

MSc Computer Science  
Final Project

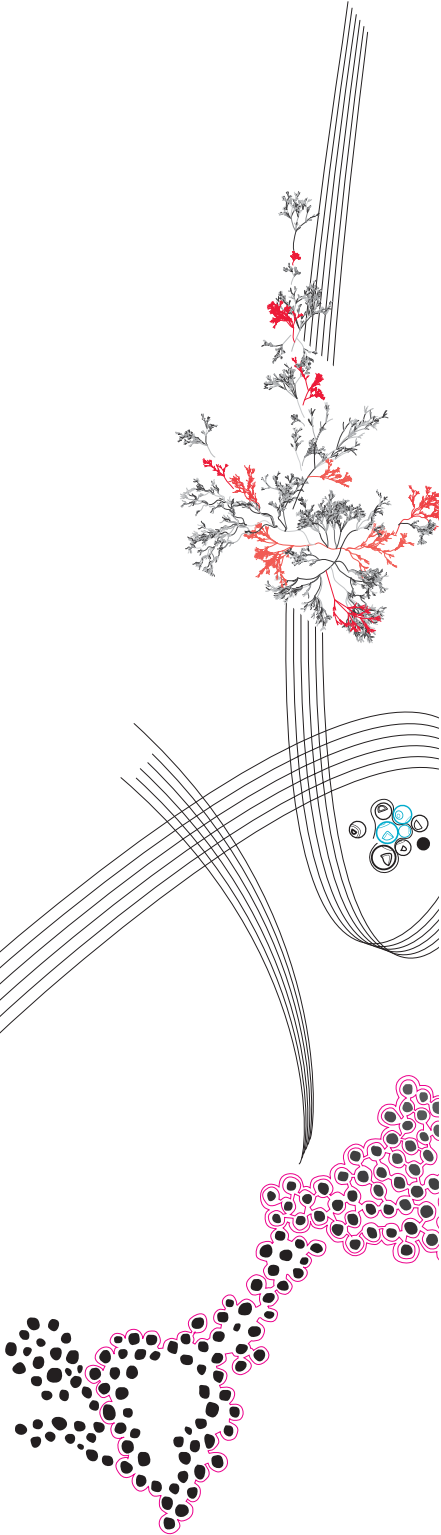
**Temporal Aspects of Stock Price Prediction:  
Quantifying the Role of Historical Data using  
Partitioned Dynamic Bayesian Networks**

Cristian Verdecchia

Supervisors: Peter Lucas, Jörg Osterrieder

October, 2024

Department of Computer Science  
Faculty of Electrical Engineering,  
Mathematics and Computer Science,  
University of Twente



# Contents

## Abstract

<b>Acknowledgments</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Introduction to Stock Market Analysis . . . . .	2
<b>2 Related Research</b>	<b>5</b>
2.1 Random Walk and Efficient Market Hypothesis . . . . .	5
2.2 Behavioural Finance in the Efficient Market Hypothesis . . . . .	5
2.3 Consequences of the Efficient Market Hypothesis . . . . .	6
2.4 Different Trading Paradigms . . . . .	6
<b>3 Background</b>	<b>8</b>
3.1 Time series Approaches . . . . .	8
3.2 Bayesian Networks Approach . . . . .	9
3.3 Data Usage . . . . .	11
<b>4 Preliminaries</b>	<b>12</b>
4.1 Data . . . . .	12
4.1.1 Types of Market Data . . . . .	12
4.2 Partitioned Dynamic Bayesian Networks . . . . .	14
4.2.1 Directed acyclic graph . . . . .	14
4.2.2 Bayesian Networks . . . . .	14
4.2.3 Dynamic Bayesian Networks . . . . .	17
4.2.4 Learning a Bayesian Network . . . . .	19
4.2.5 Partitioned Dynamic Bayesian Networks . . . . .	28
<b>5 Methodology</b>	<b>31</b>
5.1 Data Extraction and preparation . . . . .	31
5.1.1 Retrieval of the S&P 500 constituents . . . . .	31
5.1.2 Retrieval of historical data . . . . .	32
5.1.3 Retrieval of sectors data . . . . .	34
5.2 Application of Partitioned Dynamic Bayesian Networks . . . . .	34
<b>6 Results</b>	<b>40</b>
<b>7 Discussion</b>	<b>60</b>
7.1 Tool Selection and Transition to R . . . . .	60
7.2 Data Handling and Factor Representation . . . . .	60

7.3	Tested Modeling Approaches . . . . .	61
7.4	Final Approach and Computational Improvements . . . . .	61
7.5	Prediction Methods and Execution Time . . . . .	62
7.6	Influence of Historical Trends on Predictions . . . . .	63
<b>8</b>	<b>Conclusions and Future Developments</b>	<b>70</b>
8.1	Final Model and Prediction Approach . . . . .	71
8.2	Results Analysis . . . . .	71
8.3	Limitations . . . . .	72
8.4	Future Work . . . . .	73
	<b>Appendices</b>	<b>74</b>
	<b>A Results of Executions with Structure Learning</b>	
	<b>B Results of Executions Without Structure Learning</b>	

## Abstract

Financial markets, characterized by their perpetual evolving nature, have long been an active research field. With new artificial intelligence technologies emerging, and the increasing availability of data, there is a growing need to uncover the relationship between historical data and actual predictions. Despite the abundance of studies in this area, few have investigated Bayesian Networks, which are traditionally perceived as unsuitable for financial forecasting due to their inherent complexities. Despite this, we believe that the Standard and Poor's (S&P) 500 index, known to be a good reflection of the status of the American economy, presents an ideal case study for this research. Comprising a vast amount of companies across various sectors, the S&P 500 offers a rich dataset for analysis. Despite their complexities, the explainability that Bayesian Networks (BNs) inherently bring, could in the future, provide important information on hidden market dynamics. Leveraging Partitioned Dynamic Bayesian Networks (PDBNs), originally designed for health-related data, provides a unique opportunity to test the evolving behavior of the market. Their ability to change in structure provides a great advantage over the more traditional Dynamic Bayesian Networks (DBNs) when dealing with this type of data. In this thesis, great emphasis was dedicated to acquiring high-quality data tailored for this approach. Multiple networks were created capturing various aspects of PDBNs through which the effects of historical data on predictions were analyzed. The models were compared to evaluate their performances where the final employed model provided a good balance between specificity and sensitivity.

*Keywords:* Partitioned Dynamic Bayesian Network, Bayesian Network, Machine Learning, Quantitative Finance, Stock Price Prediction, Financial Forecasting

# Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Peter Lucas. His invaluable guidance, support, and availability, even during his holidays, have been essential to the successful completion of this project. I am also grateful to my second supervisor Jörg Osterrieder, for introducing me to this field and sharing his technical expertise.

I would also like to express my heartfelt appreciation to my family, who, despite the distance, offered constant encouragement and emotional support on countless calls. Without your unwavering belief in me and your sacrifices, this project would have never been possible. A special thanks to my cousin Samuel, for his patience and for always being willing to listen to my endless explanations and ideas.

My gratitude goes also to my far friends, and close ones - thank you for enduring my endless complaints and occasional meltdowns in the past years over a countless number of coffees that fueled my energies and spirit.

Last but definitely not least, I would like to thank my colleagues Carlos and Ruben for the flexibility given and for accommodating my many requests throughout this period, allowing me to achieve this result.

# Chapter 1

## Introduction

### 1.1 Introduction to Stock Market Analysis

Stock price prediction is one of the most challenging tasks in the financial domain due to the amount of variables involved in this process. This complexity is further amplified when combined with the inherent volatility and non-stationarity of the data in financial markets. Traditional time series models, renowned for their ability to model time dependent variables, often struggle with these challenges, leading to suboptimal predictions. This research explores the application of Partitioned Dynamic Bayesian Networks (PDBNs) in improving the accuracy of stock price forecasting, specifically in the context of predicting financial indexes, where an evaluation of the historical data and how the amount used affects the prediction is conducted.

In the 1960s Eugene Fama proposed a theory asserting that in financial markets the price of a security reflects the information available at any given time. This theory became known as the Efficient market hypothesis (EMH). Despite its success, this hypothesis faced a lot of opposition. Notably, Taylor (1982) [42] challenged a theory defining that prices cannot be predicted since they follow a random path. This theory known as the Random Walk Hypothesis (RWH), is considered a component of the EMH. Taylor demonstrated that previous statistical tests done to test this theory were inadequate. When Taylor applied more robust statistical methods, empirical results indicated a rejection of the random walk hypothesis. This ongoing debate [47] highlights the complexity and unpredictability of financial markets, displaying the need for sophisticated predictive models.

Given the implications of those theories, we can say that non-stationarity and noise are essential aspects to take into account. Non-stationarity refers to the inconsistency of statistical properties like mean and variance, over time. This inconsistency is common in stock market data where long-term trends are prevailing. For instance, market indexes like the S&P 500 have exhibited a consistent upward trend over decades, contrasting with the varied short-term patterns and downward trends. In the study by Liu [26], the dynamics of stock price changes - defined as  $S(t)$  - are characterized as a stochastic process. The immediate return of a stock, denoted as  $R(t, t_\delta)$ , over a brief time interval, is presumed to follow a normal distribution. This assumption implies that the stock price follows a log-normal distribution in the short run. However, as our view is extended to the long-term behavior of the stock market, it diverges from the simple patterns of short-term movements exhibiting more complex dynamics.

