

IONIAN UNIVERSITY

Department of Informatics



IONIAN
UNIVERSITY

-- Diploma Thesis --

**Cryptocurrency Price Prediction Using Deep Learning
Methods**

Ilias Tzanis

Supervising Professor: Andreas Kanavos

February 19, 2024

Supervising Professor

Andreas Kanavos, *Associate Professor,*
Department of Informatics

Committee of three

Andreas Kanavos, *Associate Professor,*
Department of Informatics

Katia - Lida Kermanidis, *Associate Professor,*
Department of Informatics

Vasileios Karyotis, *Associate Professor,*
Department of Informatics

Abstract

In recent years we have seen a huge increase in digital currencies and especially cryptocurrencies with the most prominent being Bitcoin, Ethereum, and Ripple. Investors, Financial experts, and the scientific community have various opinions, both positive and negative, regarding on the future of these currencies and on transactions with them. For a long-time, attempts to develop a reliable and accurate predictive model for cryptocurrency prediction have been made. Many machine learning and deep learning algorithms have been used by the researchers to predict and analyze factors that affect the cryptocurrency prices. The purpose of this thesis, is to focus on the challenge of cryptocurrency price prediction, with a particular emphasis on Bitcoin, due to its leading role in the market. Utilizing a comprehensive dataset that contains daily Bitcoin prices from September 24, 2019, to February 6, 2024, we developed and compared three distinct deep learning models: ARIMA (AutoRegressive Integrated Moving Average), SVM (Support Vector Machine), and LSTM (Long Short-Term Memory). Each model was carefully trained and tested to evaluate its effectiveness in predicting future cryptocurrency prices. Our findings reveal significant differences in the performance and reliability of each model, with the ARIMAX model demonstrating superior accuracy and forecasting capabilities. This research contributes to the ongoing dialogue on the effectiveness of predicting cryptocurrency prices and offers valuable insights for investors, financial experts, and the scientific community. By analyzing the predictive power of these models, we aim to shed light on the potential of deep learning methods in financial forecasting, particularly in the unstable and unpredictable realm of cryptocurrencies.

Acknowledgments

This project is devoted to my family and close friends, who consistently support me during challenging circumstances and assist me in attaining my objectives!

Contents

A	Introduction	1
B	Background and Literature Review	4
B.1	BACKGROUND	4
B.1.1	Cryptocurrencies	5
B.1.2	Blockchain	6
B.1.3	Bitcoin	8
	<i>B.1.3.1 Bitcoin Transactions</i>	10
B.1.4	Ethereum	11
	<i>B.1.4.1 Advantages of Ethereum</i>	12
	<i>B.1.4.2 Ethereum 2.0</i>	13
	<i>B.1.4.3 Smart Contracts</i>	14
	<i>B.1.4.4 EVM (Ethereum Virtual Machine)</i>	15
B.1.5	Ripple	16
	<i>B.1.5.1 XRP Ledger Consensus Protocol</i>	17
B.2	EXPOSITION OF DIVERSE FORECASTING MODELS	19
B.2.1	ARIMA	20
B.2.2	Garch	21
B.2.3	SVM	22
B.2.4	LSTM	24
B.3	LITERATURE REVIEW	25
B.3.1	Forecasting Ethereum: Deep Learning Insights by Zoumpekas, Houstis, & Vavalis [53]	25

B.3.2	Deep Learning in Cryptocurrency Forecasting: A Comprehensive Study by Maglogiannis, Iliadis, & Pimenidis [37]	26
C	Design Principles and Implementation	28
C.1	DESIGN PRINCIPLES	28
C.1.1	Dataset Overview	28
C.1.2	Python	29
C.1.3	Visual studio Code	31
C.2	IMPLEMENTATION	32
C.2.1	Implementation of Arima model	32
C.2.1.1	<i>ArimaX model</i>	37
C.2.2	Implementation of SVM	40
C.2.3	Implementation of LSTM	44
C.3	COMPARISON OF THE MODELS	46
D	Conclusions and future research	49
E	Appendix: Python code of the Prediction Models	51
Appendix		51
ARIMA MODEL		51
ARIMAX MODEL		53
SVM MODEL		56
LSTM MODEL		58
Abbreviations		68

List of Figures

B.1	<i>Example of a blockchain [52]</i>	7
B.2	<i>The Bitcoin paradigm [8]</i>	9
B.3	<i>Example of a Smart Contract [6]</i>	15
B.4	<i>Forecasting Models</i>	19
B.5	<i>Linear SVM</i>	23
B.6	<i>Differences Between RNN and LSTM [50]</i>	24
C.1	<i>Bitcoin Closing Price</i>	33
C.2	<i>Autocorrelation and Partial Autocorrelation Function</i>	33
C.3	<i>Model Diagnostic Plots</i>	35
C.4	<i>Arima Actual vs. Predicted Bitcoin Prices</i>	37
C.5	<i>ArimaX Actual vs. Predicted Bitcoin Prices</i>	39
C.6	<i>SVM Actual vs. Forecasted Bitcoin Prices</i>	43
C.7	<i>LSTM Actual vs. Forecasted Bitcoin Prices</i>	46
C.8	Comparison of Model Performance: ARIMAX, SVM, and LSTM Actual vs. Forecasted Bitcoin Prices	47

List of Tables

<i>B.1 Bitcoin Transaction Statistics:</i>	8
<i>B.2 Ethereum Transaction Statistics</i>	12
<i>B.3 XRP Transaction Statistics</i>	16
<i>C.1 Arima Diagnostics</i>	34
<i>C.2 Arima Performance Metrics</i>	36
<i>C.3 ArimaX Performance Metrics</i>	38
<i>C.4 SVM Model Performance Metrics</i>	42
<i>C.5 LSTM Model Performance Metrics</i>	45
<i>C.6 Summary of Model Performance Metrics</i>	48

Chapter A

Introduction

In recent years, the emergence of digital currency has become increasingly prominent, with a rapidly growing acceptance rate as a means of conducting transactions, surpassing that of traditional physical currency. Cryptocurrency is a type of digital currency that utilizes cryptographic functions along with blockchain technology. These features combine transparency, decentralization, and immutability [33]. The first and most well-known cryptocurrency is Bitcoin (BTC), which was created in 2009. Although Bitcoin and other cryptocurrencies initially experienced a gradual rise in popularity over the course of several years, they have subsequently undergone a swift and extensive expansion. We now have more than 5000 cryptocurrencies available, and the most popular alongside Bitcoin are Ethereum (ETH) and Ripple (XRP) [37]. The quantity of available cryptocurrencies indicates the emergence of the cryptocurrency market in the financial sector. As a result, studying and researching cryptocurrencies has piqued the interest of researchers. A very interesting research topic that has been explored by numerous scholars is that of predicting the future prices of cryptocurrencies. Although many will contend that accurate price forecasting is unachievable, examining various methods and seeing the outcomes of research are fascinating. Forecasting cryptocurrency prices can offer valuable assistance to investors in making well-informed investment choices, enabling them to optimize returns. Simultaneously, it can also aid policymakers in decision-making and assist financial researchers in examining the behavior of cryptocurrency markets [37]. There are two distinct approaches for predicting the dynamics of currency exchange rates. The first approach is similar to forecasting stock prices, where predicting cryptocurrency prices is considered a time series

challenge. This method involves making predictions based on the processing and analysis of past observations. A commonly used model in this approach is the Auto-Regressive Integrated Moving Average (ARIMA) model. The second approach involves employing machine learning, particularly deep learning algorithms. In this method, relationships between variables are modeled in a deep and layered hierarchy. By employing this approach, we can capture intricate non-linear patterns in complex prediction problems, resulting in enhanced performance when forecasting time series [42].

Machine learning involves the examination of algorithms capable of learning from experience. As a machine learning algorithm gains more experience, often through observational data or interactions with environments, its performance tends to enhance [51]. Deep learning refers to powerful machine learning algorithms that consist of multiple processing layers, enabling them to acquire representations of data characterized by various levels of abstraction. These algorithms tackle intricate and non-linear challenges, frequently harnessing extensive datasets to augment their efficiency as predictive models [28]. Consequently, capabilities in various domains, including genomics, speech recognition, visual object identification, and object detection, have experienced notable enhancements. Predicting cryptocurrency prices with precision is an intricate and challenging endeavor, given their substantial and dynamic variations over time, which manifest in almost chaotic and unpredictable patterns [37]. Consequently, the utilization of deep learning methodologies emerges as a valuable approach to navigating this complex problem.

In this thesis, the effectiveness of advanced deep learning algorithms is examined for forecasting the price dynamics of cryptocurrencies. The focus of this study is on addressing three key questions: (i) To what extent can statistical models and deep learning methods, including neural networks, efficiently forecast short-term fluctuations in cryptocurrency prices? (ii) How does the predictive accuracy of these models in cryptocurrency price prediction respond to external factors such as market sentiment, regulatory developments, and macroeconomic indicators? (iii) In what ways does the inclusion of traditional time series forecasting methods, improve the overall predictive capabilities of deep learning models in the context of cryptocurrency price prediction?

The structure of the remaining chapters in this research is as follows: Chapter B encompasses the Background and Literature Review, offering an in-depth exploration of foundational knowledge and existing literature relevant to the forecasting of cryptocurrency prices. Chapter C begins by analyzing Design Principles, which includes the dataset, an

evaluation of the programming language and a review of the utilized environment . It then moves on to Implementation, where it presents our three models: 'ARIMA', 'SVM', and 'LSTM'. Finally, Chapter D presents the Conclusions, synthesizing insights and discoveries resulting from the research efforts.

Chapter B

Background and Literature Review

B.1 Background

Forecasting prices in financial markets can be advantageous for several reasons. It can improve investment decision-making by leveraging historical data, provide valuable assistance to investors, aid policymakers in decision-making, and assist financial researchers in their analyses. A diverse array of methods and approaches has been tested and employed. Among the widely recognized prediction techniques, Neural Networks and Genetic Algorithms stand out as particularly popular. These methods adeptly uncover non-linear connections in time series sequences, often demonstrating superior performance compared to traditional linear approaches such as Box-Jenkins ARIMA models [53].

Forecasting cryptocurrency prices can prove to be quite challenging. The price dynamics of cryptocurrencies are influenced by numerous latent factors, and these key drivers have not been fully understood and identified yet [41, 12, 13]. The prevailing belief among the majority of researchers is that fundamental factors do not exert a significant influence on cryptocurrency rates. Instead, it is argued that their prices are primarily determined by the demand-supply ratio [16].

This chapter will concentrate on the background research related to cryptocurrency price prediction, deep learning, cryptocurrency terms, and will also delve into a comprehensive literature review on the subject.

B.1.1 Cryptocurrencies

Investors, entrepreneurs, regulators, and the general public have garnered substantial attention over the years for cryptocurrencies. The noteworthy fluctuations in cryptocurrency prices, along with claims that the cryptocurrency market lacks intrinsic value and may resemble a speculative bubble, have fueled widespread discussions. These concerns about the evasion of regulatory and legal oversight have prompted extensive public discourse, occasionally resulting in heightened regulations or even complete bans [21].

Cryptocurrencies possess the capability to streamline financial transactions by eliminating intermediaries, reducing transaction fees, ensuring accessibility to anyone with an internet connection, and providing enhanced security and anonymity[9, 39]. Nevertheless, the precise extent of actual economic value exchanged in transactions involving freely floating cryptocurrencies such as Ethereum’s Ether and Bitcoin’s BTC remains uncertain [21].

Bitcoin, proposed by Nakamoto in 2008 as the pioneering decentralized digital currency, marked a transformative milestone in the financial landscape [32]. While the term cryptocurrency was not used in the original proposal, Nakamoto pitched the project as a peer-to-peer ‘currency’ in a network and cryptography email list [36]. Since then, the cryptocurrency ecosystem has burgeoned, with over 5000 cryptocurrencies currently in existence[29]. Peer-to-peer electronic currency systems, or cryptocurrencies, enable internet payments to be transferred straight between parties without passing via a banking institution. As such, what sets cryptocurrencies apart from most conventional financial instruments is their infinite divisibility, non-physical form, and decentralized operation. Furthermore, blockchain contributes to the immunity of the traditional approach of the government regulating system and the decentralization of the cryptocurrency mechanism. As a result, the primary goal of cryptocurrencies is to enable transaction and transfer while confirming required security via the private or public key.

In contrast to traditional assets, the value of cryptocurrencies is determined by the strength of the algorithm that monitors each transaction, rather than relying on a physical asset, the national economy, or a company. The inherent value of cryptocurrencies is primarily derived from the confidence of their users, which is further reinforced by distinctive features and properties. The growing popularity of cryptocurrencies is driven by their decentralized, peer-to-peer structure, cost-effectiveness, and independence from governmental regulation.

In today's digital landscape, this form of currency is gaining massive interest and emerging as an independent payment mechanism, particularly in digital finance regimes worldwide. These factors have contributed to significant upswings in cryptocurrency trade volume, pricing dynamics, and volatility, leading to regular mentions in mainstream media [14].

B.1.2 Blockchain

Blockchain emerged as a decentralized system for managing transactions and data, originally conceived for the Bitcoin cryptocurrency. Since its introduction in 2008, Blockchain technology has gained considerable attention, primarily driven by its core attributes:

- **Encompassing Security.** It suggests that security is not limited to a specific aspect but includes a broad range of measures and considerations to ensure the overall safety, protection, and integrity of a system, process, or entity. In the context of Blockchain it implies addressing various security aspects such as data protection, encryption, access control, and overall system resilience to safeguard against potential threats and vulnerabilities comprehensively.
- **Anonymity.** Refers to the state of being anonymous or unidentified meaning that the identity of a person or entity is concealed or not readily apparent to others. Anonymity allows individuals to participate in activities, transactions, or communications without revealing their personal information or true identity.
- **Data Integrity.** Data integrity refers to the accuracy, consistency, and reliability of data throughout its lifecycle. It ensures that data remains unaltered and unchanged during storage, transmission, or processing. The concept of data integrity is crucial in maintaining the trustworthiness and reliability of information in various systems and databases.

All these attributes operate without the involvement of any central authority in transaction control [45]. Transactions between individuals or businesses often involve the intervention of a third-party entity for consolidation and management. Whether it's a digital payment or currency transfer, a bank or credit card issuer traditionally plays the role of a mediator, accompanied by transaction fees. This centralized transaction system is not limited to financial interactions; it extends to various fields such as gaming, music, and

software. In this conventional setup, a third party oversees and manages all data and information, diverging from a direct interaction between the two main entities involved in the transaction. Blockchain technology has emerged as a solution to this challenge, aiming to decentralize and enhance the efficiency of transaction systems.

Blockchain is a distributed database system that maintains an expanding record of data entries verified by participating nodes. This data is stored in a public ledger, documenting every completed transaction. Unlike centralized systems involving third-party organizations, it operates in a decentralized manner. All transactions conducted in Blockchain are openly shared and accessible to every node, enhancing transparency compared to centralized third-party transactions. Additionally, the anonymity of all nodes in the system contributes to a safer environment for transaction confirmation. Bitcoin was the pioneering application to adopt this technology.

