Novel Machine Learning Models Based Uncertainty Estimation and Sequential Predictions for Blockchain Networks

Ismail Alarab

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of

Doctor of Philosophy

December 2022

Department of Creative Technology
Faculty of Science & Technology
Bournemouth University
Poole, BH12 5BB
United Kingdom

Copyright

Abstract

Cryptocurrencies based-blockchains – decentralised banks of public ledgers of transactions and pseudonymous identities – lure criminals to incognito behind an alphanumeric address to conduct illicit activities over the network. Consequently, this double-edged sword technology urges the necessity of analysing blockchain data to detect illicit activities. In the existing literature, visual analytics have been widely used to gain useful insights from large-scale blockchain data via graph network analysis and visual analytics tools. However, a straightforward visualisation is not very effective with the increasing complexity of the blockchain network. On the other hand, a machine learning approach is capable of dealing with the massive amount of data generated by the public blockchain. Machine learning techniques have provided promising results in many big data applications e.g. social media, IoT, and recently blockchain. Nevertheless, the public blockchain is subject to unseen events that the machine learning model might not be aware of. In this work, machine learning models are developed in a novel way and uncertainty quantification methods are exploited to detect illicit activities in the public blockchain. Various supervised learning models are thoroughly explored using two datasets derived from Bitcoin and Ethereum blockchains. The unexpected events of blockchain are addressed by computing uncertainty estimates besides the machine learning model's predictions. Consequently, a novel uncertainty estimation method is proposed that reveals an effective performance in comparison to existing methods. In this context, uncertainty estimation reflects the model's uncertain predictions about a given input. Subsequently, two distinct frameworks are presented that serve different purposes using blockchain data. The first framework is viewed as an end-to-end prototype of the temporal-GNN classification model – temporal graph neural network – based on an active learning process to reduce the laborious labelling process of blockchain data. In particular, the active learning approach utilises the predicted uncertainties to query the most informative data points where the active learning framework is performed and evaluated using a variety of acquisition functions. The other framework presents a novel model based on sequential predictions. This model refers to RecGNN – a recurrent graph convolutional network – that requires the predictions of the antecedent nodes in the Bitcoin transaction graph as input features.

As a result, this project shows the effectiveness of using tree-based classifiers in classifying data derived from the public blockchain. This allows the detection of illicit activities (e.g. illicit transactions, users, accounts) that operates over the blockchain. It also highlights the competence of models based on graph learning algorithms throughout this project. The active learning frameworks using the temporal-GNN model have revealed promising results. Moreover, the highest performance provided by the RecGNN model is discussed to classify illicit Bitcoin transactions with an accuracy of 98.09% and f_1-score of 91.75%. The experimental results are evaluated using classification metrics and other statistical measurements. The limitations, challenges and possible future directions are also demonstrated.

Keywords: Machine Learning, Supervised Learning, Graph Neural Network, Temporal Graph Neural Network, Uncertainty Estimation, Bayesian Methods, Active Learning, Sequential Prediction, Blockchain.

Contents

List of Figures

List of Tables

Acknowledgements

My biggest fortune in this life is the kind and supportive people I've always been surrounded by. Also, these years during the PhD journey have enriched my insights and cognition on many aspects where I have been riding the crest of the wave!

First, I am highly indebted to Bournemouth University for the fulfilment of this project of my PhD degree under the supervision of Dr. Simant Prakoonwit and Dr. Edmond Prakash.

My deepest gratitude and heartfelt thanks go to Dr. Simant Prakoonwit, who is insanely supportive and experienced. There is no greater word than *Father of AI* to express his knowledge and experience in that field. I cannot be more lucky than having him as a supervisor who has taken me under his wings. I would like to be immensely thankful to him for his continuous support, guidance, and patience throughout the entirety of this experience. A very big thank you goes to the academic staff. I have to say I have been graced with an amazing number of caring and supportive teachers who touched my mind and opened my heart to science.

My gratitude and heartfelt thanks go to my parents and siblings, who endorsed my aspirations and whose endless love fed my enthusiasm.
I would also like to express my sincere thanks to my friends and colleagues (In Lebanon and across Europe) who supported me during these years.
I would like to acknowledge couple of names for their direct support: Nemer Alarab, Salwa Alarab, Darin Alarab, Ziad Ayyad and Dr. Mohammad Jouni.
Thanks and gratitude to any person who triggered my Dopamine.

Table of Acronyms

AHC	Aggloremative Hierarchical Clustering
AI	Artificial Intelligence
AL	Active Learning
AML	Anti-Money Laundering
ANF	Anticident Neighbouring Features
AUC	Area-Under-Curve
BALD	Bayesian Active Learning by Disagreement
BFS	Bread First Search
BNN	Bayesian Neural Network
BTC	bitcoin (digital cryptocurrency)
Bitcoin	Bitcoin blockchain network
CURE-SMOTE	Clustering Using REpresentatives SMOTE
DAG	Directed Acyclic Graph
DEAGO	DEnoising Autoencoder-based Generative Oversampling
DGVAE	Degree-gated Variational Auto Encoder
DKL	Deep Kernel Learning
DNN	Deep Neural Network
DUE	Deterministic Uncertainty Estimation
DUQ	Deterministic Uncertainty Quantification
DoS	Denial-of-Service
ELBO	Evidence Lower BOund
ENN	Edited Nearest Neighbour
ETH	ether cryptocurrency
Ethereum	Ethereum blockchain network
FGSM	Fast Gradient Sign Method

FN	False Negatives
FP	False Positives
GAN	Generative Adversarial Network
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GNN	Graph Neural Network
GP	Gaussian Process
GRU	Gated Recurrent Unit
IP	Internet Protocol
IoT	Internet of Things
K-NN	K-Nearest Neighbours
KL	Kullback-Leibler
KYC	Know Your Customer
LLE-SMOTE	Locally Linear Embedding SMOTE
LSTM	Long Short-Term Memory
MC-AA	Monte-Carlo based Adversarial Attack
MC-dropout	Monte-Carlo dropout
MI	Mutual Information
MLP	Multi-Layer Perceptron
MNIST	Modified National Institute of Standards and Technology dataset
NLLLoss	Negative Log Likelihood Loss
NPV	Negative Predictive Values
OSCCD	Over Sampling-based Classification Contribution Degree
OoD	Out-of-Distribution
P2P	peer-to-peer protocol
PoW	Proof of Work
R-GCN	Relational Graph Convolutional Network
RBF	Radial Basis Function

ROC	Receiver-Operation-Curve
ReLU	Rectified Linear Unit
SHA	Secure Hash Algorithm
SMOTE	Synthetic Minority Oversampling TEchnique
SMOTE-SF	SMOTE using Subset Features
SOMO	Self-Organizing Map Oversampling
SVM	Support Vector Machine
TN	True Negatives
TPR	True Positive Rate
TP	True Positives
TXID	Transaction ID
VAE	Variational Auto Encoder
XAI	eXplainable Artificial Intelligence
dApps	Decentralised Applications
txs	transactions

Chapter 1

Introduction

Contents

In the era of the technological revolution, big data technology has undeniably become the fuel of the data science engine. The term 'Big Data' has been diversely utilised to commonly refer to the ever-growing and rapid pace data. As an example of big data technology, the massive number of data generated daily in blockchain technology provides a fruitful environment in the data analytics approach. This approach eventually contributes to decision-making, analytical reasoning, pattern analysis, and other insightful derivations.

1.1 Blockchain Technology: Short Overview

Blockchain technology has rapidly emerged and gained momentous attention as a mutual distributed ledger of a peer-to-peer (P2P) network. According to the European parliament paper [32], blockchain has substantially had a potential impact in many applications as the most democratic distributed system which could change the lives of many in the globe. Blockchains are digital ledgers of cryptographically signed transactions that are amassed into blocks and

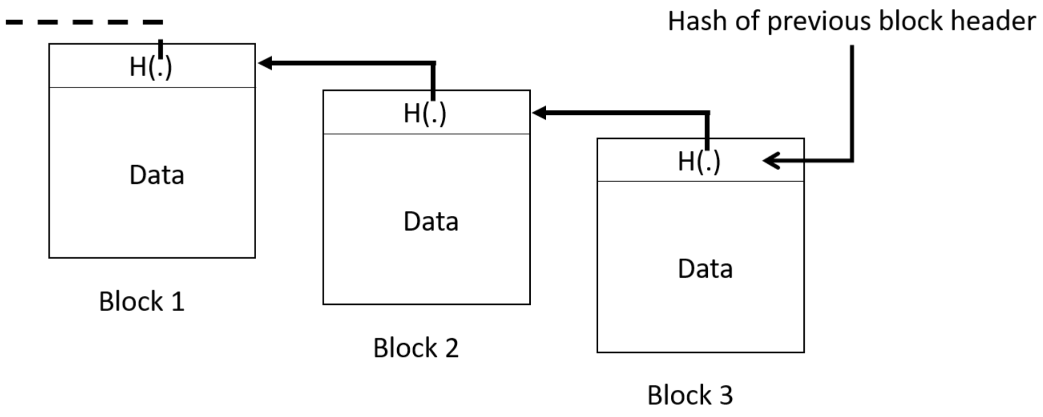


Figure 1.1: Abstract representation of blockchain system.

implemented in a decentralised fashion without any central authority (e.g. government, bank, etc.) [230].

As represented in Figure 1.1, each block cryptographically points to its previous block after checking the validity of the transactions and reaching a consensus decision which prevents malicious users from subverting the system. Each block contains a hash of the content stored in the very previous block as well as a pointer to it, and the data that needs to be stored in the current block. This makes the blockchain tamper-evident as changing the data of 'Block 2' requires changing the data of the blocks on top of it (i.e., in 'Block 3') by referring to Figure 1.1. As more blocks are added after 'Block 3', the difficulty in modifying the previous blocks increases which makes the blockchain tamper-resistant. In addition, each node on the blockchain network owns a copy of the blockchain ledger in which any conflicts on the blockchain are resolved automatically using established rules to ensure consistency in the network.

The main components of the blockchain are cryptographic hash functions, transactions, asymmetric-key cryptography, addresses, ledgers and blocks. In what follows, I concisely present the blockchain components for the reader to understand the blockchain system.

1.1.1 Components of Blockchain

Cryptographic hash functions are one-way functions used to produce a unique output known as *message digest* for the input of any size. Any slight modification on the input produces completely a different output value. One of the popular hash functions that work in many blockchain systems is the Secure Hash Algorithm (SHA) of 256 bits, denoted as SHA-256, where the output is normally represented as a hexadecimal string of 64-characters. Hashing is used to ensure that the data sent is not altered.

Transactions in the blockchain refer to the interaction between the participants as but not limited to the transfer of money between two parties. Transactions use asymmetric-key cryptography to sign or verify the authenticity of the transactions by any node on the network [240]. Thus, no trust is needed between the sender and the participants, where a valid transaction propagates exponentially among the nodes which then becomes validated for inclusion in the blockchain. In the Bitcoin blockchain, transactions are subjected to a set of established rules for independent verification of transactions to prevent spamming, denial-of-service (DoS) attacks or any other malicious attack against the blockchain [16]. Sending a transaction in the blockchain is the transfer of data/value from the source called *input* to the destination called *output* which creates unspent transaction output (UTXO) that contains the value of the transaction and a set of conditions to be spent known as *locking-script*. Each transaction draws the whole amount of BTC from the sender's account balance, in which the receiver gets the required amount of the BTC and the excess amount is sent back to the sender by issuing another output in the same transaction. This new output is known as *change address*.

The sender creates a transaction by hashing the data and then signing it using the sender's private key. This process is called *digital signature*. Then, the participants across the blockchain network decrypt the transaction using the sender's public key to verify the owner of the transaction. The original data is hashed and compared to the already hashed version by the sender in which the matching hash outputs ensure that the data provided is not tampered.

Address, a string of alphanumeric characters, is the hashed version of the user's public key with some additional data, such that the address could be used as the input/output of the transaction. A ledger is the collection of the transactions that are added to the block. Each block contains a block header (i.e., block number, previous block header's hash value, hash representation of the current block, a timestamp, size of the block, the nonce value used for mining), and the block data (i.e., the list of transactions and any other available data).

1.1.2 Applications of Blockchain

Initially, blockchain was proposed in 2008 and developed first in 2009 with the launch of the Bitcoin blockchain [145]. Since then, many currencies blockchain-based have emerged with different features and aims such as the Ethereum blockchain that deploys smart contracts and decentralised applications (dApps) or the Litecoin blockchain that is used for cheaper transactions. Moreover, blockchain applications can also be deployed for non-financial tasks where the ownership histories, the immutability and the integrity of the data are important. Apart from the virtual currencies, blockchain technology is potential in supporting public services [42], record-keeping [119], improving supply chain management [183], reducing risk in business sectors [91], storing medical records in healthcare [173] and the suchlike.

1.1.2.1 Blockchain as the Best-Known System for Cryptocurrencies

Despite the wide-ranging applications of blockchain technology, cryptocurrencies, as one of many possible applications, have remained dominant in this technology. Likewise, the Bitcoin network is the best-known cryptocurrency of blockchain technology. Bitcoin blockchain has become prominent since its first appearance in 2009 by Satoshi Nakamoto as an elusive pseudonym in the white paper [145]. This paper describes how cryptography and a distributed public ledger could be combined to implement virtual currency without any central authority to authenticate and validate payments. Bitcoin, a blockchain of digitally signed virtual currency known as bitcoin (BTC), has been viewed as a decentralised bank of the most secure P2P system [145]. Unlike centralised banks, the transactions in Bitcoin operate inter-personally between individuals apart from any intermediaries. The transactions in Bitcoin occur between the addresses of the relevant users issued from the public and private keys of their wallets. These transactions are then propagated across the Bitcoin network whilst waiting for publishing nodes to verify these transactions and accomplish the required consensus using the proof-of-work (PoW) consensus model [241]. The publishing nodes are known as miners undergoing a consensus mechanism (i.e., PoW) to publish a new block after assuring the validity of the transactions. Briefly, PoW in Bitcoin is the consensus algorithm where the miners apply computational efforts to solve the complex mathematical puzzle. This aims to find the *nonce* of the block which is combined with the block data to produce a hashed output (i.e., *message digest*) starting with a leading number of pre-set zeros in compliance with the Bitcoin blockchain rules [230]. The miner who discovers the hashing puzzle first is rewarded with BTC tokens as mining incentives. The second most popular cryptocurrency blockchain is Ethereum which is well-known for deploying smart contracts besides its financial payment network powered by ether tokens (ETH). Briefly, smart contracts, such as in the Ethereum blockchain, are a collection of functions and states that are deployed via transactions on the Ethereum network [230]. For example, users could create transactions and send them to the smart contracts that offer functionality or service to be executed. Thus, the Ethereum blockchain bolsters tamper-proofed decentralised contracts, and applications, as well as storing data.

1.2 Growing Interest in Currencies Based-Blockchains

Cryptocurrency blockchains have offered a secure P2P network for investors to attractively transact under a pseudonym across the globe without any central authority. For instance, transactions in permissionless blockchains such as Bitcoin and Ethereum are created using pseudonyms (i.e., addresses) in which the real identity of the user is anonymous but the information about its transaction is publicly available. Normal banks are subjected to the Know-Your-Customer (KYC) principle, which is a mandatory requirement for individuals to validate the identity of account holders [50]. While, in the public blockchain ledgers, the addresses are pseudonyms unless they are associated with the identity information [142]. In the early years of the Bitcoin blockchain, many business and non-profit organisations have adopted Bitcoin as an accepted payment, such as WordPress [224], 4chan [1], Wikileaks [6], and many other e-commerce websites and business applications.

Meanwhile, the advent of blockchain has provided a mysterious intriguing technology, between high anonymity (commonly known as pseudo-anonymity) and public availability of transactions. Due to the pseudo-anonymity of the public blockchains, they permit the criminals that are concealed in plain sight in conducting illicit activities such as money laundering, scams, ransomware, mixing services and other illicit activities across the network [27]. For instance, the online black market Silk Road, an online black market platform for selling drugs, presents moderately a popular example in this context [47, 85]. The study in [47] has reported that 23 million USD of illicit items were yearly being exchanged for Bitcoin on the Silk Road website.

On the other hand, the transparency of records in the cryptocurrency blockchains has made intelligence companies and financial regulators vigilant of the blockchain risks, such as technical developments in economic issues [85]. Consequently, the looming of illicit patterns in the complex blockchain network has urged the need for research and crowdsourcing in developing methods to track, analyse and discover patterns in the blockchain data to spot illicit services. Such methods could assist intelligence companies and law enforcement regulations in enhancing the safeguarding of financial systems and boosting AML regulations.

1.2.1 Illicit Activities in Cryptocurrency Blockchain

Regarding blockchain data, the main focus of this project is going to be on capturing illicit activities in cryptocurrency blockchains. The prominence of forensic analysis in cryptocurrency blockchains has widely arisen with the advance of blockchain technology, in which criminals use the blockchain for illicit services. Since the Bitcoin blockchain has gained more popularity than the Ethereum blockchain, I focus more on the Bitcoin blockchain. Though, it is also good to note that while my study focuses more on the Bitcoin blockchain, the same methodology can be extended to data derived from other currencies based-blockchains.

1.2.2 Blockchain as Graph-Structured Data

With the proliferation of data, visualisation has become an effective way to provide comprehensive views and gain actionable insights. The uncovered structures in big data analysis have been often discovered using data visualisation, particularly graph networks. Consequently, graph network visualisation has become an appropriate tool to explore hidden patterns in complicated data. A graph network is a set of interconnected nodes represented by points with edges represented by lines [31]. The main difference between directed and undirected graphs is that the edges in directed graphs are ordered, while they are not in undirected graphs. The combination between the geometry of the graph and the relation between its nodes is the so-called graph network visualisation.

Recent researchers have ardently exploited visual analytics tools by visualising the transactions' and users' graph networks of the Bitcoin blockchain [207]. The graph network of Bitcoin transactions could be constructed by considering nodes as transactions and edges as the payments flow as abstracted in Figure 1.2. In the given example, the payment flow is represented between the transactions in the Bitcoin blockchain where the transaction fees are disregarded. It is also good to note that a single user can be associated with multiple ad-

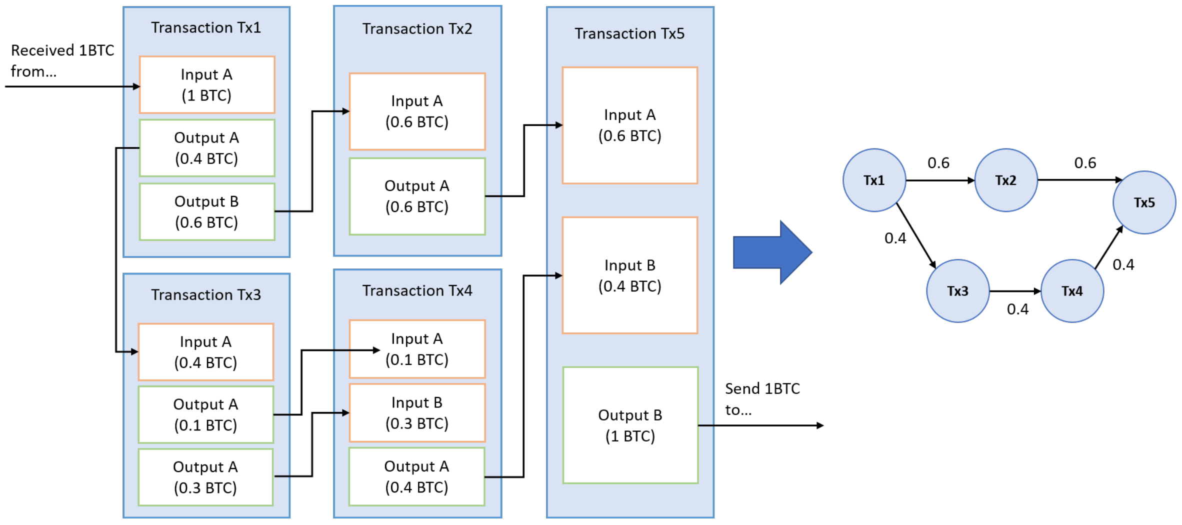


Figure 1.2: An abstracted example of converting payments flow in Bitcoin blockchain to graph network of transactions.

dresses. Multiple connections between two transactions are represented by a single edge only as of the case in the transactions *Tx3* and *Tx4* of the given example. Consequently, the graph network of transactions provides an easier understanding and readability of the payments flow as depicted in Figure 1.2 which will be the main focus of my experiments.

Another representation of the graph network in Bitcoin is the user graph network. It is derived from the transaction graph network by constructing nodes as users and the payments among them as edges. In the Bitcoin blockchain, each user can possess multiple addresses, so the construction of users as nodes is not straightforward. Previous researchers have followed a set of heuristics to convert transaction graph networks to user graph networks. For example, multiple input addresses of a transaction are more likely to be derived from the same user [195]. The latter reference has appeared as one of the early studies to present the heuristics of the user graph which have been subjected to different changes with the evolution of blockchain technology. Henceforth, the user graph is not included in my studies.

1.2.3 Tracking, Analysing and Discovering Illicit Activities

Visual analytics, the core of analytical reasoning, has shown a renaissance with the vast increase in data generation in the past decade. Visual analytics have been widely used as analytical inference to facilitate the decision-making processes and to analyse the patterns of massive data in a favourable way. Visualization underlying the graph network has appeared as a prominent way of gaining insights in blockchain data as appeared in the previous studies [180, 53, 134, 27]. These studies have developed visual analytics methods for Bitcoin data since a straightforward visualisation of such complicated data is not a viable approach. The major act in these studies has addressed tracking, analysing and discovering illicit patterns in the transactions of the Bitcoin network. Moreover, visual analytics methods have been equipped with filtering out unnecessary transactions and displaying the interesting ones to grant the user an insightful visualisation. For instance, the work in [180] has shown the possibility of linking the *change address* of a transaction back to the initial user.

Other researchers have focused on descriptive statistics and quantitative analysis. Such studies encompass analysing and exploring the structural patterns that are tied with Bitcoin transactions such as linking transactions to entities or studying the behaviour of input/output transactions where anonymised identities exist [153, 181, 135, 23]. Visualisation systems of Bitcoin data also exist to extract insights such as 'BitIodine' and 'BitconeView'. For instance, 'BitIodine' is a modular framework that extracts Bitcoin data to cluster and label addresses with the same entities and visualise information derived from the Bitcoin data [195]. This framework has been deployed in popular cases e.g. Silk Road black market. Also, 'BitconeView' has been developed to visualise the flow of the Bitcoin transaction graph that detects mixing processes and transactional patterns [53].

Machine learning has played a dominant role in blockchain data that is favoured with an extensive amount of data and hence boosting the statistical power. In the early stage of the blockchain technology, anomaly detection based unsupervised learning method has been applied to the Bitcoin data in [168, 170] and later in [29]. Generally, these studies have focused on detecting anomalies in the Bitcoin transaction network where the evaluation of the used clustering methods was effective in knowing the suspicious transactions and users. Meanwhile, these methods are not reliable because, without prior knowledge, there might be unjustified detection of anomalies that are not really conducting illicit activities. In the past few years, extensive studies on Bitcoin and Ethereum blockchains data have been conducted using various supervised learning methods [85, 61, 221]. One of the earliest studies in this context is in [85] which has performed a variety of classical supervised learning methods to predict the yet-unidentified entities given a set of known categories in the Bitcoin blockchain. These entities belong to a single person or an organisation that are presumably in control with a single or multiple addresses. Supervised learning methods in [85] have revealed promising results with Bitcoin data. Since then, machine learning has inevitably become an important approach in blockchain technology in which the learning algorithms are capable of digesting the massive amount of data generated daily by the blockchain.

1.3 Aims and Objectives

1.3.1 Research Aims

The main aim of this research is to develop a framework using a novel machine learning method assessed by uncertainty quantification for detecting illicit activities in the blockchain.

Our ultimate goal is to detect illicit services such as money laundering in the cryptocurrency blockchain in an automated way to deal with the encumbrance of the blockchain data. Thus, machine learning, the core of this project, will deal with the complex data derived from blockchain to perform predictions. As unexpected events hinder the performance of machine learning, the uncertainty of the predictions is estimated using various Bayesian approaches which assist in the decision-making process. Moreover, an active learning solution is presented that mimics a real-world application of a human-in-the-loop approach. This research foresees a powerful tool for forensic analysis and intelligence companies to effectively spot illicit services in the blockchain network, wherein this work can be easily extended to other applications with the proliferation of Big Data analysis e.g. anomalies in the internet of things (IoT), fraud in businesses, drug abuse in healthcare, and others.

1.3.2 Research Objectives

Concretely, this research can be roughly divided into four main objectives. The main objectives are:

- To develop a machine learning method to capture illicit activities in the cryptocurrency blockchain
- To propose an uncertainty estimation method that assists the learning algorithms to obtain reliable predictions for enhanced decision-making
- To present an active learning approach as a solution that mimics a real-world human-in-the-loop approach
- To evaluate the proposed methods using real-world cryptocurrency blockchain datasets

1.4 Why Machine Learning Among AI methods?

To answer this question, it is easier to highlight the points that emphasise my choice of adopting machine learning among other AI is a wider field. The massive number of daily generated data from the blockchain network creates a fruitful environment for machine learning – a subset of AI – that learns from historical data. Learning from data could be unsupervised or supervised.

Meanwhile, unsupervised methods including anomaly detection are not reliable to detect illicit activities as stated earlier in Section 1.2.3. Whereas, the preliminary studies using supervised learning had attained adequate results to classify the cryptocurrency blockchain datasets. Moreover, blockchain is a young technology that is subject to evolving events in conducting illicit activities that develop new patterns. Henceforth, supervised learning is one of the adequate tools for learning historical data, uncovering patterns, and quantifying uncertainties.

1.5 Contributions

1.5.1 Major Contributions

Based on this project, the outcomes of this thesis can be summarised as follows:

- Tree-based algorithms show the best performing classical supervised learning methods to detect illicit activities in the cryptocurrency based-blockchain
- Data resampling changes the feature importance scores which in turn affects the explainability of the model.
- A neural network concatenated graph convolutional network layers shows increased performance
- A novel uncertainty estimation method – MC-AA – is proposed as an effective uncertainty estimation method to detect uncertain predictions. The uncertainty estimation using MC-AA outperforms other recent uncertainty estimation methods on the given datasets.
- Including temporal and graph information of the blockchain data leads to higher accuracies using – temporal-GCN model – a combination of LSTM and GCN layers.
- A framework based on an active learning approach is proposed where several acquisition functions are examined. Among the used methods, mutual information using MC-AA uncertainty estimation reveals the best-performing acquisition function to query the labels using the temporal-GCN model.
- A model-based sequential prediction is proposed that outperforms the benchmark methods. Sequential prediction is a promising approach wherein money laundering is a sequence of developed events and not a single occurrence event.

1.5.2 Publications

1.5.2.1 Journal Papers

1. **I. Alarab**, S. Prakoonwit, and M. I. Nacer. Illustrative discussion of mc-dropout in general dataset: Uncertainty estimation in bitcoin. Neural Processing Letters, 53(2):1001–1011, 2021.
2. **I. Alarab** and S. Prakoonwit. Adversarial attack for uncertainty estimation: Identifying critical regions in neural networks. Neural Processing Letters, pages 1–17, 2021.
3. **I. Alarab** and S. Prakoonwit. Effect of data resampling on feature importance in imbalanced blockchain data: Comparison studies of resampling techniques. Data Science and Management, 2022.

4. **I. Alarab** and S. Prakoonwit. Uncertainty estimation based adversarial attack in multiclass classification. Multimedia Tools and Applications, Jun 2022.

5. **I. Alarab** and S. Prakoonwit. Graph-based lstm for anti-money laundering: Experimenting temporal graph convolutional network with bitcoin data. Neural Processing Letters, Jun 2022.

6. **I. Alarab** and S. Prakoonwit. Uncertainty Estimation Based Adversarial Attacks: A Viable Approach for Graph Neural Networks. (Under review in Soft Computing journal)

7. **I. Alarab** and S. Prakoonwit. Robust Recurrent Graph Convolutional Network Approach based Sequential Prediction of Illicit Transactions in Cryptocurrencies. (Under review in Multimedia tools and applications journal)

1.5.2.2 Conference Papers

1. **I. Alarab**, S. Prakoonwit, and M. I. Nacer. Comparative analysis using supervised learning methods in anti-money laundering of bitcoin data. 2020.

2. **I. Alarab**, S. Prakoonwit, and M. I. Nacer. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In Proceedings of the 2020 5th International Conference on Machine Learning Technologies, pages 23–27, 2020.

1.6 Thesis Overview

This manuscript can be roughly structured as follows:

- Chapter 2: **Literature Review**. This chapter covers extensive related works on analysing and detecting patterns and illicit services in cryptocurrency blockchain data. This review covers the wide-range approaches of existing studies that are used to explore transaction and user behaviour in the cryptocurrency based-blockchain data. The used approaches encompass graph visualisation through graph networks, visual analytics tools, statistical measurements using properties and algorithms on the graph network and machine learning algorithms. For the machine learning approach, a comprehensive review of several datasets is also conducted that appeared in the literature where the data are derived from the Bitcoin and Ethereum blockchains to detect fraudulent activities.

- Chapter 3: **Supervised Learning for Detecting Illicit Activities in Bitcoin**. This chapter presents the preliminary study on the graph-structured data derived from the Bitcoin blockchain. First, a detailed presentation of the used dataset is provided. Secondly, the effect of resampling techniques on the feature importance of this dataset is discussed after further data preprocessing and resampling where this study is published in [9]. Subsequently, a comparative analysis – published in [12] – is provided using supervised learning methods to classify the illicit transactions in the Bitcoin dataset. In addition, a model as a combination of GCN and MLP models – published in [13] – is also proposed that provides adequate results with this dataset.

- Chapter 4: **Supervised Learning for Fraudulent Accounts in Ethereum**. This chapter presents another dataset derived from the Ethereum blockchain to classify fraudulent accounts. The same experimental procedure is followed as in the previous chapter where possible. Thus, data preprocessing is applied and a comparative analysis of various supervised learning methods is performed. The class-imbalance issue is addressed by applying a variety of resampling techniques; consequently, the influence of resampling techniques on the feature importance and explainability of the machine learning model is addressed. This chapter is published in [9].
- Chapter 5: **Uncertainty Estimation in Machine Learning**. The behaviour of simple and efficient uncertainty estimation methods and their limitations is discussed. Then, a novel uncertainty estimation method is proposed which is comprehensively examined using different datasets (Bitcoin, Ethereum, GitHub, Cora, MNIST) and several models (GNN and MLPs). The proposed method has revealed significant success in detecting uncertainty besides the predictions of the neural network model. These studies are published in [14, 8, 11].
- Chapter 6: **Proposed Frameworks: Machine Learning for Illicit Activities in Blockchain.** This chapter presents two different frameworks to detect illicit activities in Bitcoin dataset for anti-money laundering purposes. The active learning framework is published in [10].
- Chapter 7: **Conclusions and Future Directions**. This chapter states the conclusions, perspectives and future directions of the thesis.

Chapter 2

Literature Review

Contents

2.1 Introduction

Since its first conceptualisation and development by the alias Satoshi Nakamoto [145], blockchain technology has received widespread attention and found its way towards many financial and non-financial applications. This technology has become the foundation of digital trust using the decentralisation aspect where nobody is in control of the system. In particular, the functionality of the cryptocurrency blockchain has potentially lured many of the public worldwide to transact crypto-tokens due to its strong security as an immutable ledger and the pseudo-anonymity of its transactions [35]. Due to its pseudo-anonymity, criminals have perceived the blockchain as a facilitator for financial crimes to conduct unlawful activities. Thus, criminals remain anonymous as there is no link between their identities and the alphanumeric addresses they use to create transactions. Consequently, many researchers have intriguingly conducted massive studies on cryptocurrency blockchains due to the transparency of the blockchain data since the early years of the blockchain's appearance.

In this chapter, I present the types of illicit activities that are exploited by criminals to illegally transact crypto-tokens on the public blockchain. I state the main categories of services operating over the Bitcoin network, which support licit/illicit or both activities. The rest of the chapter can be roughly categorised by the nature of the approach used to detect suspicious patterns in the public blockchain. I comprehensively review various approaches that have been adopted to detect suspicious patterns or illicit activities in the public blockchain. These approaches encompass graph analysis, visualisation systems, visual analytics, quantitative analysis and machine learning approaches that have been considered in the previous studies.

2.2 Unlawful Activities in Cryptocurrency Blockchain

This section provides a short presentation for the reader to become familiar with the illegal activities carried out by criminals in Bitcoin.

Bitcoin, as an example of the blockchain, has witnessed rapid growth in transactions a few years after its development, whereby it has become a double-edged sword used for legal and illegal activities. In 2021, Chainalysis, a blockchain analysis company, reported that about $US 14 billion were involved in criminal activities of all cryptocurrencies [3]. Previous studies have addressed various types of illegal activities using different approaches in the Bitcoin ecosystem and others. The common types of unlawful activities that appeared in the transactions of public blockchains can be categorised as follows: fraud [236, 98] such as scams [143], PonziSchemes [211, 21], ransomware [111, 160], human trafficking [174], drug trafficking [133, 125], child exploitation [156], darknet market trading [103], extortions [172], money laundering [221, 49], sextortion [161], cybercrimes [135, 37], and so on. These activities are mainly run by services that operate on the Bitcoin network so that the transactions are hard to scrutinise and detect the identity of the user or the origin of the illicit funds. The major types of services that run on the Bitcoin network are listed in Table 2.1 referring to [151].

As mentioned earlier, each Bitcoin user can create multiple addresses to send or receive BTCs. For example, consider a Bitcoin user of address X wants to send BTCs to another user of address Y, where X has a higher amount of BTCs than the required ones by Y. Hence, X sends Y the required amount of BTCs and sends X (him/herself) the excess amount known as a change. As a result, this creates a transaction with input being X (sender) and outputs being Y (receiver) and X (as a change back to the sender). Due to the transparency of the Bitcoin network, this type of transaction is straightforward to de-anonymise. An obscure scenario is when X and a new address Z correspond to the same user and want to complete a payment similar to the previous example. Hence, a transaction is created with an input X (sender) and output Y (receiver) and Z (changed back to the sender with a new address). This makes such transactions hard to de-anonymise. Some services operate on Bitcoin to ensure high privacy for the Bitcoin user, as the case in 'Mixers'. 'Mixers', known as mixing services, are a way to obfuscate the origin of the transactions by mixing the funds of a single user with other users so that it is hard to trace back their source. However, criminals have exploited these services for laundering money, theft, and other illicit activities. For instance, the breach at the Bitcoin exchange, known as Mt Gox, has caused one of the largest thefts in Bitcoin of 850k BTCs which was worth $US 450 million in February 2014 as reported in [215]. Also, blockchain intelligence and forensics company CipherTrace have reported a global amount of

Type of Service	**Purpose**
Exchanges	Trading of BTC to fiat currencies
Payments Gateways	Accepting payments for services in BTC
Gambling	Gambling sites that accept payments in BTC
Pools	Combining processing powers for mining blocks
Miner	Nodes competing to mine blocks
Darknet Markets	Buying and Selling illicit goods and services online
Mixers	Increasing untraceability of BTC sources
Wallet	Storing BTC private keys and balance
Trading site	Exchanging or investing using BTC
P2Plenders	Obtaining loans in P2P using BTC
Faucets	Rewards in BTC to participants
Explorer	Educational websites to explore Bitcoin
P2PMarket	Market place that accepts payments in BTC
Bond Markets	Buying bonds or debts instruments in BTC
Affiliate Marketers	Pay per click in BTC
Video Sharing	Viewing videos by paying in BTC
cyber-Security Providers	Providing cyber-security products for Bitcoin
Cyber-Criminals	Conducting cyberattacks e.g. ransomware
Ponzi	Yielding investment scams
Money Launderers	Converting illicit funds to legitimate ones

Table 2.1: Services operating in the Bitcoin network modified from [151]

$US 4.5 billion in 2019 of Bitcoin crime related to illicit services [2].

The lack of a governance structure and financial regulation on the public blockchain has raised many speculations and concerns since the launching of the Bitcoin system [30]. Due to the public records in Bitcoin, researchers have followed various approaches to explore, analyse and discover the illicit activities in the public blockchains such as but not limited to Bitcoin. This is demonstrated in the following sections.

2.3 Graph Visualisation Using Directed Acyclic Graphs

One of the undertaken approaches to analyse Bitcoin data is through graph network visualisation. Visualisation has received significant attention as a prominent and quintessential way to visualise complicated blockchain data to gain deep insights using graph networks. Many researchers have adopted the graph visualisations of Bitcoin transactions and users which permit one to analyse transactions and extract visual cues. Analysing the Bitcoin network has been considered initially in [180, 181, 135] who provided the fundamental pioneering techniques of studying the behaviour of the transactions using directed acyclic graphs (DAGs).

Reid and Harrigan [180] have performed anonymity analysis and investigated an alleged theft on the Bitcoin network by visualising the transactions' and users' graph networks. The user graph derived from the transaction graph network is constructed by assigning each pair of input transactions to the same entity as noted by Nakamoto in [145]. Breaking the system into an analysable transaction and user graph networks [180] is to reveal patterns and identities originating from different data sources. Reid and Harrigan have also shown the capability of reducing Bitcoin anonymity through graph networks by clustering the public key addresses of the Bitcoin users according to the IP addresses in the ancillary network. Using flow and temporal analyses of the graph network, the egocentric visualisation in [180] has revealed a flow of severe transactions that was transferred from a victim's address to a thief in its case study.

Ron and Shamir [181] have performed an exploratory study using statistical properties to identify the characteristic behaviours of the transaction flow in the Bitcoin transaction graph that might lead to illicit activities. For that, the interesting patterns in the transaction graph are stated as follows: long chains of transactions, fork-merge and self-loops, BTCs that are kept aside for an amount of time and binary tree-like distributions of BTCs among several addresses. The descriptive statistics followed in [181] are performed by revealing the number of small and large transactions, their sums, and others. Consequently, the work in [181] has shown that most of the minted BTCs remained dormant in addresses which never appeared in any outgoing transactions as well as unveiling a sub-graph of large transactions that belongs to a single transaction. This study has also highlighted that a subgraph with large transactions has revealed a significant pattern which could be an attempt to increase anonymity between the transactions.

Meiklejohn et al. [135] have developed a clustering methodology based on heuristics to group Bitcoin wallets (grouping transactions to a single user), then identify the major institution associated with these wallets by evidence. The first heuristic considers that multiple input addresses to the same transactions belong to the same user. Another heuristic used to construct the user graph is by linking the change address of a transaction back to its initial user to cluster addresses of the same entities in the network using a set of conditions (e.g. the transaction is not a coin generation). These conditions are used to show that a public key is considered a one-time change address. This paper has introduced the *peeling chain* concept in which a large amount of BTCs is split between a small amount that is transferred to another address and the remaining BTCs are transferred to a one-time change address. This process is repeated many times to break down the large amount of the remaining BTCs into smaller amounts. Afterwards, the BTCs in the remaining addresses are aggregated together to be transferred again to a single address. It has been shown that the peeling chain can be captured using the heuristic of the change address which is effective in tracking and de-anonymising illicitly-obtained BTCs.

Tharani et al. [202] have discussed how the visualisation of graphs can reflect the suspicious behaviour of fraudulent activities. The graph analyses provided in this section require manual searching and careful scrutinising of the suspicious addresses with prior knowledge. Eventually, these analyses have become hard to pursue with the rapid emergence of blockchain data.

2.4 Visual Analytics

Other researchers have adopted graph visualisation of Bitcoin through visual analytics to reduce the burdensome of data processing and filtering. This has enhanced the visual understanding of large-scale graph networks in the blockchain which provides more effective visualisation such as in [53, 195, 27, 134, 105].

Di Battista et al. [53] have introduced 'BitconeView' as the first visual analytics tool to graphically track the flow of BTCs that are spent and the storage of the unspent transaction outputs over a certain period of time in a portion of a transaction graph visualised on demand. This study has introduced the notion of *purity* ratio which allows understanding when transactions/sources are mixed in a suspicious way. Furthermore, the tool in [53] has shown effectiveness in detecting mixing processes and suspicious patterns.

Spagnuolo et al. [195] have proposed 'BitIodine', a modular framework, to assist in large-scale visualisation of blockchain data by filtering the unnecessary details and presenting the desirable data flow in Bitcoin such as visualising transactions, clustering transactions corresponding to the same entity, finding a path between users or addresses manually. In this framework, user and transactions graphs were built to analyse transactions by collapsing addresses into clusters, which simplifies the huge transaction graph [195]. Moreover, the applications on BitIodine using different case studies have granted another important piece of contribution in [195]. These applications have encompassed identifying an address that likely belongs to the encrypted Silk Road wallet, investigating the CryptoLocker ransomware and finding accurately the number of ransoms paid,...etc. As introduced, Cryptolocker ransomware encrypts the files of the victims using strong encryption.

'BlockchainVis' by Bistarelli et al. [27] is a visual analytics tool to analyse specific characteristics and suppress undesired information. Visual analytics is the integration of visualised data and other factors such as analysis, decision making and the human factor [100]. The idea of BlockchainVis is to either visualise targeted information (transactions) or offer an archipelago view of Bitcoin with further filtering (indicating the number of miners, balance-based filter,...etc) to rapidly visualise information of interest. This tool has provided effective results on the Bitcoin blockchain by detecting a bunch of transactions that were induced from just two Bitcoin addresses. The detected results were shown by the means of the visual analytics tool with further analysis and filtering nodes. However, even with a more advanced visualisation tool, the rapidly evolving events induced by Bitcoin and the manual searching of interesting nodes hinder the performance of BlokchainVis.

The visualisation system by Mcginn et al. [134] has dealt with a systematic top-down approach of Bitcoin to analyse the visually generated dynamic patterns in real time. The system has been displayed at a high resolution with distributed rendering cluster with a canvas of 132 Megapixels to derive effective insights. This work has taken into consideration an effective graph layout visualisation with large-scale data. The visualisation system has spotted high-frequency transaction patterns corresponding to DOS attacks and automated laundering processes. Moreover, this system has considered a general visualisation unlike previous studies with bottom-up approaches that require a starting point. The visualisation in [134] deals with scalability by presenting a limited number of evolving transactions in a short period of time. On the other hand, the visualisation display has reduced the effectiveness of data representation, especially with high-frequency patterns in large-scale graphs. For instance, the 2D representation of this graph reduces the effectiveness of the system wherein many geometrical drawings are subjected to overlapping nodes. Moreover, there is no automated method to detect high-frequency patterns.

Kinkeldey et al. have introduced 'Bitconduite' [105] as a tool that assists visual exploration of financial transactions through filtering and clustering interactions of blockchain data. This tool has considered an appropriate infrastructure to analyse Bitcoin data at different levels such as filtering to derive entities, clustering entities according to similarities, and viewing transactions through an interface for each entity cluster. However, this tool is not oriented to a specific target in filtering undesired information or spotting illicit transactions associated with these entities. Furneaux [66] explores different tools (e.g. Numisight, Maltego, Learnmeabitcoin.com) to visualise transactions and discover illicitly-obtained transactions. For instance, consider that one needs to investigate the connection between a suspected address and a dark web knowing the suspected address. This could be done using the site 'Learnemeabitcoin' - a website that initially teaches about Bitcoin - by tracking the transactions between the two parties. For readers who would like to dig a little deeper into suspicious activities in public blockchains, this book [66] is recommended as a starting point.

Generally, the preceded granular researchers have provided a great contribution to the Bitcoin blockchain, however, they commonly lack a complementary visualisation that is based on the strength of data science tools to deal with the large-scale environment of blockchain data. The visualisation of large-scale data is computationally very expensive, and a preprocessing phase is needed to reduce the size of the data before the actual rendering [155]. In simple words, these frameworks perform manual filtering which lacks intelligent methods to detect illicit activities.

2.5 Graph Network Properties and Algorithms

Other studies have exploited the properties of graph networks or algorithms to study the behaviour of transactions on the cryptocurrency blockchain. Graph algorithms techniques are based on extracting the topological structure of the graph network using various graph algorithms techniques (e.g. indegree/outdegree of nodes, the centrality of the graph network...etc). Regarding the graph properties of the blockchain, the transactions' behaviour can be extracted by analysing statistical properties of the transactions such as the number of inactive addresses, number of transactions per address, per entity, ...etc.

Ron and Shamir [181] have studied the variance between the small and large transactions in the Bitcoin network. Ober et al. [153] have studied the dynamical effects of the Bitcoin transaction graph such as entity merging, number of used public keys, dormant addresses, and other analyses related to the graph network patterns. This study has also shown that dormant coins are crucial when quantifying anonymity since inactive users might own the dormant coins where anonymity is reduced. Baumann et al. [23] have explored the Bitcoin transaction graph by performing descriptive statistics based on the graph mining algorithms to analyse the graph network of transactions. These statistics are derived from the graph structure of Bitcoin transactions such as degree contribution that reflects the individual connectivity of the nodes, average clustering coefficient that measures the global clique of the graph, eigenvalue centrality that measures the influence of a node on others, and the like. This study has shown the possibility of deanonymising some entities by linking the large highly active entities to the publicly known entities on the Bitcoin network such as the case of the Mt. Gox exchange. Bartoletti et al. [20] have studied the transactions' behaviour by analysing the redistribution of the money flow to identify patterns related to Ponzi schemes – an investment scam in which the old investors receive the alleged profits from the new investors – on the Ethereum blockchain. Consequently, a large number of Ponzi schemes have been implemented as smart contracts in the first 3 years of Ethereum's life as observed in [20]. Another study by Fleder et al. [65] has performed graph analysis in Bitcoin by studying the addresses with high importance using the PageRank algorithm. PageRank algorithm provides a metric that reflects the most interesting users in the Bitcoin network. The highly ranked users by PageRank were forwarded for further investigation by linking them to known users on the Bitcoin network. Another study analysing the Bitcoin network during its first 4 years in [126] also exists. Lischke et al. have examined the graph metrics of the Bitcoin transaction graph at that period such as in-degree, out-degree, clustering coefficient and other metrics.

Gaihre et al. [67] have performed an analysis of the transaction behaviour to conclude if the user is concerned about anonymity and privacy on the Bitcoin blockchain. Thus, they have examined three metrics such as reuse frequency of addresses, zero balance, and the idea of splitting up the amount of BTCs into smaller ones via the change address. Also, Gaihre et al. [67] have performed another graph network analysis on the Bitcoin transaction graphs by finding the incoming degree (the number of input edges to a node), connected components (a subgraph such that all its nodes are reachable by any node), and diameter of the network (longest of the all shortest paths) using Bread First Search (BFS) algorithm. As a result, a miner, who accumulated his/her funds in 2010 and split them up into smaller amounts in 2017, has revealed a strong relation between the anonymity-privacy concern and the exchange rate.

Maesa et al. [54] have performed an analysis on the user's Bitcoin graph using graph network properties such as densification, distance analysis, degree distribution, clustering coefficient and centrality measurements. This work [54] has shown that the average distance between nodes is low despite the high diameter value of the user Bitcoin graph. This explains the occurrence of a few node pairs connected by long paths in the graph network which act as hubs between different parts of the network. The preceding statement provides a hint for the artificiality or artificial user behaviour as discussed by [131]. On the other hand, the in-degree frequency distribution of the user Bitcoin graph has revealed some outliers [54]. Maesa et al. [131] have further extended their previous work to investigate the presence of outliers in the in-degree frequency distribution and high diameter by manually analysing the Bitcoin user graph. They have found that the occurring outliers and long hubs are induced by peculiar chains of transactions. Moreover, this work has demonstrated the transactions with *spam* graph measures as pseudo-spam transactions, which were studied and investigated in [131]. Spam graph measures are the measurements that have visible effects on the graph properties as introduced in [131]. However, this work requires huge efforts to manually extract cues from the user graph of Bitcoin.

These studies have effectively explored transaction behaviour by performing quantitative statistics and extracting other graph properties using graph network techniques. These studies have revealed the capability of de-anonymising some entities and discovering other events in the graph network of Bitcoin users and transactions. Indeed, the graph analysis discussed in the previous sections is also used with graph network measurements to derive insights. Though, these measurements are amendable with the evolution of blockchain technology over time. On the other hand, statistical studies are only conducted to validate assumptions or to study some existing events. Moreover, manual searching for patterns requires experienced observers over the blockchain network as well as prior knowledge about earlier illicit transactions.

2.6 Machine Learning Approach

The applications of artificial intelligence (AI) have been immensely favoured with the availability of big data in the past decade. As a subfield of AI, machine learning has become an emergent approach in blockchain technology due to the huge amount of daily-generated data that can be digested by learning algorithms. Primarily, machine learning techniques have been widely applied to detect anomalies [157, 146], fraud [18, 203] and such like applications involving abnormal events. Such techniques have become an apex tool in detecting abnormal events in financial (e.g. fraud that involves financial scams [17], anomalies in blockchain [169]) and non-financial sectors (e.g. anomalies in heart rate [17]).

Machine learning is an inevitable approach when dealing with complex data to provide faster and more reliable outcomes to the user. On the other hand, the public nature of the blockchain data has gained significant interest from the data science community as in [169, 171, 29, 206, 221]. Thus, I adopt the machine learning approach to build upon the existing literature to capture illicit activities in the cryptocurrency blockchain. In what follows, I review the various related contributions that have adopted the machine learning approach in this context.

2.6.1 Unsupervised Learning

One of the earliest studies using unsupervised learning methods on Bitcoin data has been produced in [89]. Hirshman et al. [89] were interested in exploring Bitcoin transactions using machine learning via unsupervised learning to detect anomalous behaviour. This work has shown the ability to identify a user that performed a mixing process in the conducted transactions. The work by Pham et al. [169, 171] has underpinned anomaly detection tasks using unsupervised learning on data derived from Bitcoin. Pham et al. [169] have performed clustering methods such as K-means, Mahalanobis distance and unsupervised support vector machine (SVM) to detect the most suspicious users and transactions using features derived from the user and transaction graphs of Bitcoin. Consequently, they were able to detect three known cases (2 theft cases and 1 loss case) out of 30 known cases. Besides the K-means algorithm, a subsequent study by Pham et al. has adopted the laws of power degree and densification and the local outlier factor (LOF) method that is based on the K-means clustering algorithm using the transaction and the user graphs. Consequently, the densification power law was shown to be more satisfactory for the transaction graph network. Moreover, this study has performed a dual evaluation between the user and transaction graph network to check whether the detected suspicious users match with the detected suspicious transactions [171]. With the LOF method, it was shown the potentiality to detect one known theft that occurred in 2012.

K-means clustering is capable of grouping instances into clusters that are represented by their centroids. However, this algorithm is not effective with outlier detection [139]. LOF is a popular method for detecting outliers, however, it does not scale well in large datasets. Monamo et al. [139] have addressed the preceding weaknesses by investigating the trimmed K-means unsupervised learning method to detect fraudulent activity in Bitcoin transactions. The trimmed K-means algorithm groups the instances together and accounts for existing outliers within the dataset by trimming the outliers lying far from their centroids. This provides a more robust method than that in the classical K-means clustering [139]. The trimmed K-means method has provided successful detection of some known fraudulent activities as well as its promising performance with an improvement in the detection rate of known fraudulent cases [139]. Similarly, Bogner et al. [29] have utilised unsupervised learning methods to detect anomalies on the Ethereum blockchain focusing on the transaction structure in the network.

Using unsupervised learning techniques has been a good attempt to detect suspicious transactions or users. However, the evaluation of unsupervised learning is a tricky problem, especially when dealing with highly diversified data. Also, there is no evidence that the detected anomalies are really conducting illicit activities.

2.6.2 Supervised Learning

Harlev et al. [85] have applied various classical supervised learning methods to predict entities not type of the yet identified in Bitcoin whereby the classifications have provided acceptable outcomes. This study has investigated several supervised learning techniques such as k-Nearest Neighbours (KNN), bagging, and boosting classifiers for reducing the anonymity of the entities in Bitcoin. Consequently, this could pave the way for detecting high-risk transactions. The dataset used in the latter work was provided by Chainalysis that labelled the entities as follows: *Exchange, Hosted-Wallet, Merchant Services, Mining Pool, Mixing, Gambling, Scam, Tor Market* (i.e., marketplaces for illegal trades), and *Ransomware.* The multi-class classification of the entities in [85] has achieved the highest accuracy and f_1-score of 77% and 75%, respectively, using a gradient boosting classifier. Yin et al. [233] have built upon their work using the same dataset in [85]. The authors have applied and compared a variety of 13 supervised learning techniques to predict the entity type in Bitcoin. The distribution of 100k uncategorised entities has revealed that the cybercrime-related entities have formed 29.81% of the entities using the bagging classifier and 10.95% using gradient boosting. Moreover, the methodology performed in [233] provided a beneficial prototype to assist in analysing and investigating the suspicious addresses by their corresponding entities. One of the drawbacks of the dataset in the latter studies is that the training algorithms might be subjected to new entities that are not yet included in the provided categories.

2.6.2.1 Elliptic Dataset: Graph Dataset of Bitcoin Transactions

Meanwhile, graph-structured data have gained much attention in machine learning with a vastly increasing interest [36, 124, 52, 75, 106]. Graph-structured data have led to the exploration of various graph neural network (GNN) algorithms such as graph convolutional networks (GCNs), which have received significant interest as an emergent learning technique operating on graph networks. Briefly, GCNs are neural networks operating on graph-structured data, where the node features are convolved with a kernel to induce new features of nodes that are considered real-valued embeddings. Precisely, GCN seeks to filter the graph signal with a trainable kernel, in which the localised kernel approximates the graph spectra using Chebyshev polynomials [52]. Subsequently, Weber et al. [221] have provided an important piece to the literature in the context of detecting illicit transactions in Bitcoin using GCN.

Weber et al. [221] have introduced graph-structured data of Bitcoin transactions known as the Elliptic dataset which is provided and becomes publicly available by Elliptic company – a cryptocurrency intelligence company. The latter study has proposed an anti-money laundering (AML) solution on the given dataset which is one of the largest publicly labelled data of the Bitcoin transaction graph that incorporates more than 200k nodes as transactions and approximately 234k edges as payments flow. Moreover, this dataset is a collection of 49 distinct graph networks of Bitcoin transactions belonging to 49 timestamps. Also, it is considered as highly imbalanced data where only 2% of the transactions are labelled as illicit that justify illicit services, while 21% are licit, where the rest are of unknown labels. These labels were provided by Elliptic using a heuristic procedure. For instance, the transactions involved in legitimate exchanges such as wallet or miners entities are labelled as licit, whereas the transactions involved with ransomware scams and terrorist organisations' services are labelled as illicit [208].