Traditional forecasting models typically assume data stationarity, which simplifies the mathematical complexity but often leads to overfitting and poor future predictions due to the incapacity to deal with evolving patterns. These models may perform well on

historical data but usually struggle to adapt to new data exhibiting different statistical properties. This thesis addresses these issues by employing PDBNs, which offers a novel approach to modeling time-series data. PDBNs partition the data into segments that can be independently modeled through the use of DBN, thereby improving the effect of non-stationary behaviors on the data and enhancing predictive performances.

The importance of this research lies in understanding how Bayesian networks designed to address specific problems, particularly those involving temporal data, can improve the current state of the art in stock price prediction. By exploring the temporal aspect of PDBNs, and thanks to their inherent explainability, we can make more precise assumptions about market behavior, leading to better performance in predictive tasks. The analysis of the effects of historical data on predictions, despite being tailored to this type of networks, can gain insights that can have broader implications, potentially benefiting other fields where the time domain is a critical aspect.

## Motivation

Stock markets are entities that mediate between traders and that allow for companies' shares to be exchanged. Two parties, buyers and sellers, are always involved in the market, the higher the demand compared to the offer, the higher the price of a certain asset, building wealth for investors. Since the beginning of the stock market, investors tried to find ways to "beat" the market and obtain higher returns compared to the performance of the reference index. Over the decades many techniques were tested and machine learning methods proved somewhat useful to increase returns. In the current era of data and machine learning-driven approaches, there is a growing demand for more and better performing techniques that can achieve higher returns. In this regard, time series approaches, renowned for their capacity to identify patterns that evolve over time, are often employed. This is proved by the literature that is rich in evidence of their effectiveness in many contexts when dealing with unimodal data distributions, where data displays a single pattern or trend. However, applying these kinds of models to financial time series does not come without its drawbacks. Inherited in their mathematical background, time series methods mostly fail to grasp the behavior of multimodal distributions, a common characteristic of most financial data. In this context, data is influenced by a multitude of factors, leading to different patterns. Specifically, the additional complexity could be caused by geopolitical events, news, economic indicators, the market sentiment that can itself be affected by news, the macroeconomic situation, and other complex factors. Each of these aspects can modify trends by changing what initially could look like a unimodal distribution into a multimodal distribution. Exploring the complexities of financial data necessitates sophisticated modeling approaches. Bayesian networks have emerged as a viable solution, being able to better cope with multimodal distributions. In particular, DBNs, an extension of the more classic BNs, were found to be better able to capture temporal dependencies compared to BNs. In the financial domain, the variable of time is a fundamental factor that exponentially increases the complexity of data by creating new patterns and trends. Despite the improvements obtained in this context, DBNs are not without limitations. Their inability to model time-varying structures, especially in a domain where temporal dynamics are crucial, poses a big constraint. This architectural gap is the motivation for exploring PDBNs. PDBNs introduce the concept of time partitioning, instead of using a single network structure for all time steps, PDBNs dissect time into discrete intervals or partitions. Each partition represents a specific time period during which the relationships between variables are considered relatively constant. This innovative structure is designed to handle non-homogeneous data, characterized by processes evolving over time. Every

partition, distinguished by its distinct time frames, should facilitate the stock analysis. This approach is well-suited to manage the variability and swift changes characteristic of financial markets.

### **Thesis Structure**

We begin this work with a literature review that outlines the existing research, starting from time-series approaches and proceeding to Bayesian networks and their applications in the financial sector. Next, we present the methodology section, which begins by introducing fundamental technical concepts. This section includes a description of the types of data in the stock market, followed by an explanation of Bayesian Networks (BNs), Dynamic Bayesian Networks (DBNs), and the foundational blocks for constructing these networks that are parameters and structure learning. These concepts are foundational for understanding the functioning of Partitioned Dynamic Bayesian Networks (PDBNs) that are the primary model used in this research. Following the methodology, we discuss the process of extracting and processing the data, as well as the construction of the Partitioned Dynamic Bayesian Networks (PDBN). The results of the conducted experiments are then provided. Finally, a conclusion and discussion are given where those results are discussed together with the key findings of the project.



## Chapter 2

# Related Research

### 2.1 Random Walk and Efficient Market Hypothesis

The study of stock market behaviors has a longstanding history. In 1965, after his first publication of the efficient market hypothesis, as explained at the beginning, Fama brought attention to an intriguing concept known as the Random Walk Hypothesis (RWH) [19], a notion originally introduced by Louis Bachelier in 1900 [9]. Bachelier defined this theory by comparing the movement of stock prices to a random walk, like that of a drunk man. In essence, this theory posits that the consistent prediction of stock movements is impossible given the randomness of the stock market behaviour. Note that it is important not to confuse the two topics, while the RWH describes the unpredictability of stock price movements, the Efficient market hypothesis (EMH) asserts that prices of a certain security already reflect all available information. Over the years many doubts surged over the EMH, in 1978 Ball [10], after examining previous studies on the correlation between earnings announcements and the stock price, found abnormal results in the price adjustments post release when compared to Fama's, in fact returns in the period post announcement are non-zero. The explanation that Ball gives to this behaviour is the inadequacy of the test used to compute the returns, and a more powerful asset pricing model could instead provide a better overview of the actual returns.

### 2.2 Behavioural Finance in the Efficient Market Hypothesis

Subsequent years saw an increasing influx of research dedicated to evaluating the validity of the Efficient market hypothesis (EMH). In the 1990s we can distinguish a pivotal moment when Richard Thaler, an advocate of the behavioral finance paradigm - a field that explores the effects of human psychology to understand how this is reflected in the movements of the stock market - unveiled his work in the book *Advances in Behavioral Finance* [43] where he delineated the basis of the behavioral finance in the context of the efficient market hypothesis. In his work, Thaler wasn't aiming to counter the EMH but rather expand upon it. He delved into how different types of information influence investors with obvious results being reflected in the stock market. For this reason, he categorizes the levels of market efficiency based on the extent of information that is believed to be already incorporated into asset prices, the three proposed variants are: weak, semi-strong, and strong form. The principles under which those are based, have to do with the rationality of the people trading in the stock market. In the weak form, it's recognized that investors can sometimes act irrationally, leaving some space for market inefficiencies. However, this behavior doesn't last long thanks to other players in the market playing an important role

in correcting these misalignments at some points in the future. On the other hand, the semi-strong form asserts that systemic issues exist, this causes prices to frequently and significantly deviate from correct levels for extended periods. The strong form escalates this perspective, suggesting that there is little correlation between the security prices and the company's underlying financial performance. Instead, it suggests that price movements are largely influenced by the overall sentiment and current market trends. In the following years, the debate surrounding the EMH and behavioral finance continued. Although a few more papers on the topic were published, no definitive conclusion was reached.

## 2.3 Consequences of the Efficient Market Hypothesis

The presence of a role between human psychology and market dynamics is evident where both individual and collective psychological behaviors, including those affected by new information, have an impact on the fluctuations of the markets. Understanding these dynamics is crucial for developers aiming to use machine learning models to make future predictions about market movements. In a Utopian world where markets are fully efficient, and promptly and accurately react to every news and information available, predicting future prices wouldn't be possible. In such a scenario, every variable that could influence the markets' valuations and price changes would instantaneously be accounted for, making predictions redundant and unnecessary.

However, the reality is far different from this. In the real world, even in the most efficient markets, the efficiency would not be absolute. As also described by Fama [20], abnormalities may exist in the short term but those tend to disappear in the long term, bringing with them the ability to predict future movements, which, in the worst-case scenario are equivalent to short-term predictions. Even in such a scenario, an additional challenge persists due to the nature of the financial markets which is always changing. While a machine learning model may effectively predict market trends in the short term, its efficiency shrinks over an extended period due to the markets and players' constant evolution. This phenomenon is also described by López de Prado in his "Advances in Financial Machine Learning" book [28]. He underscores the necessity to iterate over different models, frameworks and types of data to be able to be competitive in the market. Over time financial strategies decay and must be abandoned when their performance consistently falls below anticipated benchmarks.

In essence, the task of forecasting market movements necessitates an ongoing effort, being able to create a model that is better able to take into account the time domain is a crucial feature.

## 2.4 Different Trading Paradigms

Within the automated financial trading landscape, various specialized tasks can be distinguished where each offers different approaches. As delineated in [34], four main categories can be defined: quantitative trading, algorithmic trading, Automated Trading System (ATS), High-Frequency Trading (HFT).

Quantitative trading makes use of complex mathematical models to identify trading opportunities. Algorithmic trading does not employ complex models but usually uses rules defined by the users based on time, price, volume, and other metrics that are usually provided by the trading platforms. These sets of rules allow to automate trades to minimize human errors and avoid overreacting due to emotional involvement. ATS extends these types of automations, making use of algorithms managed by software to issue and execute

---

market orders with usually no human intervention needed. HFT is a version brought to its extreme of an ATS characterized by elevated speed in retrieving and processing data, allowing the execution of a vast number of orders within seconds. Each of the defined category serves a different role, in the context of this thesis we will be mostly focusing in ATS.

