting the Dataset

The Ethereum fraud detection dataset was split into a training and testing sample using an 80/20 split ratio respectively. The 20% allocated to the Testing data was not manipulated throughout the course of the pre-processing and transformation stage. This was done to prevent the lookahead bias which occurs when information from the future – in this instance, the testing data is used to make predictions during training

5.1.5. Handling the class imbalance

Using the Synthetic Minority Oversampling technique (SMOTE) discussed in section 3.2. of this paper, the class imbalance within the training dataset was handled. Figure 9 illustrates the class imbalance in the raw dataset. Once the dataset was balanced using the oversampling technique, its output is illustrated in figure 9

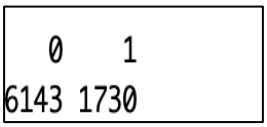


Figure 9: FLAG distribution before balancing

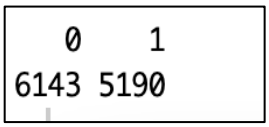


Figure 10: FLAG distribution after balancing

5.1.6. Hyperparameter tuning

During the modelling phase, another important step taken to ensure optimal results was hyperparameter tuning.

A hyperparameter in machine learning refers to a parameter derived from the training sample made up of a value that is set before the initiating the training process. The model's parameters are derived using algorithms from the data and refer to its weights and coefficients (Elgeldawi et al, 2021).

These parameters, therefore, have to be initialised before the training phase of the model building commences. A key step in maximising the model's performance and yielding optimal results on the validation set lies in fine tuning the model's hyperparameters (ibid).

To fine tune these hyperparameters, this study performed the K- Fold Cross-Validation ("CV"). K-Fold Cross Validation is implemented by randomly dividing a set of observations into K number of folds. The method fits on k-1 folds as the first fold is used as the validation set. The Mean Square Error is then computed on the isolated fold and repeated k number of times holding out different folds each time. The K-fold CV is computed as an average of the test errors (Gareth et al, 2023).

Grid search hyper parameter tuning:

Despite far-reaching research into global optimization and hyper-parameter optimization, grid search has continuously prevailed as being described as 'state of the art' in this domain as it concerns machine learning (Bergstra and Bengio 2012). This is as a result of the ease in its execution and parallelization as well as its performance in low-dimensional spaces (Belete and Huchaiah 2021)

Following manual definition of the algorithms hyperparameter space, grid search works by performing an exhaustive search or by applying brute force method in testing all possible hyperparameter combinations which are fed into the grid configuration. (ibid).

This study uses the K- fold CV together with grid search in obtaining the optimal model parameter values.

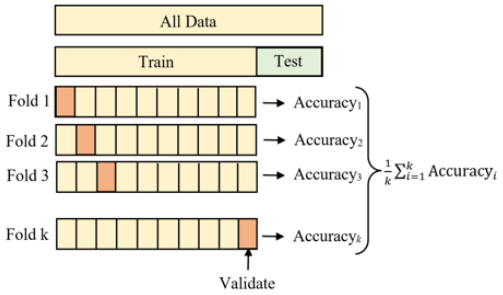


Figure 11: K – Fold Cross validation

6. RESULTS AND EVALUATION

There are a number of metrics which are able to efficiently measure the performance of binary classification algorithms, three of which are being applied in this study.
The models are evaluated on the following performance metrics: recall / sensitivity, accuracy and specificity.
All the aforementioned performance metrics are based on the following factors:
True Positive ("TP") : refers to all positive classes that were predicted positive by the model.
True Negative ("TN"): all positive class predicted as negative by the model.
False Positive ("FP") : all negative classes predicted positive by the model.
False Negative ("FN") : all positive classes predicted negative by the model.
In this study, valid transactions are the negative class whilst fraudulent transactions are the positive class

Recall, also commonly referred to as the model's sensitivity is defined as the number of true positive – in this case, fraudulent transactions predicted by the various algorithms in comparison to the total number of fraudulent transactions (Powers, 2011). It is interesting to note that in typical fraud prediction problems, the sensitivity of the model has over time shown to be the most relevant performance metric.

$$Sensitivity / \quad Recall = \frac{TP}{(TP+FN)} \tag{3}$$

The next performance metric is accuracy. The accuracy of a model can be defined as the number of all correct predictions divided by the total number of predictions made (Powers, 2011).
In detecting fraud within the Ethereum network, accuracy can be seen as misleading due to the high class imbalance where more transactions skewed towards being valid.