Regarding the conducted experiments in [221], the authors have benchmarked various classical supervised learning methods against the emergent graph convolutional network (GCN) to classify licit/illicit transactions in Bitcoin. Another purpose of this study is to exploit the performance of the GCN algorithm with the graph-based dataset of the Bitcoin blockchain. Although the random forest classifier does not consider the topology of the graph-structured data, this algorithm has revealed a superior success with an accuracy of 97.3% and f_1-score of 75.9% over GCN and all other benchmark methods on the Elliptic dataset [221]. Moreover, other extensions of GCN models were examined such as Skip-GCN – a GCN model with skip-connection between the intermediate embeddings – and EvolveGCN [162] – a GCN model that trains a separate GCN model for each time-step. Since then, substantial studies have been carried out on the Elliptic dataset as in [127, 234]. Lorenz et al. [127] have addressed money laundering detection using unsupervised and supervised learning methods on the Elliptic dataset. Moreover, they have proposed an active learning solution that is capable of training the supervised learning model using only 5% of the dataset derived from Bitcoin. Subsequently, Lorenz et al. [127] have shown that the unsupervised clustering methods for anomaly detection are not suitable for detecting illicit patterns in the transaction data of Bitcoin. The reason is that anomalies in the feature space are not a representation of any illicit transactions, wherein illicit transactions lie nearby the data points of the licit transactions in the feature space. In [234], Yu et al. have proposed a deep learning approach to address the previous weaknesses in the original work of the Elliptic dataset. Thus, the authors have addressed the dynamical changes of the degree and the adjacent matrix of the graph network, spatial structure similarities between vertices even if they are far in the graph, and the uniform weight assignment which conforms to the characteristics of the transactions. Subsequently, the number of transactions per timestep is included as an additional feature to the given dataset. This paper [234] has examined the performance of several supervised learning models such as random forest, logistic regression, multi-layer perceptron (MLP), graph attention network (GAT) model and GAT extended; GAT is a graph learning model which is somehow similar to the GCN model but learns the edge weights that represent the attention between nodes. 'Extended GAT' introduced in [234] is a GAT model that operates with an extended adjacency matrix which is the addition of the first-order and the second-order adjacency matrices. Embedding derived from the extended GAT model with the original features of the Elliptic dataset has revealed a superior success over other learning models using the random forest classifier with f_1-score of 83.4%. Poursafaei et al. [175] have proposed a graph embedding technique, SIGTran – signature vectors for detecting illicit activities in blockchain transactions. Applying logistic regression classifier on the Elliptic dataset, the node embeddings by SIGTran have shown a significant improvement over other embeddings techniques such as trans2vec [227], node2vec and RiWalk [175]. Briefly, trans2vec performs random walk-based graph representation on the graph network, node2vec learns graph representations by exploring the neighbourhood of each node via truncated random walks [79], and RiWalk targets the structural node representations via role identification [129]. Tasharrofi et al. [201] have proposed DE-GCN – a differential evolutional optimisation for the GCN model – which suits non-convex problems of node classification tasks instead of gradient-based methods. The authors have found that DE-GCN produces better node embeddings than the standard GCN with a shorter time for training using the Elliptic dataset.

Oliveira et al. [154] have devised more features by introducing the GuiltyWalker method that performs random walks from a given node to the antecedent illicit transactions. Briefly, these features correspond to the quantitative statistics of the collected random walks such as their mean size, median size, number of distinct illicit nodes, ...etc. These new features besides the original ones of the Elliptic dataset have provided a significant improvement with an accuracy of 98.3% using the random forest as well as improving predictions of the occurred shutdown event. The pitfall of this study [154] is that the nodes with no path to preceding fraudulent nodes are dismissed and assigned the value -1 as missing features. This prior knowledge before processing random walks has indirectly caused data leakage in the node features where random walks should be fairly performed on all the nodes. Eloul et al. [60] have exploited the dynamical changes of the Bitcoin transaction graph of the same dataset and introduced graphlet spectral correlation analysis. This work has proposed a two-stage random forest classifier associated with the GCN model using different configurations on train/test set split. With the same configuration as in [221], an accuracy of 97.8% was reported with their proposed method in [60]. Also, Jatoth et al. [95] have conducted studies on the Elliptic dataset. In their experiments, they performed feature selection and then applied ensemble learning methods and other classical supervised learning techniques such as KNN to classify licit and illicit transactions. The recent boosting learning algorithm – an adaptive version of extremely gradient boosting (ASXGBoost) – has been proposed by Vassallo et al. [212] to classify illicit Bitcoin transactions. The latter work has presented an extensive comparative analysis on the Elliptic dataset wherein boosting algorithms have admitted promising results. Sheu et al. [192] have explored the self-attention mechanism with the same data using graph attention networks which revealed high performance. Also, a combination of GCN followed by long short-term memory (LSTM) model, the so-called MGC-LSTM, was presented by Xia et al. [228] that explores the graph structure up to k-hops and the temporal information of this dataset. The latter two studies have been carried out using different settings of train/test sets, where the results do not convey a fair comparison with the original work by Weber et al. [221]. Yang et al. [231] have targeted in their work the inductive learning settings of the graph datasets. The authors have introduced a deep generative model based on the variational autoencoder (VAE) framework referred to as DGVAE. The encoder is composed of degree-gated GNN architecture to gather neighbouring information for node representation learning, wherein the encoder learns the parameters of a normal distribution. Subsequently, the decoder is formed of the MLP layer to provide the class predictions. The results in [231] have shown the effectiveness of DGVAE for classifying the Elliptic dataset.

Bellei et al. [24] have shown the effectiveness of the label-GCN algorithm on real-world graph-structured datasets including the Elliptic dataset. Label-GCN assists in labelling the centre node if the labels of its neighbouring nodes are available. This algorithm has admittedly shown further improvement when performed on the Elliptic dataset in a transductive fashion. A transductive setting is when the predictions are part of the graph network that is used in the training phase, whereas an inductive setting is when the graph network used for training is distinct from the one used for testing. Motivated by the original study on the Bitcoin transaction graph [220], my work is complementary to the latter study using the Elliptic dataset–which is detailed in the next chapter–for the following reasons:

- This dataset is one of the largest cryptocurrency labelled data in Bitcoin and all other networks
- The labels target illicit activities in Bitcoin which can be useful to assist anti-money laundering systems.
- This dataset contains temporal and graph information besides its features which is very useful to exploit different aspects of machine learning techniques on this dataset.

2.6.2.2 Ethereum Fraud Dataset

The Ethereum fraud dataset is one of the publicly released tabular datasets derived from the Ethereum blockchain. This dataset was introduced by Farrugia et al. in [61] to detect malicious accounts over the Ethereum network. The Ethereum dataset comprises 4681 accounts labelled between normal and fraudulent accounts with 42 features per account ID derived from Ethereum. The illicit labels represent Ethereum accounts that are involved to scam lotteries, fake initial coin offerings, and Ponzi schemes [61].

The original study [61] has sought to apply an XGBoost classifier on this dataset to detect the accounts that are shrouded with illegal activities over the Ethereum blockchain. The experimental results have shown an effective performance of the classifier. Moreover, feature importances were studied to provide insights about the involved features [61]. Vassallo et al. [212] have also shown significant evaluations on this dataset using XGBoost, LGBM and CatBoost classifiers against the random forest. Bynagari et al. [40] have also performed classification of the same Ethereum dataset in which the XGBoost classifier has outflanked the random forest at the account level.

I include the Ethereum fraud dataset in the experiments that do not require graph-structured data. I choose this dataset as another example of the cryptocurrency blockchain dataset which is enriched with a remarkable number of features. Moreover, it is the largest publicly labelled dataset available for the Ethereum network.

2.6.2.3 Other Cryptocurrency Datasets

Tan et al. [200] have claimed that an effective and automated method is needed to detect Ethereum fraud. For that, this work [200] has performed GCN using an Ethereum-based transaction dataset to detect Ethereum fraud. Consequently, the experimental results have revealed an accuracy of 95% for detecting fraudulent transactions in data derived from the Ethereum blockchain. Subsequently, Aziz et al. [19] have examined the performance of various classical supervised learning methods with the light gradient boosting machine (LGBM) algorithm to classify fraudulent accounts in Ethereum. LGBM and XGBoost have provided the highest accuracies, while LGBM has revealed a significant outperformance in comparison to other supervised learning methods [19].

Hu et al. [93] have addressed the behaviour of the transactions generated by the smart contracts and users to identify the malicious smart contracts in the Ethereum network. The authors have proposed a data slicing algorithm for the smart contracts and then trained the LSTM model to classify these malicious contracts. The evaluation metrics have yielded satisfactory results [93].

Wu et al. [226] have tackled the detection of the addresses belonging to mixing services. For that, they have applied the network motifs concept in a novel way for Bitcoin mixing detection. Network motifs are small subgraph patterns that appear more frequently in the graph network than in a randomised case. Wu et al. have conducted a systematic study by performing feature-based analysis followed by model training and detection. Their extensive experimental results have demonstrated the effectiveness of the detection model applied to Bitcoin real-world datasets.

Yuan et al. [235] have conducted a systematic study to detect phishing scams on Ethereum from the transaction records in which they have extracted the features using a network embedding method such as node2vec. Their experimental results have attained f_1-score of value 84.6% using one-class SVM to classify the phishing scams in the Ethereum blockchain.

Nerurkar et al. [151] have proposed an ensemble tree model and compared it against various classical supervised learning methods for multi-class classification using data derived from Bitcoin. The main aim of the learning algorithm task is to identify the different licit-illicit categories of users which are highlighted in Table 2.1. For that, the authors have engineered nine features for a dataset composed of 1216 real-life entities on Bitcoin. This study has reported the highest accuracy for the random forest classifier and their proposed ensemble tree model over XGBoost, logistic regression and SVM models.

Wang et al. [218] have carried out a large-scale analysis of ransomware activities in Bitcoin. The authors have developed a clustering method to group the Bitcoin addresses of a single user whereby more addresses can be associated with ransomware payments. In their experiments, they have applied a variety of graph embedding algorithms to extract features in the presence of known labels whereby they trained multi-class classification models to identify the industry identifiers of users. Consequently, Wang et al. have found that ransomware criminals are more likely to distribute ransoms into multiple industries rather than using the mixing Bitcoin services. This study helps to understand ransomware activity in Bitcoin.

The recent work by Tian et al. [204] has proposed an attention-based GNN model which computes the graph embeddings of neighbouring addresses with different distances according to the structural information of the transaction network. Moreover, another part of the model is the auto-encoder which extracts the temporal features derived from the transaction records. The experimental results have shown the importance of the address's neighbour embeddings and hidden temporal features for enhanced illicit identification.

On the other hand, Hassan et al. [86] have comprehensively surveyed anomaly detection in the blockchain network with a variety of machine learning approaches as well as a huge collection of papers adopting different cryptocurrency datasets. I refer to this survey as supplementary material for the reader who would like to be immersed more in the approach of machine learning for anomaly detection using data derived from the public blockchain.

2.7 Major Knowledge Gaps

The preceded studies have significantly revealed some gaps and consequently arisen new challenges. One of the drawbacks of these studies lies in the encumbrance of representing large-scale data of blockchain. These researches have underpinned visual analytics tools which have admittedly provided an effective exploratory understanding of patterns in a visualised system. However, they lack to an intuitive and automated method for effective visualisation to derive deep insights. For example, filtering out manually to only visualise interesting patterns is rather encumbered in a rapidly increasing technology, wherein an automated and intelligent method is needed.

Another drawback lies in using machine learning algorithms in a vast increasing technology. Meaning, there is always a chance of new arising events in the blockchain network which might not be captured with a pre-trained machine learning model. For instance, the clustering algorithms to detect anomalies are weakened by the dynamical changes of data distributions which might erroneously be reflecting outlier patterns. Supervised learning has shown promising results with reliable predictions and evaluations of blockchain data. However, the newly unexpected events in the blockchain data might hinder supervised learning performance.

Henceforth, the public availability of blockchain data and the need for an automated method to detect illicit activities have pointed this research towards exploring machine learning methods including graph learning algorithms. Due to the evolving events encountered by the blockchain, various uncertainty estimation methods of the machine learning models are examined. Due to the massive amount of the generated blockchain data, an active learning approach is performed which aims to reduce the burden of the labelling process of the blockchain data.

2.8 Concluding Remarks

Tracking, discovering and extracting knowledge have become an indispensable and growing field in the blockchain network since its appearance. The pioneering studies on Bitcoin have started by analysing the graph network of transactions followed by identifying heuristics to convert the transaction graph to a user graph. Subsequently, the studies on the blockchain network have encompassed several approaches such as graph network analysis, visual analytics tools, quantitative statistics, measurements of the graph network properties, and machine learning methods. These approaches have addressed the suspicious patterns and illicit activities on the public blockchain. Moreover, some of these studies have provided the capability of visual analytics tools to de-anonymise some transactions, users or behaviour in the public blockchain. In addition, other contributions have provided exploratory work to gain insights into the users' behaviour over the blockchain network. In my review, I have focused on the Bitcoin and Ethereum blockchains. Generally, the evolving blockchain technology and its complex nature have limited the performance of visual analytics and descriptive tools to study the suspicious behaviour of transactions in the public network. Its complex nature has unleashed the strength of the machine learning approach in this field. Machine learning has been extensively exploited by many researchers. As an automated and intelligent method, machine learning has provided promising results in detecting anomalies that are conducting illicit activities on the cryptocurrency blockchain. Even though, the machine learning approach is still very challenging in the blockchain. Mainly, machine learning relies on historical data which is normally considered static data. However, blockchain is an evolving technology in which new services come into operation and other services are shut down. Furthermore, the techniques used by criminals to perform illegal activities are evolving over time. Considering the machine learning approach, I have surveyed different studies using data derived from the public blockchain. However, I merely focus on the Elliptic dataset and Ethereum fraud dataset. The link to the latter datasets and other data derived from the public blockchain are tabulated in Table 2.2.

Though, supervised learning has provided significant success in predicting illicit activities on the blockchain of different cryptocurrencies. More precisely, random forest and XGBoost classifiers have become the pioneering classifiers in detecting illegal patterns in the public blockchain. This will be discussed in the next chapters.

Data Type	**Description**	**Website link**
Bitcoin [221]	Bitcoin graphs of partially labelled transactions as licit or illicit	kaggle.com/ellipticco/elliptic-data-set
Ethereum [61]	Labelled dataset of Ethereum accounts as non-fraud or fraud	github.com/sfarrugia15 /Ethereum_Fraud_Detection
Ethereum [20]	Dataset of Ponzi Ethereum addresses	docs.google.com/spreadsheets/d /1cPanswNPVjoP4xENF7a2AjQQXX _ByZPbk1qOs6aFEPs/pubhtml
Bitcoin [21]	List of scam Bitcoin addresses	docs.google.com/spreadsheets/d/e /2PACX-1vS8JHql9CP _XWCfJ9VG-9VXDTizHPX1w SQBwnsg9QDasPypmfelpfc_ 9uIXk73vaMHXlLGRk2veYXFk /pubhtml
Bitcoin [205]	List of Bitcoin scam addresses and domains in Bitcoin	goo.gl/k5PCOZ
Bitcoin [123]	List of illicit Bitcoin addresses	https://raw.githubusercontent.com /JetQe/Bitcoin-network-illicit-addresses /master/illicit_address.csv
Bitcoin	Public database of Bitcoin addresses used by criminals	https://www.bitcoinabuse.com/
Bitcoin [151]	Public dataset of Bitcoin addresses and features of illicit entities	https://drive.google.com /open?id=1YdPj8whgbCKORuW3E0 rhgfQIHkcFj9S5
Bitcoin [151]	Public repository of a collection of Bitcoin addresses	https://github.com /pranavn91/blockchain

Table 2.2: Public available blockchain based-cryptocurrencies data.

Chapter 3

Supervised Learning for Detecting Illicit Activities in Bitcoin

Contents

Money laundering is the process of disguising illegal funds with legitimate ones by mixing or exchanging [152]. With the advance of Bitcoin technology, criminals have seen the Bitcoin blockchain as a facilitator of the money laundering process, where the user's identity is concealed behind a pseudonym known as the address. Although this trait permits concealing in plain sight, the public ledger of the Bitcoin blockchain provides more power for investigators and allows collective intelligence for anti-money laundering and forensic analysis. The preceded studies on blockchain technology have attained considerable success using a machine learning approach to detect illicit activities on the public blockchain, particularly in Bitcoin.

Motivated by the machine learning approach applied to the blockchain, I perform experimental studies on the data derived from the Bitcoin blockchain. In the first part of this chapter, I provide a detailed description of the given Bitcoin blockchain dataset as the input for the machine learning models. In the second part, I address the class imbalance by applying a variety of resampling techniques to the Bitcoin dataset and I study the feature importance of the dataset before and after data resampling. Henceforth, I discuss the effect of the resampling on the explainability of the machine learning model using the provided dataset. In the third part, I propose ensemble learning – a combination of machine learning predictors [194] that prevails over other classical learning methods at predicting licit/illicit transactions of the Bitcoin dataset. Lastly, I exploit the graph structure of the Bitcoin dataset by proposing a model-based graph convolutional network (GCN) to classify the Bitcoin transactions between licit and illicit. In this part, I highlight the competence of the proposed GCN model in classifying Bitcoin transactions.

3.1 Bitcoin Dataset Description: The Elliptic Data

As an example of the cryptocurrency blockchain data, the so-called *Elliptic data*[1] belongs to real Bitcoin transactions, which form a directed graph network consisting of nodes representing Bitcoin transactions and edges representing payments flow from the source to the destination. The transactions of this dataset are distinguished between licit, illicit or unknown labels. The licit labels belong to Bitcoin mining, exchanges, wallet provider and the such like licit services. Whereas, the illicit labels are associated with illegal transactions such as thefts, scams, malware, ransomware, and other illicit services. As depicted in Figure 3.1, this dataset comprises 49 distinct directed acyclic graphs (DAG) of Bitcoin transactions associated with 49 timesteps uniformly spaced within an interval of two weeks each. Moreover, each timestep represents a distinct collection of transactions to form a single connected DAG that has appeared within less than three hours on the Bitcoin blockchain [221]. There are 203,769 nodes as transactions and 234,355 directed edges of the payments flow, where 2% (4,545) of the nodes are labelled as illicit transactions, and 21% (42,019) of nodes transactions are licit. However, the remaining transactions are enriched with node features but with unknown labelling. The nodes of the graph network are formed of 166 features which are constructed using only publicly available information from the Bitcoin blockchain [221]. The first 94 features (a timestep + 93 local features) of the nodes belong to the local information of the Bitcoin transactions such as timestep, transaction fees, number of input/output, etc. The remaining 72 features (aggregated features) represent the aggregated information, obtained from one-hop backward/forward aggregation of graph nodes, which are associated with the structural information of the graph network as forward from the centre node, giving the maximum, minimum, standard deviation and correlation coefficients of the neighbour transactions for the same information data (number of inputs/outputs, transaction fee, etc.). By disregarding the unknown labels of this dataset, the total number of labelled transactions becomes 46,564 distributed between the licit and illicit class distributions as depicted in Figure 3.2.

[1]https://www.kaggle.com/datasets/ellipticco/elliptic-data-set

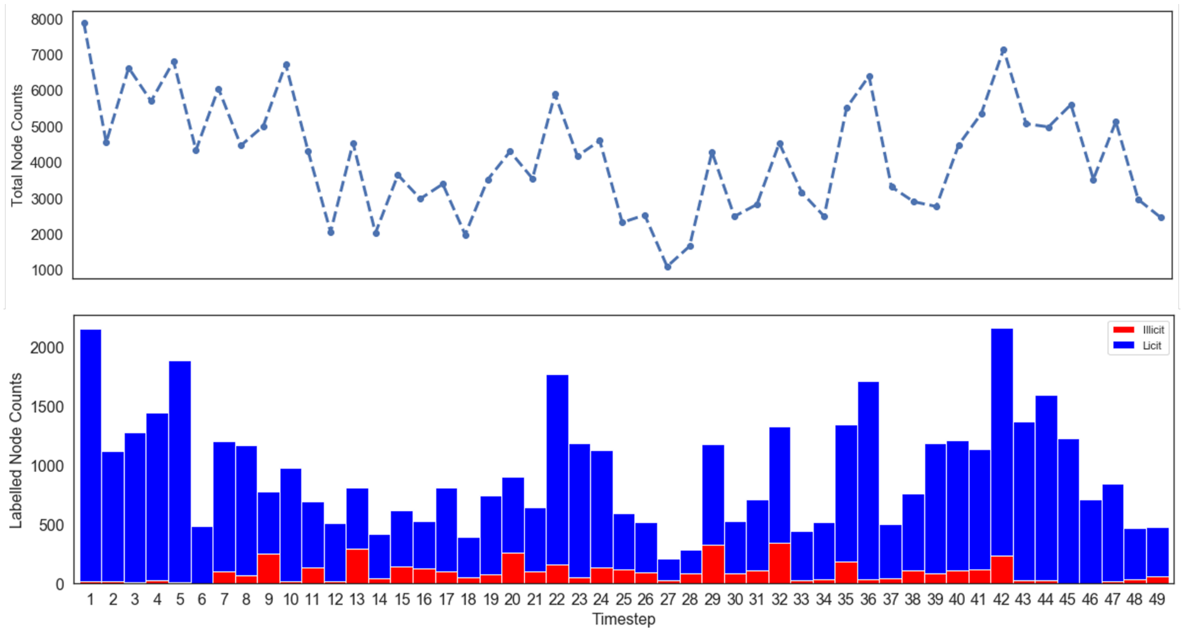


Figure 3.1: Total node counts (top subplot) and labelled node counts (bottom subplot) versus timesteps of the Elliptic data.

The provider of the Elliptic data has anonymised its features and transactions IDs (TXID). Thus, I represent the features of this dataset in my experiments as follows:

- First local feature: *Timestep*
- Local features: *local_feat_2*, *local_feat_3*, ..., *local_feat_94*
- Aggregated features: *aggr_feat_1*, *aggr_feat_2*, ..., *aggr_feat_72*

Moreover, the licit class is represented with value '0' and the illicit class is represented with value '1'.

3.2 Influence of Class-Imbalance on Model's Explainability

Cryptocurrency blockchain dataset encounters a class-imbalance problem due to only a few known labels of illicit activities on the blockchain network. As an example of a blockchain dataset, the Elliptic data is a highly imbalanced graph-structured data of Bitcoin transactions. In its original work in [221], the authors have shown that random forest has provided the best performance on this dataset whereby this work has been followed by substantial contributions as specified in Chapter 2. Regardless of the promising results provided by the preceding contributions, there is no comprehensive study addressing the high-class imbalance that is embedded in this dataset. Moreover, dealing with this type of dataset is challenging due to its high dimensionality and class imbalance.

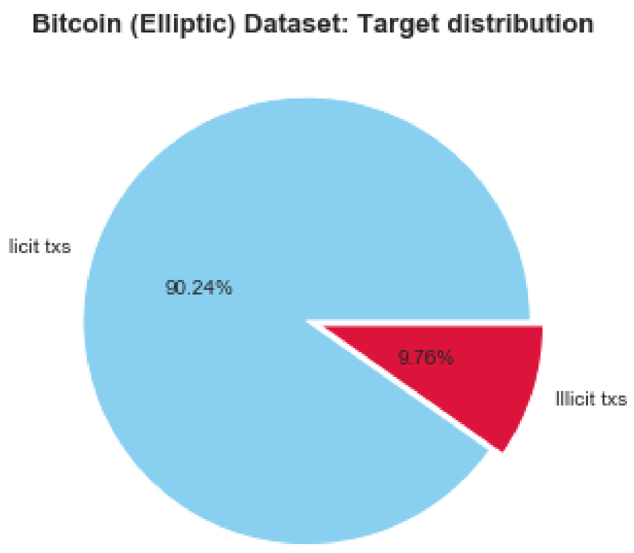


Figure 3.2: Licit/illicit class distributions of the Bitcoin dataset.

Meanwhile, resampling techniques have evolved in the past few decades starting from the generic Synthetic Minority Resampling Technique (SMOTE) and its variants, to the most recent adaptive oversampling techniques to tackle the class-imbalance problem [87, 214, 109, 62]. Addressing class imbalance using SMOTE and its variants has shown significant success in the literature due to their simplicity and outperformance. For this purpose, I provide a comprehensive study using a variety of resampling techniques on the Bitcoin dataset to point out the impact of resampling techniques on the importance which is tied to the explainability of the model.

Firstly, I perform a preprocessing step on the Elliptic data to reduce the dimensionality of the data. Then, I apply various classical supervised learning methods to classify the licit/illicit transactions. I show that the random forest has provided the best performance among other classical models on this dataset. Moreover, the random forest with the preprocessed data has outperformed the models used in the dataset's original contribution [221].

Subsequently, I address the class imbalance by applying a variety of oversampling and undersampling techniques. I evaluate and compare the performance of the random forest classifier after data sampling using the following metrics: accuracy, precision, recall, f_1-score receiver-operation-curve (ROC), and area-under-curve (AUC) scores. I discuss my best results using Edited Nearest Neighbours (ENN) applied to the whole dataset that admits an accuracy greater than 99%. On the other hand, feature importance in the blockchain datasets is indispensable and plays a pivotal role in the explainability of the supervised learning model. Unlike feature selection, the term "feature importance" refers to studying the importance of the features after training the model whereas the former term refers to studying the importance of the features before training the model. So, I point out the influence of data

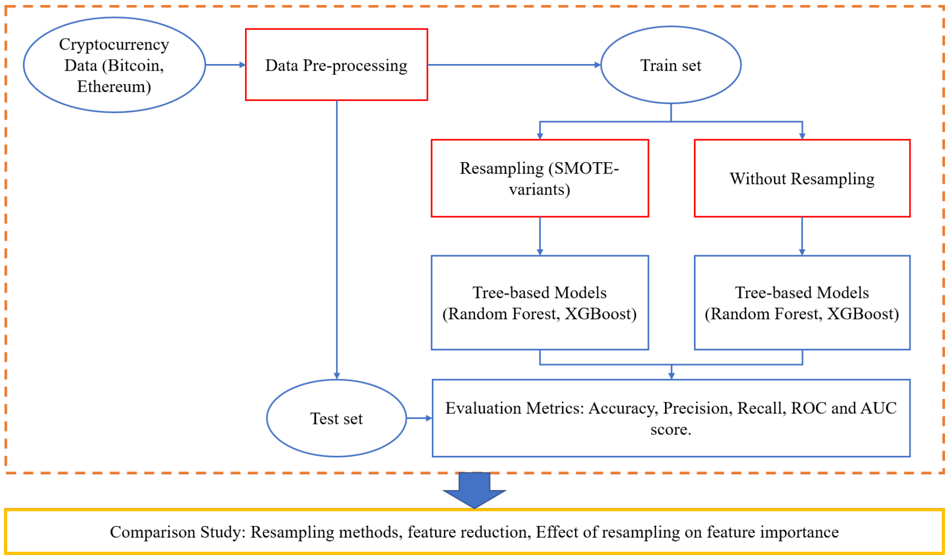

Cryptocurrency Data (Bitcoin, Ethereum)
Data Pre-processing
Train set
Resampling (SMOTE-variants)
Without Resampling
Tree-based Models (Random Forest, XGBoost)
Tree-based Models (Random Forest, XGBoost)
Test set
Evaluation Metrics: Accuracy, Precision, Recall, ROC and AUC score.
Comparison Study: Resampling methods, feature reduction, Effect of resampling on feature importance

Figure 3.3: Schematic representation of the method used in this study.

resampling on the feature importance that has a significant impact on the explainability of the used model. I verify the preceded statement by performing the Wilcoxon signed-rank test – a non-parametric statistical hypothesis test – where I can provide evidence that the scores of the feature importance are different before and after applying a resampling technique.

In what follows, the methods used in this experiment are demonstrated. Subsequently, the experimental study, discussions and a conclusion are provided. A schematic representation summarising the whole process in this section is depicted in Figure 3.3.

3.2.0.1 Resampling of Blockchain Data

Despite the promising results provided by the preceded studies, only a few of them have considered resampling techniques to address the class imbalance problem in the given datasets. For instance, the classification results of the blockchain datasets in [21] and [85] have shown a further improvement with the random undersampling/oversampling techniques and SMOTE technique, respectively. Also, the work in [41] has applied data sampling techniques to the Elliptic data [221] where the classification outcomes of the resampled data have revealed effective results on the data derived from the blockchain. However, the preceded studies lack a comprehensive discovery of the wide range of the existing resampling techniques such as SMOTE-variants on the data derived from the blockchain. The class-imbalance problem can be tackled through oversampling by adding new instances, undersampling by removing noisy instances, or hybrid sampling as the combination of oversampling and undersampling methods. The main idea of oversampling is to increase the number of instances in the positive class near the decision boundary that is already subject to vast class skews. SMOTE is a well-known technique that blindly interpolates positive instances to address class-imbalance [44]. Other SMOTE-variants also exist that are more guided than SMOTE e.g. borderline-SMOTE [81] in which these variants take into consideration the informative areas near the decision boundary to generate new data points.

3.2.0.2 Feature Importance and Model's Explainability

The scarcity of the labelled datasets is a key challenge in the machine learning domain where the researchers have limited knowledge about the fraudulent accounts/transactions in the public blockchain and generally in the financial sector. On the other hand, explainable artificial intelligence (XAI) is an emergent research direction which assists the user in interpreting the predictions provided by the machine learning models [113]. The study in [221] has addressed the explainability of the machine learning predictions through visualisations to support anti-money laundering. Also, the study in [61] has provided insights about the importance of all involved features in analysing the activity of the fraudulent accounts in the dataset derived from the Ethereum. Explainability could be any technique that justify the decision made by the model. One of the popular techniques of explainability is feature importance which is finding the most important features that lead to the predicted output.

Motivated by the preceded studies on the blockchain, I perform a comprehensive study using various oversampling (SMOTE and its variants) and undersampling techniques to address the class imbalance in the Elliptic data. Since feature importance is one of the popular XAI techniques, I will study if the resampled data affects feature importance which directly influences the explainability of the machine learning models.

3.2.1 Methods

In this section, I provide the necessary details of the experiments that are carried out using the described Bitcoin dataset.

3.2.1.1 Data Preprocessing

Firstly, I compute the linear correlation by finding the Pearson's correlation coefficient [222] between the features to remove those having strong correlation coefficients in the Elliptic data. The features with a correlation coefficient greater than 0.9, which is arbitrarily chosen, are excluded. Thus, the feature space is reduced to 91 features. An additional feature reduction step is applied to the columns by removing the features with non-informative distributions. In other words, the features having some unique values less than 10 and highly skewed to a single value as the case in *local_feat_16* as depicted in Figure 3.4. This feature acquires 6 unique values in which most of the feature vectors belong to a single value. By disregarding these features, this eliminates further dimensions resulting in a dataset of 85 features which are visualised in Figure 3.5.

The dataset is divided into train/test sets according to the temporal split in which the

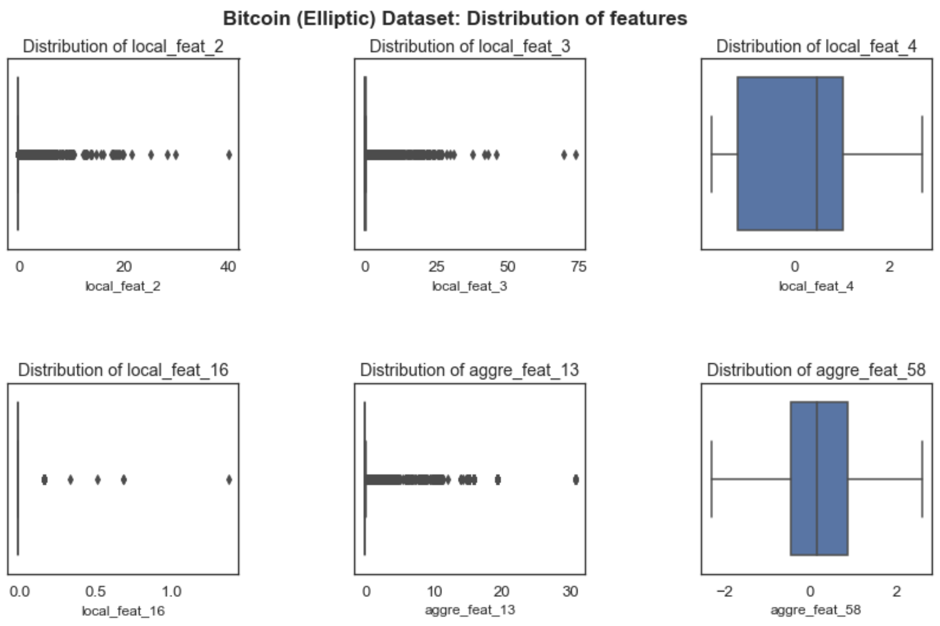


Figure 3.4: Boxplot of some features in Bitcoin dataset. Indication of how the features in the data are spread out.

first 34 timesteps belong to the train set and the remaining 15 timesteps belong to the test set, to perform licit/illicit transactions classifications using supervised learning algorithms. The validation set – a subset of the train set – is chosen to tune hyper-perameters and it corresponds to timestamps from 29 to 34.

3.2.1.2 Benchmark Methods

In my experiments, I study the performance of the Elliptic data after further feature reductions using various supervised learning algorithms as follows:

we have applied a variety of the supervised machine learning algorithms to classify the Elliptic data as follows:

- Random Forest
- ExtraTrees
- Gradient Boosting
- XGBoost
- Logistic Regression
- Multi-Layer Perceptron (MLP)

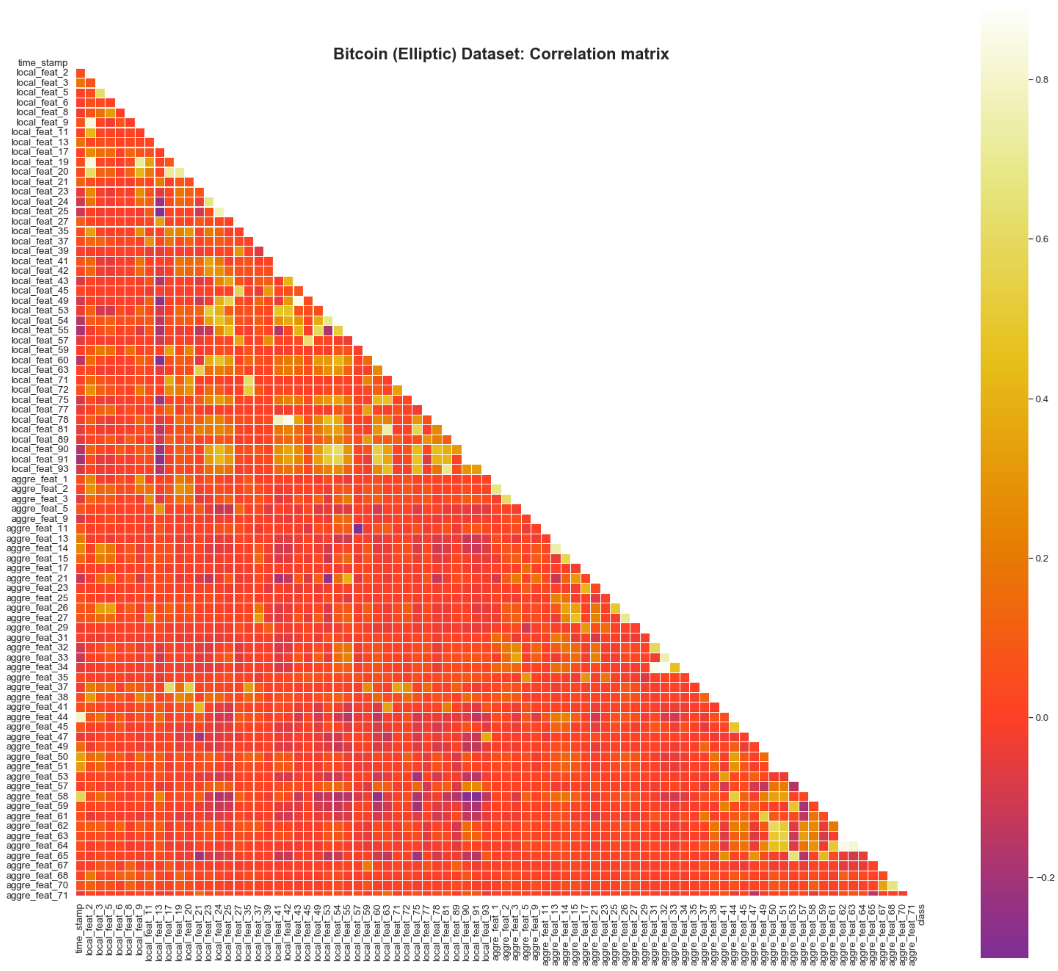


Figure 3.5: Correlation matrix highlighting the features after feature reduction process of the Bitcoin dataset. The number of features becomes 85.

These supervised learning methods have gained popularity in the blockchain data due to their promising performance, referring to [221, 85, 61, 12]. Basically, the random forest classifier randomly chooses a subset of features to construct a decision tree with the best split over its nodes, where multiple trees are formed to provide an ensemble of decision trees [34]. ExtraTrees algorithm is similar to the random forest that constructs a decision tree but with a random split over the nodes [74]. These bagging algorithms are known to reduce overfitting. XGBoost is an optimisation of gradient boosting algorithm [45]. Gradient boosting is formed of a sequential number of trees as a weak classifier to obtain a strong classifier using gradient descent. Lastly, logistic regression and MLP are function approximations, wherein the former one models a linear decision boundary to classify the data [225], whereas the latter one handles non-linearly separated data [73].

3.2.1.3 Resampling Methods

We have studied the effect of more than 80 resampling techniques on the Elliptic data using SMOTE, its variants, and other recent resampling methods (see Appendix 7.4). However, I present the best performing resampling techniques on the given dataset including SMOTE as follows: Kmeans-SMOTE [56], AHC [48] (Agglomerative Hierarchical Clustering), Borderline-SMOTE1 [82], Borderline-SMOTE2 [82], SOMO [55] (Self-Organizing Map Oversampling), SMOTE-TomekLinks [22], DEAGO [25] (DEnoising Autoencoder-based Generative Oversampling), Safe-level-SMOTE [38], TRIM-SMOTE [176], CURE-SMOTE [128] (Clustering Using REpresentatives SMOTE), LLE-SMOTE [217] (Locally Linear Embedding SMOTE) and the recent techniques SMOTE-SF [132](SMOTE using Subset Features) and OSCCD [96] (Over Sampling-based Classification Contribution Degree).

These techniques oversample new instances near the decision boundary in guided and more sophisticated ways than SMOTE. For instance, Kmeans-SMOTE is a combination of clustering algorithm and SMOTE, Borderline-SMOTE selects the most informative regions near the class boundary to oversample the minorities, and SMOTE-SF tackles high dimensional datasets by using SMOTE on a subset of features. Regarding undersampling, I consider applying the ENN technique to remove noisy instances in overlapping distributions. I refer the reader to the reference [109] for a comprehensive overview of the SMOTE-variants techniques.

3.2.2 Experiments

3.2.2.1 Experimental Settings

In my experiments, I use sklearn [166] and smote-variant packages [110] in Python programming language. I apply a variety of the classical supervised learning methods using the Bitcoin dataset, wherein the hyper-parameters are empirically tuned in these models on the validation set by looping over set of values that are arbitrarily chosen. These hyper-parameters are summarised in Table 4.1.

I evaluate supervised learning algorithms using accuracy, f_1-score and AUC-score as provided in Table 4.2. Subsequently, I apply resampling methods to the Bitcoin dataset, wherein I perform training and evaluations using the same supervised learning algorithm per dataset for a fair comparison. Thus, I opt for the best performing algorithm which is the random forest, as depicted in Table 4.2, to classify illicit Bitcoin transactions. Afterwards, I apply the above-mentioned oversampling and undersampling methods to study the effect of the class-imbalance problem on the Bitcoin dataset. I use the default hyper-parameters for all oversampling methods except for the following resampling methods which are tuned on the validation set as follows:

- OSCCD: Number of clusters is set to 3.
- LLE: Number of components of the embedded feature space is set to 5.

Model	**Hyperparameters**
Random Forest	Number of trees=50; max depth=50; max features=5
ExtraTrees	Number of trees=50
Gradient Boosting	Learning rate=0.1
XGBoost	Number of trees=300; max depth = 50; learning rate=0.1
Logistic Regression	C=10.0; epochs=50
MLP	Optimiser='Adam'; hidden layer size=50, epochs=50

Table 3.1: Hyper-parameters of the supervised learning models using the preprocessed Bitcoin dataset.

Model Used	**%Accuracy**	**%F_1-Score**	**%AUC**
Random Forest	98.02	82.39	91.9
ExtraTrees	97.34	80.34	92.4
Gradient Boosting	96.79	74.3	89.9
XGBoost	97.7	80.2	93.5
Logistic Regression	88.33	41.72	87.6
MLP	96.11	67.95	90.5

Table 3.2: Classification results of supervised learning models on the preprocessed Bitcoin dataset.

- SMOTE-SF: Number of selected features is set to 40.

Meanwhile, I distinguish between two ways of applying the edited nearest neighbours – undersampling technique – which I refer to each way as ENN and ENN-all, respectively. ENN corresponds to undersampling applied on the training set only, whereas ENN-all is applied to the whole dataset. The latter way allows us to understand the positions of the misclassified predictions in the feature space.

For the various resampling techniques under the random forest classifier, the experimental results of the using accuracy, precision, recall and f_1-score are tabulated in Table 3.3. In addition, I plot ROC-AUC curves to analyse the goodness of classification with the resampled dataset as shown in Figure 3.6. I compute the feature importance on the resampled datasets with ENN-all, NoSMOTE and Kmeans-SMOTE that are arbitrarily chosen. I compute the feature importance scores of the resampled dataset derived from ENN-all, SMOTE-SF and Kmeans-SMOTE resampling techniques that are arbitrarily chosen. I compare the most important features of the model obtained from a non-resampled dataset (i.e., represented by *'NoSMOTE'*) with models obtained from the latter three resampling techniques. The feature importance scores are computed for the train and test sets using the feature permutation method.

3.2.2.2 Evaluation and Comparison of Feature Importance

The feature permutation method [34] shuffles the data of each feature to amass the prediction error with respect to a baseline model (i.e., the model with a non-shuffled dataset). The overall process is repeated several times to find the average of the importance of each feature. In my experiments, I perform feature permutation, using sklearn package [166], with five repetitions to mitigate the biasedness caused by random shuffling. The feature importance on each of the train and test sets of the used datasets are depicted in Figures 8 and 9. As mentioned earlier, I arbitrarily choose three resampling techniques to study feature importance, however, the concept is viable with other resampling techniques and datasets. Moreover, I only visualise a set of six features (with the highest scores) since the visualisation of all features is quite large and non-informative. Moreover, I use the Wilcoxon signed-rank test [223] – a statistical method that tests the null hypothesis between two related paired samples derived from the same distribution. Using a paired sample test, the data can be expressed as:

$$(P(f_1), Q(f_1), ..., (P(f_n), Q(f_n)),$$

where n is the number of features, f_i is the feature at the i^{th}-dimension, $P(f_i)$ is the feature importance (i.e., feature score) of the feature f_i on the resampled dataset using a certain resampling technique, and $Q(f_i)$ is the importance of the feature f_i using the original dataset which I refer to by *'NoSMOTE'* as the baseline model.

The terms of the preceded expression can be replaced by the difference of scores as:

$$(D_1), ..., (D_n),$$

where

$$D_n = P(f_n) - Q(f_n) \tag{3.1}$$

Henceforth, the steps to perform the Wilcoxon test can be listed as follows:

1. Find $|D_1|$, ..., $|D_n|$, where $|.|$ is the absolute value notation.
2. Sort $|D_1|$, ..., $|D_n|$ in the increasing order.
3. Assign ranks to the sorted values in the step 2 as R_1,..., R_n, where R_i is the rank corresponding to $|D_i|$ at feature i. The ranks are assigned such that the smallest $|D_i|$ corresponds to rank 1, and the second smallest to rank 2 and so on.
4. Find the test statistic of the signed rank sum $\mathcal{T}$ as:

$$\mathcal{T} = \sum_{i=1}^{n} \text{sign}(D_i) R_i, \tag{3.2}$$

where sign(.) denotes the sign function that returns 1 if the input value is positive and -1 otherwise.

5. Find the p-value – i.e., probability value given that null hypothesis is true – by comparing the test statistic $\mathcal{T}$ to Student's t-distribution.

To test if the feature importance scores have changed after applying the resampling technique, I can formulate the hypothesis test as follows:

- Null Hypothesis

  H_0: Feature scores (before resampling) = Feature scores (after resampling).
- Alternative Hypothesis

  H_1: The scores of features are influenced by the resampling technique.

We then choose the values of α – the significance level – to be equal to 0.05. This value is an area in the t-distribution where I can reject the null hypothesis with a confidence level of 95%. Consequently, the p-value smaller than the significance level α means that I have strong evidence against the null hypothesis, and I can accept the alternative one which states that the importance of features is influenced by the resampled technique. For instance, I express the Wilcoxon test for the effect of the SMOTE resampling technique on the feature importance as follows:

$$\text{Wilcoxon(SMOTE, NoSMOTE)},$$

where $P(f)$ is derived from the feature importance after using SMOTE and $Q(f)$ is derived from the feature importance using the original dataset under the same model. We perform the Wilcoxon test for the resampling technique shown in Figure 3.7 using the Bitcoin dataset. The p-values of the Wilcoxon test are computed for the three chosen resampling techniques as tabulated in Table 3.5.

3.2.3 Discussions

3.2.3.1 Results of Classifications and Resampling Techniques

Referring to Table 3.3, ENN-all has outperformed all other resampling techniques as well as the non-sampled data (NoSMOTE) using the random forest classifier on the data derived from the Bitcoin. the experimental results with ENN-all have shown a remarkable increase in the accuracy and f_1-score, respectively, from 98.02% to 99.42% and 82.39% to 93.93% in comparison to NoSMOTE. This increase explains the high number of noisy instances that are removed by ENN-all to provide a a good decision boundary that perfectly fits the whole dataset. The remaining resampling techniques have revealed a trade-off between the number of false positives and false negatives, either in improving precision or recall as shown in Table 3.3. In comparison to NoSMOTE, ENN has boosted the precision from 97.96% to 99.34% but this comes at the cost of decreasing recall from 71.09% to 69.89%. This happens because ENN has removed noisy instances that are derived from the illicit transactions on the train set resulting in fewer false positives. Oversampling methods have played a remarkable role in

Resampling Techniques	**%Accuracy**	**%Precision**	**%Recall**	**%F_1-Score**
ENN-all	99.42	99.31	89.10	93.93
NoSMOTE (Original dataset)	98.02	97.96	71.09	82.39
Kmeans-SMOTE	98.02	97.96	71.09	82.39
LLE-SMOTE	98.02	97.96	71.09	82.39
DEAGO	98.02	97.96	71.09	82.33
ENN	98.01	99.34	69.89	82.05
AHC	97.96	96.38	71.37	82.01
CURE-SMOTE	97.95	96.96	70.72	81.79
Safe-level-SMOTE	97.96	97.93	70.17	81.76
OSCCD	97.82	95.34	69.89	80.66
SMOTE-SF	97.65	90.13	71.74	79.89
TRIM-SMOTE	97.66	90.72	71.37	79.89
SMOTE	97.57	88.87	71.56	79.28
SOMO	97.66	97.01	66.02	78.57
SMOTE-TomekLinks	97.41	86.14	71.74	78.28
Borderline-SMOTE1	97.25	84.00	71.28	77.12
Borderline-SMOTE2	97.35	88.12	68.51	77.09

Table 3.3: Comparison between resampling techniques applied to the preprocessed Bitcoin dataset. The performance of these techniques is evaluated using random forest classifier.

improving recall such as in the SMOTE-SF technique that attained a recall of value 71.74% wherein the train set is randomly oversampled on a particular subset of features.

Accordingly, oversampling is not able to reduce the misclassified instances, while still able to provide a better classification rule by improving AUC scores using different oversampling techniques as depicted in ROC-curve analysis in Figure 3.6. Normally, oversampling influences the model's performance when the generated data lies near the decision boundary of the used model. Another finding is that the random forest classifier after data preprocessing on the Elliptic data has outperformed the benchmark methods provided in the dataset's original work [221]. The random forest has attained an accuracy of 98.02% compared to 97.7% in [221] to classify the Bitcoin dataset using 85 features instead of 166 features as revealed in Table 3.4.

3.2.3.2 Results of Feature Importance

We discuss the influence of resampling techniques on the feature importance using the given supervised learning models. Mainly, the feature permutation method provides the highest scores for the most important features used by the classifier to perform predictions. Particularly, the feature permutation method with the test set is tied with the explainability of the model's predictions. Random forest reveals different feature importance on the train and test sets with different resampling methods referring to Figure 3.7. However, the feature '*lo*-

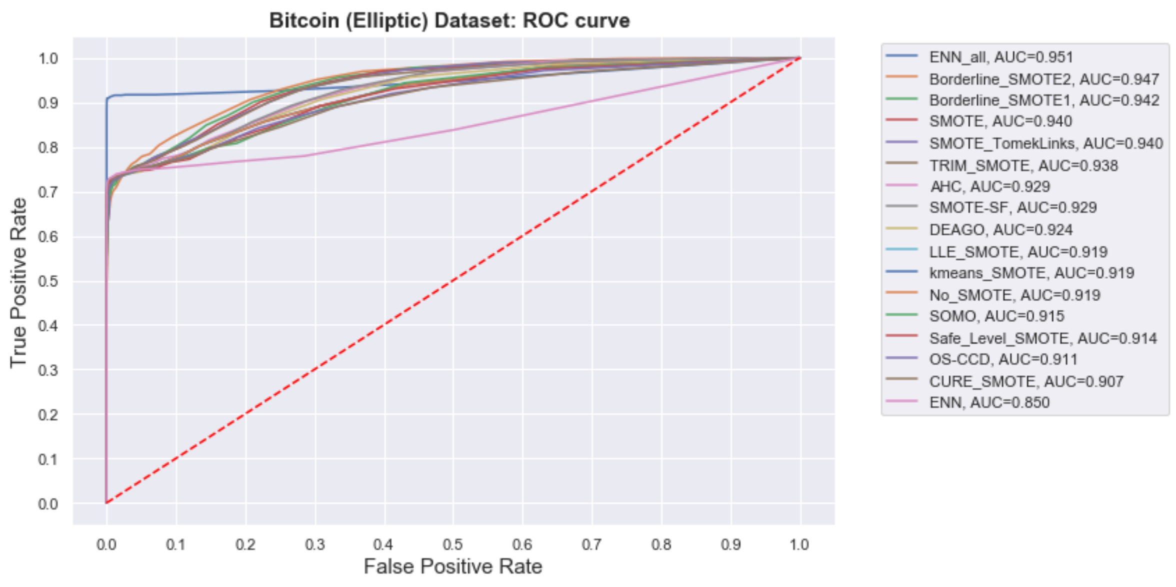
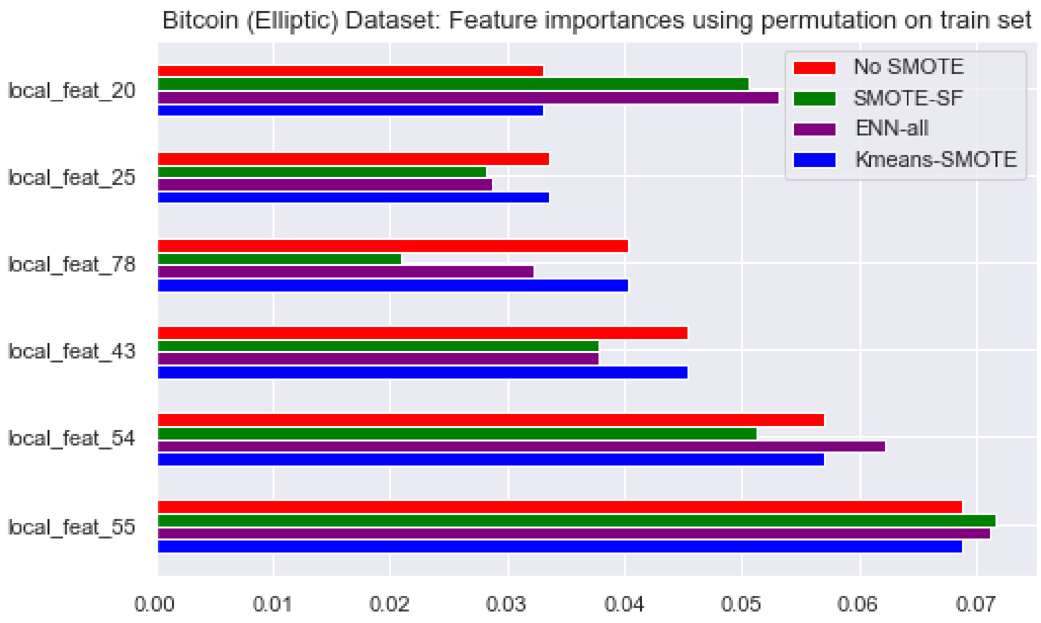


Figure 3.6: Experimental results of resampling methods: ROC-curve analysis using random forest classifier with various data resampling methods on the Bitcoin dataset.

cal_feat_55' has revealed the highest importance score which means that the latter feature plays an important role in providing decisions on the test set. I also notice that the local features on the Bitcoin dataset have appeared with higher importance than the aggregated ones.