## Chapter 3

# Background

The use of Bayesian Networks (BNs) in the financial sector is relatively rare. Despite the broad application of Bayesian and Dynamic Bayesian Networks (DBNs), thanks to their explainability and ability to model complex relationships, their application to "simple" stock price prediction tasks is limited. Traditional Bayesian Networks are not typically considered to be effective for this purpose. While there is more extensive research involving DBNs, to the best of my knowledge, no studies have yet applied Partitioned Dynamic Bayesian Networks (PDBNs), as defined by Bueno [11], to stock price prediction.

In this section, we will provide the background context necessary to understand the reasoning behind this project. We will begin by explaining the EMH and how we can possibly trade in the markets. Following, we will introduce time series approaches for stock price prediction and continue with reviewing research involving Bayesian Networks (BNs) in the financial field, including stock price prediction.

### 3.1 Time series Approaches

As mentioned earlier, the time domain analysis is a crucial aspect in the field of stock price prediction. Over the years, multiple time series algorithms have been employed to analyze stock price data and among these algorithms, the AutoRegressive Integrated Moving Average (ARIMA) model has gained popularity and demonstrated its efficacy across diverse domains, as evidenced by several studies [14, 44, 30]. Notably, the ARIMA model has also found application in the specific task of stock price prediction [8] where historical Open, High, Low, Close, Volume (OHLCV) data from both the New York Stock Exchange (NYSE) and the Nigeria Stock Exchange (NSE) were employed. This model makes use of a combination of three different ideas, notably, the Auto Regressive (AR), Integrated (I), Moving Average (MA). The AR is a representation, part of the model, that outputs a value obtained as a linear combination of the previous values. The 'Integrated' part denotes a crucial component that represents the transformation process to obtain stationarity in a time series dataset. This implies that statistical properties such as mean and variance remain invariable over time. This transformation is often obtained by computing the difference between the raw observations. Finally, the MA accounts the prediction errors of the past stock prices to smooth the prediction. This process is fundamental to avoid outliers in the data that could be obtained for multiple reasons (for instance, at the opening of the stock market all the orders that were waiting are processed at a faster rate, this creates periodical high fluctuation in the price). In addition to the ARIMA model, in literature, multiple other time-series models were employed for stock price prediction. Generalized Autoregressive Conditional Heteroskedasticity (GARCH) is one of such models. GARCH

as used in [22] is a statistical model that helps identify how the volatility of stock returns changes over time. In ARIMA, while the moving average component can help mitigate the impact of high volatility to some extent, it may not fully capture time-varying volatility patterns. In financial time series this has a high degree of importance since the volatility, as already said, is not constant and can fluctuate based on the time period. By being able to evaluate the fluctuations, GARCH is also able to enhance the accuracy of the predictions.

## 3.2 Bayesian Networks Approach

### Application of Bayesian Networks in the Financial Domain

Moving towards machine learning approaches in the financial domain that made use of Bayesian Networks (BNs), we find the research conducted by Liu et al. (2021) [27]. This research explores the relationship between China's macroeconomy and its stock market through the application of Dynamic Bayesian Networks (DBNs). The study employs Gaussian Bayesian Networks (GBNs) - a variant of BNs - that makes use of logarithmic rates of return for various indexes. The rate of return indicates the growth - expressed as a percentage - of a certain asset where the return is compounded over time. By analyzing data from different sectors over the period 2007 to 2020, the researchers investigated the relationships between the selected indexes and other macroeconomic indicators, such as the export growth rate and the Industrial Added Value (IAV) growth rate, which measures the industrial sector's contribution to the Gross Domestic Product (GDP) through its production processes. The study makes use of a sliding window approach, where per each month, a network is constructed based on data from the previous 12 months. This approach helps in tracking the evolution of the relationship between China's macroeconomy and the stock market over time.

Fonseca and Carvalho (2021) [21] employ DBNs to model the dependencies between different sectors within the U.S. economy and investigate how these relationships are affected during financial crises. Specifically, they analyze the so-called contagion effect, that is the correlation between sectors in terms of how each sector impacts each other positively or negatively. In their research, they employed a Dynamic Bayesian Network (DBN) where each node represents the index of a specific economic sector from the Dow Jones Industrial Average (DJI) indices such as insurance, real estate, and oil and gas. In the network, each node represents the daily log returns of a sector. The study found that certain sectors — like Oil and Gas and Real Estate — propagate their effects (downturns or upturns in the market) more frequently than other sectors.

### General Application of Bayesian Networks in Stock Price Prediction

As also highlighted in the motivation section (chapter 1.1), traditional time series models encounter significant issues when applied to financial data. This limitation derives from their inherited mathematical structure that is not well-suited for multimodal data distributions nor for data that follow an evolving trend. On the other hand, Bayesian networks have often been employed as a good substitution for time-series algorithms. Zuo and Kita in their paper "*Stock price forecast using Bayesian network*" [48] compare the Bayesian network method to time-series algorithms such as AR, MA, AutoRegressive Moving Average (ARMA) and Autoregressive Conditional Heteroskedasticity (ARCH) models. The comparison involves evaluating forecast accuracy and correlation coefficients, demonstrating the advantages of the Bayesian network approach over the mentioned traditional time-series methods for stock price forecasting. Despite its advantages, Bayesian Networks are

not able to cope with continuous variables, in the mentioned paper both uniform clustering and the Ward method were employed as clustering algorithms to discretize the continuous variables. As demonstrated, the clustering algorithm plays a crucial role in obtaining an accurate prediction. In this case the Ward algorithm was proved to improve the accuracy by 15% on the NIKKEI (an index reflecting the Japanese stock market) stock average and 20% on the Toyota sock price when comparing it to the uniform clustering method.

### Dynamic Bayesian Networks and Hierarchical Models

Dynamic Bayesian Networks (DBNs) extend the more traditional Bayesian Networks (BNs) by incorporating temporal dynamics, making them particularly suitable for time-series data like stock prices.

A study by Jangmin et al.[23] proposes a three-level Hierarchical Hidden Markov Model (HHMM) - a variant of DBNs - to predict stock prices and generate trading signals based on the predicted values. In this kind of structure, each level has a different role. The first level categorizes the trends into five states: strong bear, weak bear, random walk, weak bull, and strong bull, where a bear pattern is considered to be a market following an uptrend and a bull one is a market following a downtrend. The second layer provides a higher-level categorization of the trends derived from the first layer. Finally, the third layer contains Hidden Markov Models (HMMs) that are responsible for emitting observable outputs. Training the HHMM involves a semi-supervised learning approach. The trend states in the first level are manually labeled based on the gradient of the moving average of stock prices. The Expectation Maximisation (EM) algorithm is then used to refine the model parameters. Their HHMM-based system uses the probability distributions of different trend states to generate trading signals. For example, a buy signal is generated if the combined probability of the strong bull and weak bull states exceeds a certain threshold, and a sell signal is generated if the combined probability of the strong bear and weak bear states falls below a threshold. Experimental results were conducted on 20 companies from the Korean stock market and showed that the proposed HHMM outperformed the Triple Exponential Average (TRIX) technical indicator, which identifies when a certain market is overbought or oversold, particularly in minimizing losses during market downturns.

In its paper Duan (2016) [17] introduces a new model for stock price prediction called Auto-Regressive Dynamic Bayesian Network (AR-DBN). This model improves on traditional Dynamic Bayesian Network (DBN) by adding connections between consecutive observed stock prices, which help to better understand the trends in the stock market prices.

In a traditional DBN, each observed stock price is only influenced by hidden variables that represent underlying factors. However, in reality, stock prices are also influenced by their previous values. The term "auto-regressive" means that the model takes into account that today's stock price is partly determined by past stock prices. The AR-DBN model adds connections between consecutive observed stock prices, allowing the prediction of future prices to be influenced by past ones. The model uses the Expectation Maximisation (EM) algorithm to find the best parameters for these connections. Experiments executed using historical Standard and Poor's (S&P) 500 data show that AR-DBN provides better predictions and faster convergence compared to traditional DBNs.

Shen and Winstanley (2019) [38] in their project propose using Dynamic Bayesian Networks (DBNs) to model and predict the behavior of prediction markets which are platforms where participants trade securities based on the outcomes of public events like presidential elections. This kind of market aggregates public opinion given the information provided by the polling prices. For example, the price trend for a bet in the case of presidential elections might reflect the public opinion of the candidates or their chances of

winning.

The authors use DBNs to model the political climate and predict future outcomes, incorporating both hidden variables (political climate) and observed variables (polling data) represented by the polling prices. The model is validated using data from the 2012 U.S. presidential election, showing that it outperforms the baseline models used, by accurately capturing market trends and handling noisy data. This research does not make use of stock market data. However, the type of data used and the structure employed have a structure similar to financial data and their applications.