A Blockchain consists of sets of data organized into a sequence of data packages known as blocks. Each block comprises multiple transactions (TX1–n), as illustrated in Figure B.1. As new blocks are added, it expands, creating a comprehensive record of all transactions. Verification of blocks within the network involves the application of cryptographic techniques. Each block includes transactions, a timestamp, the hash value of the preceding block (referred to as the "parent"), and a nonce—a randomly generated integer utilized to validate the hash. This concept ensures the integrity of the Blockchain up to the initial block, or "genesis block." Since each hash value is unique and immediately alters if any block in the chain is altered, the system effectively prevents fraudulent activities. The addition of a block to the chain is contingent upon a consensus mechanism wherein a majority of nodes in the network must concur on the block's validity and the transactions it contains.

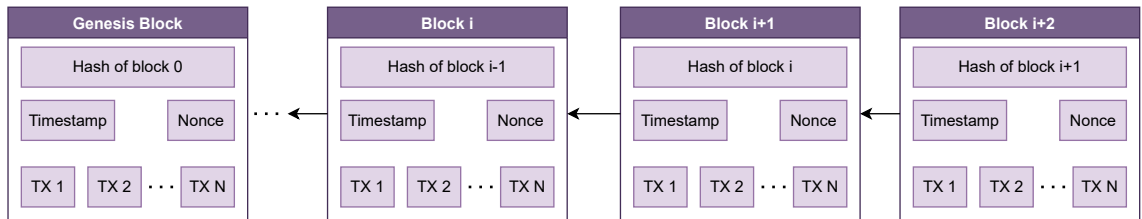


Figure B.1: *Example of a blockchain [52]*

This mechanism constitutes the method which a majority, or sometimes all, of the network

validators reach agreement regarding the state of a ledger [46]. It entails a set of rules and procedures facilitating multiple participating nodes to uphold a unified and accurate set of information. Consequently, the ledger doesn't instantly reflect new transactions; instead, they are held in a block for a specific duration (such as ten minutes in the Bitcoin blockchain) before being incorporated into the ledger through consensus process [35]. Following that, the data contained on the blockchain becomes immutable. Individuals can trust each other and transfer various types of assets peer-to-peer across the internet by using cryptography. Unlike centralized systems, the network remains operational even if a particular node experiences a malfunction. This increased resilience eliminates the need for individuals to assess the reliability of intermediaries or other members within the network, leading to enhanced trust.

B.1.3 Bitcoin

Bitcoin is a communication protocol that runs over the internet and makes it easier to utilize virtual currencies for electronic payments. It was created in 2009 by a group of developers that want to remain unknown (Nakamoto 2008). The daily transaction volume of Bitcoin is approximately \$650,040, based on the data for December 2023[1]. Table B.1 provides a concise overview of the Bitcoin transactions and operations that took place in December 2023.

Table B.1: *Bitcoin Transaction Statistics:*

Transactions avg. per hour:	27,085
Avg. Transaction Value:	0.5094 BTC (\$21,803)
Median Transaction Value:	0.0000054 BTC (\$0.233)
Avg. Transaction Fee:	0.00066 BTC (\$28.12) 355.1 sats/vB
Median Transaction Fee:	0.00038 BTC (\$16.26) 205.3 sats/vB

Source: <https://bitinfocharts.com/bitcoin/>

The foundation of Bitcoin is a decentralized ledger of transactions, maintained by a network of interconnected computers. Engineers established the protocols, and no regulatory body or attorney was visibly involved in the process [10]. The system's design incorporates features that promote genuine participation, facilitate early user adoption, and mitigate

the risk of power imbalances. Bitcoin is crafted to enable irreversible transactions, produce currency in a predetermined manner, and maintain a transparent public transaction record. It employs a distinct unit of value measurement known as bitcoin (indicated with a lowercase "b" and abbreviated as BTC). The system is intricate, encompassing elements such as cryptography, distributed algorithms, and incentivized behavior. Essentially, a bitcoin can be defined as a digital token that lacks any connection to a tangible asset or official currency, and does not represent a liability on any financial statement. The possession of bitcoins primarily entails the capacity to transfer these digital assets within the Bitcoin ecosystem. Users can create an account without incurring fees, bypassing centralized screening processes, and without disclosing their actual identity. In comparison to alternative payment methods, this system is perceived as more flexible, private, and resistant to regulatory oversight.

Figure B.2 depicts a schematic representation of the payment structure within the Bitcoin users' network. The entities within the system are represented as nodes, while the directional arrows symbolize the flow of payments in the form of bitcoin. According to the diagram provided, the entities engage in direct transactions. This stands in contrast to conventional payment systems, wherein intermediaries such as banks, processors, and networks typically mediate between the payer and the recipient. In the case of Bitcoin, however, there is no designated intermediary involved.

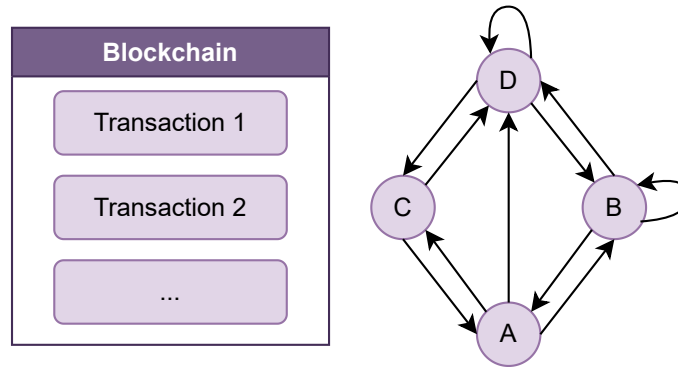


Figure B.2: *The Bitcoin paradigm [8]*

Note: In this scenario, entities A, B, C, and D directly transact with each other, without intermediaries. The diagram also suggests B and D could trade with itself. Blockchain public ledgers record all transactions chronologically.

Participants in the network diligently record each transaction in a publicly accessible ledger known as the blockchain, ensuring a chronological order of entries.

B.1.3.1 Bitcoin Transactions

In order to authenticate transactions, facilitate the processing of payments, and regulate the availability of bitcoins, the process of making Bitcoin transactions is dependent on the application of cryptographic techniques. There is nothing new about the cryptographic schemes that are utilized within the Bitcoin protocol; in fact, these schemes have been utilized in a variety of information security applications. Two cryptographic mechanisms, namely digital signatures and hash functions, are essential to the operation of Bitcoin [8]. When it comes to recording transaction details in a ledger that is accessible to the public, digital signatures allow for the secure transmission of precise payment instructions among participants in the transaction, while hash functions ensure compliance with established protocols. Digital signatures provide a means of verifying the authenticity of a message exchanged between a sender and a receiver, guaranteeing:

- **Integrity.** Digital signatures guarantee that the message remains unchanged during transmission, preventing tampering.
- **Non-repudiation.** Ensures that the sender cannot disavow sending the message, thereby establishing a degree of responsibility.
- **Authentication.** Digital signatures verify the identity of the sender by confirming that the message truly originated from the alleged sender.

Implementing digital signatures entails utilizing public key encryption, which involves a pair of public and private keys are created with specific advantageous characteristics. A cryptographic hash function takes an input string of any length and generates an output string of a predetermined length. The input, known as message m , and the output, represented as hash h , adhere to a deterministic pattern, guaranteeing that the identical input m will always produce the same output h . However, possessing knowledge of the hash value offers little to no insight into the original message. This attribute is crucial for hash functions and is formally articulated as follows:

1. **Pre-image Resistance.** Given a hash h , it is computationally difficult to find any message m such that $hash(m) = h$.
2. **Second Pre-image Resistance.** Given a specific message m_1 , it is computationally difficult to find a different message m_2 such that $hash(m_1) = hash(m_2)$.
3. **Collision Resistance.** It is computationally difficult to find two different messages m_1 and m_2 such that $hash(m_1) = hash(m_2)$.

These properties together enhance the overall security and dependability of cryptographic hash functions. They guarantee the resistance of hash functions against different types of attacks and ensure a high level of confidence in the integrity and uniqueness of hash values. Another favorable attribute of the hash function is that even slight modifications to the message m produce a substantial alteration in the hash value $h = hash(m)$. This significantly reduces the likelihood of deducing the content of the message based on its hash. In summary, hash functions yield results that appear highly unpredictable in their appearance, despite operating deterministically.

B.1.4 Ethereum

Ethereum, established in 2015, is the second most prominent digital currency. Compared to Bitcoin, it serves not only as a cryptocurrency but also as a platform for creating and executing smart contracts. It is a decentralized network of virtual machines that enables developers to execute distributed applications and utilizes ether as its native currency. These programs have the potential to encompass a wide range of functionalities, however, the network is specifically designed to execute them only when certain conditions are fulfilled, similar to a contractual agreement. Ethereum utilizes a decentralized and publicly accessible Blockchain to securely store, execute, and safeguard contracts through the implementation of cryptographic technologies. Every computer within the network downloads a compact virtual machine to synchronize with the Ethereum Blockchain and remains accessible for contract execution. This decentralized network of computers provides robust security, unwavering reliability, and substantial computational capabilities for various applications[30].

The daily volume of Ethereum transactions is approximately \$1,220,343, based on the data available for December 2023[1]. Table B.2 presents a succinct summary of the

Ethereum transactions and operations during December 2023.

Table B.2: *Ethereum Transaction Statistics*

Transactions avg. per hour:	50,848
Avg. Transaction Value:	1.68 ETH (\$3,713)
Median Transaction Value:	0.0000000004 ETH (\$0.00000089)
Avg. Transaction Fee:	0.005 ETH (\$10.96)
Median Transaction Fee:	0.0022 ETH (\$4.94)

Source: <https://bitinfocharts.com/ethereum/>

At this point it is crucial to distinguish between Ethereum and Ether. Ethereum refers to the decentralized network that is programmable and supports various functions. In contrast, Ether is the currency utilized on the network for conducting transactions. Ether operates similarly to Bitcoin as a value storage and exchange platform. The significant distinction lies in its utilization within smart contracts and its role in facilitating transactions within the contracts and the broader network, as all payments are conducted exclusively with it. The Ethereum network guarantees the security of ether transactions, similar to how Bitcoin transactions are safeguarded by its own network. The nodes responsible for validating transactions expend significant computational power to ensure their legitimacy.

Although the Ethereum network is considered secure because of its open source software, allowing for the identification of any vulnerabilities, the applications operating on the network lack a similar level of security. This is primarily due to the insufficient review and verification of the code used in these applications, despite their open source nature. The Ethereum network protocol is in continuous development, with the introduction of significant changes (Ethereum 2.0) that aim at enhancing the security, speed, and cost-efficiency of the decentralized network.

B.1.4.1 Advantages of Ethereum

Ethereum is a pioneering force in the world of digital currencies. It possesses distinct advantages in the dynamic field of blockchain technology due to its decentralized platform, smart contract capabilities, and thriving ecosystem. These advantages are:

- **Multi-functional Purpose.** Ethereum's objective extends beyond serving as a mere currency. In contrast to Bitcoin, its primary aim is to allow the creation and advancement of decentralized applications.
- **Efficient Transaction Processing.** The Ethereum Blockchain does not have a specific size limit. The number of transactions that can be included in a block is determined by the miners. At present, a block in Ethereum is found approximately every 12 seconds [2].
- **Comprehensive State Handling.** Ethereum has the ability to remember and maintain more states at the Blockchain level. Other cryptocurrencies like Bitcoin lack the capability to have states as they are limited to conducting transactions [5]. In contrast, Ethereum not only supports maintaining a monetary balance but also has the ability to handle contract code and data.
- **Turing Complete Architecture.** Ethereum is constructed using code that possesses Turing completeness [5]. A software or program is considered Turing complete if it contains sufficient system resources that can execute any code. This eliminates the requirement for specialized software or computers that are limited to running specific code.

Ethereum employs a blockchain that incorporates a Turing-complete programming language, allowing individuals to create smart contracts and decentralized applications. This platform enables users to create personalized rules for owning smart contracts, carry out various types of transactions, and implement functions that can be simple or complex, ultimately leading to changes in the stored data. This leads to the development of an alternative protocol designed for the execution of decentralized applications. It prioritizes scenarios where swift deployment, security for less frequently utilized applications, and the seamless interaction between diverse applications are crucial factors.

B.1.4.2 Ethereum 2.0

Ethereum has a rich and eventful past, marked by numerous advancements and occurrences that brought about notable improvements as well as setbacks. In September 2022, Ethereum underwent a significant transition from the proof-of-work (PoW) to the proof-of-stake (PoS) model[23]. Its primary objective is to optimize velocity and efficiency

(network throughput), while upholding stringent security measures and minimizing energy consumption. This is a courageous and demanding decision, given that the consensus mechanism is the pivotal characteristic of a decentralized system.

The PoS consensus algorithm is not only utilized in the Ethereum 2.0 network upgrade but also in numerous third-generation cryptocurrencies, as it is favored over the PoW algorithm. Through the utilization of PoS, the role of miners, who previously employed computational power to solve the mathematical problem, is effectively replaced by the validators. The validated users are chosen at random to verify transactions within a block and receive the corresponding reward in Ether.

Over the course of several years, the Ethereum developers have carefully polished the platform with the goal of long-term adoption. Thus, the upgrade wasn't an isolated incident. However, the Beacon Chain, which enabled ether staking[?], was introduced in December 2020, marking the official launch. The two current blockchains will combine into one, unified blockchain after Ethereum 2.0 is finished and thoroughly tested.

B.1.4.3 Smart Contracts

Smart contracts represent complicated transactions that function as a result of the complete digitization, storage, and verifiability associated with all components related to each individual transaction by the blockchain. They rely on the accuracy of their governing code, making programming errors potentially destructive. However, correct code guarantees the desired outcome consistently. Smart contracts have the capability to store data, engage with decentralized digital currencies, and receive user input. They utilize digital agreements on the decentralized protocol, allowing them to carry out both economic functions and application-related operations, such as rule enforcement, verification, and leasing. Smart contracts are designed to automatically execute agreement terms once pre-determined conditions are satisfied and provide a cost-effective alternative to traditional systems that rely on a middleman to carry out agreement terms, resulting in reduced transaction fees. A smart contract comprises three essential components: an account balance, a private storage space, and executable code. The contract's state consists of the allocated storage space and the current balance. The state is stored on the blockchain and is updated every time the contract is invoked by a user. In Figure B.3, we are able to observe the interaction among a smart contract and users as well as the blockchain[6].