3.2.3.3 Influence of Resampling Techniques on Feature Importance

As explainability of the models in this field is highly desirable, the change in feature importance caused by resampling methods affects the explainability of the model as it is tied with the feature importance. I verify this statement by performing the Wilcoxon test for the feature importance between the resampled and non-sampled datasets. This statistical method provides the p-values as revealed in Table 3.5. The p-values with less than the significance level of 0.05 show strong evidence to reject the null hypothesis and eventually accept the alternative one. For the Bitcoin test set, the Wilcoxon test for the resampling technique SMOTE-SF has revealed a p-value equal to 0.001 which means that the test is statistically significant. For the Bitcoin train set, there is no evidence to verify the influence of the given resampling techniques on the feature importance referring to Table 3.5. However, I still observe a difference in the scores of *'local_feat_20'* with the Bitcoin train set among the provided resampling techniques as depicted in Figure 3.7. In general, the difference in feature importance means that the data distribution is changed after applying the resampling methods using the same classification model. Furthermore, the model with the highest performance should produce more accurate feature importance and hence better explainability. This is reasonable because an explainable machine learning method seeks to interpret the predictions of a given model wherein these predictions are desired to be correct.



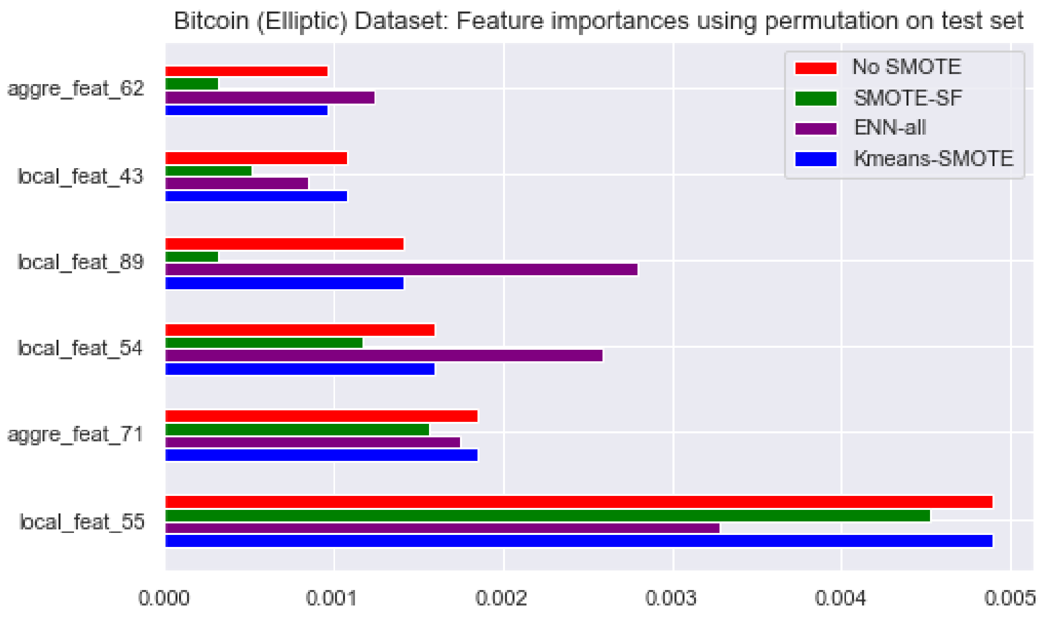


Figure 3.7: Effect of resampling techniques on feature importance in Bitcoin. The top figure belongs to the feature importance on the train set. The bottom figure belongs to feature importance on the test set.

Methods	%Accuracy	$\%F_1$-Score
Random Forest [221]	97.7	78.8
Data Preprocessing + Random Forest (Ours)	**98.02**	**82.39**

Table 3.4: Comparison between my experiments and the original contribution of the Elliptic (Bitcoin) dataset. This table highlights the effectiveness of the data preprocessing step in my experiments.

Model Used	Dataset	Wilcoxon Test	P-values
Random Forest	Bitcoin Train Set	Wilcoxon(SMOTE-SF, NoSMOTE)	0.995
		Wilcoxon(ENN-all, NoSMOTE)	0.354
		Wilcoxon(Kmeans-SMOTE, NoSMOTE)	0.999
	Bitcoin Test Set	Wilcoxon(SMOTE-SF, NoSMOTE)	0.001
		Wilcoxon(ENN-all, NoSMOTE)	0.227
		Wilcoxon(Kmeans-SMOTE, NoSMOTE)	0.999

Table 3.5: Wilcoxon test for the feature importance between resampled and non-sampled datasets of Bitcoin using different resampling techniques.

3.2.4 Concluding Remarks

Based on my conducted experiments, I have shown that random forest has performed the best in detecting illicit transactions in the Elliptic data which is derived from the Bitcoin. I have then studied the class-imbalance problem on the given dataset by applying various resampling techniques (oversampling, undersampling and hybrid resampling). ENN-all – an undersampling technique – has provided the best performance with an accuracy greater than 99%. Moreover, I have also provided the experimental results of other resampling techniques using accuracy, precision, recall, f_1-score and ROC-AUC curve analysis. As a result, oversampling techniques have been shown to improve the model's recall at the cost of its precision and vice-versa. Meanwhile, most oversampling methods have revealed a remarkable increase in AUC scores on the given datasets. We also claim the outperformance of the used models on the Elliptic data after data pre-processing in comparison to the results in their original contributions. On the other hand, I have also studied the effect of data resampling on feature importance. For that, I have utilised the feature permutation method to compute the feature importance on the train and test sets of the used dataset. The provided results have depicted changes in feature importance scores among different resampling techniques which influence the explainability of the model, where the model's explainability is more reliable with the high performing models. To show that resampling methods affect the feature importance, I have performed the Wilcoxon statistical method to test the statistical evidence to reject the null hypothesis which states that the feature importance scores remain the same before and after data sampling. Out of the three discussed resampling techniques, the SMOTE-SF technique has shown significant evidence with a 95% confidence level to state that the resampled data has affected the feature importance with respect to the non-resampled data. As for the study limitation, none of the oversampled data has shown better performance in terms of the model's accuracy where I cannot assert the effect of the resampling data on the feature importance with the outperforming models.

Transactions	Licit	Illicit	Unknown	Total
Train set	26,432	3,462	106,371	136,265
Test set	15,587	1,083	50,834	67,504
Total	42,019	4,545	157,205	203,769

Table 3.6: Dataset preparation summary of the Elliptic data.

3.3 Proposed Ensemble Learning Against Classical Supervised Learning Methods

In this study, my main focus is to find out the best performing classification model derived from the classical supervised learning methods for classifying data derived from the Bitcoin blockchain. For this purpose, I present a comparative analysis of the performance of classical supervised learning methods using the Elliptic data to predict licit and illicit transactions in the Bitcoin network. Besides, I present an ensemble learning classification method – a combination of the given supervised learning models – which outperforms the given classical methods. My main finding points out that the proposed ensemble learning method outperforms the classical supervised learning models in the original paper [221] of the same dataset. Thus, I show that the proposed model is able to predict licit/illicit transactions with an accuracy of 98.13% and f_1-score equals 83.36% using the proposed method.

3.3.1 Methods

3.3.1.1 Data Preparation

Using the Elliptic data, the 93 local features (excluding the timestep) concatenated with the aggregated features are used in my experiments as the input of the machine learning models. The total number of input features counts to 165 features which describe the dimensional feature space. Similarly, the train/test set split is divided using the first 34 timesteps as the train set and the remaining 15 timestamps are the test set. The distribution of the dataset between the train and test sets is summarised in Table 3.6.

3.3.1.2 Benchmark Methods

In this experiment, I utilise the supervised machine learning algorithms which are popular for the analysis of Bitcoin transaction data, referring to [85] and [221], as follows:

- Random Forest
- ExtraTrees
- Bagging

- Gradient Boosting
- AdaBoost
- K-Nearest Neighbours (KNN)

Logistic regression and support vector machine (SVM) are excluded from the current experiment as both algorithms do not perform well due to the highly imbalanced data. The reason is that the boundary decision is skewed toward the majority class (licit transactions) on one hand. On another, the minority class exists in the neighbourhood of the majority class, unlike the anomaly data points. Consequently, these algorithms are not suitable for the given dataset as they revealed a low performance.

Supervised learning methods in this experiment focus on anomaly detection tasks. The challenge here is to identify the criminals in a highly imbalanced dataset. In terms of machine learning, the aim is to achieve a good classification rule by reducing the false positives (licit transactions detected as illicit), without increasing the false negatives (illicit transactions detected as licit).

3.3.2 Experiments

In my experiments, I have used the scikit-learn package [166] in Python programming language to perform the classification of licit/illicit transactions of Elliptic data. I fit the variety of the supervised algorithms with the train set, while the test set is used to predict the performance of the model. The following hyper-parameters are empirically tuned as follows: random forest algorithm (where *n_ estimators*=100, *Bootstrap*=False, *min_ samples_ leaf*=2, *max_ depth*=50), ExtraTrees (using the same settings as for the random forest), Gradient Boosting (using *learning_ rate*=0.01, *min_ samples_ leaf*=2), AdaBoost algorithm and Bagging classifier (both classifiers using random forest model as the base estimator). Meanwhile, I apply K-Nearest Neighbour (K-NN) algorithm after choosing K=8 as the optimal value tuned in the range of K$\in$ [1, 26]. Besides the mentioned classifiers, I present ensemble learning as the combination of the three best-performing methods on the Elliptic data. Ensemble learning is defined as a classification learning model derived from combining a variety of machine learning algorithms, to enhance the performance of the final predictions [194]. Ensemble learning has been widely investigated in previous studies for its capability of achieving higher accuracy. This is due to the predictions from several learning models that jointly contribute to the final classifications.

In my experiments, the ensemble learning classifier is based on the so-called average probability ensemble as in [115]. In an average probability ensemble, the classification is performed by using several pre-trained machine learning models, in which the final predictions are derived from averaging the sum of the probability predictions obtained from each of the learning algorithms. In my experiments, the ensemble learning is based on the combination of the following methods: random forest, ExtraTrees and bagging classifiers. Each of these models provides output predictions as probability values that demonstrate the confidence of the algorithms in

Model	**%Acc.**	**%Prec.**	**%Rec.**	**%F_1-Score**	**%G-Score**	**FP**	**FN**
Ensemble Learning	**98.13**	**99.11**	**71.93**	**83.36**	**84.79**	**7**	**304**
Random Forest	98.06	97.38	72.2	82.92	84.91	21	301
ExtraTrees	98.01	98.70	70.36	82.15	83.85	10	321
Bagging	98.01	96.41	72.11	82.51	84.84	29	302
AdaBoost	97.99	96.28	71.83	82.28	84.67	30	305
Gradient Boosting	97.35	99.84	59.37	74.46	77.05	1	440
KNN	95.1	61.6	63.99	62.77	78.87	432	390

Table 3.7: Evaluation of machine learning metrics: Experimental results of the supervised learning models using the Elliptic data. 'Acc.' is the short for accuracy, 'Prec.' for precision, 'Rec.' for recall, 'G-score' for geometric score.

Model	**%Accuracy**	**%Precision**	**%Recall**	**%F_1-score**
Logistic Regression [221]	93.1	40.4	59.3	48.1
Multi-Layer Perceptron [221]	96.2	69.4	61.7	65.3
Random Forest [221]	97.7	95.6	67	78.8
Ensemble Learning(our results)	**98.13**	**99.11**	**71.93**	**83.36**
Random Forest (our results)	**98.06**	**97.38**	**72.2**	**82.92**

Table 3.8: Comparative results between original research in [221] and ours using different supervised learning methods on Elliptic data

labelling the given input vectors. Admittedly, ensemble learning based on the average probability ensemble has outperformed all the classical models used from the benchmark methods. Using Elliptic data, I fit the mentioned models, after tuning the model hyper-parameters. I present the evaluated results derived from the classification metrics such as accuracy, precision, recall, f_1-score, as well as the number of false positives and false negatives as provided in Table 3.7. Furthermore, I present the receiver operation curve (ROC) to roughly reveal the performance of the used supervised methods, as well as computing the area-under-curve (AUC) of each model as depicted in Figure 3.8.

3.3.3 Discussions

The proposed method of ensemble learning has performed the best in comparison to the other supervised learning methods to classify the Elliptic data as provided in Table 2. My results show that ensemble learning is able to perform classification with an accuracy of 98.13% and f_1-score 83.36% to predict licit/illicit transactions. Moreover, ensemble learning also outperforms the results provided in the dataset's original work [221], as shown in Table 3.8. Meanwhile,

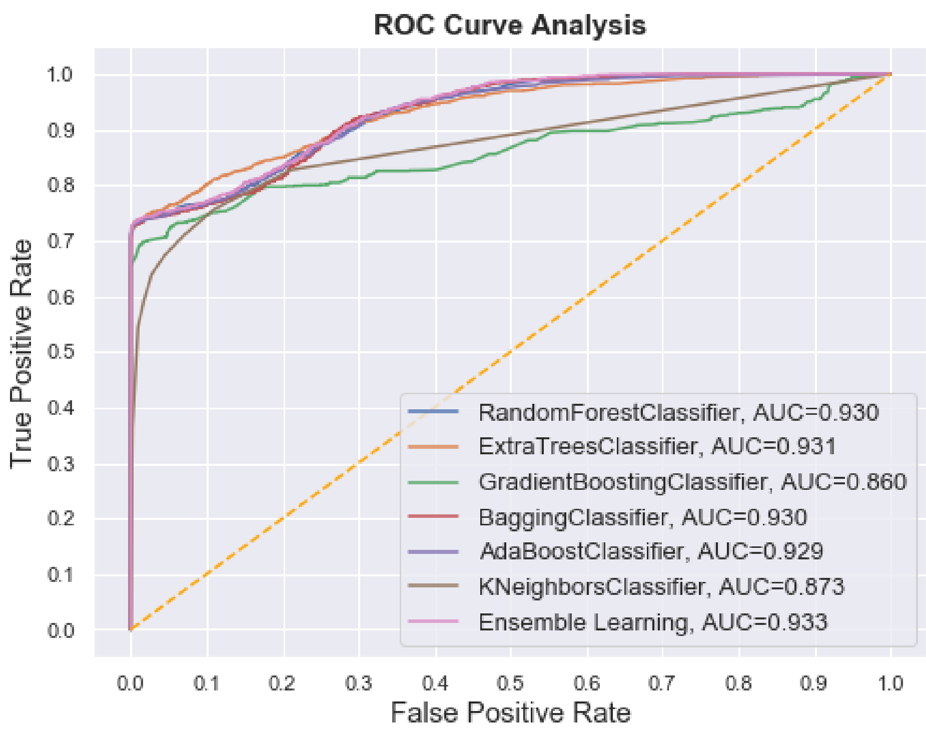


Figure 3.8: ROC-curves of the supervised learning models trained on Elliptic data. Area-Under-Curve is denoted by AUC and the bisector straight line represents the random guess line.

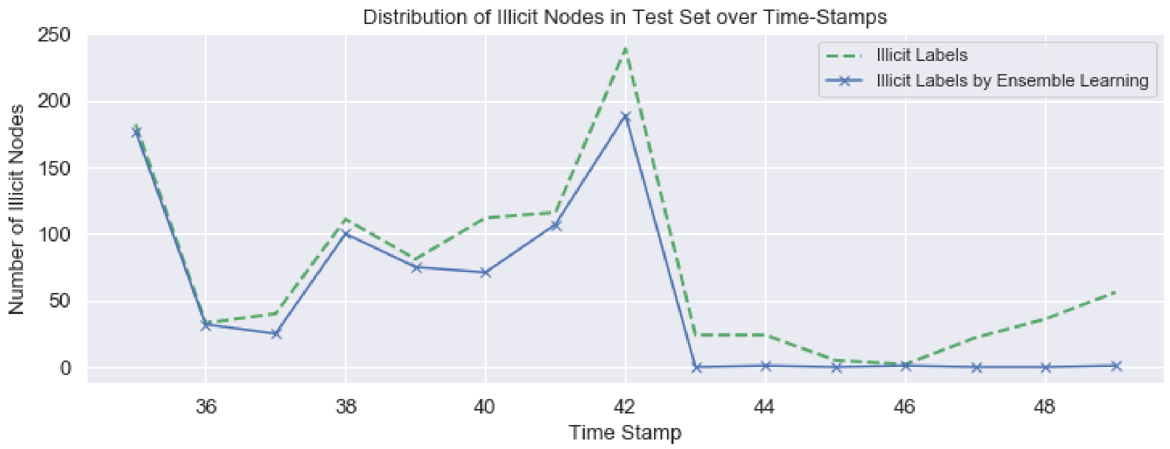


Figure 3.9: Illicit transactions in test set of true labels and their associated predictions using Ensemble Learning.

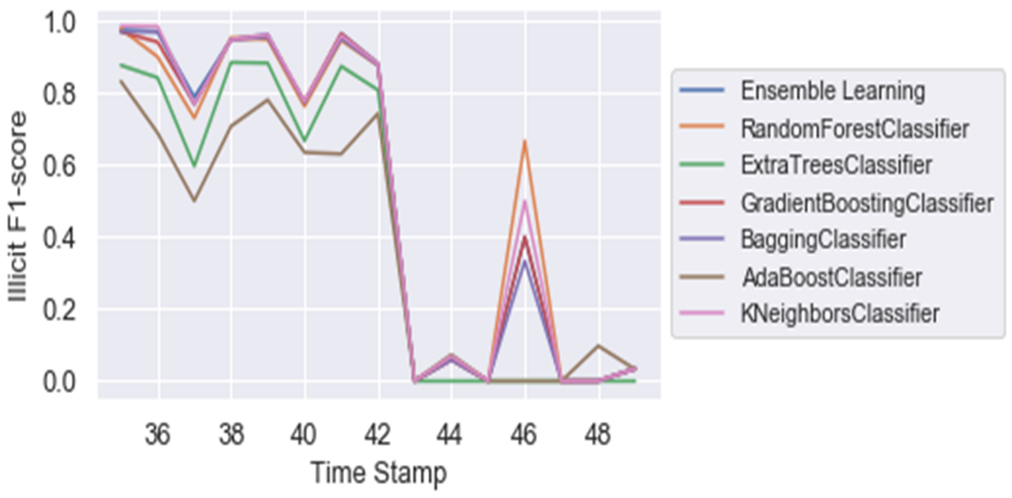


Figure 3.10: Performance of supervised learning methods on each timestep using Elliptic data. F_1-score is computed for illicit instances.

supervised learning algorithms based on decision trees such as random forest and ExtraTrees methods have revealed remarkable performance which interprets the appropriateness of the used methods on this dataset. K-NN algorithm performs poorly among the examined models admitting the least performance in this task with an accuracy equal to 95.1%. Since K-NN is based on Euclidean distances, it is computationally expensive to search for the best values of K. On the other hand, the two drawbacks of K-NN in my case are the high dimensional space of this dataset and its highly imbalance-class issue. For example, K-NN relies on the k-neighbours in the feature space to vote for the best class [238]. Due to the existence of numerous negative instances in a neighbourhood of a small number of positive instances, the voting mechanism in K-NN is more likely to be skewed toward the majority.

With the Bitcoin transaction dataset, random forest performs well in this task as it uses a voting mechanism to aggregate the prediction results from a certain number of decision trees. Using ensemble learning, the combination of random forest, ExtraTrees, and Bagging classifier revealed a potential performance by acquiring the predictions based on averaging the probabilities obtained from these algorithms. Referring to Table 2, ensemble learning has accomplished the minimum number of false positives equal to 7, thus increasing the precision without significant variation of recall. For instance, false positive instances might appear commonly between different learning algorithms. These instances will indeed remain the same after using ensemble learning. In contrast, I desire to acquire different classification models, in which each model will fail on classifying correctly certain data points that are distinct in each. Thus, ensemble learning will try to adjust the predicted probabilities by combining several models, so that I can reduce the number of false instances. For the time-complexity, Bagging Classifier is less computationally efficient than random forest and ExtraTrees, because Bagging here is an ensemble that utilises random forest as a base estimator. Therefore, I associate the time-complexity of ensemble learning method as the complexity of Bagging Classifier, since the latter one describes the worst-case scenario. To do this, let n = training instances, p = number of features, $n_{trees} = n\_estimators$, $d = max\_depth$, n_{voters} = number of constructed voting samples of Bagging Classifier which is set to 10. Thus, the time complexity of the ensemble learning, assuming parallel processing of the models, can be expressed as $O(3d.n.\sqrt{p}.n_{trees}.n_{voters})$ which is less computationally efficient than the classical supervised learning methods.

The maximum AUC is recorded for ensemble learning which is equal to 0.933, outperforming the other models as shown in 3.8. Apparently, the ROC curve of the Gradient Boosting Classifier has shown the worst performance among other supervised models. The AUC ratio of the latter algorithm is equal to 0.86 which is lower than the K-NN associated with AUC of 0.873. As shown in Figure 3.9, the number of illicit transactions at every timestep is plotted for the true labels and the true prediction at these data points of an ensemble learning algorithm. Ensemble learning has revealed good discrimination of illicit transactions until 39th timestep. In the range of 40-42 timestep, the number of actual illicit transactions has admitted a rapid increase, which after has decreased sharply at 43th timestep. This area shows the highest difference between the actual and the predicted labels where the dark market shutdown occurred referring to [221]. In addition, f_1-scores are plotted for Elliptic data derived from the performance of the used supervised learning methods as shown in Figure 3.10. The f_1-scores demonstrate the performance of supervised methods regarding the illicit class, which is a matter of interest. Not only the performance of ensemble learning has performed poorly during the dark market shutdown, but also all other supervised methods used in my experiment. At this remarked event, none of the provided learning methods is able to detect illicit transactions. The reason for this degraded performance is due to the occurrence of an event that the algorithm has not learned before. As known, the Bitcoin graph network of Elliptic data is a subgraph derived from the transaction graph of the Bitcoin blockchain. The other reason that the models are performing poorly on some timesteps might be a result of the loss of structural information derived by the formation of the sub-graph. It is more likely to lose some important links and patterns that are necessary for training the model. Moreover, the nodes with the unknown labels are disregarded in my study where there might be some interesting features and node structures to enhance the detection of the model. Regarding the high difference between positive and negative instances, it is always desired to have a balanced data set. However, resampling methods such as undersampling and oversampling techniques are shown in the previous part to be not effective for this dataset. They could decrease the performance or add nothing.

3.3.4 Concluding Remarks

For AML compliance in Bitcoin, I have done a comparative analysis of Elliptic data to spot illicit transactions using different supervised learning methods. I have shown that the combination of supervised learning methods known as ensemble learning outperformed all other methods using local features and aggregated features derived from Bitcoin transaction graphs. As a result, the classification results have shown a reduced number of false positives without increasing the false negatives. The results have shown a noticeable improvement in comparison to the classification results provided in the dataset's original contribution.

3.4 Graph Convolutional Network For Anti-Money Laundering in Bitcoin

Classical supervised learning methods have been widely applied on Bitcoin datasets which provided promising results such as in [85, 12]. However, the classical learning algorithms learn directly from the raw data without considering any structural topology. The graph-structured data of Bitcoin has inherently motivated the exploration of graph-based learning approach to perform predictions. Referring to Chapter 2, graph networks have been extensively used as an essential framework to analyse the interconnections between transactions and capture illicit behaviour in the Bitcoin blockchain. Due to the complexity of the Bitcoin transaction graph, the prediction of illicit transactions has become a challenging problem to unveil illicit services over the network. Graph Convolutional Network (GCN) – a graph neural network-based spectral approach – has recently emerged and gained much attention regarding graph-structured data. Weber et al. [221] have tested the performance of the GCN model, a neural network that operates on graph-structured data, using the Elliptic data to predict illicit transactions. Consequently, their experimental results have highlighted the success of the random forest classifier over the GCN model.

3.4.1 Motivation

GCN is a neural network that operates on the local graph neighbourhoods involving a learnable kernel to produce the node embeddings of the relevant graph network [106]. However, the GCN model applied to the Elliptic data in [221] was incapable of encoding the graph data into useful node embeddings where the embeddings are used to perform predictions. This drawback has induced the motivation to investigate the performance of GCN when backed up with linear layers. Another reason is that the graph-structured data of the Elliptic data has ardently led to investigate graph convolutional layers which exploit the structural topology of the graph input promoted by node features. Thus, a novel model, based on the combination of GCN with linear layers, is proposed to efficiently predict illicit transactions in the Elliptic Bitcoin dataset. Inspired by [94], the proposed model is based on the concatenation of two sets of features; the first set is the node embeddings derived from GCN embeddings of the graph network with the local features, whereas the second set is obtained from the latent representation of a linearly transformed hidden layer from the local features of the Elliptic data. This concatenation forms new latent features which are squashed by non-linear function and followed by the MLP model. Concisely, the proposed approach is motivated by a graph-based spectral approach and leveraged with the latent representation of local features that represent the Euclidean proximity. The proposed approach serves as an assistant framework to spot illicit transactions by performing node classification of the Bitcoin transaction graph. My main finding is that the combination of GCN and linear layer features performs better in comparison to GCN models used in [221].

3.4.2 Methods

3.4.2.1 Graph Convolutional Network (GCN)

Firstly, I describe the existing GCN model introduced in [106]. I refer the reader to [239] for a comprehensive review of different graph neural network versions. As mentioned earlier, GCNs are neural networks that are fed with graph-structured data in which this model considers the neighbouring features of each node besides its features.

In [106], GCN has shown to be an efficient algorithm for node classification tasks which is motivated via localised first-order approximation of spectral graph convolutions. The embedding matrices in GCN are considered the induced features of the nodes, and they depend on the number of convolutional layers used. Referring to [106], a neural network formed by GCNs is a stack of multiple GCN layers and each layer is followed by a point-wise non-linearity, where the layer-wise convolution is limited to 1-hop aggregation. Using one convolutional layer, GCN aggregates information from the immediate neighbours of the node of interest. If one builds convolutional layers on top of each other, this algorithm can capture up to k-hops away from the central node, where k is the number of the stacked GCN layers.

More formally, consider the Bitcoin transaction graph from Elliptic data as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are sets of nodes (Bitcoin transactions) and edges (payments flow) respectively, where $|\mathcal{V}|$ is the number of transactions or the graph network size. Let $\mathcal{A}$ be the adjacency matrix of the transaction graph network, $\mathcal{H}^{(l)}$ be the node embedding matrix of the l^{th} layer as input and consider $\mathcal{W}^{(l)}$ as a trainable weight matrix used to update the embedding matrix to $\mathcal{H}^{(l+1)}$ as output. Referring to [106], a multi-layer GCN is described with the following layer-wise propagation rule:

$$\mathcal{H}^{(l+1)} = \sigma(\hat{\mathcal{A}}\mathcal{H}^{(l)}\mathcal{W}^{(l)}), \tag{3.3}$$

where $\hat{A}$ is the normalization of A defined by:

$$\hat{\mathcal{A}} = \tilde{D}^{-\frac{1}{2}}\tilde{\mathcal{A}}\tilde{D}^{-\frac{1}{2}}, \quad \tilde{\mathcal{A}} = \mathcal{A} + I, \quad \tilde{D} = \mathrm{diag}(\textstyle\sum_j \tilde{\mathcal{A}}_{ij})$$

$\tilde{\mathcal{A}}$ is the adjacency matrix of the graph $\mathcal{G}$ with the added self loops. σ denotes the typical activation function such as $ReLU(.) = max(0, .)$. $\mathcal{H}^{(l)}$ is the activation matrix and known as node embedding matrix. The first embedding matrix is derived from the node features which is denoted by $\mathcal{H}^{(0)} = \mathcal{X}$.

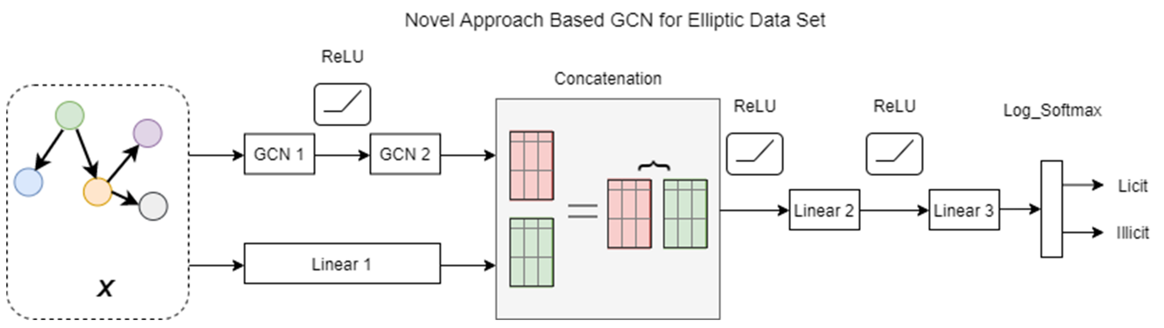


Figure 3.11: Architecture of the proposed method. $\boldsymbol{X}$ represents the node feature matrix accompanied by a graph network derived from Bitcoin. The output represents the predictions of licit/illicit transactions.

Regarding 2-hop neighbouring aggregation of features, a 2-layer GCN is used and it is often expressed by:

$$\mathcal{H}^{(2)} = \text{softmax}(\hat{\mathcal{A}}.\text{ReLU}(\hat{\mathcal{A}}\mathcal{X}\mathcal{W}^{(0)}).\mathcal{W}^{(1)})$$

where $\mathcal{W}^{(0)})$ and $\mathcal{W}^{(1)}$ are learnable matrices trained using gradient descent, and softmax function is defined as: softmax(x) $= \frac{1}{\mathcal{Z}} \exp(x_i)$, where $\mathcal{Z} = \sum_i \exp{(x_i)}$.

Model	**%Accuracy**	**%Precision**	**%Recall**	**%F_1-score**
GCN [221]	96.1	81.2	51.2	62.8
Skip-GCN [221]	96.1	81.2	62.3	62.8
GCN-based (Ours)	**97.4**	**89.9**	**67.8**	**77.3**

Table 3.9: Comparison of results between the original work in [221] and the proposed method using Elliptic data.

3.4.2.2 Proposed Method

The stated GCN inherently operates on undirected graphs, whilst Bitcoin graphs are directed. In other words, the stated model works more efficiently with symmetric normalised Laplacian. Thus, we refer to [184] wherein a different approach of GCN is introduced known as Relational-GCN (R-GCN). Briefly, R-GCN uses the aggregation of the transformed feature vectors of local neighbouring nodes through a normalised sum. Motivated by the normalised constant as stated in the latter reference, I have used a modified version of GCN which empirically works better with directed graphs. Thus, the modified GCN differs from the regular one by using the so-called random walk normalisation.The normalised adjacency matrix is modified into: $\hat{\mathcal{A}} = \tilde{D}^{-1}\tilde{\mathcal{A}}$. In this experiment, I refer to the GCN as the modified version. The proposed method is based on GCN using graph convolutional layers and accompanied by linear layers. First, the model consists of 2-layers GCN as represented in Figure 3.11. The output of the last layer is concatenated with the output of a linear layer having the original node features as input. The overall output is then squashed with the ReLU activation function, and forwarded into two consecutive linear layers. Subsequently, the second linear layer is squashed with the ReLU function and the last layer outputs are transformed with *log_ softmax* function to produce the log of the prediction probabilities of the different classes as depicted in Figure 3.11. The given architecture is motivated by the GCN model, in which the convolutional layer is based on the local neighbourhood aggregations and provided with self-loops to include the features of the given node. Likewise, the idea here is to supposedly ensure that the features accompanied by nodes are reproduced in the following layers. The proposed method can be viewed as leveraging intuitively two sub-models; the first model is a graph-based spectral approach and the second is based on a linearly transformed feature matrix in the Euclidean domain. In the proposed model, the idea of concatenation encourages the reuse of the latent features to maintain the maximum flow of information between the layers. The input of GCN is given by the Bitcoin transaction graph which is accompanied by node feature matrix $\mathcal{X}$, while the input of the linear layer is solely provided by $\mathcal{X}$.

3.4.3 Experiments

To train the proposed model, I use PyG – Pytorch Geometric package – in Python programming language [63]. My aim is to perform node classification of licit/illicit transactions of Elliptic data. In this experiment, the used features express the local information of the used data excluding the timestep feature, resulting in 93 features. The hyper-parameters of the neural network are empirically tuned to achieve the highest accuracy on the validation set, after fixing the number of epochs to 50. During each epoch, the model is trained in a graph-wise way referring to Figure 3.12; gradient descent is used to minimize the loss, whereas each of the 34 graphs (train set) is fed to the model to update its parameters. I use the Adam optimizer to train the model with a learning rate of 0.001 and a weight decay of $5x10^{-4}$. The sizes of the first and second convolutional layers are set to 50 and 10 respectively. Moreover, a dropout layer is applied to the former convolutional layer with a probability equal to 0.5 to avoid overfitting. Regarding the linear layers, the sizes of the first and second are

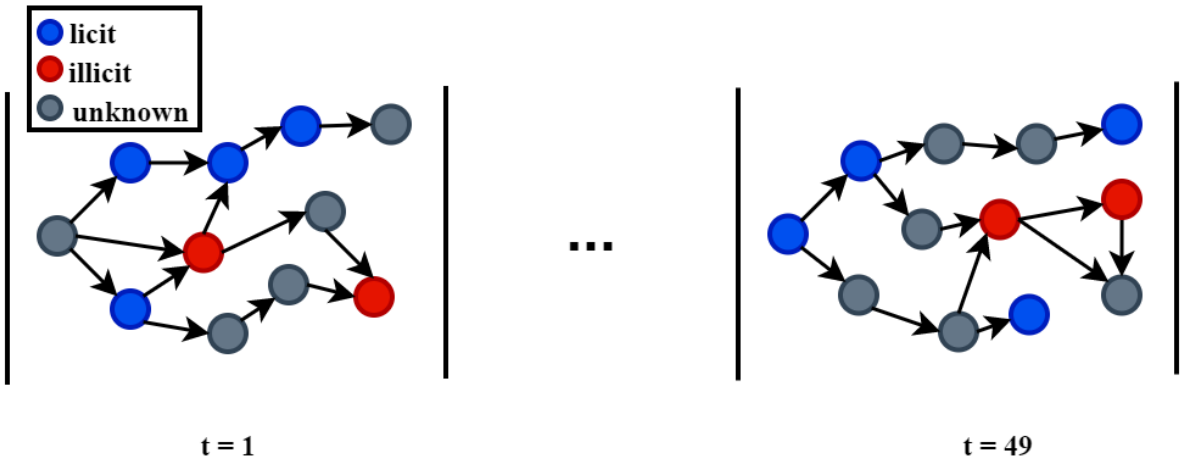


Figure 3.12: Representative structure of Elliptic data where t represents the corresponding timestep of the graph.

set to 100 and 81 respectively. Subsequently, the output is squashed with $log\_softmax(.)= \log(softmax(.))$, resulting in two output values that correspond to the licit and illicit classes respectively. We trained the model using a weighted Negative Likelihood Loss, in which I opt for 0.3/0.7 weights for the licit and illicit classes to include more innocent transactions. Table 3.9 reveals the evaluation of the proposed model in terms of precision, recall, f_1-score and accuracy of the test set. Eventually, the proposed GCN model outperforms the GCN models used in the original work of this dataset.

3.4.4 Discussions

In this experiment, the proposed GCN assisted with linear layers has significantly achieved adequate results. Admittedly, it outperforms the results achieved in the previous work in [221] using the Bitcoin transaction graph of the Elliptic data. The reuse of the latent features with GCN has efficiently enhanced the predictions, rather than using only multiple convolutional layers. The typical GCN based spectral approach can be viewed as an aggregation of the neighbouring node features through a normalised sum. The aggregated nodes might not contribute fairly to the node of interest through the weights associated with the normalised Laplacian. For this reason, the output signal given by the GCN model might be distorted or modified. Another reason is that the GCN spectral approach is originally an appropriate approach for undirected graphs, albeit I use the modified version. Thus, the proposed method maintains the latent features of the input matrix at the output of the convolutional layers where the output of the latter layers is supposedly subjected to unfair weights regarding the normalisation factor. Besides, I have checked the performance of the proposed model without using GCN layers (only linear layers) and under the same conditions to highlight the competence of GCN in the same context. Hence, the proposed model with graph convolutional layers has surpassed a similar model without GCN as depicted in Figure 3.13. Consequently, this comparison ensures the importance of utilising a concatenation between the GCN and

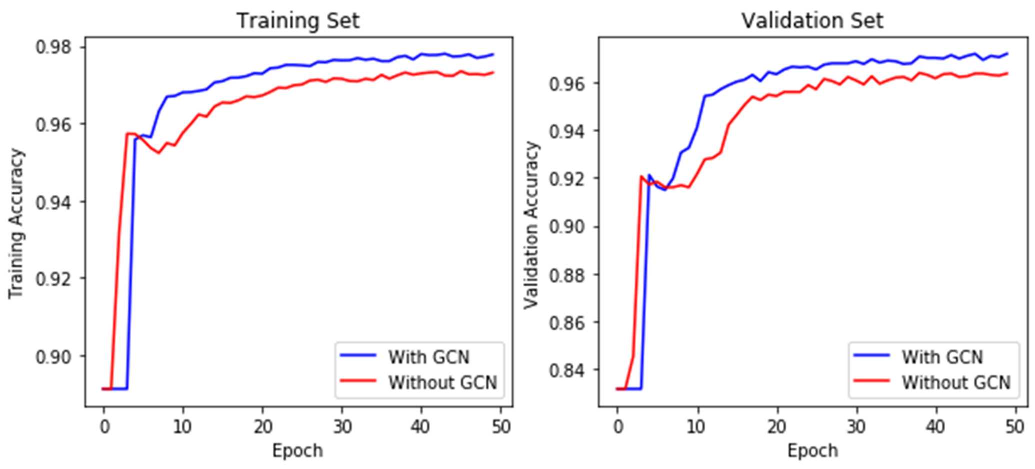


Figure 3.13: Comparison of two models. "With GCN" indicates the proposed method. "Without GCN" denotes the proposed method after removing GCN layers. Training/Validation accuracy is depicted on the left/right figure.

the linear layer, in which a higher performance is obtained. From the perspective of the linear layers, the features formed by GCN have provided useful information to the following layers. The timestep feature is excluded in my experiment because it is not very informative which represents the timestep when the transactions of every graph network were extracted. The real-time associated with the transactions might be more useful for learning. For instance, criminals might appear at a certain time, in which a significant pattern is processed by the Bitcoin blockchain. Henceforth, the real-time associated with transactions might be a good idea as an additional feature in the GCN model.

3.4.5 Concluding Remarks

We present a novel model that utilises the GCN model to predict illicit transactions in the Bitcoin transaction graph. The proposed method highlights the competence of GCN when combined with Multi- Layer Perceptron that is consolidating the graph-based spectral approach with a feedforward neural network. The experimental evaluations demonstrate that the concatenation of features derived from GCN with the latent representation of a linear layer improves the classification results rather than merely applying graph convolutions. My proposed method outperforms the GCN models used in the original paper [221] on the same dataset.

3.5 Summary

To wrap up, this chapter has comprehensively explored several aspects of graph-structured data derived from the Bitcoin blockchain named Elliptic data.

We have pointed out the data preprocessing and resampling techniques to address the high dimensionality of this dataset and the class-imbalance issue. Thus, I carry out the study on the Elliptic data after reducing the number of features and comprehensively applying a variety of resampling techniques. Eventually, resampling techniques tend to change the original class distribution of the dataset. Thus, I study the effect of the resampling techniques on the feature importance which is a part of XAI methods. Consequently, I highlight the influence of data sampling on the explainability of the machine learning model by performing a hypothesis test to have evidence for accepting my hypothesis. The limitation of this study is that none of the resampling techniques has revealed improved results in terms of accuracy but not in regards to the precision, recall or AUC scores.

In the subsequent study, I perform a comparative analysis using various supervised learning methods to achieve the state-of-the-art performance on the Elliptic dataset by proposing an ensemble learning method. In this experiment, I have made use of the original features of the Elliptic data without any feature reduction step to ensure better classification results. Ensemble learning-based averaging probability predictions of three bagging classifiers have achieved the highest accuracy followed by the random forest as the second-best performing model. The limitation of this model is the high computational complexity of the proposed ensemble learning algorithm exceeding the time complexity of the random forest model.

As the Elliptic data comprises graph networks, I propose a GCN model in a novel way that is combined with linear layers to classify the transactions of this dataset. The GCN model exploits the graph structure information of the given dataset, unlike the classical supervised learning methods. In this study, I highlight the competence of using GCN layers backed up with linear layers to provide adequate classification results. However, the experimental results of the classical supervised learning method outperform the models based on GCN layers.

This dataset is also included in Chapter 5 to capture and evaluate the model uncertainty of the machine learning models. Yet, the temporal information of this dataset is not covered in this chapter which will be addressed in Chapter 6.

Chapter 4

Supervised Learning for Fraudulent Accounts in Ethereum

Contents

To carry out studies on another dataset derived from the public blockchain, I utilise the Ethereum account dataset – a tabular dataset of fraudulent accounts derived from the Ethereum blockchain. Here, I apply supervised learning to classify the fraudulent accounts of this dataset. In this chapter, I follow the process that I have done in the experiments using the Elliptic dataset of the Bitcoin blockchain. I note that the Ethereum dataset is not an example of graph-structured data. Therefore, this dataset is not included in the studies where I apply graph learning algorithms in Chapter 5.

The current chapter can be roughly categorised into the following main points:

- I provide a detailed presentation of the Ethereum account dataset which I use in our experiments.
- I perform feature reduction and data preprocessing, similar to Chapter 3.

- I evaluate the results and highlight our main findings where I study the effect of re-sampling techniques on the feature importance and the explainability of the machine learning model using the Ethereum dataset.

4.1 Ethereum Dataset Description: Ethereum Account Data

The Ethereum account dataset[1] comprises known fraud accounts and valid transaction history over the Ethereum blockchain extracted by a combination of two sources; a local Geth client and the Etherscamdb linked to the Ethereum network for normal and scam accounts, respectively. This dataset is introduced by Farrugia et al. in [61]. The accounts are labelled by the Ethereum community for illicit behaviour in several cases e.g., scams, Ponzi schemes and phishing. This dataset involves 9841 labelled accounts distributed as non-fraud/fraud in Figure 4.1 associated with 49 numerical and categorical features e.g., "*total number of sent/received transactions*" and "*average value of ether ever sent*".

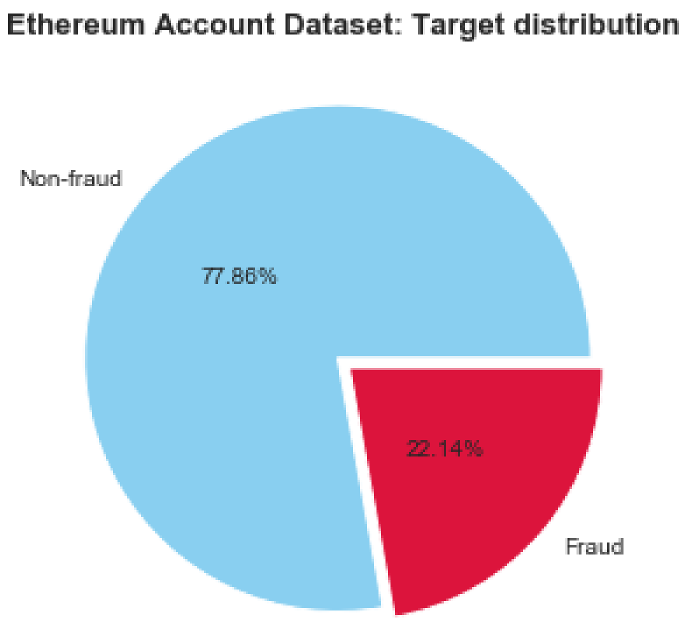


Figure 4.1: Non-fraud/fraud distribution of Ethereum account data.

4.1.1 Data Preprocessing

The main challenges of this dataset encompass the lack of neighbouring nodes between the Ethereum accounts, the lack of temporal information, and the missing values occurring in the

[1]https://github.com/sfarrugia15/Ethereum_Fraud_Detection/blob/master/Account_Stats/Complete.csv

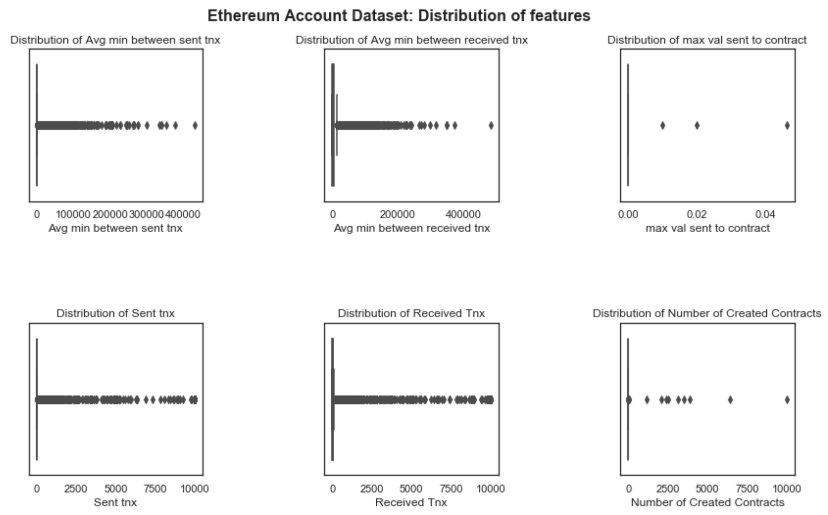


Figure 4.2: Boxplot of some features in Ethereum account data. This indicates how these features in Ethereum dataset are spread out.

categorical features. Therefore, this dataset is used without any graph or temporal information as well as the categorical features are disregarded in our experiments. Further feature reduction is applied by removing the linearly correlated features whose correlations are greater than 0.9, arbitrarily chosen, as well as features with zero variance. Moreover, another feature reduction is done by removing the features with unique numerical values of less than 10 values as in the case of the feature "Distribution of max val sent to contract" depicted in Figure 4.2. This further reduces the number of features to 28 dimensions as shown in Figure 4.3. After this step, I split the data set randomly after fixing the seed to zero with a 70/30 split for the train and test sets respectively. A validation set as 10% of the train set is chosen to tune the hyper-parameters.

4.2 Methods

4.2.1 Benchmark Methods

As in the experimental procedure in Section 3.2 in Chapter 3, I classify the Ethereum dataset using various classical supervised learning algorithms as follows:

- Random Forest
- ExtraTrees
- Gradient Boosting
- XGBoost

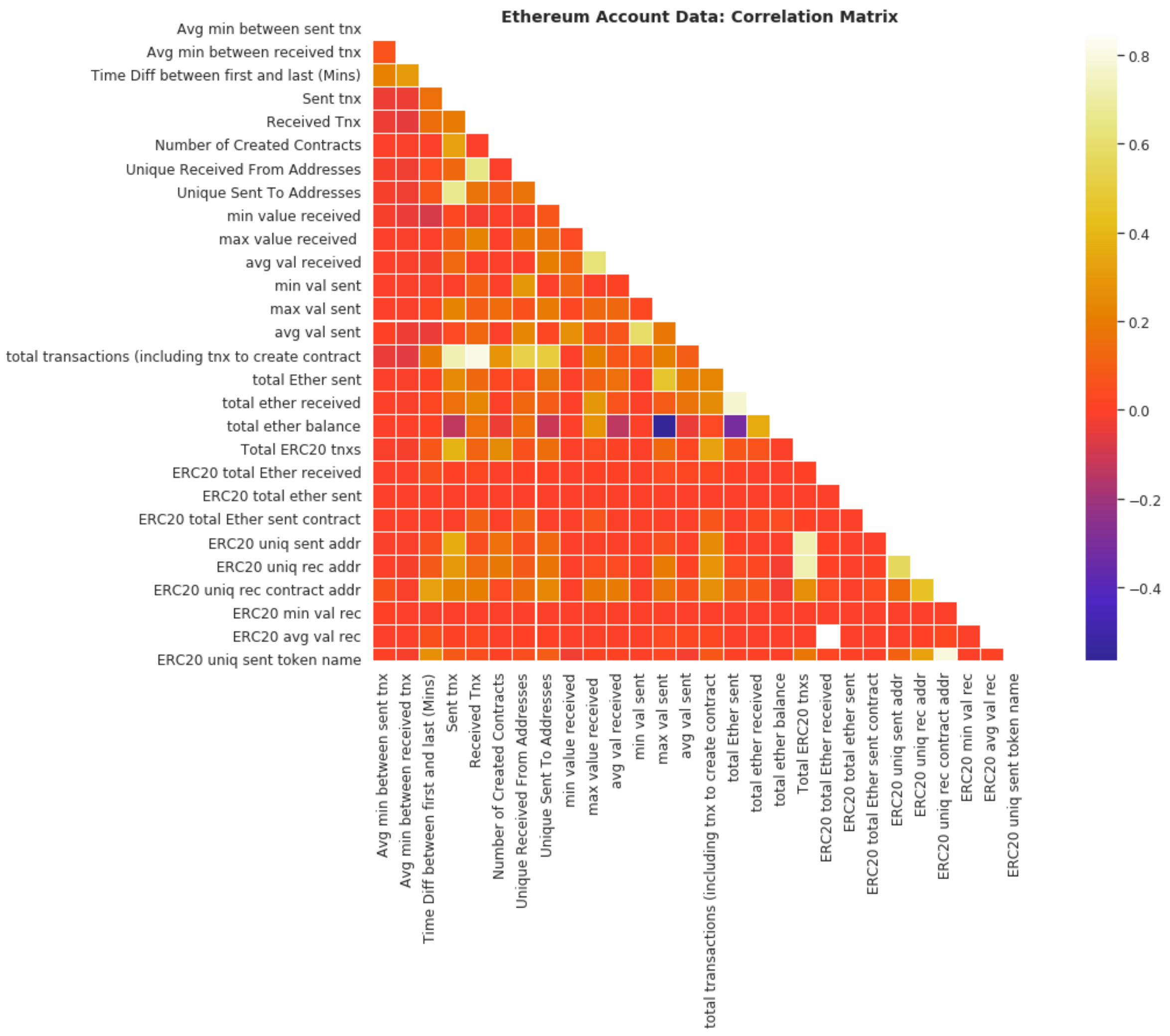


Figure 4.3: Correlation matrix of Ethereum account data.

- Logistic Regression
- Multi-Layer Perceptron (MLP)

4.2.2 Resampling Methods

We have conducted a comprehensive study to address the class imbalance of this dataset using a variety of more than 80 resampling techniques that are listed in Appendix 7.4. However, I only present the resampling techniques that revealed the highest performance with this dataset. The experimental results of resampling techniques are only evaluated using the best-performing supervised learning method on the original dataset. Subsequently, I resample this dataset using the following set of oversampling methods: SMOTE, Kmeans-SMOTE, AHC, Borderline-SMOTE1, Borderline-SMOTE2, SOMO, SMOTE-TomekLinks, DEAGO, Safe-level-SMOTE, TRIM-SMOTE, CURE-SMOTE, LLE, SMOTE-SF and OSCCD. For the undersampling methods, I present the results derived from the ENN method which removes the noisy data points between the different class distributions. I refer to ENN by the undersampling applied to the train set and ENN-all to the undersampling applied to the whole dataset. Regarding the hyper-parameters of the resampling techniques, I use the default hyper-parameters for all oversampling methods except for the following techniques that are tuned among a set of arbitrarily chosen values:

- OSCCD: the number of clusters is set to 3 among the values {2,3,4,5,6}.
- LLE: the number of features of the embedded space is set to 5 among the values {2,3,4,5,6}.
- SMOTE-SF: This method performs resampling on a subset of features where the number of selected features is arbitrarily set to 10.

4.3 Experiments

4.3.1 Experimental Settings

In our experiments, I use sklearn [166] and smote-variant [110] in Python programming language. I apply supervised learning algorithms from the benchmark methods to classify fraudulent accounts. The hyper-parameters of these methods are empirically tuned on the validation set and provided in Table 4.1. The supervised learning models are evaluated using classification metrics which are accuracy, f_1-score, and AUC-ROC curve analysis as provided in Table 4.2. Since the XGBoost classifier reveals the highest performance, I choose this model to evaluate the classification results using the resampled dataset under a given resampling technique.

Using the XGBoost algorithm, the classification results corresponding to different resam-

Model	**Hyperparameters**
XGBoost	Number of trees=300; max depth = 4; learning rate=0.1
Gradient Boosting	Number of trees=300; max depth=4; learning rate=0.1
Random Forest	Number of trees=100
ExtraTrees	Number of trees=100; max features=9
MLP	Optimiser='Adam'; hidden layer size=50, epochs=100
Logistic Regression	C=10.0; epochs=100

Table 4.1: Hyper-parameters of the supervised learning models using the preprocessed Ethereum dataset.

Model Used	**%Accuracy**	**%F_1-Score**	**%AUC**
XGBoost	98.91	97.61	99.8
Gradient Boosting	98.47	96.63	99.8
Random Forest	98.06	95.7	99.7
ExtraTrees	97.76	94.99	99.7
MLP	95.56	90.14	60.6
Logistic Regression	79.58	20.96	70.5

Table 4.2: Classification results of supervised learning models on the preprocessed Ethereum dataset

pling techniques are tabulated in Tables 4.3 using accuracy, precision, recall, and f_1-score. In addition, I plot the ROC-AUC curve to highlight the performance of resampled datasets with XGBoost as depicted in Figure 4.4.

4.3.2 Evaluation of Feature Importance

To compute the feature importance, I use the feature permutation method [34] which shuffles each feature vector in the dataset in order to find its importance. I apply the feature permutation method that is found in the sklearn package [166] where the test is chosen to be repeated five times. This means that the feature scores are derived from the mean of multiple shuffled versions of the feature permutation method on each feature. After computing the importance scores on the involved features for the mentioned resampling techniques, I arbitrarily choose the three resampling techniques ENN-all, SMOTE-SF and Kmeans-SMOTE to present their feature importance. Due to the high number of features, I present the highest feature scores of the original dataset as the baseline model (i.e., NoSMOTE). The feature importance scores on the train and test sets are visualised in Figure 4.5.

4.3.3 Statistical Method for Effect of Data Resampling on Feature Importance

In this study, the purpose is to find the effect of data resampling on feature importance. The reason is that the resampled data tend to change the original class distributions after data sampling. The variation of class distribution is tied to the feature importance that the model uses to perform predictions. On the other hand, the feature importance is also a method in XAI processes to explain the model's predictions. So I aim to find whether the resampling techniques influence the feature importance. I formulate a hypothesis test as follows:

- Null Hypothesis

  H_0: Feature importance is not influenced by the resampling technique.

