UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

FORECASTING METHODS IN FINANCE: A COMPARATIVE STUDY IN PREDICTING BITCOIN AND STOCK MARKET FLUCTUATIONS

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en TICs

Autor/a:

Zapatier Cortez Luis Alberto

Tutor/a:

PhD - Rigoberto Salomón Fonseca Delgado

Urcuquí, junio 2024

Autoría

Yo, **Luis Alberto Zapatier Cortez**, con cédula de identidad 1206093021, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, junio 2024.

Luis Alberto Zapatier Cortes

CI:1206093021

Autorización de publicación

Yo, **Luis Alberto Zapatier Cortez**, con cédula de identidad 1206093021, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, junio 2024.

Firmado electrónicamente por:
LUIS ALBERTO
ZAPATIER CORTEZ

Luis Alberto Zapatier Cortez

CI: 1206093021

Dedication

Con profundo agradecimiento y amor, dedico esta tesis a mi familia, cuyo apoyo incondicional ha sido esencial en mi camino personal y universitario.

A mi querida madre, cuyo incansable esfuerzo diario me ha brindado no solo una infancia llena de alegría sino también una educación sólida, permitiéndome perseguir mis metas con fervor. Su dedicación inquebrantable a mi felicidad y bienestar ha sido la luz guía en mi camino hacia la realización de mis metas.

Mi padre ha sido un pilar de sabiduría y apoyo. Su presencia y consejos, basados en su vasta experiencia, han sido guías valiosas, incluso cuando nuestras opiniones no coincidían. Siempre dispuesto a escuchar y comprender, ha sido clave en mi toma de decisiones.

A mi hermano, cuya presencia constante y genuino interés en mi vida han sido invaluables. A pesar de nuestras diferencias en mentalidad y personalidad, su fe inquebrantable en mí y su deseo de verme triunfar han sido fuente de inspiración y motivación.

A mi nana, amor y apoyo han sido constantes a lo largo de mi vida. Su presencia ha sido tan importante como la de cualquier miembro de mi familia, brindándome un apoyo incondicional en cada etapa de mi desarrollo.

Finalmente, a mi pareja, cuyo amor y compañía han sido mi refugio y fortaleza. Su fe en mis capacidades y su apoyo han sido un gran soporte en los momentos más desafiantes.

Cada uno de ustedes ha jugado un papel insustituible en mi viaje, y esta realización es tan suya como mía.

Luis Zapatier Cortez

Resumen

Este estudio evalúa modelos avanzados como SARIMA, Holt-Winters, Prophet, LSTM y SARIMAX para el análisis y predicción de series temporales, enfocándose en la evolución del precio de las acciones de Amazon y el bitcoin. Esta investigación no solo busca pronosticar la tendencia de estos activos financieros, sino también evaluar su potencial rentabilidad en estrategias de inversión, considerando escenarios de compra y venta.

Se explica detalladamente la lógica subyacente de cada modelo, proporcionando una comprensión clara de las series temporales y su relevancia en el contexto financiero, así como una introducción a las redes neuronales y su aplicación en este campo. Este fundamento teórico es esencial para asegurar una adecuada comprensión de la metodología adoptada y los experimentos realizados.

Durante la investigación, se analizaron diversas variantes de estos modelos, ajustando sus hiperparámetros y experimentando con la incorporación de dos variables exógenas distintas. Entre los hallazgos más destacados, los modelos LSTM, Holt-Winters y SARIMAX mostraron un rendimiento particularmente sobresaliente. Estos modelos demostraron ser capaces de generar rentabilidades superiores al 30% en el caso de las acciones y por encima del 3% en criptomonedas, subrayando su potencial para aplicaciones prácticas en el ámbito financiero.

Palabras Clave:
Prophet, LSTM, Holt-Winter, SARIMA, SARIMAX, Forecasting.

Abstract

This study evaluates advanced models such as SARIMA, Holt-Winters, Prophet, LSTM, and SARIMAX for time series analysis and forecasting, focusing on the price trends of Amazon stocks and Bitcoin. This research aims not only to predict the trajectories of these financial assets but also to assess their potential profitability in investment strategies, considering buying and selling scenarios.

The underlying logic of each model is explained in detail, providing a clear understanding of time series and their significance in the financial context, as well as an introduction to neural networks and their application in this field. This theoretical foundation is essential to ensure a proper grasp of the adopted methodology and the experiments conducted.

Throughout the research, various versions of these models were analyzed, adjusting their hyperparameters and experimenting with the incorporation of two different exogenous variables. Among the most notable findings, the LSTM, Holt-Winters, and SARIMAX models exhibited particularly outstanding performance. These models demonstrated the capability to generate returns exceeding 30% in the case of stocks and over 3% in cryptocurrencies, highlighting their potential for practical applications in the financial realm.

Keywords:
Prophet, LSTM, Holt-Winter, SARIMA, SARIMAX, Forecasting.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In the realm of investments, it has always been crucial to have a comprehensive understanding of the stock market movements and the flow of stocks. This understanding is complemented by accurate intuition and the ability to guess the optimal timing for buying and selling operations. The main goal of these activities is to maximize profits by acquiring stocks at low prices and selling them when their value increases.

The inherent complexity of these investment operations leads many people to avoid this method for generating income. Instead, it is common to resort to stockbrokers who carry out the necessary analyses. However, this option involves additional costs, not only the capital to invest but also the broker's fees. Thus, to invest safely and profitably, a significant initial investment is required.

This financial barrier causes many individuals to prefer safer options, such as bank deposits that offer interest rates, up to 7% annually in the best cases, under specific terms and conditions established by the bank.

1.2 Problem statement

However, current technologies offer innovative tools that can simplify the investment process. One of the main innovations in this field are time series models, capable of generating short-term predictions about specific market variables. Despite these tools, the task of investing effectively remains complex, especially for those without experience in the sector.

Furthermore, the diversity of investment options has expanded, including cryptocurrencies, which are characterized by their high volatility. This volatility, while offering opportunities for quick profit, also represents a significant challenge in terms of prediction and risk management. Therefore, the identification of effective time series models that can adapt to both the stock and cryptocurrency markets is necessary.

In this context, the main problem of the research is to determine which time series models can predict the trends in the stock and cryptocurrency markets more effectively, in order to facilitate informed and profitable investment decisions. The goal is to overcome the entry barriers for less experienced investors and facilitate access to investments in these volatile markets.

1.3 Objectives

1.3.1 General Objective

The general objective of this research is to compare the efficacy and profitability of various time series models in making investment decisions in the stock and cryptocurrency markets. The aim is to identify a model that is not only profitable but can also be implemented automatically, providing an easy-to-use tool for investing, thereby reducing the need for extensive prior market knowledge.

1.3.2 Specific Objectives

1. Evaluate the effectiveness of different time series models, Holt-Winters, Prophet, LSTM, SARIMA, and SARIMAX, in predicting market trends

2. Analyze the feasibility of these models in simulating investment operations in the stock and cryptocurrency markets, with a specific focus on Amazon stock and Bitcoin.

3. Determine which of these models offers the best combination of profitability and ease of use for investors with different levels of experience.

4. Develop a theoretical framework for the automatic implementation of a selected model that democratizes access to the world of investments, making it more accessible and understandable to a wider audience.

1.4 Contribution

This thesis provides a detailed experimental analysis of advanced forecasting models such as SARIMA, Holt-Winters, Prophet, LSTM, and SARIMAX, focusing specifically on the investment field. A key aspect of this research is the ability to identify and discard those models that prove unsuitable for implementation in the financial sector, while highlighting those that demonstrate superior performance.

A significant contribution of this study is the development of an automated process that facilitates access to the world of investments. This approach is particularly valuable for individuals with limited knowledge in the area, allowing them to participate more effectively and with greater confidence in the financial market. The integration of advanced forecasting technologies into an easy-to-use system represents a significant step forward in the democratization of access to financial investments.

Chapter 2

Theoretical Framework

2.1 Basic Definitions

This section is dedicated to exploring the fundamental concepts that were considered during the preparation of this work. It includes a detailed description of time series, highlighting their relevance and applications. Moreover, the cryptocurrency and stock markets are examined, emphasizing their dynamics and influence in the current economic context. Subsequently, the prediction methods used and the logic behind each one are described. This chapter aims to provide a solid theoretical foundation, essential for a comprehensive understanding of the subsequent sections of the work.

2.2 Time Series

Time series are defined as a sequence of data that has been collected or recorded at consecutive time intervals, and these intervals are usually regular. This methodology is applied in a wide range of contexts, underscoring its versatility and relevance in multiple fields. For example, in the commercial realm, it can be used to analyze the monthly sales figures of a store, offering crucial insight for business strategies and decision-making. In the financial sector, it is fundamental for tracking stock prices, providing investors and analysts with an essential tool for predicting trends and valuing investments. Beyond these uses, time series also play a vital role in science and technology: in seismology, seismic signals are recorded as time series to monitor and predict tectonic activity; similarly, in the field of medicine, time series are used to analyze electrical signals from the human body, such as

the electrocardiogram (ECG), which is crucial for diagnosis and health monitoring. This broad spectrum of applications demonstrates the importance and utility of time series in various areas of study and professional practice [2].

As evidenced, time series are indispensable tools in a variety of fields, including statistics, economics, engineering, and climatology, among others. The essence of time series lies in their ability to monitor and analyze how a specific variable behaves over time. In statistics, for example, time series allow analysts to identify trends and patterns in data, facilitating evidence-based decision-making [3]. In the economic field, they are used to predict indicators such as Gross Domestic Product (GDP), inflation, and employment rates, which are crucial for the formulation of economic policies. In engineering, time series play a vital role in quality control and process optimization. In the field of climatology, the collection and analysis of temporal data allow scientists to understand and predict climate patterns, which is essential for addressing environmental challenges. These examples illustrate the breadth and depth of the application of time series, highlighting their importance in research and professional practice for understanding and forecasting the behavior of phenomena over time [4].

Time series are analyzed through three common approaches. The first, descriptive analysis, is the most basic. As its name indicates, this analysis focuses on describing the events observed in the time series. It identifies critical points, trends, stationarity, as well as peaks and troughs. This analysis seeks to offer a comprehensive understanding of the observed data, allowing an understanding of what happened with the variable under study [2].

The second approach is explanatory or inferential analysis. This method goes beyond mere description and seeks to explain the reasons behind the observed patterns. It focuses on identifying cause-and-effect relationships, analyzing why certain changes occur, such as notable growth, seasonality, or downward trends. This analysis is crucial for understanding the underlying dynamics of the time series [2].

Finally, the third approach, and the one this work focuses on, is predictive analysis. This

method involves using historical data to make projections about the future behavior of the time series. In this study, we employ predictive analysis to anticipate whether the price of the variable in question will increase or decrease, which is fundamental for making informed decisions. Each of these three analyses provides valuable information and offers different perspectives on the time series. Depending on the specific objectives of the study, one can choose to use one or a combination of these methods [3].

Defining analysis objectives is a crucial step, as it guides the selection and collection of relevant data. A fundamental aspect of this process is determining the temporal frequency of the data, that is, its periodicity. In this study, a daily periodicity was chosen for the analysis of stocks, focusing exclusively on business day, reflecting the dynamics of the stock market. On the other hand, for the Bitcoin analysis, a five-minute periodicity was chosen. This choice is due to the volatile nature and the constant flow of changes in the value of Bitcoin, making a shorter time interval feasible and relevant.

It is imperative to maintain the same periodicity throughout the entire series to ensure consistency and accuracy of the analysis. Moreover, it is crucial to define the time horizon of the study, that is, the length of the historical record that will be used for predictions. In this work, time series of different historical lengths were experimented with. This allowed for comparison and selection of the most effective length for applications in real-world scenarios [3].

The main components of a time series are crucial for its analysis and understanding. These include:

- **Trend:** This component reflects a general pattern or direction in the data over time. It can manifest as a sustained increase or decrease. Identifying a trend helps to understand the long-term trajectory of the time series, indicating, for example, continuous growth in a company's sales or a gradual decrease in average temperature due to climate change [1].
- **Seasonality:** Represents regular and predictable fluctuations that occur in specific

cycles. These fluctuations can be daily, weekly, monthly, or annually. For example, an increase in toy sales during the Christmas season or daily traffic patterns in a city. Seasonality is crucial for planning and anticipating periodic variations [1].

- **Cycle:** This component refers to long-term fluctuations that are not of a seasonal nature. Cycles can be less predictable and not follow a fixed pattern. For example, economic cycles that include periods of recession and expansion. These cycles are important for understanding the longer-term dynamics that could influence the time series [1].

- **Noise or Irregularity:** This component captures random variability in the data, that is, fluctuations that are not due to trend, seasonality, or cycles. Noise can be the result of measurement errors, unexpected events, or any other inexplicable variation in the data. Understanding and mitigating noise is essential for improving the accuracy of predictions and analysis of the time series [2].

In time series analysis, it is fundamental to distinguish between two main categories of models used for the interpretation and prediction of future data. These categories are:

- **Linear Models:** These models assume a linear relationship between past and future data. A prominent example is the ARIMA (Autoregressive Integrated Moving Average) model. ARIMA is particularly useful for analyzing and forecasting time series that are stationary, i.e., those whose statistical properties such as mean and variance do not change over time. This model combines autoregressive and moving average techniques, allowing it to model a variety of temporal patterns in data [1].

- **Non-Linear Models:** These models are suitable for capturing more complex relationships that cannot be adequately described by linear models. An example is neural network models. These networks are especially valuable for time series with complex and non-stationary patterns, as they can learn from data and detect non-linear interactions and long-term dependencies. Their flexibility and deep learning capability make them ideal for a wide range of applications, from financial market prediction to climate trend analysis [2].

The choice between a linear and a non-linear model depends on the nature of the time series and the specific objective of the analysis. While linear models are often simpler and easier to interpret, non-linear models offer greater flexibility and potential to capture more complex dynamics in data [3].

In summary, time series emerge as a powerful and versatile analytical tool, fundamental for understanding how data evolves over time. Their ability to identify consistent patterns and make informed predictions makes them indispensable in multiple disciplines. Particularly, their application in the financial field, which will be the focus of this work, is of vital importance. In the financial world, time series facilitate the understanding of market trends, allowing investors and analysts to make more informed decisions based on historical analysis and future projections of financial instruments like stocks and cryptocurrencies. The ability to predict market behavior and anticipate price movements is crucial for success in this field, and time series provide the necessary tools to perform these tasks with greater accuracy and confidence. This work will delve into the application of time series in the financial field, demonstrating their effectiveness and relevance in a sector where precision and predictive analysis are of utmost importance.

2.3 Types of Prediction

In the field of prediction, two fundamental methodologies stand out: the walk-forward method and the recursive method. Each of these approaches has specific characteristics and applications that make them suitable for different types of data analysis and forecast scenarios.

2.3.1 Walk-Forward

Walk-Forward Analysis is a key technique in the prediction and modeling of time series, particularly standing out in the field of finance and investments. This methodology is based on making predictions using current actual values, instead of relying solely on past predictions as a basis, which is essential to differentiate it from the recursive method [7].

Key features of Walk-Forward Analysis:

- **Basic Concept:** It involves training a predictive model with a set of historical data, followed by testing on a subsequent time period not included in the training. In each cycle, the model forecasts one step ahead using the latest real data available and then compares the outcome with the following actual data to evaluate its accuracy [8].
- **Practical Approach:** This approach simulates a realistic trading situation, using recent and current data for each prediction, which helps to assess how the model would have performed under real market conditions and reduces the risk of overfitting [6].
- **Process Steps:**
  1. **Data Division:** The data is divided into a training period and a testing period [7].
  2. **Model Training:** The training set is used to develop the model [7].
  3. **Validation and Testing:** The model is applied to the test set [7].
  4. **Repetition:** The process is repeated, advancing the training and testing periods to cover different data segments [7].
- **Prevention of Overfitting:** One of the main reasons for using Walk-Forward Analysis is its ability to prevent overfitting, maintaining the model's accuracy and relevance for future data [7].
- **Applications in Finance:** This method is widely used in finance to demonstrate a model's effectiveness for predictions, leveraging its ability to adapt to changing market data [6].

The figure 2.1 [9], presents the Walk-Forward time validation scheme through five successive passes. Each pass illustrates the division of the available time series into segments used for training and testing the model. The dark gray sections represent the discarded data, the light gray sections show the data used for training the model, and the orange sections

indicate the forecast periods. This sequential approach of advancing the training and forecast period with each pass reflects how the model adjusts and validates in relation to the proximity of the present time, thereby improving its ability to predict future observations more effectively.

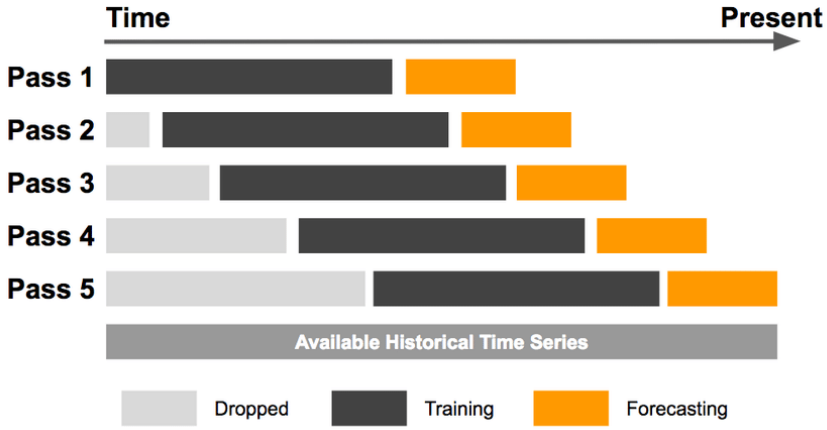


Figure 2.1: Walk-Forward Time Validation Scheme.

2.3.2 Recursive

Recursive prediction is an advanced technique in the realm of forecasting, especially relevant in contexts where a continuous projection over time is required. This approach is distinguished by its use of predictions generated by the model itself as inputs for future predictions, representing a fundamental difference from methods that rely exclusively on actual historical data [10].

Key features of the recursive method:

- **Basic Concept:** In recursive prediction, a model relies on its own previous predictions, in addition to historical data, to generate new predictions. This approach allows the model to "learn" from its own performance and adjust its future predictions accordingly [11].
- **How the Method Works:**
  1. **Start with Historical Data:** The model begins its analysis with a set of historical data to make the first prediction [11].

2. **Recursive Update:** Subsequent predictions incorporate both the historical data and the previous predictions made by the model [10].
3. **Continuous Process:** This cycle of updating and predicting repeats continuously, with each new prediction feeding the dataset for the next one [11].

- **Advantages and Challenges:**
  1. **Adaptability:** Recursive prediction is valuable in scenarios where real data are not immediately available or when consistent trends are expected [10].
  2. **Risk of Deviation:** A significant challenge is the potential accumulation of errors. Incorrect predictions can negatively influence future ones, increasing the risk of significant deviations from reality [10].
- **Applications in Finance:** This method is used in modeling long-term investment scenarios and in the development of anticipative trading strategies. However, it is crucial to be aware of the risk associated with the accumulation of errors [10].
- **Contrast with Walk-Forward:** Unlike Walk-Forward analysis, which focuses on avoiding overfitting and validating the robustness of the model with real data, recursive prediction focuses on continuous projection based on its own outputs, which can increase susceptibility to cumulative errors if initial predictions are not accurate.

In the context of this work, where the goal is to predict short-term changes in stock and cryptocurrency prices, the recursive method stands out for its ability to project several steps into the future, using its own predictions to advance. This offers a broader timeframe for making informed and strategic decisions.

Figure 2.2 [12], shows the successive phases of the recursive prediction method. At the top, the starting point with the initial prediction T_{n+1} is based on actual historical data up to T_n. In the middle, the next prediction T_{n+2} uses both the historical data and the previous prediction T_{n+1}. At the bottom, the progression of the model is observed, where the prediction T_{n+3} is made based on the previous predictions T_{n+1} y T_{n+2} ,showing how the model feeds on its own predictions to advance in time. Solid bars represent observed

values, dotted lines represent the predicted values, and circles indicate the point at which the model is used to predict the next step.

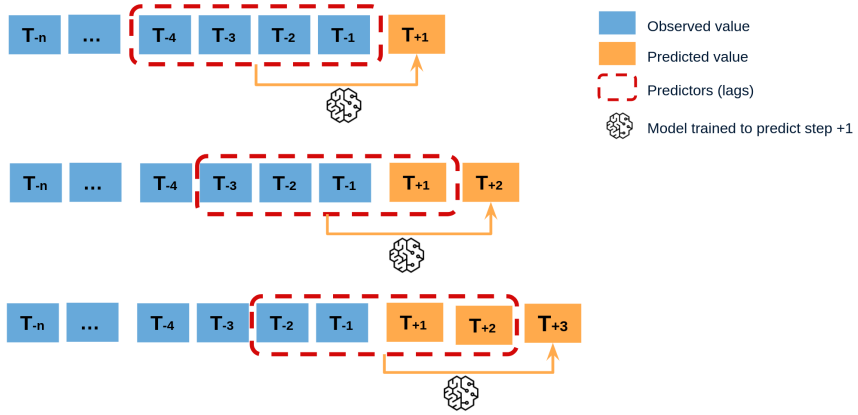


Figure 2.2: Illustration of the Recursive Prediction Process.

2.4 Methods

2.4.1 Univariate Method

The univariate method in time series prediction uses the historical values of a single variable to predict its future values. This approach assumes that the past information of the variable is sufficient to understand and project its future behavior. Univariate models, such as ARIMA, rely on the series' own dynamics to make forecasts [1].

2.4.2 Method with Exogenous Variables

Unlike the univariate method, the approach with exogenous variables incorporates additional information in the form of other variables that are believed to influence the main variable to be predicted. This method is useful when the variable of interest is affected by external factors, allowing for modeling and prediction that takes into account the broader environment [3].

In the present study, a comprehensive comparative analysis of various predictive models employing both the univariate method and the method with exogenous variables has been

undertaken. The goal is to evaluate and contrast the efficacy of each model in a variety of scenarios.

2.5 Stocks

In simple terms, stocks represent a fraction of ownership in a publicly traded company. The buying and selling of these stocks involve acquiring or relinquishing parts of the company, based on market fluctuations. Investors can earn income in two ways: through dividends, which are smaller but consistent earnings as the value of the company increases, or by selling the stocks themselves. The latter method relies on buying stocks when their price is low and selling them when their value increases. The central focus of this work is the ability to predict short-term fluctuations in stock prices. By anticipating these changes, the goal is to buy at the lowest possible price and sell at the highest estimated point, aiming to maximize profits [13].

2.6 Cryptocurrencies

To understand cryptocurrencies, it is essential to recognize that they function as a digital monetary system. Unlike traditional currencies in the gold standard era, their value is not derived from the material they are made of nor their physical utility. Cryptocurrencies are virtual currencies whose value is based on the social consensus of market participants who agree on their value. This value is primarily determined by their relevance in the market: the greater the interest and demand from investors, the higher their value. It's important to note that Bitcoin, while the most well-known, is just one among hundreds of cryptocurrencies available in the market today [15].

The term "cryptocurrency" comes from its encrypted nature, which offers robust security, making them extremely difficult to alter or hack. This security is one of their most significant advantages. Additionally, cryptocurrencies operate independently of banking or governmental entities, meaning they are not subject to the influence or control of these institutions. A distinctive feature of cryptocurrencies is their foundation on decentralized and encrypted technology, known as "Blockchain". This system records all transactions in

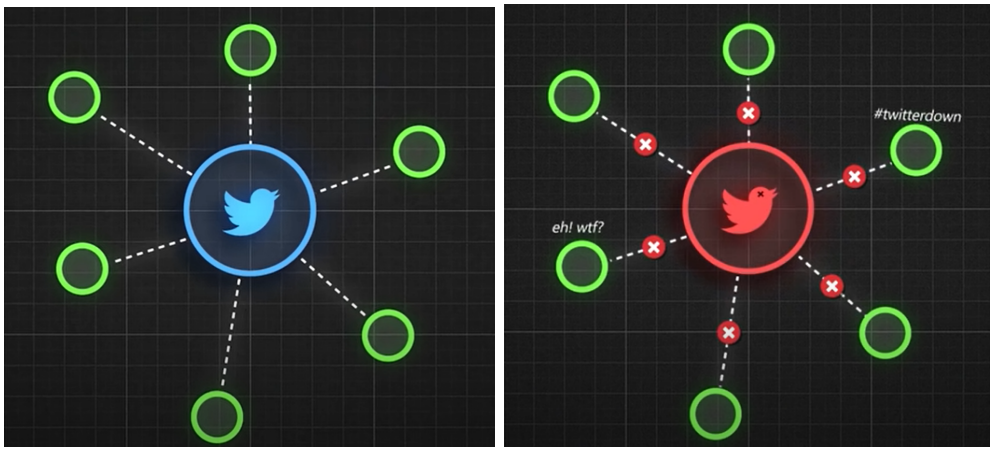
a block chain, ensuring transparency and traceability while maintaining user privacy and security [14].

Despite their advantages, it is crucial to recognize that the cryptocurrency market is largely based on speculation. This speculative nature leads to high volatility, causing prices to vary significantly from one day to the next or even within weeks. Such instability represents a considerable risk, especially for impulsive and uninformed investments. Moreover, in the realm of cryptocurrencies, it is not uncommon to find new coins that appear and disappear quickly, often created with the purpose of deceiving investors [15].