In the study by Malagrino [33], the authors evaluate the influence of global stock market indices on Brazil's primary index, iBOVESPA. The research proved effective in the prediction of the daily direction and also provided insights into which markets are the most influential for iBOVESPA's trend prediction. The best results were obtained when using a single index per continent. Index data was input to the network for multiple indexes (different amount of indexes were used in each combination tested) and this was used to predict iBOVESPA's trend.

### 3.3 Data Usage

The concept of forecasting financial markets dates back to the beginning of the stock markets although in the latest years thanks to new tools and methods, this process has evolved significantly and so has the amount of data used in the process. Already in 2003, Wang et al. in [45], applied artificial neural networks on financial forecasting placing emphasis on trading volume as a distinctive feature of stock market data. Their research involved two major datasets: the Standard and Poor's (S&P) 500 and the Dow Jones Industrial Average (DJI). Their results show that trading volume can sometimes be an indicator of market sentiment, however, its effectiveness can vary depending on the dataset and model. One of their primary observations was that, in general, trading volume did not significantly improve the forecasting accuracy within the scope of their study. Trading volume is just one of many data points researchers and analysts use for stock market prediction. Over time, multiple types of data have been employed, each offering its unique insights. Several instances of BNs applied in stock price prediction can be cited from past studies. In [48], Zuo made a noteworthy contribution by utilizing Bayesian networks in combination with the K2 algorithm to obtain the optimal network structure. In this study, they made use of the Price/Earnings (P/E) ratio of the NIKKEI index and Toyota Motor Corporation stock. In [23], Jangmin utilized the relative closing price built as a function of the closing price. This research showed that utilizing their approach outperforms the TRIX technical indicator showcasing the importance of using historical data as a baseline for machine learning models. A similar approach was followed by Malagrino in [33] that by comparing each day's closing price to the previous day was able to classify the data into "higher" or "lower" respectively based on the increase or decrease compared to the previous day's price. Each of these studies outlines a different perspective in terms of data used, highlighting the high degree of versatility of Bayesian networks.



# Chapter 4

## Preliminaries

### 4.1 Data

In the landscape of asset management, engineers often opt for uniqueness and innovation in their data sources and data processing techniques. It's relatively rare for them to utilize data that has been previously employed by other entities, shared data sources can lead to similar, if not identical, conclusions and insights, reducing the competitive advantage over other companies. The need for differentiation in the final outcome drives engineers towards the use of raw, unprocessed data. This strategic choice also provides them with the ability to build their own pre-processing and modeling steps. Financial data is renowned for its complexity and high degree of variance, in such context, the kind used in stock market prediction tasks can vary a lot, both in terms of type and how those are processed. In [28] De Prado divided the types of financial data into four different categories, among the most used we can find: market data, analytics and fundamental data. In addition to those, it is possible also to find alternative source of data such as Google searches trends, Twitter derived data, satellite or CCTV images and more. All of those types can be further divided into multiple categories. In table 4.1 I highlighted the main categories where each offers unique insights into market dynamics.

TABLE 4.1: Types and Categories of Data

Market Data	Analytics	Fundamental Data	Alternative Data
Price	News Sentiment	Assets	Satellite/CCTV
Yield	Earnings Expectations	Liabilities	Google Searches
Volume	Analyst Recommendations	Sales	Twitter
Dividends		Costs/Earnings	

#### 4.1.1 Types of Market Data

Market data include all the daily activities that take place in an exchange, every participant in the stock market leaves a footprint whenever exchanging a security, leading to terabytes of data being generated on a daily basis, this makes it one of the most used types of data in the context of stock price prediction due to its high availability level, low price, and amount of data that can be obtained. Within the family of market data, we can find key metrics such as the **price**, which represents the amount of money that sellers are asking for a single share of a stock and the amount that buyers are willing to pay for that same share. When discussing price, it's important to understand the different types of data we



can access. For instance, the "order table" or "order book" offers an in-depth snapshot of live market orders. Each entry in this ledger details the number of shares an entity wishes to exchange and the associated price. Such data can be relevant for predicting short term prices although more difficult to retrieve due to the amount of data processed per second. Since not everyone requires such granularity, some analysts or investors might be more interested in broader price trends over time. For the described purpose, it is possible to retrieve an alternative type of data where prices are averaged over specific time intervals (the most common ones are one or five minutes ranges). These intervals can be particularly useful when creating charts and executing technical analysis, as they are able to capture smaller price oscillations and can help highlight patterns or trends when predicting over a short interval. When retrieving prices most data providers offer not just the average price but also the volume exchanged in a specific time frame, minimum, and maximum prices. A really interesting additional metric that can be used in combination with the price, is the Volume Weighted Average Price (VWAP) which is a metric used to provide an average price to which a certain security traded during the day. As defined in eq 4.1, this metric is computed by multiplying the volume and price for each timeframe and dividing it by the total volume traded on a specific day. For instance, on a 5 minutes timeframe, the VWAP would compute the volume processed in the five minutes, times the average price in the same period divided by the cumulative daily volume. This ensures that more importance is given to timeframes with a higher exchange in terms of volume.

$$\text{VWAP} = \frac{\sum(\text{Volume} \times \text{Price})}{\text{Total Volume}} \quad (4.1)$$

Moreover, it's worth noting that in today's world, high-frequency trading systems often use really short time frames, making even minute-by-minute data seem relatively broad, which is also one of the reasons for using order tables as a source of truth for those highly advanced models that usually execute hundreds of operations in the stock market every hour. In the context of the VWAP, the metric can be re-computed at every exchange of a security.

An additional metric often used is the **Yield**. This metric computes the annual return on an investment based on the dividends a certain company pays out every year, usually represented as a percentage Year Over Year (YOY) of its current price, as defined in formula 4.2. Note that although most companies pay out dividends, not all companies do, those who pay out dividends do so to attract new investors and prove the good financial situation of the company.

$$\text{Dividend Yield} = \left( \frac{\text{Annual Dividends per Share}}{\text{Price per Share}} \right) \times 100 \quad (4.2)$$

One of the disadvantages of the price bars is the arbitrariness in the number of ticks since we consider specific time periods when retrieving the data. For instance, if we retrieve data with a time window of five minutes, we will retrieve the averaged price value in this time frame, this could lead to a loss of information due to the inability to grasp the actual behaviour of the transactions in this five minutes period. To better solve this issue Clark [13] in 1973, realised that by sampling the returns by volume instead of price, we can achieve better results. In fact, the price is averaged over the specific time frame independently from the amount of volume exchanged over that specific time period. For this reason we can

consider the **volume** a critical metric to be used for stock price prediction. The way volume works is by sampling ticks based on the amount exchanged. For instance, one could sample a tick every 1 million shares exchanged, and the price will be averaged. In this way we can overcome the issue of oversampling during low peak hours and undersampling during high peak hours. However, note that the amount of shares we sample is a critical metric that needs to be correctly and accurately defined to avoid undersampling during low peak hours that could lead to missed patterns.

## 4.2 Partitioned Dynamic Bayesian Networks

Partitioned Dynamic Bayesian Networks (PDBNs) initially presented in the paper "Understanding disease processes by partitioned dynamic Bayesian networks" [11], are a specialized variant of Dynamic Bayesian Network (DBN). These networks are specifically designed to address challenges associated with data scarcity and non-homogeneous time distributions. In the medical domain data scarcity is a common problem, a standard approach to deal with it is to stretch the available data across the entire time frame to build a comprehensive model. A common generalisation to multivariate problems makes use of DBNs for this approach. In the following chapters, we will begin with an exploration of directed acyclic graphs, which serve as the building blocks for BNs. After discussing the technical foundations, we will move on to a detailed look at how Bayesian Networks (BNs) operate. Following, our focus will move towards specialized adaptations of these networks, specifically Dynamic Bayesian Network (DBN) and Partitioned Dynamic Bayesian Networks (PDBN), to gain insights into their advanced functionalities and applications.

### 4.2.1 Directed acyclic graph

Directed Acyclic Graphs (DAGs) are a concept mainly relevant in graph theory with different fields of application. A DAG is a graph characterized by the absence of direct cycles, meaning that each edge is unidirectional, connecting one vertex (also called node) to another, in a way that does not form a closed loop. The non-cyclic behaviour is a key property and distinguishing feature of this graphs, it is crucial for certain applications since it enables the chronological ordering of the vertexes. The acyclic component of these kind of graphs is a consequence of the exclusive presence of direct edges. This structural characteristic ensures that every node can't be revisited via the same path.

### 4.2.2 Bayesian Networks

Bayesian Networks (BNs), otherwise known as Belief Networks, are a specific category of probabilistic graphical models. A common challenge in statistics is analyzing events that are inherently uncertain. A typical way to do it is through the use of random variables to which we assign different values to portray possible states of the world.

The choice of the most suitable probabilistic model to employ depends on the specific domain and nature of the data involved. BNs are particularly known for their ability to provide a detailed representation of the intricate relationships among real-world events or objects. Although HMMs have been the go-to models for tasks involving time-varying patterns, the advent of DBNs introduced a new paradigm for dealing with time-series data thanks to their ability to deal with multivariate problems.