$$Accuracy = \frac{(TP + TN}{(TP + TN + FP + FN)} \tag{4}$$

Specificity of a model is commonly used as diagnostic performance measure of a test. (Zhang, 2016) It defined as a measure of negative values predicted that are truly negative – the true negatives

$$Specificity = \quad \frac{TN}{TN+FP} \tag{5}$$

Support Vector Machine

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1477   86
         1   42  363

               Accuracy : 0.935
                 95% CI : (0.9231, 0.9455)
    No Information Rate : 0.7718
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8087

 Mcnemar's Test P-Value : 0.0001443

            Sensitivity : 0.9724
            Specificity : 0.8085
         Pos Pred Value : 0.9450
         Neg Pred Value : 0.8963
             Prevalence : 0.7718
         Detection Rate : 0.7505
   Detection Prevalence : 0.7942
      Balanced Accuracy : 0.8904
```

Figure 12: Confusion Matrix of the Support Vector Machine Algorithm

The Support Vector Machine demonstrated good performance in detecting fraud within the Ethereum network. With an accuracy of 0.935, SVM was able to predict the correct class in 93.5% of the instances. In terms of its sensitivity/ recall, SVM also had a high recall of 0.972 or 97.2% SVM was also useful in predicting the valid class with a specificity of 0.8085 or 80.8%.
Overall SVM had a strong performance. In particular, its high sensitivity rate indicates it is promising outlook in detecting fraudulent transactions.

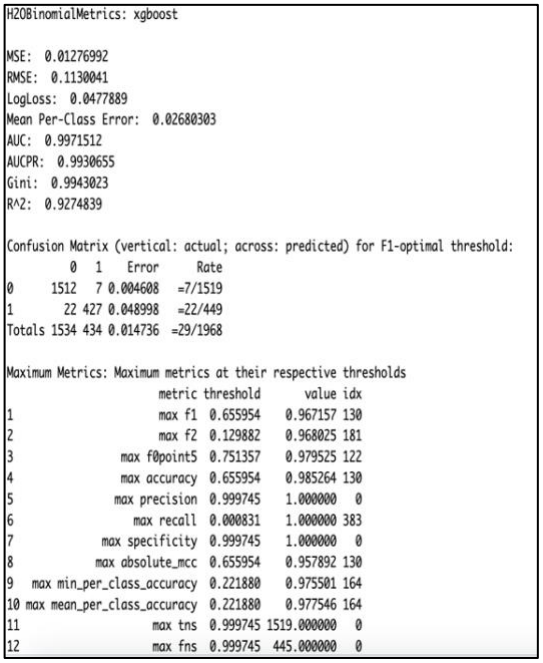
K-NEAREST NEIGHBOUR

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1429  157
         1   90  292

               Accuracy : 0.8745
                 95% CI : (0.859, 0.8888)
    No Information Rate : 0.7718
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6239

 Mcnemar's Test P-Value : 2.675e-05

            Sensitivity : 0.9408
            Specificity : 0.6503
         Pos Pred Value : 0.9010
         Neg Pred Value : 0.7644
             Prevalence : 0.7718
         Detection Rate : 0.7261
   Detection Prevalence : 0.8059
      Balanced Accuracy : 0.7955
```

Figure 13: Confusion Matrix of the kNN Algorithm

Although slightly below the SVM model in its performance, kNN also performed well in its predictions. With an accuracy of 0.874 and a sensitivity of 0.9408. This shows that the kNN model is able to predict the right class correctly in 87.45% of the instances and predict 94.08% of the fraudulent transactions accurately.
A key distinguishing attribute between kNN and SVM is its specificity as the kNN model reported a significantly lower specificity of 0.6503 which indicates the model does not perform very well in predicting valid transactions in a dataset.
In addition, previously discussed literature attributed a better model performance with a larger K value. In this study, upon hyperparameter tuning, the kNN model used a K value = 5 and perhaps utilising a higher K might result in better model performance.