It is fundamental, therefore, to weigh both the positive and negative aspects when considering investments in cryptocurrencies. For this reason, this work has chosen to focus on Bitcoin. This decision is due to Bitcoin being one of the most established and reliable cryptocurrencies in the market, offering an additional degree of security against the risks mentioned.

2.6.1 Blockchain

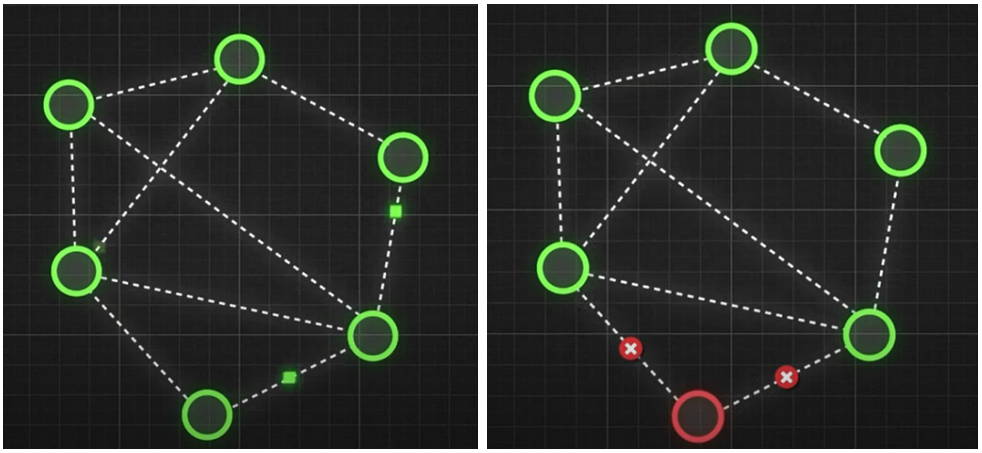
The concept of blockchain is intrinsically linked to the realm of decentralized systems in computing. To better understand this relationship, it's useful first to examine what constitutes a centralized structure, something most people are familiar with. A daily example is the downtime of services like WhatsApp, during which no one can access the app. This issue arises due to its centralized structure: all users connect to the company's servers to interact with other users on the network. In this model, the service is supported by a set of central nodes responsible for coordinating and processing every aspect of the network's processing. Although this implementation is effective and simple, it is not without disadvantages. For example, a failure in the central server can cause the entire network to collapse. Additionally, concentrating control, decision-making, and management at a single network point can open the door to unethical practices and a complete collapse when the server fails, see Figure 2.3 [14][16].



(a) Centralized System (b) Centralized System Down

Figure 2.3: Fallibility of a Centralized System.

The alternative to centralized structures is decentralized ones, where, instead of relying on a central point for network management, the participating nodes themselves manage the data. In these structures, decision-making is done collectively, ensuring that if one of the nodes fails, the network can continue to operate without significant interruptions, see Figure 2.4 [16]. However, this architecture presents much greater complexity and is harder to implement, entailing several unique challenges. This complexity arises from the need to coordinate multiple nodes, ensure data consistency across the network, and manage security in a distributed environment [15].



(a) Decentralized System (b) Decentralized System Down

Figure 2.4: Fallibility of a Decentralized System.

Coordination in a decentralized system presents unique challenges, as it operates under

the principle of collective self-management, without depending on a central authority. In this context, all participants interact in a peer-to-peer (P2P) model. This methodology, applied to the financial sector, constitutes the basis of the operation of cryptocurrencies, a decentralized financial system [15].

In cryptocurrency networks, each node transmits specific information, such as transactions between users. For example, in the case of a cryptocurrency like Bitcoin, what is transmitted and recorded on the network is the complete set of transactions made by users. When a user performs a transaction, this information is distributed to all other network nodes. In this way, each participant maintains an updated and consistent copy of the transaction record, ensuring the transparency and verifiability of all operations within the network. The decentralized nature of cryptocurrencies presents significant challenges in terms of information distribution. It is crucial that all users receive consistent and up-to-date information to maintain accurate and synchronized records. However, in practice, information in these networks tends to flow at variable rates, meaning not all nodes receive the same information at the same time [14].

This lag can lead to a lack of coordination in the records. For example, while some nodes continue to record and share new transactions, others may not have yet received previous updates. This can result in different nodes having identical transaction records but in different orders. This situation poses a fundamental challenge: ensuring the consistency and reliability of information across the network, a vital task to maintain the integrity and trust in the cryptocurrency system [14].

To overcome the challenges of synchronization and transaction verification in blockchain networks, a technique known as “block formation” is employed. In this system, transactions are grouped into blocks within a set time window. This approach facilitates the verification of the validity of transactions, as any user can access and examine the complete record, organized by blocks. By verifying each block, users can confirm the authenticity of each transaction [14].

This block structure provides a significant level of security against fraud and data misrepresentation. Since each user maintains a copy of the complete record, manipulating or altering transactions would require modifying the record on all network nodes, an extremely difficult task. If an individual record is altered, it will be considered invalid, as it will not match the majority of the records in the network. Thus, the integrity and reliability of transactions on the blockchain are maintained through distributed consensus and the inherent difficulty of altering multiple records simultaneously. The blockchain system incorporates an advanced encryption mechanism based on secure hash algorithms. These algorithms generate a unique hash from a set of data. The distinctive property of these hashes is their sensitivity: even the slightest change in the input data results in a completely different hash. In the context of blockchain, each block is encrypted in this way, producing a unique hash for each block [14].

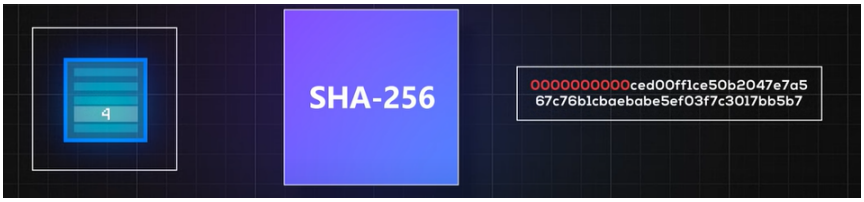


Figure 2.5: Hash algorithm output.

In the case of Bitcoin, for example, the hash algorithm dictates that the generated hash must start with a specific number of zeros, see Figure 2.5 [17]. To achieve this, the block and a variable integer are provided as input to the algorithm. The difficulty lies in the fact that the only way to find the correct hash is by using significant computational resources to find the right combination of the integer [14].

This process, known as "mining," is open to any user on the peer-to-peer network and is public in nature. When a user successfully finds the correct hash for a block, they have the right to add that block to the common record maintained by all users. In simple terms, this user has "mined a block," thereby contributing to the extension of the blockchain and receiving a cryptocurrency reward for their contribution. The constant operation of mining on the Bitcoin blockchain network generates a continuous sequence of encrypted blocks,

representing the transactions made. A scenario that may arise is two users mining a valid block at the same time, leading to the temporary creation of two different versions of the blockchain. These two chains coexist until a new block is mined on one of them. At this point, the conflict is resolved by applying the "longest chain" rule, where the chain with the additional block becomes the official version accepted by the network [14].

This process ensures that, although temporary discrepancies may arise, the network as a whole will converge towards a single version of the truth. Block mining, given its computational intensity and associated costs, comes with a reward. The miner who succeeds in adding a block to the chain receives a special transaction, known as a block reward, which grants them a certain amount of bitcoins. This reward not only compensates for the costs of mining but also incentivizes active and honest participation in the network [14].

The inherent security of the blockchain system derives from its decentralized nature, its public transparency, and the mining processes involved. A crucial aspect of this security is the way each block uses the hash of the previous block in its own encryption, see example in Figure 2.6 [17]. As mentioned previously, any change in the data of a block alters its hash, which in turn affects the hashes of all subsequent blocks. This means that if an attacker attempts to modify an old block, they would have to redo the mining of all the following blocks to maintain the chain's consistency [14].

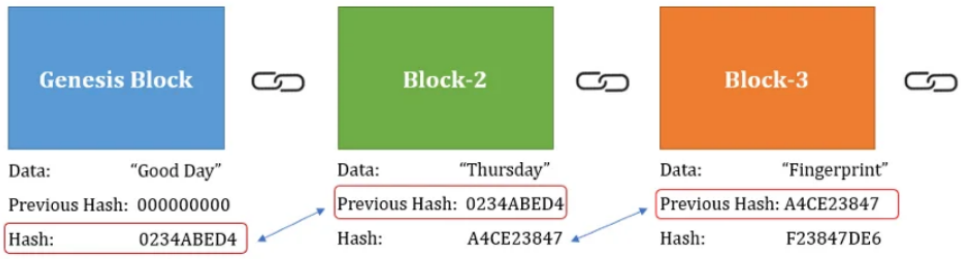


Figure 2.6: A Simplified Illustration of Blockchain Technology.

In practice, altering a block in the blockchain chain is further complicated by the fact that, while the attacker works on modifying a block, the rest of the network continues to add new blocks. This creates a situation where the attacker would need computational capacity superior to the total sum of all other network users for their altered chain to be considered valid. In large-scale networks, like Bitcoin, this becomes practically an impossible task. Therefore, the combination of these factors ensures that the blockchain system is extremely secure and reliable, being open and accessible to all users without compromising its integrity [15].

2.7 Neural Networks

A neural network is a computational model that mimics the way biological neural networks in the human brain process information. It consists of a series of interconnected nodes or "artificial neurons" that collaborate to perform specific tasks. Unlike traditional neural networks, LSTM (Long Short Term Memory) networks are designed to remember information for longer periods, making them particularly suited for tasks involving data sequences, such as natural language processing. Neural networks, including LSTMs, are key components in the field of artificial intelligence and machine learning, allowing machines to learn from data and improve their performance in complex tasks [20].

A typical neural network has three types of layers: an input layer, one or more hidden layers, and an output layer. Each layer consists of units or nodes, which can receive and process signals [20].

Neural Network Structure::

- **Input Layer:** The input layer is the starting point where the network receives data from the outside world. Each node in this layer represents a unique feature of the input data. For example, in an image recognition task, each node could correspond to the intensity of a specific pixel in the image. This layer acts as an intermediary, passing the data to the subsequent layers without performing any significant transformation on them [18].

- **Hidden Layers:** After the input layer come one or more hidden layers. These layers are where most of the network's computational processing occurs. The nodes in these layers transform the inputs received from the previous layer through complex mathematical operations. They are the "heart" of the network, enabling the learning and recognition of complex patterns and features from the input data [18].

- **Output Layer:** The output layer marks the end of the processing pathway of the network. The nodes in this layer take the transformations made by the hidden layers and generate the final output of the network. The design of this layer varies according to the specific task the network performs. For example, in a classification task, each node in the output layer could represent a different category, delivering as a result the probability of the input belonging to each category [18].

Node Components

Nodes, or artificial neurons, are the basic elements of a neural network [20]. Each node is composed of several key components:

- **Neurons or Nodes:** Each node in a neural network simulates the behavior of a biological neuron. These nodes receive input signals (either from external data or from previous nodes in the network), process them, and transmit an output to the following nodes. The operation of each node is crucial for data processing and generating meaningful results [19].

- **Weights:** These are coefficients applied to the inputs of the nodes. During the network's learning process, these weights are adjusted to improve the accuracy of the network's predictions or classifications [19].

- **Biases:** Function as an independent adjuster within the node, allowing to modify the node's output in addition to the weighted inputs. This component helps the network better adapt to patterns in the data, even when inputs are low or null [19].

- **Activation Functions:** The activation function in each node determines if and how that node will be activated, i.e., how it will process and transmit the signal. These functions can be linear or, more commonly, non-linear, allowing the network

to learn and represent complex relationships in the data. Common examples include the sigmoid function, the ReLU (Rectified Linear Unit) function, and the hyperbolic tangent. Choosing the right activation function is crucial for the performance and effectiveness of the network [18].

- **Connections Between Nodes:** Nodes in different layers of the network are interconnected. These connections are the pathways through which signals are transmitted from one node to another. The way these nodes are connected largely defines the network's architecture and affects its ability to process information and learn from data [19].

- **Varied Architectures:** There are various neural network architectures, each optimized for specific tasks. For example, Convolutional Neural Networks (CNNs) are ideal for image processing, while Recurrent Neural Networks (RNNs) are suitable for processing temporal sequences such as language [18].

Information Processing in Nodes

The processing of information in a neural network begins at the individual nodes of each layer [20]. The operation of each node can be broken down into the following key steps:

- **Reception of Inputs:** Each node receives a series of inputs. In the input layer, these inputs are the raw data that the network is processing. For example, in a network designed for image recognition, the inputs would be the pixel values of the image. In subsequent layers, the inputs to a node come from the outputs of the nodes in the previous layer [18].

- **Multiplication by Weights:** Weights are crucial parameters that determine the influence of each input on the output of the node. Each input is multiplied by a corresponding weight. These weights are adjustable and are modified during the training of the network. This multiplication weights the importance of each input, allowing the network to focus on particular features of the data. For example, in object identification in images, certain weights may emphasize aspects such as edges or specific colors [18].

- **Integration of Weighted Inputs:** Once each input has been multiplied by its corresponding weight, all these weighted inputs are added together to form a total sum. This sum represents the combined information that the node will receive and process [18].

Weighted Sum and Bias:

Once the nodes of a neural network have received and multiplied their inputs by the corresponding weights, the next step is the weighted sum and the integration of the bias [20].

- **Weighted Sum of Inputs:** The weighted inputs of a node, resulting from multiplying each input by its associated weight, are added together to form a total sum. This weighted sum is the basis for the next stage of processing in the node. It represents the integration of all input signals, each adjusted in importance by its specific weight [19].

- **Incorporation of Bias:** A bias term is added to the weighted sum. The bias is an additional parameter in each node that allows adjusting the output of the node independently of the inputs. This addition is crucial because it provides the node with a kind of activation threshold. Even in the absence of significant inputs, or when all inputs are zero, the bias allows the node to produce a non-null output. This aspect is especially important to ensure that the neural network can handle a variety of situations, including those where input information may be minimal or nonexistent [19].

Activation Function

The activation function in the nodes of a neural network is a critical component that adds non-linearity to the network's learning and processing. Once the net input of a node has been calculated through the weighted sum and bias, it is passed through the node's activation function [20].

1. **Purpose of the Activation Function:** The activation function determines the output of the node based on the net input received. Its main objective is to introduce

non-linearities into the system, allowing the network to learn and model complex relationships in the data. Without these functions, the neural network would essentially be a linear model, limiting its ability to handle complex and varied tasks [18].

2. **Common Types of Activation Functions:**

   - **Sigmoid Function:** Produces an output in the range of 0 to 1. It is useful for models where we need probabilities, as its output can be interpreted as a probability [18].
   - **ReLU (Rectified Linear Unit) Function:** Provides a non-negative output (zero or positive). It is popular in deep neural networks due to its computational simplicity and effectiveness in training [19].
   - **Hyperbolic Tangent Function:** Offers an output in the range of -1 to 1. It is similar to the sigmoid function but can handle negative values more effectively [20].

3. **Impact on Network Learning:** The choice of the activation function has a significant impact on how the network processes information and learns. Different functions may be more suitable for different types of tasks. For example, ReLU functions are commonly used in convolutional networks for image processing, while the sigmoid function is often used in the last layer of networks intended for classification tasks [19].

Forward Propagation

Forward propagation is the process through which a neural network processes information, from the input layer to the output layer [18].

- **Information Flow in the Network:** In forward propagation, information flows sequentially through the network. It begins at the input layer, where the network receives external data. Then, this data is transmitted through the hidden layers, where it is successively processed [19].

- **Processing at Each Node:** At each network node, the calculation of the weighted sum of the inputs is performed, the bias is added, and the activation function is applied. This process transforms the input data in such a way that each subsequent layer receives a more processed and abstract version of the data [19].

- **Importance of Hidden Layers:** Each hidden layer can learn different aspects of the input data. For example, in a neural network for image recognition, the early hidden layers may learn to identify edges and textures, while deeper layers may learn to identify more complex shapes and patterns [19].

- **Result at the Output Layer:** Finally, the information reaches the output layer, where the network produces its final result. This result can vary depending on the task for which the network was designed, such as classification into categories, generating a continuous value in regression tasks, or even generating new images or text sequences in more advanced networks [19].

Feedback and Weight Adjustment (Backpropagation)

Once a neural network has processed data through forward propagation and generated an output, the next step is feedback and weight adjustment, a process known as backpropagation [20].

- **Error Evaluation:** Feedback begins with evaluating the network's performance by comparing the generated output with the desired or actual output. This comparison results in determining an error or difference, measuring how much the network's prediction has deviated from the expected result [18].

- **Error Propagation:** The calculated error is propagated backward through the network, from the output layer to the hidden layers. This process of reverse error propagation is what gives the term "backpropagation" its name [19].

- **Adjustment of Weights and Biases:** During backpropagation, the network's weights and biases are adjusted. This adjustment is made in a way that reduces the error in future predictions. The most common algorithm used for this adjustment is

gradient descent, where changes in weights are made in the direction that minimizes the error [19].

- **Importance of Gradual Learning:** Backpropagation allows the network to adjust its weights incrementally. In each iteration, the network learns from its errors and improves its ability to make more accurate predictions. This gradual learning is key to developing a neural network that is effective and precise in its task [19].

Iterative Learning

This process involves repeating cycles of forward propagation and backpropagation several times, allowing the network to continuously improve its performance [20].

- **Definition of Epoch:** Each complete step of sending the data through the network (forward propagation) and adjusting the weights (backpropagation) is called an "epoch". During an epoch, the network processes the input data set, makes predictions, compares these predictions with the actual results, and adjusts its weights accordingly [20].

- **Learning Process Over Epochs:** In each epoch, the network learns from its previous errors. The adjustments made to the network's weights and biases are designed to reduce the error in future predictions. With each iteration, the network becomes more accurate in its predictions or in performing the task it was designed for [19].

- **Convergence Towards an Optimal Solution:** Ideally, over many epochs, the neural network converges towards an optimal solution, where the error in its predictions is minimal. This convergence process is indicative of the network's effective learning [19].

- **Continuous Evaluation and Adjustment:** It is crucial to continuously evaluate the network's performance during the iterative learning process. This may involve adjusting the learning rate, modifying the network's architecture, or applying techniques such as cross-validation to ensure that the network is not only learning effectively but also generalizing well to new data [19].

Network Training and Validation

Network Training: Training is an essential process in the development of neural networks. This phase involves the iterative adjustment of the network's weights in response to a training set, which includes examples of inputs and the corresponding desired outputs. The main goal is for the network to learn to effectively map these inputs to their outputs, optimizing its performance through successive iterations. It is crucial to highlight that the quality and variety of the training set play a crucial role in the network's learning effectiveness, as they determine how and what the model learns [20].

Network Validation: Validation is a crucial step to assess the network's generalization ability. It is carried out using a validation set, consisting of data not present during the training phase. This process is vital to detect issues like overfitting, where the network shows high performance with training data but fails when facing new data. Validation helps ensure that the network not only memorizes the training examples but also learns to correctly infer from previously unseen inputs [18].

Adjustment of Hyperparameters in Neural Networks: Hyperparameters, which are variables not learned during training but that configure the architecture and behavior of the network, include the learning rate, the number of layers and neurons, the type of activation function, and regularization techniques, among others. Adjusting these hyperparameters is a significant and crucial challenge, as they have a direct impact on the network's efficiency and performance. Strategies such as grid search and random search are popular techniques for exploring different hyperparameter combinations. However, it is important to consider more advanced methods like Bayesian optimization, which can be more efficient in finding the optimal configuration for a given problem [19].

In the context of this research, the LSTM neural network architecture is adopted for its robustness and proven ability to make accurate predictions that require not only the retention of relevant short-term information but also the effective integration of long-term knowledge. This approach is aligned with the goal of developing predictive models that not

only capture the temporal essence of the data but also dynamically adapt to the evolution of underlying patterns see a description of the LSTM in Section 2.8.4.

2.8 Models

2.8.1 Prophet

Predicting time series involves unique challenges, such as accounting for trends and the influence of special events, such as holidays or festive days. These complexities require a prediction model specially adapted to this type of data. In response to this need, Facebook's Core Data Science team developed Prophet, an advanced prediction model designed to efficiently address these peculiarities [7].

The core of Prophet is based on an additive model where time series are composed of several parts:

$$X_t = T_t + S_t + H_t + \epsilon_t \tag{2.1}$$

- T_t**:** This component models the long-term trend of your data. It can capture non-periodic patterns and changes in the direction of the time series, such as increases or decreases not tied to seasonality. For example, it could represent a gradual growth of users on a social media platform over several years [21].

- S_t**:** The seasonality component handles patterns that repeat at regular intervals. Prophet allows flexibility in modeling both weekly and annual seasonality. This means it can adapt to fluctuations in the data that occur in consistent cycles, such as increases in sales over weekends or seasonal variations in hotel bookings [7].

- H_t**:** This component is for variations that occur on specific days that can be anticipated but do not follow a periodic pattern, like holidays or special events. For example, the model can be adjusted to forecast increases in sales during Black Friday or reductions in website traffic on New Year's Day [21].

- ϵ_t**:** Finally, the noise term captures the random irregularities in the data that the other components do not model. These could be anomalies or noise inherent in

the data measurement. It's an error term that helps keep the model realistic by recognizing that not all variations can be explained [7].

This formulation allows Prophet to flexibly adjust to the different characteristics of a time series, making it robust against changes and variations in the data.

Generalized Additive Models (GAM)

Generalized Additive Models (GAM) are an extension of linear models that allow the relationship between independent variables and the dependent variable to be non-linear and modeled by smooth functions. This makes them particularly suited for capturing complex and non-linear patterns in data. The Kolmogorov-Arnold representation theorem is a profound mathematical result stating that any continuous multivariate function can be represented as a sum of compositions of univariate functions. This is crucial for GAMs, as it provides a theoretical basis for decomposing a complex, multivariate function into simpler components that are easier to estimate and understand [21].

This functions as the heart of Prophet, the practical application of the Kolmogorov-Arnold representation theorem, which mathematically underpins the model's ability to decompose a complex time series function into more manageable components. Prophet leverages this theorem to transform the task of modeling a multivariate time series into estimating several simpler univariate functions. Each of Prophet's components - trend, seasonality, and holiday effects - can be viewed as a univariate function that adds up to reconstruct the original time series. This allows the model to flexibly and accurately address the various patterns and changes in the data, resulting in a powerful and adaptable prediction tool [21].

$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{2.2}$$

Where:

- Φ_q are external univariate functions.
- $\phi_{q,p}$ are internal univariate functions that only depend on one variable x_p.
- The outer sum iterates over a set of $2n$ functions Φ_q.

- The inner sum is over the n variables de entrada x_p, for each function Φ_q.

The Kolmogorov-Arnold representation theorem provides a theoretical basis for the decomposition of complex multivariate functions into simpler univariate components. However, the theorem itself does not offer a constructive methodology for such decomposition, which is essential in statistical modeling and machine learning. In practice, especially in predictive tools like Prophet, a simplification is necessary to make the approach computationally viable and applicable to predictive modeling problems [21].

The simplification can be expressed in the following mathematical form:

$$f(x_1, \ldots, x_n) = \Phi\left(\sum_{p=1}^{n} \phi_p(x_p)\right) \tag{2.3}$$

- f is the multivariate target function we want to model, in this case, the complete time series.
- Φ is a combination function that synthesizes the contributions of the univariate components.
- $\phi_p(x_p)$ are univariate functions representing the individual components of the time series, such as trend and seasonality.

This approach not only facilitates understanding and interpretation of the model but also ensures that Prophet can be specifically tailored to each time series, resulting in a powerful and versatile predictive model.

2.8.2 SARIMA

The SARIMA model, known as "Seasonal Autoregressive Integrated Moving Average Model", is an extension of the ARIMA model, specifically designed to capture seasonality in time series data. ARIMA, which stands for "Autoregressive Integrated Moving Average", is a popular statistical approach for analyzing and predicting time series data. This model is particularly useful when data exhibit non-stationary patterns but lacks tools to handle seasonality [1].

The main components of the ARIMA model are three, described through the parameters p, d, and q:

- **AR (Autoregressive) p:** his models the dependent relationship between a current observation and a number of previous observations. In the ARIMA model, it is assumed that future values of a time series can be explained by its past values [2].

  For example, in an AR model of order 1 $AR(1)$, the relationship is typically expressed as:

  $$X_t = c + \phi X_{t-1} + \varepsilon_t \tag{2.4}$$

  where X_t is the current value of the series, c represents a constant, ϕ is the coefficient measuring the degree of influence of the previous value X_{t-1} on the current value, and ε_t s the error term that captures random variations. This model is especially useful for predicting time series where a clear trend is observed in the past data. In higher-order AR models, like $AR(2)$ or more, the relationship extends to more past values, increasing the model's complexity and allowing it to capture more sophisticated dynamics in time series data [1].

- **I (Integrated) d:** This component is crucial for achieving stationarity in the time series through differencing. Differencing is performed by subtracting the current value of the series from its previous value, which helps eliminate trends and seasonality in the data. For example, the first difference of a series X_t is calculated as $\Delta X_t = X_t - X_{t-1}$. In some cases, it may be necessary to apply higher-order differences (like the second difference $\Delta^2 X_t = \Delta X_t - \Delta X_{t-1}$), especially when the time series shows more complex trends or non-linear patterns [1].