Mathematically, a BN can be defined as follows. Given a DAG  $\mathcal{G}$  and a joint probability distribution  $P$ , a Bayesian Network  $\mathcal{B} = (\mathcal{G}, P)$  is defined as a probabilistic graphical model representing a set of random variables and their conditional dependencies. The DAG is

defined as  $\mathcal{G} = (V, E)$  where  $V$  is the set of nodes (or vertices) and  $E$  is the set of directed edges. In a BN each node  $v \in V$  represents a random variable  $X_v$  while the relationships between nodes - conditional dependencies - are identified through the use of directed edges  $(v, w) \in E$  where  $(v, w)$  denotes a directed edge from node  $v$  to node  $w$ .

These networks utilize DAGs to represent a group of variables along with the conditional dependencies between them as in fig 4.1. In a DAG nodes are linked to represents general dependencies between nodes. A Bayesian network extends this concept, each node symbolizes a random variable and the edges connecting these nodes illustrate probabilistic dependencies. These dependencies demonstrate the relationships between the corresponding random variables, while the absence of an edge represent conditional independence between random variables.

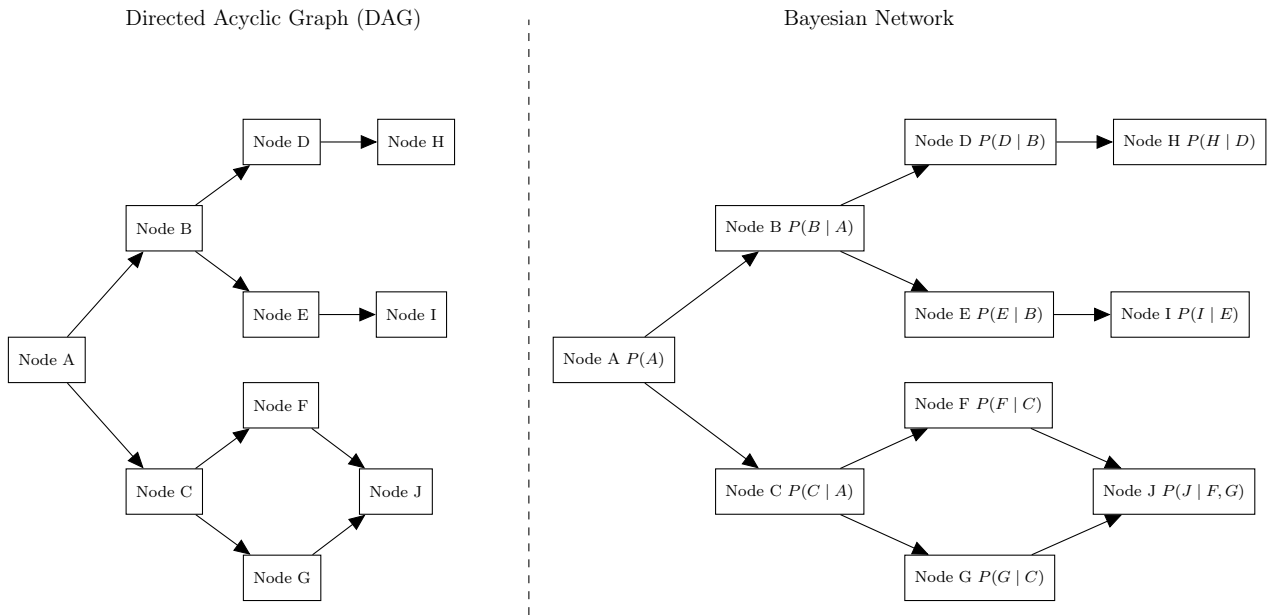


FIGURE 4.1: Directed Acyclic Graph (DAG) and Bayesian Network compared

For every node in the network, there is an associated probability function, that assigns probabilities to the various possible states of the node's corresponding variable. This probability function is typically represented as a Conditional Probability Table (CPT), which is a table outlining the probability of each possible state of the node, given every combination of states for the node's parent. The joint probability distribution in a Bayesian Network (BN) is a way to describe the probability of an outcome as a combination of the involved random variables. Computing the full joint probability distribution can sometimes be impractical due to the elevated number of combinations. To overcome the issue, Bayesian Networks make use of the conditional independence assumption represented in the network structure to factorize the joint probability distribution into a simpler problem making it more manageable.

Given a set of random variables  $X_V = \{X_1, X_2, \dots, X_n\}$  with  $n = |V|$ , the joint probability distribution  $P$  can be defined as:

$$P(X_V = x_V) = \prod_{v \in V} P(X_v = x_v | X_{\text{Pa}(v)} = x_{\text{Pa}(v)}) \quad (4.3)$$

In this formulation,  $\text{Pa}(v)$  represents the set of parent nodes of the node  $v \in V$  and  $P(X_v = x_v | X_{\text{Pa}(v)} = x_{\text{Pa}(v)})$  stands for a family of conditional probability distributions,

one distribution for each value  $X_{\text{Pa}(v)} = x_{\text{Pa}(v)}$ , which when lumped together is called a CPT (a table detailing the conditional probability distributions of the random variable  $X_v$  given the values of its parent variables  $X_{\text{Pa}(v)}$ ).

It follows that a node  $X_v$  where  $\text{Pa}(X_v) = \emptyset$ , the conditional probability for  $X_v$  given its parents -  $P(X_v | \text{Pa}(X_v))$  - is simply the node's marginal probability, denoted as  $P(X_v)$ , reflecting that its probability distribution is not conditioned on any other node.

Let's now consider a scenario where  $V = \{1, 2, 3\}$ ,  $v = 1$  and  $\text{Pa}(v) = \{2, 3\}$  are the corresponding variables  $X_1, X_2$  and  $X_3$  are binary (can assume one of two values). The CPT for node  $X_v$  given the parents  $X_{\text{Pa}(v)}$  would look like in Table 4.2. Note that according to the laws of probability theory we have that  $P(X_1 = \text{no} | X_2, X_3) = 1 - P(X_1 = \text{yes} | X_2, X_3)$  for all values of  $X_2, X_3$ .

TABLE 4.2: Example computation of a CPT

$X_2$	$X_3$	$P(X_1 = \text{yes}   X_2, X_3)$
no	no	0.4
no	yes	0.2
yes	no	0.9
yes	yes	0.7

Constructing or learning CPTs are an important step in the construction of a Bayesian Network. By making use of *posterior probability distributions*, the constructed probability distributions  $P(X_V)$  can be used for *probabilistic inference* that is the process of forecasting outcomes or states based on known information or evidences. The *posterior probability distribution* updates the probabilities after considering the *evidences*  $E$ . Mathematically, this can be represented as:

$$P(X_U | X_E = x_E) = \frac{P(X_U, X_E = x_E)}{P(X_E = x_E)} \quad (4.4)$$

where  $X_E = x_E$  is a set of *evidence variables with assigned values*, i.e. observations of a set of particular variables  $X_E = \{X_v | v \in E\}$ , and  $X_U$  represent the random variables for which we want to compute the probability distribution given the evidence where in most cases  $E \cap U = \emptyset$  (otherwise the probability will be 1, as  $P(X_v = x_v | X_v = x_v) = 1$  by definition).

Computing probabilistic inference through the posterior probability distribution although a crucial task for prediction, is a NP-hard problem. In real life situations, computing exact inference is not always possible due to the task being computationally intensive. For this reason, researchers have developed approximation algorithms, which have also shown to be NP-hard [16]. The network structure is an important factor in determining the computational speed of probabilistic inference. By exploiting conditional independence, as imposed by the network structure and therefore reflected in the joint probability distribution, probabilistic inference is usually feasible.

### Application of Bayesian Networks

Bayesian Networks can be used for a variety of tasks and are often preferred in situations - like the healthcare domain - where making decisions can have a high impact not just as a monetary aspect. Their importance is specifically relevant in this domain since graphical visualizations can be employed for understanding the reasoning behind the choices made by

the network. Additionally, the use of visualizations can be combined with domain experts knowledge that can be incorporated in the network by identifying how valid the network's choice is and help achieve the best result. In a Bayesian Networks (BNs), each node represents a random variable. In the context of the research, these variables can represent factors such as the price of a company, the sector it belongs to, or the price change of the S&P 500. While BNs are mainly designed to handle discrete variables, they can also be adapted to incorporate continuous random variables.

One common approach for handling continuous variables is the use of discretization techniques, where continuous random variables are divided into discrete buckets based on their values. For example, an increase in the stock's price as a percentage of the company's value, could be divided into ranges representing different levels of increase.

An additional method to incorporate incorporating continuous variables involves using Gaussian distributions, where the data is represented through the use of tuples consisting of mean and standard deviation.

Although continuous variables can be used, handling discrete data is less challenging and can be analysed in an easier way compared to continuous distributions. In both cases, the data distribution is typically represented in the form of Conditional Probability Tables (CPTs).