Resampling Techniques	**%Accuracy**	**%Precision**	**%Recall**	**%F_1 Score**
ENN-all	99.42	99.31	89.10	93.93
NoSMOTE (Original dataset)	98.02	97.96	71.09	82.39
Kmeans-SMOTE	98.02	97.96	71.09	82.39
LLE-SMOTE	98.02	97.96	71.09	82.39
DEAGO	98.02	97.96	71.09	82.33
ENN	98.01	99.34	69.89	82.05
AHC	97.96	96.38	71.37	82.01
CURE-SMOTE	97.95	96.96	70.72	81.79
Safe-level-SMOTE	97.96	97.93	70.17	81.76
OSCCD	97.82	95.34	69.89	80.66
SMOTE-SF	97.65	90.13	71.74	79.89
TRIM-SMOTE	97.66	90.72	71.37	79.89
SMOTE	97.57	88.87	71.56	79.28
SOMO	97.66	97.01	66.02	78.57
SMOTE-TomekLinks	97.41	86.14	71.74	78.28
Borderline-SMOTE1	97.25	84.00	71.28	77.12
Borderline-SMOTE2	97.35	88.12	68.51	77.09

Table 4.3: Comparison between the resampling techniques applied to the preprocessed Ethereum dataset. The performance of these techniques is evaluated using random forest classifier.

- Alternative Hypothesis

  H_1: Feature importance is influenced by the resampling technique.

We use the Wilcoxon hypothesis test [223] which is suitable for the arisen hypothesis. The Wilcoxon hypothesis test is demonstrated in Chapter 3. Using the same procedure, I use the sklearn package in Python programming language to find the p-values of each resampling technique against the original dataset (NoSMOTE). For instance, the p-value feature importance of SMOTE technique on the Ethereum dataset is denoted as follows:

$$\text{Wilcoxon(SMOTE, NoSMOTE)},$$

The p-value is the probability value given that the null hypothesis is true where the p-value is found by comparing the test statistic T to Student's t-distribution. We choose the significance level α equal to 0.05. When the p-value is lower than α, I have evidence to reject the null hypothesis where the test is statistically significant. The p-values of the three mentioned resampling techniques against the NoSMOTE model are provided in Table 4.4.

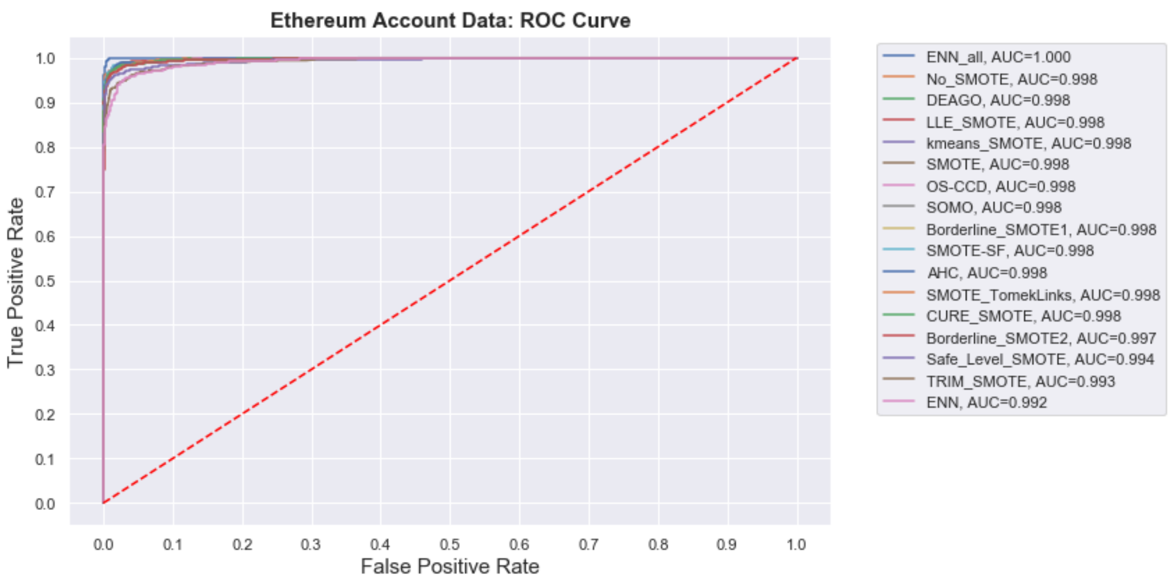


Figure 4.4: Experimental results of resampling methods: ROC-curve analysis using random forest classifier with various data resampling methods on the Ethereum dataset.

4.4 Discussions

4.4.1 Results of Classification and Resampling Techniques

ENN-all has outperformed all other resampling techniques as well as the non-sampled data (NoSMOTE) using the Ethereum dataset. I note the slight increase from 98.91% to 99.38% for accuracy and from 97.61% to 98.34% for f1-score using ENN-all undersampling technique as provided in Table 4. This slight increase in the model's performance illustrates the few noisy instances that already exist in this dataset. Consequently, the results derived from the various resampling techniques have revealed good decision making due to a smaller number of noisy instances. SMOTE has recorded the highest recall on this data of value 96.91%. However, this reduces the precision from 99.09% to 97.34%. Oversampling techniques have not shown any improved performance in terms of accuracy wherein the misclassified instances are not reduced. However, the AUC scores by different oversampling techniques have revealed adequate results referring to the ROC-curve analysis in Figure 4.4. Using the original Ethereum dataset, the classification results using XGBoost have revealed a significant success in comparison to the results obtained in the dataset's original paper as provided in Table 4.5.

4.4.2 Feature Importance

The feature "Total ERC20 tnxs" feature has played an important role in the whole dataset using SMOTE-SF as shown in Figure 4.5, whereby this resampling technique oversamples a subset of features that are selected with the highest Fisher-score. Meanwhile, the feature "Time

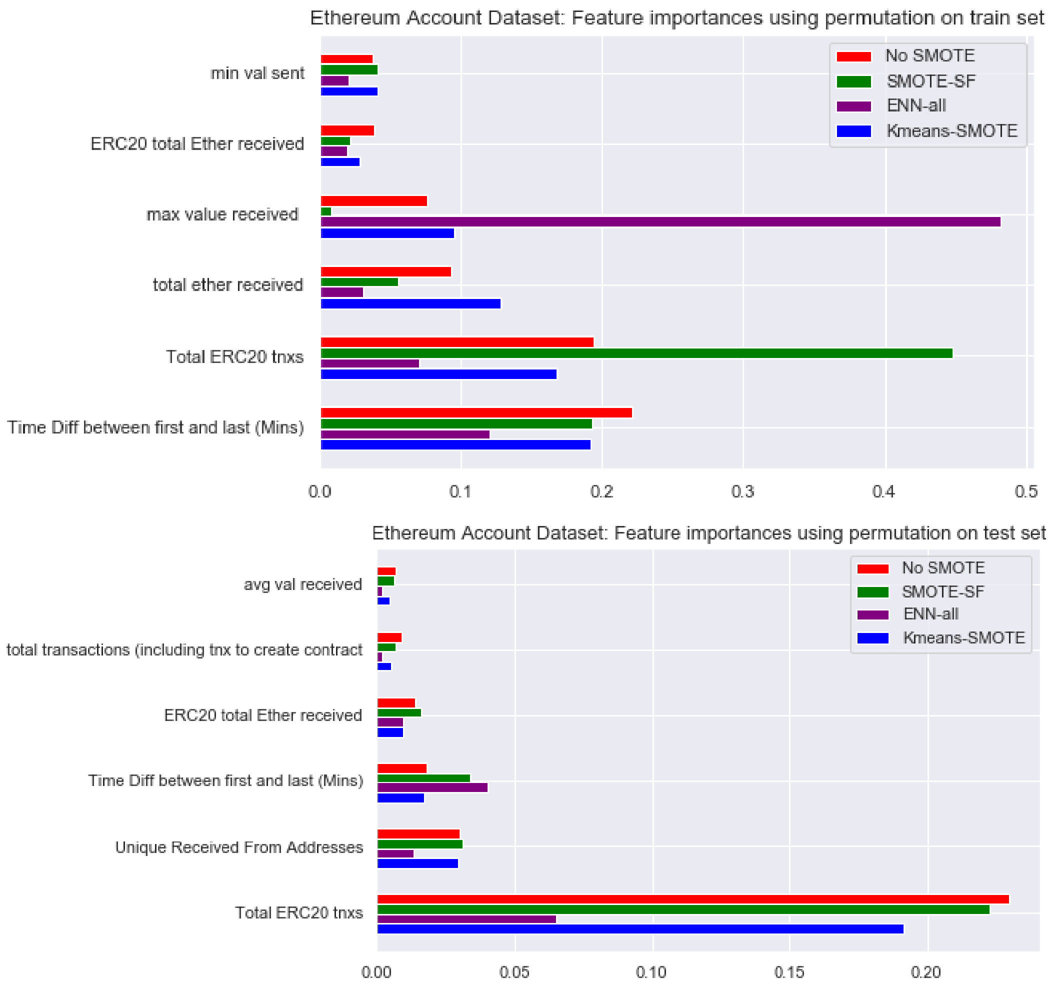


Figure 4.5: Effect of resampling techniques on feature importance in Ethereum. The top figure belongs to the feature importance of the train set. The bottom figure belongs to feature importance on the test set.

Model Used	**Dataset**	**Wilcoxon Test**	**P-values**
XGBoost	Ethereum Train Set	Wilcoxon(SMOTE-SF, NoSMOTE)	0.013
		Wilcoxon(ENN-all, NoSMOTE)	0.003
		Wilcoxon(Kmeans-SMOTE, NoSMOTE)	0.564
	Ethereum Test Set	Wilcoxon(SMOTE-SF, NoSMOTE)	0.061
		Wilcoxon(ENN-all, NoSMOTE)	0.866
		Wilcoxon(Kmeans-SMOTE, NoSMOTE)	0.259

Table 4.4: Wilcoxon test for the feature importance between resampled and non-sampled dataset of Ethereum using different resampling techniques.

Methods	**%Accuracy**	**%F_1-Score**
Preprocessing + XGBoost [61]	96.3	96
Preprocessing + XGBoost (Ours)	**98.91**	**97.6**

Table 4.5: Comparison between our experiments and the original contribution of the Ethereum dataset. This table highlights the effectiveness of the data preprocessing step in our experiments.

Diff between first and last (Mins)" reflects the total duration of account usage in Ethereum which reveals a high impact on the classification of fraud accounts.

As mentioned earlier, Farrugia et al. have pointed out the most important features as:

- *time difference between the first and last transaction*
- *total available Ether balance*
- *the minimum value in Ether received by an account.*

The common features that match their and our study is:

- *time difference between the first and last transaction*
- *the minimum value in Ether received by an account*

4.4.3 Influence of Resampling Techniques on Feature Importance

Here, I discuss the statistical results of the formulated test using the Ethereum dataset. As the explainability of the models in this field is highly desirable, the change in feature importance caused by resampling methods affects the explainability of the model as it is highly tied to the feature importance. The test is statistically significant with SMOTE-SF and ENN-all in the train set where the p-values record 0.013 and 0.003 in Table 4.4, respectively. As a result, there is enough evidence to say that the latter resampling techniques affect the explainability of the model. For the test set, the test is not significant to reject the null hypothesis where the explainability is not influenced.

4.5 Summary

In this chapter, I present the Ethereum dataset derived to classify the fraudulent accounts in the Ethereum blockchain. Subsequently, I perform data preprocessing to perform classification using a variety of supervised learning methods. XGBoost classifier reveals the highest performance in comparison to other models. Then I address the class imbalance by performing data resampling using a variety of resampling techniques. None of the oversampling techniques shows an improved performance in terms of accuracy. I also study the feature importance and compare our results with the dataset's original study. The classification results provided in our study outperform the results of the original work using this dataset. Furthermore, I perform test statistics to verify the influence of the resampling techniques on the feature importance. Some resampling techniques reveal good evidence to say that the feature importance is influenced which in turn affects the explainability of the model. In Chapter 5, I include this dataset in our experiments to study uncertainty estimation.

Chapter 5

Uncertainty Estimation in Machine Learning

Contents

5.1 Chapter Preface

One of my aim in this project is to study the uncertainty of the predicted input in the machine learning model. In the blockchain, the data derived from this ever-evolving technology might be subjected to unexpected events which are not learned by the machine learning model. To handle such events, uncertainty estimation is an inevitable approach to acquire a degree of belief about the model's predictions, where the uncertainty estimates play a pivotal role besides the machine learning model to predict illicit activities in the blockchain. Mainly, machine learning algorithms are often optimised to find the best point estimation (e.g. maximum likelihood estimation) in order to approximate the best function that maps the feature vectors into probability outputs. The predictive probability is defined as the confidence of the feature vector falling in an output class. However, the point estimation approach in machine learning is not capable of conveying reasonable confidence in its predictions. Thus, the misclassified instances are often provided erroneously with unjustified high confidence where uncertainty estimation is needed besides the predictions.

This chapter discusses the limitation of existing uncertainty estimation methods that I examine using datasets derived from the Bitcoin and Ethereum blockchain. Afterwards, I present a novel uncertainty method which I evaluate using real-world blockchain datasets and compare it to the existing uncertainty estimation methods.

First, I present and discuss the limitation of the uncertainty estimation method known as Monte-Carlo dropout (MC-dropout). This is one of the methods that I utilise during my studies due to its simplicity and efficiency to produce uncertainties. However, due to some limitations that I have encountered in the latter method and other existing methods, I propose a novel uncertainty method based on the adversarial attack idea – so-called MC-AA – to capture the model's uncertainty. MC-AA is proposed to tackle the issue which occurs in the previous uncertainty estimation methods. To validate the proposed MC-AA, I perform a comprehensive study to test this method using a variety of neural network models on Bitcoin and Ethereum datasets. Moreover, I extend my study to other datasets such as the graph-structured Cora and GitHub datasets as well as the MNIST image dataset. The purpose of experimenting with the proposed uncertainty estimation method MC-AA with datasets that are out of scope is to provide solid work on the validity of the performance of this uncertainty estimation approach. Thus, this chapter includes comprehensive work about the MC-AA performance with MLP and various graph neural network (GNN) models where the binary and multi-class classification tasks are addressed.

5.2 MC-dropout: An Overview, Discussion and Limitation

5.2.1 Abstract

Monte-Carlo dropout (MC-dropout) method has been introduced as a probabilistic approach based on Bayesian approximation which is more computationally efficient than Bayesian neural networks. MC-dropout has revealed promising results on image datasets regarding uncertainty quantification. However, this method has been subjected to criticism regarding the behaviour of MC-dropout and what type of uncertainty it actually captures. For this purpose, I discuss the behaviour of MC-dropout on classification tasks using synthetic and real data. I empirically explain different cases of MC-dropout that reflect the relative merits of this method. My main finding is that MC-dropout captures data points lying on the decision boundary between the opposed classes using a synthetic dataset. On the other hand, I apply the MC-dropout method on a dataset derived from Bitcoin known as Elliptic data to highlight the outperformance of the model with MC-dropout over the standard model.

5.2.2 Introduction

Deep neural networks (DNNs) have attained successful performance in a variety of fields such as medical imaging [190], computer vision [179], fault detection [104], and attractively in big data applications, [144]. DNNs are a powerful tool to learn parameters to find the best point estimation that maps the given feature vector to the desired output in classification tasks. Due to the point estimation, these mappings are assumed to be exact which is not always the case in the presence of the softmax function that outputs single predictions. Consequently, the misclassified examples are often provided erroneously with unjustified overconfidence. On the other hand, Bayesian approaches such as Bayesian neural networks (BNNs) and Gaussian processes have received significant attention for providing uncertainty measurements besides the predictive probabilities. Unlike single predictions, BNNs and Gaussian processes yield predictive distributions in which the weights of BNNs are incorporated with priors distribution [150], whereas Gaussian processes introduce priors over functions. Gaussian processes sample prior functions from a multivariate Gaussian distribution and update these priors with newly collected data using the Bayesian method. BNNs and Gaussian processes are computationally expensive when dealing with a high number of parameters as in the case of neural networks with a high number of units and multiple layers. BNNs require getting the posterior distribution across the network's parameters, in which all possible events are obtained at the output. Gaussian processes need to sample prior functions from multivariate Gaussian distribution, wherein the dimension of Gaussian distribution increases proportionally with the number of training points involving the whole dataset during predictions. Recently, the outgrowth of Bayesian approaches has witnessed a resurgence of interest in which is computationally much more efficient than BNNs known as the Monte-Carlo dropout (MC-dropout) method. As introduced in [69], MC-dropout is shown to be an approximation of the probabilistic Bayesian model known as the deep Gaussian process (GP). Furthermore, this technique is viewed as the minimisation of Kullback–Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process. MC-dropout has achieved promising results in regression and classification problems, especially in image datasets [102, 193, 138]. Concisely, MC-dropout is a method of performing multiple stochastic forward passes with the means of activated dropout in a neural network during the testing process to provide an ensemble of predictions that could reflect uncertainty estimations. Establishing uncertainty is necessary for the model to know what it does not know. The main types of uncertainty are known as epistemic, aleatoric, and predictive uncertainty where the latter is the combination of the former types [68]. Epistemic uncertainty is derived from the lack of training data in the prediction region. This uncertainty is reducible by collecting more data which enhances the model's belief. Aleatoric uncertainty is accounted for by noisy observations such as data collected by sensors. This uncertainty is irreducible by the model but rather mitigated by the source of observations. Apart from uncertainty, another term introduced in [158] is called risk. It has been defined as the inherent stochasticity in the model's parameters and can be seen as the variability of the decision boundary in classification applications. MC-dropout has been criticised by many researchers regarding the type of uncertainty that is captured especially in [158]. For instance, the work in [158] has claimed that MC-dropout is tied with risk and not uncertainty estimations which contradicts MC-dropout's behaviour in [68]. For this purpose, we aim to provide an illustrative experimental discussion about the behaviour of MC-dropout using a toy example in the light of the previous contradictory contributions in [69] and [158]. Also, I discuss the performance of MC-dropout on real data derived from the Bitcoin blockchain (i.e., Elliptic data) and I provide the strength and drawbacks of this method. Furthermore, I show that MC-dropout is not able to deal with out-of-distribution data, whereas it works well in some cases which I discuss in the upcoming sections.

5.2.3 Background

The growing interest in BNN models has been extensively established in [39, 130] in which Gaussian distribution is identified over the network's parameters around the mode to compute the posterior distributions. Although Bayesian models are powerful in uncertainty estimation, the exact inference of posterior distribution is intractable. Meanwhile, variational inference, Bayesian training by the Hybrid Monte Carlo method, and Markov Chain Monte Carlo with Hamiltonian Dynamics also exist referring to [77, 148, 149] in which assumptions to be made regarding the approximated posterior distribution. These methods are difficult to scale to large datasets [68]. The mentioned studies are limited in some cases or require additional costs. Recently, an efficient method to compute uncertainties is introduced known as Monte-Carlo dropout (MC-dropout) [69]. Regarding classification tasks, I explain the practical implementation of MC-dropout, induced uncertainties, and the criticism of this work in more detail in the following sections.

5.2.3.1 How Does MC-Dropout Work?

Primarily, dropout is introduced as a simple regularisation technique to reduce overfitting in neural networks [196]. This technique is often applied during the training process of neural networks in which a less over-fitted and more generalisation of the model over the predictions occurs. On the other hand, applying dropout before every weight layer has shown to be mathematically equivalent to an approximation to the probabilistic deep Gaussian process [69].

Consider $\hat{y}$ as an output of a neural network model with arbitrary layers L and parameters $w = \{W_1, ..., W_L\}$ as the weight matrices. Let y^* be the observed output associated by the input vector x^*. Given a dataset $\mathcal{X} = \{x_1, ..., x_N\}, \mathcal{Y} = \{y_1, ..., y_N\}$, the predictive distribution can be expressed as:

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, w)p(w|X, Y)dw, \tag{5.1}$$

where $p(y^*|x^*, w)$ is the model's likelihood and $p(w|X, Y)$ is the posterior over the weights. The predictive distribution involves a predictive mean and variance yielding uncertainty estimation. However, the posterior distribution is analytically intractable. Instead, an approximation of variational distribution $q(w)$ is obtained from the Gaussian process to be close as much as possible to $p(w|X, y)$ in which the optimisation process occurs by minimising the Kullback-Leibler divergence (KL) between the preceded distributions as follows:

$$KL(q(w)|p(w|X, Y)). \tag{5.2}$$

Using the variational inference method, the predictive distribution can be approximated as:

$$q(y^*|x^*) = \int p(y^*|x^*, w)q(w)dw. \tag{5.3}$$

Referring to [69], $q(w)$ is chosen to be the distribution over matrices whose columns are randomly set to zero according to Bernoulli distribution expressed as:

$$W_i = M_i.diag([z_{i,j}]_{j=1}^{K_i}), \tag{5.4}$$

where $z_{i,j} \sim$ Bernoulli(p_i) for $i = 1, ..., L$ and $j = 1, ...K_{i-1}$, with K_ixK_{i-1} the dimension of matrix W_i. p_i refers to the probability of dropout and M_i is a matrix of variational parameters (Please refer to [69] for more details). Thus, drawing T sets of vectors of samples from Bernoulli distribution yields $\{W_1^t, ..., W_L^t\}_{t=1}^T$. Therefore, the predictive mean can be written as:

$$E_{q(y^*|x^*)(y^*)} \approx \frac{1}{T}\sum_{t=1}^{T} \hat{y}^*(x^*, W_1^t, ..., W_L^t) = p_{MC-dropout}(y^*|x^*), \tag{5.5}$$

where $\hat{y}^*$ is the mapping of x^* by the given neural network and $p_{MC-dropout}$ is the predictive mean of MC-dropout, which is equivalent to performing T stochastic forward passes over the neural network during the testing process with dropout then averaging the results. This method is viewed as an ensemble of approximated functions with shared parameters which is an approximation of the probabilistic Bayesian approach known as deep Gaussian processes. Since this method yields multiple outputs per input, therefore uncertainty could be inspected by computing the variance, entropy, mutual information. . . etc. of these outputs; I consider mutual information only in this study which is discussed later. More practically, the neural network with dropout learns a variety of hypotheses by randomly subsampling from the units of the hidden layer. De-activating dropout during the testing process, the neural network reproduces all hypotheses simultaneously as an ensemble of decision functions to perform a single prediction as to the case in random forests to reduce the variance and consequently prevent overfitting. In the case of MC-dropout, the only difference is that multiple outputs are provided for a single input wherein uncertainty estimation is desirable. Furthermore, this method provides higher independency between outcomes that reflects the behaviour of the variety of decision functions.

5.2.3.2 Predictive Uncertainty Using MC-Dropout

Several uncertainty measurements have been presented in previous studies such as variance, mutual information and others. I only consider mutual information (MI) to express predictive uncertainty. Mutual information distinguishes the type of uncertainty by capturing the epistemic uncertainty in the model [193]. Referring to [138], mutual information (MI) can be expressed as:

$$\hat{I}(y^*|x^*, w) = \hat{H}(y^*|x^*, w) + \sum_{c} \frac{1}{T}\sum_{t=1}^{T} p(y^* = c|x^*, w) \log p(y^* = c|x^*, w), \tag{5.6}$$

where c is the class label, and

$$\hat{H}(y^*|x^*, w) = -\sum_{c} p_{MC-dropout}(y^* = c|x^*, w) \log p_{MC-dropout}(y^* = c|x^*, w). \tag{5.7}$$

MI captures the mutual dependency between the hypotheses derived from Monte-Carlo samples over the predictions in which MI identifies the information gain of the model's confidence about its output.

5.2.3.3 Uncertainty and Risk

An opposed discussion regarding MC-dropout has been introduced in [158]. Risk is defined as the inherent variability in a model, whereas uncertainty appears to capture my uncertain belief about the predicted value. Consider the popular example of tossing a coin. The probability of drawing a head or tail will hold some risk. However, in the case of a biased coin, my beliefs are updated when I get more observations which are denoted as uncertainty. This "risk" term matches the case when dealing with feature vectors falling near the decision boundary in binary classification tasks. Consequently, MC-dropout provides flipped outputs across extremely close classes. Furthermore, the test points occurring between two overlapping classes also identify aleatoric uncertainty. Thus, the two terms risk and aleatoric uncertainty hold the same concept to some extent. In both types, the risk in the model and the aleatoric uncertainty of the predictions are irreducible. This example will be illustrated later in the experiments of this study.

5.2.4 Experiments, Discussions and Limitations

5.2.4.1 Classification Using Synthetic Data: Toy Example

In this section, I provide an empirical study of the MC-dropout method using synthetic datasets for the classification task. I perform two experiments (separable and non-separable data) using two different datasets generated by *make_ classification* and *make_ circles* functions provided by scikit-learn package in Python programming language [166]. The *make_ classification* is used to randomly generate balanced n-class distributions drawn from a normal distribution situated on the vertices of a 2D hypercube with length 2 (using two clusters per class). The

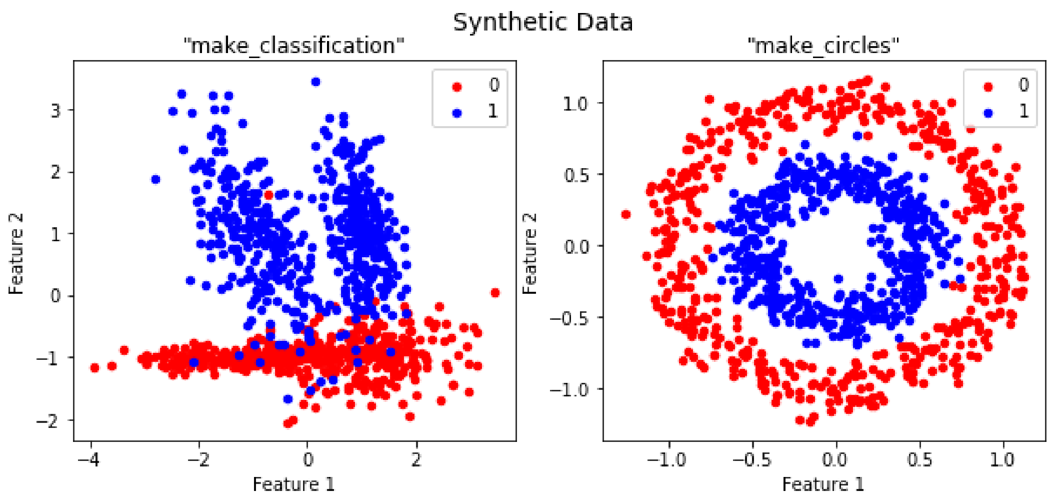


Figure 5.1: Synthetic data: Toy example of 2D separable (left subplot) and non-separable (right subplot) datasets.

make_ circles produces Gaussian samples with a spherical decision boundary in which the function's parameters acquire noise of 0.1 and a scale factor of 0.5 to separate the circles. For separable and non-separable data, I generate 1000 samples of 2D features corresponding to two different classes altogether as shown in Figure 5.1. I train a neural network for each dataset with the same settings of two hidden layers of widths 100 and 81 respectively, and an output layer squashed with a softmax function for the binary classification. A dropout function is applied after every hidden layer with a dropout ratio of 0.4. After that, the same set is used in testing with an activated dropout setting to obtain the Monte-Carlo samples with 100 iterations (T=100). The learning rate is set to 0.001 and the number of epochs is fixed at 50. The average of the Monte-Carlo samples is used as the predicted mean referring to Equation 5.5, and its predictive uncertainty is computed using MI. In the upcoming experiments, I denote by uncertainty threshold as T_u as introduced previously in [138]. This threshold can be randomly assigned between the minimum and the maximum uncertainty values in the test set.

5.2.4.1.1 Using data of *make_ classification* : We compute the predictive mean and uncertainty of each instance as shown in Figure 5.2. Clearly, the data points with a predictive mean in a neighbourhood of 0.5 are more likely to acquire higher predictive uncertainty. I assign different thresholds T_u to the predictive MI, where $T_u \in \{0.6,\ 0.4,\ 0.2\}$. The data points with MI values exceeding T_u are highlighted in which uncertain predictions occur. Subsequently, I highlight the plottings of the uncertain predictions assigned by these thresholds as depicted in Figure 5.2. Interestingly, these uncertain instances are located between the boundaries of the class distributions. Starting with $T_u = 0.6$, a few green points that correspond to uncertain predictions lie between class 0 (red) and class 1 (blue) as shown in Figure 5.3.

A higher number of green points follows the boundary patterns as more uncertain points are added with the increased threshold. Specifically, these green instances show different types of uncertainty. Referring to Figure 5.3, the highlighted instances falling between the transition region of the opposed classes are known as aleatoric uncertainty. The other green points that are not falling in the transition region of the class distributions are rather falling on the edge of

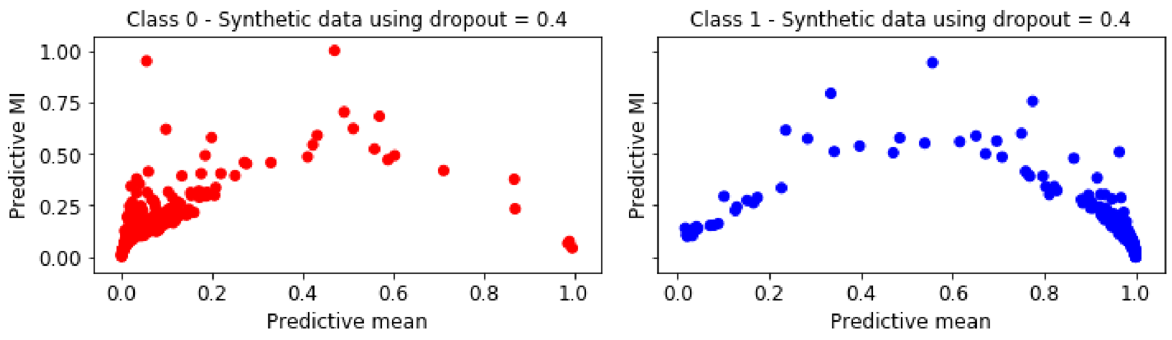


Figure 5.2: Data derived from "make_classification". Scatter plot of predictive MI vs mean of class 0 (red) and class 1 (blue).

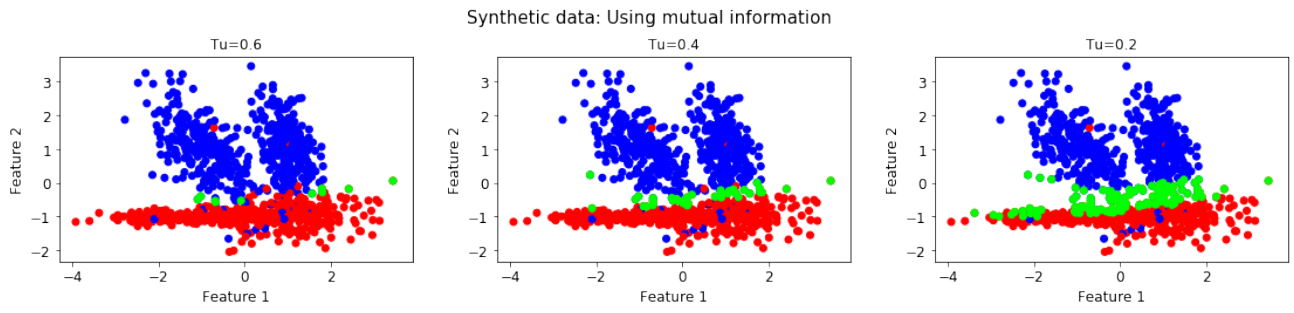


Figure 5.3: Synthetic data with different uncertainty thresholds T_u= {0.6, 0.4, 0.2} respectively in the subplots. Class 0 is plotted in red and class 1 is in blue. The green plots belong to the uncertain instances with respect to T_u using predictive MI.

the class itself in which the model has a weak belief about the predictions (shown on the most left and right sides of each subplot in Figure 5.3). These points correspond to the model's epistemic uncertainty where the lack of training points occurs as mentioned earlier. Apart from the major types of uncertainties, I realise that MC-dropout captures the points falling on the boundaries of the class distributions. Thus, MC-dropout allows the variability in the model's parameters (ensemble of hypotheses) resulting in variations of the boundary lines between class distributions. For brevity, we generate three different test points of coordinates (-0.5, 4), (-3, 0), (0, -0.5) as shown in Figure 5.4. Afterwards, I perform multiple stochastic passes to compute the MI of each case, in which MC-dropout behaviour can solely be summarised in three different cases as follows:

1. First test point of coordinates (-0.5, 4): The predictive MI of this point is 0. This case reveals the drawback of MC-dropout in which the test point lacks training data and cannot be detected as epistemic uncertainty as it is far from the decision boundary between the given classes.

2. Second test point of coordinates (-3, 0): This point has acquired predictive MI of value 1. This point falls near the decision boundary of the given classes, which causes variability

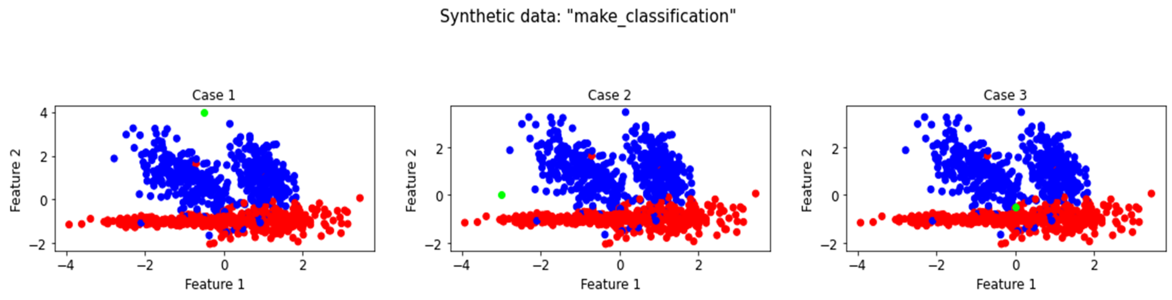


Figure 5.4: Representation of *make_ classification* data. Each subplot represents the given training set with a single test point shown in green that reflects a unique behaviour of MC-dropout.

in the decision functions using Monte-Carlo samplings.

3. In this case, the point of coordinates (0, -0.5) is generated on the region of overlapping classes (decision boundary). The predictive MI at this point is 0.229. The reason for the low uncertainty on the decision boundary is due to the limited variability of the line separating these two classes, whereas the test point occupies a region of overlapped classes. Hence, the multiple passes on this point provide nearly stable predictions.

5.2.4.1.2 Using data of *make_ circles* : In this experiment, I apply MC-dropout on non-linearly separable data generated by *make_ circles* function. Similarly, I choose different values of T_u of 0.6, 0.4 and 0.2 to study the effect of MC-dropout using predictive MI as shown in Figure 5.5. As depicted in Figure 5.6, MI apparently captures the data falling on the spherical boundary between the class distributions. Furthermore, it is reasonable to obtain more uncertain instances that belong to class 0 (outer circle in red). This issue is due to the dispersed distribution of class 0 on a wider circle which increases the variability of the decision boundary resulting in weak predictions. In general, there is no doubt that part of the uncertain instances captured by MC-dropout belongs to the aleatoric uncertainty type and others to the epistemic type. However, the original behaviour of this method is not tied to uncertainty, but to the variability of the decision boundary. For instance, the outer circumference of the outer circle in Figure 5.6 does not include any type of uncertainty. Therefore, any data point falling on the outer region apart from any data is provided with high certainty that belongs to class 0 (red) which contradicts the concept of uncertainty estimation.

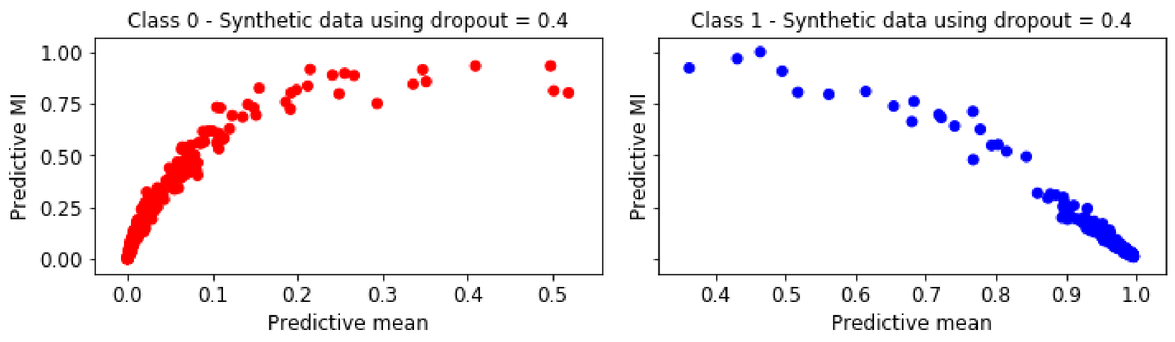


Figure 5.5: Data derived from "make_circles". Scatter plot of predictive MI vs mean of class 0 (red) and class 1 (blue).

5.2.4.2 Classification Using Real-World Dataset: Elliptic Data

In this section, I study the effect of the MC-dropout method using Elliptic data. In this experiment, I discuss the relative merits of using MC-dropout.

5.2.4.2.1 MC-dropout with Elliptic data : Using Elliptic data, the first 34 graphs are used as a training set including a validation set, whereas the remaining 15 graphs are used as

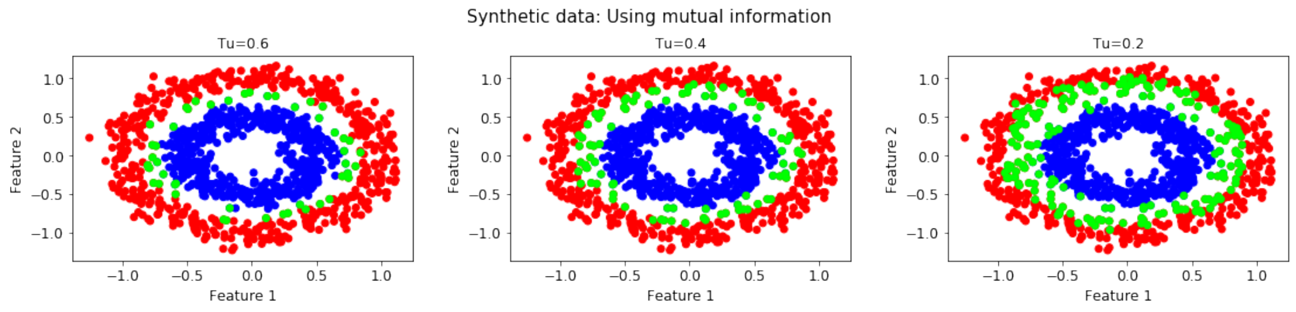


Figure 5.6: Synthetic data with different uncertainty thresholds T_u= {0.6, 0.4, 0.2} respectively in the subplots. Class 0 is plotted in red and class 1 is in blue. The green plots belong to the uncertain instances with respect to T_u using predictive MI.

Model	**%Accuracy**	**%F_1-score**	**%AUC**
Standard (no dropout)	94.3	61.9	89.2
MC-dropout	96.6	72.9	89.4

Table 5.1: Comparison between standard and MC-dropout models using Elliptic data. Area-Under-Curve (AUC) represents the goodness of classification between class 0 (licit) and class 1 (illicit).

a test set. I assess the performance of MC-dropout by training two neural networks with and without dropout each of two hidden layers of 100 and 81 neurons respectively. I refer to the first model as the standard model, where the dropout layer is not involved in the algorithm. The second model, the MC-dropout model, is tied with a dropout layer after every hidden layer with a dropout ratio of 0.3, wherein the Monte-Carlo sampling is performed on the test set with 100 stochastic forward passes. Hence, predictive uncertainties are measured using mutual information (MI). I evaluate the performance of these two models to reflect the goodness of classification as shown in Table 5.1. Unsurprisingly, MC-dropout has outperformed a standard model since the former model uses an ensemble of decision functions shared where the final prediction results are enhanced.

5.2.4.2.2 Illustration of MC-dropout behaviour on Elliptic data : Regarding MC-dropout model, I plot predictive MI versus mean for licit and illicit classes as shown in Figure 5.7. Referring to Figure 5.7, the two subplots refer to the true labels of class licit and illicit respectively, whereas the predictive mean corresponds to the model's predictions after using multiple stochastic forward passes. Generally, I realise that false predictions are associated with high certainty as the model is able to detect few with uncertain estimations. There are two assumptions behind the high certainty of false predictions. The first assumption is the data. The second assumption could be the lack of features, wherein the test points are erroneously embedded in the wrong class, and these predictions cannot be detected by any machine learning model. On the other hand, assuming that a threshold T_u of 0.5 is assigned

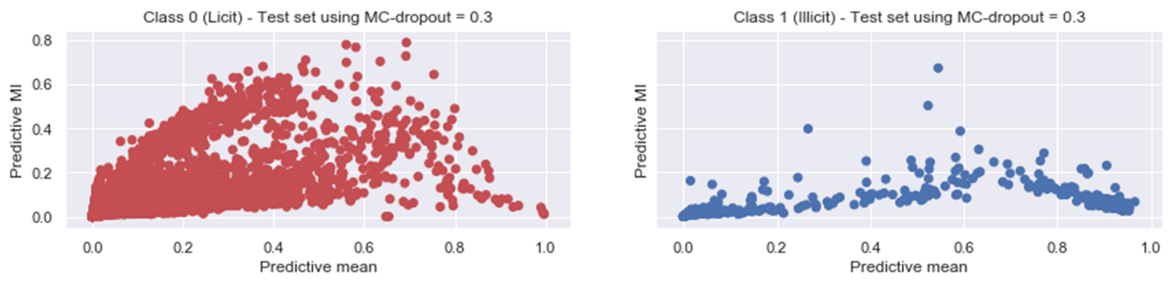


Figure 5.7: Predictive mean and MI of the test set using Elliptic data. The left subplot corresponds to the licit class (red) whereas the right subplot corresponds to the illicit class (blue).

to the given test set, I reject the uncertain test points. Consequently, the accuracy of the accepted predictions (remaining test points) becomes 0.967 with f_1-score 0.736.

5.2.5 Limitations

In this section, I discuss some limitations of the MC-dropout method in the context of the captured uncertainty. However, it depends on the application whether these limitations matter or not.

5.2.5.1 Out-of-Distribution Data

As I realised, many researchers consider that MC-dropout computes data points that are out of distribution. To represent Out-of-Distribution (OoD) data, I generate 100 test points with 93 features each sampled from Gaussian distribution of mean 3.0 and standard deviation 1.0 were chosen arbitrarily, knowing that the original data is normalised. With no doubt, MC-dropout is not able to provide any uncertainty on these points, in which they are predicted as a licit class with predictive uncertainty equal to zero.

5.2.5.2 Misclassified Examples in the Overlapping Regions

We have seen that the MC-dropout method captures the data points that lie near the decision boundary of the class distributions. This is caused by perturbing the decision boundary T-times to reflect the uncertainty about the predicted inputs. In the case of overlapping regions, class distributions are overlapped in which a set of points of a given class falls completely in the other one. This set of points cannot be captured by MC-dropout where the decision boundary cannot be perturbed.

5.2.6 Concluding Remarks

Dealing with uncertainty estimates is a desirable approach when exposed to critical predictions such as the case in anti-money laundering, where licit and illicit services are involved. The MC-dropout method as a Bayesian approximation is able to capture a portion of epistemic and aleatoric uncertainty types, however, this method is effective in capturing data points falling near the decision boundary of the given classes. This method is not a reliable approach for out-of-distribution data and does not differ from unjustified predictions provided by a standard neural network. Moreover, the misclassified instances that occur in the wrong class distribution cannot be detected using MC-dropout. This is due to the behaviour of the MC-dropout where the decision boundary is subjected to multiple perturbations that will not influence this type of misclassified instance. However, the misclassified instances that occur between the different class distributions are still detected by MC-dropout. In this case, MC-dropout deals efficiently with misclassified where this case is highly encountered in image datasets.

5.3 MC-AA: A Novel Uncertainty Estimation Method

5.3.1 Abstract

To tackle the limitations provided in the preceded part, I propose a novel method to capture data points near the decision boundary in the neural network that are often referred to as a specific type of uncertainty. In this experiment, I present an uncertainty estimation based on the idea of the adversarial attack method. Here, uncertainty estimates are derived from the input perturbations, unlike previous studies that provide perturbations on the model's parameters as in the Bayesian approach. The method is able to produce uncertainty with a couple of perturbations on the inputs. I test the proposed method using datasets derived from Bitcoin and Ethereum blockchains. I compare the performance of model uncertainty with the most recent uncertainty methods. I show that the proposed method has revealed a significant outperformance over other methods and provided less risk to capture model uncertainty in machine learning.

5.3.2 Introduction

Standard Neural network models have admitted great success in approximating functions that map the inputs to desirable outputs in several domains [190, 104, 144]. Nevertheless, a single function approximation often leads to overconfident and erroneous predictions. To reliably perform predictions, uncertainty estimation is a desirable approach to provide on which inputs the model is not certain about its predictions. Uncertainty estimation has increasingly received considerable attention in recent years. Several studies are conducted to express uncertainty such as Monte Carlo dropout (MC-dropout) [69], Deterministic Uncertainty Quantification (DUQ) [210] and Deterministic Uncertainty Estimation (DUE) [209]. However, these methods are unable to capture noisy data points between the overlapping class distributions (critical region) due to their inherent approximations. For instance, the perturbations of the decision boundary given by MC-dropout can only cause fluctuations in the model output by the inputs lying in this critical region (decision boundary zone). However, a noisy data point lying completely in a different region hinders the performance of MC-dropout wherein data points falling in the wrong classes cannot be triggered by the variability of the decision boundary. This issue is further illustrated in the upcoming sections. Generally, existing methods rely on weight variability or function approximation variability which in effect cause perturbations in the decision boundary.

This behaviour hinders the performance of these approaches since they cannot capture noisy data points in overlapping regions leading to a high number of misclassified and certain predictions. Here, I propose a new method based on the adversarial attack to estimate uncertainties in neural network models based on the binary classification task. I refer to my proposed method as Monte Carlo based Adversarial Attack abbreviated by *MC-AA*.

The term "Critical region" is inspired by the known hypothesis test in statistics wherein the null hypothesis is accepted or rejected. By analogy, classification in machine learning resembles a hypothesis test to perform decision-making, wherein critical region means the region between the opposite classes or the decision boundary zone.

Initially, adversarial attacks/examples are defined as the inputs to machine learning that an attacker has introduced to fool the outputs of the model [26, 199, 76]. These attacks have a great impact on the security and integrity of the machine learning model resulting in poor decision-making. Motivated by previous work, I estimate the uncertainty of the predictions using an adversarial attack. Firstly, I train a standard neural network then I apply the MC-AA method during the testing phase so that multiple outputs on each test point are introduced. Hence, I compute the mutual information on these outputs to estimate the uncertainty. Unlike previous studies that apply perturbations to the model parameters (e.g. in the Bayesian approach), my method sought to estimate uncertainty by perturbing the relevant inputs in a guided way. The uncertainty estimates are highly tied to the location of the test point with respect to the decision boundary. As a result, a perturbed input lying near the decision boundary produces fluctuated outputs between different classes. Furthermore, an input lying in overlapping regions is enforced to move back and forth between the different classes leading to uncertain predictions using MC-AA. Consequently, the proposed method is likely to consider any point lying on the border of the class distributions as uncertain. Accordingly, this reduces the risk to produce wrong and certain predictions. The proposed method is backed up by the results to obtain uncertainties and support my main idea.

The achievements in this study can be roughly stated as follows:

- I show that an adversarial attack is able to capture model uncertainty in binary classification tasks.
- I demonstrate a well-defined way to estimate and evaluate the uncertainty of the neural network model using MC-AA. In addition, I illustrate the behaviour of the proposed method in the feature space.
- I introduce the relevance of my proposed method to Bayesian approximations under given conditions.
- I perform a comparative analysis between the most recent uncertainty methods using datasets derived from the blockchain. My method shows superior success over the previous work to obtain uncertainty.

In what follows, I provide the background in light of the proposed method. Then, I present the proposed method and state how the uncertainty is obtained. Subsequently, I provide the experiments, discussions and concluding remarks.

5.3.3 Background

Recent studies have proposed more efficient methods to estimate uncertainties in neural networks known as Monte-Carlo dropout (MC-dropout) [69] as described earlier, Deterministic

Uncertainty Quantification (DUQ) [210], and Deterministic Uncertainty Estimation (DUE) [209]. In MC-dropout, the points falling near the decision boundary are more likely to be uncertain about [14]. Nevertheless, this method reveals two drawbacks. It requires many stochastic forward passes to output stable estimates, and its uncertainty estimates are not reliable due to the high number of erroneous and certain data points. DUQ is based on the idea of a Radial Basis Function (RBF) network incorporated with gradient penalty, wherein its output is squashed with Gaussian function to quantify the distance from the tested point to the centroids of the given class distributions [210]. This method is good at finding Out-of-Distribution (OoD) data. However, it is unreliable to capture noisy data points where the distance is not a sufficient metric to convey uncertainty. DUE is based on the approximation of the Gaussian process that scales to high dimensional data using variational inducing points with feature extractor of Deep Kernel Learning (DKL) [209]. Although its efficiency in capturing uncertainty, this method is not able to capture data points lying in between the overlapping class distributions. As the DUE method is based on the Gaussian process, this approach has never scaled to high dimensional datasets because of a lack of well-performing kernel function [210]. In addition, a comprehensive review of most uncertainty methods in [7] revealed the vigilance of previous studies on failures of model uncertainty caused by adversarial attacks. Also, the methods in [237, 243] have sought to improve the robustness of adversarial examples in machine learning. However, none of them has straightforwardly used the idea of adversarial attack to estimate uncertainty to the best of my knowledge.

Adversaries are crafted perturbations of legitimate inputs. The perturbed inputs influence the predictions of a trained model to output incorrect predictions [43]. Such inputs are designed by attackers to affect the security and integrity of the model. Some of these attacks incorporate white-box models meaning that the attacker acquires a detailed knowledge of the model's parameters and architecture and tries to design a perturbed input using the well-known Fast Gradient Sign Method (FGSM).

5.3.3.1 FGSM

FGSM is based on the gradient of the loss function with respect to the initial inputs. Consider a neural network incorporating a set of parameters $W = \{W_1, ..., W_L\}$ where L is the number of layers. Consider a dataset with size N as X=$\{x_1, ..., x_N\}$ and Y=$\{y_1, ..., y_N\}$, then the loss function of this model can be written as: $J(x, y)$. Hence, FGSM can be reformulated as follows:

$$x_{Adv} = x + \varepsilon.sign(\nabla_x J(x, y)), \tag{5.8}$$

where x_{Adv} is the adversarial example, ε is a small number, ∇_x is the gradient with respect to the input, $x \in$ X and $y \in$ Y. Basically, FGSM adds noise to the input data in the direction to the nearest decision boundary and scales it with ε as shown in Figure 5.8. By providing the incorrect class to the loss function, FGSM enforces the data points to walk towards the decision boundary in the direction of the incorrect class. This type of attack is rather tackled by adversarial training.

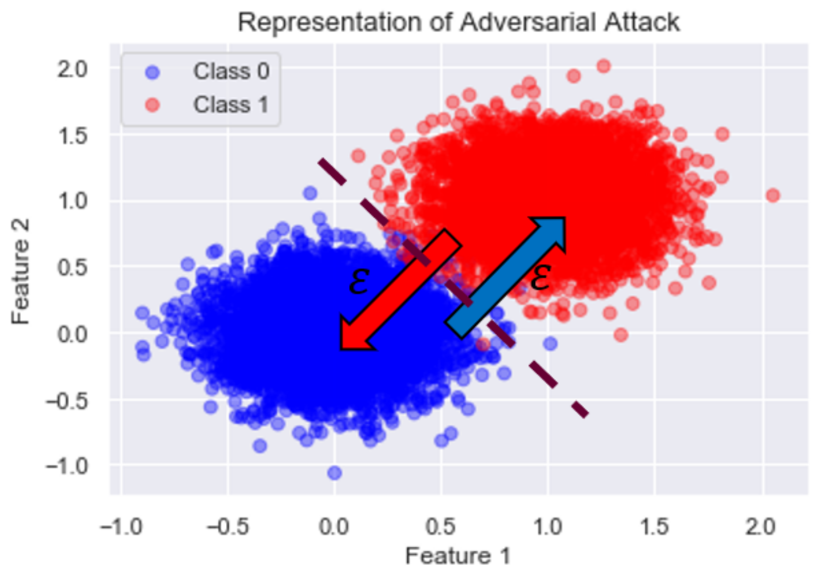


Figure 5.8: Toy example of adversarial attack with a linear classifier.

5.3.3.2 Adversarial Training

Adversarial training increases the model robustness by learning adversarial examples over the train set [193]. This is also referred to the data augmentation method so that the decision boundary is further extended to these sensitive perturbations. The robust model is obtained either by feeding the augmented data [112] or by minimisation of the modified objective loss function [76] that can be expressed as:

$$\tilde{J}(x, y) = \alpha J(x, y) + (1 - \alpha) J(x + \varepsilon.sign(\nabla_x J(x, y)), y), \tag{5.9}$$

where α is a regularisation parameter between 0 and 1, and other notations similar to that in Equation 5.8. The modified objective function allows to "minimax" the loss with the consideration of the adversaries.

5.3.3.3 Adversarial Attacks and Uncertainty

The emergent uncertainty studies have discussed the impact of adversarial attacks on model uncertainty. For instance, Smith et al. [193] have demonstrated the mode failures of MC-dropout that are caused by adversaries. These failures are derived from overconfident and certain predictions that cannot be captured by the dropout. The review in [7] has revealed the awareness of previous studies regarding adversarial attacks in uncertainty. For instance, Bradshaw et al. [33] have proposed a hybrid model of GP and DNN which is robust to adversarial attacks to produce uncertainties. Pawlowski et al. [164] have introduced a new technique of variational approximation that produced competitive accuracies and robustness against adversarial attacks. Unlike previous studies, I perform uncertainty estimation directly by using the adversarial attack method with random class label assumption on the whole data points. This is demonstrated in the upcoming section.