  The importance of this process lies in the fact that ARIMA models are more effective with time series that do not exhibit strong trends or seasonal patterns. By converting the series into a stationary one, the identification and modeling of the underlying structures in the data are significantly facilitated. This is essential for making accurate predictions and for understanding the true nature of the temporal relationships in the series [2].

- **MA (Moving Average) q:** This component focuses on modeling the prediction error as a combination of past errors. In a moving average model, the current value of the time series is expressed as a function of past error terms. For example, in an MA model of order 1 $MA(1)$, the relationship can be described as:

$$X_t = \mu + \varepsilon_t + \theta\varepsilon_{t-1} \tag{2.5}$$

  where μ is the mean of the series, $\theta$is the moving average coefficient measuring the degree of influence of the past error ε_{t-1} on the current value, and ε_t is the current error term [1].

  This component is especially valuable for adjusting the model to random fluctuations or "noise" present in historical data. By incorporating past error terms, the MA model can effectively 'smooth' the time series, making it more representative of underlying trends and less sensitive to short-term random variations. This is crucial for improving the accuracy of predictions and for better understanding the dynamics of errors in time series [2].

When the AR, I, and MA components are combined, the ARIMA model becomes a powerful tool capable of capturing and leveraging both long-term trends and short-term patterns in historical data. This process involves fitting the model to historical data to optimally estimate the parameters that best describe these patterns and trends. Once calibrated with these parameters, the ARIMA model is equipped to extrapolate and predict future values effectively. Consider, for example, a time series of monthly sales that shows an increasing trend over time, along with regular patterns in past prediction errors. In this scenario, a properly adjusted ARIMA model has the capability to predict future sales with reasonable accuracy. It relies on recognizing these trends and patterns, using historical information not only to understand how sales have changed in the past but also to anticipate how they might evolve in the future. This approach makes the ARIMA model particularly valuable in environments where accurate understanding and prediction of time series dynamics are crucial for decision-making [7].

It is crucial to recognize that the accuracy of the predictions generated by the ARIMA

model is intrinsically linked to the quality and characteristics of the used historical data, as well as to the suitability of the specific model to those data. The importance of selecting a model that faithfully reflects the nature of the time series cannot be underestimated, as this directly impacts the accuracy and reliability of the predictions [1].

In this context, when a seasonal component is introduced, we evolve towards the SARIMA model, that is, the "Seasonal Autoregressive Integrated Moving Average Model". This model extends the capabilities of ARIMA by explicitly incorporating seasonality, making it particularly suitable for data that exhibit trends and patterns repeating at regular intervals. Common examples of such time series include monthly sales that fluctuate according to the season or temperatures that vary with the seasons. By integrating a seasonal component, the SARIMA model can identify and model these periodic variations, thus enhancing the accuracy of predictions in contexts where seasonality plays a crucial role [6].

SARIMA models are defined by a set of parameters denoted as (p, d, q) x (P, D, Q, S), where:

- **p, d, q:** Represent the non-seasonal ARIMA model parameters, corresponding to autoregression, differentiation, and moving average, respectively. These parameters are fundamental for modeling the trends and patterns inherent in the time series, regardless of seasonality [7].
- **P, D, Q:** Are the seasonal equivalents of the previous parameters. These are used to capture and model seasonality in the data, allowing the SARIMA model to adjust and predict time series with clear seasonal patterns [7].
- **S:** Indicates the periodicity of the seasonality. For example, S would be 12 for monthly data showing annual seasonality [7].

Seasonal Component:

- **SAR (Seasonal Autoregression):** Similar to the AR component, but focuses on seasonal dependency relationships. For example, in a monthly time series with annual

seasonality, the model might use the values of each December to predict the values of December in the following year [6].

- **Seasonal Differencing (D):** This is an extension of the integration (I) component. Instead of differentiating consecutive terms, terms that are separated by a full seasonal interval are differentiated. For example, in monthly data with annual patterns, the value of each month would be subtracted from the same month in the previous year to eliminate seasonality [6].
- **SMA (Seasonal Moving Average):** Functions similarly to the MA component, but focuses on prediction errors that follow a seasonal pattern. This component helps adjust the model to the specific seasonal fluctuations observed in the data [6].

Success in applying a SARIMA model depends heavily on the correct identification of the nature and periodicity of the seasonality present in the data. In addition, it is crucial to adjust the model's parameters accurately to adequately reflect the observed patterns in the time series. This fine-tuning allows the SARIMA model to make more accurate and relevant predictions for data with seasonal characteristics.

2.8.3 SARIMAX

Understanding the SARIMA model and its ability to handle seasonality in time series, it becomes crucial to explore the SARIMAX model. This extension of SARIMA incorporates exogenous variables, as indicated by the "X" in SARIMAX. The model is especially valuable when the behavior of the time series is influenced not only by its own past trends and seasonal patterns but also by external factors. SARIMAX adopts the form (p, d, q) x (P, D, Q, S), similar to SARIMA, for its autoregressive, integration, and moving average components, both in seasonal and non-seasonal aspects. Its distinction lies in the ability to integrate one or more exogenous variables. These can range from economic indicators to weather conditions, significant events, or any other external element considered influential for the analyzed time series [22].

Consider, for example, modeling the monthly sales of a store. The SARIMAX model, beyond analyzing the seasonality and past trends of sales, can incorporate exogenous variables

such as marketing campaigns, holidays, or changes in the local economy. This inclusion allows the model not only to rely on historical sales patterns but also to consider how these external factors may impact the future. The integration of exogenous variables into SARIMAX transforms it into a more comprehensive and adaptable analytical tool, offering more accurate and detailed predictions than models that only take into account the internal information of the time series [22].

Success in using SARIMAX critically depends on the accurate selection of relevant exogenous variables and a deep understanding of their interaction with the main time series. An inappropriate choice or a misinterpretation of these variables can result in incorrect conclusions or inaccurate predictions.

2.8.4 LSTM

LSTM Networks (Long Short-Term Memory) represent a significant advancement in the field of deep learning, demonstrating their effectiveness particularly in handling temporal sequences. These networks, an advanced variant of Recurrent Neural Networks (RNN), are designed to capture long-range dependencies, efficiently addressing the gradient vanishing problem posed by traditional RNNs. Their distinctive architecture allows LSTMs to store information over extended periods, facilitating the learning and retention of complex patterns in input data that unfold over time. This capability makes them particularly suitable for applications requiring detailed analysis of temporal sequences, such as natural language processing and time series prediction [24].

The architecture of an LSTM network is distinguished by its set of specialized "gates": the forget gate, the input gate, and the output gate. These gates allow for regulating the flow of information within the network, granting it the ability to determine which data are relevant to retain or discard during the processing of each element in a data sequence. This selection mechanism enables LSTMs to excel in complex tasks like natural language processing, time series prediction, and in advanced automatic translation systems [24].

Figure 2.7 [27], provides a schematic diagram of the internal structure of an LSTM cell,

illustrating how the different gates interact with the cell state (the network's long-term memory) and the hidden state (the short-term memory) to control and preserve relevant information over time.

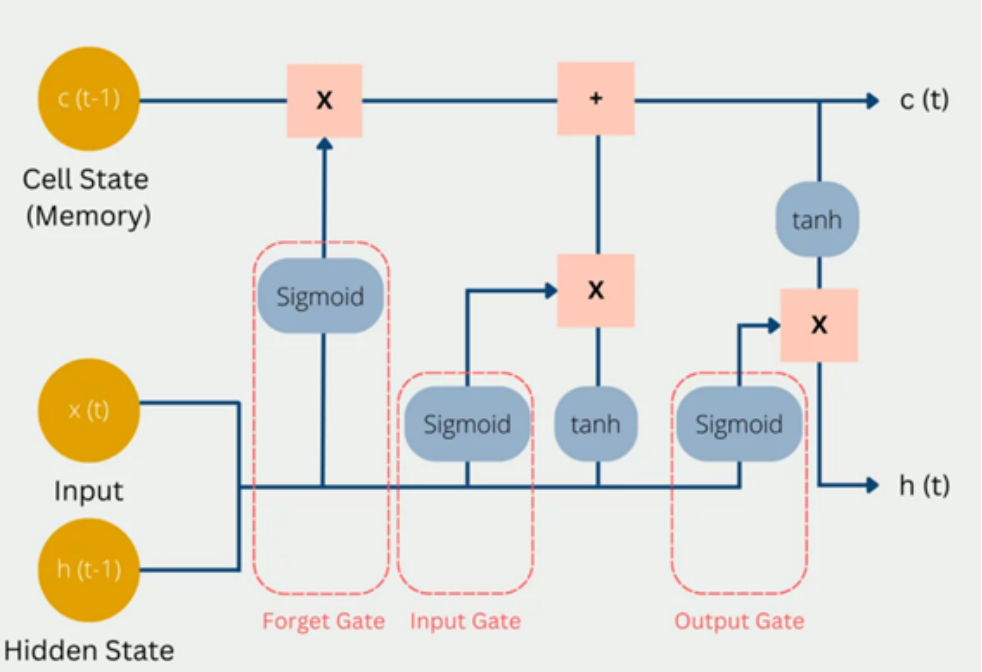


Figure 2.7: LSTM.

Cell State (Long-Term Memory)

The cell state in LSTM units represents a computational analogy of long-term memory. This component, visualized as a line extending across the cell, is the retention of information across extended intervals of time. Its main function is to store and preserve internal states for understanding the data sequence [23].

The management of information within the cell state is conducted through a dynamic system of gates. The forget gate, for example, has the capability to remove obsolete or irrelevant information from the cell state, ensuring that only significant information is retained. Concurrently, the input gate introduces new data into the cell state, integrating the current network input with previously accumulated memory. The ability of LSTM cells to overcome the gradient vanishing problem, inherent in conventional RNNs, allows for the effective capture of extended temporal dependencies [23].

Hidden State (Short-Term Memory)

In LSTM networks, the hidden state acts as a short-term memory mechanism, analogous to working memory in human cognition. This state not only transmits information to the next temporal step within the network but also contributes to the LSTM cell's output at each moment. Visually, in the diagram, this state is represented as a line that flows horizontally and then splits towards the output, carrying relevant and current information for the data sequence in process [24].

Unlike the cell state, designed to store information across extensive intervals of time, the hidden state is more ephemeral and dynamic. It is renewed at each temporal step, integrating information from the previous state and the current input, also influenced by the cell state. This constant updating allows the network to adapt flexibly and quickly to variations in input data [24].

LSTM Gates

Forget Gate: The forget gate is an essential component in the architecture of LSTMs, designed to control the retention or deletion of information within the cell state. It operates by applying a sigmoid activation function to the combination of the current input $x(t)$ and the previous hidden state $h(t-1)$. This function produces a vector with values between 0 and 1 for each element of the cell state $c(t-1)$. A value close to 0 indicates that the corresponding information should be "forgotten", that is, discarded from the cell state, while a value close to 1 means that the information should be retained [23].

Mathematically, the forget gate $f(t)$ is calculated as follows:

$$f(t) = \sigma(W_f \cdot [h(t-1), x(t)] + b_f) \tag{2.6}$$

where σ denotes the sigmoid function, W_f is the weight matrix for the forget gate b_f is the bias of the forget gate, and $[h(t-1), x(t)]$ represents the concatenation of the previous hidden state and the current input.
The result of this operation is multiplied element-wise with the previous cell state, directly

affecting the information that will be retained or eliminated in the next stage:

$$c(t)' = f(t) * c(t-1) \tag{2.7}$$

this mechanism of selective forgetting is crucial for the LSTMs' ability to handle long-term dependencies and prevent old and irrelevant information from saturating the network's memory, allowing more effective management of the gradient during training and reducing the risk of issues like gradient vanishing [23].

Input Gate: This gate determines which part of the new information received at each time step should be stored in the cell state. Like the forget gate, the input gate uses a sigmoid activation function to decide which information is relevant and thus should be preserved.

Simultaneously, a separate operation involves the hyperbolic tangent (tanh) activation function, which generates a vector of new candidate values that could be added to the cell state. This candidate vector is created considering the same input and the previous hidden state. The tanh function has an output range of -1 to 1, allowing this step to adjust the scale of the values to be added to the cell state, thus providing a new candidate for state update [23].

The input gate is calculated as follows:

$$i(t) = \sigma(W_i \cdot [h(t-1), x(t)] + b_i) \tag{2.8}$$

$$\tilde{C}(t) = \tanh(W_C \cdot [h(t-1), x(t)] + b_C) \tag{2.9}$$

where $i(t)$is the output of the input gate, W_i and W_C are the weight matrices corresponding to the input gate and the creation of the cell state candidates, respectively, while b_i and b_C are the corresponding biases. The output of the input gate $i(t)$ is multiplied with the candidate vecto $\tilde{C}(t)$ to determine what new information will be added to the cell state [23].

Finally, the cell state is updated by adding the product of the input gate and the candidate

vector to the result of the forget gate:

$$c(t) = f(t) * c(t-1) + i(t) * \tilde{C}(t) \tag{2.10}$$

Output Gate: The output gate in LSTM cells plays a fundamental role in determining which part of the current cell state information will be used in the next hidden state and, therefore, in the cell's final output at that specific moment. This gate is responsible for filtering and transmitting only the relevant information from the cell's internal state to the outside, which is crucial for decision-making and predictions based on the sequence of data processed up to that point [24].

Just like the other gates in the LSTM, the output gate employs a sigmoid activation function to decide which components of the cell state should pass to the hidden state. This decision is based on both the current input and the previous hidden state. The output of the output gate is then combined with the cell state, which has been processed through a hyperbolic tangent (tanh) activation function, to produce the new hidden state. This additional step of applying the tanh function helps to regulate the values of the cell state, keeping them within a manageable range [24].

The operation of the output gate can be described mathematically as:

$$o(t) = \sigma(W_o \cdot [h(t-1), x(t)] + b_o) \tag{2.11}$$

$$h(t) = o(t) * \tanh(c(t)) \tag{2.12}$$

here, $o(t)$ represents the output of the output gate, W_o is the weight matrix associated with this gate, and b_o is the corresponding bias. The new hidden state $h(t)$is calculated by multiplying the output of the output gate by the cell state passed through the tanh function. This ensures that the hidden state reflects the most relevant and updated information from the cell state [23].

2.8.5 Holt-Winters

The exponential smoothing method, fundamental in time series analysis, was initially developed by Charles C. Holt in 1957. Holt introduced an innovative approach for forecasting time series data, focusing primarily on adapting to level changes. In 1958, Holt evolved his model to incorporate data trends, allowing for a more dynamic forecast adapted to temporal variations. Subsequently, in 1960, Peter Winters extended Holt's model to include seasonality, a crucial component in many time series, especially those related to economic, financial, and climatological data. This extension enabled the method to address seasonal patterns, in addition to trends and levels [26].

As a result of these contributions, the method known as "Holt-Winters" emerged, an integrated approach that combines level, trend, and seasonality in the analysis and forecast of time series. This method has proven to be particularly effective in situations where data exhibit clear trends and seasonal patterns, being widely adopted in various fields for short and medium-term prediction [26].

In the context of exponential smoothing, the indices that play a fundamental role in the modeling and forecasting of time series are:

- **Smoothing Index (α):**This parameter controls the level of smoothing of the most recent data, giving them more or less weight in the prediction. In simple exponential smoothing, only this index is used, making it suitable for time series without clear trends or seasonal patterns [25].
- **Trend Linear Index (β):** This index is introduced in double exponential smoothing to handle data with trends. Here, in addition to α, β is used to adjust and predict the trend of the data over time [25].
- **Seasonal Factor Index (γ):** In triple exponential smoothing, also known as the Holt-Winters method, this third index is incorporated. γ allows the model to address seasonality [25].

The Holt-Winters method adapts to different seasonal patterns through two main approaches: additive and multiplicative [6]. The choice between these two depends on the nature of the data's seasonality:

- **Multiplicative Effect:** This approach is used when the seasonal pattern varies in proportion to the time series values. In other words, the amplitude of the seasonal pattern increases or decreases based on the data level. This is typical in situations where seasonal effects intensify as the series values increase, making it suitable for time series where seasonal patterns are proportionally more pronounced at higher levels [26].

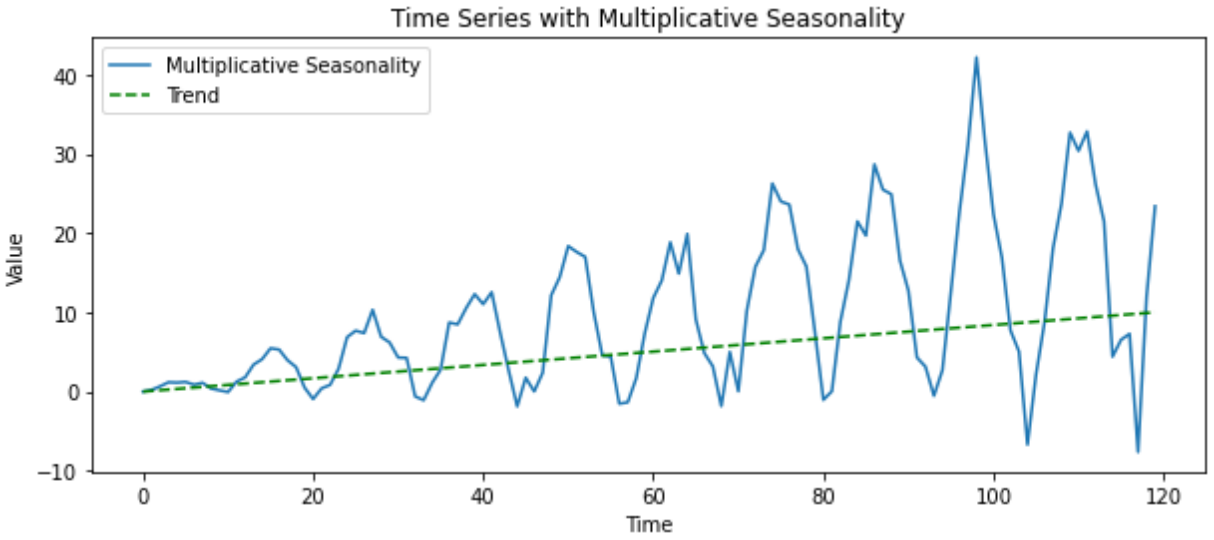


Figure 2.8: Multiplicative Seasonal Pattern.

- **Additive Effect:** In contrast, the additive approach is applied when the series' seasonal pattern is constant, regardless of the data level. This means that the magnitude of the seasonal effect does not change even if the time series increases or decreases in value. This method is appropriate for time series where seasonal patterns are consistent and do not vary in proportion to the series values [26].

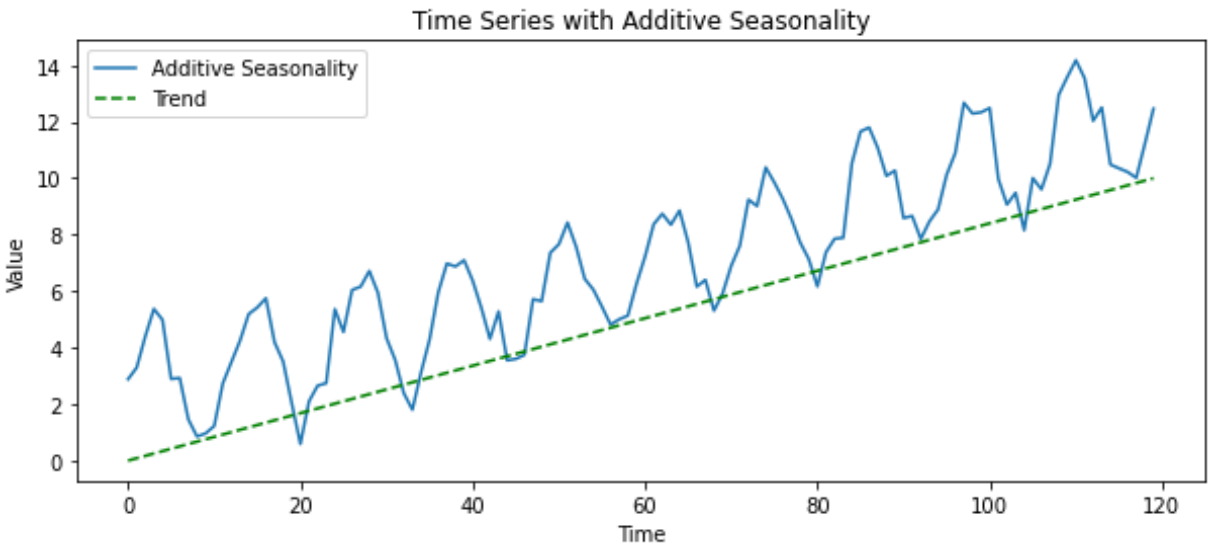


Figure 2.9: Additive Seasonal Pattern.

Formulas Used for the Multiplicative Model

Estimated Level

The estimated level or exponentially smoothed series, denoted as L_t. This formula is for updating the series level at time t, and is calculated as follows [26]:

$$L_t = \alpha \left(\frac{Y_t}{S_{t-s}} \right) + (1 - \alpha)(L_{t-1} + T_{t-1}) \tag{2.13}$$

where:

- Y_t represents the observed value at time t.
- S_{t-s} is the seasonal component corresponding to time $t - s$, where $s$is the length of the season.
- L_{t-1} is the estimated level at time $t - 1$.
- T_{t-1} is the estimated trend at time $t - 1$.
- α is the smoothing coefficient for the level, a parameter chosen between 0 and 1.

This equation adjusts the current estimated level based on the most recent observation Y_t, correcting for seasonality, and combining this value with the sum of the estimated level

and trend from the previous period, weighted by $1 - \alpha$. The weighting reflects the relative contribution of the most recent observed value compared to the projection based on past level and trend. In this way, L_t provides a basis for future projections, continuously adjusting as new data becomes available [26].

Trend Estimation

The trend estimation is calculated using the formula [26]:

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1} \tag{2.14}$$

where:

- T_t: Estimated trend at time t.
- T_{t-1}: Estimated trend at time $t - 1$.
- L_t: Estimated level at time t.
- L_{t-1}:Estimated level at time $t - 1$.
- β: Smoothing coefficient for the trend.

This formula adjusts the current trend T_t by taking into account the change between the last two estimated levels $(L_t - L_{t-1})$ and the value of the previous trend T_{t-1}. The coefficient β, which varies between 0 and 1, determines the relative weighting between the difference in levels and the previous trend, allowing the model to adapt to changes in the time series trend over time. A higher value of β gives more weight to the recent change in levels, while a lower value places more emphasis on the historical trend [26].

The estimated trend T_t allows understanding and projecting the long-term direction of the time series, which is especially useful in strategic planning and decision-making based on historical trends [25].

Seasonal Component

The seasonal component in the Holt-Winters method is designed to adjust to seasonal

patterns in time series. The formula for estimating seasonality S_t is as follows [26]:

$$S_t = \gamma \left(\frac{Y_t}{L_t} \right) + (1 - \gamma) S_{t-s} \tag{2.15}$$

where:

- S_t: Seasonal component at time t.
- Y_t: Observed value at time t.
- L_t: Estimated level at time t.
- S_{t-s}: Seasonal component of the same period in the last season.
- s: Length of the seasonal cycle.
- γ: Smoothing coefficient for seasonality.

In this equation, the seasonal component is updated by weighting the observed value Y_t adjusted by the estimated level L_t, providing a relative measure of seasonality at the current time. The term $(1 - \gamma)S_{t-s}$ represents the influence of the seasonality from the same period in the last season, ensuring continuity and consistency of seasonal patterns year after year. Like with the other components of the Holt-Winters method, the coefficient γ is selected between 0 and 1 and determines the relevance given to the most recent observation compared to historical seasonality [26].

The estimated seasonality S_t helps forecast periodic fluctuations and allows the Holt-Winters model to capture seasonal patterns that can be used to enhance the forecast accuracy in data with significant seasonal variations [25].

Forecasting

Forecasting future values in the Holt-Winters method is performed through the forecast formula for p future periods, which is expressed as [25]:

$$\hat{Y}_{t+p} = (L_t + pT_t) S_{t-s+p} \tag{2.16}$$

where:

- $\hat{Y}_{t+p}$: Forecasted value for p periods in the future.
- L_t: Estimated level at time t.
- T_t: Estimated trend at time t.
- S_{t-s+p}: Seasonal component adjusted for the forecast period.
- p: Number of periods in the future for the forecast.

The formula combines the current level and trend multiplied by the number of periods p in the future, with the seasonal component corresponding to the future period. This approach allows the Holt-Winters model to adjust forecasts to reflect not only the current trend and level but also the expected seasonality [26].

The use of pT_t allows the model to project how the trend will impact the forecasted value in each of the p future periods, while S_{t-s+p} adjusts this forecast to reflect the seasonality anticipated for that specific period. The Holt-Winters model's ability to integrate these three components – level, trend, and seasonality – makes it suitable for forecasting in contexts where seasonal patterns are a relevant factor [26].

Formulas Used for the Additive Model

Estimated Level

In the additive model of the Holt-Winters method, the estimated level or exponentially smoothed series are calculated with the following formula [26]:

$$L_t = \alpha(Y_t - S_{t-p}) + (1 - \alpha)(L_{t-1} + T_{t-1}) \tag{2.17}$$

where:

- L_t is the estimated level at time t.
- α is the smoothing coefficient for the level.
- Y_t represents the observed value at time t.

- S_{t-p} is the seasonal component in the corresponding period of the previous season.
- L_{t-1} is the estimated level at time $t-1$.
- T_{t-1} is the estimated trend at time $t-1$.