### 4.2.3 Dynamic Bayesian Networks

In real-world scenarios, we often have to deal with problems where uncertainty is omnipresent. How likely is it for a mail to be spam? How likely is it for the weather to be rainy tomorrow? Bayesian Networks (BNs) in general are well suited to cope with missing data and uncertainty and are well renowned for being able to incorporate prior or domain experts knowledge in the network where by looking at the nodes in a network as natural events or actions, we can model - most of the times - complex scenarios. Although Bayesian Networks (BNs) are powerful tools for representing relationships and dependencies among a set of variables, due to their nature, they lack the mechanism to represent temporal evolution. By using BNs, we can have a model with a good representation of spam emails as in [18], however, the prediction of events where time is a fundamental aspect, as in a weather forecasting task, gets more complex.

Dynamic Bayesian Networks (DBNs) - similarly to Bayesian Networks (BNs) - are a kind of probabilistic graphical model structured in the form of Directed Acyclic Graphs (DAGs) and can better cope with this issue by extending BNs to model the temporal aspect, capturing the probabilistic dependencies among variables across different time steps. This new approach has led to a growing interest in exploring those networks further, especially in the context of interpreting diverse types of data sources.

This enhancement allows time series data to be effectively used by capturing the temporal dynamics to define the development of the process over time. In a DBN nodes are duplicated across consecutive time slices, creating connections between nodes from one time slice and those in the consecutive one.

DBNs employ a Markovian approach to delineate the evolution of a system's state over time. The Markov principle posits that the condition of a system at any given time, denoted as  $t$ , is influenced solely by its state at the preceding moment,  $t - 1$ , meaning that each state transition doesn't depend on the entire history of the system but only on the previous state. This is an important concept in the field of DBNs since we can think of nodes and edges in terms of Markov chains.

As visible in figure 4.2 that represents an expanded DBN, the network isn't restricted to connections within the same time slices, called intra-slice connections. It instead extends

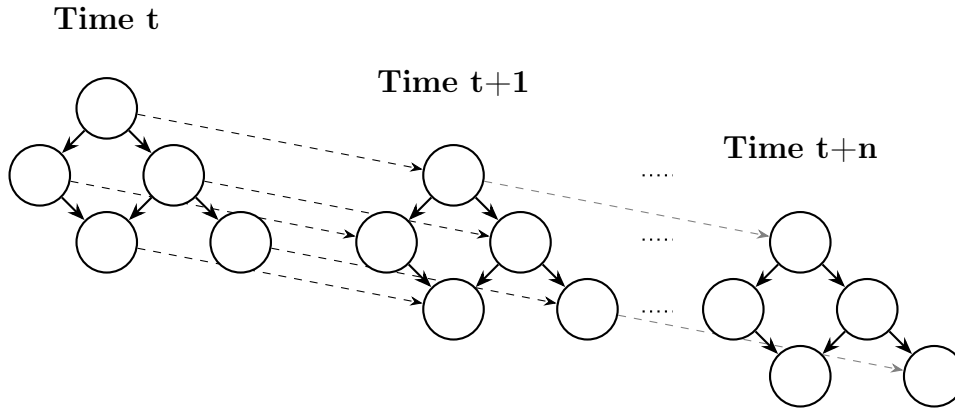


FIGURE 4.2: A Dynamic Bayesian Network represented over multiple time slices.

also to inter-slice connections, which bridge states across consecutive time frames. These connections are a fundamental aspect of DBNs, as they represent conditional dependencies between variables spanning between different time intervals. These transitions create a joint probability distribution over multiple time steps as represented in formula 4.5.

$$P(X^{0:T}) = P(X^0) \prod_{t=1}^T \prod_{v \in V} P(X_v^t | X_{Pa(v)}^{t-1}) \quad (4.5)$$

Here,  $P(X^{0:T})$  denotes the joint probability distribution of the set of random variables  $X$  from time  $t = 0$  to time  $t = T$ .  $P(X_0)$  is the initial probability distribution of the random variables at time  $t = 0$ , and  $P(X_v^t | X_{Pa(v)}^{t-1})$  represents the conditional probability distribution of each random variable  $X_v$  where  $v \in V$  at time  $t$ , given its parents  $X_{Pa(v)}$  at time  $t - 1$ . It follows that, to compute the joint probability for the variables in two consecutive time steps, the formulation is as follows:

$$P(X^t, X^{t+1}) = P(X^t) \prod_{v \in V} P(X_v^{t+1} | X_{Pa(v)}^t) \quad (4.6)$$

In a DBN each slice represents the state of the system at a specific time  $t$  consisting of nodes and edges, structured similarly to a Bayesian Network. Dependencies are represented through the use of edges that connect nodes between consecutive time slices, displaying the temporal evolution of variables.

Considering the example previously used where node  $X_v$  has two parents  $X_{Pa_1(v)}$  and  $X_{Pa_2(v)}$  and considering a DBN with two time slices, the smallest DBN structure can be represented as in figure 4.3.

In this example we can see how the variables at time  $t$  affect the variables at time  $t + 1$ . This graph allows the Dynamic Bayesian Network to show and deal with the evolution of the data over time. For instance, in the hypothesis of financial data, a structure of this kind might represent the evolution of the prices where each time frame represent a week of data. By modelling temporal dependencies, DBNs provide more accurate forecasts into the future behaviour of financial markets.

Understanding how things change over time is crucial in a DBN. To help us understand how conditional probabilities are employed in DBNs, we need to look at one tool that helps us to understand the transition dynamics that is called Conditional Probability Table

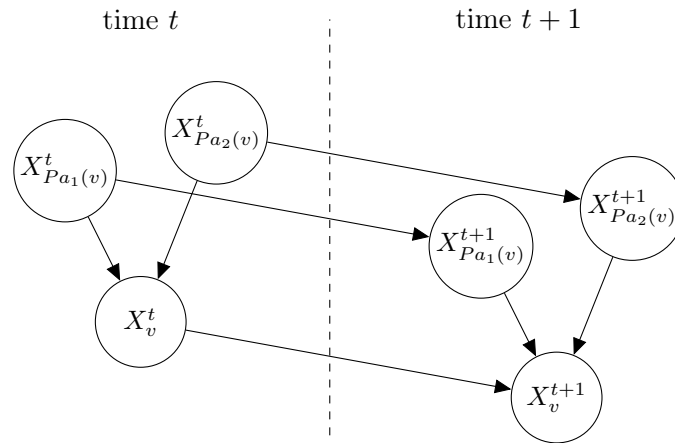


FIGURE 4.3: Dynamic Bayesian Network with two time slices

(CPT). As detailed in sec 4.2.2, think of a CPT as a detailed chart that contains specific information on how likely it is for the system to move from one state to another at each time step and between time steps in the case of DBNs. Even within a single moment in time, there are often numerous variables and factors at play, each interacting with the others in complex ways. CPTs aren't just useful for understanding changes over time; they are also handy for mapping out the relationships and interactions between different variables at any given moment (referred to as "intra-slice" Conditional Probability Distributions (CPDs)). In table 4.3 we can see a classic example where we want to predict the weather.

$W_t$	$S_t$	$P(W_{t+1} W_t, S_t)$		
		Sunny	Rainy	Cloudy
Sunny	Winter	0.4	0.4	0.2
Sunny	Summer	0.6	0.3	0.1
Rainy	Winter	0.2	0.5	0.3
Rainy	Summer	0.3	0.4	0.3
Cloudy	Winter	0.3	0.4	0.3
Cloudy	Summer	0.5	0.3	0.2

TABLE 4.3: Conditional Probability Table for  $W_{t+1}$ 

The right side of the table, represents the weather probabilities for the next day, while the first two columns represent the different combinations that we might encounter. Assuming that today (Time  $t$ ) is sunny and it is winter, we can see that the probability of tomorrow being sunny is equal to 0.4 which represents a 40% chance. In figure 4.4 we can see the graphical representation of a DBN that depicts this example.

#### 4.2.4 Learning a Bayesian Network

Training a Bayesian Network is a complex task that can involve a combination of two processes: structure learning and parameters learning. Figure 4.5 outlines some of the different options for learning the network grouped by category. Structure learning is the process of determining how nodes are interconnected through edges to build a DAG. The structure represents conditional dependencies and independencies between random variables and is usually represented as a state transition matrix, while the parameters represent the conditional probabilities in the form of Conditional Probability Tables (CPTs). Despite the



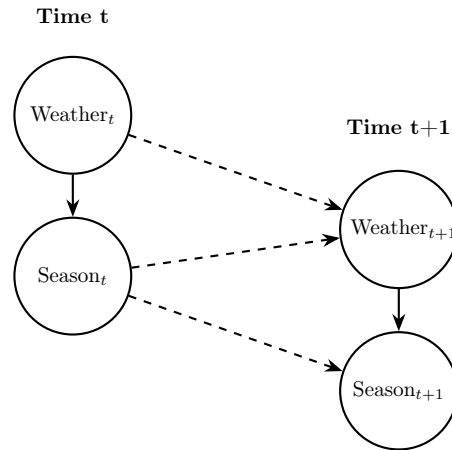


FIGURE 4.4: A Simplified Dynamic Bayesian Network illustrating dependencies between weather and season over two consecutive days.

two being tightly coupled, you do not necessarily need the exact parameters to learn the structure of the BN for its correct functioning, although this is dependent on the kind of approach that is followed. The motivation behind it is that by having enough observations, the patterns and dependencies between variables can still be detected despite not having the exact conditional probability distributions. Vice versa, the data distributions can be learned from the data. However, it is not possible to infer the parameters without knowing the structure. In fact, parameters are nothing less than the probabilities that define the relationships represented by the structure.