5.3.4 Method

Mainly, the uncertainty of the model falls into two major categories either epistemic or aleatoric uncertainty. Epistemic uncertainty underlies the new instances that are not learned yet by a model. Aleatoric is the noisy observations such as data lying between the overlapping classes. This type of noise is irreducible by acquiring more data in contrast to epistemic uncertainty. Here, I intend to capture uncertainty using adversarial attacks on the test set. For every data point, this is performed by feeding the neural network with multiple perturbed versions of the input linearly back-and-forth in the direction of the decision boundary as illustrated in Figure 5.9. Hence, multiple outputs are produced which inform the tendency of a given data point, after perturbations, to fall in the opposite class. The variants of the produced output introduce the uncertainty at the given point using mutual information. The multiple perturbations are the multiple scaled gradients added to the input data as a function of epsilon. However, in order to apply FGSM, I need the target class in order to maximise or minimise the loss function. Thus, I need to arbitrarily assign a class to compute the gradients.

5.3.4.1 Procedure of MC-AA

Adversarial examples are computed by feeding input data to the neural network and assigning a target class in the loss function. Then, The gradient of the loss function with respect to input is scaled and added as mentioned in Equation 5.8. To perform MC-AA, I apply the following procedure:

1. Assume that all the test set belongs to the same class, e.g. class 0.
2. Perform multiple FGSM on each input for all values of $\varepsilon \in$ I=$\{\varepsilon_{min}, \varepsilon_{min} + \beta, ..., 0, 0 + \beta, ..., \varepsilon_{max}\}$ with $\varepsilon_{max} = -\varepsilon_{min}$ as provided in Algorithm 1, where the given interval is evenly spaced by β and symmetric around zero. One should note that standardisation of the dataset is a necessary step.
3. Compute the mutual information metric to introduce uncertainty on each data point.

If the assumed class is equal to the actual one and ε>0, the output will maximise the loss function so that the perturbed instance will shift towards the incorrect class. In contrast, if ε<0, the output will minimise the loss function and the perturbed instance will shift towards the correct class. Otherwise, the opposite of the preceding statements is true. As a result, the inputs falling near the decision boundaries are firstly targeted to provide fluctuating outputs with various values of ε, and the model classifies them as uncertain referring to Figure 5.9. I further note the data point falling in the overlapping region of different classes as in case

Algorithm 1 MC-AA

Require:

- $\mathcal{M}$: is a neural network that maps the feature vectors to binary predictions(0 or 1).
- J(.,.): is a loss function that requires as input a data point and its label.
- **Input:**
  - **I:** is a discrete interval comprising different values of ε. ε_{max} is a tunable hyper-parameter.
  - x: is a feature vector to be tested.
  - $\tilde{y}$: is an arbitrary output assumed on the input. Here, I assume that all test points are labelled $\tilde{y}$=0.
- **Output:**
  - $\hat{y}$: is the set of outputs for a single data point. $\hat{y} = \{\hat{y}_{\varepsilon_1}, ..., \hat{y}_{\varepsilon_n}\}$, where $\varepsilon_i \in I$.

```
Function
Compute the gradients of the loss function with respect to $x$ as:
grad = $\nabla_x J(x, \tilde{y})$
for all $\varepsilon_i \in I$ do
    $x_{\varepsilon_i}$ = x + $\varepsilon_i$.sign(grad);
    $\hat{y}_{\varepsilon_i} = \mathcal{M}(x_{\varepsilon_i})$ ;
    return $\hat{y}_{\varepsilon_i}$
end for
$\hat{y} = \{\hat{y}_{\varepsilon_1}, ..., \hat{y}_{\varepsilon_n}\}$;
return $\hat{y}$
```

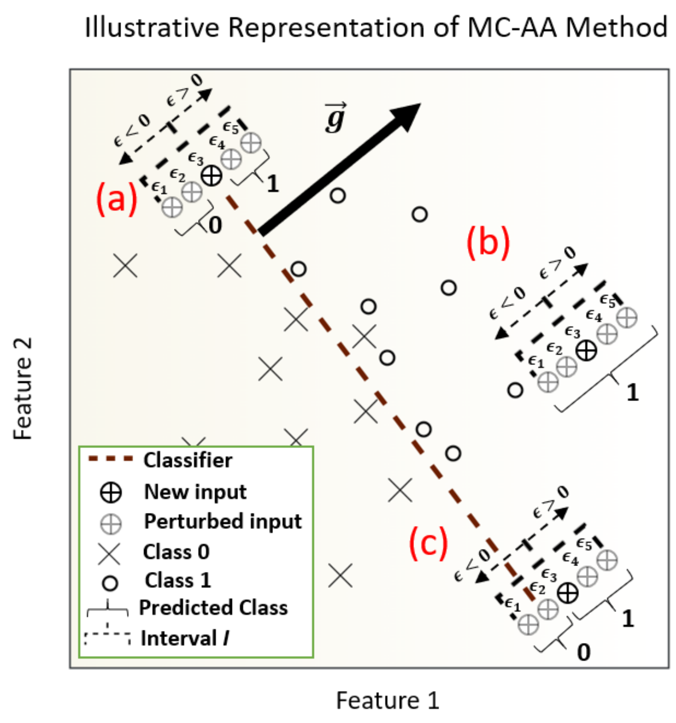


Figure 5.9: Illustrative representation of MC-AA method. Case (a) describes an input lying on the decision boundary. Case (b) describes an input lying completely in its corresponding class 1. Case (c) describes an input with its actual label 0 that lies in the region of class 1 (overlapping region). $\vec{g}$ represents the sign gradient of the loss function with respect to the input.

(c) in Figure 5.9; MC-AA is able to capture such points, unlike the MC-dropout method. MC-dropout performs slight variations on the decision boundary. As this method is based on stochastic forward passes, the variations on the decision boundary may not be able to move around misclassified instances near the decision boundary with highly trained instances of a similar class. In other words, the decision boundary is only able to perform variations with dropout in a critical region. instances with misclassified predictions are more likely to occur near the decision boundary. While MC-AA enforces data points to move back and forth in the direction of the decision boundary regardless of region. This leads any instance near the decision boundary to move between different classes, hence reflecting good uncertainty estimates. This is done at the cost of capturing some correct predictions with uncertainty, but they do not affect the performance of model uncertainty which is favoured. Also, the interval of epsilon is chosen symmetric and evenly spaced to provide fair uncertainty results by the model output. The model output is influenced by the linearly added perturbations (derived from gradients) to its relevant input. Linear perturbations are the simplest and most efficient way to reach the decision boundary in a small range of epsilon. Thus, the evenly spaced and symmetric interval enforces the inputs to move back and forth in an equal fashion with respect to the decision boundary, without any bias to any of the opposite class distributions.

5.3.4.2 Relation to Bayesian Approach

In fact, the proposed method can be viewed as a special case of an approximated Bayesian model. Under required conditions, applying prior to weights in a neural network can be matched with adding perturbations to the inputs. However, this is feasible in a very small interval of ε in the neighbourhood of zero. Let y^* be the observed output corresponding to the input x^* with M dimensions. Then, the mapping function of a neural network can be written as:

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, w)p(w|X, Y)dw, \tag{5.10}$$

where $p(y^*|x^*, w)$ is the likelihood of the model and p(w|X, Y) is the posterior distribution among its weights. I can write an approximation of the model's weight underlying the posterior distribution as: $\hat{w}_{ij} = w_{ij} + \delta w_{ij}$, where δw_{ij} is a small perturbation on the model's parameter which can be equivalent of adding priors to the weights.

For simplicity, consider as an example a neural network with a single output layer. Then the predictive posterior on each neuron c_j can be expressed in its canonical form as:

$$c_j = \sum_{k=1}^{M} x_k.\hat{w}_{kj} = \sum_{k=1}^{M} (x_k.w_{kj} + x_k.\delta w_{kj}), \tag{5.11}$$

where j=1,..., P with P being the number of output neurons. Using MC-AA method, the adversaries over an input are written as: $\hat{x}_i = x_i + \delta x_i$. Similarly, I plug the adversaries into the model's equation as follows:

$$c_j = \sum_{k=1}^{M} \hat{x_k}.w_{kj} = \sum_{k=1}^{M} (x_k.w_{kj} + \delta x_k.w_{kj}). \tag{5.12}$$

Assuming Equation 5.11 is equivalent to 5.12, I obtain the necessary conditions:

$$\frac{\delta x_k}{x_k} = \frac{\delta w_{kj}}{w_{kj}}. \tag{5.13}$$

Generally, the condition in Equation 5.13 safely holds true on very small perturbations. In other words, I can guarantee the satisfaction of this condition by choosing $\frac{\delta x_k}{x_k}$ in a neighbourhood of zero. Consequently, this is true when ε is very small.

5.3.5 Model Uncertainty Using MC-AA

5.3.5.1 Obtaining and Evaluating Model Uncertainty

Firstly, the different values of ε introduce multiple outputs per instance. Similarly to Equation 5.5, the predictive mean can be written as:

$$p_{MC-AA}(y^*|x^*) \approx \frac{1}{T} \sum_{i=1}^{T} \hat{y}^*(x^*_{\varepsilon_i}, W_1, ..., W_L), \tag{5.14}$$

where $x^*_{\varepsilon_i}$ is the perturbed x by FGSM method associated to ε_i, with T being the length of the interval $I = \{\varepsilon_{min}, \varepsilon_{min} + \beta, ..., 0, ..., \varepsilon_{max}\}$, where I is evenly spaced by constant β and $\varepsilon_i \in I$. This interval should be chosen symmetric with respect to 0 with ε_{max} =-ε_{min}, where ε_{max} is a hyper-parameter to be tuned. Referring to [69], Mutual information (MI) is shown to be an effective measurement of uncertainty prediction. Also, it is sometimes referred to as epistemic uncertainty [193]. MI reflects the amount of information of the multiple outputs on each input. Thus, the uncertain prediction requires more amount of information from the model and consequently produces a higher MI. Similarly to MC-dropout in Equation 5.6 thanks to [68], MI can be expressed as:

$$\hat{I}(y^*|x^*, w) = \hat{H}(y^*|x^*, w) + \sum_c \frac{1}{T} \sum_{i=1}^{T} p(y^* = c|x^*_{\varepsilon_i}, w) \log p(y^* = c|x^*_{\varepsilon_i}, w), \tag{5.15}$$

where c is the class label, and

$$\hat{H}(y^*|x^*, w) = -\sum_c p_{MC-AA}(y^* = c|x^*, w) \log p_{MC-AA}(y^* = c|x^*, w). \tag{5.16}$$

The uncertainty of the model can be evaluated using the measurements of a binary classification problem. The uncertainty evaluations provide a similar approach to using Area-Under-Curve (AUC) score and the Receiver-Operation-Curve (ROC). However, these metrics are accompanied by the ground truth of the labels beside the predictive uncertainty derived from MI as evaluated in [137]. Meaning, the output measurements of uncertainty estimates are derived from correct/incorrect classification tied with certain/uncertain predictions. As a result, I can distinguish between four possible states regarding uncertainty measurements as follows:

- TN (Correct and certain): The predictions match the labels and the mutual information is low
- FP (Correct and uncertain): The predictions match the labels but the mutual information is high
- FN (Incorrect and certain): The predictions do not match the labels but the mutual information is low
- TP (Incorrect and uncertain): The predictions do not match the labels and the mutual information is high

Model Uncertainty	Certain: $MI < T_u$	Uncertain: $MI \geq T_u$
Correct	True Negatives: TN	False Positives: FP
Incorrect	False Negatives: FN	True Positives: TP

Table 5.2: Model uncertainty states: This table shows the four possible states of model uncertainty to classify every test sample. This evaluation resembles a binary classification task.

The uncertainty mapping (certain/uncertain) is derived from the predictive uncertainty measurement. After computing these measurements, I set an uncertainty threshold, T_u. This threshold moves between the minimum and the maximum predictive uncertainty estimates in the test set. The correct/incorrect and certain/uncertain maps correspond to a new binary classification task as shown in Table 5.2. From the abovementioned states, true positive (TP), false positive (FP), true negative (TN), and false negatives (FN) values reflect the performance of the uncertainty classification task. These terms yield the following expressions that reflect the goodness of uncertainty estimates:

1. Accuracy of uncertainty estimation:

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{5.17}$$

2. Negative Predictive Value (NPV): I desire that if the model is certain about its prediction, the prediction is assumed to be correct. This can be reformulated as conditional probability:

$$p(correct|certain) = \frac{p(correct, certain)}{p(certain)} = \frac{TN}{TN + FN} \tag{5.18}$$

   which is equivalent to the NPV ratio in the binary test.

3. True Positive Rate (TPR): If the model is incorrect about its predictions, I desire to be uncertain about the predicted value. This is expressed as a conditional probability:

$$p(uncertain|incorrect) = \frac{p(uncertain, incorrect)}{p(incorrect)} = \frac{TP}{TP + FN} \tag{5.19}$$

   which is equivalent to the *recall* metric used in the binary test.

The abovementioned metrics reflect a good uncertainty performance of the model where better performance corresponds to the higher values of these metrics. These metrics are studied according to the variation of the uncertainty threshold T_u. As the threshold varies between the min/max uncertainty values, ROC and AUC are hence computed, in which I can easily derive a summarisable performance of the metrics. Instead of changing the threshold between min/max uncertainties, I normalise this threshold with respect to the uncertainty, where $T_u \in [0, 1]$. Hence, the expression of the threshold can be written as: $T_u = \frac{u - u_{min}}{u_{max} - u_{min}}$, where u_{max} and u_{min} correspond to the maximum and minimum values of the predictive uncertainty measurement over the data, respectively.

5.3.5.2 Behaviour of MC-AA

To illustrate the behaviour of MC-AA, I generate 2D synthetic data of 12,000 instances by sampling from random normal distribution on each dimension with a common standard deviation of 0.25 and averages of values 0 and 1 corresponding to classes 0 and 1, respectively, referring to Figure 5.8. I fit this data into a neural network comprised of two hidden layers

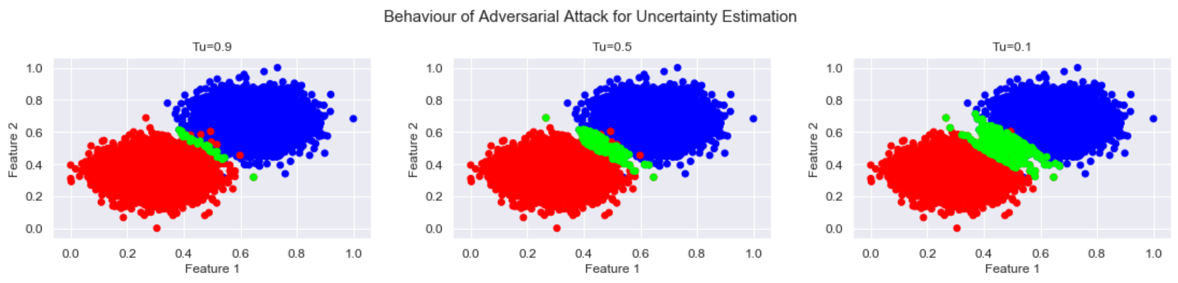


Figure 5.10: Representation of MC-AA behaviour on 2D synthetic data. Each subplot highlights the uncertain data points that fall above the provided threshold T_u.

with 20 neurons each chosen arbitrarily. The outputs are then squashed with the softmax function to produce the output classes. I use the NLLLoss function and Adam optimiser, then I arbitrarily set the learning rate to 0.01 and the number of epochs to 100. I assume that all labels belong to class 0. Hence, I apply the MC-AA method using different values of ε that belong to a symmetric interval with $\varepsilon_{max} = -\varepsilon_{min} = 5\text{x}10^{-3}$ chosen arbitrarily, wherein the interval values are evenly spaced by β which is arbitrarily set to $\frac{\varepsilon_{max}}{10}$. Hence, only a few output samples are required to produce uncertainty estimates using MI. Referring to Figure 5.10, I highlight the captured uncertainty in green, using different thresholds T_u. This method behaves to some extent as an MC-dropout that captures epistemic and aleatoric uncertainty of the data points near the decision boundary [14]. These data points lack knowledge about the model's decision. Since the perturbations directly influence the input data, MC-AA is more robust to wrong labels lying between the overlapping regions (aleatoric uncertainty) where the recent uncertainty methods have failed to detect them. This will be further clarified in the next sections.

5.4 MC-AA vs Other Uncertainty Methods Using Bitcoin and Ethereum Datasets

To test the performance of the MC-AA method, I choose two datasets derived from Bitcoin and Ethereum blockchains. I test the performance of MC-AA on these datasets. Then, I provide a fair comparison between my method (MC-AA) with the previous methods such as MC-dropout, DUQ and DUE using the same experimental setup on each of the datasets. In these experiments, I use Pytorch [163] and GPytorch [72] packages in Python programming language. I further divide this section into two parts, wherein I test the proposed method with Bitcoin and Ethereum datasets in the first and second parts, respectively.

5.4.1 Experimenting with Bitcoin Data

5.4.1.1 Experiments

Using the Elliptic dataset, the primary 34 graphs correspond to the train set, and the 15 remaining graphs correspond to the test set, whereas the last 5 graphs of the train set are used as a validation set. Afterwards, I exclude the timestamp feature and standardise the dataset in my experiments. The nodes of each timestamped graph are considered as a batch in this experiment. Using this data, I apply MC-AA, MC-dropout, DUQ and DUE each using a similar neural network as a base network consisting of two hidden layers of widths n_1=100 and n_2=81, chosen empirically by manual searching, and squashed by ReLU function. However, each of the uncertainty methods is associated with its corresponding experimental settings, which are provided in what follows.

MC-AA: Here, the base network is followed by the softmax output layer to produce the binary licit/illicit predictions. I empirically assign 0.3/0.7 to the weights of the NLLLoss function to mitigate the class imbalance. I use an Adam optimiser with a learning rate that is empirically set to 0.01. After training the model, I perform MC-AA on the test set to capture the model uncertainty using multiple forward passes. Beforehand, all test points are assumed to be in the licit class. Using the best AUC-score of model uncertainty on the validation set, the hyper-parameter ε_{max} is empirically tuned to be 0.1. I limit the number of samples per data point to 20 by setting β arbitrarily to $\frac{\varepsilon_{max}}{10}$ Then, the predictive uncertainty is obtained using MI over the provided samples.

MC-dropout: Like the preceded model in MC-AA, I use a similar neural network but with a dropout function after every hidden layer. I empirically choose dropout equals 0.3. Then, I arbitrarily perform 50 stochastic forward passes on the test set to produce uncertainty estimates using MI.

DUQ: This model comprises a feature extractor and an additional learnable layer known as the RBF kernel with a two-sided gradient penalty to produce reliable uncertainties with a single forward pass [210]. Here, the feature extractor is the given base network to produce a fair comparison. The output layer (RBF kernel) is formed of a learnable weight matrix per class in order to compute the distance from the data points to the centroids of the different classes. So, the weight matrix embeds the output of the feature extractor in a new space of embedding size (centroid size) set by default to 10. The centroid of each class is updated by an exponential moving average of the features corresponding to the data points in accordance with that class. The gradient penalty is the l_2 norm of the gradient's output with respect to the input bounded by the Lipschitz constant of 1 as introduced in [210]. This penalty incorporates a regularisation factor λ. The binary-cross-entropy loss function is used to minimise the distance to the correct centroid and maximise the others. Using Bitcoin data, I empirically set the hyper-parameters of DUQ as σ of RBF kernel equals 0.3, λ=0.1, γ as a factor of moving average over centroids is set to 0.9 and learning rate equals to 0.001.

DUE: This model comprises a feature extractor that is followed by Gaussian Process (GP)

[209]. I use the same feature extractor as preceded. The model arbitrarily utilises 15 inducing points that are initialised equally to the centroids by clustering the feature extractor output using the K-means algorithm with 15 clusters. Then, the output of the feature extractor is fed to a sparse GP kernel with the variational inducing points that are used to approximate the full dataset by maximising a lower bound on the marginal likelihood known as ELBO (evidence lower bound). This model produces various predictions using the softmax likelihood function by a single forward pass, wherein the predictive uncertainty is then computed via MI. Using Bitcoin data, I empirically obtain the hyper-parameters settings as follows: learning rate=0.01 and epochs=50.

5.4.1.2 Discussions

We plot and compare the different uncertainty measurements on the above-mentioned methods as shown in Figures 5.11 and 5.12. Remarkably, the model uncertainty based on MC-AA has shown superior performance in comparison to other uncertainty methods. Referring to Figure 5.11, NPV and TPR measurements have revealed a remarkable improvement using MC-AA, whereas the accuracy curve has shown acceptable but lower performance. This means that the method is able to capture more true positives (incorrect and uncertain) and fewer false negatives (incorrect and certain) at the cost of having more false positives (correct and uncertain). This scenario is more favoured in model uncertainty because being certain on incorrect predictions affects the integrity of the model. Whereas, the uncertain and correct predictions can be forwarded to an annotator for further analysis. In addition, the goodness of classification of model uncertainty has recorded the highest AUC score equal to 0.8 with MC-AA, whereas others have recorded an AUC score of less than or equal to 0.75. Also, the Precision-Recall curve has admitted a significant outperformance using my proposed method as shown in Figure 5.12.

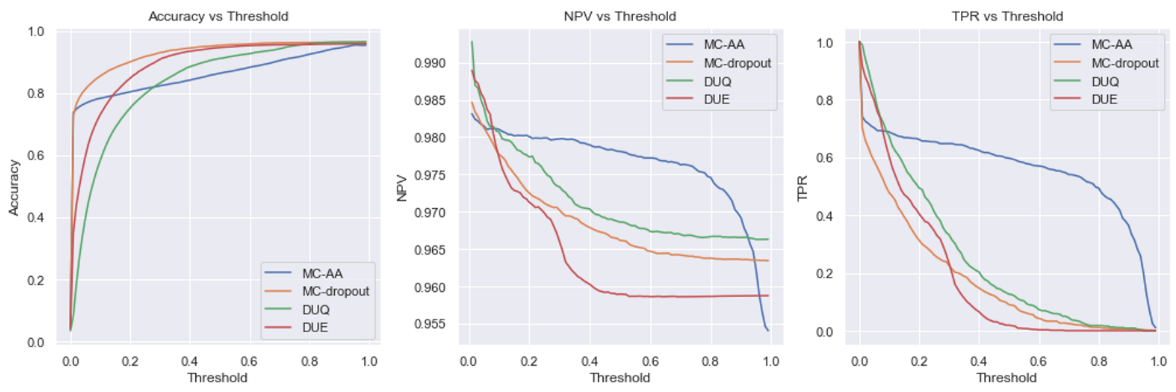


Figure 5.11: Comparison of model uncertainty using different uncertainty methods in Bitcoin data. The subplots (from left to right) correspond to accuracy, NPV and TPR as a function of threshold T_u.

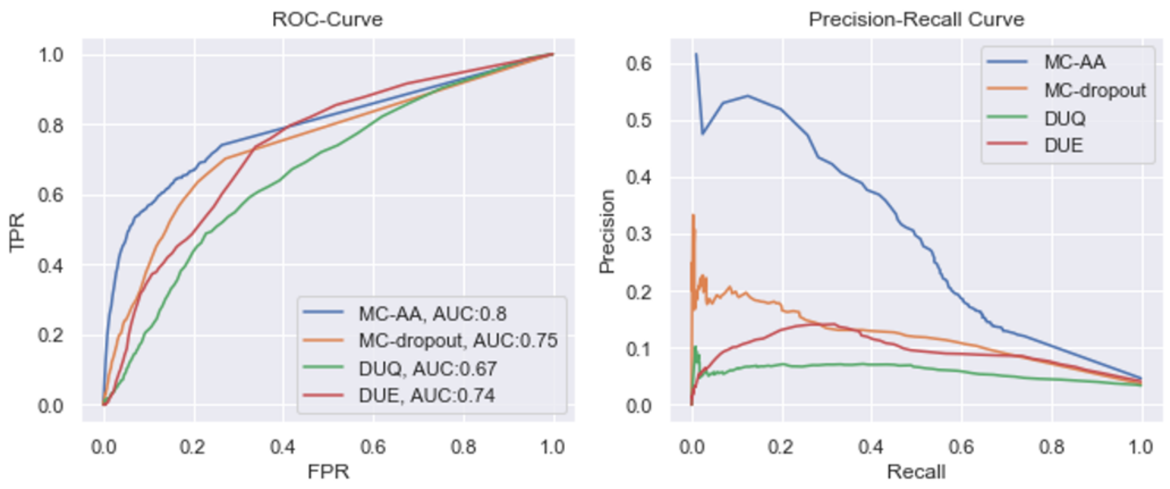


Figure 5.12: ROC-curve and Precision-Recall as a function of threshold T_u of model uncertainty using Bitcoin data. The model uncertainty is performed using MC-AA (Ours), MC-dropout, DUQ and DUE.

5.4.2 Experimenting with Ethereum Data

5.4.2.1 Experiments

Using the exact data preprocessing of the Ethereum dataset in Chapter 4, the feature space is reduced to 28 dimensions and the data is randomly split to train/validation/test by the ratios 0.7/0.1/0.2, wherein the features are standardised. Subsequently, I apply MC-AA, MC-dropout, DUQ and DUE each using a neural network as a base network consisting of two hidden layers of widths n_1=50 and n_2=25 chosen empirically by manual searching, and squashed by the ReLU function. In the following parts, I provide the necessary experimental setup for each of the uncertainty methods.

MC-AA: I apply the softmax function to output the non-fraud/fraud predictions on the output layer of the mentioned base network. I use the weighted NLLLoss with weights 0.4/0.6, chosen empirically. The number of epochs and the learning rate are empirically set to 50 and 0.01, respectively. The batch size is arbitrarily set to 512. Using MC-AA on the trained model, I assume that all testing points belong to class non-fraud. I empirically choose the hyper-parameter ε_{max} equals to 8.1x10^{-4} that provides the highest AUC score on the validation set. I set the number of samples per data point to 20 by setting β arbitrarily to $\frac{\varepsilon_{max}}{10}$.

MC-dropout: Similarly to the preceded model in MC-AA, I apply an additional dropout function after every hidden layer of empirically chosen dropout ratio equals 0.3. Then, I perform 50 stochastic forward passes on the test set, in which the uncertainty estimates are computed through MI.

DUQ: I use the base network as the feature extractor for DUQ. For this dataset, I empirically opt the following values for hyper-parameters: epochs=50, learning rate=0.001, σ= 0.3, λ=0.01, γ=0.9 and batch size=512.

DUE: With Ethereum data, I use the base network as a feature extractor and I empirically choose the following settings: epochs=50, learning rate=0.01 and batch size=512.

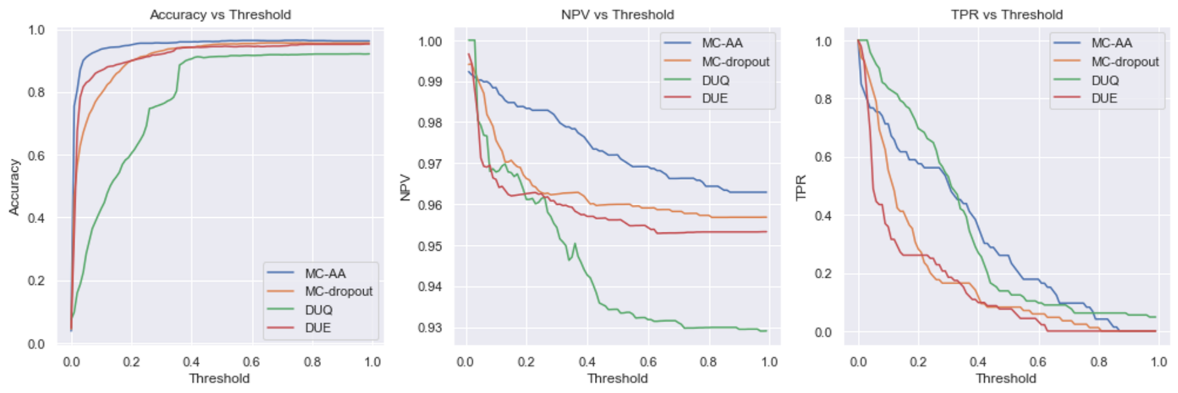


Figure 5.13: Comparison of model uncertainty using different uncertainty methods on Ethereum data. The subplots (from left to right) correspond to accuracy, NPV and TPR as a function of threshold T_u.

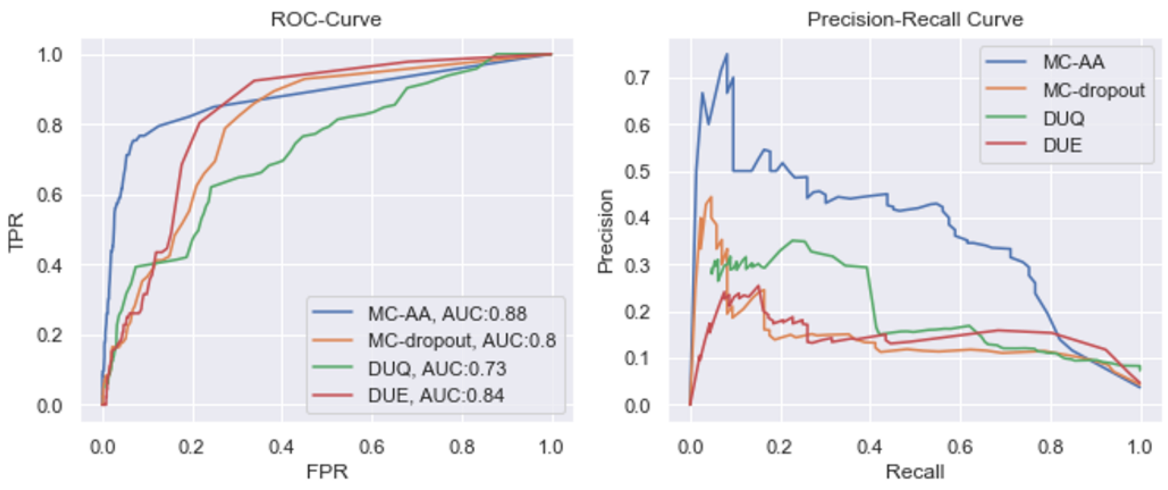


Figure 5.14: ROC-curve and Precision-Recall as a function of threshold T_u of model uncertainty using Ethereum data. The model uncertainty is performed using MC-AA (Ours), MC-dropout, DUQ and DUE.

5.4.3 Discussions

With the Ethereum dataset, I follow the same procedure to evaluate and compare the MC-AA method in comparison to previous uncertainty methods. Referring to Figures 5.13 and 5.14, the model uncertainty based on MC-AA has outperformed the other mentioned methods with an AUC-score equals to 0.88. The performance of the MC-AA method is tied to the tuned hyper-parameter ε_{max}. It is feasible to force the "incorrect and certain predictions" to be lower by increasing the value of this hyper-parameter but at the cost of other measurements. MC-AA can be also viewed as a generalised form of the standard neural network model and can be approximated with MC-dropout for a certain dropout value and ε_{max}. However, MC-AA has shown to be more favoured than MC-dropout since I have higher guidance and influence on the uncertainty performance in accordance with the classification problem. For instance, the changes in ε_{max} of MC-AA are tied with perturbed inputs in which I can force the model not to tolerate data points lying in the wrong class. While in MC-dropout, the changes in the dropout factor are tied with the perturbation of the decision boundary, wherein the model cannot be uncertain about data points in the wrong class. In general, MC-AA is a viable approach in binary classification tasks and is not limited to the used datasets. On the other hand, dealing with multi-class classification is not applicable with this method unless the classifier is converted into *one versus all* classification. According to the results, MC-AA has shown to be more efficient than MC-dropout as my proposed method has produced uncertainty estimates with 10 forward passes whereas the latter method has performed 50 forward passes to provide stable uncertainty estimates using mutual information. Meaning, the method is capable of achieving good uncertainty estimates with 10 back-and-forth movements using MC-AA.

5.4.4 Concluding Remarks

We have proposed a novel method (MC-AA) based on adversarial attacks to capture model uncertainty in binary classification tasks. I have shown that this method provides reliable uncertainty estimations with a reduced number of misclassified and certain predictions. I have compared the performance of MC-AA with previous methods. My model has outperformed the recent studies using datasets derived from Bitcoin and Ethereum blockchains.

5.5 MC-AA vs MC-dropout with GNN Models Using Bitcoin Dataset

The preceded study has shown promising results using MC-AA uncertainty estimation with MLP models. Here, I also validate the effectiveness of the MC-AA uncertainty method on GNN models. I apply MC-AA to capture model uncertainty of various GNN models of node classification to predict licit/illicit transactions of the Elliptic data that is described in Table 5.3.

Besides, I obtain model uncertainty using my method, and then I evaluate and compare it with MC-dropout. My main finding is that the proposed method is effective in capturing the data points in overlapping classes. My method attains significant success over MC-dropout to capture model uncertainty in my experiments.

Graph Network Description	Elliptic data
Directed	Yes
Temporal	Yes
# Nodes	203,769
# Edges	234,355
# Node features	166
Binary Node Labels	Licit/Illicit txs

Table 5.3: Graph description of Elliptic dataset.

5.5.1 Graph Neural Networks (GNNs)

We conducted my experiments using several graph learning models to perform binary node classification on the given dataset. We use PyTorch [163] and PyTorch-geometric package [63] in Python programming language. I choose a set of popular graph learning models as follows:

- GCN: Graph Convolutional Network based spectral approach [107]. The layer of GCN can be written as:
$$x'_i = \Theta \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} x_j \tag{5.20}$$

- GraphConv: Graph Convolutional Network based spatial approach [141]. It is expressed as:
$$x'_i = \Theta_1 . x_i + \Theta_2 \sum_{j \in \mathcal{N}(i)} e_{j,i} . x_j \tag{5.21}$$

- GAT: Graph Attention Network [213]. It is expressed as:
$$x'_i = \alpha_{i,i} \Theta . x_i + \Theta \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} . x_j \tag{5.22}$$

- SAGEConv: Graph SAGE Convolution [80]. It is expressed as:
$$x'_i = \Theta_1 . x_i + \Theta_2 . \mathrm{mean}_{j \in \mathcal{N}(i)} x_j \tag{5.23}$$

- LEConv: Local Extremum Convolution [178].
$$x'_i = \Theta_1 . x_i + \sum_{j \in \mathcal{N}(i)} e_{j,i} . (\Theta_2 . x_i - \Theta_3 . x_j) \tag{5.24}$$

- TAGConv: Topology Adaptive GCN [58]. It is expressed as:
$$x'_i = \sum_{k=0}^{K} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \Theta_k \left(\frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}}\right)^k x_j \tag{5.25}$$

- DeepGCN: Deep Graph Convolutional Network [122]. This model is accompanied by GENConv (Generalised Convolution) in which the layer can be written as:

$$x_i' = \text{MLP}(x_i + \text{AGG}(\{\text{ReLU}(x_j) + e_{ji} + \epsilon : j \in \mathcal{N}(i)\} \tag{5.26}$$

where x_i' is the embedding derived from the input node i in the hidden layer, Θ_k is the learnable weight matrix of the layer k, $e_{i,j}$ is the edge weight which is arbitrarily 1 here, $mean$ is the average over the sum, AGG is the softmax aggregation, MLP is the multi-layer perceptron, d_i is the degree of node i and $\mathcal{N}(i)$ is the set of nodes in neighbouring to node i.

5.5.2 Experimental Settings and Classification Results

Hyper-parameters	**All GNN Models**
Hidden layers size	100
Learning rate	0.01
Dropout	0.2
# Epochs	50
Activation function	ReLU
Output function	softmax
Weighted loss function	NLLLoss

Table 5.4: Hyper-parameters used for GNN models.

Since the Elliptic dataset comprises 166 features which are local and aggregated features, I only use local features (referred to as LF) in my experiments which count to 94. Similar to the preceded experiments, I use the same experimental settings on the Elliptic dataset by dividing the train and test sets following the temporal split of this data in which the first 35 graphs (136,265 nodes) belong to the train set and the remaining graphs belong to the test set. Then, this dataset is followed by a standardisation step.

For all graph learning models except DeepGCN, the models are formed of input and output layers and they are trained in a full-batched fashion in which I empirically chose the hyper-parameters as provided in Table 5.4. Regarding DeepGCN, a linear layer is used at the beginning to map the input linearly to 100-dimensional space on both datasets. Then, this layer is followed by 4 GENConv hidden layers of the same size. The number of MLP layers in each GENConv layer is empirically set to 2. The weights of the loss function are empirically set to 0.3/0.7 for Elliptic data only, to mitigate class imbalance. Furthermore, the dropout function is added after all layers except for the output layer in order to perform MC-dropout. The evaluation of the classification using the given GNN models using the Elliptic dataset is provided in Table 6.1.

Models Using Elliptic	**% Accuracy**	**% Precision**	**% Recall**	**% F_1 Score**
GCN	95.6	73.9	50.1	59.7
GraphConv	95.15	74.3	38.8	51
GAT	95.5	77.1	45.15	56.9
SAGEConv	95.5	76	45	56.5
LEConv	95.9	79.3	49.95	61.3
TAGConv	93.93	52.8	62.3	57.2
DeepGCN	**97.1**	**85.3**	**66.7**	**74.9**

Table 5.5: Test set evaluation of graph models on Elliptic data.

5.5.2.1 Obtaining uncertainty estimates using Elliptic data

After training the models with Elliptic datasets, I apply MC-AA and MC-dropout methods then I compare their performance to the abovementioned GNN models. Using MC-AA as in Algorithm 1, I use a non-weighted loss NLLLoss and assume that all test points belong to class 0 and I arbitrarily set β to $\frac{\varepsilon_{max}}{10}$. The evenly spaced interval I is chosen such that it is bounded by $\varepsilon_{max} = 0.09$ chosen empirically using the highest AUC-score of model uncertainty in the GCN model. For brevity, each of ε_{max} is kept the same for all models. Hence, I perform multiple perturbations (equals to $\frac{\varepsilon_{max}}{\beta}$=10) back-and-forth in the direction of boundary decision. For every data point, I have 10 distinct output predictions that can be used to estimate uncertainty via MI measurement. Regarding MC-dropout, I simply activate the dropout during the testing phase and I arbitrarily perform 50 stochastic forward passes on each input (data point) to produce model uncertainty using MI measurements. We evaluate the performance of model uncertainty for all models using the procedure mentioned earlier in this study after computing TN, FN, FP and TP on the tested samples. I plot the evaluation metrics (NPV, TPR, ROC curves) that reflect the performance of model uncertainty of MC-AA and MC-dropout as a function of an arbitrary uncertainty threshold T_u for all mentioned models as shown in Figures 5.15, 5.16, 5.17, 5.18, 5.19, 5.20 and 5.21.

5.5.3 Discussions

Applying several GNN models on Elliptic graph data, the performance of uncertainty using the MC-AA method has noticeably revealed a superior success over the MC-dropout method. By scanning all figures on the various GNN models, NPV and TPR curves are significantly improved using the MC-AA method in comparison to MC-dropout. This shows that MC-AA has detected more data points that are correct given they are certain. Also, MC-AA has detected more uncertain data points that are incorrect. As a result, the number of FN (incorrect and certain) is further reduced with MC-AA in comparison to MC-dropout. ROC curves in MC-AA have also shown an equal or slightly improved performance to capture model

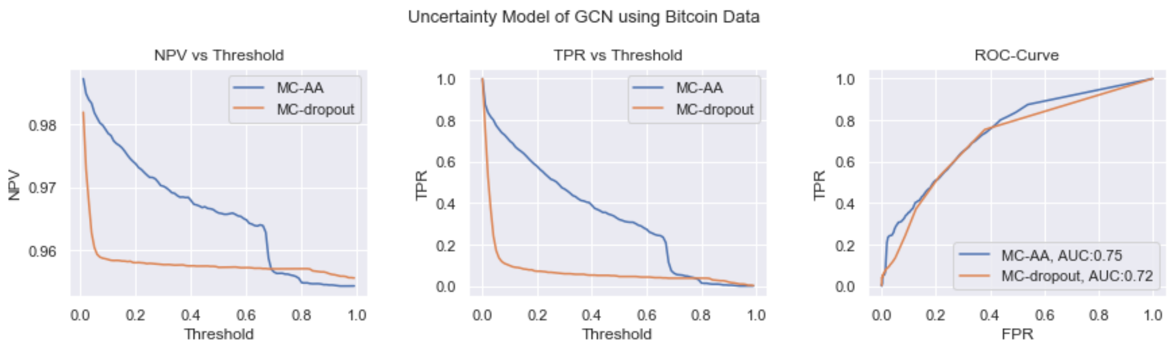


Figure 5.15: Model uncertainty of GCN via MC-AA and MC-dropout using Elliptic data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

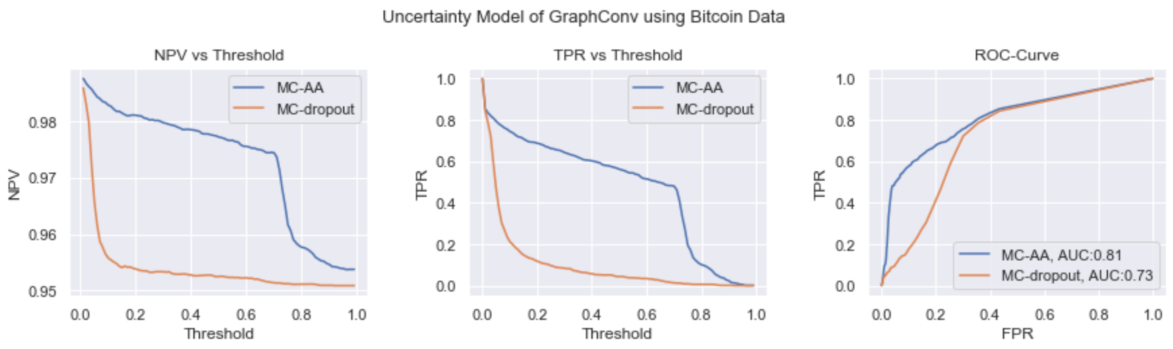


Figure 5.16: Model uncertainty of GraphConv via MC-AA and MC-dropout using Elliptic data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

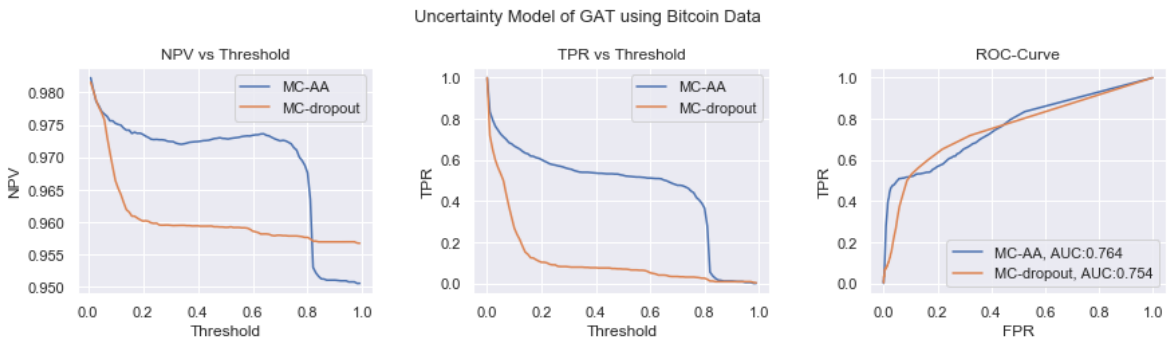


Figure 5.17: Model uncertainty of GAT via MC-AA and MC-dropout using Elliptic data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

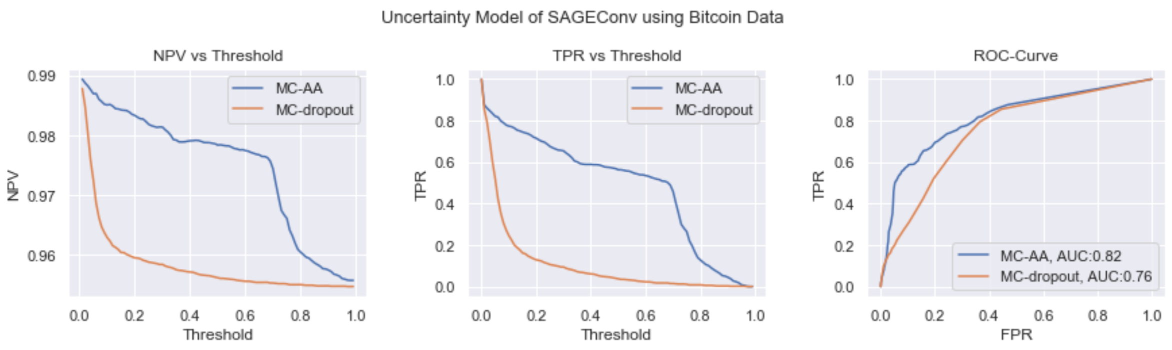


Figure 5.18: Model uncertainty of SAGEConv via MC-AA and MC-dropout using Elliptic data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

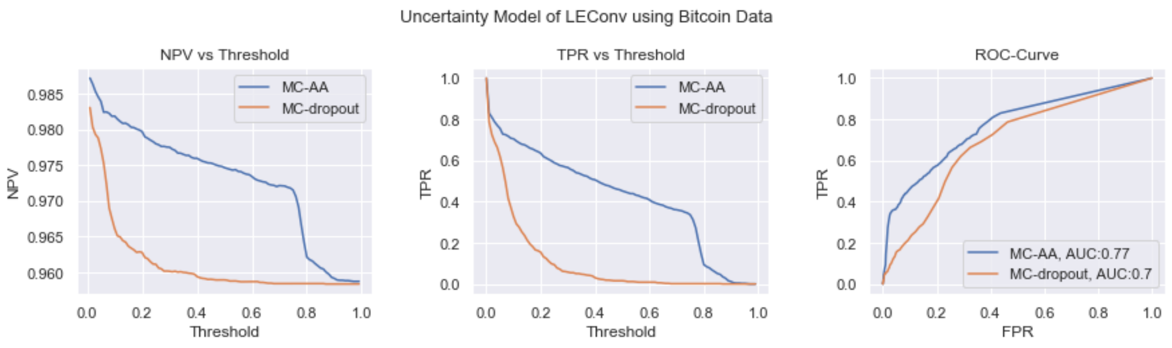


Figure 5.19: Model uncertainty of LEConv via MC-AA and MC-dropout using Elliptic data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

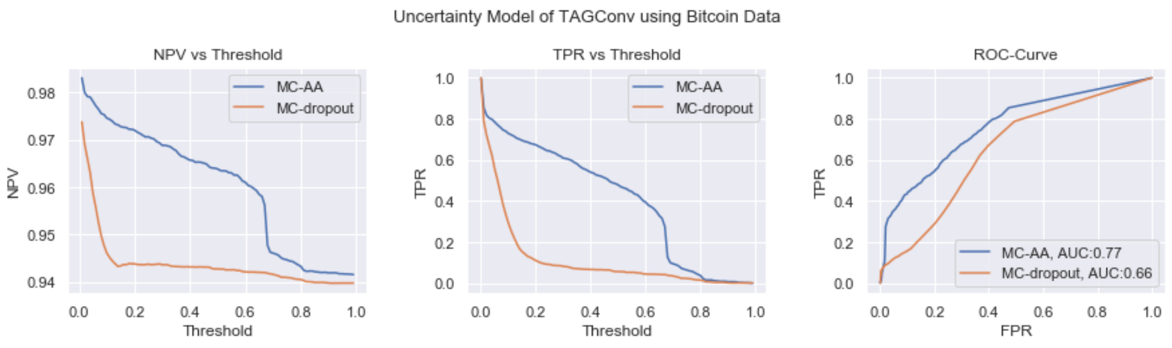


Figure 5.20: Model uncertainty of TAGConv via MC-AA and MC-dropout using Elliptic data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

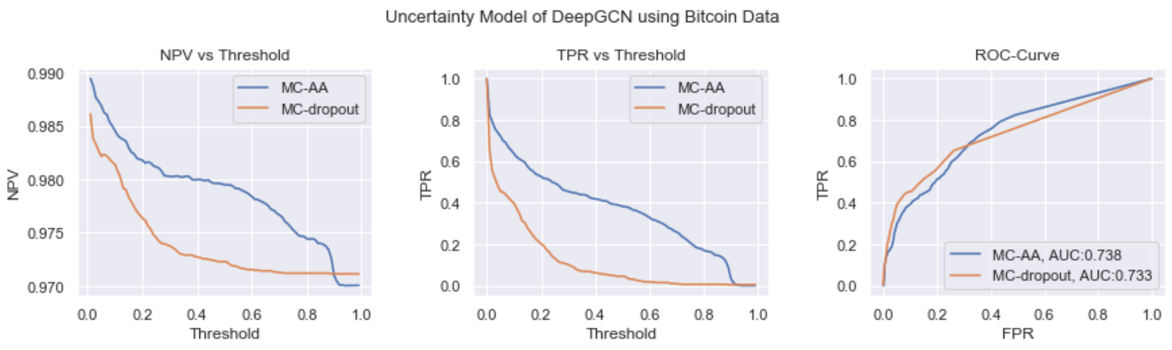


Figure 5.21: Model uncertainty of DeepGCN via MC-AA and MC-dropout using Elliptic data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

uncertainty than MC-dropout. On the other hand, I can notice a slight improvement in AUC scores in MC-AA compared to MC-dropout. This is due to the reduced false negatives, that I desire, at the cost of increasing false positives, resulting in a small change in this score. Meanwhile, false positives (correct and uncertain) do not affect model uncertainty as it is normal to have correct predictions that the model is uncertain about. Since MC-AA applies direct perturbations on the data points, the points lying near the decision boundary or in overlapping classes are included as uncertain. Thus, more points around the classifier are subjected to uncertain predictions that have been deduced by the reduced FN and increased FP.

Generally, MC-AA has attained very reasonable and consistent results in capturing model uncertainty of GNN models. The concept of the MC-AA method is a valid and viable approach which is not limited to the given models. So far, I point out only binary node classification tasks. The node classification results of the provided GNN models using Elliptic data are shown in Table 6.1, in which DeepGCN has performed the best with an accuracy of 97.1% and f_1 score of 74.9%.

5.5.4 Concluding Remarks

We have examined the viability of the MC-AA method to capture model uncertainty in graph neural networks (GNNs). Thus, I have carried out a comprehensive study on several graph-based approaches to real-world graph data known as the Elliptic dataset. Then, I applied MC-AA and compare it with the MC-dropout method. The evaluation of model uncertainty has significantly revealed the outperformance of MC-AA over MC-dropout in all GNN models. MC-AA has shown a great impact in targeting the erroneous data points falling between the overlapping classes. As a result, I have concluded that MC-AA is a viable and effective method to capture model uncertainty in GNN models in binary node classification that is not limited to the preceded dataset.

5.6 MC-AA vs MC-dropout with GNN Models Using GitHub Dataset

Following the same experimental study in the latter section, I validate the effectiveness of the proposed MC-AA using the same GNN models mentioned in List 5.5.1 on another type of graph-structured data – known as GitHub dataset – for binary node classification.

5.6.1 Data Description

GitHub dataset is a large social network of GitHub developers. This undirected graph network was extracted from the public API in June 2019 [182, 5]. The graph network consists of nodes as developers who have starred at least 10 repositories, accompanied by edges as mutual follower relationships between developers. The node features are acquired based on the location, repositories starred, employer and e-mail address. The nodes acquire binary labels, derived from the job title of each user, in order to predict whether the GitHub user is a web or a machine learning (ML) developer. GitHub data description is summarised in Table 5.6. I arbitrarily opt 0.8/0.2 ratio for the train/test split which is followed by a standardisation step.

Graph Network Description	GitHub
Directed	No
Temporal	No
# Nodes	37,700
# Edges	289,003
# Node features	128
Binary Node Labels	Web/ML developer

Table 5.6: Graph description of GitHub dataset.

Models Using GitHub	**% Accuracy**	**% Precision**	**% Recall**	**% F_1 Score**
GCN	87.26	82.65	63.28	71.68
GraphConv	81.87	61.92	74.79	67.75
GAT	86.97	77.1	80.29	71.68
SAGEConv	87.06	80.66	64.74	71.82
LEConv	82.05	64.29	66.4	65.33
DeepGCN	86.12	75.34	67.65	71.29
TAGConv	**87.4**	**80.58**	**66.56**	**72.9**

Table 5.7: Test set evaluation of graph models on GitHub data.

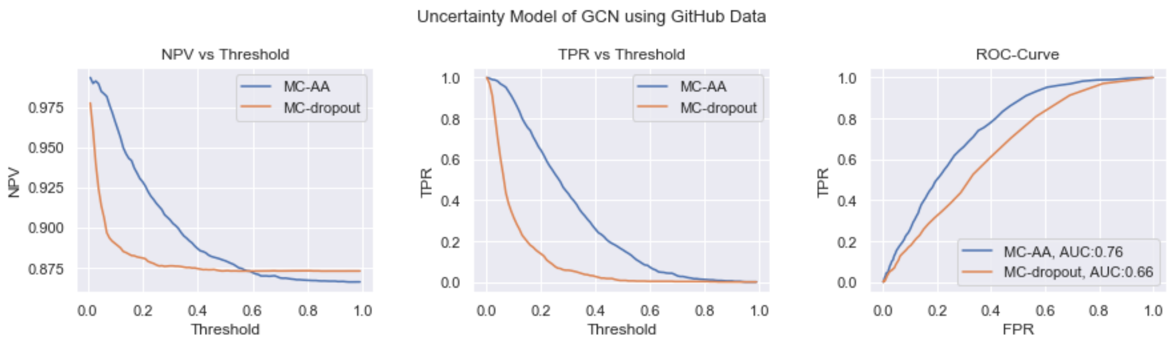


Figure 5.22: Model uncertainty of GCN via MC-AA and MC-dropout using GitHub data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

5.6.2 Experiments

Using the experimental settings as in Section 5.5, I use the same models with the same tuned hyper-parameters as appeared in Table 5.4. However, the only difference is that I use non-weighted loss in this experiment since the dataset is fairly distributed. The classification results of the given GNN models using the GitHub dataset are provided in Table 5.7.