In this version of the model, the update of the estimated level is made by adjusting the observed value Y_t for the seasonal component of the same period in the last season S_{t-p}. This means that the seasonal effect is directly subtracted from the observed value before applying exponential smoothing. The term $(1-\alpha)(L_{t-1}+T_{t-1})$ combines the level and trend from the previous period, allowing the model to gradually adjust to long-term changes in the time series [26].

The key distinction of the additive model is that it considers seasonality to have a constant effect, rather than one proportional to the magnitude of the series, as in the multiplicative model. This makes it more suitable for time series where seasonal patterns remain consistent over time, regardless of the trend or level of the series [25].

Trend Estimation

The trend estimation in the Holt-Winters method, whether in the additive or multiplicative model, is calculated using the following equation [25]:

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \tag{2.18}$$

where:

- T_t: is the estimated trend at time t.
- β: is the smoothing coefficient for the trend.
- L_t: is the estimated level at time t.
- L_{t-1}: is the estimated level at time $t-1$.

This formula allows updating the trend for the current period by taking the difference between the last two estimated levels, weighted by the coefficient β, and adjusting it with the

estimated trend from the previous period, weighted by $1-\beta$.This balances the influence of the recent trend with the historical trend, allowing the trend estimation to evolve smoothly over time [26].

The smoothing coefficient β plays a significant role, as a higher value will assign more weight to recent changes in the level, while a lower value will give more importance to the previous trend, reflecting the inertia or continuity of the trend over time [25].

Seasonal Component

The estimation of seasonality within the additive model of the Holt-Winters method is performed with the following mathematical expression [26]:

$$S_t = \gamma(Y_t - L_t) + (1-\gamma)S_{t-p} \tag{2.19}$$

where:

- S_t: Estimated seasonal component at time t.
- γ: Smoothing coefficient for seasonality.
- Y_t: Observed value at time t.
- L_t: Estimated level at time t.
- S_{t-p}: Seasonal component from the same period in the previous season.
- p: Length of the seasonal period.

The update of the seasonal component is performed by subtracting the estimated level L_t from the observed value Y_t, and then applying the smoothing coefficient γ to this result. This is added to the adjusted seasonal component from the corresponding period of the previous season, weighted by $1-\gamma$. This method ensures that seasonality is updated with the most recent information while maintaining continuity with the historical seasonal pattern [26].

The application of γ and $1-\gamma$ allows the model to balance between the influence of the new observation and the previous seasonality.

Forecasting

The forecast for future periods in the additive Holt-Winters model is calculated using the sum of the estimated components of level, trend, and seasonality. The formula is as follows [25]:

$$\hat{Y}_t = L_{t-1} + T_{t-1} + S_{t-p} \tag{2.20}$$

where:

- $\hat{Y}_t$: is the forecasted value for period t.
- L_{t-1}: is the estimated level from the previous period.
- T_{t-1}: is the estimated trend from the previous period.
- S_{t-p}: is the seasonality component from the corresponding period of the previous season.
- p: is the length of the seasonal cycle.

This equation is based on the premise that the future forecast is a direct sum of the last estimated level, the last estimated trend, and the adjusted seasonality component from the same period of the previous season. In contrast to the multiplicative model, where the seasonal component is multiplied, in the additive model, seasonality is added, reflecting a constant influence of seasonality on the time series, regardless of the level or trend [25].

This methodology is particularly useful for time series where seasonality does not vary proportionally with the trend or level but remains constant over time. By applying this formula, the Holt-Winters model can provide forecasts that accurately reflect fixed seasonal patterns and underlying trends in the data [25].

2.9 Summary

This study focuses on the implementation of time series analysis to assess the fluctuations in Bitcoin prices and Amazon stock. Through a recursive prediction strategy, their behavior patterns will be examined to derive accurate inferences about their future trends. To achieve this, advanced analytical models, including Prophet, SARIMA, SARIMAX, LSTM, and Holt-Winters, will be employed with the aim of identifying which method offers superior adaptability against the dynamics of the time series. This approach will allow for the optimization of predictions by more effectively understanding the complexities inherent in the price movements of these two significant financial variables.

Chapter 3

Methodology

3.1 Phases of the Problem

3.1.1 Description of the Problem

Investing in cryptocurrencies or stocks requires meticulous and well-informed analysis. Investment involves significant risks, and making poorly founded decisions can result in the loss of invested capital. However, accessing and understanding the information necessary to make informed decisions represents a considerable challenge, especially for those without previous experience in the financial domain.

The overwhelming volume of data, analysis, and opinions available makes it difficult for a novice to discern valuable and reliable information from that which is misleading, irrelevant, or fraudulent. This excess of information, often contradictory, can be paralyzing, leaving beginner investors in a state of uncertainty about how and where to begin. Determining a clear starting point in this vast sea of data becomes a significant obstacle, creating a barrier to effectively and safely entering the world of investments.

The complexity of investing is intensified in the case of cryptocurrencies. This market is characterized by its rapid evolution, with the constant emergence of new cryptocurrencies. Many of these emerging digital currencies present a high degree of uncertainty regarding their viability and long-term profitability potential. For the average investor, this dynamism of the cryptocurrency market represents an even greater challenge: facing

an environment saturated with misinformation and potential scams. Investment methods in cryptocurrencies are not always reliable, and the amount of information to process can be overwhelming for those seeking to gain a basic understanding of the market. Moreover, the ability to make accurate predictions in this sector is particularly complex and requires extensive and in-depth study. Even so, due to the unpredictable nature of the market, there is no absolute guarantee that such predictions will be effective or lead to successful investment decisions.

In summary, for the average individual looking to enter the world of investments, the path is fraught with uncertainties and challenges, making deep preparation and understanding essential before making any investment.

3.1.2 Analysis of the Problem

Initial Challenges for Beginners in Investments

For an average person with an interest in the field of investments, a significant barrier often encountered is the lack of prior knowledge. In their quest for information, the most common recourse is to accessible online sources, such as Google or YouTube. However, this initial approach presents two main problems. Firstly, the oversaturation of information available on these platforms can be overwhelming, making it difficult to identify a clear and reliable starting point. Given the massive volume of content, filtering out relevant and suitable information for beginners becomes a challenging task..

Figure 3.1 represents a simple Google search on how to invest in Bitcoin. The number of results is staggering: approximately 892 million. This screenshot encapsulates the colossal challenge individuals face when trying to navigate the vastness of the digital space to inform themselves about cryptocurrency investments. This volume of information, while reflecting a rich ecosystem of knowledge and resources, also presents a labyrinth of potential detours and distractions. For the novice investor, discerning which of these nearly 900 million results offers accurate, ethical, and valuable advice is a daunting task. This

information overload can not only lead to analysis paralysis but also increases the risk of encountering inaccurate advice or, worse, fraudulent schemes.

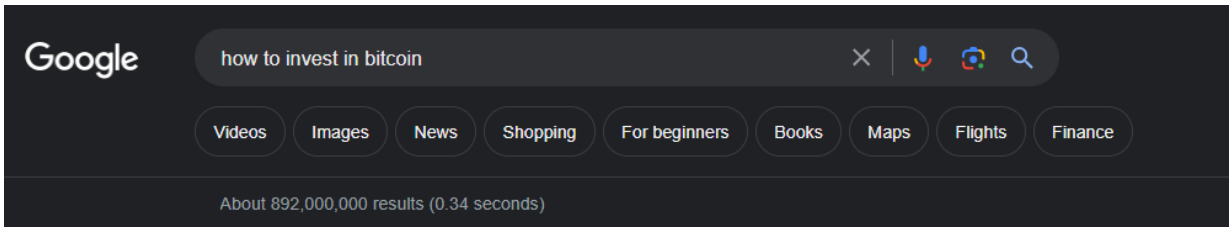


Figure 3.1: The amount of information available with just a general search google.com.

Secondly, with the growing interest in investment topics, social media and online platforms tend to actively promote websites, videos, and other investment-related resources. However, many of these resources are geared towards investors with intermediate or advanced experience. For someone just starting out, this level of information can be particularly overwhelming and unsuitable, as it often presupposes a level of knowledge and experience that the novice simply does not possess. Thus, aspiring investors find themselves at a disadvantage, where the abundance of online information, far from being a help, can become a significant obstacle to acquiring a fundamental and practical understanding of investments.

Figure 3.2 [28], shows the user interface of the Binance trading platform, one of the world's most prominent cryptocurrency exchanges. At first glance, the complexity of the environment is evident: real-time price charts, order books, and a variety of options for executing buy and sell trades. While for an investor with intermediate or advanced experience, this screen may represent a window to market opportunities and a tool for implementing sophisticated trading strategies, for a beginner it can be an overwhelming and confusing scenario.

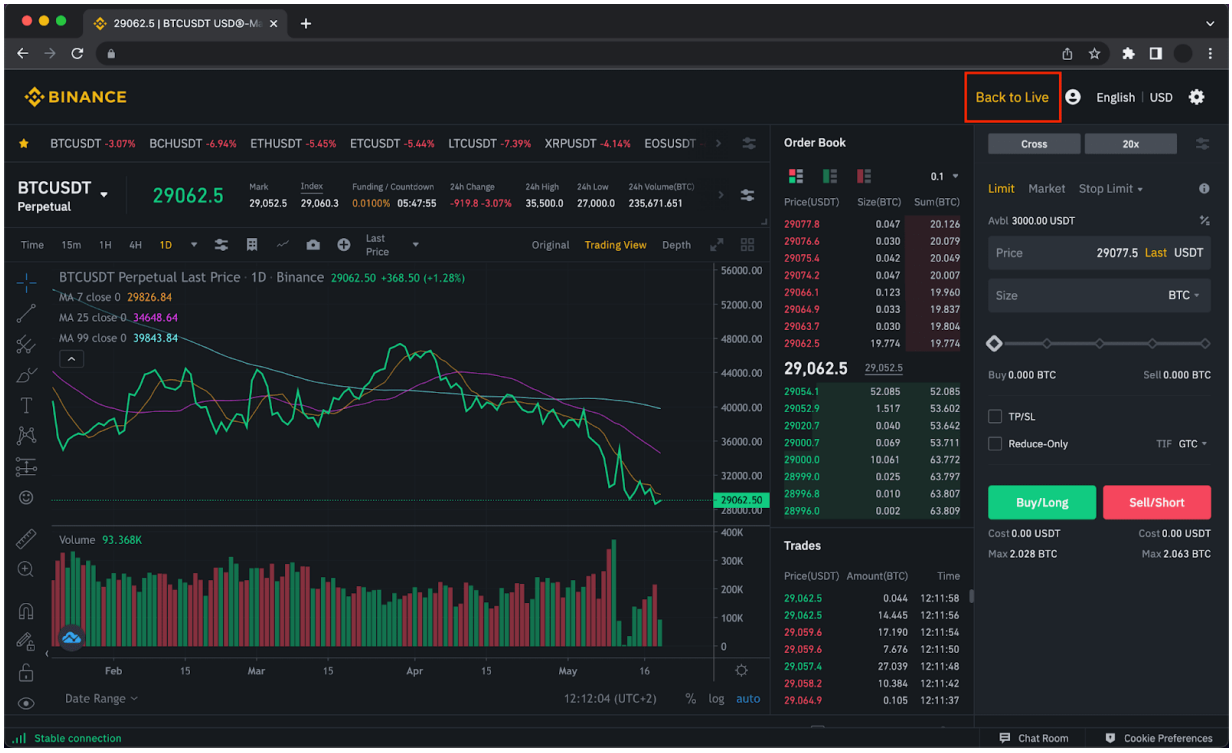


Figure 3.2: Binance platform market.

The amount of information and tools available on a trading platform like this requires considerable knowledge to be navigated and used effectively. Advanced charts, moving averages, and volume metrics are essential for conducting technical analysis, but understanding their meaning and knowing how to act accordingly is a skill that develops over time and experience.

For an individual taking their first steps in the world of cryptocurrency investments, interfaces like the one seen in the image can represent a significant barrier. Data overload can lead to hasty decision-making or, at the other extreme, inaction for fear of making mistakes.

The Need for Professional Advice in Investments

The existence of the stockbroker profession is a testament to the inherent complexity of the investment world. These professionals, trained and with the relevant education, offer legal and specialized guidance to those looking to invest. However, their services entail an additional cost, which can be discouraging for those considering entering the invest-

ment world for the first time. The absence of proper guidance poses a significant dilemma: without expert advice, it is difficult for a beginner to determine whether their investment of time and money will be profitable or, conversely, end in losses. This risk often inclines novice investors towards more stable and secure investment options, such as bank savings accounts that offer annual interest. These institutions are generally perceived as safe and reliable for money management.

Risk of Scams in the Investment Environment

As mentioned before, seeking information about investments leads interested parties into a vast universe of data. However, there's an additional risk that accompanies this search: the exposure to potential scams. Content personalization on social media and other online platforms, like YouTube, often leads to recommendations of videos and groups related to investments. But the reliability of these resources is uncertain. During research in this field, a trend is observed: the frequent appearance of promotional videos where individuals claim to have foolproof methods for making significant profits. These claims are common strategies to lure people into Telegram or Discord groups. However, these groups may be the setting for various scams. A common tactic observed is the request for payment of a monthly subscription to access a "private group", where supposedly the "real" buy and sell predictions are shared. The validity and effectiveness of these predictions are highly questionable, and often, participants end up not only risking their initial investment but also incurring additional costs for the subscription.

Furthermore, participation in these groups can expose individuals to other risks. For example, unauthorized access to personal and financial information, making them vulnerable to other types of fraud and cyber attacks. This aspect of online investments represents a significant danger, especially for those who lack the experience to distinguish between legitimate advice and fraudulent schemes.

Figure 3.3 displays a variety of content appearing on YouTube when searching how to invest in Bitcoin. The titles suggest investment guides and strategies, and the thumbnails often

show figures of potential earnings and promises of financial success. While educational content can be extremely valuable for beginners, the platform is also filled with misleading offers that prey on the inexperience and high expectations of new investors. Videos promising high returns for minimal investments, or those showcasing portfolio balances with exorbitant figures, can be signs of unethical marketing tactics or even direct scams. The challenge for novice users lies in their ability to discern between legitimate educators and malicious actors. There is no simple method to verify the truthfulness of the claims made in these videos, increasing the likelihood of well-intentioned individuals falling into costly traps.

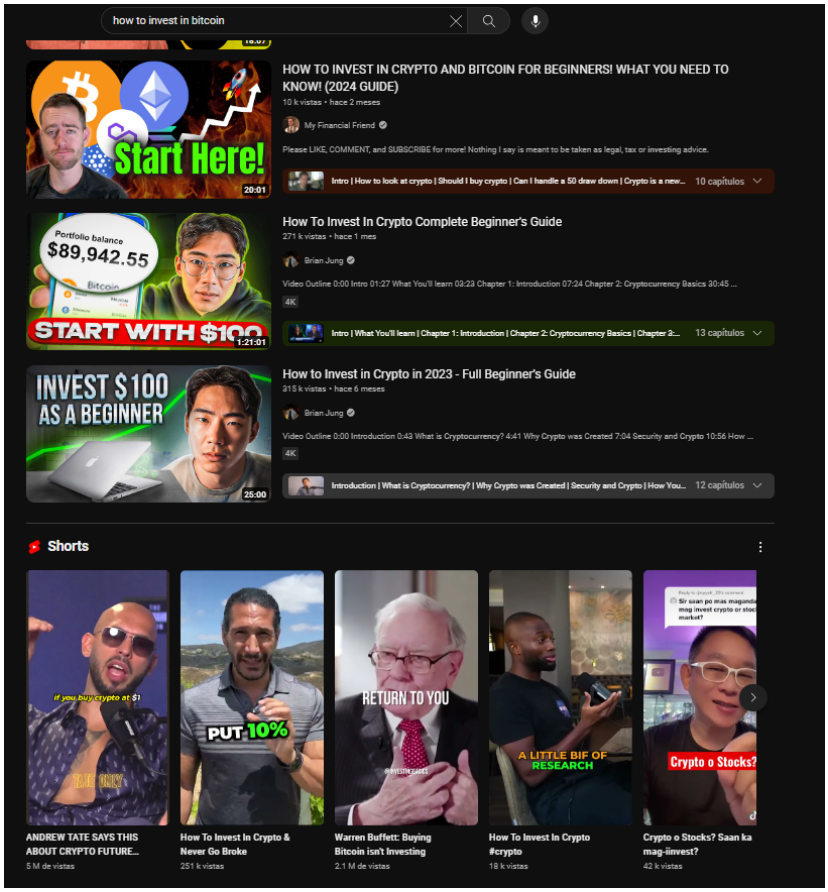


Figure 3.3: Information available on YouTube youtube.com.

The proliferation of self-proclaimed advisors and financial "gurus" on platforms like YouTube raises serious concerns. Many of these content creators use the promise of quick riches to attract subscribers to paid services, such as trading signal groups on messaging platforms, where the legitimacy and effectiveness of the recommendations provided are questionable. Some might even be involved in "pump and dump" schemes, where they artificially inflate the value of lesser-known assets to then sell them at a higher price at the expense of their followers.

Additional Factor of Uncertainty in Cryptocurrencies

In the realm of cryptocurrencies, investors face an additional risk: the constant emergence of new digital currencies. These new coins often enter the market touted as the next big revolution, with promises of being the "new Bitcoin". These appealing claims, devoid of real guarantees, lure investors seeking quick and lucrative opportunities.

However, the reality behind many of these new cryptocurrencies is concerning. It is common for these coins to not achieve the promised success or, in some cases, disappear from the market altogether. This type of failure leads investors to face significant losses. The volatility and lack of regulation in the cryptocurrency market exacerbate this issue, making investment in emerging cryptocurrencies especially risky. This scenario underscores the importance of diligence and skepticism when considering investing in new cryptocurrencies. The absence of a stable regulatory framework and the propensity for exaggerated promises make investing in cryptocurrencies a particularly unstable ground prone to speculation.

This work unfolds in two main phases. The first involves evaluating the models described above to determine their ability to make meaningful predictions in the specific areas mentioned. The second phase focuses on analyzing the profitability of the most effective methods identified in the first phase, aiming to determine their potential to generate economic benefits.

3.2 Phase 1, Univariable Algorithm Design

In this study, a comparative analysis was conducted using the previously mentioned models, evaluating their predictive capabilities in real-world scenarios. For the experiment, two main groups were established: univariable and exogenous. The goal was to determine if the models improve their performance with the inclusion of more or fewer variables. This methodology was applied to both stocks and cryptocurrencies.

For the stock analysis, the models were evaluated in a univariable manner and with exogenous variables. In the univariable scenario, the Holt-Winter, LSTM, SARIMA, and Prophet models were used. Due to the absence of exogenous variables, the SARIMAX model was not included. In the scenario with exogenous variables, the mentioned models were applied, adding SARIMAX in place of SARIMA. This resulted in a total of 8 different scenarios to evaluate the effectiveness of the models in each case.

The procedure for cryptocurrencies was identical, implying a minimum of 16 testing environments in total, not including the internal variations of some models. For the cryptocurrency analysis, Bitcoin was selected due to its stability and prominence in the market over the years, making it a good indicator of additional security. As for stocks, Amazon's shares were chosen due to their relevance in the market.

3.2.1 DataFrame, Stocks

In the initial phase of the study, the dataset with which all models will work was downloaded. It is important that it be the same for all to evaluate their performance under equal conditions.

Importing Libraries and Initial Setup

The importation of libraries in Python was proceeded with. The libraries used were:

- **NumPy and Pandas:** NumPy provides support for arrays and matrices, along with a collection of mathematical functions to operate with these objects, while Pandas

offers data structures and tools for the efficient manipulation of datasets, including DataFrames, which serve for the handling of time series and tabular data.

- **yfinance:** This library was used to obtain real-time financial data from Yahoo Finance. It is a tool that allows access to a vast amount of financial information, including historical stock prices, transaction volume, dividend data, among others.
- **Matplotlib.pyplot:** Imported for data visualization. This tool was mainly used to visualize the differences between the predicted data and the real data from the dataset.
- **Warnings:** Used to control the display of warnings in Python. This is particularly useful to avoid the output of warning messages that can be irrelevant or confusing during code execution, thus allowing a clearer presentation of the analysis results.

Downloading and Initial Setup of Financial Data with yfinance

Once the working environment with the relevant libraries was established, the data collection stage was proceeded with. Using yfinance, the stock information of Amazon was downloaded.

- **Selection of the Period and Frequency of the Data:** A time range was specified from May 13, 2022, to May 14, 2023, providing a complete year of data. Additionally, a daily interval (1d) was chosen, indicating that data are collected for each trading day within this period.
- **Data Adjustments:** This adjustment ensures that the stock price data are adjusted for dividends and stock splits, thus offering a more accurate and realistic view of the market value of the stocks over time.

Processing and Cleaning of the Downloaded Data

After downloading Amazon's stock data, the next step was the processing and cleaning of these data to prepare them for analysis. The key steps in this process were:

1. **Creation and Modification of the Data Structure:**

    - A copy of the downloaded data was made to ensure the integrity of the original data.
    - A "Price" column was added, which is a copy of the "Close" column. This column represents the adjusted closing price of Amazon's stocks, a crucial indicator in financial analysis.
    - The date was set as the index of the DataFrame, which facilitates later operations, such as reordering or filtering based on dates.

2. **Cleaning and Organizing the Data:**

    - Unnecessary columns were removed. This action simplifies the DataFrame, keeping only the essential data for the proposed analysis, in this case, the adjusted closing price of the stocks.
    - The frequency of the DataFrame was adjusted to business days, a common practice in the analysis of financial time series, as the stock market does not operate on weekends or holidays.
    - The "forward fill" method was used to fill in missing values. This is particularly useful in time series where it is reasonable to assume that the value of a non-working day will be similar to that of the last business day.

3. **Exporting Processed Data:**

    - Finally, the processed and cleaned data were saved in a CSV file. This step allows documenting the work done and facilitating access to the data in later stages of the analysis or for use in other environments or applications.

Data Visualization and Exploratory Analysis

Once the data were processed and cleaned, the next step in the study was data visualization. In this case, the focus was on visualizing the price of Amazon's stocks over time.

- Using matplotlib.pyplot, a line chart was generated to visualize the evolution of Amazon's stock price over time. This provides a direct graphical representation of how the stock price has fluctuated during the studied period.
- The chart was configured with an appropriate size for clear and detailed visualization, and a legend was included to facilitate the identification of the represented data.

3.2.2 DataFrame, Cryptocurrencies

Importing Libraries and Initial Setup

1. **Binance Client:**
    - **from binance.client import Client**, the Binance Client provides the necessary methods to interact with the Binance API. Through this connection, financial data are obtained, in this case, for the cryptocurrency Bitcoin.
    - The initial setup with API keys is performed here, ensuring the necessary authentication and authorization to access Binance's data.
2. **Matplotlib.pyplot:**
    - **import matplotlib.pyplot as plt**, Matplotlib is a Python data visualization library. It is primarily used to create static, animated, and interactive charts and diagrams.
3. **NumPy and Pandas:**
    - **import numpy as np**, NumPy is a library for scientific computing in Python, providing array objects and tools to work with them.
    - **import pandas as pd**, Pandas is a library providing high-performance, easy-to-use data structures, and data analysis tools.

Downloading and Initial Setup of Financial Data with Binance

Downloading Historical Data

- The Binance library is used to obtain historical prices of Bitcoin in USD.
- Several parameters are required:
  - **symbol**
  - **time interval**
  - **time period**

Initial Data Transformation

The data obtained from Binance are returned in the form of a list of lists, where each sublist contains data for a specific interval. These raw data are converted into a Pandas DataFrame to facilitate manipulation and analysis. The initial DataFrame contains several columns with different pieces of information.

Initial Structuring and Cleaning

The obtained DataFrame contains multiple columns. For this analysis, the relevant columns are the timestamp and the closing price.

- A new DataFrame is created, extracting the timestamp column and the closing price column.

Index and Data Type Configuration

In the DataFrame, the date and time column is set as the index. This is important for time series operations, as it allows easier handling of data based on timestamps. Furthermore, it ensures that the price data are of floating type to allow subsequent numerical calculations.

Data Visualization

Pandas' plot method, which is integrated with Matplotlib, is used to create a line chart of Bitcoin's closing prices. This line chart represents the closing prices of Bitcoin over time, with the date and time on the X-axis and the price on the Y-axis.

3.2.3 Holt-Winter

Importing Libraries and Initial Setup

For the development of this project, the libraries Matplotlib, Numpy, and Pandas have been used, which were previously described.

Definition of Key Functions

In the analysis process, the following essential functions were defined:

- **load_data:** This function handles the efficient loading of time series data from CSV files. It uses the Pandas library to read the file, assigning the "Date" column as the index.
- **split_data:** A function designed to split the dataset into two segments: training and testing. It establishes the proportion of the dataset allocated to training, allowing a balance between model learning and accuracy validation.
- **plot_forecast:** A tool used for the effective visualization of the predictive model's results. This function plots the real data from the test set parallel to the model's predictions, facilitating a direct and effective visual comparison of the model's performance.

Application of Exponential Smoothing in Time Series Modeling

In this section, the Holt-Winters method was implemented to adjust the model to the trends and seasonal patterns identified in the training dataset part. This method allows for

considering both trends and seasonalities that can be of an additive or multiplicative nature. This duality in the model's characteristics generates a total of four possible combinations:

1. Additive trend and additive seasonality.
2. Additive trend and multiplicative seasonality.
3. Multiplicative trend and additive seasonality.
4. Multiplicative trend and multiplicative seasonality.