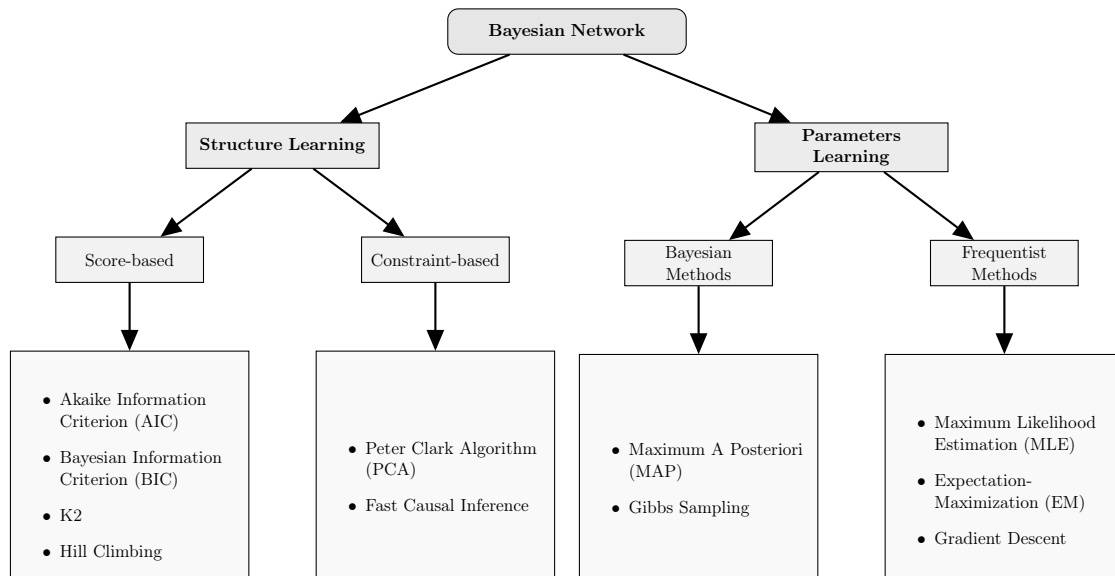


FIGURE 4.5: Example of possible algorithms for structure and parameters learning

### Structure Learning

Structure learning algorithms identify which variables are likely to be correlated and this information can be combined together with domain knowledge to build more accurate models. In general, to model a network structure (a DAG), we can identify two different



approaches called *constraint-based* and *score-based*. An additional approach is also possible and this involves a combination of the two mentioned approaches to create a hybrid approach.

The score-based is a model selection approach that comprises of two steps. First, these methods assign a numerical score to each model based on how well they fit a given dataset  $D$ . The goal is to find the structure that best fits the data, hence, we want to retrieve the model with the highest score. Once we have a scoring function to retrieve the optimal model, it is necessary to define a search strategy to go through all the possible combinations of models that apply to a certain dataset so as to be able to select the optimal model. The scoring functions can be classified into two different categories based on the approach they are based on, that is *Bayesian* and *information-theoretic* scoring functions. Among some of the most used scores, we can find the K2 score, also used in [48] and information-theory-based criteria such as Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC).

Both AIC and BIC help us understand how well a model fits the data. However, in a Bayesian Network (BN) as the amount of parameters increases, so do the nodes in the network. To obviate this problem the scoring functions add some regularization to reduce the complexity of the model and avoid overfitting on the training data. Despite its similarities, the two functions are based on different concepts. The AIC in eq. 4.7, as defined by Koller [24], is an estimate of the distance between the true likelihood function and the actual likelihood function of our model. In simple terms, it provides us with a greater understanding of the difference between the real data and what our model predicts.

$$\text{AIC}_{\text{score}}(\mathcal{G}|\mathcal{D}) = \ell(\hat{\theta}_g|\mathcal{D}) - \text{Dim}(\mathcal{G}) \quad (4.7)$$

On the other hand, the BIC in eq. 4.8, also defined by Koller [24], approximates the model's marginal likelihood with the aim of selecting the "correct" model from a large set of candidates.

$$\text{BIC}_{\text{score}}(\mathcal{G}|\mathcal{D}) = \ell(\hat{\theta}_g|\mathcal{D}) - \frac{\log |\mathcal{D}|}{2} \cdot \text{Dim}(\mathcal{G}) \quad (4.8)$$

The main goal of both scoring functions is to find a balance between the model's accuracy and complexity. As visible in the defined equations 4.7 and 4.8 both compute the log-likelihood of a graph  $\mathcal{G}$  given a dataset  $D$  knowing the parameters  $\hat{\theta}_g$ . A higher log-likelihood value also means a better fit. The second part of the functions, also called the penalty term, is what actually differentiates the two. This term promotes simpler models. When it's subtracted from the model's likelihood, a lower AIC or BIC value indicates a better model. The term  $\text{Dim}(G)$  represents the number of independent parameters in the network  $G$  while  $|\mathcal{D}|$  represents the number of elements of the dataset  $\mathcal{D}$ . As we have more and more data, the sample size  $|\mathcal{D}|$  increases and so does the penalty term, leading to a higher penalty for more complex (large) models. This means that for large datasets, AIC will tend to go in favour of larger networks compared to BIC. Given the features they provide, we can say that there is not a favourite function between the two and both are useful in different contexts. In cases where  $|\mathcal{D}|$  is small, AIC could better represent the data, whereas in cases where we have a larger dataset BIC might be favorable to reduce the network complexity and avoid overfitting.

Score-based approaches use the output value of the scoring functions as a discriminant for choosing the best model. In contrast, constraint-based approaches employ statistical tests to determine which nodes or variables are related, with the objective of identifying the connections that should exist between nodes based on observed data. A popular method among constraint-based algorithms is the Peter Clark (PC) algorithm, named after Peter Spirtes and Clark Glymour. As defined in [40], in this algorithm, we start with creating a fully connected undirected graph. Let  $V$  be the set of nodes (or vertices)  $V = \{v_1, v_2, \dots, v_n\}$  and let  $E$  be the set of undirected edges  $E = \{\{u, w\} \mid u, w \in V \text{ and } u \neq w\}$ . We define as fully connected - or complete - a graph  $\mathcal{G} = (V, E)$  where for every pair of nodes  $(u, w)$ , there exists one and only one edge  $e \in E$  such that  $e = \{u, w\}$ . Formally, this is described as:  $\forall u, w \in V, u \neq w \implies \{u, w\} \in E$

For each pair of nodes (or variables) we employ a statistical test to test the conditional independence. By identifying the conditional dependencies and independences, we are able to construct the so-called "skeleton" by removing the edges between independent nodes starting from the fully connected graph. When using the PC algorithm, multiple statistical tests can be employed to assess the (conditional) independence. As described by Parah [31], the choice of the statistical test is based on the needs. The data type is an important discriminating factor for its choice. For discrete data, the Chi-Squared and Fisher's Exact tests can be used, for continuous data, the Student's t-test is ideal. In both cases, with discrete or continuous data, the mutual information can be utilized. Once the process of computing the independence of the variables through statistical tests is complete, we continue testing the conditional independence by conditioning on additional other variables in the network. The process first starts by conditioning on one variable, then two, and so on until a specified depth is reached or an independence is found. As done in the step before, if a conditional independence is found, the edge between the two nodes is removed. After determining the skeleton, the algorithm orients the edges by applying some rules.

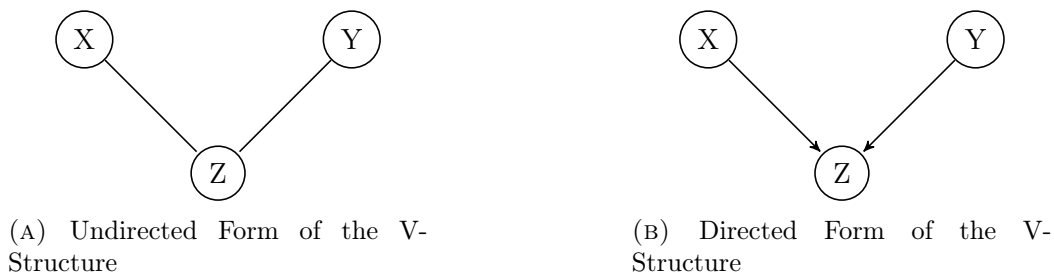


FIGURE 4.6: Comparison of a V-Structure and its Directed Form

In the PC algorithm, the second step is orienting V-structures that are defined as sets of three variables where one of the pairs is conditionally independent - no edge links the nodes. For every set of three variables  $X$ ,  $Y$  and  $Z$ , as in fig 4.6a. Assuming that there is no dependence between  $X$  and  $Y$ , we check if  $Z$  is part of the set that makes  $X$  and  $Y$  conditionally independent. If  $Z$  is not part of this set, then the edges are oriented as  $X \rightarrow Z$  and  $Y \rightarrow Z$  as depicted in picture 4.6b. Once this step is done for each triple of nodes, then an additional step is necessary to ensure that the graph follows the rules of a DAG. For easier interpretation, in table 4.4 we can see a representation of the steps that the algorithm takes. The first rule, which is also the one just described, is applied once; the other rules (2 to 4) are applied iteratively until all the edges are oriented.