5.6.3 Obtaining uncertainty estimates using GitHub data

Similarly to the preceded procedure, I apply MC-AA and MC-dropout methods on the test set of GitHub data to evaluate the model uncertainty of the given GNN models with the MC-AA method and compare it to that in the MC-dropout method. To perform MC-AA, I choose non-weighted NLLLoss with the test set arbitrarily assumed to belong to class 0 and $\beta = \frac{\varepsilon_{max}}{10}$ by default. Then, I empirically choose ε_{max} to be equal to 0.5 which has achieved the highest AUC score of model uncertainty on the GCN model. For simplicity, this hyper-parameter is assigned equally in the rest of the GNN models. On the other hand, I perform MC-dropout with 50 stochastic forward passes on the test set. The performance of model uncertainty is evaluated for all abovementioned GNN models using the same evaluation metrics as preceded as shown in Figures 5.22, 5.23, 5.24, 5.25, 5.26, 5.27 and 5.28.

5.6.4 Discussions

The evaluations of uncertainty estimations in my experiments have shown that MC-AA outperformed MC-dropout using GitHub dataset using a variety of GNN models. Thus, I still claim the effectiveness of MC-AA on another binary task which is not limited to only blockchain datasets. Some of the results have shown AUC scores for MC-AA slightly less than MC-dropout. However, this difference means that false positives (correct and uncertain) data points increase at the cost of increasing false negatives (incorrect and certain), where the

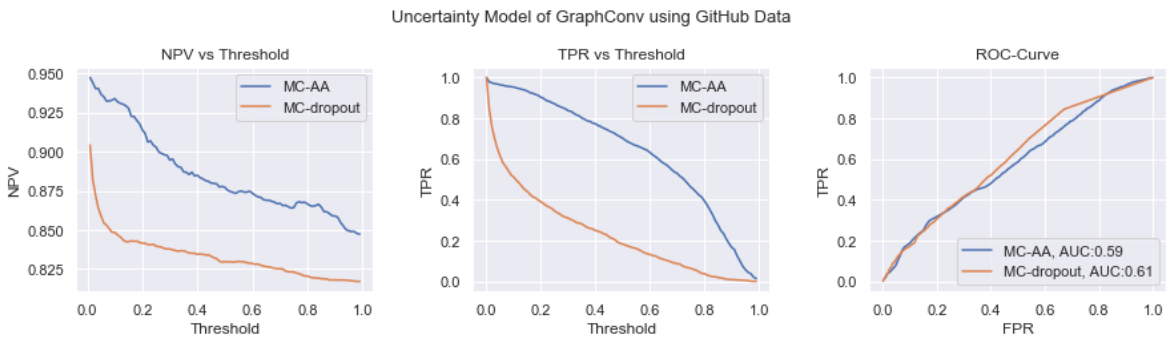


Figure 5.23: Model uncertainty of GraphConv via MC-AA and MC-dropout using GitHub data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

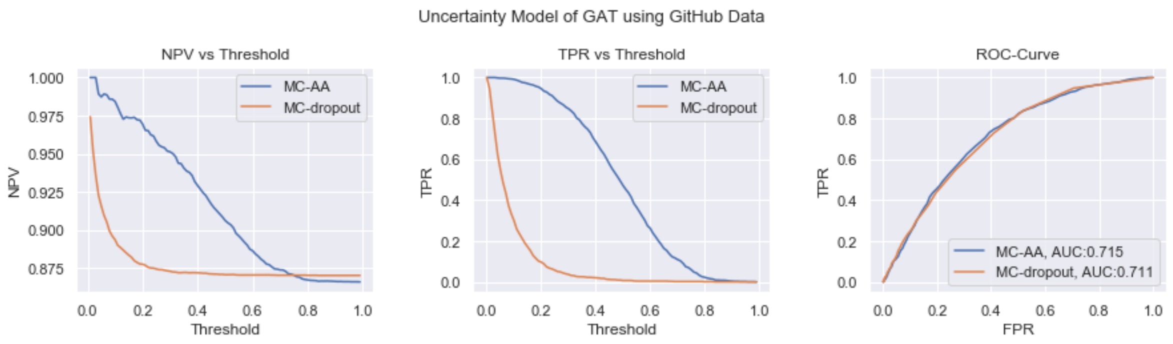


Figure 5.24: Model uncertainty of GAT via MC-AA and MC-dropout using GitHub data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

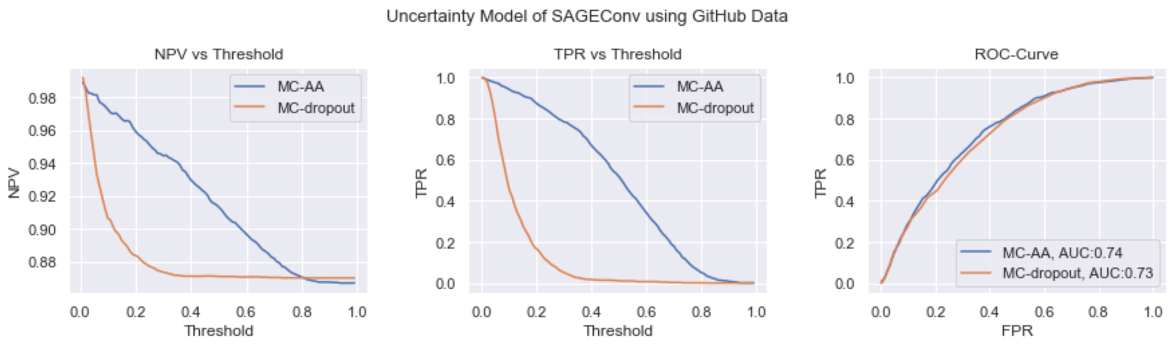


Figure 5.25: Model uncertainty of SAGEConv via MC-AA and MC-dropout using GitHub data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

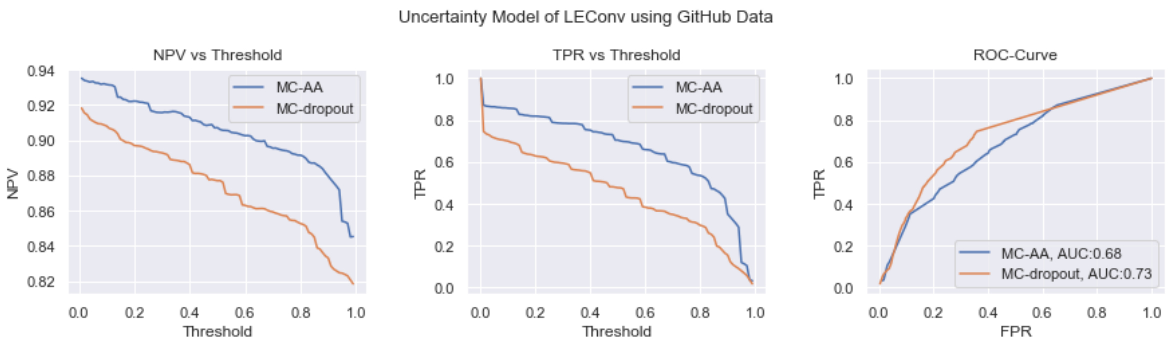


Figure 5.26: Model uncertainty of LEConv via MC-AA and MC-dropout using GitHub data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

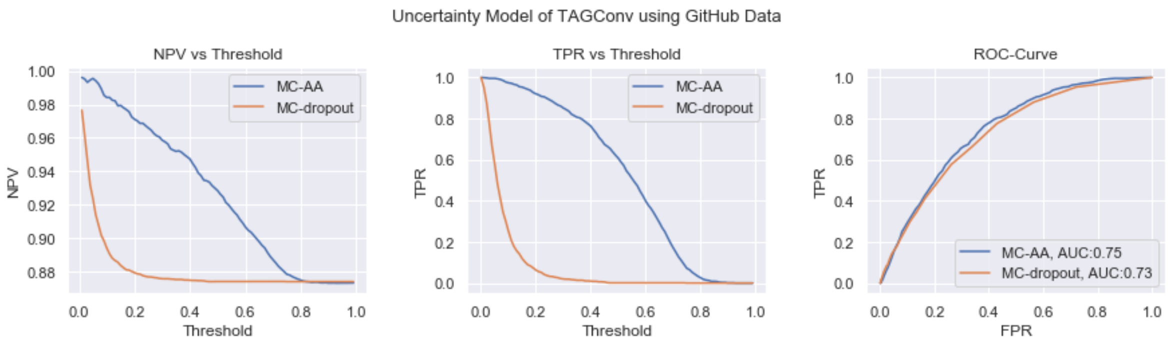


Figure 5.27: Model uncertainty of TAGConv via MC-AA and MC-dropout using GitHub data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

latter measurement directly influences the evaluated uncertainty metrics. Moreover, I desire false negatives to be as low as possible in order to obtain reliable uncertainty estimates.

I also highlight the node classification results, referring to Table 5.7, in which TAGConv has performed the best in binary node classification of the GitHub dataset with an accuracy of 87.4% and f_1 scores of 72.9%.

5.6.5 Concluding Remarks

Using the GitHub dataset, the MC-AA uncertainty method has efficiently captured the model uncertainty in GNN models where MC-AA outperforms the MC-dropout method. The GitHub dataset is a graph-structured dataset of social network developers. The aim of this experiment is to show that the performance of MC-AA is not limited to the data derived from the blockchain. In the next experiment, I point out the application of uncertainty using MC-AA with multi-class classification problems.

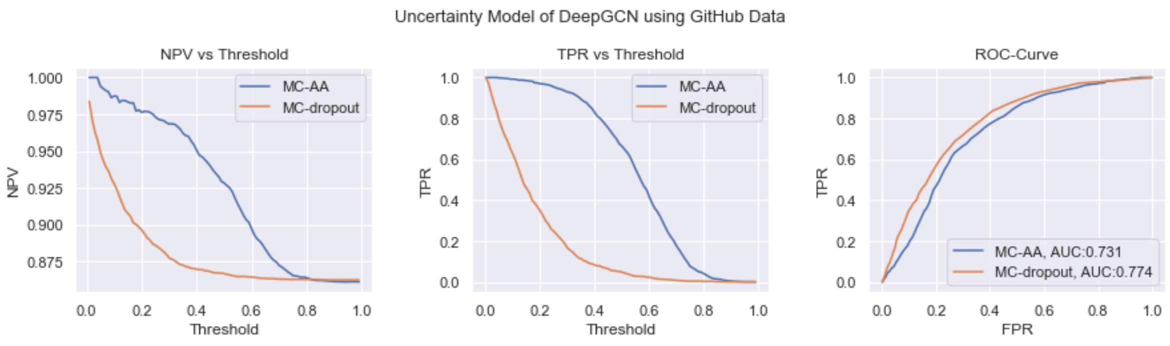


Figure 5.28: Model uncertainty of DeepGCN via MC-AA and MC-dropout using GitHub data. The subplots (from left to right) correspond to NPV, TPR and ROC-curve as a function of threshold T_u.

5.7 MC-AA in Multi-Class Classification: Experimenting with GNN models using Cora and MNIST datasets

5.7.1 Abstract

MC-AA produces uncertainties by performing back-and-forth perturbations of a given data point towards the decision boundary using the idea of adversarial attacks. Despite its efficacy against other uncertainty estimation methods, this method has been only examined on binary classification problems. Thus, I present and examine MC-AA with multi-class classification tasks. I point out the limitation of this method with multiple classes which I tackle by converting the multiclass problem into a *one-versus-all* classification. I compare MC-AA against other recent model uncertainty methods on Cora – a graph-structured dataset – and MNIST – an image dataset. Thus, the conducted experiments are performed using a variety of deep learning algorithms to perform the classification. Consequently, I discuss the best results of model uncertainty with Cora data using the LEConv model of AUC-score 0.889 and MNIST data using CNN of AUC-score 0.98 against other uncertainty estimation methods.

5.7.2 Introduction

Machine learning applications have been exhaustively attracting the interest of many around the globe with various applications such as healthcare [84, 59, 15], blockchain [221, 13], cybersecurity [83, 51], and self-driving cars [167, 136]. On the other hand, machine learning models accompanied by softmax outputs often produce overconfident predictions which lead to poor decision-making regardless of the models' performance. In Figure 5.29, an image of digit four in the MNIST dataset is predicted to be nine with high confidence. Indeed, there is no doubt that this image is confusing to humans as well between these two numbers wherein the machine learning confidently misclassifies the given example. There are more serious scenarios that provide erroneous and confident predictions leading to more cataclysmic decisions such as the case of self-driving cars [4]. In May 2016, a fatal accident is caused by the autopilot feature of a Tesla Model S, which failed to discriminate the white tractor-trailer against a bright sky. Furthermore, other applications are also subjected to misleading predictions such as misclassifying legitimate transactions as illegal in anti-money laundering [12]. These faults caused by the learning model should be detected, analysed and avoided. Consequently, model uncertainty is unavoidably needed besides the model's predictions to provide reliable decision-making.

Gal et al. [68] have tackled the computational complexity of Bayesian neural networks by proposing Monte-Carlo dropout – a Bayesian approximation method – which efficiently produces uncertainty estimates in machine learning models. A comprehensive review covering the recent advances in uncertainty quantification methods is carried out by Abdar et al. [7]. Concisely, uncertainty quantification methods have appeared to detect data points that the model is not trained on such as in deterministic uncertainty quantification (DUQ) [210] or to capture data points that fall near the decision boundary such in Deep Ensembles [114] and Monte-Carlo dropout (MC-dropout) [69]. DUQ reflects the out-of-distribution tested points which appear far from the trained data. Deep Ensembles and MC-dropout are commonly based on an ensemble of models to produce uncertainty. However, the former method is a combination of models with different initialised parameters. Whereas the latter is an ensemble of models with shared parameters by the means of the dropout. Despite their efficiency in modelling uncertainty, these methods have revealed a significant drawback [14, 8]. It is identified by the failure of these methods in detecting the points that fall into overlapping regions. Henceforth, the data points falling in the overlapping regions of multiple class distributions cannot be influenced by the variability of the decision boundary referring to [8]. The latter study has introduced the Monte-Carlo adversarial attack (MC-AA), an uncertainty method that provides perturbations on the given data point in the direction of the decision boundary. The perturbed inputs are computed using the adversarial attack method where multiple perturbed input samples are linearly generated. Consequently, these samples produce uncertainty in the same experimental procedure as MC-dropout. MC-AA has shown its capability in capturing the data points that MC-dropout has failed to capture by forcing the data points to travel between two given class distributions. Despite its promising performance, MC-AA has been only studied with binary classification problems.

5.7.3 Motivations and Challenges

Motivated by previous work in [8], I present a modified MC-AA method that can be applied to multi-class classifications. MC-AA in multi-class classification is a challenging problem because there exist multiple decision boundaries. In other words, it is not clear in which directions the multiple back-and-forth perturbations on a given data point should be performed. Here, MC-AA performs multiple perturbations in a back-and-forth fashion towards the decision boundary that is associated with the predicted class on a given data point. Subsequently, I conduct my experiments using a variety of deep learning models on Cora and MNIST datasets as a type of multi-classification problem. Then, I evaluate and compare the performance of the model uncertainty using MC-AA against recent uncertainty estimation methods. On the

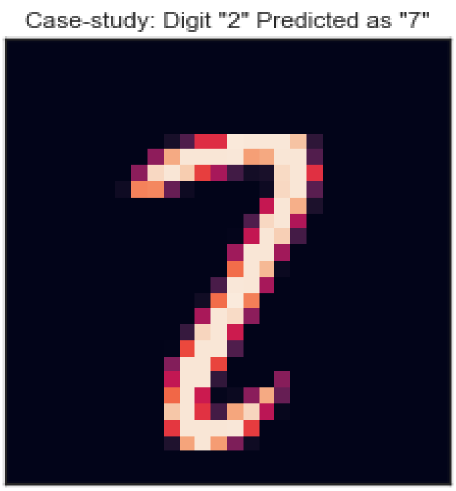


Figure 5.29: Example of digit two image in MNIST data.

other hand, the presented MC-AA has revealed limitations in some cases, which I discuss in the experiments. I effectively solve these limitations by converting multi-class classification to *one-versus-all* classification. Admittedly, I show the competence of the presented MC-AA in capturing uncertainty against other uncertainty methods on the given datasets.

To wrap up, the achievements in this study are as follows:

- I present a generalised MC-AA that performs multiple perturbations in a back-and-forth fashion towards the decision boundary that is associated with the predicted class on a given data point. The generalised MC-AA is slightly different from the MC-AA proposed earlier in Algorithm 1.
- I conduct my experiments using a variety of deep learning models on Cora and MNIST datasets as a type of multi-classification problem.
- The generalised MC-AA has revealed some limitations. I effectively overcome these limitations by converting multi-class classification to *one-versus-all* binary classification.
- I evaluate and show the success of the presented MC-AA in capturing uncertainty over other recent uncertainty estimation methods.

In what follows, I present the generalised MC-AA with multi-class classification tasks. Then, I provide the conducted experiments and results. Lastly, a discussion and a conclusion are stated, respectively.

5.7.4 Background

Primarily, neural networks with distributions over the weights have emerged as Bayesian neural networks (BNNs) that have been studied by Neal et al. [148, 147] and by Mackay et al. [130]. Subsequently, BNN models have witnessed a resurgence in recent years referring to [77, 28, 88]. Admittedly, BNNs have revealed significant success in modelling the uncertainty of neural networks. However, this approach is subjected to prohibitive computational cost referring [68]. Consequently, Gal et al. [69] have proposed the MC-dropout method as an approximation of the Bayesian approach which uses dropout as variational inference.

MC-dropout is a simple and efficient method that uses dropout [196] after each hidden layer to produce uncertainty estimates in neural networks [102]. To produce uncertainty, this is performed using dropout during the testing phase [69]; thus, a data point is subjected to multiple stochastic versions of the perturbed decision boundary which reflects the uncertainty about its predictions. Subsequently, an adaptive version of MC-dropout has appeared in [70] where the dropout parameter is optimised with respect to a given objective function. *Deep Ensemble* method is another Bayesian approximation which utilises an ensemble of multiple neural network models with different initialisations [114]. The study in [159] has shown that the Deep Ensemble method has superior success over BNNs. However, this method has shown poor performance on a simple 2D synthetic data [210]. The latter study has introduced the deterministic uncertainty quantification (DUQ) model which reliably captures the out-of-distribution data – i.e., data points distant from the trained data. DUQ is a deep model that learns feature representations in which the distance between these features and centroids derived from the training data is assessed using a kernel function. This model that uses radial basis function (RBF) kernel is known as *RBF network* [117]. Other uncertainty estimation methods also exist such as using an approximated variational inference by Gaussian processes in [209], dropconnect as another version of MC-dropout [138] and uncertainty estimation based on evidential deep learning [216]. In particular, MC-dropout and Deep Ensemble methods seek to perturb the decision boundary between different class distributions where they have revealed promising results in capturing model uncertainties. However, these methods have failed to capture data points that fall in the overlapping region of class distributions. This issue has been tackled in [8] by proposing MC-AA. MC-AA is an uncertainty estimation method that uses an adversarial attack idea to perform back-and-forth perturbations of a given data point toward the decision boundary. Primarily, adversarial attacks have been extensively discovered in various aspects of machine learning such as in [118] to improve classification, and in [191, 219] to defend against adversarial examples. However, MC-AA is used to spot data points lying near the decision boundary of neural network models, wherein noisy points can be detected with high uncertainty. MC-AA has revealed a significant outperformance over other methods in producing reliable predictive uncertainties in binary classification problems [8]. Thus, I conduct experiments on multiclass classification datasets using various deep learning models to capture model uncertainty using MC-AA. Furthermore, I compare the model performance against MC-dropout, DUQ and Deep Ensemble methods.

5.7.5 Methods for Quantifying Uncertainty

In this section, I present the methods used in my experiments to quantify the uncertainty of deep learning models.

5.7.5.1 Monte-Carlo based Adversarial Attack: MC-AA

For multiclass classification, MC-AA is modified by assigning the class predictions for FGSM as provided in Algorithm 2. This modification takes into consideration the multiple classes where the given input is perturbed towards/away from the direction of its predicted class regardless of other classes. From Equation 5.14, I state the predictive mean as:

$$p_{MC-AA}(y^*|x^*) \approx \frac{1}{T}\sum_{i=1}^{T} \hat{y}^*_{\varepsilon_i}.$$

5.7.5.2 Monte-Carlo Dropout

The dropout is activated during the testing phase which is applied after each weight layer in the neural network. I state the predictive mean of MC-dropout, referring to Equation 5.5 as:

$$p_{MC-dropout}(y^*|x^*) \approx \frac{1}{T}\sum_{t=1}^{T} \hat{y}^*(x^*, W_1^t, ..., W_L^t).$$

5.7.5.3 Deterministic Uncertainty Quantification (DUQ)

DUQ consists of a feature extractor as a base model followed by an additional learnable layer to obtain the feature vectors corresponding to each class. The predictions are performed by computing a kernel function. The kernel function is the RBF kernel which computes the distance between the feature vectors and the centroids. The centroid of each class is updated using an exponential moving average of the feature vectors of the data points corresponding to the class with momentum γ. The predictive uncertainty is obtained in a single deterministic forward pass. The output of a DUQ model can be expressed, referring to [210], as:

$$K_c(f_\theta(x), e_c) = \exp(-\frac{\frac{1}{n}||W_c f_\theta(x) - e_c||_2^2}{2\sigma_2},$$

where f_θ is the feature extractor mapping from input x of dimension m to the feature vectors of dimension d, and learnable parameters θ. W_c – for a class c – is a weight matrix of size n by d corresponding to the additional layer that transforms the output of the feature extractor to a new embedding space with centroids size. K_c – the kernel output – is computed for each centroid class e_c with σ being the hyperparameter called the length scale. The prediction of this model is represented as:

$$\underset{c}{\operatorname{argmax}} K_c(f_\theta(x), e_c).$$

Hence, the predictive uncertainty can be obtained by finding: $\max_c K_c(f_\theta(x), e_c)$. The optimisation function can be expressed as:

$$L(x, y) = -\sum_{c} y_c \log(K_c) + (1 - y_c) \log(1 - K_c).$$

Moreover, there is further regularisation using a two-sided gradient penalty (l_2 norm) where this penalty consists of regularisation factor λ to be tuned.

5.7.5.4 Deep Ensemble

Deep Ensemble is a collection of deep models with different initialisations. Training multiple models with distinct initialised weights produce multiple outputs on a given prediction like MC-dropout but with independent parameters. Hence, the predictive mean can be obtained as follows:

$$p_{ensemble}(y^* = c | x^*) = \frac{1}{T} \sum_{i=1}^{T} M_i(x^*),$$

where $M_i(x)$ is the prediction obtained by model M_i on a given input x.

5.7.6 Experiments and Results

In my experiments, I apply different machine learning models on graph and image datasets known as Cora and MNIST, respectively. Then, I estimate uncertainties besides the predictions to evaluate and compare the different uncertainty estimation methods. I use PyTorch [163] and Pytorch-geometric package [63] in Python programming language.

5.7.6.1 Experimenting with Graph Data

As an example of graph data, I use the Cora dataset to test the proposed uncertainty method. Cora is a graph-structured data that comprises academic publications as nodes, and citations as the edges [185]. This data is used in node classification tasks in which each node is classified into one of seven subjects. The node features reflect the absence/presence as 0/1 of the corresponding word in the dictionary, in which the unique words in the dictionary are the total number of features. The data is described in Table 5.8. In my experiments, I follow the same experimental setup for this data as in [232].

Algorithm 2 Generalised MC-AA

Require:

- $\mathcal{M}$: is a neural network that maps the feature vectors to multi-class predictions.
- J(.,.): is a loss function (e.g. binary cross entropy) that requires as input the predicted class derived from a given data point and the desired class label.
- **Input:**
  - **I:** is a discrete interval consists of a set of ε values such that I=$\{-\varepsilon_{max}, ..., 0, 0+\beta, ..., \varepsilon_{max}\}$, which is evenly spaced by β and centered around zero. ε_{max} is a small scalar to be tuned.
  - x: is an input feature vector.
  - c= argmax($\mathcal{M}(x)$) is the predicted class: $c \in \mathbb{N}$
- **Output:**
  - $\hat{y}$: is the set of outputs for a single data point after perturbations. $\hat{y} = \{\hat{y}_{\varepsilon_1}, ..., \hat{y}_{\varepsilon_n}\}$, where $\varepsilon_i \in I$.

```
Function
Compute the gradients of the loss function with respect to $x$ as:
**grad** = $\nabla_x J(x, c)$ (Difference from Algorithm 1)
**for all** $\varepsilon_i \in I$ **do**
  $x_{\varepsilon_i}$ = x + $\varepsilon_i$.sign(grad);
  $\hat{y}_{\varepsilon_i} = \mathcal{M}(x_{\varepsilon_i})$ ;
  return $\hat{y}_{\varepsilon_i}$
**end for**
$\hat{y} = \{\hat{y}_{\varepsilon_1}, ..., \hat{y}_{\varepsilon_n}\}$;
return $\hat{y}$
```

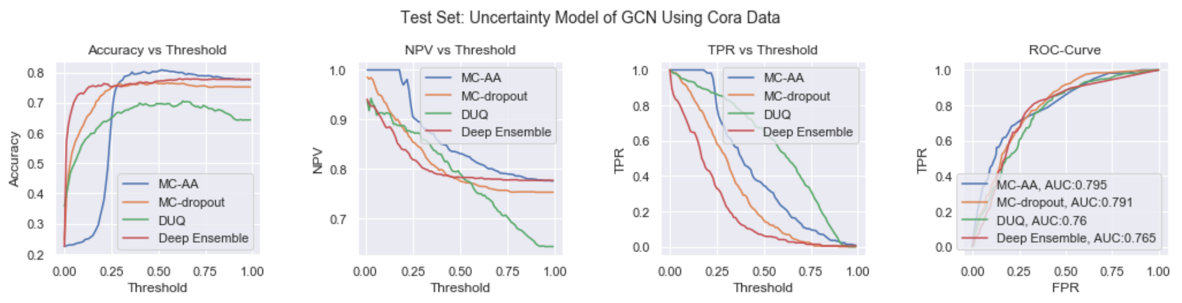


Figure 5.30: GCN model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

Cora dataset

# Nodes	2708
# Edges	5429
# Classes	7
# Features	1433
Train/Validation/Test	140/500/1000 nodes

Table 5.8: Graph network description of Cora data.

Since Cora is graph-structured data, I arbitrarily choose GCN, GraphConv, GAT, SAGEConv, LEConv, and TAGConv that are provided in List 5.5.1 to perform node classification.

The widths of all hidden layers are set to 16 neurons, a dropout after each hidden layer is set to 0.5 and the number of epochs is set to 100. I use a non-weighted NLLLoss and Adam optimiser to train the given models. Each of the preceded models consists of two graph convolutional layers. All hidden layers are squashed by ReLU and the output layers are followed by the softmax function except for DUQ which uses the RBF kernel as output. To capture model uncertainty, I use MC-AA, MC-dropout, DUQ and Deep Ensemble methods. The hyper-parameters for these methods are empirically tuned which are summarised in Table 5.10. After computing MI, I plot A_u, NPV, TPR and ROC to compare my proposed method with MC-dropout as depicted in Figures 5.30-5.35.

Model Uncertainty	Hyper-parameters (Cora)
MC-AA	ε_{max}=0.15, T=10, $\beta=\frac{2\varepsilon_{max}}{T}$
MC-dropout	T=10, dropout=0.5
DUQ	$\lambda = 1e^{-2}$, $\sigma = 0.3$, γ=0.99
Deep Ensemble	T=10

Table 5.9: Tuned hyper-parameters of uncertainty methods using Cora Data.

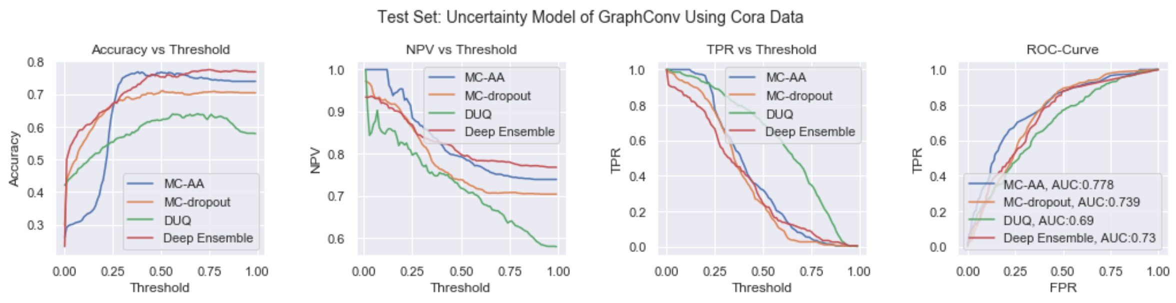


Figure 5.31: GraphConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

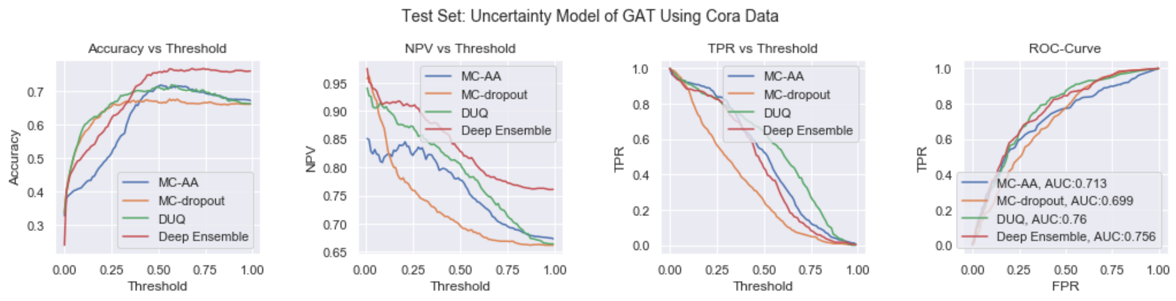


Figure 5.32: GAT model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

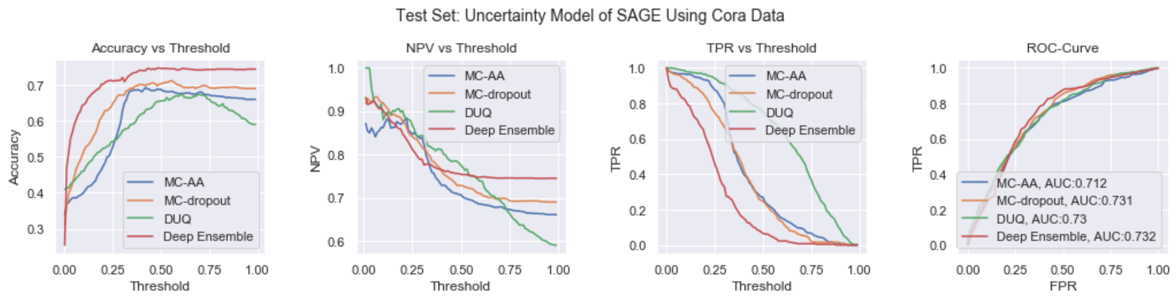


Figure 5.33: SAGEConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

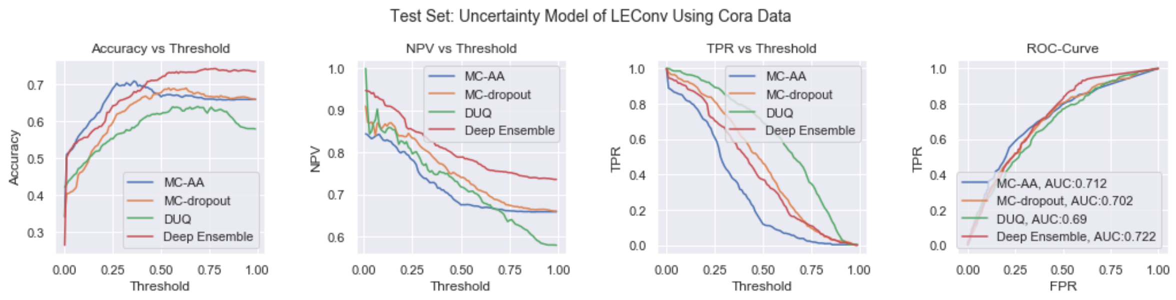


Figure 5.34: LEConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

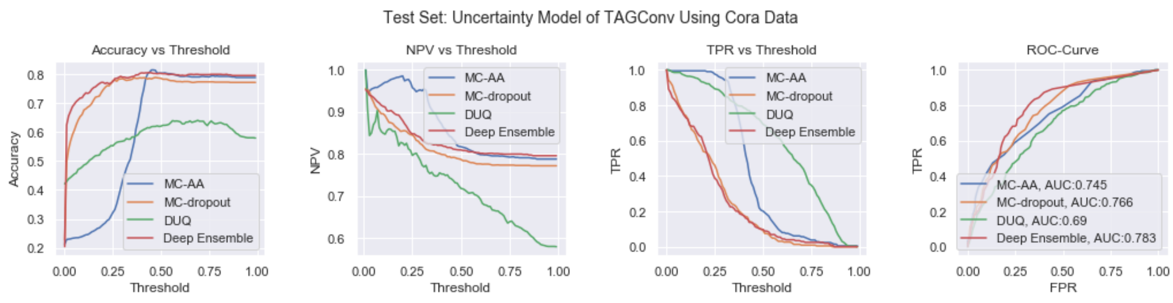


Figure 5.35: TAGConv model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

5.7.6.2 Experimenting with Image Data

Regarding MNIST dataset [116], the data is divided as 90k/10k for train/test split. I use the convolutional neural network (CNN) model that appeared in [210] as a deep feature extractor. The deep model consists of 3 CNN layers with output channels 64, 128 and 128, respectively, and a feed-forward layer with widths of 256. The kernel size is set to 3. Moreover, batch normalisation after every convolutional layer followed by 2 max-pooling of size 2 by 2 is applied. The padding in the first two convolutional layers is set to 1. A dropout of value 0.5 is empirically set after the feed-forward hidden layer. The learning rate is set to 0.001, chosen empirically. The output layer is followed by the softmax function to output the class prediction which is one of the handwritten digits from 0 to 9. The batch size is arbitrarily chosen to be 1024, then I perform 30 epochs to train the model. This model has attained an accuracy of over 98% to classify the digits. Likewise, I capture model uncertainty on the CNN model by performing MC-AA, MC-dropout, DUQ, and Deep Ensemble where the hyper-parameters are summarised in Table 4. As the input images 28x28 are grey-scale, the perturbed inputs by MC-AA should be clamped between -1 and 1 which is the range of the pixel values. I plot the preceded model uncertainty metrics as depicted in Figure 5.36.

Model Uncertainty	Hyper-parameters (MNIST)
MC-AA	ε_{max}=0.2, T=10, $\beta= \frac{2\varepsilon_{max}}{T}$
MC-dropout	T=10, dropout=0.5
DUQ	$\lambda = 1e^{-1}$, $\sigma = 0.4$, γ=0.9
Deep Ensemble	T=10

Table 5.10: Tuned hyper-parameters of uncertainty methods using MNIST Data.

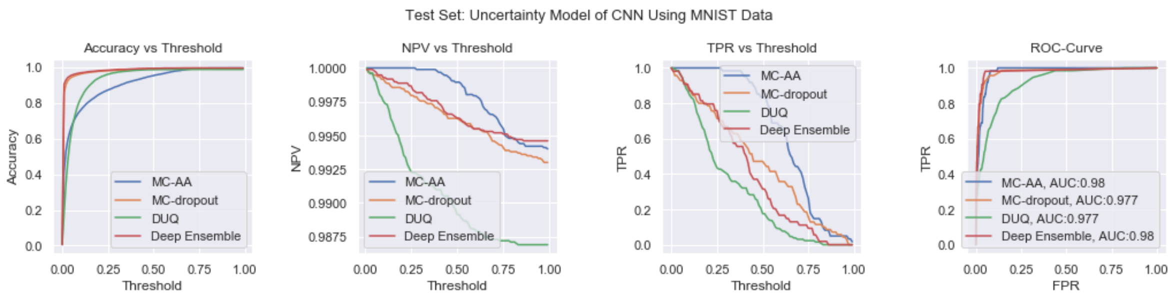


Figure 5.36: CNN model uncertainty. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

5.7.7 Discussion

5.7.7.1 Uncertainty performance with Cora Data

The performance of model uncertainty is computed by moving a threshold between 0 and 1 and computing the evaluation metrics provided earlier at each threshold. Generally, MC-AA for multiclass classification has noticeably revealed competence against other uncertainty estimation methods on various graph learning models with Cora data in terms of accuracy and ROC-AUC curve plots in Figures 5.30-5.35. Regarding other measurements, the NPV metric has shown the highest with MC-AA against other methods with GCN and TAGCN models. TPR metric has revealed acceptable outcomes with MC-AA in the different graph learning models except for LEConv. All graph learning models have admitted the outperformance of DUQ in the subplots corresponding to TPR metrics. The same models have revealed poor accuracy using the DUQ model wherein the overall model is considered with a deficient performance. Despite MC-AA having competent uncertainty estimates, this method has performed poorly with the LEConv model. The reason for the deficient performance is due to multiple class distributions that restrict the behaviour of MC-AA. In other words, the FGSM method in multiclass models might produce a perturbed input that cannot escape its relevant class. Thus, the perturbation using the adversarial attack idea is not exact towards the decision boundary as I use the sign of gradients which is the l_∞ norm and neglect the ratios corresponding to each dimensional feature. Therefore, the perturbation on the given input does not allow this data point to fall into another class. For this reason, I propose a way to avoid this drawback of MC-AA by converting multi-class classification problems into a *one-versus-all* binary classification which has appeared to be more effective with MC-AA. I choose LEConv which performed poorly with MC-AA. To convert to the *one-versus-all* classification, I choose the class with the highest false instances among all other classes to be a positive class, which is class 4, while the remaining labels are assigned as the negative class. However, the same concept can also be applied to the different permutations of *one-versus-all* binary classifications (e.g., first class versus others, second class vs others, etc.). The model LEConv is trained following the same experimental setup as preceded. Henceforth, I capture model uncertainty using MC-AA (for binary classification), MC-dropout, DUQ, and Deep Ensemble. The results are provided in Figure 5.37.

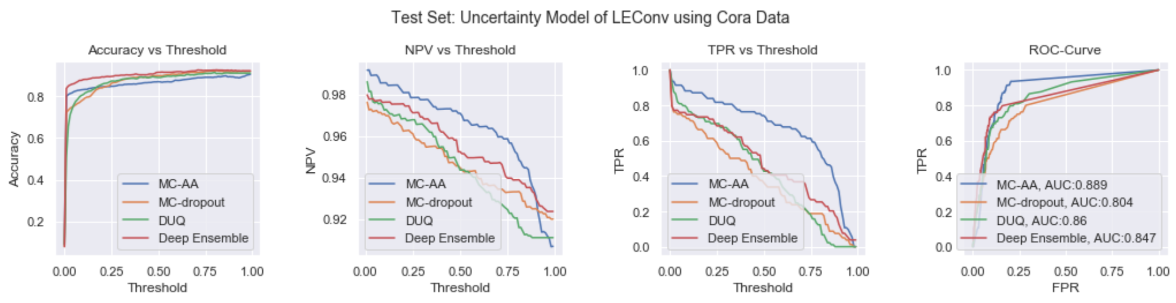


Figure 5.37: Model uncertainty of LEConv as binary classification (class 4 vs rest) using MC-AA and MC-dropout. The subplots (from left to right) correspond to A_u, NPV, TPR and ROC-curve as a function of threshold T_u.

After converting the classes of Cora data into binary labels, MC-AA has shown superior success against other uncertainty methods. Clearly, low FN (incorrect and certain) is provided. Here, the sign of the gradients in the FGSM method allows the data points to jump to the opposite class as there are only two competing classes.

5.7.7.2 Uncertainty performance with MNIST Data

Referring to Figure 5.36, the model uncertainty of CNN using MC-AA has outperformed other uncertainty methods using MNIST data. The overall model performance among all methods has attained the same accuracy. Whereas NPV and TPR metrics have depicted superior success with MC-AA wherein lower FN (incorrect and certain) has been obtained. To highlight the effectiveness of uncertainty estimates, I plot the normalised density distribution of the predictive uncertainties in Figure 5.38. I cluster the distributions according to the correct/incorrect predictions, referring to Table 1. Clearly, all methods commonly have reflected good uncertainty estimates for the correct predictions. However, the distribution of incorrect predictions among all uncertainty methods has shown different densities. The density of incorrect predictions with MC-AA is more concentrated towards the right values of mutual information where these predictions are considered uncertain. With other methods, the predictive uncertainty of incorrect predictions is distributed among the whole mutual information scale. Moreover, the uncertainty estimates with the DUQ model have depicted more mix between correct/incorrect densities which are reflected in Figure 5.38. In addition, I compute the mean/standard deviation to describe these distributions as provided in Table 5.11. The mean of incorrect predictions with MC-AA has attained the highest value with the smallest standard deviation. This is desired to obtain an effective model uncertainty that classifies incorrect predictions as uncertain. On the other hand, all methods have shown adequate results of low mean/low standard deviation for correct predictions. To pinpoint the effectiveness of MC-AA on the image MNIST data, I provide a case study where I opt for an image example, depicted in Figure 5.29, that is wrongly predicted by CNN among all methods. I investigate the predicted class of this digit two image as well as its uncertainty estimates by MC-AA, MC-dropout, DUQ and Deep Ensemble as provided in Figure 5.39. Closely, all methods have

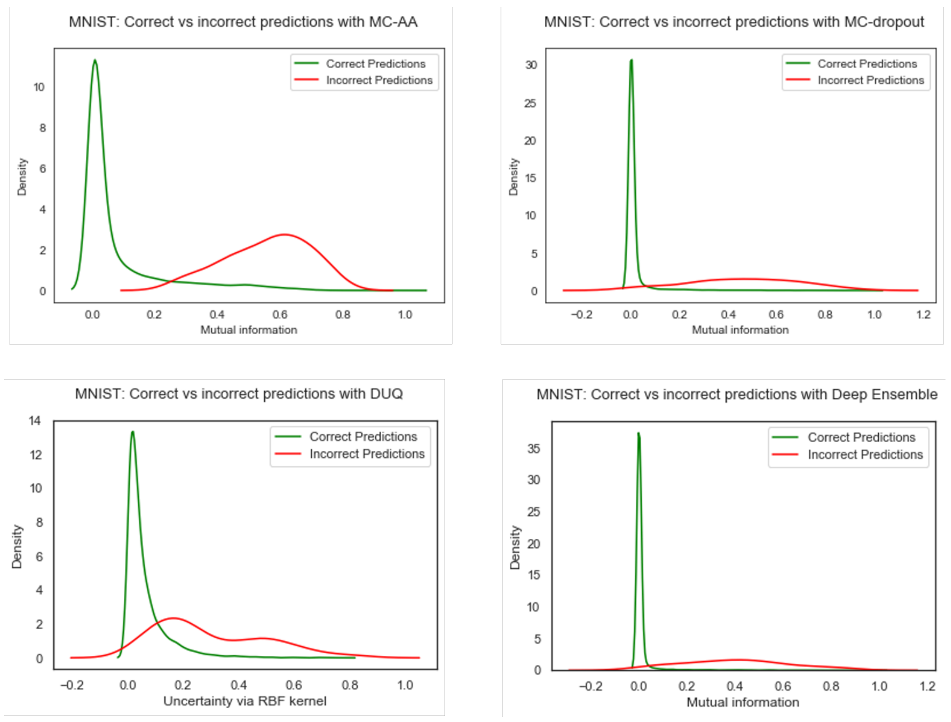


Figure 5.38: Distribution of predictive uncertainty measurements derived from MNIST test set using MC-AA, MC-dropout, DUQ and Deep Ensemble. The correct predictions curve is the uncertainty measurement of the data points that are predicted correctly. Similarly, an incorrect prediction curve is the uncertainty measurement of the data points that are predicted incorrectly.

erroneously predicted this digit as seven with high confidence, whereas class two has provided a low predicted probability.

5.7.8 Concluding Remarks

We have extended the study of MC-AA, an uncertainty estimation method based on the adversarial attack that works for multiclass classification. By benchmarking the MC-AA method against other uncertainty estimation methods, I have shown the effectiveness of MC-AA in capturing model uncertainty using deep models on graph data of Cora and image data of MNIST. Concisely, I have examined MC-AA with multiclassification tasks against MC-dropout, DUQ and Deep Ensemble methods. However, the presented method has revealed low performance in the LEConv model using Cora data. I tackle this limitation by converting the multiclass classification into a binary classification task where MC-AA outperforms other methods. I discuss the best uncertainty model performance of MC-AA which attained an AUC score of 0.889 with the LEConv model, after converting the Cora data to binary classification, where the corresponding NPV and TPR have also revealed superior success in comparison to other methods. Surprisingly, MC-AA has shown a significant outperformance using MNIST without the need to convert to binary classification. The recorded AUC score of this method

Uncertainty method	**Uncertainty Distribution**	**Mean**	**Standard deviation**
MC-AA	Correct prediction	0.099	0.177
	Incorrect prediction	0.765	0.144
MC-dropout	Correct prediction	0.016	0.069
	Incorrect prediction	0.456	0.220
DUQ	Correct prediction	0.073	0.085
	Incorrect prediction	0.272	0.1753
Deep Ensemble	Correct prediction	0.012	0.058
	Incorrect prediction	0.398	0.218

Table 5.11: Descriptive statistics summary of predictive uncertainty distribution using MNIST test set.

is 0.98 outperforming other methods in addition to their NPV and TPR curves. To wrap up, MC-AA is powerful in reducing the number of false negatives of model uncertainty (i.e., data points that are incorrect but certain). This is due to the perturbations that are performed on the input level, unlike previous uncertainty methods. The limitation of this study is that it is not known when MC-AA is a good approach in multiclassification tasks. However, the conversion to binary classification is always promising since MC-AA tends to perturb an input between decision boundaries. Consequently, having a single decision boundary leads to effective results. Whereas multiple classes mislead the perturbed input which fails to reflect a good uncertainty estimate using the FGSM sign method in MC-AA.

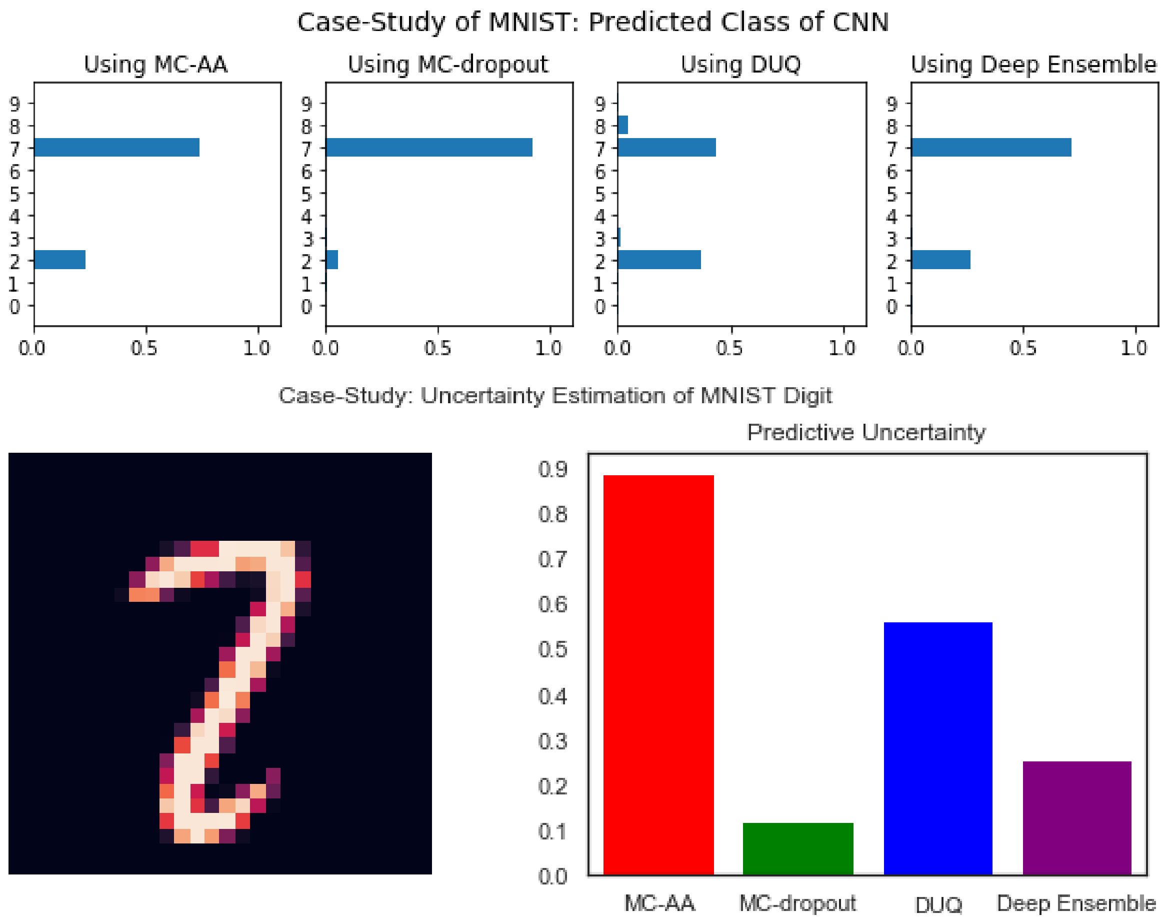


Figure 5.39: Case study on MNIST test set. The top subplot is the predictive probability by the various CNN models (CNN with MC-AA, MC-dropout, DUQ, Deep Ensemble) of a single MNIST digit shown in the bottom left subplot. The bottom right subplot belongs to its uncertainty estimate derived from different uncertainty estimation methods.

5.8 Summary

Uncertainty in machine learning is important to know what the model is uncertain about. In particular, the model uncertainty is highly desirable in the blockchain data which might be subjected to evolving events that the model is not aware of. First, I study MC-dropout – an uncertainty estimation method – behaviour on the 2D synthetic and Bitcoin datasets. Despite MC-dropout being an efficient and simple method to perform, I discuss the limitation and further discussions. In the subsequent study, I examine the most recent uncertainty estimation methods which all fail to capture data points in the overlapping region in binary classification tasks. Hence, I propose the MC-AA method based on the adversarial attack idea to estimate model uncertainty besides its predictions. MC-AA produces uncertainty using the adversaries of the inputs with respect to the output. In other words, I perturb a feature vector back and forth towards the decision boundary. The nearer the given input to the decision boundary, the more likely its associated output changes its probability predictions due to the perturbations on the input. In addition, MC-AA is also an efficient and simple method similar to MC-dropout. Using MLP models, the main finding is the MC-AA outperforms other given uncertainty estimation methods using data derived from Bitcoin and Ethereum blockchain. Motivated by graph learning models, I also test the performance of model uncertainty with MC-AA on GNN models in which I use Bitcoin transaction graphs as the input to the given models. In this study, MC-AA is compared to MC-dropout where the former method outperforms the latter one. The remaining parts of this chapter consider a comprehensive examination of the MC-AA method against MC-dropout to capture model uncertainty. Using a different dataset for binary node classification tasks, MC-AA show effective results compared to MC-dropout using the GitHub dataset. On the other hand, I extend the method of MC-AA to multi-class classification tasks using MNIST – the image dataset – and the graph-structured Cora dataset. In a multi-class classification problem, MC-AA shows different behaviour than that in the binary classification tasks. This is due to several decision boundaries that are formed per class distribution. Thus, I present a generalised MC-AA where I perturb the input vector back and forth towards its predicted class distribution regardless if it is correctly predicted or not. In this way, the back-and-forth perturbations move the input vector between its predicted class toward the other class distributions using the FGSM method. Despite the acceptable outcomes of the model performance by MC-AA, this method reveals a drawback in some cases with multi-classification tasks. Consequently, I tackle this drawback by converting the class into binary classification as *one-versus-all* classification task which reveals the outperformance of MC-AA over other recent uncertainty estimation methods.

To wrap up, MC-AA works perfectly with binary classification problems since the method is dealing with a single decision boundary between two opponent classes. However, there is no guarantee that this method will be the best choice when dealing with multi-class classification tasks due to the multiple decision boundaries.

CHAPTER 6

Proposed Frameworks: Machine Learning for Illicit Activities on Blockchain

Contents

To fulfil the project's aims, this chapter presents two frameworks to detect illicit transactions using a machine learning approach on blockchain by using the data derived from the Bitcoin blockchain. My studies involve the Bitcoin dataset which is described earlier in Chapter 3. My aim in this chapter is divided as follows:

- To propose an active learning solution to train and detect illicit transactions in Bitcoin:
  - I propose a model named temporal-GCN to classify the Elliptic dataset using its local features only. Temporal-GCN exploits the temporal and graph-structured information of the Elliptic dataset.
  - I perform active learning using a variety of acquisition functions as an example of the human-in-the-loop approach. This study imitates a real-world active learning solution.
  - I evaluate the methods and compare the experimental results.
- To propose a robust recurrent GCN approach based on sequential prediction to classify illicit transactions in the Elliptic dataset:
  - I propose a novel model, named RecGNN that exploits the temporal and graph topology of the Elliptic dataset.
  - I design new features, named antecedent neighbouring features, derived from the Bitcoin transaction graph of Elliptic data.
  - I discuss the efficiency of the evaluated model and perform a case study.

6.1 Temporal-GCN: Graph-based LSTM for Anti-Money Laundering

6.1.1 Abstract

For this framework, I develop a classification model that combines long-short-term memory (LSTM) with GCN – referred to as temporal-GCN – that classifies the illicit transactions of Elliptic data using its transaction features only. Subsequently, I present an active learning framework applied to the large-scale Bitcoin transaction graph dataset, unlike previous studies on this dataset. Uncertainties for active learning are obtained using each of the MC-dropout and MC-AA uncertainty estimation methods. Active learning frameworks with these methods are compared using various acquisition functions that appeared in the literature. My main finding is that the temporal-GCN model has attained significant success in comparison to the previous studies with the same experimental settings on the same dataset. Moreover, I evaluate the performance of the provided acquisition functions using MC-AA and MC-dropout and compare the result against the baseline random sampling model.