Each of these combinations was exhaustively evaluated to determine the model's performance in different contexts and settings. This approach allows us not only to better understand the intrinsic characteristics of the analyzed time series but also to optimize the accuracy and efficacy of the model under various data conditions.

Parameter Configuration in the Exponential Smoothing Model

The effectiveness of the exponential smoothing model largely depends on the proper setting of three fundamental parameters: trend, seasonality, and seasonal periods.

- **Number of Seasonal Periods:** This parameter is crucial for understanding the frequency with which the seasonal pattern repeats in the data. The choice of the number of seasonal periods is based on the size of the database. For a database of size X, up to X/2 seasonal periods are explored. This approach allows a broad exploration of possible seasonal configurations without exceeding half the total size of the database, maintaining a balance between detecting seasonal patterns and preventing overfitting..
- **Tests with Different Configurations of Seasonal Periods:** For each of the up to X/2 possible values of the seasonal period, the model is configured and evaluated in the four combinations of trend and seasonality (additive or multiplicative), mentioned above. This approach ensures the model's performance is examined across a wide range of possible scenarios..

Forecasting and Application of the Time Series Model

Once the training process of the model with the training dataset is completed, the forecasting phase proceeds, applying the model to the test set.

- **Application to Test Data:** The trained model is employed to make forecasts on the test data set. This step represents a test simulating a real prediction situation, where the model must operate with data not used in its training..
- **Visual Comparison with Real Data:** The effectiveness of these forecasts can be more clearly appreciated through visual comparison. Charts are prepared where the model's predictions are superimposed with the real values from the test set. This comparative visualization not only facilitates the immediate evaluation of the model's performance but also provides a deeper understanding of the areas where the model performs well and those where it might need adjustments or improvements.

Summary and Evaluation of the Holt-Winters Model

The Holt-Winters model was subjected to thorough analysis and testing, considering the four possible combinations of trend and seasonality for each of the potential values of the seasonal periods. The procedure and variants applied are detailed as follows:

- **Tests with Combinations of Trend and Seasonality:** For each possible value of the seasonal periods, tests were conducted with the four combinations of trend (additive or multiplicative) and seasonality (additive or multiplicative). Then, for a database size X, the Holt-Winters model was evaluated in 4(X/2) different prediction outcomes. This allowed a broad and detailed comparison with the real values, ensuring a comprehensive evaluation of the model in different scenarios.
- **Introduction of Data Scaling in Holt-Winters:** In addition, a variant of the Holt-Winters model was carried out, incorporating data scaling into the procedure. This adaptation allowed generating an additional set of simulations, doubling the number of scenarios tested. With this modification, 8(X/2) different simulations us-

ing the Holt-Winters model were conducted..

3.2.4 LSTM

Library Import

The code implements the Pandas and Matplotlib libraries, which have already been described previously. Additionally, the following are incorporated:

- **Keras:** A high-level API for building and training neural network models. In this code, Keras is used to design and train the LSTM model (https://keras.io/api/).
- **Scikit-learn:** Although its application in this context is not directly related to neural network modeling, the Scikit-learn library contributes through the MinMaxScaler tool. This tool is used for data normalization (https://scikit-learn.org/stable/).

Data Preparation and Handling

The methods used to load, split, and scale the data before feeding it into the LSTM model.

- **Data Loading:** The function is responsible for loading data from a CSV file using pandas. The index is set to the "Date" column, and it is converted into a datetime object to facilitate temporal analysis.
- **Data Splitting:** Once the data is loaded, the function divides it into training and testing sets. The training set is used to teach the model the dependencies in the data, while the testing set is used to evaluate its performance.
- **Data Scaling:** The function implements Scikit-learn's MinMaxScaler to scale the data so that all values fall within a defined range. Since input values that are too large can create difficulties in neural network learning, causing instabilities or slow convergence.

Generation of Time Series and Data Preparation for LSTM

This section of the code focuses on two key aspects: the generation of time series and the proper preparation of data for use in the LSTM model.

Time Series Generation with TimeseriesGenerator: Keras'TimeseriesGenerator is implemented. This tool adapts the scaled data into a format compatible with the LSTM model. The TimeseriesGenerator takes the input data and reorganizes it into a set of sequences. Each sequence is composed of groups of consecutive data points, providing the necessary structure for the LSTM model to learn and understand the temporal dependencies between these points.

Generator Configuration: Parameters such as the number of inputs are defined, which determines how many previous time steps will be used to predict the next time point. The choice of this parameter dictates the amount of historical information the model analyzes in each iteration. Additionally, the batch size is set.

Construction and Configuration of the LSTM Model

- **LSTM Model Architecture:** We use Keras' Sequential class to construct the model, allowing us to stack the layers in a sequential and organized manner. The architecture includes several LSTM layers, each carefully configured to fulfill a specific function within the model.

- **LSTM Layer Configuration and Regularization:** Each LSTM layer in the model is configured with a specific number of units, defining the dimension of the output space. Additionally, L1 and L2 regularizers are applied to control overfitting. Regularization helps improve the model's generalization by penalizing large weights in the network.

- **Dropout and LeakyReLU Layers:** To further strengthen the model against overfitting, Dropout layers are integrated after each LSTM layer. These Dropout layers randomly "turn off" a set of neurons during training, contributing to the model's robustness. Moreover, the LeakyReLU activation function is implemented, known

for its effectiveness in allowing the model to learn complex non-linear relationships in the data.

- **Model Compilation:** Finally, the model is compiled with the Nadam optimizer and a mean squared error (MSE) loss function. The Nadam optimizer is a variant of the Adam optimizer, known for its effectiveness in training neural networks, and MSE is a common measure for evaluating performance in regression tasks.

Analysis of LSTM Model Hyperparameters and Search for Optimal Architecture

The process of selecting and tuning hyperparameters in an LSTM model is a complex task that significantly impacts the model's performance. The task involves defining and experimenting with several key hyperparameters of the LSTM architecture, whose possible combinations and adjustments are vast and must be meticulously tailored to the particularities of the time series under study. These hyperparameters include:

1. **Layers and Neurons**

    - **Number of LSTM Layers:** The number of layers in an LSTM model determines the network's depth. A deeper network, with multiple LSTM layers, has a greater capacity to learn complex patterns and long-term relationships in the data. However, increasing the number of layers also poses certain challenges. A deeper model may be more difficult to train and more prone to overfitting, especially if the amount of data available for training is limited. Additionally, more complex models require more computational resources and time for training.
    - **Number of Neurons per Layer:** The number of neurons in an LSTM layer defines the dimension of the output space of that layer. In other words, it determines how much information the layer can process at each time step. A higher number of neurons allows the model to capture and retain more information about the data. However, as with the number of layers, an excess of neurons can lead to an overfitted model that does not generalize well to new data. Additionally, a higher number of neurons increases the computational complexity

of the model.

2. **Dropout**

   - **Functioning of Dropout:** The use of Dropout technique in an LSTM model is a key strategy to combat overfitting, especially relevant in deep neural networks. Dropout works by randomly "deactivating" a set of neurons during training. It randomly zeroes out the output of certain neurons in a layer with a predefined probability. This means each neuron has a specific probability of being temporarily "removed" from the model at each training step. This randomness prevents the model from becoming too dependent on any set of neurons, thus encouraging the model to learn more robust and generalizable representations.
   - **Choice of Dropout Rate:** The dropout rate is a hyperparameter that determines the probability of a neuron being deactivated. A higher rate means more neurons will be randomly deactivated. Choosing this rate is a balancing act; a rate too high can prevent the model from effectively learning from the data, while a rate too low might be insufficient to prevent overfitting. In practice, dropout rates commonly used range between 0.2 and 0.5, but the optimal rate may vary depending on the specific problem and model architecture.

3. **L1 and L2 Regularization**

   - **Principles of L1 and L2 Regularization:** L1 regularization (also known as Lasso regularization) and L2 regularization (or Ridge regularization) are two common methods to impose constraints on the network weights. L1 regularization penalizes the absolute sum of the weights, while L2 penalizes the sum of the squares of the weights. These penalties are added to the model's loss function during training.
   - **Effects of L1 Regularization:** L1 regularization tends to produce models with sparser weights, with many weights close to zero. This effect can be useful

for models where feature selection is important, as it can help identify and discard less relevant or noisy inputs.

- **Effects of L2 Regularization:** L2 regularization is effective at handling overfitting in models with a large number of parameters, as common in LSTM networks. By penalizing large weights, it prevents the model from becoming overly complex by giving too much importance to particular features of the training data.
- **Selection and Tuning of Regularization Coefficients:** Choosing the right coefficients for L1 and L2 regularization is a fine-tuning process. Coefficients too high can lead to an underfitted model, which fails to learn adequately from the training data. On the other hand, coefficients too low may be insufficient to prevent overfitting. The choice of these values often requires experimentation and can benefit from techniques like cross-validation to find an appropriate balance.

4. **Batch Size and Number of Epochs**

- **Batch Size:** The batch size refers to the number of training samples used in a single iteration of the learning process..
- **Impact on Learning:** A larger batch provides a more accurate gradient estimate but also requires more memory and can be more computationally costly. Conversely, a smaller batch offers a less accurate gradient estimate, which can lead to more "noisy" or fluctuating learning but often allows for faster convergence and can offer natural regularization due to its stochastic nature.
- **Selection of Batch Size:** Choosing the batch size is a balance between computational efficiency and gradient accuracy. A batch size too small can make the training unstable, while too large a size can make the training slower and possibly less capable of finding global optima.
- **Number of Epochs:** An epoch is a complete pass through the entire training dataset.

- **Importance in Training:** The number of epochs determines how many times the model will see the training data. A higher number of epochs may allow the model to learn more from the data but also increases the risk of overfitting.
- **Adjustment of the Number of Epochs:** Selecting the number of epochs often involves finding a point where the model has learned sufficiently from the data but has not begun to overfit.

Training

1. **Model Initialization:** Before training, the LSTM model is initialized with the defined architecture. This stage sets the foundation for how the model will process and learn from the data.
2. **Configuration of the Time Series Generator:** The use of TimeseriesGenerator ensures that the data input into the model is structured in a way that respects temporal sequentiality. This is crucial for the LSTM model to understand and learn the temporal dependencies in the data.
3. **Training Iterations (Epochs):** Each epoch represents a complete pass through the dataset. In each epoch, the model adjusts its internal parameters (weights) to minimize the loss function.
4. **Optimization and Weight Adjustment:** Using the Nadam optimizer, the model adjusts its weights to reduce the error between its predictions and the actual values. This is a gradual and iterative process, where the model seeks to find the optimal set of weights that minimizes the loss function..

Prediction and Visualization of Results

After training, the next phase in the analysis involves generating predictions and their comparative visualization with the actual data.

1. **Generation of Predictions:**

- **Prediction Mechanism:** The function is responsible for generating forecasts using the trained model. This process involves feeding the model with a dataset (in this case, the most recent from the training set) and using the model to predict the next point in the time series.
- **Preparation of Input Data:** To make predictions, a batch of initial data corresponding to the last points of the training set is selected. These data are used as a starting point for future predictions, and the model uses them to make a step-by-step projection forward.

2. **Visualization of Results:**

   - **Comparison with Real Data:** Once predictions are generated, these forecasted values are visualized alongside the actual data from the test set. This visualization is essential for assessing the accuracy of the model, allowing a direct comparison between what the model predicted and what actually occurred.

This presents the results obtained from the trained LSTM model. It's important to note that each hyperparameter can assume a variety of values, thus creating a broad spectrum of possible combinations. Each combination results in different performances and demands different levels of computational resources. Due to limitations in available computational capacity, only a restricted subset of these combinations was explored.

3.2.5 Prophet

Libraries Used

The libraries used are Pandas, Numpy, Matplotlib, and Scikit-learn, which have been previously explained. In addition to:

- **Prophet:** Developed by Facebook, this library is specifically designed for forecasting time series data. Prophet stands out for its ability to handle different types of seasonal trends and for its ease of use compared to other time series methods.

Data Splitting

The second stage of the process involves splitting the loaded data into training and testing sets.

- **Data Loading:** Using Pandas, the data is read from a CSV file into a dataframe.
- **Data Division:** The dataset is divided into two parts:
    1. **Training Set:** Comprising the initial 80% of the data, this set is used to train the forecast model.
    2. **Test Set:** Comprising the remaining 20%, this set is used to evaluate the accuracy of the model.

Training Data Preprocessing

- **Data Normalization:** Using Scikit-learn's MinMaxScaler, this technique adjusts the data values so that they are within a specific range, which is important for the performance and stability of the forecast model.
- **Data Restructuring for Prophet:** The same function also adapts the dataframe to meet the requirements of the Prophet model. The columns are renamed to "ds" and "y", standard formats that Prophet uses to identify the time variables ("ds") and the target variable to forecast ("y").
- **Removal of Unnecessary Columns:** Finally, the original "Date" and "Price" columns are removed from the dataframe to avoid duplications and maintain the clarity of the dataset..

Creation and Tuning of the Forecast Model

Once the data is preprocessed, the next step is the creation and tuning of the forecast model.

Using Prophet: Prophet, a library designed for time series forecasts, is employed. The Prophet model is initialized with specific configurations, such as the seasonality mode, which can be "additive" or "multiplicative". For the initial evaluation, the additive approach was chosen. Additionally, the change scale is adjusted, which regulates the model's flexibility in detecting changes in the data trend.

Seasonality and Holidays Configuration: The model is enriched with custom seasonality and holidays. For instance, a seasonality for "business_days" with a specific period and a Fourier order is added, and US holidays are integrated. These settings allow the model to capture specific patterns related to workdays and holidays, thus improving the accuracy of the forecast.

Model Fitting: The model is trained with the training dataset.

Making Forecasts with the Model

After fitting the model, the next step is generating forecasts..

Generating Future Dates: A future dataframe containing the future dates for which the forecast is desired is created. The number of periods and the frequency of these dates are specified according to the forecasting needs.

Prediction with the Trained Model: Using the trained Prophet model, the forecast is made for the generated future dates. Prophet provides a range of predictions, including the main estimate, and the lower and upper confidence intervals, which serve to understand the uncertainty associated with the predictions.

Inverse Transformation and Comparison with the Test Dataset

Inverse Transformation: Using MinMaxScaler, the forecasted values are rescaled to their original range.

Comparison with Test Data: The transformed forecasted values are integrated into the test dataset. This integration allows a direct comparison between the forecasts and the real values of the test set, providing a clear measure of the model's accuracy.

Visualization of Forecasts and Real Data: The function is employed to visualize the forecasts alongside the real data.

Repeating the Process with a Multiplicative Model

The forecasting cycle is completed by repeating the modeling and forecasting process using a multiplicative approach for seasonality and then visualizing the final results.

3.2.6 SARIMA

Library Imports

The libraries used include Matplotlib, NumPy, Pandas, and Sklearn, which were previously explained. Additionally:

- **Statsmodels.tsa.statespace.sarimax:** This is the central library for constructing the SARIMA model. It provides the necessary tools to fit the model to time series data, allowing the incorporation of seasonal and non-seasonal components into the model.

Data Loading and Preprocessing

- **Data Loading:** Using Pandas, the function reads a CSV file, assigning the date column as the index.
- **Training and Test Set Split:** The time series is divided into two segments, training and test. This division is done to evaluate the predictive capacity of the model. This approach allows for the validation of the model's efficacy on unseen data during training.

SARIMA Model Construction

Hyperparameter Selection: The SARIMA model is characterized by its set of hyperparameters, denoted in the notation (p, d, q) x (P, D, Q, s). Here, **p** represents the number

of autoregressive terms, **d** the degree of differentiation, and **q** the number of moving average terms for the model's non-seasonal component. Similarly, **P, D, Q** refer to the same concepts but applied to the seasonal component, and s indicates the seasonality period.

Model Initialization: Using the statsmodels.tsa.statespace.sarimax library, the SARIMA model is initialized with the selected hyperparameters. This initialization prepares the model to be fitted to the data, setting the foundation for analysis and prediction.

Training and Predictions with the SARIMA Model

Model Training: The SARIMA model, already configured with the appropriate hyperparameters, is trained using the scaled time series data from the training set. During this phase, the model learns the dependencies and underlying patterns in the historical data. This learning involves adjusting the model's coefficients to minimize the error between the predictions and the actual values.

Making Predictions: With the trained model, predictions are made on the test set. This phase involves using the model to predict future values, extending from the end of the training set to the end of the test set. The predictions are generated based on the patterns learned during training, and their accuracy is a key indicator of the model's effectiveness.

Model Performance Evaluation: The model's efficacy is assessed by comparing the generated predictions with the actual values of the test set. This evaluation is performed using the Root Mean Square Error (RMSE), which provides a quantitative estimate of the error in the predictions. Additionally, a graphical representation contrasts the predicted values with the real ones, thus facilitating a direct visual understanding of the model's accuracy.

SARIMA Model Hyperparameters

The SARIMA model is characterized by two sets of hyperparameters: order and seasonal_order.

1. **order (p, d, q):**

- **p** is the number of autoregressive terms. It indicates how many past values of the series are used to predict the current value.
- **d** is the degree of differentiation. It represents the number of times the data are differentiated to achieve stationarity.
- **q** is the number of moving average terms. It refers to the number of forecast error terms used in the model.

2. **seasonal_order (P, D, Q, s):**
   - **P, D, Q** are analogous to p, d, q, but applied to the seasonal component of the series.
   - **s** is the seasonality period.
3. **Hyperparameter Search:** Each hyperparameter in the model can take on a variety of values, and the combination of these values results in different model performances. Therefore, an exhaustive search technique known as "grid search" was implemented to explore all possible combinations of hyperparameters. However, this approach carries a high computational cost. Due to this limitation, the number of values each hyperparameter could take was restricted, in order to make the search more manageable and efficient.

3.3 Phase 1, Exogenous Algorithm Design

To develop simulations incorporating exogenous variables, it was necessary first to determine which exogenous variables would be used. In the context of stocks, the S&P 500 index values and the values of a relevant company in the same market sector were chosen, selecting Microsoft for this purpose. This resulted in the creation of three distinct datasets to work with. Each set includes Amazon's stock prices combined with the S&P 500 data, Microsoft's stock values, or a combination of both.

Consequently, the simulation environment for stocks is segmented into three main categories. In each category, the SARIMAX, Prophet, LSTM, and Holt-Winter models were

evaluated. Due to the incorporation of exogenous variables, the implementation of the SARIMA model was discarded. Each model was tested in the three main groups, resulting in a total of 12 distinct simulations, not counting the internal variations of each model.

In developing simulations focused on Bitcoin's price, the price of Ethereum, another cryptocurrency of similar relevance and presence in the market, as well as Bitcoin's own transaction volume, were selected as exogenous variables. This choice led to the creation of three different scenarios for testing: the analysis of Bitcoin's price in combination with its own transaction volume, the price of Ethereum, or the joint inclusion of both exogenous variables.

In each of these scenarios, the SARIMAX, Prophet, LSTM, and Holt-Winter models were applied. This resulted in a total of 12 different simulations for this section, not including the internal variations of each model. Therefore, in the entire segment dedicated to the implementation of exogenous variables, a minimum of 24 distinct simulations were carried out.

3.3.1 Creation of DataFrames, Stocks

Importing Libraries and Initial Setup

In this section, the focus is on preparing the programming environment for data analysis. As in the univariate analysis, essential libraries for data handling and visualization are imported. The libraries used are:

- **yfinance:** Allows downloading financial data directly from Yahoo Finance.
- **warnings:** Used to suppress warnings that may arise during code execution.
- **matplotlib.pyplot:** Provides functions for data visualization.
- **pandas:** Essential for data manipulation and analysis.

Downloading and Initial Setup of Financial Data with yfinance

In this phase, yfinance is used to download data from three different sources, each corresponding to a specific entity in the stock market. Unlike the univariate analysis that focused solely on Amazon, this multidimensional approach facilitates a deeper comparative analysis.

1. **Setting Dates:** An identical date range is defined for all downloads, ensuring consistency in the temporal comparison among the different datasets.
2. **Downloading Data:**
   - **Amazon:** Represents the data of Amazon.
   - **Microsoft:** Reflects the data of Microsoft.
   - **S&P 500 Index:** Represents the S&P 500 stock market index, providing an overall perspective of the performance of the US stock market, including a variety of sectors.
3. **Interval Setting and Automatic Adjustment:** A daily interval is set, and automatic adjustment is activated for all downloads. This ensures that stock prices are adjusted for dividends and splits, providing accurate and relevant data for financial analysis.

Processing and Cleaning of Downloaded Data

After downloading the data, the next step is their processing and cleaning. This stage is handled more complexly compared to the univariate analysis, due to the variety of data sources. The specific steps are:

1. **Creation of Individual DataFrames:** For each dataset (Amazon, Microsoft, and S&P 500), a separate DataFrame is created. This facilitates individualized handling of each data series, allowing for specific adjustments and transformations.
2. **Selection and Renaming of Columns:** From each DataFrame, only the "Close" column, representing the stock's closing price, is selected. This column is renamed

to reflect its source. This step is similar to the univariate analysis but repeated for each data source.

3. **Frequency Setting and Filling Missing Data:** Each DataFrame is set to have a business frequency, and missing data are filled using the "forward fill" method. This ensures that the DataFrames have a coherent and complete temporal structure.

Data Visualization

The final phase of the analysis is data visualization and exploratory analysis. Compared to the univariate analysis, this stage is more revealing due to the inclusion of multiple data series. The specific steps in this section are:

1. **Combining DataFrames:** Before visualization, the individual DataFrames of Amazon, Microsoft, and the S&P 500 are combined into a single DataFrame.

2. **Visualization Setup:** matplotlib.pyplot is used to create a graph with three different Y-axes, each representing one of the series. This allows visualizing and comparing the price trends of the stocks on the same graph but with scales that can be independent, highlighting the differences and similarities among them.

3. **Graph Customization:**

   - A distinct color is assigned to each data series for easy differentiation.
   - Labels for the axes and the title are added, improving the clarity and understanding of the graph.
   - Legends are strategically placed to identify each data series without obstructing the visualization.

Through this approach, we manage to form the dataset necessary to conduct simulations, simultaneously incorporating the main variable along with the two exogenous variables. However, in cases where only one of the two exogenous variables is required, the procedure is slightly adjusted. In these situations, we follow a similar process, but with one key difference: we concatenate exclusively the specific exogenous variable we wish to use.

3.3.2 Creation of DataFrames, Cryptocurrency

Importing Libraries and Initial Setup

In this first stage of the analysis, the necessary libraries are imported, and the initial setup is performed, similar to the process carried out in the univariate analysis. The imported libraries include:

- **Binance Client:** Provides the necessary functions to interact with the Binance API and perform financial data downloads.
- **matplotlib.pyplot:** Imported for data visualization, allowing the creation of charts for exploratory analysis.
- **numpy:** Provides support for mathematical operations and array handling.
- **pandas:** Essential for the manipulation and analysis of structured data.

Downloading and Initial Setup of Financial Data with yfinance

In this section, the process differs significantly from the univariate analysis. While the previous analysis focused solely on Bitcoin data, here the spectrum is broadened to include two key cryptocurrencies: Bitcoin and Ethereum. The specific steps are:

- **Selection of Cryptocurrency Symbols:** Symbols for Bitcoin and Ethereum are defined, setting the stage for a parallel download of data for both coins..
- **Retrieval of Historical Data:** Through the Binance API, historical data for both cryptocurrencies are obtained. The query is configured to retrieve the last 1000 data points.
- **Conversion of Data to DataFrames:** The downloaded data for each cryptocurrency is converted into Pandas DataFrames, which is essential for subsequent processing and analysis.

Processing and Cleaning of Downloaded Data

Here, the processing and cleaning of the data are performed for two different cryptocurrencies (Bitcoin and Ethereum). The key steps are:

- **Creation of Separate DataFrames:** This process involves the creation and handling of two separate DataFrames, one for each cryptocurrency.
- **Date Adjustment and Data Selection:** For each DataFrame, the time column is converted to a standard date and time format and set as the index. Additionally, relevant columns are selected and renamed. In the case of Bitcoin, an additional column for trading volume is added.
- **Combination of DataFrames:** Subsequently, the Bitcoin and Ethereum DataFrames are combined into a single DataFrame.

Data Visualization

In this case, the focus extends to include the comparative visualization of two distinct cryptocurrencies. The key steps are:

- **Data Preparation for Visualization:** Since the combined DataFrame now includes data from Bitcoin and Ethereum, the data are prepared for visualization that can reflect the differences and similarities between both cryptocurrencies. This involves adjusting the chart settings to accommodate multiple time series.
- **Use of Subplots for Comparison:** Subplots in matplotlib.pyplot are used to create separate but coherent charts for each cryptocurrency. For a direct visual comparison between Bitcoin and Ethereum, allowing to observe price trends and volume in parallel.
- **Customization and Detail in Visualization:** Each subplot is customized with distinct labels and colors for each cryptocurrency. Titles and legends are added to provide context and facilitate the understanding of the visualized data.

This process results in the creation of a dataset, where the price of Bitcoin, its trading volume, and the price of Ethereum are evaluated together. To generate the other two datasets, in which only one of the exogenous variables is incorporated, an analogous procedure is followed. The main difference lies in omitting the steps related to the inclusion of the unwanted exogenous variable.