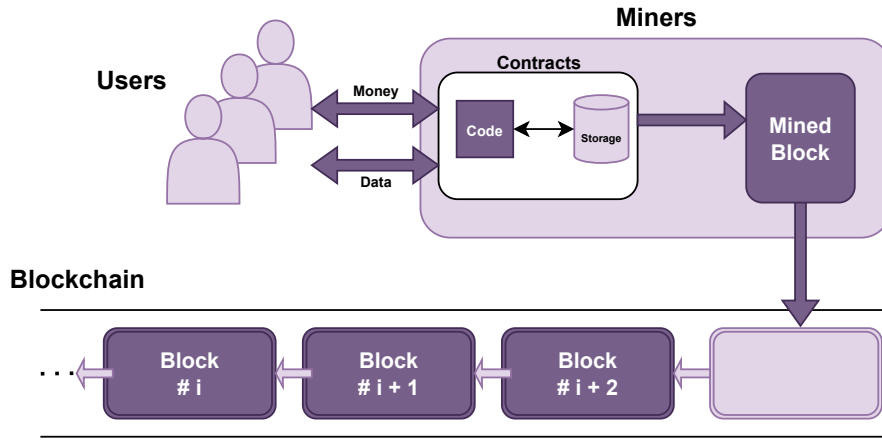


Figure B.3: *Example of a Smart Contract [6]*

In order to avoid infinite program execution and ensure equitable compensation for client applications that uses the smart contract system, it is necessary for the sender of each transaction to remunerate the miners for their involvement in the blockchain-based interaction. A fee is charged which is equivalent to the quantity of gas consumed by the contract. The network has reached a complete consensus on the fee schedule for execution. Every transaction also says how much gas it is willing to use and how much Ether it is worth in gas. In the event that a transaction exhausts its gas supply during execution, it halts, reverses any modifications made to the system's state, and the miners retain all of the gas fees associated with the transaction[22].

B.1.4.4 EVM (Ethereum Virtual Machine)

Ethereum utilizes the Ethereum Virtual Machine (EVM) to guarantee its programmability. It can be compared to a vast and worldwide computer made up of multiple individual computers that all use Ethereum's software. The Ethereum network relies on it as the computational engine for executing and verifying transactions, as well as for operating decentralized applications. The EVM is responsible for overseeing the creation and implementation of smart contracts within the Ethereum network and consists of numerous executable entities, each possessing its own dedicated storage area. It plays a crucial role in maintaining the programmability and flexibility of the Ethereum blockchain by allowing developers to deploy and execute code on a decentralized and trustless platform[22].

This worldwide computer is a state machine that is nearly Turing-complete and performs any computational process that modifies the data's state, excluding basic transactions involving accounts. However, the number of computational steps in executable processes is restricted by the amount of gas allocated for executing smart contracts.

B.1.5 Ripple

Ripple is the name of a payment network and the payment protocol, authored by Arthur Britto, David Schwartz, & Ryan Fugger[24]. It is a decentralized payment system that operates on credit networks and was developed before Bitcoin[20, 31]. At the beginning it was a payment processor that was similar to PayPal in many aspects. The Ripple network does not involve the mining process and transactions are verified by the participants of the network, which operates at a higher speed compared to Bitcoin. Approximately \$677,374 worth of Ripple XRP transactions occur every day, according to data available through December 2023[1]. Table B.3 provides a concise overview of the XRP transactions and operations that took place in December 2023.

Table B.3: *XRP Transaction Statistics*

Transactions:	567,086 txs
Payments:	129,796
Ledgers closed:	10,448
Tx rate:	14.106 txs/sec
Successful txs:	553,566

Source: <https://xrpscan.com/>

Ripple was created and made available in 2012 by a company with the same name. Its purpose was to facilitate secure, immediate, and almost cost-free global financial transactions. It is widely recognized as a cryptocurrency that shares similarities with Bitcoin in terms of its underlying principles. However Ripple's technological framework distinguishes itself from Bitcoin, primarily through the private ownership of its source code by the company. Ripple adopts a different strategy from Bitcoin, whose source code is available for public inspection. The XRP Ledger, which serves as the foundational blockchain protocol for Ripple, is not fully open-sourced. This proprietary nature of the XRP Ledger's source

code implies limited accessibility for external entities seeking to verify or scrutinize its inner workings. Nevertheless Ripple offers distinct open-source elements and utilities related to its technology. Although the company's blockchain protocol is proprietary, it has made certain aspects available for public use. Any individual possesses the capability to deploy a Ripple instance. In the Ripple network, nodes can assume three distinct roles[7]:

- **Users.** Engage in payment transactions.
- **Market Makers.** Facilitate trades.
- **Validating Servers.** Execute Ripple's consensus protocol to verify and validate all system transactions.

Ripple utilizes a set of public and private keys for transactions. When a user initiates a payment to another user, they authenticate the money transfer cryptographically using either Ripple's native currency, XRP, or any other currency[7]. It's important to note that Ripple cannot enforce payments for transactions conducted in currencies other than XRP. Instead, it maintains records of outstanding amounts owed between two parties. To elaborate, Ripple employs a decentralized credit network system in this specific instance. Users on the platform are identified using pseudonyms and money can be transferred instantly and directly between two parties thanks to the Ripple Transaction Protocol (RTXP). As a result, the protocol can avoid the costs and waiting periods associated with the conventional correspondent banking system. Additionally, any kind of currency, including dollars, euros, yen, gold, airline miles, and rupees, can be traded[24]. The security is ensured by using a shared ledger that is overseen by a network of autonomous validating servers, which consistently compare their transaction records. Servers may be owned by various entities, such as banks or market makers.

B.1.5.1 XRP Ledger Consensus Protocol

By leveraging the decentralized payment infrastructure of the XRP Ledger Consensus Protocol, individuals can easily and universally send money. It operates in a distributed peer-to-peer network and encounters similar obstacles as other digital currencies in preventing the duplication of funds and guaranteeing agreement across the network regarding the status of user accounts and balances. The XRP Ledger Consensus Protocol is a Federated Byzantine Agreement (FBA) protocol where each participant selects a validator

from a Unique Node List (UNL)[11]. While these validators may not be trusted on an individual basis, it is believed that they do not engage in collusion as a collective[47]. The Ledger contains a collection of nodes that operate using the rippled software[49]. The interconnected servers create a decentralized peer-to-peer overlay network with rippled architecture. The node owners set up the configuration to allow a specific number of incoming and outgoing connections[47]. Every outbound connection corresponds to an inbound connection at another node. When nodes establish a connection, the communication that occurs over the link is capable of transmitting data in both directions. The XRP Ledger Consensus Protocol features three main parts:

- **Deliberation:** A process where nodes suggest a set of transactions based on proposals received from reliable nodes. Following the attainment of a consensus, a node carries out the associated transactions on the preceding ledger, adhering to the rules outlined by the ledger protocol. Following this, it undergoes a validation process for the generated ledger.
- **Preferred Branch:** Refers to specific nodes determining the working branch of the ledger history. In instances of network difficulty or Byzantine failure, nodes may initially fail to achieve a consensus on validating the identical ledger. Nodes rely on trusted validations' ledger ancestry to resume deliberating on the network's preferred ledger, ensuring forward progress and complete validation of subsequent ledgers.
- **Validation:** Involves nodes making decisions on fully validating a ledger, relying on validations provided by trusted nodes. Upon reaching an adequate number of validations for the same ledger, that ledger and its preceding versions are regarded as fully validated. The status becomes authoritative and immutable.

Essentially, each peer issues a validation for a ledger, anticipating comprehensive validation. During civil executions, the process of deliberation increases the probability of reaching agreement with trusted peers and in atypical network situations, the preferred branch guarantees that peers reach a consensus on a specific branch for complete validation of the ledger.

B.2 Exposition of Diverse Forecasting Models

Researching financial markets and cryptocurrencies for the purpose of predicting future trends has been a longstanding subject of study. Accurately forecasting the path of market movements is crucial for numerous investment professionals in the market. A diverse range of strategies and techniques have been employed and utilized by market participants.

There are two separate methodologies that can be used to forecast changes in currency exchange rates. The initial approach shares similarities to the prediction of stock prices, as forecasting cryptocurrency prices is regarded to a time series activity. This approach entails generating forecasts by utilizing the processing and analysis of previous observations. The ARIMA model is frequently employed in this methodology. The second method involves making use of machine learning, specifically deep learning algorithms. This approach involves constructing a complex and multi-level hierarchy to represent the connections between variables.

By utilizing this methodology, we can effectively capture intricate non-linear patterns in complex prediction problems, leading to improved performance in forecasting time series. Figure B.4 illustrates the different methodologies employed in forecasting techniques.

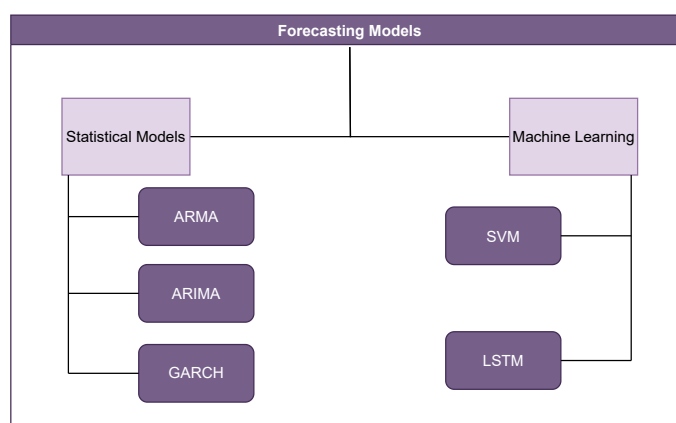


Figure B.4: *Forecasting Models*

B.2.1 ARIMA

Classical regression is frequently inadequate in elucidating the entirety of the captivating dynamics displayed by a time series. Instead, the inclusion of correlation that can arise from delayed linear relationships results in the proposal of autoregressive (AR) and autoregressive moving average (ARMA) models, as outlined in Whittle's work[48]. ARMA models are widely utilized for forecasting due to their versatility and applicability to various objectives. The model exclusively incorporates autoregressive and moving average processes, rendering it highly adaptable. The ARIMA model is an extension of the ARMA model. ARIMA models characterize the present behavior of variables by expressing it in relation to linear associations with their previous values. An ARIMA model can be partitioned into three components[34]:

- **The Autoregressive (AR)** component elucidates the correlation between the present value of the temporal sequence and a selection of its preceding values.
- **The Integrated (I)** component indicates the level of differencing required to transform the series into a stationary one.
- **The Moving Average (MA)** component represents the length of time that a random (unexplained) shock continues to have an effect.

A nonseasonal ARIMA model can be defined as a "ARIMA(p,d,q)" model, where[34]:

- **p** is the number of autoregressive terms,
- **d** is the number of nonseasonal differences needed for stationarity, and
- **q** is the number of lagged forecast errors in the prediction equation.

The forecasting equation is formulated in the following manner. Let y represent the d th variant of Y , indicating[34]:

- If $d = 0$: $y_t = Y_t$
- If $d = 1$: $y_t = Y_t - Y_{t-1}$
- If $d = 2$: $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$

The second difference of Y (when $d=2$) does not correspond to the difference from 2 periods ago. Instead, it refers to the first-difference-of-the-first difference, which is the discrete analog of a second derivative. In other words, it measures the local acceleration of the series rather than its local trend. The general equation for forecasting y is as follows:

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} - \theta_1 e_{t-1} - \dots - \theta_q e_{t-q}$$

The moving average parameters (θ 's) are defined with negative signs in the equation, in accordance with the convention established by Box and Jenkins[34]. While ARIMA is a powerful model for predicting time series data, the preparation and parameter tuning procedures can be time-consuming. Prior to implementing ARIMA, it is imperative to ensure that the series remains stationary. Therefore, it is important to determine the values of p and q .

B.2.2 Garch

The Generalized Autoregressive Conditional Heteroscedasticity (GARCH) model, was introduced by Engle (1982) and Bollerslev (1986)[19]. It is a statistical modeling technique that is highly regarded for accurately predicting short-term returns on financial assets, especially when these returns seem unpredictable. In a GARCH process, the error variances can be modeled using an ARMA process. The GARCH model has the advantageous capability of successfully reducing the excessive kurtosis present in returns. Assuming ε_t represents the innovation process, it can be expressed as:

$$\varepsilon_t | \psi_{t-1} \sim \mathcal{N}(0, h_t)$$

In this equation, h_t represents the conditional variance, ψ_{t-1} represents the set of all information available at time $t-1$, and $\mathcal{N}$ represents the conditional normal distribution. The GARCH(p, q) model, also known as GARCH- n , is defined as follows:

$$h_t = \alpha_0 + \sum_{i=1}^q \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^p \beta_j h_{t-j},$$

where $\alpha_0 > 0$, $\alpha_i \geq 0$, and $\beta_j \geq 0$ for $i = 1, 2, \dots, q$; $j = 1, 2, \dots, p$ [17]. GARCH models forecast the future volatility and assume that the time series is stationary, except for the variation in volatility, indicating the absence of a trend or seasonal pattern. These models demonstrate mean reversion and conditional heteroskedasticity, while maintaining an unchanged unconditional variance.

B.2.3 SVM

The field of machine learning encompasses four fundamental approaches: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Support Vector Machines (SVM) are supervised learning models that use specific algorithms to analyze data for classification and regression analysis. Classification is a type of supervised learning that involves mapping input data to output data using multiple example input-output pairs obtained during a training phase. This process involves utilizing features from a collection of example observations to train a decision function that accurately assigns class labels. These features can vary greatly, ranging from functional neuroimaging data to social media posts. Once the decision function "classifier" is constructed using these features, it can automatically assign class labels to new, unseen observations by applying the previously established trends. The SVM's strength lies in its capability to acquire data classification patterns with equal accuracy and consistency. While SVM is occasionally employed for regression (Support Vector Regression SVR), it has gained significant popularity as a versatile tool for classification across various types of data[38]. The advantages of Support Vector Machines include:

- The effectiveness when the number of dimensions exceeds the number of samples.
- The ability to work well with a clear margin of separation.
- The effectiveness in high-dimensional spaces.

The drawbacks include:

- Poor performance of SVM with extensive datasets.
- Reduced performance of SVM with noisy datasets containing a large amount of irrelevant information.

An SVM decision function is an optimal "hyperplane" that effectively classifies observations into different categories based on patterns of information, known as features. The hyperplane can be utilized to ascertain the most likely label for unobserved data. The features utilized to deduce the hyperplane are usually not in their original form, but rather they are frequently derived data obtained through interpolation during the feature selection phase. Features are identified and located by considering their relationship to

one another and serve as the support vectors. Working with SVM involves the task of reconciling two mutually beneficial objectives[38]:

1. Maximizing the accuracy of the classifier by assigning correct labels to new examples.
2. Optimizing the reproducibility of the classifier to be applicable to new data.

An SVM can be linear or nonlinear. The training process for an SVM decision function involves discovering a reliable hyperplane that maximizes the separation, referred to as the 'margin,' between the support vectors associated with each class label (Figure B.5). The most suitable hyperplane is the one that maximizes the margin across classes. The complexity of linear SVM problems varies depending on the number of features involved. If we consider a scenario with two feature dimensions, the hyperplane can be represented by a line. However, if we have three features, the hyperplane corresponds to a plane that has two dimensions.