6.1.2 Motivation

The bitcoin blockchain is one of the main contributors to big data technology, wherein a massive amount of data is daily added to the ever-growing blockchain. Consequently, the labelling process required by supervised machine learning on this type of data is hard and very time-consuming. This issue is tackled by the active learning (AL) approach which queries the labels of the most informative data points that attain high performance with fewer labelled examples. Using the Bitcoin dataset, the work in [127] has considered an AL solution that has shown the capability of matching the performance of a fully supervised model by using 5% of the labelled data. However, the latter study has not considered the graph structure or the temporal information of the used dataset.

In this study, I aim:

- To present a model in a novel way that considers the graph structure and temporal sequence of Elliptic data to predict illicit transactions that belong to illegal services in the Bitcoin blockchain network.
- To perform active learning on Bitcoin blockchain data that mimics a real-life situation, since Bitcoin blockchain is a massively growing technology and its data is time-consuming to label.

The presented classification model comprises LSTM (long short-term memory) and GCN models, wherein the overall model attains an accuracy of 97.7% and f_1-score of 80% which outperform previous studies with the same experimental settings. On the other hand, the presented active learning framework requires an acquisition function that relies on the model's uncertainty to query the most informative data. Here, the model's uncertainty estimates are obtained using two comparable methods MC-dropout and MC-AA that appeared in Chapter 5. I examine these two uncertainty methods using a variety of acquisition functions.

For each acquisition function, I evaluate the performance of the active learning framework. I compare the results of the different frameworks against the random sampling acquisition as a baseline model.

In what follows, I justify the motivation for performing this experiment where the recurrent neural networks and the active learning approach are used on the Elliptic dataset. I state the uncertainty methods used to model uncertainty. Subsequently, I present the various acquisition functions used to perform active learning. Then I present the temporal-GCN model used to perform classification. Afterwards, I provide the results and discussions which include the performance of the classification model, the evaluation of active learning frameworks, and a hypothesis test to study the difference between each of the active learning frameworks with respect to the random sampling. I also provide an ablation study about the classification model followed by the concluding remarks to wrap up this study.

6.1.3 Motivation

Active learning, a subfield of machine learning, is a way to make the learning algorithm choose the data to be trained on [187]. Active learning mitigates the bottleneck of the manual labelling process, such that the learning model queries the labels of the most informative data. Since it is so expensive to obtain labels, active learning has witnessed a resurgence with the appearance of big data where large-scale datasets exist [177]. Lorenz et al. [127] have presented an active learning framework in an attempt to reduce the labelling process of the large-scale Elliptic data of Bitcoin. The presented active learning solution has shown its capability in matching the performance of a fully supervised model with only 5% of the labels. The authors have focused on querying strategies based on uncertainty sampling [120, 187] and expected model change [188, 187]. For instance, the used uncertainty sampling strategy is based on the predicted probabilities provided by the random forest in [127]. Yet, there is no study that presents an active learning framework that utilises the recent advances in Bayesian methods on Bitcoin data. On the other hand, Gal et al. [71] have presented active learning frameworks on image data where the authors have combined the recent advances in Bayesian methods into the active learning framework. This study has performed MC-dropout to produce the model's uncertainty which is utilised by a given acquisition function to choose the most informative queries for labelling. Concisely, the authors in [70] have applied the entropy [189], mutual information [92], variation ratios [97], mean standard deviation (Mean STD) [101, 99] acquisition functions which are compared against the random acquisition. In this study, I conduct experiments using a classification model that exploits the graph structure and the temporal sequence of Elliptic data derived from the Bitcoin blockchain. Motivated by the studies in [127, 71], I perform the active learning frameworks, using pool based-based scenario [187] in which the classifier iteratively samples the most informative instances for labelling from an initially unlabelled pool. For each iteration, the classifier samples a batch of unlabelled data points according to their uncertainty estimates from Bayesian models using the sampling acquisition function.

6.1.4 Acquisition Functions for Active Learning

Pool-based active learning is a prominent scenario [187, 121] that assumes a set of labelled data available for initial training $\mathcal{D}_{train}$ and a set of unlabelled pool $\mathcal{D}_{pool}$ in a Bayesian model $\mathcal{M}$ with model parameters $w \sim p(w|\mathcal{D}_{train})$ and output predictions $p(y|w, \mathcal{D}_{train})$ for $y \in \{0, 1\}$ in binary classification tasks. Then, the Bayesian model $\mathcal{M}$ that is already trained on $\mathcal{D}_{train}$ queries the labels – from the unlabelled set $\mathcal{D}_{pool}$ – of an informative batch with size b by an oracle in order to obtain an acceptable performance with less training data.

Consider an acquisition function $a(x, \mathcal{M})$ that measures the score of batch of unlabelled data $\{x_i\}_{i=1}^{b} \in \mathcal{D}_{pool}$. Let $\{x^*\}_{i=1}^{b}$ be the informative batch acquired by the learning algorithm, then I can express it as follows [108]:

$$\{x^*\}_{i=1}^{b} = \underset{\{x_i\}_{i=1}^{b} \subseteq \mathcal{D}_{pool}}{\operatorname{argmax}} \quad a(\{x_i\}_{i=1}^{b}, p(w|\mathcal{D}_{train})). \tag{6.1}$$

In what follows, I demonstrate various acquisition functions that appeared in [68].

6.1.4.1 BALD: Bayesian Active Learning by Disagreement

BALD (Bayesian Active Learning by Disagreement) [92] is an acquisition method that utilises the uncertainty estimates via mutual information between the model predictions and model parameters. Hence, the learning algorithm queries the data points with the highest MI measurements. The highest MI measurements are produced when the predictions by Monte-Carlo samples are assigned with the highest probabilities where the samples are associated with different classes.

Here, I desire to acquire a batch of size b at each sampling iteration. Using BALD, this can be expressed as:

$$a_{BALD}(\{x_i\}_{i=1}^{b}, p(w|\mathcal{D}_{train})) \approx \sum_{i=1}^{b} \hat{I}(y_i; w|x_i, \mathcal{D}_{train}),$$

where $\hat{I}$ is derived from Equation 5.6 for MC-dropout and Equation 5.15 for MC-AA. Furthermore, the optimal batch is the one with the b-highest scoring data points to reduce the bottleneck of acquiring a single data point at each acquisition step.

6.1.4.2 Entropy

This acquisition method computes the entropy using the uncertainty estimates from Equations 5.7 and 5.16. During the active learning process, I choose the batch size with the maximum predictive entropy [189] which can be written as:

$$a_{Entropy}(\{x_i\}_{i=1}^{b}, p(w|\mathcal{D}_{train})) \approx \sum_{i=1}^{b} \hat{H}(y_i; w|x_i, \mathcal{D}_{train})$$

The maximum entropy explains the lack of confidence within the obtained predictions which are typically near 0.5.

6.1.4.3 Variation Ratios

Similarly, I choose the batch with the maximum variation ratios [97] where the variation ratio is expressed as:

$$\text{variation-ratio[x]} = 1 - \max_{y} p(y|x, \mathcal{D}_{train})$$

The maximum variation ratios correspond to the lack of confidence in the samples' predictions.

6.1.4.4 Mean Standard Deviation

Likewise, I sample a batch that maximises the mean standard deviation (Mean STD) [101, 99]. The predictive standard deviation can be computed as:

$$\sigma_c = \sqrt{E[p(y=c|x,w)^2] - E[p(y=c|x,w)]^2},$$

where E corresponds to the expected mean. The σ_c measurement computes the standard deviation between the predictions obtained by Monte-Carlo samples on every data point. Consequently, the mean standard deviation is average over all c classes which can be derived from:

$$\sigma = \frac{1}{C}\sum_{c}\sigma_c$$

6.1.4.5 Random Sampling: Baseline model

This acquisition function uniformly draws data points from the unlabelled pool at random.

6.1.5 Methods

In what follows, I refer to the presented model by temporal-GCN. In this study, the proposed model is based on a combination of LSTM and GCN models.

6.1.5.1 Long Short-Term Memory (LSTM)

Initially, LSTM is proposed by [90] as a special category of recurrent neural networks (RNNs) in order to prevent the vanishing gradient problem. LSTM has proven its efficacy in many general-purpose sequence modelling applications [78, 197, 198].

Consider a graph network of Bitcoin transactions at timestep t as $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ and its relevant adjacency matrix $\mathcal{A}_t \in \mathbb{R}^{n\times n}$, degree matrix $\mathcal{D} \in \mathbb{R}^{n\times n}$, where $\mathcal{V}_t$ and $\mathcal{E}_t$ are the sets of nodes as transactions and edges as payments flow, respectively, with $|\mathcal{V}_t| = n_t$ being the number of transactions at t^{th} timestep. Let $X_t \in \mathbb{R}^{n\times d_x}$ be the nodes features matrix with d_x-dimensional features and layer output $H_t \in [-1,1]^{n\times d_h}$ as and states $C_t \in \mathbb{R}^{n\times d_h}$ with d_h-dimensional embedding features. Referring to [78], the original LSTM model can be expressed as:

$$\begin{aligned}
I_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + w_{ci} \odot C_{t-1} + b_i),\\
F_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + w_{cf} \odot C_{t-1} + b_f),\\
\tilde{C}_t &= \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c)\\
C_t &= F_t \odot C_{t-1} + I_t \odot \tilde{C}_t,\\
O_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + w_{co} \odot C_t + b_o),\\
H_t &= O_t \odot \tanh(C_t),
\end{aligned} \tag{6.2}$$

where $\odot$ being the Hadamard product, $\sigma(.)$ the sigmoid function and $\tanh(.)$ the hyperbolic tangent function. The rest of the notations refer to LSTM layer parameters as follows: $I_t, F_t, O_t \in [0,1]^{n \times d_h}$ are the input, forget, and output gates, respectively. The weights $W_{x.} \in \mathbb{R}^{d_h \times d_x}, W_{h.} \in \mathbb{R}^{d_h \times d_h}, w_{c.} \in \mathbb{R}^{d_h}$ and biases b_i, b_f, b_c, b_o indicate the parameters of LSTM.

6.1.5.2 Topology Adaptive Graph Convolutional Network: TAGCN

In this study, I use a specific type of GCN called TAGCN as introduced in [57]. Generally, GCNs are neural networks that are fed with graph-structured data, wherein the node features with a learnable kernel undergo convolutional computation to induce new node embeddings. The kernel can be viewed as a filter of the graph signal (node), wherein the work in [52] suggested the localisation of the kernel parameters using Chebyshev polynomials to approximate the graph spectra. Also, the study in [106] has introduced an efficient algorithm for node classification using first-order localised kernel approximations of the graph convolutions.

On the other hand, the work in [57] has introduced TAGCN which is based on GCN but with fixed-size learnable filters and adaptive to the topology of the graph to perform convolutions in the vertex domain.

More formally, TAGCN can be expressed as follows:

$$H^{(l+1)} = \sum_{k=0}^{K} (D^{-1/2} A D^{-1/2})^k H^{(l)} \Theta_k, \tag{6.3}$$

where Θ_k is a learnable weight matrix at k-hop from the node of interest, $H^{(l)}$ and $H^{(l+1)}$ are the input and output node embeddings matrices, respectively.

6.1.5.3 Overall Model: Temporal-GCN

Since TAGCN in [57] requires no approximations in comparison to GCN by [106], I exploit the performance of TAGCN in Bitcoin data. Motivated by the work in [186] that has suggested feeding LSTM inputs with GCN node embeddings, temporal-GCN seeks to perform LSTM that learns the temporal sequence in which after is forwarded non-linearly to 2-TAGCN layers to exploit the graph structure of Bitcoin transaction graph. The temporal-GCN model can be expressed as:

$$\begin{aligned} \mathcal{H}^{(1)} &= \text{ReLU}(\text{LSTM}(\mathcal{X})), \\ \mathcal{H}^{(2)} &= \text{ReLU}(\text{TAGCN}(\mathcal{H}^{(1)}, \mathcal{E})), \\ \mathcal{H}^{(3)} &= \text{Softmax}(\text{TAGCN}(\mathcal{H}^{(2)}, \mathcal{E})), \end{aligned} \tag{6.4}$$

where $\mathcal{X}$ is the node feature matrix. LSTM(.) and TAGCN(.,.) are layers mapping a given input to an output from Equations 6.2 and 6.3, respectively. Softmax function is defined as

Model	% Accuracy	% Precision	% Recall	% F_1 Score
Temporal-GCN	**97.7**	**92.7**	**71.3**	**80.6**
GCN + MLP [13]	97.4	89.9	67.8	77.3
Evolve-GCN[221]	96.8	85.0	62.4	72.0
Skip-GCN[221]	96.6	81.2	62.3	70.5
Random Forest[221]	96.6	80.3	61.1	69.4
GCN[221]	96.1	81.2	51.2	62.8
MLP [221]	95.8	63.7	66.2	64.9
Logistic Regression[221]	92.0	34.8	66.8	45.7

Table 6.1: Classification results of Elliptic data using local features. This table shows a comparison between the presented model "Temporal-GCN" and previous studies using the same features and train/test split.

Softmax(x) $= \frac{1}{\mathcal{Z}} \exp(x_i)$, where $\mathcal{Z} = \sum_i \exp(x_i)$.

6.1.6 Experiments

6.1.6.1 Experimental Setup

Using the Elliptic data, I split the data following the temporal split as in [221], so that the first 35 graphs (i.e., $t = 1 \rightarrow t = 35$) account for the train set and the remaining are kept for testing. Since this dataset comprises partially labelled nodes, I only use the labelled nodes which add up to 29894/16670 transactions in the train/test sets, respectively. To train temporal-GCN, I use Pytorch Geometric (PyG) package [64] which is built on top of Pytorch (version 1.11.0) enabled-CUDA (version 11.3) in Python programming language. At each time-step t, I feed the relevant graph network with its node feature matrix (i.e., local features excluding timestep) to the temporal-GCN model that is summarised in Equation 6.4. LSTM layer uses only the node features without any graph-structural information to provide the output matrix $\mathcal{H}^{(1)}$. This matrix is then forwarded to 2-TAGCN layers (in $\mathcal{H}^{(2)}$ and $\mathcal{H}^{(3)}$) that consider the graph-structured data of the top-K influential nodes in the graph, where K is kept by default equal to 3. Hence, a $Softmax$ function provides the final class predictions as licit/illicit transactions. I choose $NLLLoss$ function and Adam optimiser in order to compute the loss and update the model's parameters. Using the same hyper-parameters from [13], the widths of the hidden layers are set to 50, the number of epochs is set to 50 and the learning rate is fixed at 0.001. Furthermore, I empirically opt for 0.7 for the dropout ratio to avoid overfitting. The classification results of the temporal-GCN model are provided in Table 6.1.

6.1.6.2 Active Learning

Active learning has a great impact to alleviate the bottleneck of labelling especially with this type of data. The main goal of active learning is to use less-training data with achieving acceptable performance. Here, I initially assume the train set as a pool of unlabelled data $\mathcal{D}_{pool}$ and I consider $\mathcal{D}_{train}$ as an empty set to be appended after the querying process. First, the process starts by randomly querying the first batch size for manual labelling, which is arbitrarily assigned to 2000 instances. Afterwards, I append the selected queries to $\mathcal{D}_{train}$ from $\mathcal{D}_{pool}$ to train the temporal-GCN model that is evaluated using the test set at each iteration. Subsequently, the same process is repeated using the uncertainty sampling strategy until I reach an adequate accuracy. However, I query for all instances in $\mathcal{D}_{pool}$. The uncertainty sampling is performed by using one of the acquisition functions demonstrated earlier. These acquisition functions require as input the uncertainty estimates derived by the uncertainty estimation methods. To imitate manual labelling, I append the labels to the queried instances. This experiment is performed using MC-dropout and MC-AA. I compare the performance of the active learning frameworks that use various acquisition functions on the two distinct uncertainty estimation methods. Regarding the hyper-parameters for producing uncertainty estimates, I arbitrarily set T=50 for multiple stochastic forward passes on the unlabelled pool for MC-dropout. With MC-AA, I arbitrarily choose $\varepsilon_T = 0.1$ and T=10. In addition, I perform random sampling as a baseline which uniformly queries data points at random from the pool. The process of performing active learning with the temporal-GCN model is schematised in Figure 6.1. The required time to perform the active learning process in an end-to-end fashion using parallel processing, referring to Figure 6.1, is provided in Table 6.2 using various acquisition functions under the given uncertainty methods.

6.1.7 Results and Discussions

We discuss the results of the temporal-GCN model in light of the previous studies using the same dataset. Subsequently, I provide and discuss the results provided by various active learning frameworks. Then I apply a non-parametric statistical method to discuss the significant difference between MC-AA and MC-dropout in performing active learning in comparison to the random sampling strategy.

6.1.7.1 Performance of temporal-GCN

Temporal-GCN has outperformed all previous studies on this dataset that use local features under the same train/test split settings. The presented model has leveraged the temporal sequence and the graph structure of the Bitcoin transaction graph, wherein the classification model is capable of detecting illicit transactions with accuracy and f_1-score equal to 97.77% and 80.60%, respectively. In previous studies, Evolve-GCN has attained an accuracy of 96.8%. The latter model has exploited the dynamicity of the graph by performing LSTM on the weights of the GCN layer, which outperformed GCN and skip-GCN without any tem-

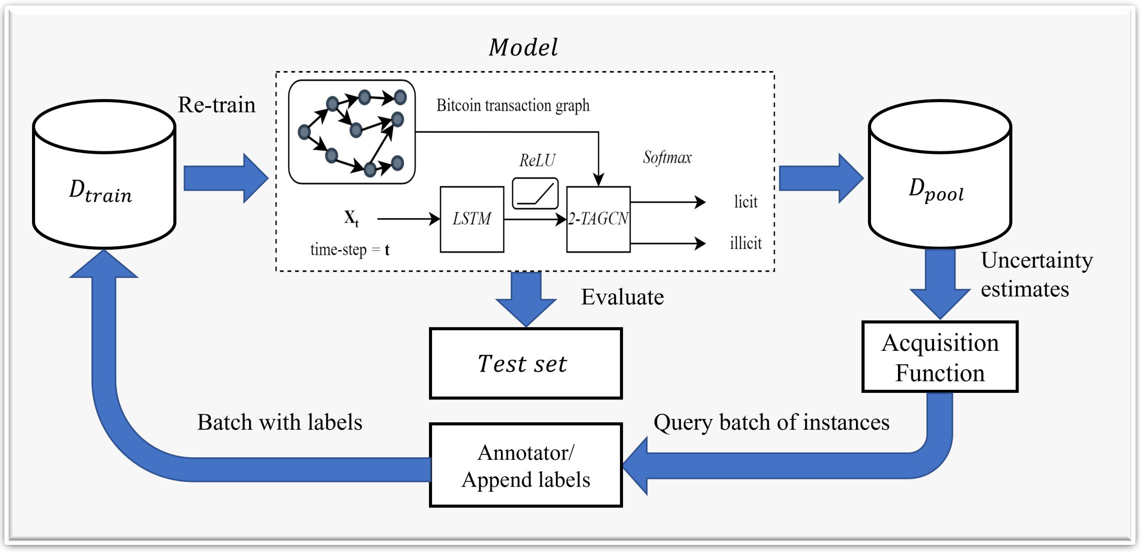


Figure 6.1: Schematic representation of the active learning framework using the proposed temporal-GCN model. This framework is a pool-based scenario where the annotator queries the data points labels from a pool of unlabelled instances, $\mathcal{D}_{pool}$ using an acquisition function. For each iteration, the queried batch is appended to the train set $\mathcal{D}_{train}$.

poral information. Whereas in temporal-GCN, LSTM has exploited the temporal sequence of Elliptic data itself before using any graph information in which the new transformed features are mapped non-linearly into the graph-based approach to perform graph convolutions in the vertex domain. Thus, the new hidden features of the TAGCN model are enriched with relevant temporal information. Similarly to [221], I also realise that the presented model performs poorly with the black market shutdown at time-step 43 as shown in Figure 6.2. Regarding the time-complexity, the complexity of LSTM is about $\mathcal{O}(4nd_h(d_x + 3 + d_h)$, whereas 2-TAGCN layers have a linear complexity which is $\mathcal{O}(n)$. Consequently, the time-complexity of the temporal-GCN model becomes $\mathcal{O}(n(4d_hd_x + 3d_h + d_h^2 + 1))$ at every epoch.

6.1.7.2 Evaluation of Active Learning Frameworks

Referring to Figure 6.3, I plot the results of various active learning frameworks using various acquisition functions (BALD, Entropy, Mean STD, Variation Ratio) which in turn utilise MC-dropout and MC-AA uncertainty estimation methods. Moreover, I plot the performance of the baseline model using a random sampling strategy. In the first subplot, BALD has revealed a significant success under MC-AA and MC-dropout uncertainty estimates which active learning is effectively better than the random sampling model. With the remaining acquisition functions, MC-dropout has remarkably achieved a significant outperformance over MC-dropout and the random sampling model. MC-AA which is utilised in entropy and variation ratio acquisition function has not performed better than random sampling. Furthermore, the active learning framework using the BALD acquisition function in Figure 6.3 is capable of matching the performance of a fully supervised model after using 20% of the queried data.

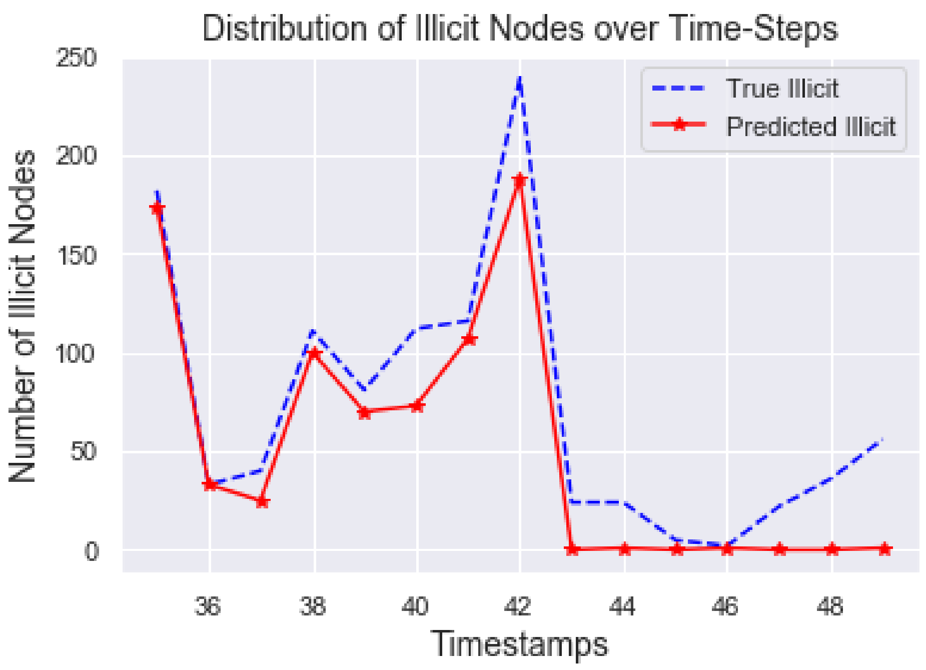


Figure 6.2: Illicit transactions distribution in the test set over the time-steps. The blue curve represents the actual illicit labels, whereas the red curve represents the illicit predictions that are actually illicit labels by temporal-GCN.

This amount of queried data belongs to the second iteration. In my experiments, MC-AA has been revealed to be a viable method as an uncertainty sampling strategy in an active learning approach with BALD and Mean STD acquisition functions. This is reasonable since the latter two methods estimate the uncertainty based on the severe fluctuations of the model's predictions on a given input wherein MC-AA suits this type of uncertainty.

Referring to Table 6.2, BALD acquisition has recorded the shortest time among other acquisition functions using MC-AA, where this framework has been processed in 28.07 minutes using parallel processing. Whereas the longest time by MC-AA is recorded by the entropy and variation ratio. For MC-dropout, the shortest time is recorded by Mean STD acquisition which is 27.09 minutes. Whereas, the framework using variation ratio has revealed the longest time which is 28.3 minutes. I also note that the frameworks using MC-AA require more time than the ones using the MC-dropout method. This is due to the adversaries computed by MC-AA which require more time to be processed.

6.1.7.3 Wilcoxon Hypothesis Test

To show the statistical significance of the various acquisition functions that appeared in Figure 6.3, I perform the non-parametric Wilcoxon signed-rank test [223]. It is used to test the null hypothesis between two paired samples based on the difference between their scores. Given two paired samples $\mathcal{P} = \{p_1, ..., p_m\}$ and $\mathcal{Q} = \{q_1, ..., q_m\}$, then the absolute value of the difference between the samples can be expressed as:

$$\mathcal{D} = |\mathcal{P} - \mathcal{Q}| = \{|p_1 - q_1|, ..., |p_m - q_m|\},$$

where m is the number of samples of each set. In summary, this test accounts for the statistical difference between the sets $\mathcal{P}$ and $\mathcal{Q}$. The Wilcoxon test compares a test statistic $\mathcal{T}$ to *Student's t-distribution*. To perform this test, I use the Wilcoxon function in sklearn [165]. Referring to Figure 6.3, I apply the Wilcoxon test for each subplot twice. The first one studies the difference between MC-AA and random sampling curves. Likewise, the second one accounts for the differences between MC-dropout and random sampling curves. For instance, I will refer to the Wilcoxon test applied between MC-AA and random sampling pairs as follows:

$$\text{p-value} = \text{Wilcoxon(MC-AA, Random Sampling)},$$

where *p-value* is a statistical measurement which is used to validate how likely the difference of the paired samples is given by the null hypothesis. Let H_0 be the null hypothesis and H_1

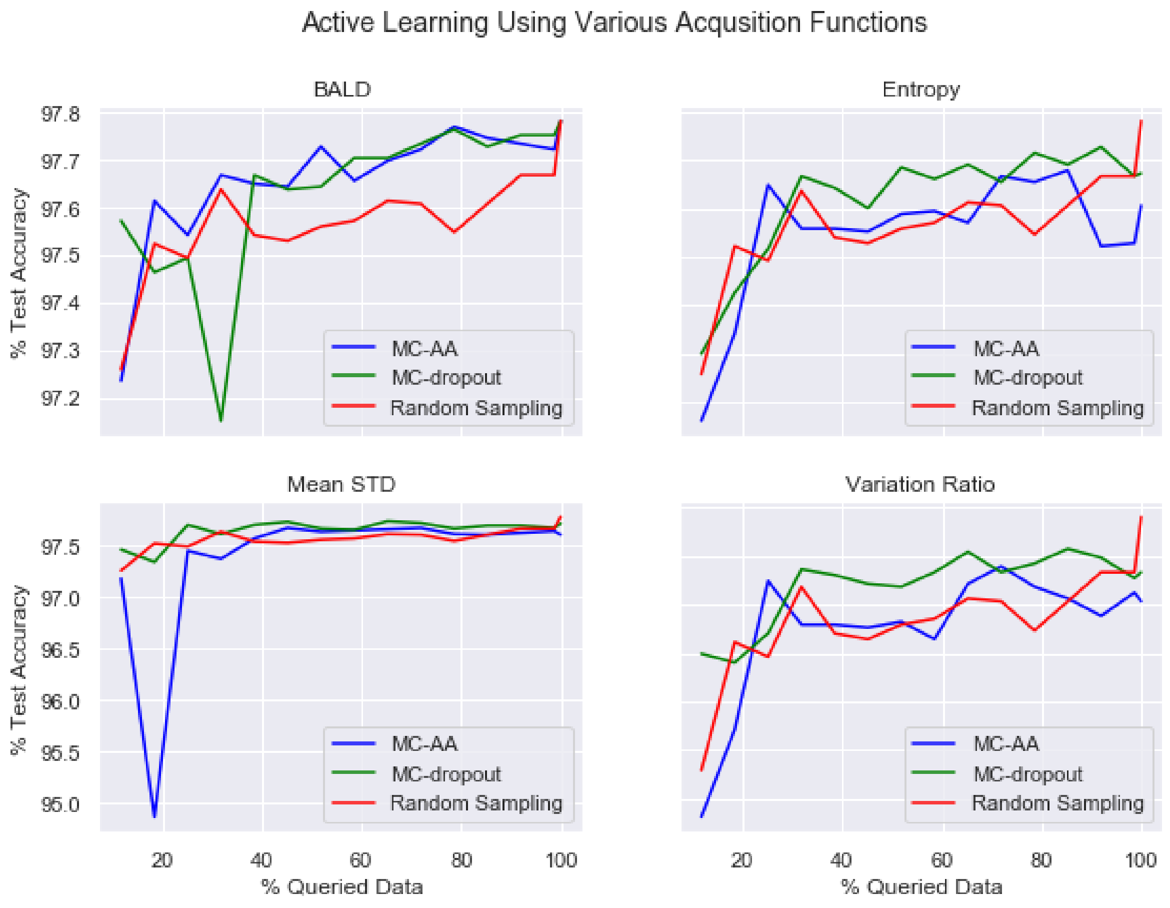


Figure 6.3: Results of active learning using BALD via MC-dropout in comparison to BALD via MC-AA.

Acquisition Function	**Runtime (mins) Using MC-AA**	**Runtime (mins) Using MC-dropout**
BALD	t_{MC-AA}=28.07	$t_{MC-dropout}$= 27.1
Entropy	t_{MC-AA}= 28.9	$t_{MC-dropout}$=27.3
Variation Ratio	t_{MC-AA}=28.9	$t_{MC-dropout}$=28.3
Mean STD	t_{MC-AA}=28.68	$t_{MC-dropout}$=27.09

Table 6.2: Time required to perform the active learning process in an end-to-end fashion using the proposed temporal-GCN model. The time is provided for each experiment that uses the corresponding acquisition function which relies on a given uncertainty estimation method.

be the alternative one. Then they can be written as:

$$H_0 = \text{No difference between the two paired samples}$$
$$H_1 = \text{There is a difference between the two paired samples}$$

Acquisition Function	Wilcoxon (MC-AA, Random Sampling)	Wilcoxon (MC-dropout, Random Sampling)
BALD	0.0009	0.0217
Entropy	0.2552	0.309
Variation Ratio	0.266	0.0071
Mean STD	0.5507	0.0112

Table 6.3: Wilcoxon statistical test between the accuracy derived by MC-AA/MC-dropout with respect to the random sampling model using various acquisition functions. The values correspond to the p-values by the test statistic.

We opt for 0.05 for the significance level α where I test against the null hypothesis. The smaller the p-value by the Wilcoxon function output, the more evidence I have to reject the null hypothesis. Precisely, the test statistic is statistically significant if the p-value is less than the significance level. Subsequently, I provide the statistical results (p-values) of the various acquisition functions against the baseline random sampling. The results are provided in Table 6.3. The p-values – which are lower than the significance level α – from the BALD acquisition function are statistically significant against the null hypothesis in which there is statistical evidence about the difference between each of MC-AA and MC-dropout in comparison to random sampling acquisition. Moreover, MC-dropout against random sampling has shown a statistical significance against the null hypothesis using the variation ratio and Mean STD acquisitions where the p-values are 0.071 and 0.0112, respectively. There is no evidence against the null hypothesis for the entropy where the p-values are 0.2552 and 0.309 are greater than α.

6.1.8 Ablation Study

In this section, I present the ablation studies performed on the proposed temporal-GCN model. Referring to Table 6.4, I have studied the importance of using LSTM and TAGCN layers. The *Model-0* corresponds to the model performed in the experiments. Subsequently, I have applied a combination of replacing each of the given layers with a linear layer (*Model-1, Model2, Model-3*). I have also studied the case in which I remove one of the first two layers from the original model (*Model-6, Model-7*). Furthermore, I have shown the performance of the models that use either LSTM (*Model-4*) with linear layers. In *Model-2*, the replacement of the second layer by a linear layer has attained the highest accuracy in comparison to *Model-0*. The removal of LSTM in all cases has provided a drop in the model's performance, especially in *Model-7* which reveals the lowest accuracy. On the other hand, using LSTM without the graph learning algorithms in *Model-2* has recorded the second-lowest accuracy in this study. I have also tweaked the K hyper-parameter that appeared in TAGCN referring to Equation 6.3. The original model uses, by default, K=3 which means that neighbourhood information is aggregated up to 3-hops. Then I checked the performance for $K \in \{1, 2\}$ as provided in Table 6.5. The highest accuracy has been recorded for using $K = 3$ and the second one for $K = 1$. Surprisingly, the drop in accuracy is not consistent between the different K values. This might be due to the informative features derived from neighbouring nodes up to 1-hop and 3-hops.

Model Number	**Layer-1**	**Layer-2**	**Layer-3**	**% Accuracy**
Model-0 (Ours)	LSTM	TAGConv	TAGConv	97.77
Model-1	Linear	TAGConv	TAGConv	97.66
Model-2	LSTM	Linear	TAGConv	97.76
Model-3	LSTM	TAGConv	Linear	97.57
Model-4	LSTM	Linear	Linear	97.44
Model-5	Linear	Linear	Linear	97.51
Model-6	LSTM	TAGConv	None	97.63
Model-7	TAGConv	TAGConv	None	96.65

Table 6.4: Ablation study over the layers of the proposed model. Each model number is provided by its architecture by changing the layers into linear layers or removing one of the layers. The term 'None' in the cells corresponds to the model having one of its layers removed.

Temporal-GCN Model

TAGCN K-hops	% Accuracy
K=3	**97.77**
K=2	**97.71**
K=1	**97.75**

Table 6.5: Ablation study by changing the number of K-hops in TAGCN layers of the temporal-GCN model as given in Equation 6.4.

6.1.9 Concluding Remarks

For anti-money laundering in Bitcoin, I have presented temporal-GCN, as a combination of LSTM and GCN models, to detect illicit transactions in the Bitcoin transaction graph known as Elliptic data. Also, I have provided active learning using two promising methods to compute uncertainties called MC-dropout and MC-AA. For the active learning frameworks, I have studied various acquisition functions to query the labels from the pool of unlabelled data points. The main finding is that the proposed model has revealed a significant outperformance in comparison to the previous studies with an accuracy of 97.77% under the same experimental settings. LSTM takes into consideration the temporal sequence of Bitcoin transaction graphs, whereas TAGCN considers the graph-structured data of the top-K influential nodes in the graph. Regarding active learning, the proposed framework is capable of achieving an acceptable performance by only considering 20% of the labelled data with the BALD acquisition function.

6.2 RecGNN: Robust Recurrent GCN based Sequential Prediction

6.2.1 Abstract

Money laundering has urged the need for machine learning algorithms for combating illicit services in the blockchain of cryptocurrencies due to its increasing complexity. The studies on the Bitcoin data of Elliptic have revealed promising results using supervised learning methods in classifying illicit Bitcoin transactions. Nonetheless, all learning algorithms have failed to capture the dark market shutdown event that occurred in this data using its original features. In this study, I propose a recurrent graph neural network model (RecGNN) in a novel way that extracts the temporal and graph topology of Bitcoin data to perform node classification as licit/illicit transactions. The proposed model mimics a sequential prediction model that utilises newly designed features, based on the previously labelled transactions up to 1-hop, named antecedent neighbouring features. Following the same configuration of the previous studies on Elliptic data, I show that my proposed model achieves the state-of-the-art with accuracy and f_1-score of 98.99% and 91.75%, respectively. Using the new set of features, I benchmark the performance of various supervised learning models against the RecGNN model. For the first time, RecGNN outperforms random forests which explain the power of the sequential prediction approach. For explainability purposes, I visualise a snapshot of a Bitcoin transaction graph of Elliptic data and I perform a case study using a backward reasoning process.

6.2.2 Introduction

The Bitcoin blockchain has rapidly evolved since its first appearance in 2008 by [145] and is viewed as the decentralised bank of the bitcoin crypto-token, where transactions are digitally signed, processed in a peer-to-peer protocol and stored in an immutable sequence of blocks known as the blockchain network. The functionality of Bitcoin has potentially lured many of the public worldwide due to its strong security as an immutable ledger and its quasi-anonymity of its transactions [35]. The term 'quasi-anonymity' explains the hidden identity but the transparency and traceability of transactions on the blockchain. Due to its quasi-anonymity, criminals have perceived Bitcoin as a facilitator for financial crimes such as money laundering which is the process of disguising illegal funds for legitimate ones by mixing or exchanging [152]. On the other hand, the public availability of crypto-coin data has promoted cryptocurrency intelligence companies to provide anti-money laundering (AML) solutions. In the early years of the Bitcoin launch, analysing the flow of payments and identifying illegal services on the Bitcoin graph network have arisen using a heuristic clustering procedure such as in [135]. However, analysing the graph network of Bitcoin transactions has become burdensome with the massive growth of Bitcoin data in recent years. Meanwhile, machine learning has adequately become a prominent way to cope with a large amount of Bitcoin data to detect illegal services, such as the case study [221]. The original work in [221] has proposed an AML solution on the public Elliptic data, one of the largest data of Bitcoin transaction graphs that incorporates more than 200k nodes as transactions and approximately 234k edges as payments flow. Referring to Figure 3.1, this data comprises 49 distinct directed acyclic graphs of Bitcoin transactions belonging to 49 time-steps and acquires partially labelled nodes of 21% licit transactions (e.g. miners) and 2% illicit transactions(e.g. scams, PonziScheme). Each node acquires 166 features, wherein the first 94 features (timestep + local features) are derived from the raw transaction information (e.g. the number of outputs/inputs, transaction fees, unique inputs/outputs,...) and the remaining 72 features (global features) are aggregated from the neighbouring transactions up to 1-hop from the main node referring to [221]. The original study on this data has performed a comparison between classical supervised learning and graph neural networks to classify licit/illicit transactions, wherein the random forest has outperformed all other methods. Hereinafter, a substantial number of studies have been conducted on this data such as in [12, 13, 162, 14, 8], where part of these studies has focused on novel models for improving the classification of licit/illicit transactions and some have experimented uncertainty estimations beside the predictions. Despite their promising results, these studies have failed to capture the 'dark market shutdown' event that occurred during the collection of this data at time-step 43. Besides the original features of the mentioned data, another study has proposed additional features by exploring the Bitcoin transaction graph using a set of random walks to the previous illicit transactions so-called GuiltyWalker method [154]. This work has contributed to a significant performance of accuracy of 98.3% using the random forest classifier, whereas the procedure used in [154] has induced, in an indirect way, some data leakage which is to be discussed.

We propose a novel approach as an AML solution to classify the licit/illicit nodes of the Bitcoin transaction graph using a recurrent graph neural network (referred to as RecGNN) to exploit the temporal behaviour and the Bitcoin graph structure of Elliptic data. My proposed model combines a modified version of the long short-term memory (abbreviated by M-LSTM) model, intertwined with evolving layer on its cell state, with the spatial graph neural network (GNN). Meanwhile, GNN involves a learnable matrix on local graph neighbourhoods up to k-hop to produce the graph embeddings. But, this aggregation of information between neighbouring nodes in GNN has restrained the classifier from further improvement on this data in accordance with previous studies. To mitigate this issue, I introduce two additional features on Elliptic data besides the local features. Given a certain node, its additional features, named antecedent neighbouring features (ANF), are derived from the representations of the antecedent nodes up to 1-hop of the given node. The overall approach imitates a sequential prediction model, in which node prediction is based on its immediate antecedent neighbouring node labels. I also train various supervised learning methods against RecGNN using the local features concatenated with the introduced features (LF+ANF) on Elliptic data. I evaluate and compare the performance of these models using accuracy, precision, recall, f_1-score, receiver-operation-curve (ROC) and area-under-curve (AUC). Moreover, I also discuss the performance of RecGNN in comparison to the best-performing models that appeared in previous studies on Elliptic data. As a result, I show that the proposed model RecGNN has attained the highest accuracy on this data outperforming all benchmark methods including a random forest for the first time. Lastly, I visualise a snapshot of the graph network of Elliptic data at timestep 43 (during the dark market shutdown event). Afterwards, I intuitively perform a backward reasoning procedure to predict the label of an antecedent node given the current node. Exploiting the temporal and graph structure of antecedent nodes with RecGNN reveals a very promising framework for combating money laundering in Bitcoin.

6.2.3 Methods

In this section, I present the RecGNN model that is used to perform node classification on the Bitcoin transaction graph. Then, I introduce the two additional features that I refer to by 'ANF' (antecedent neighbouring features).

6.2.3.1 RecGNN: Recurrent Graph Neural Netowork

The proposed model encompasses a modified version of LSTM with GNN and an additional linear layer to output the predictions.

6.2.3.1.1 LSTM: Long Short-Term Memory In my experiments, a slight modification is applied to the original LSTM by modifying cell state $\tilde{C_t}$ and C_t which leads to further improvement. I refer to the modified LSTM version by M-LSTM with the following modifications:

$$\begin{aligned} \tilde{C_t} &= tanh(W_{xc} * X_t + M_t + b_c), \\ C_t &= F_t \odot C_{t-1} + (1 - F_t) \odot \tilde{C_t}, \end{aligned} \tag{6.5}$$

and M_t is provided as:

$$\text{ReLU}(\text{EvolveLinear}(W_{hc} * H_{t-1}, W_t))),$$

where ReLU is the relu activation function, W_t is the weights of a linear layer, and EvolveLinear is similar to the EvolveGCN-O model proposed in [162] but with a linear layer instead of the graph convolutional network as defined in Algorithm 3. The modification in C_t (cell state) is performed on the hidden layers of the gated recurrent unit (GRU) referring to [46].

6.2.3.1.2 GNN: Graph Neural Network In this experiment, I refer to the GNN model as the one proposed in [140]. For various graph neural network models, refer to the comprehensive review in [242]. Generally, a graph neural network seeks to aggregate the information over node neighbourhoods in [140]. The GNN framework involves a learnable matrix as in a standard neural network but encompasses the neighbouring information of the central node. Thus, I can simply express GNN as follows:

$$x_i^{'} = \Theta_1.x_i + \Theta_2. \sum_{j \in \mathcal{N}(i)} x_i, \tag{6.6}$$

Algorithm 3 EvolveLinear

Require: :

- Linear layer with initialised weights: W_t.
- LSTM model with initialised weights.
- **Input:**
  - W_t: Weights of linear layer
  - H_t: Hidden features
- **Output:**
  - H_{t+1}: Evolved features

```
**function** $H_{t+1}$ = EvolveLinear($H_t$, $W_t$)
$W_{t+1}$ = LSTM($W_t$)
$H_{t+1}$ = $W_{t+1}$ * $H_t$
**end function**
```

where Θ_1 and Θ_2 are learnable matrices, $x_i^{'}$ is the embeddings of the node i with inputs x_i and x_j as the main node features and its neighbouring node, respectively. $\mathcal{N}(i)$ is the set of neighbouring nodes belonging to i. In Equation 6.6, the first term is a linear transformation of the node itself followed by aggregated features of the neighbourhood information from the source to the target node (current to the previous transaction).

6.2.3.1.3 RecGNN: Recurrent Graph Neural Network

Bitcoin data incorporates temporal information in a graph network of Bitcoin transactions. My matchless model exploits the temporal information first. Then, the hidden features are squashed by the ReLU activation function and forwarded to the GNN model in order to exploit the graph topology of the Bitcoin data. The graph embeddings are followed by ReLU and forwarded to a linear layer in order to output the node classification of licit/illicit transactions. More formally, RecGNN can be expressed as:

$$\begin{aligned} X_t^1 &= \text{ReLU}(\text{M-LSTM}(X_t)), \\ X_t^2 &= \text{ReLU}(\text{GNN}(X_t^1)), \\ X_t^3 &= \text{softmax}(\text{Linear}(X_t^2)), \end{aligned} \tag{6.7}$$

where X_t is the node feature matrix associated with graph $\mathcal{G}_t$; X_t^l is the hidden features in layer l. The softmax activation function is used to output class predictions.

The power of graph neural networks is involved in the aggregated information from the neighbouring nodes providing promising results in several fields e.g. social networks [229].

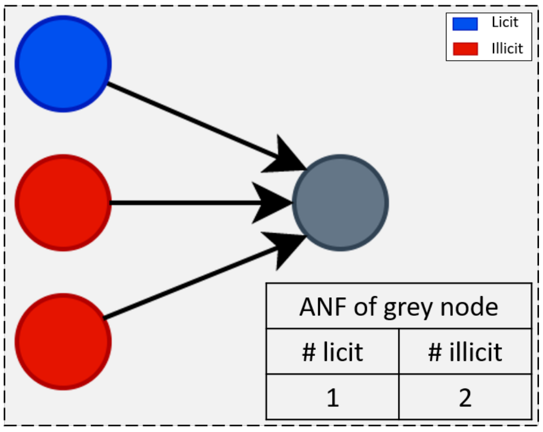


Figure 6.4: Example of extracting antecedent neighbouring features (ANF).

However, the studies on Elliptic data have shown that the classical random forest is superior to graph convolutional networks (GCNs). Thus, the embedding features with GCNs are subjected to mixed information once aggregated leading to lower performance. For this purpose, I propose additional features from the neighbouring nodes that are concatenated with the local features of Elliptic data and used in my experiments. This solution is assumed to alleviate any information loss caused by feature aggregation and provide further improvement by including additional well-designed and informative features.

6.2.4 ANF: Antecedent Neighbouring Features

Antecedent neighbouring nodes are the incoming transaction nodes to the node of interest. The idea of representing these nodes as a backup in the main node is due to the loss of information caused by the GNN model on this data. Representing the neighbouring nodes by concatenation in a fixed number of features is a challenging task. Intuitively, I propose two features that correspond to the two types of labels licit and illicit, respectively. These new features are defined by the number of licit and illicit antecedent nodes that are attached up to 1-hop of the main node as depicted in Figure 6.4. If the labels of all previous nodes are not available, I assign zeros for both features. My approach can be viewed as a sequential node classification of the Bitcoin transaction graph, in which the new nodes are based on the antecedent nodes in the graph network. Thus, I can express the predictions of a node x_i associated with y_i as the class label as follows:

$$\hat{y}_i = P(y_i | x_i, \hat{y}_{\mathcal{N}(i)}) \tag{6.8}$$

where y_i is the class predictions and $\hat{y}_{\mathcal{N}(i)}$ are the labels derived from the antecedent neighbouring nodes of node i to be included as ANF features. I also note that GNN here is supposed to explore the neighbouring information up to 1-hop. On the other hand, the antecedent nodes

Table 6.6: Evaluation of models' performance using the overall features (AF+ANF): Experimental results of various supervised learning algorithms on Bitcoin data.

Model$^{(\mathrm{LF+ANF})}$	**% Accuracy**	**% Precision**	**% Recall**	**% F1-score**
RecGNN	**98.99**	**97.9**	**86.33**	**91.75**
Random Forest	97.77	91.68	72.29	80.84
Extra Trees	97.61	89.51	71.74	79.65
Bagging	97.75	91.75	71.92	80.64
AdaBoost	97.75	91.26	72.39	80.74
Gradient Boosting	97.39	99.84	60.01	74.97
MLP	97.37	87.12	69.99	77.62

also acquire information from their ancestors. Thus, the proposed model technically explores the graph network up to 2-hop.

6.2.4.1 Data Preparation

As mentioned earlier, I use Elliptic data that is presented in Chapter 3. Here, I use the local features (LF) excluding the time-step and concatenated with two additional features (ANF) that I refer to as "LF+ANF" which in total counts to 95 features. For a fair comparison, I follow the same procedure as in [221] for choosing the train/test split. The first 34 graphs belong to the train set and the remaining 15 are used for testing referring to Figure 3.1. The distribution of licit/illicit transactions in train/test sets is summarised in Table 3.6.

6.2.4.2 Classical Supervised Learning Model

To provide strong evidence about the performance of RecGNN, I benchmark the proposed model against tree-based algorithms as in [12] as follows:

- Random Forest
- Extra Trees
- Bagging
- AdaBoost
- Gradient Boosting

In this study, I opt for the above-listed supervised learning methods that have performed better than the graph convolutional network used in [221] with the described data.

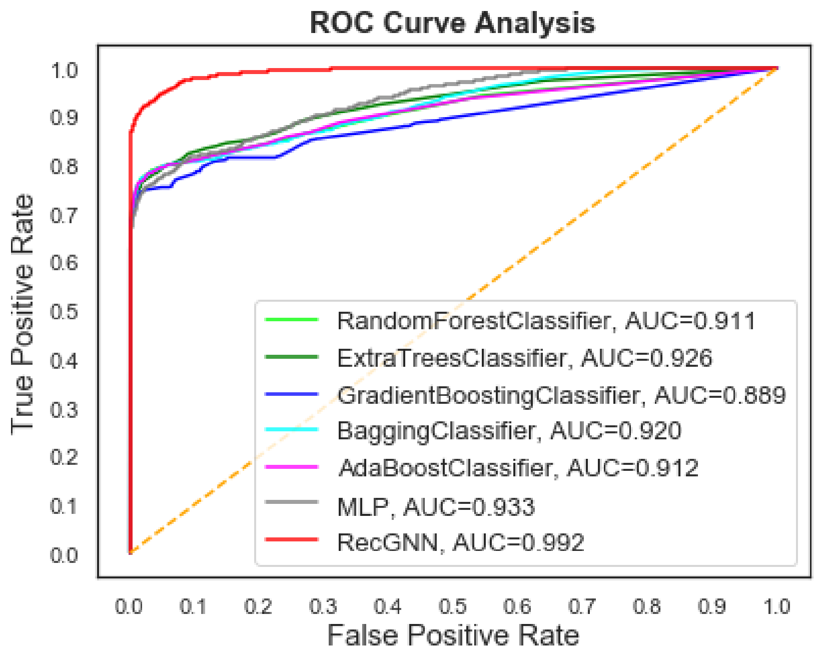


Figure 6.5: ROC-AUC plot on Bitcoin data.

6.2.5 Experiments and Results

This section presents the experimental setup used to train the proposed RecGNN and other supervised learning models.

Regarding the RecGNN model, I use the PyG package (Pytorch Geometric package in Python programming language) to conduct my experiments [64]. I use the same hyper-parameters that are chosen in [13] on Elliptic data. Thus, the model is trained with 50 epochs, wherein the graphs are fed in 34 batches per epoch. Adam optimiser is used to train the model with a learning rate of 0.0015. The size of hidden layers (M-LSTM and GNN) is set to 50 and a dropout layer is added with a dropout ratio of 0.5. GNN learns the graph embeddings from the present node to its antecedent neighbours. I use the softmax output as mentioned earlier to provide the class predictions. I choose a non-weighted negative likelihood loss function. Also, I train the multi-layer perceptron (MLP) using the same procedure of RecGNN but with linear layers instead of LSTM and GNN to be included in the benchmark methods.

For the provided classical supervised learning algorithms, I use the scikit-learn package [165] to classify Elliptic data. Following the same settings in [12], I fit the train set to Random Forest with hyper-parameters $n\_trees = 100$ and $max\_depth = 50$, Extra Trees (same hyper-parameters as a random forest), Gradient Boosting with learning rate 0.01, AdaBoost and Bagging classifiers with Random Forest as the base estimator.

After training the models, I evaluate the performance of RecGCN using accuracy, precision,

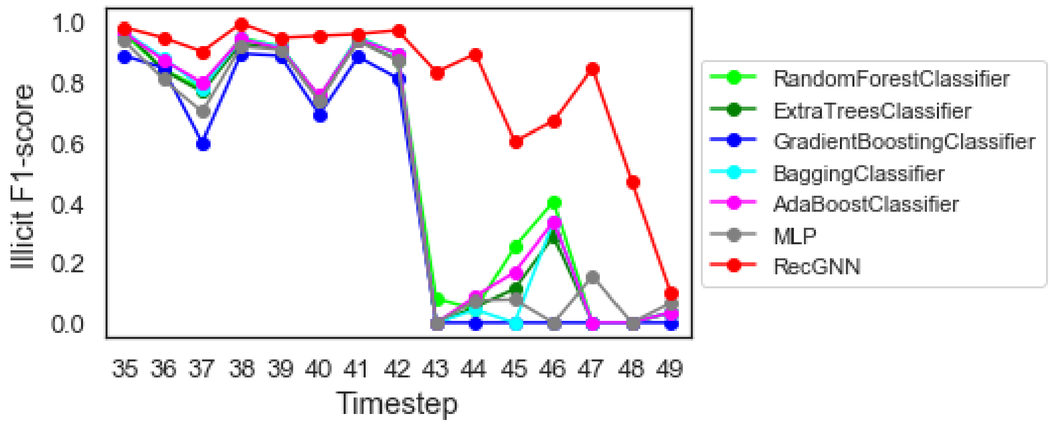


Figure 6.6: Illicit f_1-scores for test set versus timestep of supervised learning algorithms on Elliptic data.

recall and $f1$-score, then I compare it against the given supervised learning using the new set of features (LF+ANF) as shown in Table 6.6. I plot the ROC curve and present the AUC scores in Figure 6.5 to reveal the goodness of the classification of these models on "LF+ANF" features. In addition, I show f_1-scores of the illicit transactions per timestep on the test set of the given models that are provided in Figure 6.6.

To highlight the effect of "ANF" features, I evaluate the performance of RecGNN that is only fed with "LF" features. Moreover, I have also gathered the evaluations of the best-performing algorithms on Elliptic data from previous studies. I then compare RecGNN with "LF+ANF" against RecGNN with "LF" and evaluations from previous studies using accuracies and f_1-scores as tabulated in Table 6.7.

6.2.6 Discussion

The proposed model "RecGNN" has attained significant success over other benchmark methods, including random forest, with accuracy and f_1-score of 98.99% and 91.75%, respectively, using the "LF+ANF" set of features. Also, RecGNN has revealed the highest AUC score of value 99.2% achieving state-of-the-art on Elliptic data. Referring to Table 6.7, I show that RecGNN with "LF+ANF" features outperforms the same model with "LF" features only. Consequently, this explains the importance of the recurrent graph-based learning algorithm besides the proposed features. On the other hand, RecGNN with "LF+ANF" has revealed a noticeable outperformance over previous studies that attained the highest accuracy of 98.3% using the GuiltyWalker method on Elliptic data, referring to Table 6.7. In addition, the work in [192] has reported an accuracy of 99.08% but with 62.81% precision on Elliptic data using the naive Bayes classifier. Also, the study in [228] has reported f_1-score of 90.27% on the same data but with 97.8% accuracy using MGC-LSTM. Since the latter two studies have followed different train/test splits, the experimental results of these studies cannot be fairly compared with ours.