3.3.3 Holt-Winter

For the analysis with the Holt-Winters model, we employed various Python libraries, each with a specific role in the process, similar to the Univariable section. We use matplotlib.pyplot for data visualization and Pandas for efficient data handling and manipulation.

The procedure begins with reading a CSV file, which is transformed into a Pandas DataFrame. Once the data is in DataFrame format, a crucial aspect is the proper configuration of the index to reflect the nature of the time series. This is achieved by setting a business day frequency on the index.

Data Division

After loading the data, the next step is to divide the dataset into two distinct segments: the training set and the test set.

The training set is used to fit the model, allowing it to learn the trends and patterns inherent in the data. On the other hand, the test set, consisting of data not seen by the model during the training phase, is used to evaluate its performance and predictive accuracy. Maintaining the logic of the univariate section, this data division ensures that the Holt-Winters model is trained and evaluated effectively.

Modeling and Prediction

Following a methodology similar to the univariate version, we fit the Holt-Winter model to the time series.

During the model fitting, there is the flexibility to specify whether the trend and seasonality should be treated as additive or multiplicative. The ability to experiment with both options allows finding the most suitable combination for our specific case, giving an equal number of combinations to the Univariable case. Parallel to exponential smoothing, a linear regression is implemented on the exogenous variables.

Finally, the predictions obtained from Holt-Winter are integrated with the results of the linear regression. This integration of models allows generating adjusted predictions that are more robust and reliable, taking into account both the internal patterns of the time series and the external influences captured by the exogenous variables.

Results Visualization

The final stage of the analysis focuses on visualizing the results obtained from the predictive model. Charts are used to represent both the predictions generated by the model and the actual data, thus facilitating an immediate visual comparison between the two.

3.3.4 LSTM

Diferences in Data Preparing

One of the notable differences between the current approach and the methodology described in the univariable section lies in the treatment of the time index. In this new approach, the time series index is set to the first column. Additionally, a differentiated strategy is implemented in the scaling of the data, using multiple scalers for different time series. Each series is independently scaled using MinMaxScaler.

The inclusion of additional data further highlights the evolution of this model's approach. The incorporation of this additional series brings a richer and more diverse dimension of analysis, allowing not only a deeper understanding of the main time series but also offering the possibility to explore more complex correlations and patterns within a broader market context.

Model Architecture and Training

LaThe architecture in essence is equal to the Univariable section. However, unlike the univariable section, where the analysis focused on a single time series, the second code introduces an advanced approach that integrates exogenous variables into the model. This multidimensional approach allows the model to capture more complex interactions and hidden patterns in the data, offering a richer and more nuanced analysis. In preparing the data for training, multiple time series are combined into a single dataset. This integration of exogenous variables is key to building a model that not only predicts based on past values of a time series but also considers the influence of other relevant series. By doing so, the model can learn how movements in one time series may be related to or influenced by fluctuations in others.

During the training of the model, these exogenous variables are presented to the LSTM model as additional features along with the main time series. It allows the model to learn from a broader spectrum of information, potentially leading to more accurate predictions and a better understanding of market dynamics.

Generating Predictions with Exogenous Variables

The generation of predictions represents a significant improvement over the univariable approach, thanks to the inclusion of exogenous variables. This section of the code addresses how these additional variables are used to enrich the prediction process. The prediction is based on the last available data sequence, which includes not only the past values of the main time series but also the additional series.

Visualization of Results with Exogenous Variables

The final stage of the analysis is the visualization of the results, to graphically show the model's predictions compared to the actual values of the time series.

3.3.5 Prophet

The same procedure as in the Univariable section is maintained, but with the following differences:

Improvement of the Prophet Forecast Model with Exogenous Variables

The function now accepts an additional argument, exogenous_features, which is a list of the names of the columns in the DataFrame that represent these variables. We use Prophet's add_regressor function to add each of these variables to the model. This approach allows the model to not only consider internal patterns of the time series but also adapt to external influences.

3.3.6 SARIMAX

Most of this section is the same as SARIMA explained previously, but with the following differences.

To integrate the exogenous variables into the SARIMAX model, a specific methodology was employed within the code. The syntax of the SARIMAX function in Python was adjusted to include these external data, taking into account aspects such as seasonality and potential correlation with the dependent variable. The integration of these exogenous variables is expected to improve the model's accuracy, allowing a better understanding of the factors influencing Amazon's price.

Technical Implementation of the SARIMAX Model:

To implement the SARIMAX model in Python, it is done using the statsmodels library. The key syntax used is as follows:

$$model = SARIMAX(x, exog, order = order, seasonal\_order)$$

- **x:** represents the main time series
- **exog:** introduces the exogenous variables.
- The parameters order and seasonal_order are used to define the structure of the ARIMA and SARIMA model, respectively.
- **order:** This argument receives a tuple of three elements (p, d, q), where "p" is the number of autoregressive terms, "d" is the number of differentiations needed to stationarize the series, and "q" is the number of moving average terms.
- **seasonal_order:** Similarly, this four-element tuple (P, D, Q, s) defines the seasonal part of the model, where "P", "D", and "Q" represent the seasonal equivalents of "p", "d", and "q", and "s" is the periodicity of seasonality.

As in the Univariable section (SARIMA), this model also presents a wide variety of possible combinations for each hyperparameter. Given this diversity, an exhaustive search strategy, known as "grid search", is implemented to identify the most optimal model. This methodology allows systematically exploring various combinations of hyperparameters, with the goal of finding the configuration that provides the best results in terms of performance and accuracy of the model.

3.4 Phase 2, Evolution and Analysis of the Best Models

The results primarily highlighted three models: LSTM, Holt-Winter, and SARIMAX. Given this evidence, it was decided to proceed with additional experimentations that mimic

conditions closer to a real market situation. To this end, different data sets were employed to simulate various market phases. Based on the findings from the initial tests, adjustments were made to the models to refine specific aspects of the predictions, such as the size of the data sets used. However, in the case of the LSTM model, computational limitations prevented an exhaustive exploration of its potential. Despite this, it's important to note that LSTM showed a significant margin for improvement regarding the obtained results, although reaching its maximum performance requires considerably more advanced computational resources.

Based on previous experimentations, it was decided to develop four different versions for stock predictions. This decision was made after analyzing data sets of different sizes, ranging from one year to four years. Each version included the analysis of four distinct scenarios: one with two exogenous variables, two with only one of the exogenous variables, and another univariate. In each scenario, the SARIMAX and Holt-Winter methods were applied, except in the univariate scenario where SARIMAX was not evaluated due to the absence of exogenous variables. Additionally, within each method, two variants of SARIMAX and Holt-Winter were developed, differentiated by the length of the predictions. Therefore, for stocks, a total of 4 versions were created, each with 4 scenarios. Each scenario employed the Holt-Winter and SARIMAX methods, with the exception of the univariate one. Furthermore, within each method, two variants were considered. This resulted in a total of 56 different simulations in the context of stock prediction.

In the context of cryptocurrencies, the decision was made not to implement the four different versions that were initially used for stocks. This resulted in a total of 14 distinct simulations for cryptocurrencies.

Once the predictions were obtained through these models, a detailed analysis of them was conducted. This analysis involved identifying specific points to perform buying and selling operations based on the predictions. Subsequently, these buying and selling points were placed within the timeline of the real market values of cryptocurrencies. This allowed accurately marking the moments of purchase and sale according to the predictions. Finally,

an evaluation of the profitability of the models was carried out, analyzing the financial performance of the identified buying and selling points.

Chapter 4

Results and Discussion

In this section, we will discuss the most notable results obtained. Although the Prophet and SARIMA models showed acceptable performance, they did not reach the level of effectiveness required to be considered in this analysis. Therefore, due to their lesser relevance compared to other models that did achieve viable results, they will not be included in this discussion. This allows focusing on those models that met the minimum criteria of viability and that are more promising for future experimentation.

Accordingly, this section will primarily focus on the Holt-Winter and SARIMAX models. Regarding the LSTM model, its potential to generate superior results is acknowledged. However, due to computational limitations, it was not possible to obtain sufficiently relevant results to develop a deeper exploration of this model..

4.1 Phase 1 Results

This section presents the results obtained from the first detailed analysis in the methodology. This analysis focused on comparing the predictions generated by the models with the actual values. The main objective was to evaluate the accuracy and effectiveness of the models in projecting data, providing an initial insight into their performance.

4.1.1 Datasets

As described in the methodology, the downloaded data sets were graphically represented to facilitate the visualization and analysis of the gathered information. During the examination of the exogenous variables, it was observed that the behavior of Microsoft's stocks and Ethereum shows a notable similarity to that of Amazon's stocks and Bitcoin, respectively, see Figures 4.1 and 4.2. This trend remained constant over long periods.

This similarity suggests a possible strong correlation between the prices of Amazon and Microsoft stocks, indicating that a movement in the price of one of these stocks could predict a similar movement in the other, although it is worth noting that they operate in different price ranges. In the case of Ethereum and Bitcoin, an even more marked correlation was found, with very similar patterns in the various data sets. However, as in the previous case, these two cryptocurrencies operate at different price scales.

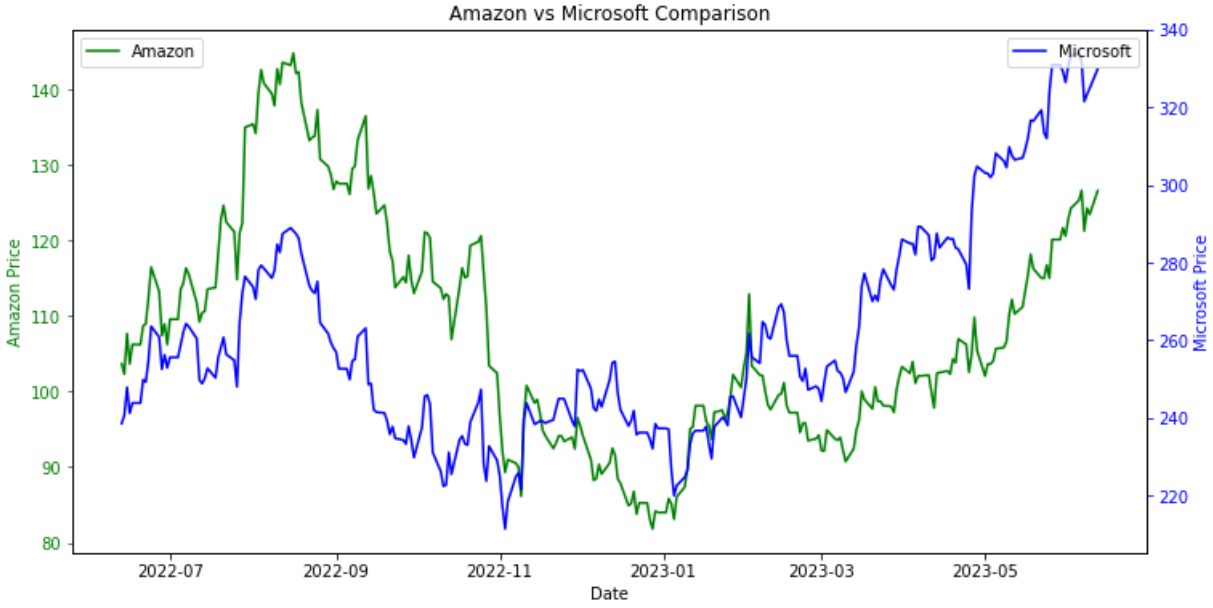


Figure 4.1: Amazon and Microsoft Dataset.

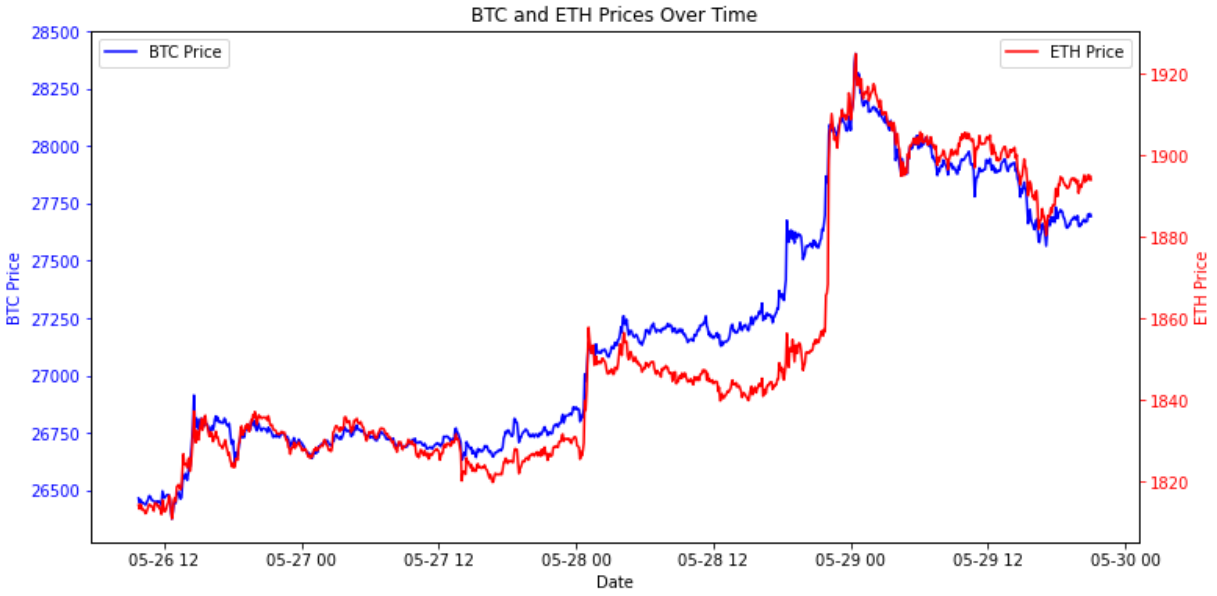


Figure 4.2: Bitcoin and Ethereum Dataset.

4.1.2 Holt-Winter, Univariable

As mentioned earlier, a total of 8(x/2) different simulations were conducted. In this section, selected results from all these simulations will be presented, focusing specifically on those considered most relevant. This selection aims to highlight the most significant and representative findings from the set of simulations conducted.

The analysis of the three graphs 4.3, 4.4 y 4.5 reveals that the proposed model manages to approximate the general trend of the actual Bitcoin prices. Although there are specific deviations between the predicted and real values, the model reasonably reflects both the direction and scale of the price changes. This level of similarity indicates that the model has potential to be applied in predicting the price of Bitcoin, which could be useful for informing buying and selling decisions. However, it is important to recognize and consider the observed error margins when applying these predictions in a real investment context.

Cryptocurrency

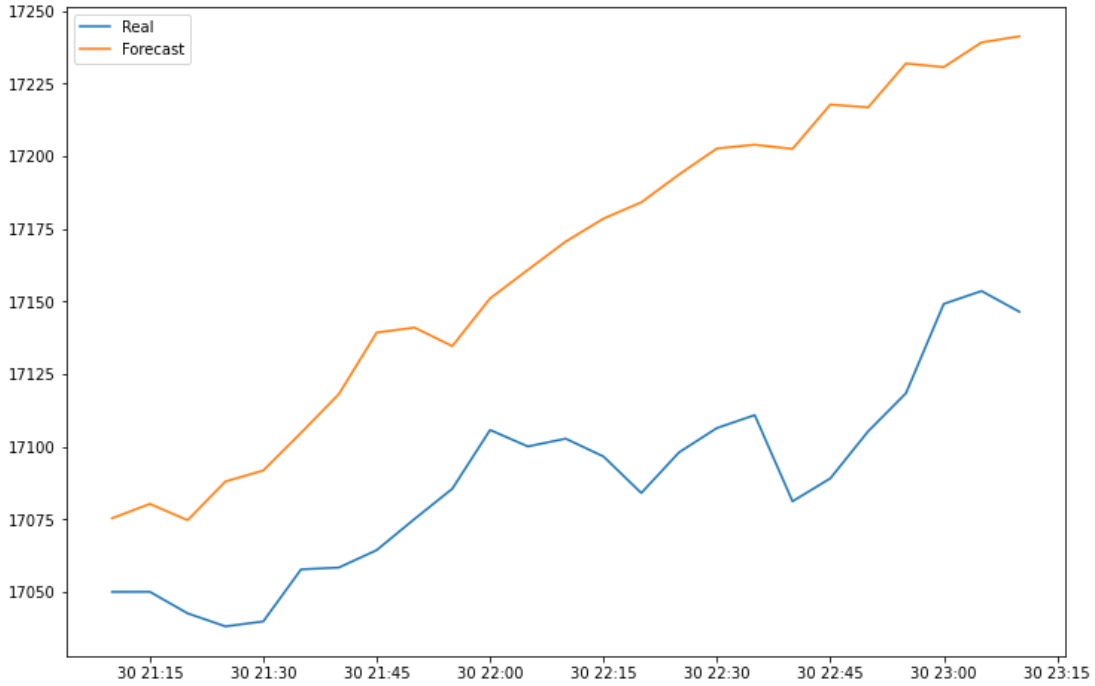


Figure 4.3: Bitcoin Prediction 1.

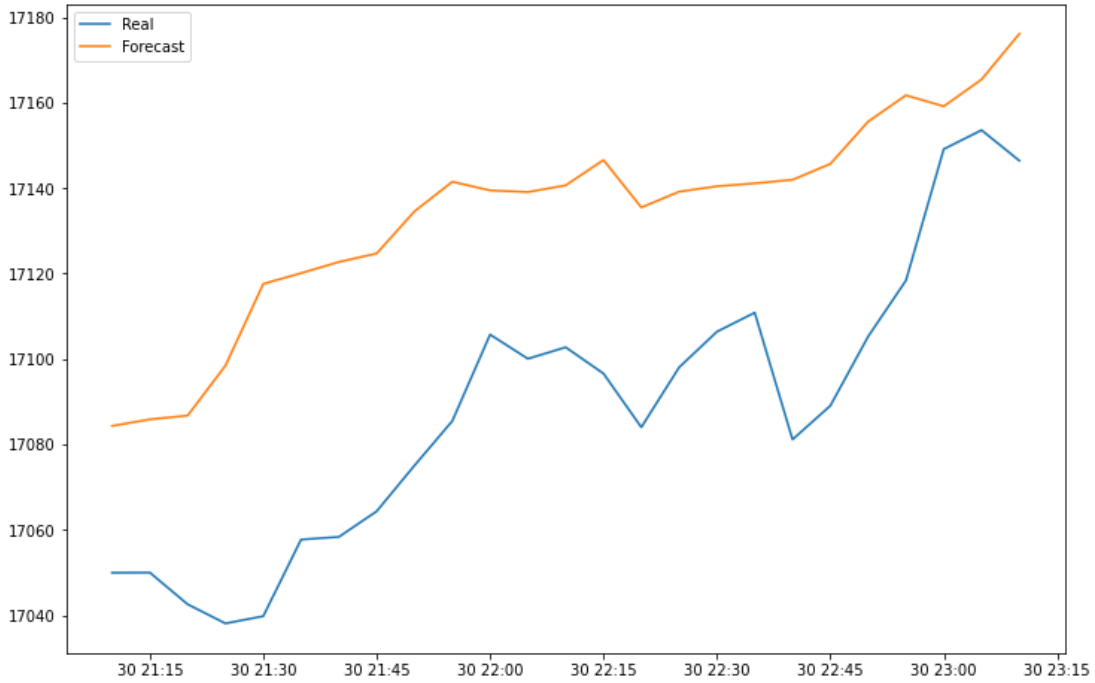


Figure 4.4: Bitcoin Prediction 2.

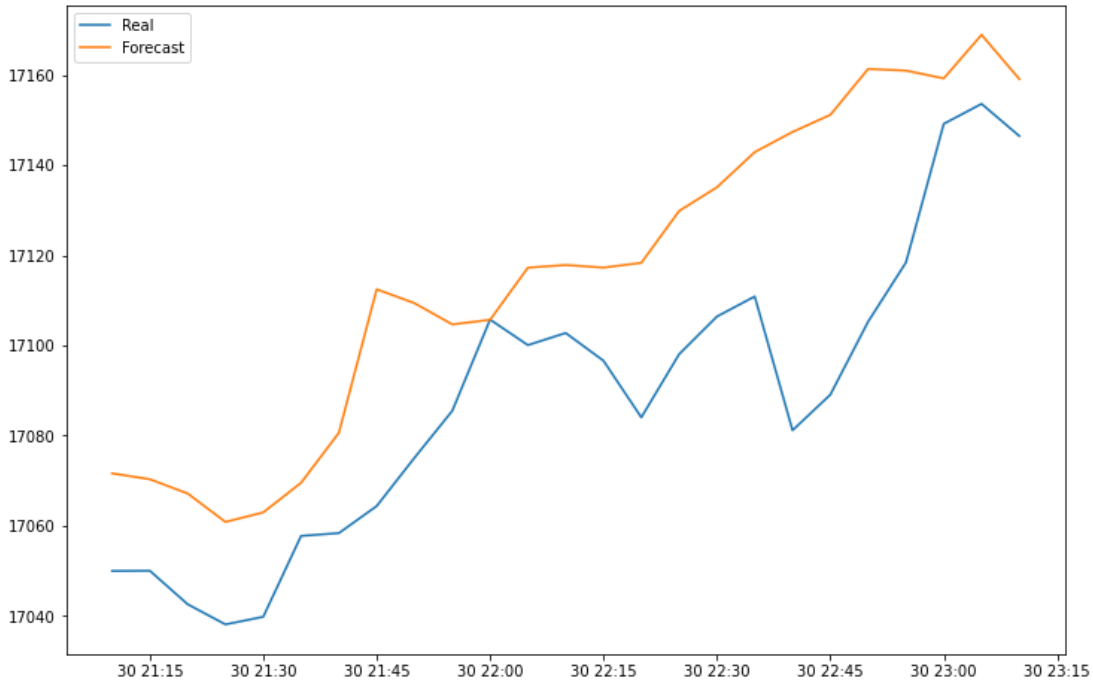


Figure 4.5: Bitcoin Prediction 3.

Stocks

The graphs 4.6, 4.7 and 4.8 demonstrate the comparison between the actual stock prices and their corresponding predictions over the period from March to April 2023. It is observed that the predictions follow a trend parallel to the actual evolution of the prices, although with some discrepancies in specific values. The general trend indicates that the model can detect the market movement direction but also highlights the existence of error margins that must be considered.

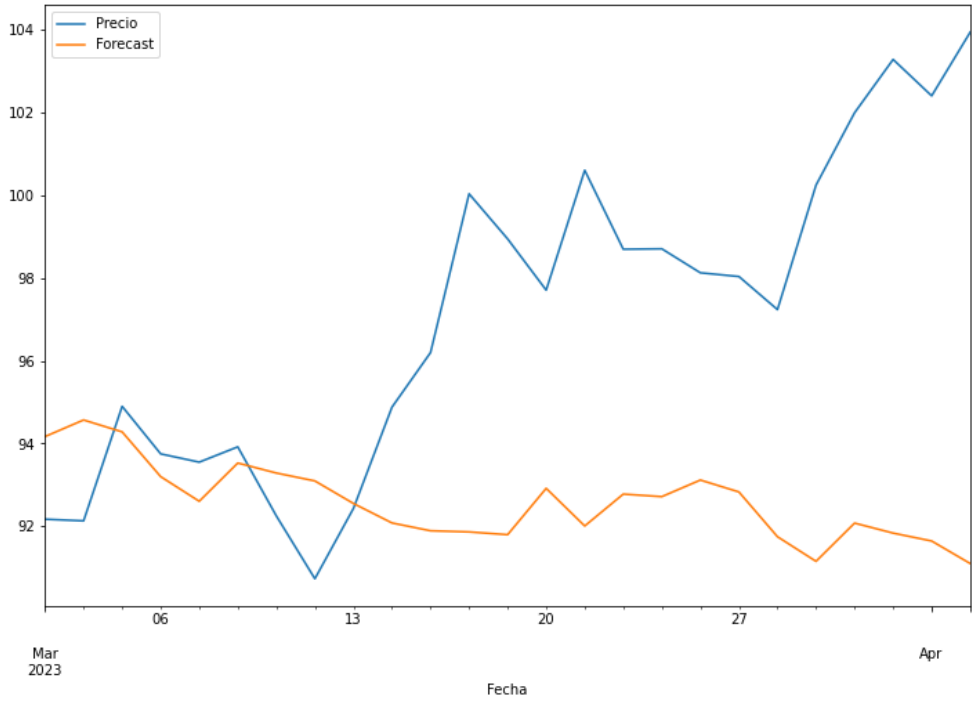


Figure 4.6: Amazon Prediction 1.