XGBOOST



H2OBinomialMetrics: xgboost

MSE: 0.01276992
RMSE: 0.1130041
LogLoss: 0.0477889
Mean Per-Class Error: 0.02680303
AUC: 0.9971512
AUCPR: 0.9930655
Gini: 0.9943023
R^2: 0.9274839

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:

	0	1	Error	Rate
0	1512	7	0.004608	=7/1519
1	22	427	0.048998	=22/449
Totals	1534	434	0.014736	=29/1968

Maximum Metrics: Maximum metrics at their respective thresholds

	metric	threshold	value	idx
1	max f1	0.655954	0.967157	130
2	max f2	0.129882	0.968025	181
3	max f0point5	0.751357	0.979525	122
4	max accuracy	0.655954	0.985264	130
5	max precision	0.999745	1.000000	0
6	max recall	0.000831	1.000000	383
7	max specificity	0.999745	1.000000	0
8	max absolute_mcc	0.655954	0.957892	130
9	max min_per_class_accuracy	0.221880	0.975501	164
10	max mean_per_class_accuracy	0.221880	0.977546	164
11	max tns	0.999745	1519.000000	0
12	max fns	0.999745	445.000000	0

Figure 14: Confusion Matrix of the XGBoost Algorithm

Unlike the kNN and SVM models, in order to get the sensitivity and recall of the XGBoost algorithm, manual computation had to be carried out by utilising the information contained in the confusion matrix. Using formulas contained in equations (3) and (5) above, the models recall / sensitivity was calculated thus

For the model's sensitivity, we use the following equation = $\frac{TP}{(TP+FN)}$ = $\frac{1512}{(1512+22)}$

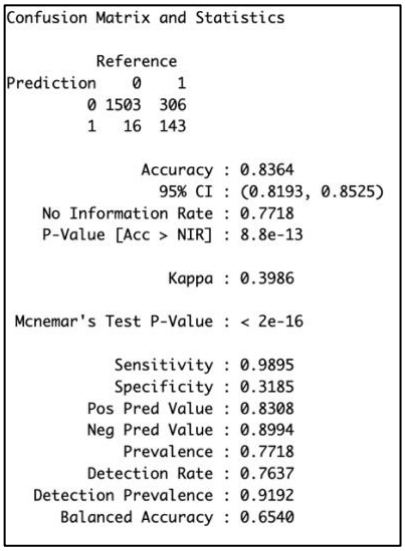
Sensitivity = 0.9857

For the model's specificity, we use the following equation $\frac{TN}{(TN+FP)}$ = $\frac{427}{(427+7)}$

Specificity = 0.9839

The accuracy was computed using RStudio and XGBoost had a high accuracy of 0.973 indicating that the model is able to predict the correct class in 97.3% of the instances.

Neural network with back propagation



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1503  306
         1   16  143

               Accuracy : 0.8364
                 95% CI : (0.8193, 0.8525)
    No Information Rate : 0.7718
    P-Value [Acc > NIR] : 8.8e-13