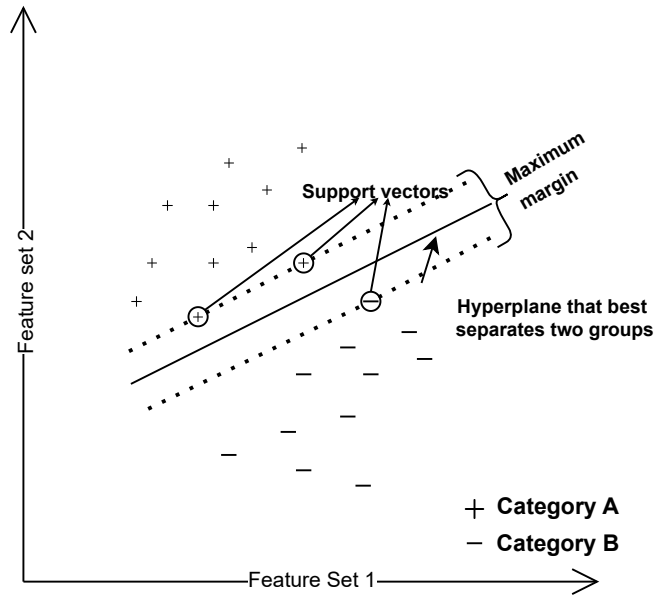


Figure B.5: *Linear SVM*

Note: Visualization of the hyperplane that effectively separates the support vectors to the greatest extent.

Despite the complexity of the SVM, classification problems typically involve linear hyperplanes, which are straight rather than curved[38]. If the features are assumed to be linearly separable, a straight hyperplane, known as a linear classifier, can be easily drawn on a graph of the features to separate the two labels of the class of interest. As a result, there are essentially two categories of margins that need to be maximized for a linear SVM.

B.2.4 LSTM

Machine learning focuses on the creation of algorithms that enhance their performance through practice. Optimally, the algorithm's performance improves with each iteration of the learning process. The learning algorithm is responsible for generating a classifier function based on the provided training data. The efficacy of this constructed classifier is subsequently evaluated by applying it to data that has not been seen before[44]. Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) are widely recognized as highly effective dynamic classifiers and have a significant influence in language modeling, speech-to-text transcription, machine translation, and various other applications.

The main distinction between RNN and LSTM lies in the temporal range they can consider. Figure B.6 demonstrates that RNNs utilize their internal state, also known as memory, to handle sequences of inputs. On the other hand, LSTM networks incorporate an additional long-term memory component to retain information from previous data.

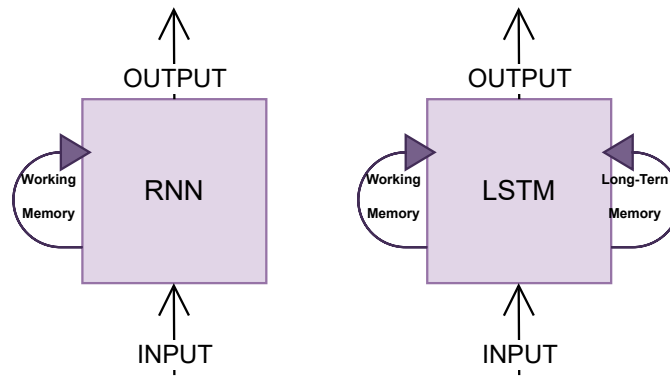


Figure B.6: *Differences Between RNN and LSTM [50]*

RNNs are constrained to examine the past for around ten time steps[44]. This occurs when the feedback signal either diminishes to zero or increases exponentially. This matter was resolved when LSTM-RNN was introduced. The primary component of an LSTM network is the memory block within the recurrent hidden layer. This block consists of memory cells that have self-connections, allowing them to retain temporal information[29]. Additionally, there are adaptive gate units that regulate the flow of information within the block. By addressing the concealed layer as a memory unit, LSTM has the ability to learn beyond 1,000 time steps, which depends on the complexity of the network architecture and can effectively manage the dependencies within time-series data across both brief and extended durations.

B.3 Literature Review

This section focuses on the literature review. The emphasis is specifically on showcasing article papers that possess a prominent research quality. Their empirical research shares several similarities with the research conducted in the subsequent chapters.

The present analysis tries to analyze and evaluate published articles related to cryptocurrency prediction-forecasting. By conducting reviews on other research, we can gain a better understanding of the methodology and techniques employed by other researchers. Additionally, this facilitates the comparison of the research that is covered in this thesis.

The analysis of Literature review can also provide valuable insights for other researchers and practitioners regarding the current topic. To conduct the research review, we utilized databases such as IEEE Xplore, Springer Link, and Google Scholar to ensure a thorough examination of research papers.

B.3.1 Forecasting Ethereum: Deep Learning Insights by Zoumpekas, Houstis, & Vavalis [53]

Cryptocurrency markets have emerged as an important area of research due to their rapid growth and volatility. The decentralized nature of these currencies, along with their technological background, presents unique challenges and opportunities for data analysis and predictive models. This paper makes a contribution to the growing field of research by using advanced statistical and machine learning methods to examine changes in prices and

make predictions about how Ethereum prices will end up.

Previous work in this area has extensively explored various aspects of cryptocurrencies, focusing particularly on Bitcoin due to its popularity and market capitalization. However, the rise of Ethereum as a leading platform for decentralized applications has increased interest in its market behaviors and predictive analysis. The research conducted by Zoumpekias et al. builds on existing studies, not only by analyzing the Ethereum market, but also by developing comparative deep learning models to predict its closing prices. Specifically, the study uses a Convolutional Neural Network (CNN) and four types of Recurrent Neural Networks (RNNs): Long Short Short Term Memory Networks (LSTM), Stacked LSTM, Bidirectional LSTM and Gated Recurrent Unit (GRU). These models are evaluated based on their prediction accuracy and their potential for real-time application to trading strategies.

Analysis of the literature reveals a growing degree of consensus on the effectiveness of deep learning approaches for financial market forecasting. The novelty of the Zoumpekias et al. approach lies in the integrated evaluation of multiple deep learning architectures and their application to the relatively less studied Ethereum market. Their findings highlight the potential to use these models for real-time predictions with promising accuracy, contributing to the development of expert systems for cryptocurrency trading.

This paper's focus on Ethereum and its research methodology aligns with the broader research path that seeks to understand the power of machine learning to navigate the complexities of cryptocurrency markets. It is an important step towards understanding the dynamics of these markets and developing tools for investors and analysts to make better and well-informed decisions.

B.3.2 Deep Learning in Cryptocurrency Forecasting: A Comprehensive Study by Maglogiannis, Iliadis, & Pimenidis [37]

The financial sector has witnessed the rapid emergence of cryptocurrencies as a major factor influencing investment tactics and market growth. The unpredictable nature of cryptocurrency prices, which is also marked by their random movements, poses a significant challenge for predictive modeling. Traditional time series forecasting methods, such as ARIMA, have been proven to be unsuccessful in capturing the non-linear complexity that is present in cryptocurrency price movements. In contrast, deep learning approaches,

with their ability to model complex and non-linear relationships in the data, have been considered to be potentially more capable at helping to effectively navigate the complexities of cryptocurrency markets.

This study provides a research that determines how well three deep learning algorithms can predict the values of the top three cryptocurrencies, Ethereum (ETH), Ripple (XRP), and Bitcoin (BTC). CNNs, LSTMs, and bipolar LSTMs were the models that were implemented. Despite the theoretical promise of these deep learning models for handling complex predictive tasks, the findings presented in the study question their practical utility for predicting cryptocurrency prices. The results highlight the mysterious nature of cryptocurrency price movements, showing that even the most advanced deep learning models are unable to produce accurate and dependable predictions.

Moreover, the study compares the performance of these models with traditional machine learning algorithms, finding only small improvements in prediction accuracy. This result suggests that the problem of predicting cryptocurrency prices may be approaching a random walk process, where predictability is minimal and historical data offers little insight into future price movements. Such a discovery questions the application of advanced forecasting models in the cryptocurrency sector.

By combining these insights, the paper not only contributes to the academic field of financial forecasting models, but also offers a reflection on the limitations of current technologies in dealing with the volatility of cryptocurrency markets. It opens the way to future research aimed at developing more sophisticated and effective models that can explore the complexity of cryptocurrency price behavior, helping investors and researchers to make more informed decisions.

Chapter C

Design Principles and Implementation

C.1 Design Principles

The design principles are presented in this section. This presentation includes the dataset, the selection of programming environments and the tools that have been used to implement the forecasting models. This decision influences not only the design and implementation but also the efficiency, reproducibility, and scalability of the research outcomes. In this section, we will analyze the dataset that forms the foundation of our study and explore the basic concepts of the design phase of our research. We will explain in a clear and detailed way the reasons behind selecting Python as the programming language, given its wide use in scientific computing and data analysis. Furthermore, we will discuss the choice of Visual Studio Code as our preferred integrated development environment (IDE). This choice is motivated by its versatility, extensive support for Python, and a robust set of extensions facilitating efficient development workflows. These decisions helped us ensure a balance between flexibility, performance, and ease of use.

C.1.1 Dataset Overview

The dataset that we used to train and test the models is formatted in a CSV file, featuring the following columns: *timeOpen*, *timeClose*, *timeHigh*, *timeLow*, *name*,

open, *high*, *low*, *close*, *volume*, *marketCap*, and *timestamp*. Each column provides a unique insight into the market's behavior on a given day, from opening and closing prices to daily highs and lows, volume traded, and market capitalization, with timestamps indicating the precise period of data capture.

The data collection spans over four years, and they capture a critical phase in the cryptocurrency market. This makes our dataset valuable for understanding market trends and predicting future prices. Each record inside the dataset represents a day of market activity for the Bitcoin cryptocurrency and provides a daily snapshot of its performance. These details allow for an in-depth analysis of price trends over time.

The data was sourced from CoinMarketCap, a provider that contains many cryptocurrency market data, ensuring its accuracy and reliability[3]. Many different steps were taken to verify the integrity of the data, including cross-referencing with other sources and validating the consistency of the data points. The dataset underwent cleaning and preprocessing before being used on each model. This included checking for handling any missing values, correcting any detectable anomalies, and ensuring the dataset was formatted correctly for analysis. More details of the preprocessing process are being presented in the implementation chapter of each model.

This dataset forms the foundation for developing each model that is presented in this thesis. Using it we aim to predict future Bitcoin prices based on historical data. Key features such as daily price movements, volume, and market capitalization were used for the train process, with a focus on identifying patterns that show significant price changes.

The dataset's coverage of Bitcoin's daily market activity from September 2019 to February 2024 provides a solid foundation in analyzing the complicated price trends of cryptocurrencies and predicting future movements. The datasets breadth and depth has a significant role in developing a robust model for cryptocurrency price prediction.

C.1.2 Python

Python is a high-level, object-oriented programming language that incorporates dynamic semantics. In combination with dynamic typing and binding, its high-level built-in data structures make it an extremely attractive language for Rapid Application Development, scripting, and tracking existing components[18]. Python is an easy to learn, straightforward and accessible programming language. The syntax prioritizes legibility and is simple

to maintain. One major advantage is the robust support that it has for modules and packages, helping at the organization of our programs as we can break them into smaller, reusable components. Python has become known as the favored programming language for data science because of a number of elements that serve both novices and seasoned industry professionals. The reasons Python does well in data science, and was chosen for our task such to predict cryptocurrency trends, are demonstrated below:

1. **Ease of Learning and Use:** Python is known for its simplicity and readability. This makes it an ideal choice for beginners. It has a straightforward syntax that resembles natural language, which helps beginners as well as experts to quickly write and understand code, reducing the learning curve and increasing productivity.
2. **Extensive Libraries and Frameworks:** Python has a rich ecosystem of libraries and frameworks specifically designed for data science and machine learning. Libraries like NumPy and pandas make the data manipulation and analysis more simple, while TensorFlow, PyTorch, and Keras are some of the many powerful tools that can help build and train deep learning models. These libraries offer pre-built functions and utilities to streamline the development of complex algorithms, making it easier to implement models.
3. **Community and Support:** Python has a large community of developers and data scientists. This community helps by contributing in the vast repository resources, including tutorials, forums, and documentation, which can be very useful in solving problems and improving skills.
4. **Versatility and Integration:** Python's versatility allows it to be used for the entire data science workflow, from data collection and cleaning to model training and deployment. It can be easily integrated with other languages and tools, enabling data scientists to leverage specialized algorithms or technologies as needed. Python also supports various data formats and sources. This is crucial for our task in analyzing cryptocurrency markets where data comes from diverse sources.
5. **Performance and Scalability:** While Python is not the fastest language, its performance is more than enough for most data science applications. With libraries like NumPy that are optimized for performance, and the ability to use Python together with faster languages like C or C++, it strikes a good balance between speed and

usability. Additionally, Python's compatibility with distributed computing technologies helps in scaling complex computations over large datasets.

6. **Visualization Tools:** Python offers powerful libraries such as Matplotlib, Seaborn, and Plotly, which are crucial for data exploration and the visualization of models' results. These tools have helped us to visualize the cryptocurrency market trends and predictions made by the statistical and deep learning models. Furthermore, it helps in understanding the complex dynamics of these markets.

Due to the numerous benefits pointed above, Python was selected as the programming language for this project. It is a user-friendly programming language and can effectively handle extensive datasets, execute and evaluate intricate algorithms, and display patterns and forecasts.

C.1.3 Visual studio Code

The selected editor for implementing our models was Visual Studio Code (VS code). It is a free and open source code editor developed by Microsoft for Linux, Windows, and MacOS operating systems. Visual Studio Code is a versatile tool that enables the implementation of a wide range of projects. It offers support for almost all the programming languages including JavaScript, Python, Java, and C++. The main reason for selecting VS code for our project's implementation is its support and wide range of extensions for Python-based projects. As an Editor, it is highly efficient, rapid, and simultaneously provides a plethora of features to the user. One of the features we previously mentioned are the extensions. By adding them to the project, we can enhance the Editor with additional features and functions. This makes possible to implement the Project at a faster pace, more efficient and expedient. The VS code Debugger is a highly valuable feature that allows us to identify and address different bugs present in the Project. It enables us to navigate to the source code, examine variables, and execute commands in the console. Additionally, it offers Git integration, allowing us to conveniently work with source control without having to switch to a separate application. Overall VS code is a straightforward and user-friendly editor that can be easily customized and it offers several useful features.

C.2 Implementation

This section outlines the practical aspects of applying statistical models and deep learning to predict cryptocurrency prices, specifically focusing on the Bitcoin dataset that was introduced on the Section C.1.1. We will discuss the implementation details of three different models: Autoregressive Integrated Moving Average (ARIMA), Support Vector Machine (SVM), and Long Short-Term Memory (LSTM) networks. Each model was selected for its unique approach to time series forecasting, offering a detailed perspective on the predictive performance of each model. The implementation process involves data preprocessing, model configuration, training, and evaluation, based on the specific characteristics and assumptions of each model. The goal is to analyze their effectiveness in capturing the complex dynamics of cryptocurrency markets and predicting future price movements with high accuracy.

C.2.1 Implementation of Arima model

The purpose of this section is to provide a comprehensive analysis about the application of the Autoregressive Integrated Moving Average (ARIMA) model in predicting cryptocurrency prices. To evaluate the model, we utilized the dataset that has been presented in C.1.1. The dataset included the Bitcoin prices spanning from September 24, 2019, to February 6, 2024. Due to the unpredictable nature of cryptocurrency markets, precise forecasting tools are extremely valuable to investors and analysts. This ARIMA model is a fundamental tool in our time series analysis. It provides a simple and logical approach to understanding and predicting future Bitcoin prices using past data. This model has been created using the following structure:

1. Data Preprocessing: The analysis starts with data acquisition, where the closing values of Bitcoin were imported from the CSV file and stored in a pandas data frame. The timestamps were converted to datetime objects and assigned as an index, ensuring the ability to efficiently perform time series operations. Initial examination of the chart shown in Figure C.1 revealed the highly unpredictable behavior of Bitcoin prices over the time period.