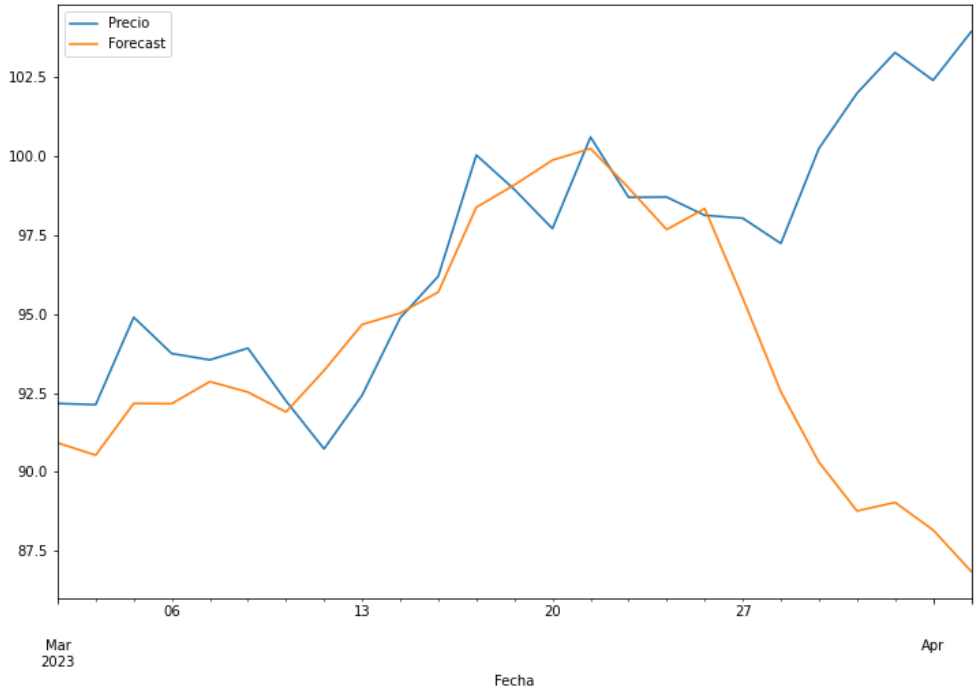


Figure 4.7: Amazon Prediction 2.

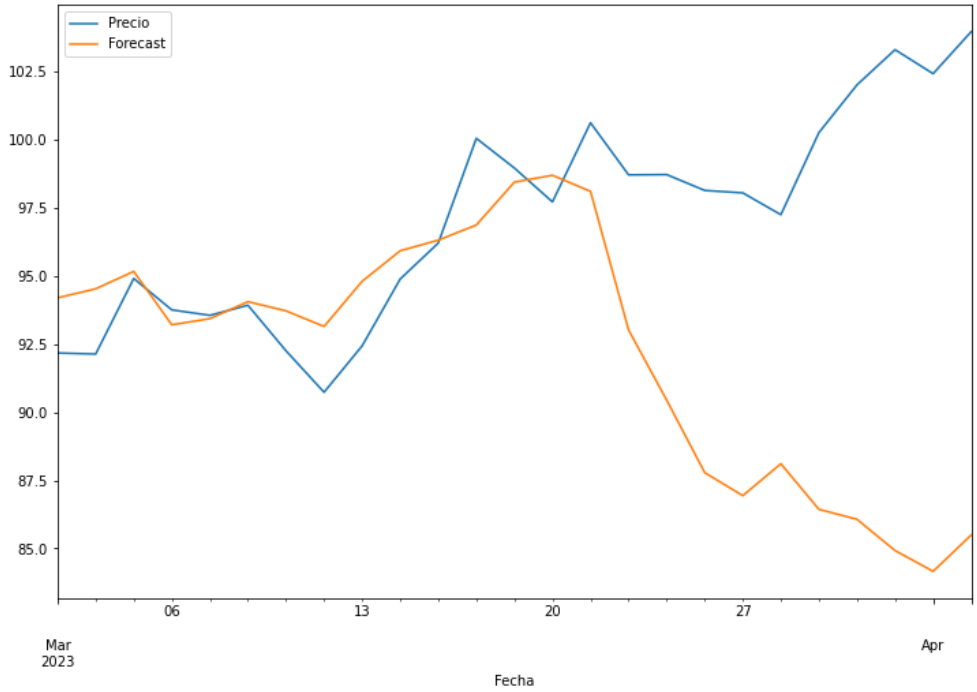


Figure 4.8: Amazon Prediction 3.

4.1.3 Holt-Winter, Exogenas

As detailed in the methodology section, two exogenous variables were incorporated into the analysis, leading to the creation of three different scenarios. In the first and second scenarios, only one of the exogenous variables was used, while in the third scenario, both were employed simultaneously. This approach is similar to the one used in the univariable analysis, where each scenario presents the same number of possible internal combinations. For effective comparison, we selected the two most significant results from each scenario. This selection highlights not only the differences between the individual scenarios but also the variations compared to the univariable analysis.

Cryptocurrency

As seen in Graphs 4.9, 4.10 and 4.11, the three models demonstrate notable adaptability to the dataset, showing precision in predicting both the trend and magnitude of Bitcoin prices. Particularly, Figure 4.9 stands out for its ability to more faithfully replicate the market trend over the prediction period. On the other hand, Figures 4.10 and 4.11, although initially close to real values and replicating the general market behavior, tend to deviate

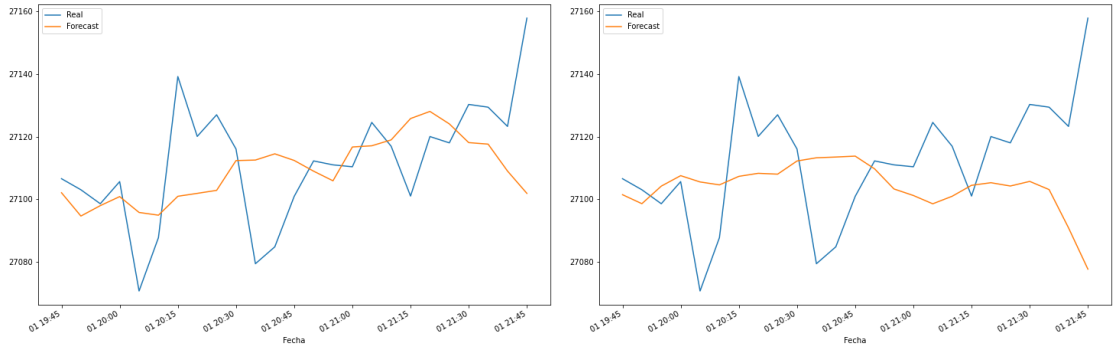
from these values as the prediction extends, thus increasing the margin of error.

(a) Result 1 (b) Result 2

Figure 4.9: Bitcoin Predictions with Ethereum as an exogenous variable in a and b, but with different hyperparameters.

(a) Result 1 (b) Result 2

Figure 4.10: Bitcoin Predictions with Volume as an exogenous variable in a and b, but with different hyperparameters.

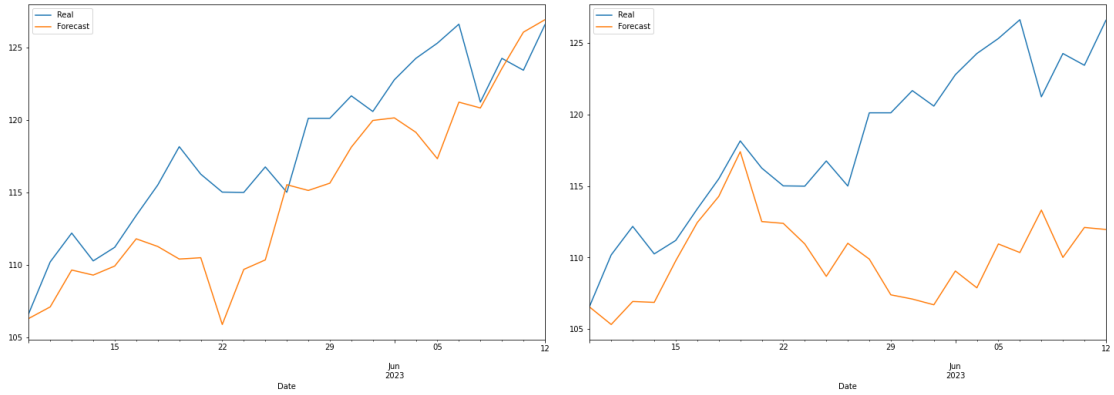


(a) Result 1 (b) Result 2

Figure 4.11: Bitcoin Predictions with Volume and Ethereum as exogenous variables in a and b, but with different hyperparameters.

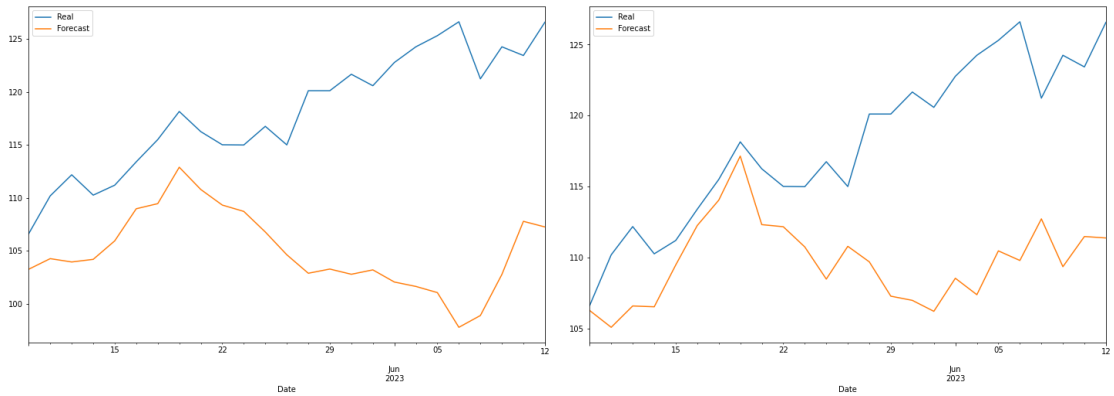
Stocks

Figures 4.12, 4.13 and 4.14 provide an interesting insight into the predictive capability of the models in relation to Amazon's stock prices. At the start of the prediction period, the three models demonstrate reasonable competence in capturing market behavior. However, significant differences are observed as the analysis period extends. In particular, Figure 4.12 appears to have an advantage in terms of predicting price behavior over time, though the associated margin of error must be considered. This observation suggests that while the models have some utility in forecasting short-term trends, their capabilities for long-term forecasts vary, and caution should be exercised when interpreting these results.



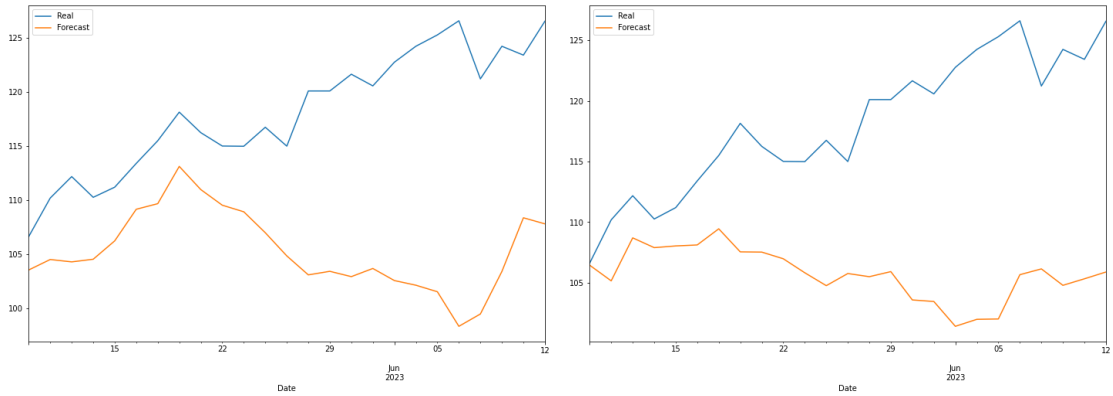
(a) Result 1 (b) Result 2

Figure 4.12: Amazon Predictions with Microsoft as an exogenous variable in a and b, but with different hyperparameters.



(a) Result 1 (b) Result 2

Figure 4.13: Amazon Predictions with S&P500 as an exogenous variable in a and b, but with different hyperparameters.



(a) Result 1 (b) Result 2

Figure 4.14: Amazon Predictions with S&P500 and Microsoft as exogenous variables in a and b, but with different hyperparameters.

4.1.4 SARIMAX

As detailed in the methodology, the SARIMAX model incorporates several hyperparameters, each with a wide range of possible values. This leads to an extensive variety of potential combinations, thus increasing the demand for computational resources. Given this requirement, it was decided to limit the selection to a narrower range of possible values for each hyperparameter. It is important to recognize, therefore, that the results obtained reflect only a fraction of the full potential and predictive capacity of the SARIMAX model.

Cryptocurrency

Graphs 4.15 and 4.16 clearly illustrate that the models analyzed exhibit a notable ability to predict certain sections of the time series. In these specific parts, the models not only capture the behavior of the series but also its magnitude accurately. However, this predictive capacity does not uniformly extend across the entire time series. It is important to note that, compared to other models examined, SARIMAX proves to be superior in terms of predictive capacity in specific segments of the series.

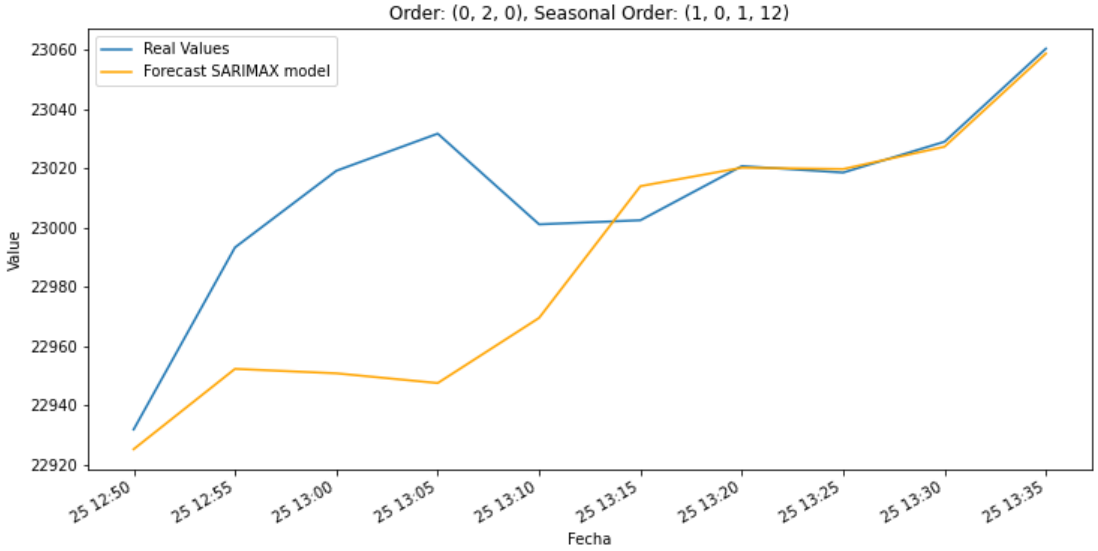


Figure 4.15: Bitcoin Prediction with Ethereum as an exogenous variable.

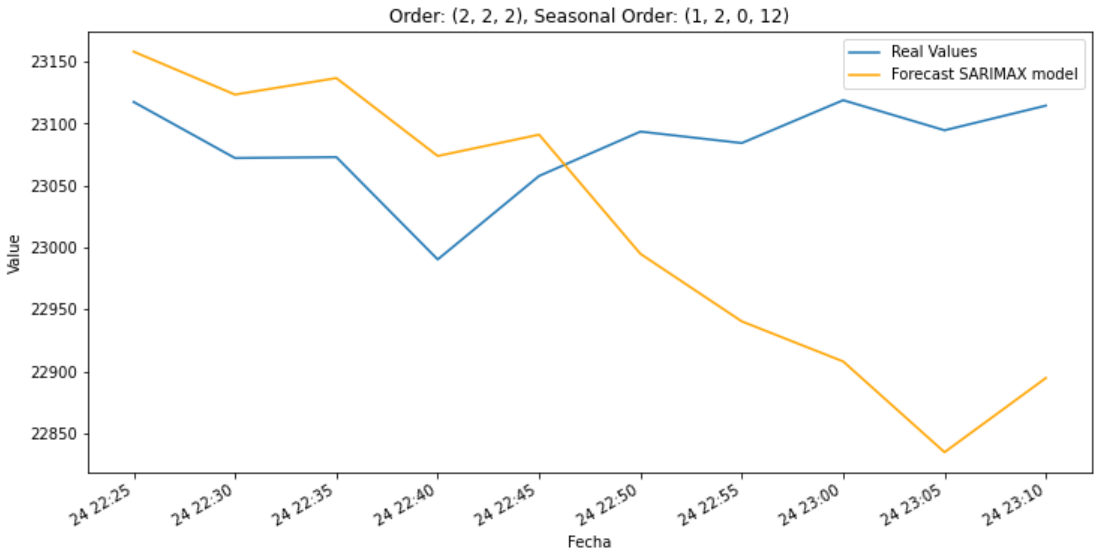


Figure 4.16: Bitcoin Prediction with volume as an exogenous variable.

Stocks

Graphs 4.17, 4.18 and 4.19 from the SARIMAX model did not show outstanding adaptability in predicting Amazon's stocks. However, as mentioned earlier, these results represent only one facet of SARIMAX's potential as a predictive tool. Considering its performance in the cryptocurrency section, it is important to recognize the relevance of the SARIMAX model in diverse contexts.

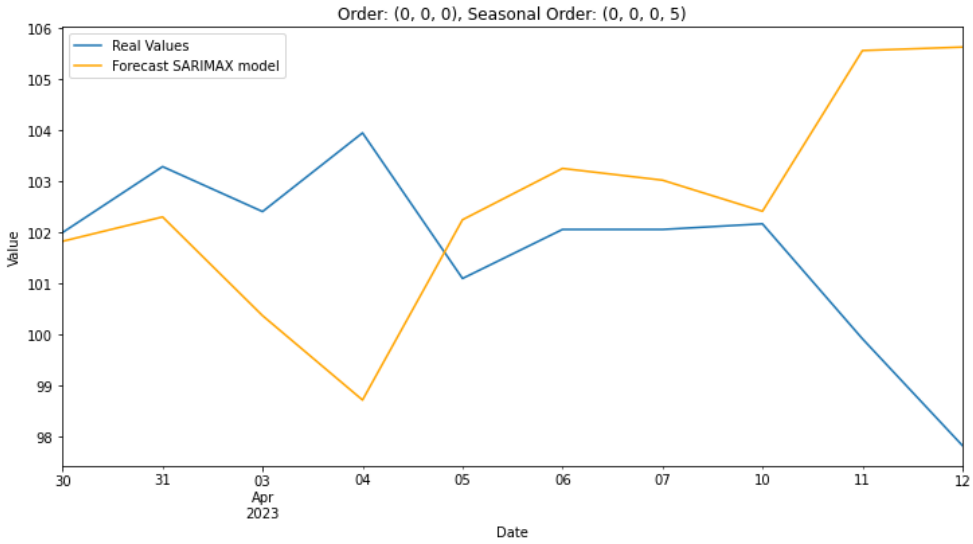


Figure 4.17: Amazon Prediction with Ethereum as an exogenous variable.

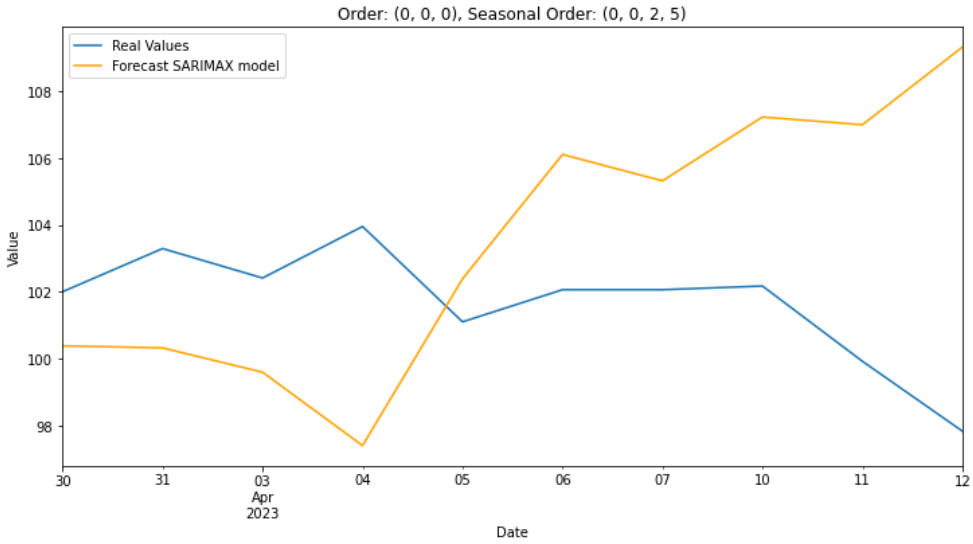


Figure 4.18: Amazon Prediction with volume as an exogenous variable.

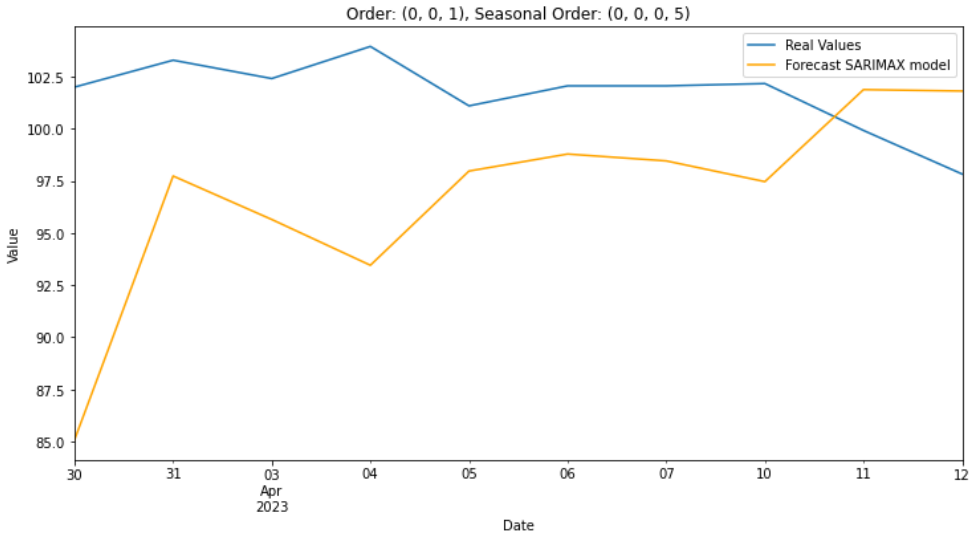


Figure 4.19: Amazon Prediction with volume and Ethereum as exogenous variables.

4.2 Phase 2 Results

The results obtained in the previous phase led to the decision to continue with experimentation but with a more refined strategy. At this stage, improvements were made to the proposed models, and a mechanism for simulating buy and sell transactions was integrated. Thus, the models evolved not only to analyze the time series of Bitcoin prices and Amazon's stocks and generate predictions based on these series but also to make informed decisions about the optimal times to buy and sell based on these predictions. This advanced approach represents a significant step in the practical application of the models in the context of the financial market.

The experimentations conducted led to the definition of specific ranges for the lengths of the predictions. These ranges were established with a minimum limit, denoted as **C**, and a maximum limit, referred to as **Z**. The need to establish a minimum limit **C** arises from the fact that prediction lengths shorter than this threshold do not provide enough data to execute buy and sell operations effectively. On the other hand, lengths exceeding the maximum limit **Z** tend to introduce an unacceptable level of errors in the predictions. To determine the optimal buying and selling moments, the model focuses on analyzing the predictions and establishing the key growth points. In situations where an increase in values is projected, the model executes the purchase at the predicted minimum value and

proceeds to sell at the maximum point. A critical aspect of this model is its rule not to make purchases unless a corresponding sale can be guaranteed within the same time interval. In scenarios where predictions indicate multiple upward and downward fluctuations, the model adopts a strategy of buying at the lowest values and selling at the highest peaks.

As a result of this methodology, in each of the simulations carried out, the number of buy transactions could vary from zero to several, depending on the asset analyzed. This is because the program's algorithm is designed to identify multiple optimal points within a single prediction where buys and sells could be beneficial. Therefore, in a given scenario, the program might decide to execute several buy and sell operations, based on its analysis of the most auspicious points within the series of predictions.

Consequently, each simulation was evaluated using the following metrics: Good, Bad, Null, No Purchase, Confidence, Utility, and % Utility. These metrics allow a detailed understanding of the performance of each simulation:

- **Good:** Assigned when all buy and sell operations in a prediction result in profits.
- **Bad:** Assigned when all buy and sell operations in a prediction result in losses.
- **Null:** Assigned when the buy and sell operations in a prediction result in a combination of gains and losses.
- **No Purchase:** Indicates that the model did not find favorable conditions to perform any buy and sell operation.
- **Confidence:** Represents the percentage of the totality of buy and sell operations based on predictions that resulted in profits.
- **Utility:** Corresponds to the net benefit obtained in the simulations, considering that each operation was performed at market price.
- **% Utility:** Indicates the percentage of the net benefit in relation to the total investment, which allows visualizing what percentage of the capital invested was recovered as profit.

Accordingly, this section will be dedicated exclusively to the results obtained in the simulations, focusing on the specific metrics resulting from each one. The indicators such as Good, Bad, Null, No Purchase, Confidence, Utility, and % Utility will be examined in detail, thus providing a comprehensive understanding of the performance of the models in various simulated scenarios.

The structure of the tables in this analysis is organized around a main title indicating the model used, accompanied by the variable C or Z that signals the length of the predictions. Each table consists of eight columns; the first column specifies the implemented method, while the remaining seven present the metrics that were described previously.

In the case of the Holt-Winter models, four rows corresponding to the Univariable methods and the three variants with exogenous variables of the asset to be predicted are shown. On the other hand, the tables corresponding to the SARIMAX model contain only three rows since this model is not applied in the univariable approach. Each row represents a specific method, followed by the values corresponding to the mentioned metrics.