                  Kappa : 0.3986

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.9895
            Specificity : 0.3185
         Pos Pred Value : 0.8308
         Neg Pred Value : 0.8994
             Prevalence : 0.7718
         Detection Rate : 0.7637
   Detection Prevalence : 0.9192
      Balanced Accuracy : 0.6540
```

Figure 15: Confusion Matrix of the Neural Network with Back Propagation Algorithm

The Neural Network with Back Propagation model showed mixed results. With an accuracy of 0.8364, the model correctly predicted the class in 83.6% of the instances.
The true positive rate or sensitivity was impressive at 0.9895 highlighting the models ability to effectively detect fraudulent transactions within the Ethereum network. However, the model performed poorly in detecting valid transactions with a specificity score of 0.3185.

SUMMARY OF RESULTS				
	KNN	**SVM**	**XGBoost**	**Neural Network with Back Propagation**
Recall	0.940	0.972	0.985	0.989
Accuracy	0.874	0.935	0.973	0.836
Specificity	0.650	0.808	0.983	0.318

Table 2: Summary of model performance

Overall, the three benchmark algorithms performed well, with XGBoost yielding the highest sensitivity of 0.985 or 98.5%. This is consistent with past literature discussed above where these three benchmark algorithms continuously performed well in classification problems
When compared with the Neural network with back propagation algorithm, they were slightly out performed as the Neural networks algorithm had a sensitivity / recall value of 0.989. This performance metric is particularly useful as it indicates the model's ability to detect fraudulent transactions – the true positives.
In terms of accuracy and specificity, the Neural Network with back propagation model however yielded the lowest performance of 0.836 and 0.318 respectively.
However, this low specificity does not have a significant impact on the model's performance within the context of this study. This is as a result of specificity concerning the detection of the negative class – in this study, the valid transactions whereas this study was aimed at the detection of fraudulent transactions in the Ethereum network.

7. CONCLUSION AND FUTURE WORK

The study aimed at enhancing the detection of fraudulent transactions within the Ethereum network using machine learning algorithms. Three benchmark supervised machine learning algorithms were developed followed by a novel approach - the Neural network with back propagation. Testing its performance against the benchmark models, it outperformed them in its sensitivity or ability to detect the true positive class. However, it showed less promising results based on the accuracy and specificity metrics.

This study was however limited in the size of the dataset used. The dataset used in this study contained 9461 observations. This was due to a limitation in available open source datasets pertaining to fraud within the Ethereum network

Future work in this domain can utilise ensemble techniques implemented by authors in section 2.3 of this paper. By doing so they can combine the Neural Network with Back propagation model with other supervised or unsupervised machine learning algorithms. The AdaBoost algorithm can prove useful in this instance as it showed great effect on performance. In addition, future research can see how using a higher K value would affect the kNN's model performance.

REFERENCES

Ahamed, K. and Akthar, S., (2016), 'A study on neural network architectures', *Computer Engineering and Intelligent Systems,* vol. 7(9), pp. 1 – 7.

Afriye, J., Tawiah, K., Pels, W., Addai-Henne, S., Dwamena, H., Owiredu, E., Ayeh, S. and Eshun. J., (2023), 'A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions', *Decision Analytics Journal,* vol. 6, pp. 1 – 12.

Ashfaq, T., Khalid, R., Yahaya, A., Aslam, S., Azar, A., Alsafari, S. and Hameed, I. (2022), 'A machine learning and blockchain based efficient fraud detection mechanism', *Sensors,* pp. 1 – 20.

Ayele, W., (2020), 'Adapting CRISP-DM for Idea Mining: A data Mining Process for Generating Ideas Using a Textual Dataset', *International Journal of Advanced Computer Science and Applications,* vol. 11(6), pp. 20 – 32.

Aziz, R., Baluch, M., Patel, S., and Kumar, P. (2022), 'A Machine Learning Based Approach to Detect the Ethereum Fraud Transactions with Limited Attributes', *Karbala International Journal of Modern Sciences,* vol. 8(2), pp. 139 – 151. Doi: 10.33640/2405-609X.3229