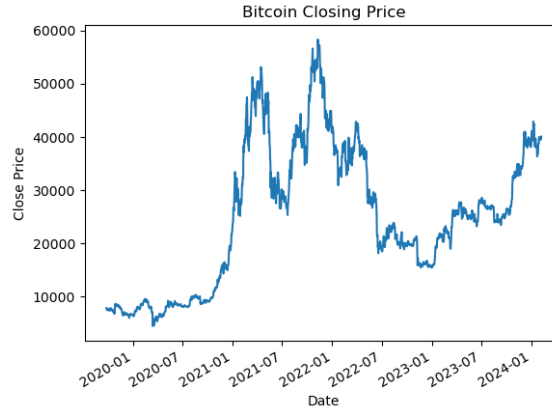


Figure C.1: *Bitcoin Closing Price*

2. Stationarity Assessment: Before fitting the ARIMA model, it is important to determine the state of the time series, as the model requires a stationary series. The Augmented Dickey-Fuller (ADF) test is used for this purpose. The initial result of the test revealed a p-value of 0.5649, exceeding the conventional threshold of 0.05. This suggests that the data is non-stationary. To address this issue, differencing was performed, resulting in a significantly reduced p-value (approximately $1.53e-23$) in the differenced data set, which confirmed its stationarity.

3. ACF and PACF Analysis: In order to clarify the sequence of the model, autocorrelation function (ACF) and partial autocorrelation function (PACF) were applied to the differenced series. Based on the analysis of the Bitcoin dataset, it can be observed from the figure C.2 that both plots do not show significant delays, indicating that a low-order ARIMA model may be appropriate.

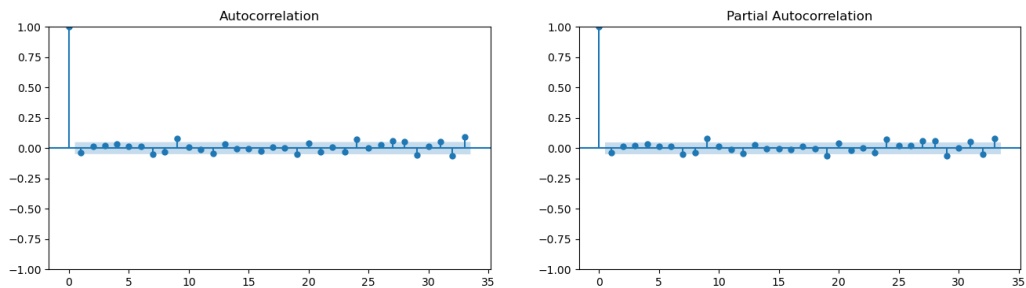


Figure C.2: *Autocorrelation and Partial Autocorrelation Function*

4. Model Training and Diagnostics: An ARIMA(1, 1, 0) model was fitted to the training dataset, which includes 80% of the total data. By choosing as parameters ($p=1$, $d=1$, $q=0$) we created a model with one autoregressive term and one difference, with no moving average components. The model summary is indicated in the table C.1. The lack of significance in the Autoregressive term (p-value: 0.204) indicates that the model does not adequately capture the process. Further diagnostics show that the model does not find all parameters statistically significant and there are problems with the distribution of residuals and heteroskedasticity. The model may be able to help us understand the patterns behind the data, however the lack of normality and stable variance in the residuals suggests that the assumptions of our model are invalid and the results must be treated with caution.

Table C.1: *Arima Diagnostics*

	coef	std err	z	$P > z $	[0.025	0.975]
ar.L1	-0.0277	0.022	-1.270	0.204	-0.070	0.015
sigma2	$1.029e + 06$	$2.06e + 04$	49.895	0.000	$9.88e + 05$	$1.07e + 06$

Ljung-Box (L1) (Q)	0.00	Jarque-Bera (JB)	1818.77
Prob(Q)	1.00	Prob(JB)	0.00
Heteroskedasticity (H)	9.42	Skew	-0.19
Prob(H) (two-sided)	0.00	Kurtosis	8.84

During this stage, four evaluation plots were also used to assess the reliability of the model as presented in Figure C.3. Below we give an explanation for each of the diagnostic plots and what they indicate about the performance of our model:

- **Standardized Residuals:** This plot shows the residuals of our model over time. Ideally, we want to avoid seeing obvious patterns (e.g., cycles or trends). This would indicate that a certain aspect of the data structure was missed by the model. In the figure plot, the residuals appear to move randomly around zero with no clear pattern, which is a good sign. However, there are some notable spikes, which could indicate occasional shortcomings of the model in capturing price movements[26].

- **Histogram and Density Plot:** This plot includes the histogram of standardized residuals, overlaid with a kernel density estimate (KDE) and a normal distribution ($N(0,1)$). The goal here would be that the KDE follows closely the curve of the normal distribution, suggesting that the residuals are normally distributed. A normally distributed residual indicates that the model errors are random and not biased. In the diagram provided, there are deviations from the normal distribution, particularly in the tails, indicating potential problems with the model, such as non-normality in the residuals[26].

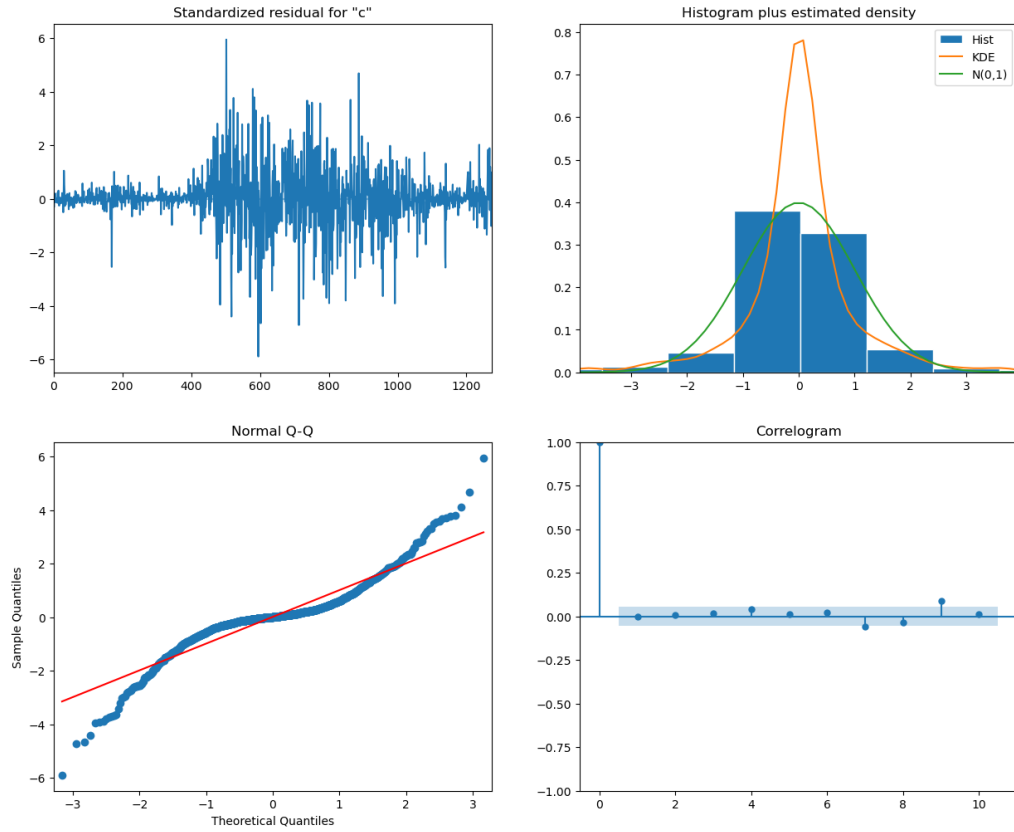


Figure C.3: Model Diagnostic Plots

- **Normal Q-Q Plot:** The Quantile-Quantile chart is used to visually check the normality of the residuals. If the residuals are normally distributed, the points should fall approximately along the red line. In the plot of our model, the points deviate from the line, particularly in the tails, suggesting that the residuals have

a distribution with heavier tails than the normal distribution. This means that our model cannot fully capture the risk of extreme price movements (clustering of volatility that is common in financial data)[27].

- **Correlogram (ACF plot):** This diagram shows the autocorrelation of the residuals at different lags. Ideally, all autocorrelations should be within the blue band, which represents a 95% confidence interval for statistical significance. If they are within the band, this indicates that there is no significant autocorrelation. In the plot shown, almost all autocorrelations are within the band, indicating that the residuals are not autocorrelated and that the model has effectively captured the autocorrelations of the time series.

5. Prediction and Evaluation: The next step was to actually use the model and produce predictions for the length of the test set. The predicted values were then compared against the actual test data. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were computed to measure the accuracy of the prediction, resulting in values of approximately 4601.57 and 7012.59, according to the Table C.2. These metrics indicate that the predictions have a significant margin of error, which could be attributed to the unpredictable nature of Bitcoin prices or inaccuracies in the model.

Table C.2: *Arima Performance Metrics*

Metric	Value
Mean Absolute Error (MAE):	4601.57
Root Mean Squared Error (RMSE):	7012.59

6. Visualization of the Prediction: The final plot presented in Figure C.4, illustrates the actual versus predicted Bitcoin prices. The visual comparison shows that the model has not captured the recent increase in Bitcoin prices. We can clearly observe that the predicted prices deviate significantly from the actual data, especially during periods of sharp price changes. This misalignment can be explained by the simplicity of the model and the natively unpredictable nature of financial markets.

The forecasted data appear as a flat line, indicating that the ARIMA(1,1,0) model predicts future values of the series to remain constant, a behavior typical in models specified with a non-seasonal MA(0) component. This suggests the model's inadequacy in capturing the

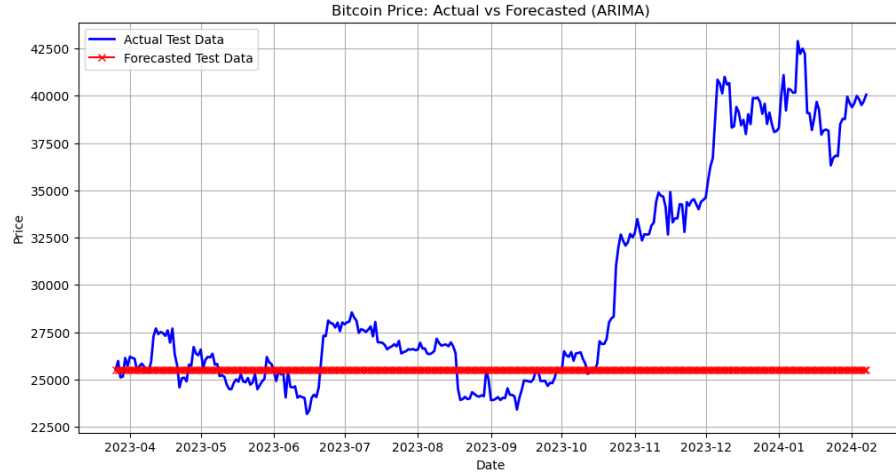


Figure C.4: *Arima Actual vs. Predicted Bitcoin Prices*

volatility and trend of Bitcoin prices due to its simplicity and reliance on only one lagged value for predictions. The lack of strong autocorrelation at lag 1 results in a forecast that is not dynamic, highlighting the model's poor fit for the dataset.

The model's oversimplified approach fails to align with the known volatility and potential non-linear patterns of Bitcoin price data. Experimenting with additional lags for the autoregressive (p) and moving average (q) components, as well as including a seasonal component, might address these issues. Financial time series like Bitcoin prices, influenced by numerous external factors, pose a challenge for models relying solely on historical data.

Given these observations, it is clear that the ARIMA(1, 1, 0) model's limitations necessitate a more advanced approach for accurately predicting Bitcoin prices. Incorporating external factors or adopting more complex time series models could enhance predictive accuracy. The cryptocurrency market's dynamic nature demands a periodic evaluation and adjustment of model parameters in order to capture the emerging trends[4, 34, 42].

C.2.1.1 ArimaX model

The ARIMAX model extends our ARIMA model by adding exogenous variables (other variables besides the time series itself) to potentially improve the forecasting accuracy. We applied the same differencing and autoregressive terms as the ARIMA model but we

included additional features such as 'volume', 'marketCap', and the 7-day moving average of the closing price. By adding these extra features in the model we provide more context to analyze other factors that may have an effect on Bitcoin price and as a result those factors will lead to a more accurate forecast. We used the same dataset to the ARIMAX model and we followed these steps in order to train the model:

1. Data Preprocessing: In order to preprocess our dataset we began with the loading of Bitcoin price data into a Data Frame. We followed a similar approach as that in the ARIMA model and we transformed the timeClose column to datetime format. The datetime was set as the Data Frame's index, followed by a sort to ensure order. We chose the 'close' price, along with 'volume' and 'marketCap' for the ARIMAX model. To align these features on a consistent scale, MinMaxScaler was used for normalization. Additionally, we created a new feature to represent the 7-day moving average of the 'close' price, providing a smoothed trend indicator. Any NaN values that showed up during the phase of the rolling mean calculation were thrown away. The dataset was then split into 80% training and 20% test sets. We chose the "close" price as the target and added the external factors "volume", "marketCap", and 7-day moving average for training.

2. Model Training and Evaluation: For the model training we used the order (1,1,1), which means that we included one autoregressive term, one differencing step, and one moving average term. The training phase did not enforce stationarity, allowing the model to adapt more freely to the data's characteristics. The training was done using "close" price data and the specified exogenous variables "volume", "marketCap", and 7-day moving average. The next step was to test the performance of our model by trying to predict the test data along with the exogenous variables that were linked to it. In table C.3 The Root Mean Squared Error (RMSE) was used to measure the accuracy of the model's predictions and was found to be approximately 508.45, indicating a high level of precision. Furthermore, the R^2 score of 0.992 suggests that the model explains 99.2% of the variance in the target variable, demonstrating an excellent fit to the data.

Table C.3: *ArimaX Performance Metrics*

Metric	Value
Root Mean Squared Error (RMSE):	508.45
R^2 :	0.9920

3. Visualization of the Prediction: Figure C.5 showcases the performance of the ARIMAX model. The Bitcoin prices are visually distinguished by a blue color, while the predicted prices are indicated by red 'x' marks and dashed lines. It is clear that the model did a good job on following the ups and downs in Bitcoin's value. The close match between the predicted and real prices indicates that the model worked well. When the predicted line is close to the actual line, it means that the predictions are accurate, showing that the ARIMAX model understood the historical patterns well.