These values are the result of simulating each method 100 times. Ten different datasets were used, and the program made predictions at various points of each set, ensuring that changes in the length of the training section did not significantly affect the results. In summary, the tables show the accumulated results of each method after operating 100 times with 10 different datasets.

4.2.1 Stocks

As detailed in previous sections, four different versions were developed for stock prediction. Therefore, in the following part, the corresponding tables showing the results of each of these versions will be presented.

Tables from 4.1 to 4.16 demonstrate a high utility index, which can be significantly attributed to the variability in Amazon's stock prices. For example, in 2019, the price of an

Amazon stock was valued at 90 monetary units, while by 2023, this value rose to about 170 units, nearly doubling its value over a four-year period. This notable increase is reflected in the utility indices reported since the 10 selected data sets span different years. This time range allows for observing cases where the appreciation in Amazon's stock value has had a significant impact on the utility results recorded in the tables.

In version 1, tables 4.1 and 4.2 present the results obtained through the Holt-Winter methods. In table 4.1, the "Exogenous Microsoft" method stands out as the most effective according to the first four metrics: "Good", "Bad", "Null", and "No Purchase". The results obtained are as follows: 39% of the predictions resulted in net gains, 29% ended in net losses, 25% in operations that combined gains and losses, and 7% of the predictions did not lead to any operation.

When evaluating the metrics of "Confidence", "Utility", and "% Utility", we observe that 54.55% of the buy and sell operations resulted in profits. A utility of $53.06 was obtained in all operations at market prices, representing a return of 47.12% on the investment made.

In table 4.2, the "Exogenous S&P500" method shows remarkable performance, especially in the "Good" metric, where it surpasses the best method in table 4.1. Although the difference in the "Bad" metric is only 2 points compared to the highlighted method in table 4.1, the "Confidence" value is higher. However, it is important to highlight that both the "Utility" and "% Utility" are lower in this method. This is because, although the "Exogenous S&P500" method is more frequently correct than the best method in table 4.1, the losses incurred in the incorrect predictions are more significant, resulting in lower overall gains..

This situation highlights the importance of considering the metrics of "Utility" and "% Utility" in the analysis of the models. These metrics allow a deeper evaluation of the models, not only in terms of frequency of correct predictions but also in terms of the magnitude of the gains or losses generated. This facilitates the selection of the most effective model in general terms.

Tables 4.3 and 4.4 focus on the analysis of the SARIMAX model. A distinctive feature of this model, compared to Holt-Winter, is its high value in the "No Purchase" metric. This indicates that the SARIMAX model tends to make fewer buy operations, suggesting a more cautious or selective strategy in executing transactions. Despite this lower frequency in making purchases, it is observed that the "Utility" and "% Utility" of the SARIMAX model do not differ significantly from those obtained in the Holt-Winter models.

This observation leads to the conclusion that although the SARIMAX model executes a smaller number of transactions, the operations it performs tend to be safer or more profitable. In other words, the SARIMAX model seems to prioritize the quality of the buy and sell operations over the quantity, resulting in comparable efficiency in terms of utility and percentage of utility to the Holt-Winter models.

The detailed analyses performed for tables 4.1 to 4.4 can be similarly applied to versions 2, 3, and 4 of our models. That is, we can compare and evaluate these additional versions using the same metrics of "Good", "Bad", "Null", "No Purchase", "Confidence", "Utility", and "% Utility". This comparison allows understanding how different iterations of the models affect their performance in terms of frequency and safety in buy and sell operations, as well as their overall impact on utility and efficiency. By applying the same analytical approach to versions 2, 3, and 4, we will be able to identify consistent patterns or significant differences among the versions. This is crucial for determining which versions of the models are more effective and under which circumstances, providing a deeper understanding of the robustness and applicability of our models in various market scenarios.

Version 1

Table 4.1: Results with Holt-Winters Model Z.

Holt-Winters Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	36	32	24	8	52.12	16,285	38.36%
Exogenous Microsoft	39	29	25	7	54.55	53,06	47.12%
Exogenous S&P500	35	31	24	10	50.63	27,181	36.53%
2 Exogenous	37	27	28	8	53.94	64,417	54.40%

Table 4.2: Results with Holt-Winters Model C.

Holt-Winters C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	49	37	11	3	54.47	-5,236	-11.66%
Exogenous Microsoft	49	32	18	1	57.97	13,264	25.30%
Exogenous S&P500	49	31	19	1	59.12	27,748	30.23%
2 Exogenous	46	31	21	2	56.74	23,275	31.32%

Table 4.3: Results with SARIMAX Model Z.

SARIMAX Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	15	5	12	68	62.5	28,732	13.04%
Exogenous S&P500	17	8	8	67	62.3	22,625	13.03%
2 Exogenous	11	6	11	72	57.14	19,542	7.98%

Table 4.4: Results with SARIMAX Model C.

SARIMAX C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	22	21	5	52	54.69	-13,733	-14.80%
Exogenous S&P500	33	27	8	32	56.98	2,424	2.24%
2 Exogenous	26	24	8	42	55.26	3,032	-0.43%

Version 2

Table 4.5: Results with Holt-Winters Model Z.

Holt-Winters Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	34	24	37	5	54.12	-3,018	19.65%
Exogenous Microsoft	45	23	27	5	59.39	55,522	63.25%
Exogenous S&P500	40	31	28	1	53.63	34,086	56.32%
2 Exogenous	39	28	30	3	56.74	53,424	39.72%

Table 4.6: Results with Holt-Winters Model C.

Holt-Winters C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	46	34	14	6	55.56	27,28	21.80%
Exogenous Microsoft	50	32	13	5	59.66	9,624	8.88%
Exogenous S&P500	43	35	12	10	54.55	5,23	18.77%
2 Exogenous	51	34	12	3	57.14	31,514	36.52%

Table 4.7: Results with SARIMAX Model Z.

SARIMAX Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	12	7	8	73	58.7	19,269	20.13%
Exogenous S&P500	18	7	9	66	64.29	12,093	29.65%
2 Exogenous	9	9	11	71	52	11,722	12.33%

Table 4.8: Results with SARIMAX Model C.

SARIMAX C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	24	23	11	42	49.33	-90,513	-101.90%
Exogenous S&P500	24	21	13	42	51.35	-8,616	-20.75%
2 Exogenous	31	26	12	31	52.81	5,492	16.03%

Version 3

Table 4.9: Results with Holt-Winters Model Z.

Holt-Winters Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	35.29	21.57	37.25	5.88	55.76	59,775	85.53%
Exogenous Microsoft	36.27	27.45	25.49	10.78	57.42	5,75	38.33%
Exogenous S&P500	37.25	26.47	30.39	5.88	55.77	58,503	57.31%
2 Exogenous	33.33	28.43	30.39	7.84	52.94	-2,038	13.40%

Table 4.10: Results with Holt-Winters Model C.

Holt-Winters C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	38.28	40.2	19.61	1.96	50.39	15,917	22.54%
Exogenous Microsoft	39.22	39.22	18.63	2.94	52.27	-4,571	-13.64%
Exogenous S&P500	42.16	37.25	17.65	2.94	53.91	7,41	44.69%
2 Exogenous	49.02	28.43	18.63	3.92	61.07	24,549	30.14%

Table 4.11: Results with SARIMAX Model Z.

SARIMAX Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	11.76	7.84	8.82	71.57	57.69	6,275	17.26%
Exogenous S&P500	14.71	8.82	6.86	69.61	55.93	26,413	35.02%
2 Exogenous	9.8	10.78	4.9	74.51	42.55	-10,565	-33.09%

Table 4.12: Results with SARIMAX Model C.

SARIMAX C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	23.53	23.53	10.78	42.16	52.56	-21,762	-11.83%
Exogenous S&P500	33.33	24.51	6.86	35.29	55.56	976	9.93%
2 Exogenous	20.59	30.39	12.75	36.27	43.68	-25,736	-13.71%

Version 4

Table 4.13: Results with Holt-Winters Model Z.

Holt-Winters Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	38.24	26.47	27.45	7.84	53.76	25,343	60.25%
Exogenous Microsoft	37.25	23.53	37.25	1.96	56.28	27,712	66.01%
Exogenous S&P500	42.16	34.31	20.59	2.94	53.89	11,716	37.38%
2 Exogenous	30.39	35.29	32.35	1.96	48.35	36,094	26.20%

Table 4.14: Results with Holt-Winters Model C.

Holt-Winters C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	38.24	44.12	11.76	5.88	49.22	-26,864	-22.68%
Exogenous Microsoft	40.2	38.24	16.66	4.9	48.41	-11,059	7.22%
Exogenous S&P500	43.14	38.24	9.8	8.82	53.78	-75,683	-74.43%
2 Exogenous	46.08	32.35	17.65	3.92	54.14	10,181	31.80%

Table 4.15: Results with SARIMAX Model Z.

SARIMAX Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	17.65	8.82	7.84	65.69	66.1	34,352	45.39%
Exogenous S&P500	12.75	13.73	11.76	61.76	46.67	-16,303	-5.10%
2 Exogenous	12.75	7.84	6.86	72.25	54.72	16,511	14.21%

Table 4.16: Results with SARIMAX Model C.

SARIMAX C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Microsoft	19.61	20.59	16.67	43.14	48.81	3,671	4.17%
Exogenous S&P500	24.51	29.41	10.78	35.29	43.75	-39,293	-44.02%
2 Exogenous	17.65	29.41	14.71	38.24	40	-19,609	-16.80%

Analysis of Versions 1 to 4

Table 4.17 presents the average of the results obtained in the different parameters of the Holt-Winters model, with the aim of analyzing which of the three versions showed superior performance. It is important to highlight that all versions recorded a low percentage in the "No Purchase" category, suggesting that the Holt-Winters model generally performs active buying and selling operations.

It is observed that all versions have approximate values of 30 in the "Bad" category, indicating that approximately 30% of the prediction sets result in inevitable losses. A similar pattern is identified in the "Null" parameter, where the values are predominantly close to 20%, suggesting that 20% of the predictions conclude in operations where gains and losses are inevitable.

Notably, all versions exhibit a "Confidence" index higher than 52%. This implies that there is a degree of control in the predictions, dismissing the possibility that the results are due to chance, similar to flipping a coin. This percentage indicates that more than 52% of the predictions result in profits.

Finally, in the "% Utility" parameter, with the exception of version 4, all versions exceed 30%. This means that in versions 1 to 3, the Holt-Winters models achieved an average of 30% profitability on the investment made.

Table 4.17: Analysis of Results with Holt-Winters.

Holt-Winters							
Versión	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Versión 1	42.5	31.25	21.25	5	54.94	27.49	31.45%
Versión 2	43.5	30.12	21.62	4.75	56.34	26.70	33.11%
Versión 3	38.85	31.12	24.75	5.26	54.94	20.66	34.78%
Versión 4	34.78	34.06	21.68	4.77	52.22	-0.32	16.47%

In Table 4.18, the analysis of the average of the different parameters for each version of the SARIMAX model is presented. One of the most notable findings is the high percentage recorded in the "No Purchase" category. All values exceed 50%, indicating that in more than 50% of the predictions, the model opts not to carry out buying or selling operations of the asset.

Despite the low frequency of transactions, it is observed that in the "Confidence" category, most versions of the model present values above 50%. This suggests that, although buying and selling operations are infrequent, when these are performed, more than half turn out to be correct.

Regarding the "% Utility" parameter, low percentages are registered. This situation seems to be due to the fact that different implementations of the SARIMAX model yielded very varied results, generating significant gains in some cases and losses in others. This behavior of disparate results translates into a lower average utility.

Therefore, it can be concluded that the SARIMAX model prioritizes quality over quantity in buying and selling operations. However, this approach entails less stability, as minor variations in the model's configuration can result in significantly different outcomes. This characteristic underlines the sensitivity of the SARIMAX model to changes in its parameters, which can affect both the effectiveness and consistency of its predictions.

Table 4.18: Analysis of Results with SARIMAX.

SARIMAX							
Versión	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Versión 1	20.66	15.16	8.66	55.5	58.14	10.43	3.51%
Versión 2	19.66	15.5	10.66	54.16	54.74	-8.42	-7.41%
Versión 3	18.95	17.64	8.49	54.90	51.32	-4.06	0.59%
Versión 4	17.48	18.3	11.43	52.72	50	-3.44	-0.35%

4.2.2 Cryptocurrencies

The values of Bitcoin fluctuate at much higher levels compared to stocks. Due to this, although the "Utility" values are higher, the "% Utility" is not significantly higher than what is obtained in the stock section.

Table 4.19 shows that the values in the "Good" category exceed 30, indicating that more than 30% of the predictions result in buying and selling operations with beneficial outcomes. Similarly, in the "Bad" category, the values are close to 30, suggesting that approximately 30% of the predictions lead to operations with unfavorable outcomes. The data in the "Null" parameter show greater variability but remain between 25 and 30. This implies that a significant proportion of the predictions result in operations where gains and losses occur. It is noteworthy that in the "No Purchase" category, most of the values are below 10%. This indicates that the Holt-Winters model carries out buying and selling operations with high frequency. Regarding the "Confidence" parameter, most implementations exceed 50%, suggesting effective control over the predictions, beyond the random outcome comparable to flipping a coin. However, it should be mentioned that the performance of one of the methods does not reach this 50% threshold.

Table 4.20 shows that most of the analyzed methods exceed the value of 40 in the "Good" parameter. This indicates that approximately 40% of the prediction sets result in buying and selling operations with net benefits. On the other hand, most methods present values closer to 30 in the "Bad" category, suggesting that around 30% of the predictions conclude in inevitable losses.

The values recorded in "Null" are notably low, generally around 10%, indicating that only 10% of the predictions result in a balance between gains and losses. This can be interpreted as a greater ability of the model to generate predictions that lead to either clearly favorable or unfavorable outcomes, rather than a neutral balance.

Consistently low values are observed in "No Purchase," suggesting high activity in terms of

buying and selling operations by the model. With the exception of one method, the values in "Confidence" exceed 55%, demonstrating a higher degree of control in the predictions, with more than 55% of the operations resulting in profits. It is relevant to note that the same model that does not reach this threshold in "Confidence" is the same one that performs poorly in the previous table, which might indicate its lower reliability compared to the other analyzed methods.

Table 4.21, focusing on the SARIMAX model, shows low values in the "Good," "Bad," and "Null" categories. This trend is attributed to the high values recorded in "No Purchase," indicating the model's predisposition to refrain from carrying out buying and selling operations. Despite this tendency towards less transaction activity, the "Confidence" values are notably high, except in cases where the "No Purchase" parameter is lower. This suggests that although the model performs buying and selling operations less frequently, more than 50% of these operations are beneficial. Furthermore, it is observed that the "% Utility" is considerably low, never exceeding 1%. A particularly notable aspect is that the most effective methods correspond to those where the model performed fewer buying operations. This suggests that this version of the SARIMAX model achieves more optimal results when adopting a more conservative approach in its market operations.

Table 4.22 shows greater variability in the first three parameters, though the "No Purchase" values continue to be predominant. Examining the "Confidence" parameter, it is observed that most of the values either do not exceed 50% or just equal it. This indicates that the model does not offer high reliability in guaranteeing a high success rate in buying and selling operations.

Table 4.19: Results with Holt-Winters Model Z.

Holt-Winters Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	36.11	30.56	25	8.33	54.55	1,437.02	5.28%
Exogenous Ethereum	33.33	28.7	30.56	7.41	51.43	1,374.51	4.68%
Exogenous Volume	31.48	31.48	25.93	11.11	47.59	525.68	1.75%
Exogenous with 2	35.19	25.93	29.63	9.26	52.6	866.52	3.05%

Table 4.20: Results with Holt-Winters Model C.

Holt-Winters C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Univariate	41.67	30.56	17.59	10.19	55.3	1,413.54	5.06%
Exogenous Ethereum	48.15	33.33	12.04	6.48	57.78	1,354.21	4.45%
Exogenous Volume	37.96	40.74	9.26	12.04	49.18	760.73	2.38%
Exogenous with 2	43.52	34.26	11.11	11.11	55.56	1,158.52	3.92%

Table 4.21: Results with SARIMAX Model Z.

SARIMAX Z							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Crypto	10.19	6.48	6.48	76.85	54.9	158.95	0.53%
Exogenous Volume	22.2	23.15	18.52	36.11	47.83	-581.95	-1.94%
Exogenous with 2	13.89	8.93	9.26	68.52	57.14	249.13	0.85%

Table 4.22: Results with SARIMAX Model C.

SARIMAX C							
Method	**Good**	**Bad**	**Null**	**No Purchase**	**Confidence**	**Utility**	**% Utility**
Exogenous Crypto	28.7	21.3	11.11	38.89	56.18	50.03	0.31%
Exogenous Volume	30.56	40.74	3.15	5.56	46.26	247.93	0.79%
Exogenous with 2	21.3	22.22	9.26	47.22	50.67	140.65	0.45%

4.3 Results of the Updated Version, Phase 3

The results obtained in phases 1 and 2 demonstrate the viability of implementing the proposed models profitably. Additionally, throughout the development of this document, further large-scale simulations were conducted. These simulations aimed to theoretically determine the effectiveness of the models over specific periods.

During the experiments, four different versions were developed to stabilize the models and achieve more consistent and reliable results. Among these, particularly notable are the results of two versions.

The first version of the model proved capable of reaching a theoretical utility of 10-16% over a period of 5-6 months, maintaining a stability of 60%. This means that in 60% of the occasions the model was applied, results were sufficiently reliable to make investments. It is important to note that the remaining 40% does not imply errors in the model but rather that on these occasions, the model did not meet the established minimum prediction standards, and therefore, no investments were made..

The second version of the model showed a theoretical utility of 5-9% over a period of 3 months, though its stability was considerably lower, at 30%. This implies that in 70% of the occasions, the model did not recommend making investments because it did not reach the established minimum prediction standards.

Chapter 5

Conclusions

This study focused on evaluating the effectiveness of various models for predicting Bitcoin values and Amazon stocks. The main goal was to offer a viable alternative to address the challenges faced by individuals interested in entering the world of investments. To achieve this goal, a methodology was implemented that included the evaluation of several methods: Holt-Winters, Prophet, LSTM, SARIMA, and SARIMAX. Each of these models was used to analyze the time series corresponding to Amazon and Bitcoin, aiming to make accurate predictions about their values.

In this way, the main challenge faced by those interested in entering the investment world is addressed: preparation and obtaining relevant information. In this context, the implemented models take on the responsibility of studying and analyzing the historical values of stocks and cryptocurrencies, generating predictions based on this data. This eliminates the need for investors to have extensive prior knowledge of these assets or advanced training in probability and investments areas.

Throughout this research, it was observed that the Prophet and SARIMA models were not suitable for the specific needs of this study. These models yielded results with low accuracy, and generally, their predictions proved to be incorrect. This limitation highlights the importance of selecting the appropriate model for this type of analysis.

As for the LSTM model, notable potential was identified, positioning it as one of the most

effective and accurate for prediction in this field. However, its main limitation lies in the high computational demand it requires. Additionally, the vast amount of possible hyperparameter combinations, each with a range of different values, significantly increases the complexity of its implementation. These limitations prevented a deeper exploration of the LSTM model in this study. Nonetheless, it is crucial not to discard this model, as its potential for making accurate predictions is considerably high.

Finally, the models that proved to be most satisfactory in terms of results were Holt-Winters and SARIMAX. These models not only provided valid and reliable predictions but also, when evaluating their profitability, showed their potential to be effectively profitable in the context of investments. Therefore, Holt-Winters and SARIMAX stand out as valuable options to consider in making predictions in this field. It is important to highlight that, although the SARIMAX model also presents a wide range of possible combinations of hyperparameter values, these do not reach the level of complexity observed in the LSTM model. This allowed for its more detailed analysis in our research. However, there was a significant limitation in experimenting with SARIMAX due to the need to restrict the number of values that each hyperparameter could adopt. Therefore, the positive results obtained with SARIMAX represent only a fraction of the model's full potential.

Therefore, this work successfully achieves the proposed objectives, demonstrating the utility and relevance of the studied models in solving the challenges faced by individuals wishing to enter the world of investments. The results obtained underscore the potential of these analytical tools to facilitate more informed and strategic investment decisions.

Additionally, although it was not the main focus of this study, a notable relationship was observed between Amazon and Microsoft stocks, showing similar behavior patterns over extended periods. Regarding cryptocurrencies, this trend was even more evident between Bitcoin and Ethereum, which exhibited parallel behaviors over significant periods, which is particularly notable given the known volatility of these digital currencies.

5.1 Future Research Directions

The future steps of this study will involve a thorough evaluation of LSTM and SARIMAX models, leveraging their potential to the fullest by accessing the appropriate computational resources for their training and analysis. This will be followed by an extended experimentation phase, simulating several months of activity to yield more robust and reliable results crucial for assessing their practical applicability. Subsequently, models that meet stringent criteria will be selected for implementation in real market environments with actual monetary investment, mirroring the same time interval used in the experimental phase. This approach will enable a direct comparison between theoretical outcomes and those actually obtained, serving as the definitive test of the proposed models' viability.

Bibliography

[1] Villavicencio, J. (2010). Introducción a series de tiempo. Puerto Rico.

[2] Hurtado, C., & Ríos, G. (2008). Series de tiempo. Santiago: Universidad de Chile.

[3] Sarmiento, E. M. (2008). Predicción con series de tiempo y regresión. Panorama, 2(4), 36-58.

[4] Juárez, A. C., Zuñiga, C. A., Flores, J. L. M., & Partida, D. S. (2016). Análisis de series de tiempo en el pronóstico de la demanda de almacenamiento de productos perecederos. Estudios Gerenciales, 32(141), 387-396.

[5] Garrido-Jarrín, E., Quinaluisa-Molina, M., & Talavera-Carvajal, C. (2020). Trading en el mercado financiero. Journal of finance, 2(1), 17-26.

[6] Peris Tena, F. (2022). Modelos de predicción en series temporales: Un estudio comparativo entre métodos estadísticos y machine learning.

[7] Jaimes Campos, D. L., & López Zúñiga, E. (2021). Modelo de Forecast para predecir la demanda semanal de alimentos y bebidas de consumo masivo.

[8] Fang, Y. (2014). Walking forward and backward: Towards graph-based searching and mining. University of Illinois at Urbana-Champaign.

[9] RPUBs - Train_Model.Time-series Final. (s. f.). https://rpubs.com/joperezpe/series

[10] Herrera, L. J., Pomares, H., Rojas, I., Guillén, A., & Rubio, G. Modelado Recursivo de Series Temporales Utilizando Modelos TSK Avanzados.

[11] González, I., & Jiménez, J. M. (2003). Predicciones de la variación del tipo de cambio con Redes Neuronales: Rolling versus Recursividad.

[12] Rodrigo, J. A. (s. f.). Introduction to Forecasting - SKForecast Docs. https://joaquinamatrodrigo.github.io/skforecast/0.4.3/quick-start/introduction-forecasting.html

[13] Konecny, L. (2022). Acciones y Bolsa: el único Libro que necesitas. BoD–Books on Demand.

[14] Bitcoin, N. S. (2008). Bitcoin: A peer-to-peer electronic cash system

[15] Nakamoto, S. (2009). Bitcoin. A peer-to-peer electronic cash system, 21260.

[16] Blockchain ¿Qué es? Neetwork - escuela de negocios digitales. (s. f.). Neetwork - Escuela de Negocios Digitales. https://neetwork.com/blockchain-que-es/

[17] Dot CSV. (2021, 23 may). Hoy Sí Vas a Entender Qué Es el Blockchain - (Bitcoin, cryptos, NFTs y más) [Vídeo]. YouTube. https://www.youtube.com/watch?v=V9Kr2SujqHw

[18] Larranaga, P., Inza, I., & Moujahid, A. (1997). Tema 8. redes neuronales. Redes Neuronales, U. del P. Vasco, 12, 17.

[19] López, R. F., & Fernández, J. M. F. (2008). Las redes neuronales artificiales. Netbiblo.

[20] Izaurieta, F., & Saavedra, C. (2000). Redes neuronales artificiales. Departamento de Física, Universidad de Concepción Chile.

[21] Navratil, M., & Kolkova, A. (2019). Decomposition and forecasting time series in the business economy using prophet forecasting model. Central European Business Review, 8(4), 26.

[22] Elshabrawy, M., Eid, M. M., Abdelhamid, A. A., El-Kenawy, E. S. M., & Ibrahim, A. (2023, October). Forecasting of Monkeypox Cases Using Optimized SARIMAX Based Model. In 2023 3rd International Conference on Electronic Engineering (ICEEM) (pp. 1-6). IEEE.

[23] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7), 1235-1270.

[24] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

[25] Chatfield, C. (1978). The Holt-winters forecasting procedure. Journal of the Royal Statistical Society: Series C (Applied Statistics), 27(3), 264-279.

[26] Kalekar, P. S. (2004). Time series forecasting using holt-winters exponential smoothing. Kanwal Rekhi school of information Technology, 4329008(13), 1-13.

[27] Wang, J., Xue, M., Culhane, R., Diao, E., Ding, J., & Tarokh, V. (2020, May). Speech emotion recognition with dual-sequence LSTM architecture. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6474-6478). IEEE.

[28] Guía paso a paso sobre el trading de opciones Binance — Blog de Binance. (s. f.). Binance Blog. https://www.binance.com/es-LA/blog/futures/gu%C3%ADa-paso-a-paso-sobre-el-trading-de-opciones-binance-9103837153565975386