Model	% Accuracy	% F1-score
RecGNN$^{\text{LF+ANF}}$	**98.99**	**91.75**
RecGNN$^{\text{LF}}$	97.42	76.98
Random Forest [221]	97.4	77.3
Ensemble Learning [12]	98.13	83.36
Graph Convolutional Network [13]	97.4	77.3
Random Forest + GuiltyWalker [154]	98.3	85
ASXGBoost [212]	96.1	71.8

Table 6.7: Experimental results of RecGNN with LF vs LF+ANF features. Comparison of results between the previous studies and ours on Elliptic data.

On the other hand, an interesting event, called "The Dark Market Shutdown" in [221], has occurred at timestep 43. This event is the sudden closure of the black market during the time span of this Elliptic data. At this timestep, supervised learning models have failed to capture illicit transactions in previous studies using the original features of the data. My proposed approach has revealed a significant improvement in capturing illicit transactions during this timestep of f_1-score greater than 80% as depicted in Figure 6.6.

Regarding the limitations of my approach, I plot the total number of illicit transactions and their true positives under RecGNN as shown in Figure 6.7. I note that RecGNN is capable of matching the true labels of illicit transactions most of the timesteps, whereas its poorest performance is revealed at the 49th timestep. This is due to the topology of the Bitcoin transaction graph of Elliptic data at this timestep, where the illicit transactions are not associated with any incoming transactions. So, there are no available "ANF" features in these transactions. Furthermore, a real-time framework of my approach is based on sequential prediction in which the time complexity of the predictions increases with the number of nodes.

6.2.7 Visualisation and Case-Study

Visualisation is an inevitable part of AML compliance. Using the Pyvis package in Python, I plot a snapshot of the Bitcoin transaction subgraph network of Elliptic data at timestep 43 as depicted in 6.8 in which the dark market shutdown has occurred. Clearly, it is very hard to trace any source of funds using only the visualisation tools. However, a little visualisation is still required to support the explainability of the model. Referring to Figure 6.8, I pick an arbitrary illicit transaction such that it is incorrectly predicted by RecGNN as a case study. I plot this node up to 1-hop neighbourhood as shown in Figure 6.9. The node of interest is indexed by "442" which represents the order of this transaction in Elliptic data at timestep 43. Furthermore, this transaction incorporates an in-degree transaction indexed "441" with an unknown label (not provided by Elliptic) and out-degree transactions indexed "304" (licit) and "4636" (illicit). The reason behind studying this incorrectly predicted transaction is due

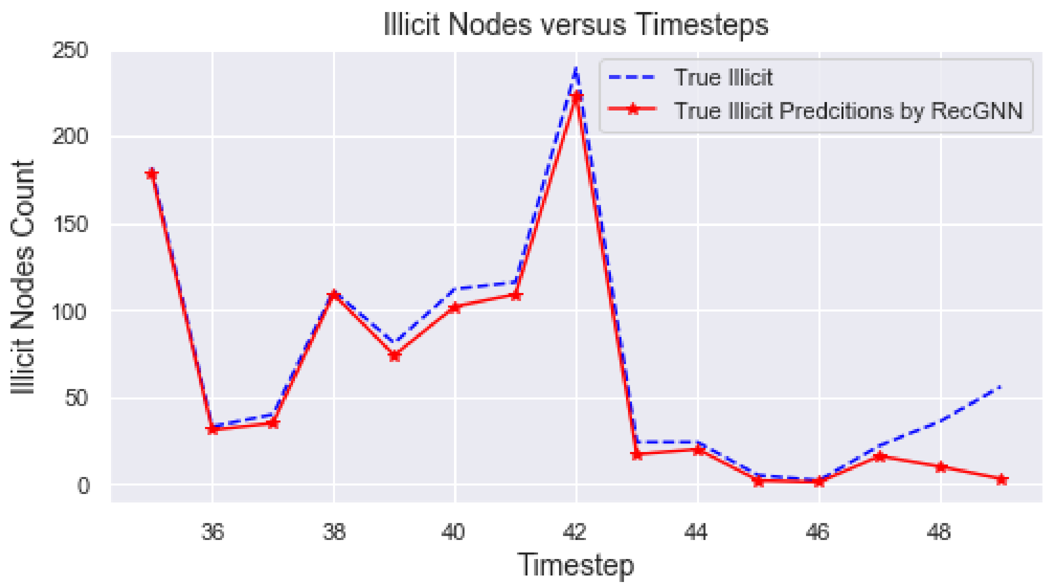


Figure 6.7: True Illicit number of nodes versus timesteps by RecGNN.

to the unknown label of the in-degree transaction in which ANF features of the chosen node are unavailable ((0,0) for (# licit, # illicit)). Intuitively, I perform backward reasoning for the unlabelled node by supposing two cases as follows:

1. Assuming "441" node transaction is licit, I assign (1,0) for the ANF set of features for "442" node transaction.
2. Assuming "441" node transaction is illicit, I assign (0,1) for the ANF set of features for "442" node transaction.

Using RecGNN, I predict the class of transaction "442" after considering each of the preceded cases. Consequently, the "442" node transaction is predicted as licit with the first case and as illicit with the second one. Thus, it would be reasonable to say that the unlabelled node "441" is an illicit transaction, wherein RecGNN correctly predicts "442" as illicit. Surprisingly, the node "441" is predicted as an illicit transaction after forwarding its features to RecGNN, which emphasises my reasoning process. The transaction with index "441" is originally associated with the transaction ID of "94336887" as an encrypted ID provided by the Elliptic company.

6.2.8 Concluding Remarks

To combat money laundering in Bitcoin, I have proposed a novel approach using a recurrent graph neural network (RecGNN) to predict illicit transactions in Elliptic data, a graph network of Bitcoin transactions. RecGNN, based on a modified LSTM and graph neural network, exploits the temporal and graph structure of graph-based data, wherein the nodes belong

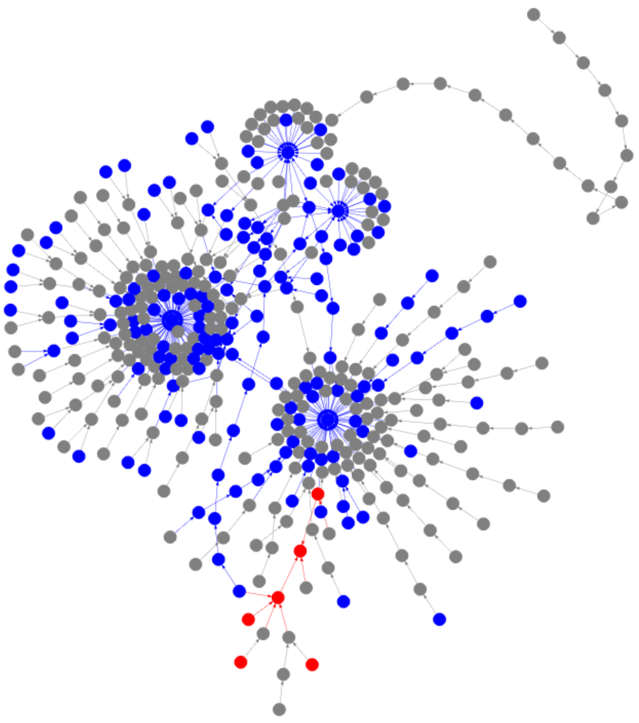


Figure 6.8: Snapshot of Bitcoin transaction subgraph of Elliptic data at timestep 43. Blue, red and grey coloured nodes belong respectively to licit, illicit, and unknown labelled transactions.

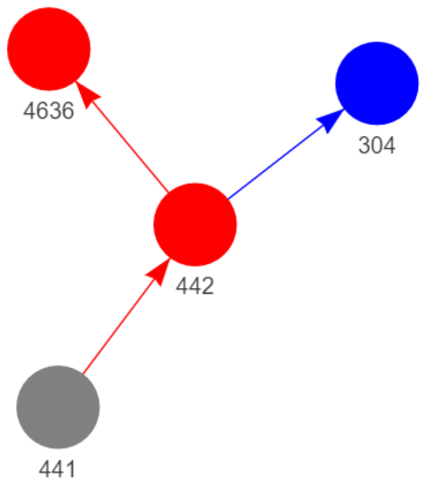


Figure 6.9: Snapshot of a transaction up to 1-hop neighbourhood of Bitcoin transaction subgraph of Elliptic at timestep 43. Blue, red and grey coloured nodes belong respectively to licit, illicit, and unknown labels of the transactions. The indices on the nodes denote the order of these nodes in the graph network of Elliptic at the provided timestep.

to real transactions in Bitcoin and the edges to the flow of payments. My novel approach incorporates a new set of features named antecedent neighbouring features (ANF). These features besides the original local features of Elliptic have attained an accuracy and f_1-score of 98.99% and 91.72% achieving the state-of-the-art using RecGNN, rather than its capability in capturing the dark market shutdown event at 43^{th} timestep. On the other hand, I have provided a case study to reveal the strength of "ANF" features in backward reasoning by predicting the antecedent node labels of an already identified label of the main node.

We will study the uncertainty of RecGNN to produce these predictions. Also, I will seek to develop an end-to-end framework that is featured with the predictions, backward reasoning and explainability of the predictions.

6.3 Summary

This chapter has presented two distinct frameworks using a machine learning approach to assist AML processes in the blockchain. The first framework has proposed an end-to-end active learning solution using the temporal-GCN model to classify illicit transactions and query the most informative data points using the AL approach with the BALD acquisition function to reduce the time required for the huge labelling process. This framework has provided promising results which mimic a real-world application of the human-in-the-loop approach. The uncertainty estimates derived from MI values have provided reliable uncertainty estimates as mentioned in Chapter 5. That's why I have computed MI measurements to perform the AL process where the MI measurements derived from MC-AA have performed better than random sampling. Also, the queried results by MC-AA have depicted consistency in improving the model's performance in comparison to MC-dropout. Regarding the limitations, the performance of the temporal-GCN model in this study does not show a better performance than random forest which is discussed in Chapter 3 even after using the graph structure and the temporal information of the Bitcoin transaction graph of Elliptic data. But I note that I have made use of the local features as input unlike the random forest classifier wherein I consider the whole features as input.

In the second framework, I have presented RecGNN – a graph-based learning algorithm – that uses the temporal information of the Elliptic data in the same way as the preceded model. Unlike temporal-GCN, RecGNN is based on sequential prediction which uses the labels of the antecedent nodes referred to as antecedent neighbouring features (ANF). Moreover, LSTM layers are applied in a novel by a simple modification at the layer level to boost the performance of the algorithm. This model has outperformed the different models proposed in the whole thesis and in the existing work of this dataset. Also, this model has shown effective results in capturing the black market shutdown event that occurred in the used dataset at timestep 43 which other models have failed to detect. The drawback of this model is that it requires the labels of the previously occurring nodes to perform predictions on the current one.

Chapter 7

Conclusions and Future Directions

Contents

The decentralisation virtue of public blockchain networks (e.g. Bitcoin) has attracted many around the globe to transact payments over the network without any intermediaries. Whereas the lack of central authorities has attracted criminals to anonymously conduct unlawful activities over this network. As the blockchain has gained increased popularity in the recent few years, researchers have crowdsourced the public blockchain data to track, explore and capture suspicious patterns and users. As a result, many conducted studies have focused on exploring the Bitcoin blockchain through visualisation, visual analytics tools, graph network algorithms, and recently via machine learning approach. Meanwhile, a massive amount of data is daily generated by the public blockchains where the detection of illegal activities requires an approach that digests the immensely generated data. At the commencing time of this project, there were very limited studies that adopted the machine learning approach to use unsupervised and some classical supervised learning methods. In this thesis project, I explore the performance of various supervised learning algorithms on two datasets derived from the Bitcoin and Ethereum blockchain. I perform data preprocessing and resampling then I study the feature importance and how it is influenced by the resampled data. As blockchain data is subjected to rapidly evolving events, machine learning models might not be able to capture the new events which the model has not learnt before. Thus, I perform uncertainty estimation and I develop a novel method that outperforms other existing uncertainty estimation methods. Lastly, I propose two different frameworks that point out the use cases of machine learning using blockchain data.

To this end, the main aim of this thesis project is to develop a novel machine learning approach to detect illegal activities conducted on the blockchain. In what follows, the conclusions of the performed experiments are provided followed by the limitations of this project. Subsequently, future directions and further perspectives are stated.

7.1 Conclusions

In Chapter 1, I provide a general introduction that briefly discusses the mechanism of the blockchain followed by the evolving interest in the blockchain system.

In Chapter 2, I provide a literature review which covers the comprehensive studies that have been performed to detect suspicious behaviour in the public blockchain networks such as Bitcoin and Ethereum ecosystems. The review covers a variety of approaches adopted to derive insights from the cryptocurrency based-blockchain system in an attempt to spot suspicious patterns.

In Chapter 3, I present the dataset used throughout the project which is derived from the Bitcoin blockchain. Interestingly, this dataset is one of the largest labelled datasets available in this field which incorporates time-series information and graph structure of Bitcoin transactions – so-called Elliptic data. This dataset is suited for machine learning to classify licit/illicit transactions in Bitcoin. Subsequently, I perform data preprocessing and resampling to reduce the dimensionality of the feature space and to address the class imbalance. With a lower number of features, I have shown that random forest, with an accuracy of 98.02% and f_1-score of 82.39%, outperforms other classical supervised learning methods as well as the models used in the original work of Elliptic data. Then, I examine various resampling techniques on this dataset. I find out that the edited nearest neighbour algorithm applied to the whole dataset using random forest provides the highest performance with an accuracy of 99.42% and f_1-score of 89.1%. This big improvement is due to the removal of noisy data points in the train and test sets at the same time. However, none of the used resampling techniques has shown any improvement in terms of the model's accuracy. On the other hand, resampling techniques have some influence on the distribution of the original dataset. For this purpose, I study the influence of the resampling techniques on the feature importance that is derived from the data distribution under a given model. The main motivation is that the feature importance is tied to the explainability of the given model which is favoured in this type of application. Henceforth, I perform a non-parametric statistical test – called the Wilcoxon test – in which I can have statistical evidence to say that the feature importance scores are influenced by the resampling technique. Out of three highlighted resampling techniques, only one (i.e, the SMOTE-SF technique) has shown evidence of the effect of data sampling on the involved features on the test set of the Elliptic data.

In my subsequent study, I examine the performance of various supervised learning methods using the whole number of features on this dataset. My main finding is that an ensemble learning model – based on averaging the probability predictions of bagging, random forest and ExtraTrees classifiers – provides the highest performance with an accuracy of 98.13% and f_1-score of 83.36%. The second-best performing metrics are recorded for the random forest classifier. However, the presented ensemble learning algorithm is computationally less efficient than the random forest. Afterwards, I explore the performance of graph learning models on this graph-structured dataset. Thus, I propose a GCN model combined with the MLP model which produces an adequate result with an accuracy of 97.4% and f_1-score of 77.3%. These results reveal the outperformance of my proposed model in comparison to the typical GCN model used in the original work of Elliptic data [221]. Despite the GCN model utilising the graph structural information, ensemble learning still outperforms the model-based GCN.

In Chapter 4, I present another example of a dataset derived from the Ethereum blockchain. This study is limited to one experiment as the dataset does not incorporate any graph structural information. This dataset comprises raw data of Ethereum accounts which are labelled between fraudulent and non-fraudulent account records. In this study, I perform a data preprocessing step in which I carry out a comparative study using supervised learning methods to detect fraudulent accounts. The classification results show that the XGBoost model outperforms all other classical supervised learning methods. This model is able to classify the fraudulent accounts with an accuracy of 98.91% and f_1-score of 97.6% which exceeds the performance of the classifications of the dataset's original work [61].

Then I address the class imbalance of this dataset by comprehensively applying various resampling techniques. I discuss the highest results by the edited nearest neighbour technique which removes the noisy instances in the train and test sets. Thus, XGboost using the undersampled dataset by the edited nearest neighbour technique attains an accuracy of 99.42% and f_1-score of 93.93%. Subsequently, I study the effect of the resampling techniques on the feature importance and explainability of the model. Similar to what preceded, I use the Wilcoxon test to verify my hypothesis. The resampling techniques SMOTE-SF and edited nearest neighbours are statistically significant; I can say that these resampling methods affect the feature importance derived from the train set of the Ethereum account dataset.

In Chapter 5, I cover the uncertainty estimation of the machine learning model. I first examine the MC-dropout uncertainty estimation on the Bitcoin dataset where I discuss its behaviour and limitations. Here, I find out that MC-dropout captures the data points that fall between different class distributions. Therefore, I propose a new uncertainty estimation method, abbreviated by MC-AA, which is derived from the adversarial attack idea to perform uncertainty estimation besides the model's predictions. This method shows its effectiveness and capability in capturing data points in the overlapping region of the class distributions in binary classification tasks. I examine this method with Bitcoin and Ethereum datasets and compare the performance of model uncertainty with the most recent uncertainty estimation method. As a result, MC-AA has outperformed the other used methods. Hence, I examine MC-AA with graph learning models for node classification tasks using Bitcoin and another dataset known as the GitHub dataset where MC-AA reveals a significant success over the MC-dropout method. To extend my method to multi-class classification tasks, I study the uncertainty of various classifiers using Cora and MNIST datasets. MC-AA shows limitations when applied to multi-class classifications due to the multiple class involved. Thus, I resolve this drawback by converting the multi-class problem to *one-versus-all* in which MC-AA reveals promising results.

In Chapter 6, I consolidate all my preliminary studies using the Bitcoin dataset by proposing two distinct frameworks using supervised models to classify the illicit transactions in the Bitcoin dataset. In this study, I exploit the graph structure and the temporal information simultaneously of the Elliptic data by using the LSTM model combined with GCN in both frameworks.

The first framework includes a temporal-GCN model using LSTM layers followed by TAGCN [57] in which the temporal information and graph structure data of Elliptic data are provided in a single model. This work presents an active learning solution to reduce the burden of the labelling process with the massive amount of Bitcoin datasets. I perform active learning using various acquisition functions. The uncertainty required by the active learning frameworks is computed using each of MC-AA and MC-dropout. Consequently, BALD querying data points using MC-AA provides consistency in the model's performance in comparison to MC-dropout. Moreover, a model with active learning shows significant success over a model that samples the training data randomly. This work imitates a real-life active learning application which produces promising results. Despite the proposed temporal-GCN model, the overall performance does not outperform the classical tree-based supervised models such as ensemble learning and random forest. For this purpose, I present a novel model – a recurrent graph neural network model – which I refer to as RecGNN. In this work, a new set of features is presented beside the dataset's local features of Elliptic data. The new features show significant success in the classification results. The idea is to include the labels of the antecedent nodes in the features of the current node. This way intuitively keeps the consistency of the class distributions where the new nodes require the very previous node labels. Consequently, this reduces the misclassifications caused by the evolving events which are not seen by the model. This model shows an accuracy of 98.99% and f_1-score of 91.75% achieving state-of-the-art on the Elliptic data. Moreover, this model successfully captures some of the black market shutdown event that occurred in this dataset. In the same experiments, I provide a case study to highlight the capability of performing backwards-reasoning for a given unlabelled node in an intuitive way.

7.2 Achievements in Correspondence to Research Objectives

In this section, I recall the research objectives then I demonstrate to what extent are these objectives achieved.

We restate the first objective which is:

- *To develop a machine learning method to capture illicit activities in the cryptocurrency blockchain*

We fulfil this objective by exploring and examining various supervised learning models to classify illicit activities in data derived from each of the Bitcoin and Ethereum blockchains. For the data derived from Bitcoin, I apply a variety of classical supervised learning, MLPs, LSTM and graph neural network models. Generally, tree-based bagging classifiers show significant success in classifying the experimented data at the transaction level. In particular, ensemble learning and random forest interchangeably provide adequate results to classify illicit transactions in Bitcoin. The former methods outperform graph neural network models as well. The models using LSTM layers show a slight improvement however this is at the expense of the model's complexity. The overall project provides the highest performance using the RecGNN model, which uses additional features on the given dataset. For the Ethereum dataset, I perform comparative analysis using classical supervised learning methods and MLP, where the XGBoost classifier produces the best classification results to classify fraudulent Ethereum accounts. The conducted experiments can be extended to other cryptocurrencies based-blockchain and indeed other blockchain data.

By recalling the second objective, it is:

- *To propose an uncertainty estimation method that assists the learning algorithms to obtain reliable predictions for enhanced decision-making*

Another good piece of study is on capturing the model uncertainty in the cryptocurrency blockchain. This is due to the new data points that appear in the Bitcoin network which might not be seen by the algorithm. For that, I study several uncertainty estimation methods which have shown success in previous studies. However, these methods are effective in capturing the uncertainty of points that fall into the different class distributions. Meanwhile, the data points corresponding to the black market shutdown event in the Bitcoin dataset do not reflect any type of uncertainty with the previous methods. Moreover, these points cannot be detected using the dataset's original features only. Hence, I present a novel uncertainty estimation method which is effective in including points in the overlapping regions between two class distributions. The performance of model uncertainty using the proposed uncertainty method on the Bitcoin and Ethereum datasets reveal promising results. The proposed uncertainty estimation is not limited to blockchain data and this method is capable of providing effective results in any data for binary classification problems.

The third objective to be recalled is:

- *To present an active learning approach as a solution that mimics the real-world human-in-the-loop approach*

We present a framework in Chapter 6 which proposes a temporal-GCN model. In this framework, I study an active learning approach using a pool-based approach where I use a pool of unlabelled and labelled sets to act like a human-in-the-loop process. This work provides acceptable outcomes in comparison to the random sampling labelling process.

The last objective is:

- *To evaluate the proposed methods using real-world cryptocurrency blockchain datasets*

All provided models are evaluated using the classification metrics of machine learning and other statistical measurements. I use two datasets derived from the Bitcoin and Ethereum blockchain. However, more studies are being conducted on the Bitcoin dataset due to the availability of the graph structure and temporal information of this dataset.

7.3 Limitations and Challenges

Despite the discussed achievements, I highlight the limitations and challenges arising in this project.

Ensemble learning models reveal a significant success but with high time complexity. The resampling techniques used in this study do not improve the classification results. Graph neural networks aggregate the feature of neighbouring nodes with the main node where this mechanism seems to reduce the classifier's performance. The evolving events induced by Bitcoin data cannot be fully captured by uncertainty estimation wherein such events are more likely to fall under the aleatoric uncertainty type. This type is hard to reduce unless more features within the data are provided. The public blockchain data is subject to new events which might change the distribution between the classes. In other words, it might be tricky to have a trained machine learning classifier that works well at any given time of this data. One could argue that LSTM is a good approach to tackle the latter issue. For instance, I examine LSTM using the given datasets which do not mitigate the misclassified predictions caused by the black market shutdown event of the Bitcoin dataset. The effective approach here is to include some informative feature that can keep the track of the transactions at any given time.

On the other hand, I realise that the data points belonging to illicit activities are not conceived as outliers. Consequently, unsupervised learning methods for outlier detection might not be a good practice in this type of application.

Other limitations could be outlined as follows:

- Hyper-parameters are manually tuned after running trials on only five values.
- Feature reduction is examined using only the Pearson correlation.
- The supervised learning methods are not tested for real-time detection.
- There is no significant pattern of illicit activity to be highlighted in the studied datasets.

7.4 Future Directions

In future work, other hyper-parameters should be examined instead of the manual search that is used in this project. Feature reduction can be examined using the statistical significance between the features, different correlation coefficient methods, and other feature reduction techniques. Furthermore, the real-time deployment and the exploration of the patterns derived from illicit activities are foreseen as interesting future work.

Another possible future direction is to investigate the outcomes of the new events in cryptocurrency blockchain data on a trained classifier. Also, I foresee studying the uncertainty estimates of such events. There are two possible hypotheses. I hypothesise that blockchain data belongs to some distribution in which I can perfectly fit a model using a sufficient and informative set of features. Eventually, this model should perform well at any time on this data. Another hypothesis is that blockchain data distribution is subject to changes over time. This hypothesis could be tackled using RecGNN introduced in Chapter 6. Although I have tried to address these statements, these should be examined with multiple blockchain datasets. These two hypotheses could be addressed by extending the current work to different datasets. Another resampling method could be explored that is based on generative adversarial networks (GANs) that work for graph-structured datasets. Other acquisition functions in the active learning approach could be examined in comparison to the presented experiments.

Bibliography

[1] 4chan: 4chan pass. `https://www.4chan.org/pass`.

[2] Ciphertrace (2020). spring 2020 cryptocurrency crime and anti-money laundering report. `https://ciphertrace.com/spring-2020-cryptocurrency-anti-money-laundering-report/`. Accessed: 2022-02-08.

[3] Crypto crime trends for 2022: Illicit transaction activity reaches all-time high in value, all-time low in share of all cryptocurrency activity. url=https://blog.chainalysis.com/reports/2022-crypto-crime-report-introduction/.

[4] Dan tynan danny yadron. tesla driver dies in first fatal crash while using autopilot mode. `https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk`. Accessed: 2021-07-16.

[5] Github social network.

[6] Wikileaks: Donate to wikileaks. `http://shop.wikileaks.org/donate#dbitcoin`.

[7] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 2021.

[8] I. Alarab and S. Prakoonwit. Adversarial attack for uncertainty estimation: Identifying critical regions in neural networks. *Neural Processing Letters*, pages 1–17, 2021.

[9] I. Alarab and S. Prakoonwit. Effect of data resampling on feature importance in imbalanced blockchain data: Comparison studies of resampling techniques. *Data Science and Management*, 2022.

[10] I. Alarab and S. Prakoonwit. Graph-based lstm for anti-money laundering: Experimenting temporal graph convolutional network with bitcoin data. *Neural Processing Letters*, Jun 2022.

[11] I. Alarab and S. Prakoonwit. Uncertainty estimation based adversarial attack in multi-class classification. *Multimedia Tools and Applications*, Jun 2022.

[12] I. Alarab, S. Prakoonwit, and M. I. Nacer. Comparative analysis using supervised learning methods in anti-money laundering of bitcoin data. 2020.

[13] I. Alarab, S. Prakoonwit, and M. I. Nacer. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, pages 23–27, 2020.

[14] I. Alarab, S. Prakoonwit, and M. I. Nacer. Illustrative discussion of mc-dropout in general dataset: Uncertainty estimation in bitcoin. *Neural Processing Letters*, 53(2):1001–1011, 2021.

[15] L. S. Ambati and O. El-Gayar. Human activity recognition: a comparison of machine learning approaches. *Journal of the Midwest Association for Information Systems (JMWAIS)*, 2021(1):49, 2021.

[16] A. M. Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies.* " O'Reilly Media, Inc.", 2014.

[17] M. N. Ashtiani and B. Raahemi. Intelligent fraud detection in financial statements using machine learning and data mining: a systematic literature review. *IEEE Access*, 2021.

[18] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare. Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 international conference on computing networking and informatics (ICCNI)*, pages 1–9. IEEE, 2017.

[19] R. M. Aziz, M. F. Baluch, S. Patel, and A. H. Ganie. Lgbm: a machine learning approach for ethereum fraud detection. *International Journal of Information Technology*, pages 1–11, 2022.

[20] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia. Dissecting ponzi schemes on ethereum: identification. *analysis, and impact*, 1, 2017.

[21] M. Bartoletti, B. Pes, and S. Serusi. Data mining for detecting bitcoin ponzi schemes. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 75–84, 2018.

[22] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, June 2004.

[23] A. Baumann, B. Fabian, and M. Lischke. Exploring the bitcoin network. In *WEBIST (1)*, pages 369–374, 2014.

[24] C. Bellei, H. Alattas, and N. Kaaniche. Label-gcn: An effective method for adding label propagation to graph convolutional networks. *arXiv preprint arXiv:2104.02153*, 2021.

[25] C. Bellinger, N. Japkowicz, and C. Drummond. Synthetic oversampling for advanced radioactive threat detection. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 948–953, Dec 2015.

[26] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[27] S. Bistarelli and F. Santini. Go with the-bitcoin-flow, with visual analytics. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–6, 2017.

[28] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[29] A. Bogner. Seeing is understanding: anomaly detection in blockchains with visualized features. In *Proceedings of the 2017 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2017 ACM international symposium on wearable computers*, pages 5–8, 2017.

[30] R. Böhme, N. Christin, B. Edelman, and T. Moore. Bitcoin: Economics, technology, and governance. *Journal of economic Perspectives*, 29(2):213–38, 2015.

[31] J. A. Bondy, U. S. R. Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[32] P. Boucher. How blockchain technology could change our lives. `https://www.europarl.europa.eu/thinktank/en/document.html?reference=EPRS_IDA%282017%29581948`, february 2017.

[33] J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

[34] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[35] C. Brenig, G. Müller, et al. Economic analysis of cryptocurrency backed money laundering. 2015.

[36] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[37] D. Buil-Gil and P. Saldaña-Taboada. Offending concentration on the internet: An exploratory analysis of bitcoin-related cybercrime. *Deviant Behavior*, pages 1–18, 2021.

[38] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '09, pages 475–482, Berlin, Heidelberg, 2009. Springer-Verlag.

[39] W. L. Buntine and A. S. Weigend. Bayesian back-propagation. *Complex systems*, 5(6):603–643, 1991.

[40] N. B. Bynagari and A. A. Ahmed. Anti-money laundering recognition through the gradient boosting classifier. *Academy of Accounting and Financial Studies Journal*, 25(5):1–11, 2021.

[41] N. B. Bynagari and A. A. Ahmed. Anti-money laundering recognition through the gradient boosting classifier. *Academy of Accounting and Financial Studies Journal*, 25(5):1–11, 2021.

[42] D. Cagigas, J. Clifton, D. Diaz-Fuentes, and M. Fernández-Gutiérrez. Blockchain for public services: a systematic literature review. *IEEE Access*, 9:13904–13921, 2021.

[43] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[44] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[45] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[46] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[47] N. Christin. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd international conference on World Wide Web*, pages 213–224, 2013.

[48] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, and A. Geissbuhler. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine*, 37(1):7 – 18, 2006. Intelligent Data Analysis in Medicine.

[49] J. Crawford and Y. Guan. Knowing your bitcoin customer: Money laundering in the bitcoin economy. In *2020 13th International Conference on Systematic Approaches to Digital Forensic Engineering (SADFE)*, pages 38–45. IEEE, 2020.

[50] J. B. Crawford. *Knowing your bitcoin customer: A survey of bitcoin money laundering services and technical solutions for anti-money laundering compliance.* PhD thesis, Iowa State University, 2019.

[51] A. Cuzzocrea, E. Fadda, and E. Mumolo. Cyber-attack detection via non-linear prediction of ip addresses: an innovative big data analytics approach. *Multimedia Tools and Applications*, 81(1):171–189, Jan 2022.

[52] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[53] G. Di Battista, V. Di Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia. Bitconeview: visualization of flows in the bitcoin transaction graph. In *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 1–8. IEEE, 2015.

[54] D. Di Francesco Maesa, A. Marino, and L. Ricci. Data-driven analysis of bitcoin properties: exploiting the users graph. *International Journal of Data Science and Analytics*, 6(1):63–80, 2018.

[55] G. Douzas and F. Bacao. Self-organizing map oversampling (somo) for imbalanced data set learning. *Expert Systems with Applications*, 82:40 – 52, 2017.

[56] G. Douzas, F. Bacao, and F. Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1 – 20, 2018.

[57] J. Du, S. Zhang, G. Wu, J. M. Moura, and S. Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

[58] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar. Topology adaptive graph convolutional networks, 2018.

[59] O. F. El-Gayar, L. S. Ambati, and N. Nawar. Wearables, artificial intelligence, and the future of healthcare. In *AI and Big Data's Potential for Disruptive Innovation*, pages 104–129. IGI Global, 2020.

[60] S. Eloul, S. J. Moran, and J. Mendel. Improving streaming cryptocurrency transaction classification via biased sampling and graph feedback. In *Annual Computer Security Applications Conference*, pages 761–772, 2021.

[61] S. Farrugia, J. Ellul, and G. Azzopardi. Detection of illicit accounts over the ethereum blockchain. *Expert systems with applications*, 150:113318, 2020.

[62] A. Fernández, S. del Río, N. V. Chawla, and F. Herrera. An insight into imbalanced big data classification: outcomes and challenges. *Complex & Intelligent Systems*, 3(2):105–120, 2017.

[63] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[64] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[65] M. Fleder, M. S. Kester, and S. Pillai. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*, 2015.

[66] N. Furneaux. *Investigating cryptocurrencies: understanding, extracting, and analyzing blockchain evidence.* John Wiley & Sons, 2018.

[67] A. Gaihre, Y. Luo, and H. Liu. Do bitcoin users really care about anonymity? an analysis of the bitcoin transaction graph. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1198–1207. IEEE, 2018.

[68] Y. Gal. Uncertainty in deep learning. *University of Cambridge*, 1(3):4, 2016.

[69] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[70] Y. Gal, J. Hron, and A. Kendall. Concrete dropout. *arXiv preprint arXiv:1705.07832*, 2017.

[71] Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[72] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[73] M. W. Gardner and S. Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[74] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[75] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[76] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[77] A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[78] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[79] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[80] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs, 2018.

[81] H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[82] H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In D.-S. Huang, X.-P. Zhang, and G.-B. Huang, editors, *Advances in Intelligent Computing*, pages 878–887, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[83] A. Handa, A. Sharma, and S. K. Shukla. Machine learning in cybersecurity: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1306, 2019.

[84] I. U. Haq, X. Du, and H. Jan. Implementation of smart social distancing for covid-19 based on deep learning algorithm. *Multimedia Tools and Applications*, Apr 2022.

[85] M. A. Harlev, H. Sun Yin, K. C. Langenheldt, R. Mukkamala, and R. Vatrapu. Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[86] M. U. Hassan, M. H. Rehmani, and J. Chen. Anomaly detection in blockchain networks: A comprehensive survey. *arXiv preprint arXiv:2112.06089*, 2021.

[87] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[88] J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015.

[89] J. Hirshman, Y. Huang, and S. Macke. Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network. *3rd ed. Technical report, Stanford University*, 2013.

[90] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[91] F. Holotiuk, F. Pisani, and J. Moormann. The impact of blockchain technology on business models in the payments industry. 2017.

[92] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[93] T. Hu, X. Liu, T. Chen, X. Zhang, X. Huang, W. Niu, J. Lu, K. Zhou, and Y. Liu. Transaction-based classification and detection approach for ethereum smart contract. *Information Processing & Management*, 58(2):102462, 2021.

[94] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[95] C. Jatoth, R. Jain, U. Fiore, and S. Chatharasupalli. Improved classification of blockchain transactions using feature engineering and ensemble learning. *Future Internet*, 14(1):16, 2022.

[96] Z. Jiang, T. Pan, C. Zhang, and J. Yang. A new oversampling method based on the classification contribution degree. *Symmetry*, 13(2):194, 2021.

[97] E. H. Johnson. Freeman: Elementary applied statistics: For students in behavioral science (book review). *Social Forces*, 44(3):455, 1966.

[98] E. Jung, M. Le Tilly, A. Gehani, and Y. Ge. Data mining-based ethereum fraud detection. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 266–273. IEEE, 2019.

[99] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–9, 2016.

[100] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In *Information visualization*, pages 154–175. Springer, 2008.

[101] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[102] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[103] S. Kethineni, Y. Cao, and C. Dodge. Use of bitcoin in darknet markets: Examining facilitative factors on bitcoin-related crimes. *American Journal of Criminal Justice*, 43(2):141–157, 2018.

[104] S. Khan and T. Yairi. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107:241–265, 2018.

[105] C. Kinkeldey, J.-D. Fekete, and P. Isenberg. Bitconduite: Visualizing and analyzing activity on the bitcoin network. In *EuroVis 2017-Eurographics Conference on Visualization, Posters Track*, page 3, 2017.

[106] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[107] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[108] A. Kirsch, J. Van Amersfoort, and Y. Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32:7026–7037, 2019.

[109] G. Kovács. An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. *Applied Soft Computing*, 83:105662, 2019. (IF-2019=4.873).

[110] G. Kovács. Smote-variants: A python implementation of 85 minority oversampling techniques. *Neurocomputing*, 366:352–354, 2019.

[111] N. Kshetri and J. Voas. Do crypto-currencies fuel ransomware? *IT professional*, 19(5):11–15, 2017.

[112] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[113] D. V. Kute, B. Pradhan, N. Shukla, and A. Alamri. Deep learning and explainable artificial intelligence techniques applied for detecting money laundering–a critical review. *IEEE Access*, 9:82300–82317, 2021.

[114] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[115] I. H. Laradji, M. Alshayeb, and L. Ghouti. Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, 58:388–402, 2015.

[116] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[117] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[118] H. Lee, S. Han, and J. Lee. Generative adversarial trainer: Defense to adversarial perturbations with gan. *arXiv preprint arXiv:1705.03387*, 2017.

[119] V. L. Lemieux. Blockchain and distributed ledgers as trusted recordkeeping systems. In *Future Technologies Conference (FTC)*, volume 2017, 2017.

[120] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[121] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. *CoRR*, abs/cmp-lg/9407020, 1994.

[122] G. Li, C. Xiong, A. Thabet, and B. Ghanem. Deepergcn: All you need to train deeper gcns, 2020.

[123] Y. Li, Y. Cai, H. Tian, G. Xue, and Z. Zheng. Identifying illicit addresses in bitcoin network. In *International Conference on Blockchain and Trustworthy Systems*, pages 99–111. Springer, 2020.

[124] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[125] J. Liang, L. Li, S. Luan, L. Gan, and D. Zeng. Bitcoin exchange addresses identification and its application in online drug trading regulation. In *23rd Pacific Asia Conference on Information Systems: Secure ICT Platform for the 4th Industrial Revolution, PACIS 2019*, 2019.

[126] M. Lischke and B. Fabian. Analyzing the bitcoin network: The first four years. *Future Internet*, 8(1):7, 2016.

[127] J. Lorenz, M. I. Silva, D. Aparício, J. a. T. Ascensão, and P. Bizarro. Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20, New York, NY, USA, 2020. Association for Computing Machinery.

[128] L. Ma and S. Fan. Cure-smote algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests. *BMC Bioinformatics*, 18(1):169, Mar 2017.

[129] X. Ma, G. Qin, Z. Qiu, M. Zheng, and Z. Wang. Riwalk: Fast structural node embedding via role identification. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 478–487. IEEE, 2019.

[130] D. J. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[131] D. D. F. Maesa, A. Marino, and L. Ricci. Detecting artificial behaviours in the bitcoin users graph. *Online Social Networks and Media*, 3:63–74, 2017.

[132] S. Maldonado, J. López, and C. Vairetti. An alternative smote oversampling strategy for high-dimensional datasets. *Applied Soft Computing*, 76:380–389, 2019.

[133] J. Martin. *Drugs on the dark net: How cryptomarkets are transforming the global trade in illicit drugs.* Springer, 2014.

[134] D. McGinn, D. Birch, D. Akroyd, M. Molina-Solana, Y. Guo, and W. J. Knottenbelt. Visualizing dynamic bitcoin transaction patterns. *Big data*, 4(2):109–119, 2016.

[135] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: Characterizing payments among men with no names. IMC '13, page 127–140, New York, NY, USA, 2013. Association for Computing Machinery.

[136] R. Michelmore, M. Kwiatkowska, and Y. Gal. Evaluating uncertainty quantification in end-to-end autonomous driving control. *arXiv preprint arXiv:1811.06817*, 2018.

[137] A. Mobiny, H. V. Nguyen, S. Moulik, N. Garg, and C. C. Wu. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *arXiv preprint arXiv:1906.04569*, 2019.

[138] A. Mobiny, P. Yuan, S. K. Moulik, N. Garg, C. C. Wu, and H. Van Nguyen. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 11(1):1–14, 2021.

[139] P. Monamo, V. Marivate, and B. Twala. Unsupervised learning for robust bitcoin fraud detection. In *2016 Information Security for South Africa (ISSA)*, pages 129–134. IEEE, 2016.

[140] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[141] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks, 2020.

[142] M. Möser, R. Böhme, and D. Breuker. An inquiry into money laundering tools in the bitcoin ecosystem. In *2013 APWG eCrime researchers summit*, pages 1–14. Ieee, 2013.

[143] A. Murko and S. L. Vrhovec. Bitcoin adoption: Scams and anonymity may not matter but trust into bitcoin security does. In *Proceedings of the Third Central European Cybersecurity Conference*, pages 1–6, 2019.

[144] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, 2015.

[145] S. Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system.(2008), 2008.

[146] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab. Machine learning for anomaly detection: a systematic review. *IEEE Access*, 2021.

[147] R. Neal. Bayesian learning for neural networks [phd thesis]. *Toronto, Ontario, Canada: Department of Computer Science, University of Toronto*, 1995.

[148] R. M. Neal. Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical report, Citeseer, 1992.

[149] R. M. Neal. Bayesian learning via stochastic dynamics. In *Advances in neural information processing systems*, pages 475–482, 1993.

[150] R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[151] P. Nerurkar, S. Bhirud, D. Patel, R. Ludinard, Y. Busnel, and S. Kumari. Supervised learning model for identifying illegal activities in bitcoin. *Applied Intelligence*, 51(6):3824–3843, 2021.

[152] J. Nicholls, A. Kuppa, and N.-A. Le-Khac. Financial cybercrime: A comprehensive survey of deep learning approaches to tackle the evolving financial crime landscape. *IEEE Access*, 2021.

[153] M. Ober, S. Katzenbeisser, and K. Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013.

[154] C. Oliveira, J. Torres, M. I. Silva, D. Aparício, J. T. Ascensão, and P. Bizarro. Guiltywalker: Distance to illicit nodes in the bitcoin network. *arXiv preprint arXiv:2102.05373*, 2021.

[155] E. Olshannikova, A. Ometov, Y. Koucheryavy, and T. Olsson. Visualizing big data with augmented and virtual reality: challenges and research agenda. *Journal of Big Data*, 2(1):1–27, 2015.

[156] E. Olson and J. Tomek. Cryptocurrency and the blockchain: Technical overview and potential impact on commercial child sexual exploitation. 2017.

[157] S. Omar, A. Ngadi, and H. H. Jebur. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications*, 79(2), 2013.

[158] I. Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, volume 192, 2016.

[159] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[160] M. Paquet-Clouston, B. Haslhofer, and B. Dupont. Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, 5(1):tyz003, 2019.

[161] M. Paquet-Clouston, M. Romiti, B. Haslhofer, and T. Charvat. Spams meet cryptocurrencies: Sextortion in the bitcoin ecosystem. In *Proceedings of the 1st ACM conference on advances in financial technologies*, pages 76–88, 2019.

[162] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5363–5370, 2020.

[163] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[164] N. Pawlowski, A. Brock, M. C. Lee, M. Rajchl, and B. Glocker. Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*, 2017.

[165] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[166] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[167] Ó. Pérez-Gil, R. Barea, E. López-Guillén, L. M. Bergasa, C. Gómez-Huélamo, R. Gutiérrez, and A. Díaz-Díaz. Deep reinforcement learning based control for autonomous vehicles in carla. *Multimedia Tools and Applications*, 81(3):3553–3576, Jan 2022.

[168] T. Pham and S. Lee. Anomaly detection in bitcoin network using unsupervised learning methods. *arXiv preprint arXiv:1611.03941*, 2016.

[169] T. Pham and S. Lee. Anomaly detection in bitcoin network using unsupervised learning methods. *arXiv preprint arXiv:1611.03941*, 2016.

[170] T. Pham and S. Lee. Anomaly detection in the bitcoin system-a network perspective. *arXiv preprint arXiv:1611.03942*, 2016.

[171] T. Pham and S. Lee. Anomaly detection in the bitcoin system-a network perspective. *arXiv preprint arXiv:1611.03942*, 2016.

[172] S. Phetsouvanh, F. Oggier, and A. Datta. Egret: Extortion graph exploration techniques in the bitcoin network. In *2018 IEEE International conference on data mining workshops (ICDMW)*, pages 244–251. IEEE, 2018.

[173] C. Pirtle and J. Ehrenfeld. Blockchain for healthcare: The next generation of medical records?, 2018.

[174] R. S. Portnoff, D. Y. Huang, P. Doerfler, S. Afroz, and D. McCoy. Backpage and bitcoin: Uncovering human traffickers. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1595–1604, 2017.

[175] F. Poursafaei, R. Rabbany, and Z. Zilic. Sigtran: Signature vectors for detecting illicit activities in blockchain transaction networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 27–39. Springer, 2021.

[176] K. Puntumapon and K. Waiyamai. A pruning-based approach for searching precise and generalized region for synthetic minority over-sampling. In P.-N. Tan, S. Chawla, C. K. Ho, and J. Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, pages 371–382, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[177] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):1–16, 2016.

[178] E. Ranjan, S. Sanyal, and P. P. Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations, 2020.

[179] W. Rawat and Z. Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[180] F. Reid and M. Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[181] D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[182] B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding, 2019.

[183] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, 57(7):2117–2135, 2019.

[184] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[185] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[186] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[187] B. Settles. Active learning literature survey. 2009.

[188] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. *Advances in neural information processing systems*, 20, 2007.

[189] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[190] D. Shen, G. Wu, and H.-I. Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[191] S. Shen, G. Jin, K. Gao, and Y. Zhang. Ape-gan: Adversarial perturbation elimination with gan. *arXiv preprint arXiv:1707.05474*, 2017.

[192] G.-Y. Sheu and C.-Y. Li. On the potential of a graph attention network in money laundering detection. *Journal of Money Laundering Control*, 2021.

[193] L. Smith and Y. Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018.

[194] P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. *Advances in neural information processing systems*, 8, 1995.

[195] M. Spagnuolo, F. Maggi, and S. Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security*, pages 457–468. Springer, 2014.

[196] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[197] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015.

[198] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[199] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[200] R. Tan, Q. Tan, P. Zhang, and Z. Li. Graph neural network for ethereum fraud detection. In *2021 IEEE International Conference on Big Knowledge (ICBK)*, pages 78–85. IEEE, 2021.

[201] S. Tasharrofi and H. Taheri. De-gcn: Differential evolution as an optimization algorithm for graph convolutional networks. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–6. IEEE, 2021.

[202] J. S. Tharani, E. Y. A. Charles, Z. Hóu, M. Palaniswami, and V. Muthukkumarasamy. Graph based visualisation techniques for analysis of blockchain transactions. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pages 427–430. IEEE, 2021.

[203] A. Thennakoon, C. Bhagyani, S. Premadasa, S. Mihiranga, and N. Kuruwitaarachchi. Real-time credit card fraud detection using machine learning. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 488–493. IEEE, 2019.

[204] H. Tian, Y. Li, Y. Cai, X. Shi, and Z. Zheng. Attention-based graph neural network for identifying illicit bitcoin addresses. In *International Conference on Blockchain and Trustworthy Systems*, pages 147–162. Springer, 2021.

[205] K. Toyoda, P. T. Mathiopoulos, and T. Ohtsuki. A novel methodology for hyip operators' bitcoin addresses identification. *IEEE Access*, 7:74835–74848, 2019.

[206] A. Turner and A. S. M. Irwin. Bitcoin transactions: a digital discovery of illicit activity on the blockchain. *Journal of Financial Crime*, 2018.

[207] A. B. Turner, S. McCombie, and A. J. Uhlmann. Analysis techniques for illicit bitcoin transactions. *Frontiers in Computer Science*, page 53, 2020.

[208] A. B. Turner, S. McCombie, and A. J. Uhlmann. Discerning payment patterns in bitcoin from ransomware attacks. *Journal of Money Laundering Control*, 2020.

[209] J. van Amersfoort, L. Smith, A. Jesson, O. Key, and Y. Gal. Improving deterministic uncertainty estimation in deep learning for classification and regression. *arXiv preprint arXiv:2102.11409*, 2021.

[210] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pages 9690–9700. PMLR, 2020.

[211] M. Vasek and T. Moore. Analyzing the bitcoin ponzi scheme ecosystem. In *International Conference on Financial Cryptography and Data Security*, pages 101–112. Springer, 2018.

[212] D. Vassallo, V. Vella, and J. Ellul. Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies. *SN Computer Science*, 2(3):1–15, 2021.

[213] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2018.

[214] N. Verbiest, E. Ramentol, C. Cornelis, and F. Herrera. Preprocessing noisy imbalanced datasets using smote enhanced with fuzzy rough prototype selection. *Applied Soft Computing*, 22:511–517, 2014.

[215] J. Wagstaff and M. Karpeles. Mt. gox bitcoin debacle: Huge heist or sloppy glitch, 2014.

[216] C. Wang, X. Wang, J. Zhang, L. Zhang, X. Bai, X. Ning, J. Zhou, and E. Hancock. Uncertainty estimation for stereo matching based on evidential deep learning. *Pattern Recognition*, 124:108498, 2022.

[217] J. Wang, M. Xu, H. Wang, and J. Zhang. Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *2006 8th international Conference on Signal Processing*, volume 3, Nov 2006.

[218] K. Wang, J. Pang, D. Chen, Y. Zhao, D. Huang, C. Chen, and W. Han. A large-scale empirical analysis of ransomware activities in bitcoin. *ACM Transactions on the Web (TWEB)*, 16(2):1–29, 2021.

[219] S. Wang, X. Wang, P. Zhao, W. Wen, D. Kaeli, P. Chin, and X. Lin. Defensive dropout for hardening deep neural networks under adversarial attacks. In *Proceedings of the International Conference on Computer-Aided Design*, pages 1–8, 2018.

[220] M. Weber, J. Chen, T. Suzumura, A. Pareja, T. Ma, H. Kanezashi, T. Kaler, C. E. Leiserson, and T. B. Schardl. Scalable graph learning for anti-money laundering: A first look, 2018.

[221] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. 2019.

[222] E. W. Weisstein. Correlation coefficient. *https://mathworld. wolfram. com/*, 2006.

[223] F. Wilcoxon. Individual comparisons by ranking methods. biometrics bulletin 1, 6 (1945), 80–83. *URL http://www. jstor. org/stable/3001968*, 1945.

[224] Wordpress. Pay another way: Bitcoin. `https://wordpress.com/blog/2012/11/15/pay-another-way-bitcoin/`.

[225] R. E. Wright. Logistic regression. 1995.

[226] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang. Detecting mixing services via mining bitcoin transaction network with hybrid motifs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[227] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng. Who are the phishers? phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[228] P. Xia, Z. Ni, H. Xiao, X. Zhu, and P. Peng. A novel spatiotemporal prediction approach based on graph convolution neural networks and long short-term memory for money laundering fraud. *Arabian Journal for Science and Engineering*, pages 1–17, 2021.

[229] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[230] D. Yaga, P. Mell, N. Roby, and K. Scarfone. Blockchain technology overview.(national institute of standards and technology, gaithersburg, md), nist interagency or internal report (ir) 8202, 2018.

[231] H. Yang, Q. Kong, W. Mao, and L. Wang. Boosting hidden graph node classification for large social networks. In *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2021.

[232] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[233] H. S. Yin and R. Vatrapu. A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3690–3699. IEEE, 2017.

[234] L. Yu, N. Zhang, and W. Wen. Abnormal transaction detection based on graph networks. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 312–317. IEEE, 2021.

[235] Q. Yuan, B. Huang, J. Zhang, J. Wu, H. Zhang, and X. Zhang. Detecting phishing scams on ethereum based on transaction records. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2020.

[236] D. Zambre and A. Shah. Analysis of bitcoin network dataset for fraud. *unpublished Report*, 27:2013, 2013.

[237] J. Zhang, J. Zhu, G. Niu, B. Han, M. Sugiyama, and M. Kankanhalli. Geometry-aware instance-reweighted adversarial training. *arXiv preprint arXiv:2010.01736*, 2020.

[238] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[239] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[240] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. IEEE, 2017.

[241] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

[242] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[243] J. Zhu, J. Zhang, B. Han, T. Liu, G. Niu, H. Yang, M. Kankanhalli, and M. Sugiyama. Understanding the interaction of adversarial training with noisy labels. *arXiv preprint arXiv:2102.03482*, 2021.

Resampling Techniques

Many resampling methods have been tested on the given datasets derived from Bitcoin and Ethereum blockchain. We apply various collection of oversampling, undersampling and hybrid sampling techniques on these datasets. The studied techniques are mostly implemented in smote-variants package in Python programming language as follows:

- Oversampling techniques: SMOTE, Borderline-SMOTE1, Borderline-SMOTE2, ADASYN, AHC, LLE-SMOTE, distance-SMOTE, SMMO, polynom-fit-SMOTE, Stefanowski, ADOMS, Safe-Level-SMOTE , MSMOTE , DE-oversampling, SMOBD, SUNDO, MSYN, SVM-balance, TRIM-SMOTE , SMOTE-RSB , ProWSyn, SL-graph-SMOTE, NRSBoundary-SMOTE , LVQ-SMOTE , SOI-CJ , ROSE, SMOTE-OUT, SMOTE-Cosine, Selected-SMOTE, LN-SMOTE, MWMOTE, PDFOS, IPADE-ID, RWO-sampling , NEATER, DEAGO, Gazzah, MCT, ADG, SMOTE-IPF, KernelADASYN, MOT2LD, V-SYNTH, OUPS, SMOTE-D, SMOTE-PSO, CURE-SMOTE, SOMO, ISOMAP-Hybrid, CE-SMOTE, Edge-Det-SMOTE, CBSO, E-SMOTE, DBSMOTE, ASMOBD, Assembled- SMOTE, SDSMOTE, DSMOTE, G-SMOTE, NT-SMOTE, Lee, SPY , SMOTE-PSOBAT, MDO, Random-SMOTE, ISMOTE , VIS-RST, GASMOTE , A-SUWO, SMOTE-FRST-2T, AND-SMOTE, NRAS, AMSCO, SSO, NDO-sampling , DSRBF, Gaussian-SMOTE, Kmeans- SMOTE, Supervised-SMOTE, SN-SMOTE, CCR, ANS, and cluster- SMOTE. In addition to all these oversampling techniques, OSCCD and SMOTE-SF are also used that are not implemented in the given package.
- Undersampling techniques: TomekLinkRemoval, CondensedNearestNeighbors, OneSidedSelection, CNNTomekLinks, NeighborhoodCleaningRule and EditedNearestNeighbors.
- Hybrid sampling techniques: SMOTE-TomekLinks and SMOTE-ENN.