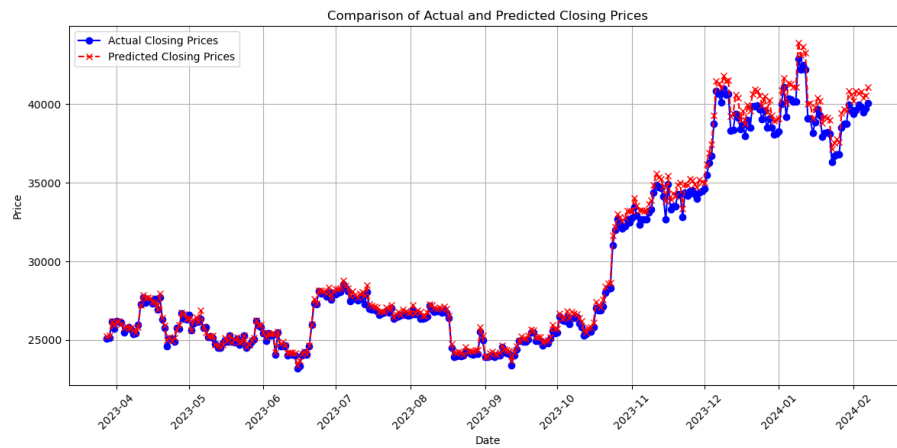


Figure C.5: *ArimaX Actual vs. Predicted Bitcoin Prices*

The model's accuracy was also hinted by the RMSE value, which we discussed on model training and diagnostics. RMSE is an important measuring that indicates how good the predictions of a model are. An RMSE value of about 508.45, given Bitcoin's price changes and unpredictability, means the model made some errors in predicting but still did reasonably well considering how complex financial markets can be. However, without comparing it to other models or standards, we can't say for sure if it's the best option for predicting the cryptocurrency prices. Any differences between the predicted and real prices would be key areas to work on. It is also important to bear in mind that the model is not perfect, as indicated by the RMSE.

C.2.2 Implementation of SVM

The following model represents a Support Vector Machine (SVM) model for predicting Bitcoin prices. The model was trained using the same historical data as the ARIMA and ARIMAX models (from September 24, 2019, to February 6, 2024). The data included features that are widely recognized as having a significant impact on price fluctuations in financial markets so preprocessing of the original dataset was done by applying the following steps:

1. Data preprocessing: Based on the foundation laid by the ARIMA and ARIMAX preprocessing steps, the preparation for the SVM model similarly begins with the import of Bitcoin pricing data into a pandas DataFrame. This process extends to include a wider range of characteristics ("open", "high", "low" and "volume") alongside the "close" price which is the predict target. The carefully chosen characteristics aim to take advantage of SVM's ability to recognize complicated patterns, with an emphasis on the relations between different market indicators. To maintain chronological accuracy, the conversion of the timestamp to datetime format is performed and this timestamp is being set as the index. Using the same practices from previous models, the dataset is divided into an 80% training set and a 20% testing set. This approach facilitates direct comparison of model performance under uniform conditions, allowing for unbiased evaluation of their predictive strengths and weaknesses. The next step is to perform feature scaling. Feature scaling is particularly important for SVMs, because this algorithm is sensitive to the scale of the input attributes. By scaling the features to a similar range, such as $[0, 1]$ or $[-1, 1]$, we ensure that each feature contributes accordingly to the distance calculation, resulting in a more balanced and well-generalized model. We used the StandardScaler function to normalize our features, making certain that each feature contributes equally to the predictions made by the model. Given that we're trying to predict bitcoin prices, which may vary greatly in scale, this normalization is crucial to prevent any feature from negatively impacting the model's performance.

2. Model Training and Diagnostics: Following that, the SVM model was configured. Four distinct configurations must be implemented:

- 1. Kernel:** The kernel function is a fundamental component of an SVM that transforms the input data into a higher-dimensional space where a hyperplane can more easily separate different classes. This transformation is crucial for dealing with non-linear

relationships between features and the target variable[40]. Common types of kernels include:

- **Linear:** No transformation, suitable for linearly separable data.
 - **Polynomial:** Transforms data by considering polynomial combinations of features, controlled by the degree of the polynomial.
 - **Radial Basis Function (RBF):** Uses the distance between points in a multi-dimensional space to determine similarity, effectively creating complex boundaries.
 - **Sigmoid:** Mimics the logistic function, transforming data in a sigmoidal manner.
2. **Regularization (C):** Regularization parameter (denoted as C) controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights, which helps to prevent the model from overfitting. A smaller value of C encourages a larger margin (allowing more misclassifications) for better generalization, while a larger C value aims for a stricter margin that tries to correctly classify all training examples, which can lead to overfitting[15].
 3. **Gamma:** The gamma parameter is specific to certain kernels like the RBF. It controls the influence of individual training examples on the decision boundary. In simple terms, gamma defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. A high gamma value leads to more complexity by allowing the model to capture more information about each specific data point, potentially leading to overfitting. In contrast, a low gamma encourages the model to consider points at a greater distance, promoting a smoother decision boundary.[25]
 4. **Epsilon:** Epsilon is a parameter used primarily in SVM models for regression (SVR), controlling the width of the epsilon-insensitive zone, used to fit the training data. The idea is to find a function that deviates from the actual observed targets (y values) by a value no more than epsilon for each training point. It helps in determining the level of tolerance for errors in the model's predictions, where a larger epsilon means we're allowing more errors within a certain margin, and a smaller epsilon tightens the criteria for what is considered an error.[43]

The Radial Basis Function (RBF) was utilized for the Kernel, as it is well-suited for non-linear data. A value of 100 was employed for the normalisation parameter (C) in order to successfully achieve a balance between minimizing the size of the weights and ensuring low error in the training set. Gamma and Epsilon were both set to 0.1 to determine the impact of an individual training example and to determine the epsilon-tube, which represents the range of predictions that avoid any penalty. Optimizing the performance of an SVM model relies heavily on tuning these parameters. This process involves finding the right balance between the model's complexity, which allows it to capture underlying patterns, and its ability to work well with new data that it hasn't seen before.

3. Prediction and Evaluation: The trained model is used to predict the closing prices on the scaled test data. The performance of the model is evaluated using two metrics: RMSE and R-squared (R^2). By reviewing table C.4, the results are as follows:

- **RMSE:** The value of 793.323 suggests that the predictions of the model are approximately 793.323 units away from the actual data points. Since Bitcoin prices can change a lot going up and down by thousands of dollars, an RMSE of this level could work fine in some situations. One such situation is that of our model.
- **The R^2 score:** It shows how close the regression predictions are to the actual data points. A value of 0.9807 for R^2 is very close to 1, meaning that the model can explain about 98.07% of the changes in that same variable. This level of accuracy is considered very high, meaning that the model's predictions are very close to the true values.

Based on the high R^2 score, it looks like the model can explain a good amount of the price changes in Bitcoin. However, the RMSE shows that the wrong predictions can be very different from the actual prices.

Table C.4: SVM Model Performance Metrics

Metric	Value
Root Mean Squared Error (RMSE)	793.32
R^2 Score	0.9806

4. Visualization of Prediction: In Figure C.6, the Bitcoin values are displayed with blue dots, which serve as a baseline for evaluating the accuracy of the forecast model. The predictions made by our model are denoted by red "x" symbols.

The proximity of the red "x" symbols to the blue dots provides a clear indication of the accuracy of the model in predicting Bitcoin price movements. The instances where the predicted prices are found close to the actual prices indicate the model's ability to understand market trends. On the other hand, the deviations between the red "x" symbols and the blue dots show occasions where the model's predictions differ from the recorded values, signaling potential areas for further improvement of the model's accuracy.

This visual analysis highlights the potential of the prediction model, but also helps to understand the dynamic and often unpredictable nature of Bitcoin's market value. Despite the occasional mismatches, the overall trend captured by the model's predictions reflects an important understanding of market directions, providing valuable insights for market participants who depend on accurate predictions to navigate the cryptocurrency investment landscape.

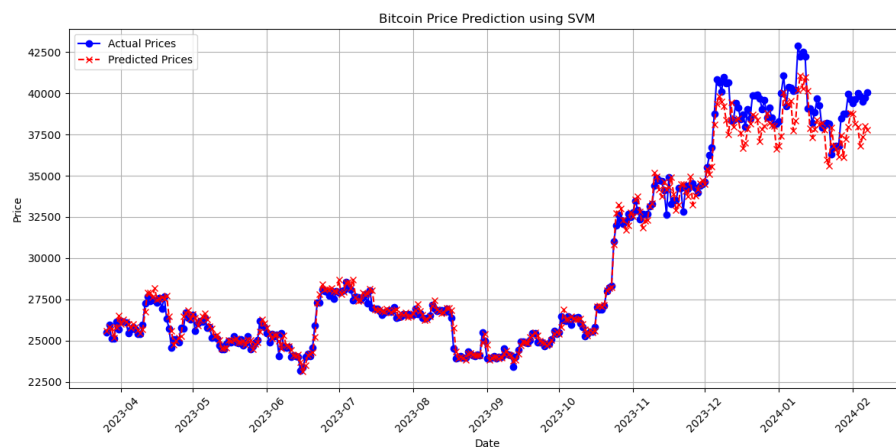


Figure C.6: *SVM Actual vs. Forecasted Bitcoin Prices*

It's important to note though that past performance is not always indicative of future results, especially in cryptocurrency markets known for their volatility and unpredictability. This model would need constant reevaluation and updating with new data to maintain its accuracy.

C.2.3 Implementation of LSTM

When it comes to economic forecasting such as cryptocurrency price prediction, the creation of a robust predictive model is very important. The long short-term memory (LSTM) neural network is considered to be one of the best architecture for this job because it can remember things for a long time. The model we will present has a carefully designed structure that aims to make it better at making predictions while reducing the risk of overfitting, which is a common mistake in machine learning. The procedure for creating the model is presented below:

1. Data preprocessing: As we already did in the previous models we loaded the dataset containing the Bitcoin values into a Pandas DataFrame. The attributes that we selected for the LSTM model were "Open", "High", "Low", "Close" and "Volume", and we also selected the timestamps to indicate the date and time of each record accordingly. To preprocess the data, these features were normalized to a range between 0 and 1 using the MinMaxScaler function in order to train the model more efficiently. A function was also defined to transform the data into sequences, with a selected length of 60 time steps per sequence, which makes the prediction of the closure value at the next time step. The preprocessed data were then divided into training and test sets of 80-20%. We used the same train-test split as with the previous models so this way we are able to compare them at the end.

2. Model Training: The next step is the training of the model. The chosen model features a design of a layered sequential model, aiming to enhance its prediction accuracy while reducing the risk of overfitting, a common problem that is faced in machine learning. The first layer of this model is an LSTM layer made up of 50 units that were uniquely set up to return sequences. This made it easy to combine with another LSTM layer that also had 50 units and helped to improve the model's ability to understand how data changes over time. A dropout layer with a dropout rate of 0.2 was added after each LSTM layer to lower the risk of overfitting. At the end of this sequential model was a Dense output layer with a single unit that was designed to predict the next close price. In order to build this complex model, we chose the Adam optimizer as it works quickly and well. It was combined with the mean square error loss function to measure how accurate the predictions are. For the training of the model we used the preprocessed training data for 100 epochs (times), with a batch size of 32.

3. Prediction and Evaluation: When the training of the LSTM model was complete, we used it to make price predictions on the test data. These predictions were then inverse transformed to the original price scale in order to make them understandable. Table C.5 provides the model’s performance that was evaluated using two metrics:

- **RMSE:** The model achieved an RMSE of 943.1766, indicating that the model’s predictions deviate from the actual prices by an average of approximately 943 units away. This level of error provides an aggregate value that indicates how accurate the prediction is.
- **R^2 Score:** The R^2 score was 0.9732, which suggests that the model explains about 97.32% of the variance in the actual Bitcoin prices. This high score indicates a strong predictive power and a good fit to the data.

Table C.5: *LSTM Model Performance Metrics*

Metric	Value
Root Mean Squared Error (MSE)	943.1766
R^2 Score	0.9732

4. Visualization of Prediction: Figure C.7 shows how the actual and predicted prices of Bitcoin have changed over time. Similarly to the previous visualizations, the Bitcoin values are displayed with blue dots, which serve as a baseline for evaluating the accuracy of the forecast model. The predictions made by our model are denoted by red "x" symbols. This visualization showcases that the model can closely monitor how Bitcoin prices are changing, catching the overall trend and responding to major changes. However, there are clear differences at certain points, which correspond to the RMSE value and show that the model could be improved.

In conclusion, the R^2 score demonstrates the high efficiency of our LSTM model in accurately predicting Bitcoin prices. The RMSE shows that some predictions were wrong, which is normal in the volatile cryptocurrency market. The model effectively captures the general pattern, which is important in predicting time series in financial markets. For enhanced accuracy in future predictions, we could employ techniques such as tuning the hyperparameters, performing feature engineering, or incorporating additional data sources.

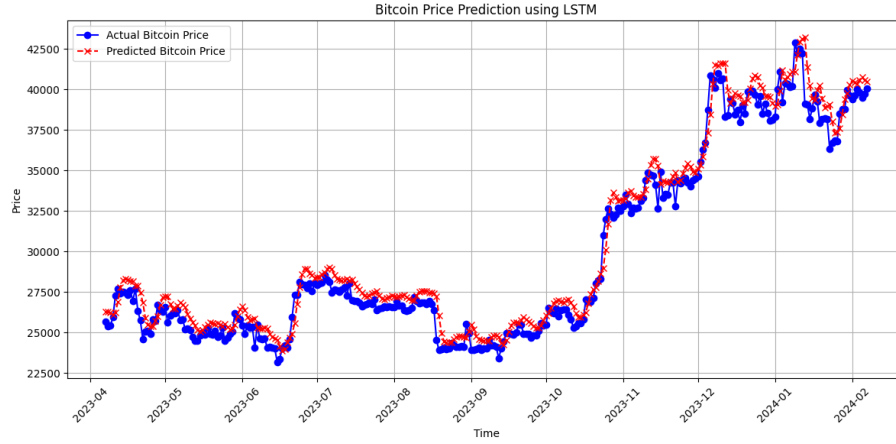


Figure C.7: *LSTM Actual vs. Forecasted Bitcoin Prices*

C.3 Comparison of the models

This section looks into the comparative analysis of the three advanced models; ARIMAX, SVM and LSTM, focusing on their performance in predicting the Bitcoin prices contained in the dataset. The evaluation is based on two primary metrics: root mean square error (RMSE) and coefficient of determination (R^2 score), which along with the plots provide information on accuracy, error rates and the ability of the models to capture the variance of actual prices.

ArimaX model: The ARIMAX model stands out with an RMSE of 508.45, the lowest among the three models. Moreover, the impressive R^2 score of 0.9920 suggests that the ARIMAX model can explain 99.20% of the variance observed in the actual prices. Such a high level of predictive accuracy and variance explanation positions the ARIMAX model as a potentially reliable tool for forecasting Bitcoin prices.

SVM model: The SVM model, with an RMSE of 793.32, demonstrates a less accurate performance compared to the ARIMAX model. This higher RMSE suggests that the predictions made by the SVM model tend to deviate more from the actual prices. In addition, its R^2 score of 0.9806 is impressive, explaining 98.06% of the price variance. Although it does not match the ARIMAX model's accuracy, the SVM model's ability to account for a significant portion of the variance makes it a strong contender.