Belete, D. and Huchaiah, M. (2021), 'Grid search in hyperparameter optimization of machine learning models for prediction of HIV/AIDS test results', *International Journal of Computers and Applications,* vol. 44 (1), pp. 1 – 12

Bergstra, J. and Bengio, Y. (2012), 'Random search for hyper-parameter optimization', *Journal of Machine Learning Research,* vol. 13, pp. 281 – 305.

Buscema, M. (1998), 'Back propagation neural networks', *Substance use and misuse,* 33(2), pp. 233 – 270, doi: 10.3109/10826089809115863

Chainanalysis (2022), The Chainanalysis 2022 Crypto Crime Report, [online], Available at: https://go.chainalysis.com/2022-Crypto-Crime-Report.html , Accessed: 2 June, 2023.

Chen T., and Guestrin, C. (2016), XGBoost: A scalable tree boosting system, [online], Available at: https://arxiv.org/pdf/1603.02754.pdf , Accessed: 18 July, 2023

Dheepa, V. and Dhanapal, R. (2012), 'Behaviour based credit card fraud detection using support vector machines', *ICTACT Journal on Soft Computing,* 2(4), pp.391 – 397.

Dietterich, T., (2000), 'Ensemble methods in machine learning', *Multiple Classifier Systems,* pp. 1 – 15.

Elgeldawi, E., Sayed, A., Galal, A. and Zaki, A. (2021), 'Hyperparameter tuning for machine learning algorithms used for Arabic sentiment analysis', *Informatics,* pp. 1 – 21.

Farrugia, S., Ellul, J. and Azzopardi, G., (2020), Detection of illicit accounts over the Ethereum blockchain, *Expert Systems with Applications,* vol. 150, pp. 1 – 11.

Friedhelm, V. and Luders, B. (2017), 'Measuring Ethereum-based ERC20 token networks', *Financial Cryptography and Data Security,* pp. 113 – 129, doi: 10.1007/978-3-030-32101-7_8

James, G., Witten, D, Hastie, T. and Tibshirani, R. (2023), 'An introduction to statistical learning: withnapplications in R', 2nd Edition, Boston, Springer

Jung, E., Tilly, L., Gehani, A. and Ge, Y. (2019), 'Data mining-based Ethereum fraud detection', *2019 IEEE International Conference on Blockchain (Blockchain),* Atlanta, Georgia, USA, pp. 266 – 273, doi: 10.1109/Blockchain.2019.00042

Kaggle (2023), Ethereum Fraud Detection Dataset, *Kaggle,* [Online], Available at: https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset?datasetId=1074447&sortBy=voteCount&searchQuery=r , Accessed: May 28th 2023.

(Kotsiantis, S., Kanellopoulos, D. and Pintelas, P, (2005), 'Handling imbalanced datasets: A review', *GESTS International Transactions on Computer Science and Engineering,* vol. 30, pp. 1 – 11.

Linh, P., Li, S. and Mentzer, K. (2019), 'Blockchain Technology and the current discussion on fraud', *Computer Information Systems Journal Articles,* Paper 28, [Online], Available at: https://digitalcommons.bryant.edu/cgi/viewcontent.cgi?article=1027&context=cisjou, Accessed: June 22, 2023.

Liu, Q. and Wu, Y. (2012), 'Supervised Learning', *Northwestern University,* [Online], Available at: https://www.researchgate.net/publication/229031588_Supervised_Learning#:~:text=Abstract,paired%20input%2Doutput%20training%20samples , Accessed: July 6th, 2023.

Malik, E., Khaw, K., Belaton, B., Wong, W. and Chew, X. (2022), 'Credit card fraud detection using a new hybrid machine learning architecture', *Mathematics,* vol. 10(9), pp. 1 – 16, doi: **https://doi.org/10.3390/math10091480**

Meher et al (2019), 'Understanding a revolutionary and flawed grand experiment in blockchain: The DAO attack', *Journal of Cases in Information Technology,* 21(1), pp. 19 – 32.

Miah, M., Rahman, M., Hossain, M and Rupai, A., (2019), 'Introduction to Blockchain', *Blockchain for Data Science',* pp. 1 – 52.

Nzuva, S. (2019), Smart contracts implementation, Application, Benefits and Limitations, *Journal of Information Engineering and Applications,* vol. 9(5), 63 – 75.

Osisanwo, F., Akinsola, J., Awodele, O., Hinmikaiye, J., Olakanmi, O. and Akinjobi, J. (2017), Supervised Machine Learning Algorithms: Classification and Comparison, *International Journal of Computer Trends and Technology (IJCTT),* vol. 48 (3), 128 – 138.

Peterson, L. (2009), K-Nearest Neighbor, *Scholarpedia,* [Online], Available at: https://www.researchgate.net/publication/220580323_K-nearest_neighbor , Accessed 15 July 2023.

Powers, D. (2011), 'Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation', *Journal of Machine Learning Techniques,* vol.2(1), pp. 37 - 63

Sanchez, M., Urquiza, L. and Estrada, J., (2023), 'Fraud detection using the fraud triangle theory and data mining techniques: A literature review', *Computers,* vol.10, pp. 1 – 22.