LSTM model: The LSTM model displays the highest RMSE of 943.17 among the three

models, indicating that its predictions are the furthest from the actual Bitcoin prices. Together with the lowest R^2 score of 0.9732, it is evident that the LSTM model, while still capturing a considerable 97.32% of the variance, trails behind the ARIMAX and SVM models in predictive accuracy. This gap highlights the LSTM model's challenges in closely mirroring the actual price movements, particularly during periods of high volatility.

A closer examination of the predictive performance through the visual graphs reveals some little differences among the models (Figure C.8):

- **ArimaX:** Shows an impressive match between predicted and actual prices, which indicates how accurate the model is.
- **SVM:** Although capturing the general trend, it shows deviations, especially in volatile market phases.
- **LSTM:** Mirrors the SVM's performance in capturing the trend but with noticeable deviations, underlining the model's difficulty with big changes in prices happen.

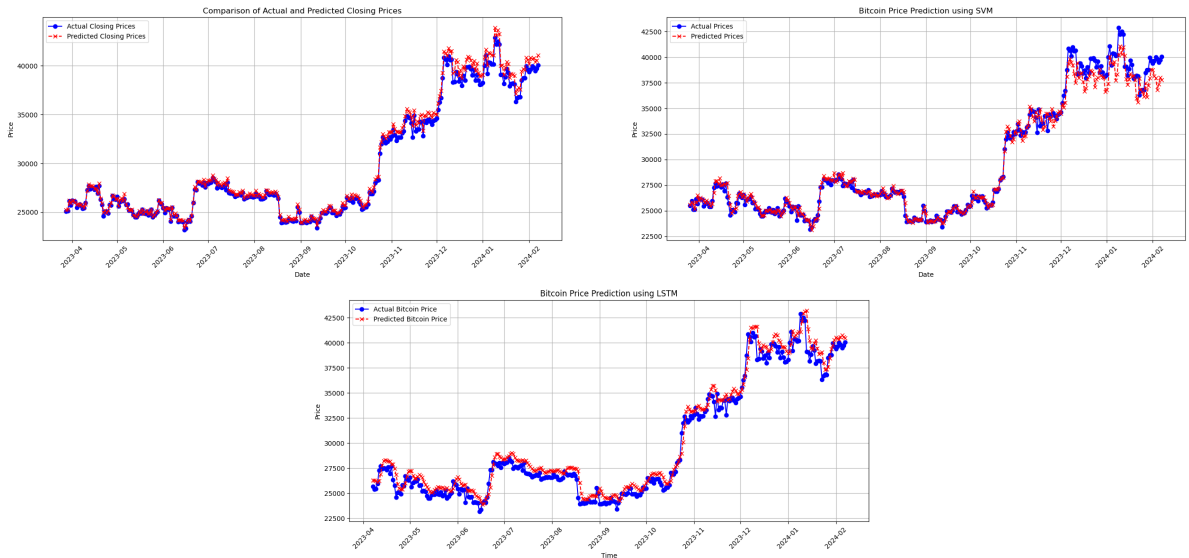


Figure C.8: Comparison of Model Performance: ARIMAX, SVM, and LSTM Actual vs. Forecasted Bitcoin Prices

Table C.6 provides a summary of all model performance metrics. The comparative analysis highlights that the ARIMAX model does a better job in predicting Bitcoin prices, demonstrated by the lowest RMSE and highest R^2 score. It showed an unmatched precision and an ability to cover nearly all variances of the data.

Model	RMSE	R^2 Score	Observations
ARIMA	7012.59	-	High error, indicates challenges in capturing price movements
ARIMAX	508.45	0.992	Best performance, accurately predicts price movements
SVM	793.32	0.9806	Good performance, but less accurate than ARIMAX
LSTM	943.17	0.9732	Captures trends well, but with higher error than others

Table C.6: *Summary of Model Performance Metrics*

The SVM model, while less accurate than ARIMAX, still presents a possible option with a strong variance explanation capability. On the other hand, the LSTM model, despite its higher error rates and lower variance capture, could still be preferable in scenarios demanding the identification of complex patterns or real-time predictions, highlighting the significance of selecting a contextual model based on more than just performance metrics.

Chapter D

Conclusions and future research

This thesis aimed to investigate the effectiveness of advanced deep learning algorithms in predicting the price dynamics of cryptocurrencies, with a focus on Bitcoin. The goal was to identify the most suitable model for the Bitcoin Historical Price time-series dataset. We used a four-year span of data to train and test the models. This period captures a critical phase in the cryptocurrency market.

Through a careful examination of the three different models -ARIMA, SVM and LSTM- this study is attempting to evaluate their quality and ability to accurately predict future Bitcoin prices. The comparative analysis was based on two primary metrics: root mean square error (RMSE) and coefficient of determination (R^2 score), followed by visual graphs to highlight the accuracy, error rates and variance capturing capabilities of the models.

The research highlights the importance of selecting an appropriate forecasting model based on specific needs and conditions. The proven ability of the ARIMAX model to make better predictions on our Bitcoin dataset makes it an excellent choice for scenarios that require high accuracy and precision. On the other hand, the SVM model, with its impressive ability to explain variance, additionally provides a good choice for applications where a balance between accuracy and computational efficiency is required. Lastly, the LSTM model, despite its higher error rates, could be preferred in situations that require the identification of complex patterns over time or real-time prediction capabilities.

This thesis is a contribution to the existing literature by providing a comparative analysis of deep learning methods applied to cryptocurrency price prediction. It highlights the strengths and limitations of each model, thus providing valuable information for re-

searchers and analysts in the field of cryptocurrency trading and investment.

Future research could focus on developing hybrid models that combine the advantages of ARIMAX, SVM and LSTM to further enhance the accuracy and efficiency of prediction. Furthermore, expanding the data set to include more cryptocurrencies and longer time frames could provide deeper insight into the performance of the models in different market conditions. Investigating the impact of external factors, such as regulatory changes or major market events could also provide further improvements in forecast accuracy.

Chapter E

Appendix: Python code of the Prediction Models

Arima Model

```
1     import pandas as pd
2     import matplotlib.pyplot as plt
3     import numpy as np
4     from statsmodels.tsa.stattools import adfuller
5     from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
6     from statsmodels.tsa.arima.model import ARIMA
7     from sklearn.metrics import mean_absolute_error, ←
        mean_squared_error
8
9     # Load the dataset
10    df = pd.read_csv('dataset/BitcoinData.csv', sep=';')
11
12    # Convert the timestamp column to datetime
13    df['timestamp'] = pd.to_datetime(df['timestamp'])
14
15    # Set the 'timestamp' as the index
16    df.set_index('timestamp', inplace=True)
17
18    # Focus on the closing price
```

```

19     df = df['close']
20
21     # Print the first few rows
22     print(df.head())
23
24     # The plot of the dataset
25     df.plot(title='Bitcoin Closing Price')
26     plt.xlabel('Date')
27     plt.ylabel('Close Price')
28     plt.show()
29
30     # ADF statistic and p-value
31     result = adfuller(df.dropna())
32     print(f'ADF Statistic: {result[0]}')
33     print(f'p-value: {result[1]}')
34     for key, value in result[4].items():
35         print(f'Critical Value {key}: {value}')
36
37     # Differencing the series in order to make it stationary
38     # and check if the differenced series is stationary
39     df_diff = df.diff().dropna()
40     result_diff = adfuller(df_diff)
41     print(f'ADF Statistic (Differenced Series): {result_diff[0]}')
42     print(f'p-value (Differenced Series): {result_diff[1]}')
43
44     # Plot the ACF and PACF on the differenced series
45     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 4))
46     plot_acf(df_diff, ax=ax1)
47     plot_pacf(df_diff, ax=ax2)
48     plt.show()
49
50     # Reverse the dataset because it's currently in descending order
51     df = df.iloc[::-1]
52
53     # Split the data into training and test 80–20%
54     split_idx = int(len(df) * 0.8)
55     train = df.iloc[:split_idx]
56     test = df.iloc[split_idx:]

```

```

57
58     # Fit the ARIMA model on the training data
59     model = ARIMA(train, order=(1, 1, 0))
60     fitted_model = model.fit()
61
62     # Summary of the fitted model on the training set
63     print(fitted_model.summary())
64
65     # Diagnostics
66     fitted_model.plot_diagnostics(figsize=(15, 12))
67     plt.show()
68
69     # Generate the forecast for the length of the test set
70     forecast = fitted_model.get_forecast(steps=len(test))
71     forecast_mean = forecast.predicted_mean
72     forecast_series = pd.Series(forecast_mean.values, index=test.index)
73
74     # Calculate error metrics
75     mae = mean_absolute_error(test, forecast_series)
76     rmse = np.sqrt(mean_squared_error(test, forecast_series))
77     print(f'Mean Absolute Error (MAE): {mae}')
78     print(f'Root Mean Squared Error (RMSE): {rmse}')
79
80     # Plot of Actual vs Forecasted values
81     plt.figure(figsize=(15, 7))
82     plt.plot(test, label='Actual Test Data', color='blue')
83     plt.plot(forecast_series, label='Forecasted Test Data', color='red', linestyle='—')
84     plt.title('Bitcoin Price: Actual vs Forecasted')
85     plt.xlabel('Date')
86     plt.ylabel('Price')
87     plt.legend()
88     plt.show()

```

ArimaX Model

```
1     import pandas as pd
2     from statsmodels.tsa.statespace.sarimax import SARIMAX
3     from sklearn.metrics import mean_squared_error, r2_score
4     import numpy as np
5     from sklearn.preprocessing import MinMaxScaler
6     import matplotlib.pyplot as plt
7
8     # Load the Bitcoin price data
9     bitcoin_data_file = 'dataset/BitcoinData.csv'
10    bitcoin_prices_data = pd.read_csv(bitcoin_data_file, delimiter=';',↵
        ')
11
12    # Change 'timeClose' to datetime format and use it as the index
13    bitcoin_prices_data['date'] = pd.to_datetime(bitcoin_prices_data[↵
        'timeClose'])
14    bitcoin_prices_data.set_index('date', inplace=True)
15    bitcoin_prices_data.sort_index(inplace=True) # Make sure the ↵
        dates are in order
16
17    # Set data frequency to daily
18    bitcoin_prices_data = bitcoin_prices_data.asfreq('D')
19
20    # Choose 'close' price and other features for the model
21    selected_features = ['volume', 'marketCap']
22    close_prices = bitcoin_prices_data['close']
23    other_features = bitcoin_prices_data[selected_features]
24
25    # Scale the features to a range
26    feature_scaler = MinMaxScaler()
27    scaled_features = feature_scaler.fit_transform(other_features)
28    scaled_features_df = pd.DataFrame(scaled_features, columns=↵
        selected_features, index=bitcoin_prices_data.index)
29
30    # Combine the scaled features with the closing prices
31    combined_data = close_prices.to_frame().join(scaled_features_df)
32
33    # Add a 7-day moving average of the close prices to the dataset
```

```

34 combined_data['7-day-moving-average'] = combined_data['close'].↵
    rolling(window=7).mean()
35 combined_data.dropna(inplace=True) # Remove rows with missing ↵
    values due to moving average calculation
36
37 # Divide the dataset into training and testing parts
38 train_test_split_ratio = 0.8
39 split_point = int(len(combined_data) * train_test_split_ratio)
40 training_data, testing_data = combined_data.iloc[:split_point, ↵
    :], combined_data.iloc[split_point:, :]
41
42 # Prepare the target and features for the model
43 training_target = training_data['close']
44 updated_features = selected_features + ['7-day-moving-average']
45 training_features = training_data[updated_features]
46
47 # Create and train the ARIMAX model
48 model = SARIMAX(training_target, exog=training_features, order↵
    =(1,1,1), enforce_stationarity=False, enforce_invertibility=↵
    False)
49 model_fit = model.fit(dispatch=False)
50
51 # Predict future values
52 testing_features = testing_data[updated_features]
53 predictions = model_fit.get_forecast(steps=len(testing_data), ↵
    exog=testing_features)
54 predicted_values = predictions.predicted_mean
55
56 # Model performance evaluation
57 root_mean_squared_error = np.sqrt(mean_squared_error(testing_data↵
    ['close'], predicted_values))
58 r_squared_value = r2_score(testing_data['close'], ↵
    predicted_values)
59 print(f'Root Mean Squared Error: {root_mean_squared_error}')
60 print(f'R-squared Value: {r_squared_value}')
61
62 # Visualization of actual vs predicted closing prices
63 plt.figure(figsize=(12, 6))

```

```

64     plt.plot(testing_data.index, testing_data['close'], label='Actual↵
        Closing Prices', color='blue', marker='o')
65     plt.plot(testing_data.index, predicted_values, label='Predicted ↵
        Closing Prices', color='red', linestyle='—', marker='x')
66     plt.title('Comparison of Actual and Predicted Closing Prices')
67     plt.xlabel('Date')
68     plt.ylabel('Price')
69     plt.legend()
70     plt.grid(True)
71     plt.xticks(rotation=45)
72     plt.tight_layout()
73     plt.show()

```

SVM Model

```

1     import pandas as pd
2     import numpy as np
3     from sklearn.model_selection import train_test_split
4     from sklearn.preprocessing import StandardScaler
5     from sklearn.svm import SVR
6     from sklearn.metrics import mean_squared_error, r2_score
7     import matplotlib.pyplot as plt
8
9     # Load the Bitcoin price data
10    bitcoin_prices = pd.read_csv('dataset/BitcoinData.csv', delimiter↵
        =';')
11
12    # Convert the 'timestamp' column to datetime format and use it as↵
        the dataframe index
13    bitcoin_prices['timestamp'] = pd.to_datetime(bitcoin_prices['↵
        timestamp'])
14    bitcoin_prices.set_index('timestamp', inplace=True)
15
16    # Ensure the data is ordered by date
17    bitcoin_prices.sort_index(inplace=True)
18

```

```

19     # Define features and the target variable for the prediction ↵
        model
20     features = bitcoin_prices[['open', 'high', 'low', 'volume']] # ↵
        Predictor variables
21     target = bitcoin_prices['close'] # Prediction target
22
23     # Split the data into training and testing sets with an 80/20 ↵
        split
24     train_size = int(len(features) * 0.8)
25     features_train, features_test = features.iloc[:train_size], ↵
        features.iloc[train_size:]
26     target_train, target_test = target.iloc[:train_size], target.iloc↵
        [train_size:]
27
28     # Normalize the feature data
29     feature_normalizer = StandardScaler()
30     features_train_normalized = feature_normalizer.fit_transform(↵
        features_train)
31     features_test_normalized = feature_normalizer.transform(↵
        features_test)
32
33     # Create and train the Support Vector Regression model
34     svr_model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=0.1)
35     svr_model.fit(features_train_normalized, target_train)
36
37     # Predict target values using the trained model
38     predictions = svr_model.predict(features_test_normalized)
39
40     # Calculate performance metrics of the model
41     mean_squared_err = mean_squared_error(target_test, predictions)
42     root_mean_squared_err = np.sqrt(mean_squared_err)
43     r_squared_value = r2_score(target_test, predictions)
44
45     print(f'Root Mean Squared Error: {root_mean_squared_err}')
46     print(f'R^2 Score: {r_squared_value}')
47
48     # Visualize actual vs. predicted Bitcoin prices
49     plt.figure(figsize=(12, 6))