Srivastava, D. and Bhambhu, L. (2010), Data Classification Using Support Vector Machine , *Journal of Theoretical and Applied Information Technology,* vol. 12 (1), pp 1 – 7.

Tekkali, C.G., and Natarajan, K.(2023), 'RDQN: ensemble of deep neural network with reinforcement learning in classification based on rough set theory for digital transactional fraud detection', *Complex and Intelligent. Systems.,* pp. 1 – 20, doi: https://doi.org/10.1007/s40747-023-01016-4

Tlamelo, E., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B. and Tabona, O. (2021), A survey on missing data in machine learning, *Journal of Big Data,* vol. 140, pp. 1 - 13

Vasek and Moore https://www.cs.unm.edu/~vasek/papers/vasekbtc18.pdf

Wu, J., Huang, B., Liu, J., Li, Q. and Zheng, Z. (2023), Understanding the dynamic and microscopic traits of typical Ethereum accounts, *Information Processing and Management,* vol. 60 (4), pp. 1 – 16.

Yahoo Finance (2023), Ethereum USD (ETH-USD), *Yahoo Finance,* [Online], Available at: https://finance.yahoo.com/quote/ETH-USD/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAFz

1T2I_62vPTNjwUcnU3hRY4K4apiVWRY9qCCn6plRA0Xjtsy3feEiGLeWqk0IOERlMtWwvK7mBF0_Vad6a2yC6F3JWTHDccvbJS6nmhIA6T3VYlICufOCxLBzhDuyutl2HM3hNzrGXKG0Vga1TwW8C88Ah8M2YokbujoMrnErP Accessed: 31 July 2023.

Ying, X. (2019), 'An overview of overfitting and its solutions', *Journal of Physics*, pp. 1 – 6, doi: 10.1088/1742-6596/1168/2/022022

Zhang, Z., (2016), 'Introduction to machine learning: K-nearest neighbors', *Annals of Translational Medicine*, vol.4(11), pp. 218 – 235.

APPENDIX 1

VARIABLE NAME	DESCRIPTION
Index	Row index number
Address	Ethereum account address
FLAG	Describes whether the transaction is fraudulent or valid
Avg min between sent tnx	The average minutes between sent transactions for accounts
Avg_min_between_received_tnx	Average minute between received transactions for account
Time_Diff_between_first_and_last(Mins):	Time difference between first and last transaction
Sent_tnx	Total sent transactions (normal)
Received_tnx	Total received transactions (normal)
Number_of_Created_Contracts	Total created contract transactions
Unique_Received_From_Addresses	All unique addressed from which an account received transactions
Unique_Sent_To_Addresses20	All unique accounts an account sent transaction
Min_Value_Received	Minimum Ether ever received
Max_Value_Received	Maximum Ether ever received
Avg_Value_Received	Average Ether ever received
Min_Val_Sent	Minimum Ether ever sent
Max_Val_Sent	Maximum Ether ever sent
Avg_Val_Sent	Average Ether ever sent
Min_Value_Sent_To_Contract	Minimum
Max_Value_Sent_To_Contract	Maximum Ether sent to contract address
Avg_Value_Sent_To_Contract	Average Ether sent to contract address
Total_Transactions(Including_Tnx_to_Create_Contract)	Total transactions
Total_Ether_Sent:	All Ether sent by an address
Total_Ether_Received	All Ether received by an address
Total_Ether_Sent_Contracts	All Ether sent to contract addresses
Total_Ether_Balance	Ether balance after a transaction
Total_ERC20_Tnxs	All ERC20 transactions
ERC20_Total_Ether_Received	All ERC20 tokens received
ERC20_Total_Ether_Sent	All ERC20 tokens sent
ERC20_Total_Ether_Sent_Contract:	ERC20 tokens sent to other contracts
ERC20_Uniq_Sent_Addr	ERC20 tokens sent to unique addresses
ERC20_Uniq_Rec_Addr	ERC20 tokens received from unique addresses
ERC20_Uniq_Rec_Contract_Addr	ERC20 tokens received from unique contact addresses
ERC20_Avg_Time_Between_Sent_Tnx	Average time between ERC20 token sent transactions
ERC20_Avg_Time_Between_Rec_Tnx	Average time between ERC20 token received transactions
ERC20_Avg_Time_Between_Contract_Tnx	Average time between ERC20 token transaction
ERC20_Min_Val_Rec	Minimum Ether received from ERC20 token transactions
ERC20_Max_Val_Rec	Maximum Ether received from ERC20 token transactions
ERC20_Avg_Val_Rec	Average Ether received from ERC20 token transactions
ERC20_Min_Val_Sent	Minimum Ether sent from ERC20 token transactions
ERC20_Max_Val_Sent	Maximum Ether sent from ERC20 token transactions
ERC20_Avg_Val_Sent	Average Ether sent from ERC20 token transactions
ERC20_Uniq_Sent_Token_Name	Total Number of ERC20 tokens sent
ERC20_Uniq_Rec_Token_Name	Total Number of ERC2O tokens received
ERC20_Most_Sent_Token_Type	Most sent token through ERC20 transactions for an account
ERC20_Most_Rec_Token_Type	Most received token through ERC20 transactions for an account

Appendix 1: Description of variables within the Ethereum fraud detection dataset (Kaggle, 2023)