```

```

50     plt.plot(target_test.index, target_test, label='Actual Prices', ↵
        color='blue', marker='o')
51     plt.plot(target_test.index, predictions, label='Predicted Prices'↵
        , color='red', linestyle='—', marker='x')
52     plt.title('Bitcoin Price Prediction using SVM')
53     plt.xlabel('Date')
54     plt.ylabel('Price')
55     plt.legend()
56     plt.grid(True)
57     plt.xticks(rotation=45)
58     plt.tight_layout()
59     plt.show()

```

LSTM Model

```

1     import pandas as pd
2     import matplotlib.pyplot as plt
3     import numpy as np
4     from math import sqrt
5     from sklearn.metrics import mean_squared_error, r2_score
6     from sklearn.preprocessing import MinMaxScaler
7     from tensorflow.keras.models import Sequential
8     from tensorflow.keras.layers import Dense, LSTM, Dropout
9
10    # Load Bitcoin price data
11    bitcoin_data = pd.read_csv('dataset/BitcoinData.csv', delimiter='↵
        ;')
12
13    # Change the 'timestamp' column to datetime type
14    bitcoin_data['timestamp'] = pd.to_datetime(bitcoin_data['↵
        timestamp'])
15
16    # Ensure data is ordered by date
17    bitcoin_data = bitcoin_data.sort_values(by='timestamp', ascending↵
        =True)
18

```

```

19 # Define the set of features for the model
20 selected_features = bitcoin_data[['timestamp', 'open', 'high', 'low', 'close', 'volume']]
21
22 # Prepare data for normalization, excluding 'timestamp'
23 data_for_scaling = selected_features[['open', 'high', 'low', 'close', 'volume']].values
24
25 # Normalize the features
26 scaler = MinMaxScaler(feature_range=(0, 1))
27 normalized_data = scaler.fit_transform(data_for_scaling)
28
29 # Combine normalized data with timestamps
30 normalized_df = pd.concat([selected_features['timestamp'].reset_index(drop=True), pd.DataFrame(normalized_data, columns=['open', 'high', 'low', 'close', 'volume'])], axis=1)
31
32 # Function to create sequences for LSTM input
33 def create_sequences(data, sequence_length=60):
34     sequences, targets, sequence_dates = [], [], []
35     for i in range(len(data) - sequence_length):
36         sequence = data.iloc[i:i + sequence_length, 1:].values # Exclude 'timestamp'
37         target = data.iloc[i + sequence_length, -2] # Target is 'close' price
38         sequence_date = data.iloc[i + sequence_length, 0] # Date for the sequence
39         sequences.append(sequence)
40         targets.append(target)
41         sequence_dates.append(sequence_date)
42     return np.array(sequences), np.array(targets), np.array(sequence_dates)
43
44 # Generate data sequences
45 sequence_length = 60
46 X, y, test_dates = create_sequences(normalized_df, sequence_length)
47

```

```

48 # Split data into training and test sets based on time sequence
49 test_size = int(0.2 * len(X))
50 X_train, X_test = X[:-test_size], X[-test_size:]
51 y_train, y_test = y[:-test_size], y[-test_size:]
52 dates_test = test_dates[-test_size:]
53
54 # Define LSTM model
55 model = Sequential([
56     LSTM(units=50, return_sequences=True, input_shape=(X_train.shape←
57         [1], X_train.shape[2])),
58     Dropout(0.2),
59     LSTM(units=50),
60     Dropout(0.2),
61     Dense(units=1) # Output layer for predicting the next 'close' ←
62         price
63 ])
64
65 # Compile model
66 model.compile(optimizer='adam', loss='mean_squared_error')
67
68 # Train model
69 model.fit(X_train, y_train, epochs=100, batch_size=32)
70
71 # Predict prices
72 predicted_prices = model.predict(X_test)
73
74 # Inverse transform predictions to original price scale
75 dummy_predicted = np.zeros((len(predicted_prices), ←
76     normalized_data.shape[1]))
77 dummy_predicted[:, -2] = predicted_prices[:, 0] # Place ←
78     predictions in the correct column
79 predicted_prices_original = scaler.inverse_transform(←
80     dummy_predicted)[:, -2]
81
82 # Inverse transform actual prices to original scale
83 dummy_actual = np.zeros((len(y_test), normalized_data.shape[1]))
84 dummy_actual[:, -2] = y_test

```

```

80     actual_prices_original = scaler.inverse_transform(dummy_actual)↵
      [:, -2]
81
82     # Calculate performance metrics
83     rmse = sqrt(mean_squared_error(actual_prices_original, ↵
      predicted_prices_original))
84     r2 = r2_score(actual_prices_original, predicted_prices_original)
85
86     print(f'Root Mean Squared Error: {rmse}')
87     print(f'R^2 Score: {r2}')
88
89     # Visualize actual vs. predicted prices
90     plt.figure(figsize=(12, 6))
91     plt.plot(dates_test, actual_prices_original, color='blue', label=↵
      'Actual Bitcoin Price', marker='o')
92     plt.plot(dates_test, predicted_prices_original, color='red', ↵
      label='Predicted Bitcoin Price', linestyle='—', marker='x')
93     plt.title('Bitcoin Price Prediction using LSTM')
94     plt.xlabel('Time')
95     plt.ylabel('Price')
96     plt.legend()
97     plt.grid(True)
98     plt.xticks(rotation=45)
99     plt.tight_layout()
100    plt.show()

```

Bibliography

- [1] Bitinfocharts. Accessed on December, 2023. URL: <https://bitinfocharts.com/>.
- [2] Ethereum average block time. Accessed: [1/12/2023]. URL: https://ycharts.com/indicators/ethereum_average_block_time#:~:text=Basic%20Info,0.08%25%20from%20one%20year%20ago.
- [3] CoinMarketCap: Cryptocurrency Market Capitalizations. <https://coinmarketcap.com/>, 2024. Accessed: February 6, 2024.
- [4] Saad Alahmari. Using machine learning arima to predict the price of cryptocurrencies. 07 2019.
- [5] Maher Alharby and Aad Van Moorsel. Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*, 2017.
- [6] A.M. Antonopoulos and G. Wood. *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly, 2018. URL: <https://books.google.gr/books?id=SedSMQAACAAJ>.
- [7] Frederik Armknecht, Ghassan O Karame, Avikarsha Mandal, Franck Youssef, and Erik Zenner. Ripple: Overview and outlook. In *Trust and Trustworthy Computing: 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings 8*, pages 163–180. Springer, 2015.
- [8] Anton I Badev and Matthew Chen. Bitcoin: Technical background and data analysis. 2014.
- [9] Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. Bitcoin: Economics, technology, and governance. *Journal of economic Perspectives*, 29(2):213–238, 2015.

- [10] Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. Bitcoin: Economics, technology, and governance. *Journal of Economic Perspectives*, 29(2):213–38, May 2015. URL: <https://www.aeaweb.org/articles?id=10.1257/jep.29.2.213>, doi:10.1257/jep.29.2.213.
- [11] Brad Chase and Ethan MacBrough. Analysis of the xrp ledger consensus protocol. *arXiv preprint arXiv:1802.07242*, 2018.
- [12] Eng-Tuck Cheah and John Fry. Speculative bubbles in bitcoin markets. an empirical investigation into the fundamental value of bitcoin. *Economics letters*, 130:32–36, 2015.
- [13] Pavel Ciaian, Miroslava Rajcaniova, and d’Artis Kancs. The economics of bitcoin price formation. *Applied economics*, 48(19):1799–1815, 2016.
- [14] Shaen Corbet, Brian Lucey, Andrew Urquhart, and Larisa Yarovaya. Cryptocurrencies as a financial asset: A systematic analysis. *International Review of Financial Analysis*, 62:182–199, 2019. URL: <https://www.sciencedirect.com/science/article/pii/S1057521918305271>, doi:10.1016/j.irfa.2018.09.003.
- [15] Corinna Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.
- [16] Vasily Derbentsev, Andriy Matviychuk, and Vladimir N. Soloviev. *Forecasting of Cryptocurrency Prices Using Machine Learning*, pages 211–231. Springer Singapore, Singapore, 2020. doi:10.1007/978-981-15-4498-9_12.
- [17] Marcin Fałdziński, Piotr Fiszeder, and Witold Orzeszko. Forecasting volatility of energy commodities: Comparison of garch models with support vector regression. *Energies*, 14(1), 2021. URL: <https://www.mdpi.com/1996-1073/14/1/6>, doi:10.3390/en14010006.
- [18] Python Software Foundation. Python: About documentation. <https://www.python.org/doc/essays/blurb/>, Retrieved 2024.
- [19] Philip Hans Franses and Dick Van Dijk. Forecasting stock market volatility using (non-linear) garch models. *Journal of forecasting*, 15(3):229–235, 1996.

- [20] Arpita Ghosh, Mohammad Mahdian, Daniel M Reeves, David M Pennock, and Ryan Fugger. Mechanism design on trust networks. In *International Workshop on Web and Internet Economics*, pages 257–268. Springer, 2007.
- [21] Giancarlo Giudici, Alistair Milne, and Dmitri Vinogradov. Cryptocurrencies: market analysis and perspectives. *Journal of Industrial and Business Economics*, 47(1):1–18, 03 2020. doi:[10.1007/s40812-019-00138-6](https://doi.org/10.1007/s40812-019-00138-6).
- [22] Everett Hildenbrandt, Manasvi Saxena, Xiaoran Zhu, Nishant Rodrigues, Philip Darian, Dwight Guth, and Grigore Roşu. Kevm: A complete semantics of the ethereum virtual machine. 2017.
- [23] Investopedia. Ethereum 2.0: What you need to know, 2023. Accessed: 2023-01-12. URL: <https://www.investopedia.com/ethereum-2-0-6455959#:~:text=Ethereum%202.0%20significantly%20upgraded%20the,%2C%20accessibility%2C%20and%20transaction%20throughput>.
- [24] Shailak Jani. An overview of ripple technology & its comparison with bitcoin technology. Retrieved from ResearchGate: <https://www.researchgate.net/publication>, 2018.
- [25] S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. In *Neural Computation*. MIT Press, 2003.
- [26] JW Kolari, W Liu, and S Pynnönen. Building the global minimum variance portfolio. In ... with Machine-Made Portfolios and Stock ... Springer, 2024. URL: https://link.springer.com/chapter/10.1007/978-3-031-48169-7_7.
- [27] JW Kolari, W Liu, and S Pynnönen. The future of investment practice, artificial intelligence, and machine learning. In ... with Machine-Made Portfolios and Stock ... Springer, 2024. URL: https://link.springer.com/chapter/10.1007/978-3-031-48169-7_13.
- [28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi:[10.1038/nature14539](https://doi.org/10.1038/nature14539).
- [29] Ioannis E. Livieris, Emmanuel Pintelas, Stavros Stavroyiannis, and Panagiotis Pintelas. Ensemble deep learning models for forecasting cryptocurrency time-series. *Al-*

- gorithms*, 13(5), 2020. URL: <https://www.mdpi.com/1999-4893/13/5/121>, doi: [10.3390/a13050121](https://doi.org/10.3390/a13050121).
- [30] Srinjoy Mahato, Tanmoy Khatua, Ankan Das, and Mr Tathagata Roy Chowdhury. A brief discussion on two different cryptocurrency: Bitcoin & ethereum. *International Journal Of Computer Trends And Technology*, 67(9):32–38, 2019.
 - [31] Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Kim Pecina. Privacy preserving payments in credit networks. In *Network and distributed security symposium*, 2015.
 - [32] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.
 - [33] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press, 2016.
 - [34] Robert Nau. Arima modeling with R, 2023. URL: <https://people.duke.edu/~rnau/411arim.htm>.
 - [35] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59(3):183–187, 2017. doi:[10.1007/s12599-017-0467-3](https://doi.org/10.1007/s12599-017-0467-3).
 - [36] Ingolf Gunnar Anton Pernice and Brett Scott. Cryptocurrency. *Internet Policy Review, Glossary of decentralised technosocial systems*, 10(2), 2021. Available at SSRN: <https://ssrn.com/abstract=3899258>.
 - [37] Emmanuel Pintelas, Ioannis E. Livieris, Stavros Stavroyiannis, Theodore Kotsilieris, and Panagiotis Pintelas. Investigating the problem of cryptocurrency price prediction: A deep learning approach. In Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 99–110, Cham, 2020. Springer International Publishing.
 - [38] Derek A Pisner and David M Schnyer. Support vector machine. In *Machine learning*, pages 101–121. Elsevier, 2020.

- [39] Chris Richter, Sascha Kraus, Ricarda B Bouncken, et al. Virtual currencies like bitcoin as a paradigm shift in the field of transactions. *International Business & Economics Research Journal (IBER)*, 14(4):575–586, 2015.
- [40] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.
- [41] Refk Selmi, Aviral Tiwari, Shawkat Hammoudeh, et al. Efficiency or speculation? a dynamic analysis of the bitcoin market. *Economics bulletin*, 38(4):2037–2046, 2018.
- [42] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018. doi:10.1109/ICMLA.2018.00227.
- [43] Alexander J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.
- [44] Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm—a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.
- [45] Melanie Swan. *Blockchain: Blueprint for a new economy*. ” O’Reilly Media, Inc.”, 2015.
- [46] Tim Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. *Report, available online*, 28, 2015.
- [47] Vytautas Tumas, Sean Rivera, Damien Magoni, and Radu State. Topology analysis of the xrp ledger. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, SAC ’23, page 1277–1284, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3555776.3577611.
- [48] Peter Whittle. Hypothesis testing in time series analysis. (*No Title*), 1951.
- [49] XRP Ledger Foundation. rippled, 2022. Online; accessed 9. Dec. 2023. URL: <https://github.com/XRPLF/rippled>.

- [50] Robail Yasrab and Michael Pound. Phenomnet: Bridging phenotype-genotype gap: A cnn-lstm based automatic plant root anatomization system. 05 2020. doi:10.1101/2020.05.03.075184.
- [51] Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. *Dive into deep learning*. Cambridge University Press, 2023.
- [52] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375, 2018.
- [53] Thanasis Zoumpiskas, Elias Houstis, and Manolis Vavalis. Eth analysis and predictions utilizing deep learning. *Expert Systems with Applications*, 162:113866, 2020. URL: <https://www.sciencedirect.com/science/article/pii/S0957417420306758>, doi:10.1016/j.eswa.2020.113866.

Abbreviations

ACF	Autocorrelation Function
ADF	Augmented Dickey-Fuller
AR	Autoregressive
ARIMA	Auto Regressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BTC	Bitcoin
CNN	Convolutional Neural Network
ETH	Ethereum
EVM	Ethereum Virtual Machine
FBA	Federated Byzantine Agreement
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GRU	Gated Recurrent Unit
IDE	Integrated Development Environment
KDE	kernel density estimate
LSTM	Long Short-Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MSE	Mean Squared Error
PACF	Partial Autocorrelation Function
PoS	Proof of Stake
PoW	Proof of Work
RBF	Radial Basis Function
RNN	Recurrent Neural Networks
RMSE	Root Mean Squared Error

RTXP	Ripple Transaction Protocol
SVR	Support Vector Regression
SVM	Support Vector Machine
UNL	Unique Node List
VS code	Visual Studio Code
XRP	Ripple