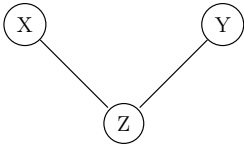
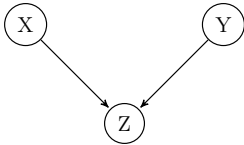
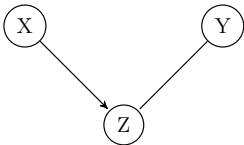
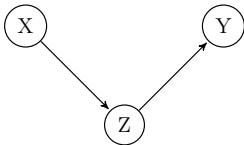
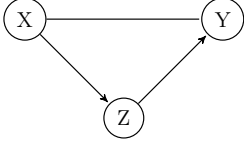
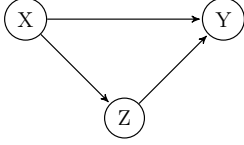
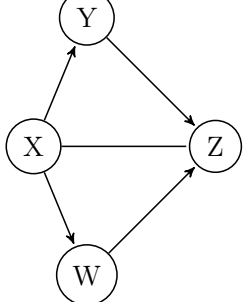
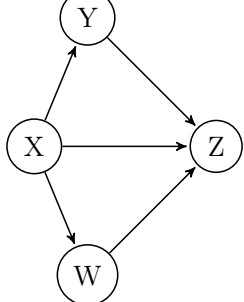
Initial Graph	Rule	Outcome
	$\forall(X, Y, Z)$ , such that $X$ and $Z$ , and $Y$ and $Z$ are adjacent, but $X$ and $Y$ are not, reorient $X - Z - Y$ as $X \rightarrow Z \leftarrow Y$ if $Z \notin \text{sepset}(X, Y)$ , i.e. if the set of variables that, when conditioned upon, renders $Y$ and $Z$ conditionally independent.	
	$\forall(X, Y, Z)$ , orient $Z - Y$ as $Z \rightarrow Y$ if $\exists$ a directed edge $X \rightarrow Z$ such that $X$ and $Y$ are not adjacent.	
	$\forall(X, Y, Z)$ , orient $X - Y$ as $X \rightarrow Y$ if $\exists(X \rightarrow Z \rightarrow Y)$ .	
	$\forall(X, Y, Z, W)$ , orient $X - Z$ as $X \rightarrow Z$ if $\exists(X \rightarrow Y \rightarrow Z$ and $X \rightarrow W \rightarrow Z)$ such that $Y$ and $W$ are not adjacent.	

TABLE 4.4: Steps of the PC Algorithm

### Parameters Learning

Once the structure is determined, the next step to construct a Bayesian Network is to estimate the Conditional Probability Distributions (CPDs) of the network that are contained in CPTs and represent the causal relationship between a child and its parents' nodes. Parameters learning is nonetheless than the task of learning the CPT. The estimation is a step needed in most of the cases to obtain the probability distributions. Due to the complexity of the model and due to the amount of data, it is often not possible to obtain a fully observed model, meaning a model where all the variables in the network are observed or known for each possible case. Essentially, this means that there are no missing values or hidden variables in the data. When all variables are observed, the conditional probability distributions can be directly estimated from the data by counting the occurrences in the dataset, hence, no parameter estimation algorithm is needed. In fact, counting occurrences acts as a *maximum likelihood estimator* for multinomial distributions. This is derived from the computation of the first derivative of the probability mass function with respect to the probability parameters, eliminating the need for parameter estimation algorithms in this scenario.

This can be demonstrated by computing the maximum likelihood estimation as follows. Consider a multinomial distribution with parameters  $P_V = (p_1, p_2, \dots, p_K)$  where  $p_i$  is the probability of the  $i$ -th category,  $K = |V|$ , where  $V$  is the number of nodes (or vertices) in the network and  $\sum_{i=1}^K p_i = 1$ . Given a dataset with counts  $N_V = (n_1, n_2, \dots, n_K)$ , where  $n_i$  is the number of occurrences of the  $i$ -th category, the probability mass function (PMF) is:

$$\begin{aligned} p(N_V = n_V \mid P_V = p_V, N) &= \frac{N!}{n_1! n_2! \dots n_K!} \cdot p_1^{n_1} p_2^{n_2} \dots p_K^{n_K} \\ &= N! \cdot \prod_{i=1}^K \frac{p_i^{n_i}}{n_i!} \end{aligned} \quad (4.9)$$

where  $N = \sum_{i=1}^K n_i$  is the total number of observations. Given the obtained PMF, the log-likelihood becomes as in formula 4.10.

$$\begin{aligned} \log(p) &= \log\left(N! \cdot \prod_{i=1}^K \frac{p_i^{n_i}}{n_i!}\right) \\ &= \log N! - \sum_{i=1}^K \log n_i! + \sum_{i=1}^K n_i \log p_i \end{aligned} \quad (4.10)$$

To be able to differentiate the log-likelihood to maximize the function, we need to implement the previously defined constraint  $\sum_{i=1}^K p_i = 1$ . A fundamental property of probability distribution is that the sum of the probabilities must be equal to 1, as a consequence, this constraint represents the probabilities of the different variables in the multinomial distribution. When all the variables are observed, this constraint must always hold. The implementation can be done using the Lagrangian where  $\lambda$  represents the Lagrangian multiplier, a term used to penalize deviations from the defined constraint, as in formula 4.11.

$$\begin{aligned}
\mathcal{L}(p, \lambda) &= \log(p) + \lambda \left( 1 - \sum_{i=1}^K p_i \right) \\
&= \log N! - \sum_{i=1}^K \log n_i! + \sum_{i=1}^K n_i \log p_i
\end{aligned} \tag{4.11}$$

We now maximize the log-likelihood by differentiating the Lagrangian with respect to  $p_i$  and then setting the result to 0 to obtain  $p_i$ :

$$\begin{aligned}
\frac{\partial}{\partial p_i} \mathcal{L}(p, \lambda) &= \frac{\partial}{\partial p_i} \log(p) + \frac{\partial}{\partial p_i} \lambda \left( 1 - \sum_{i=1}^K p_i \right) \\
&= \frac{\partial}{\partial p_i} \log(p) - \lambda \\
&= \frac{\partial}{\partial p_i} \left( \log N! - \sum_{i=1}^K \log n_i! + \sum_{i=1}^K n_i \log p_i \right) - \lambda \\
&= \frac{n_i}{p_i} - \lambda \implies p_i = \frac{n_i}{\lambda}
\end{aligned} \tag{4.12}$$

Using the initial constraint  $\sum_{i=1}^K p_i = 1$ , it follows that:

$$\sum_{i=1}^K \frac{n_i}{\lambda} = 1 \implies \lambda = \sum_{i=1}^K n_i \implies \lambda = N \tag{4.13}$$

Coming back to the previous equation, we find that:

$$p_i = \frac{n_i}{N}$$

The derivation confirms that the parameter estimation using Maximum Likelihood Estimation (MLE), in cases where all the variables are known, is the same as counting the occurrences of the variables.

As we will see later in the project, this is an ideal scenario that simplifies both the learning and the inference processes. However, in the real world, we often have to deal with variables that are not always known, making it necessary to use algorithms for learning the parameters through estimations. In figure 4.5, we can see how the parameters learning can be categorized in two main approaches, Bayesian and Frequentist methods. The former aims to calculate the probability of the parameters  $\theta$  given the data  $D$ , it provides a full tree of possibilities for the parameters  $\theta$  where the parameters are weighted by their probabilities. The latter, instead, aims at finding the point estimation of the parameters  $\theta$ , which is the single most likely set of parameters.

In the frequentist statistics, the MLE is employed to obtain a point estimate of the model's parameters, symbolized by  $\theta$ . To begin, the log-likelihood function is determined. This function represents the natural logarithm of the probability of the observed data, given a specific parameter set, and is chosen over the likelihood function due to computational advantages. Given a dataset  $D = \{d_1, d_2, \dots, d_n\}$  and a Bayesian network with parameters  $\theta$ , the log-likelihood function  $\mathcal{L}(\theta)$  is defined as in 4.14.

$$\begin{aligned}
\mathcal{L}(\theta) &= \log P(D|\theta) \\
&= \log \prod_{i=1}^n P(d_i|\theta) \\
&= \sum_{i=1}^n \log P(d_i|\theta) \\
&= \sum_{i=1}^n \sum_j \log P(X_j = x_{ij} \mid X_{Pa(j)} = x_{Pa(j)i}, \theta)
\end{aligned} \tag{4.14}$$

Where  $X_j$  is the  $j$ -th variable (or node) in the network and  $X_{Pa(j)}$  represents the parents of the node  $X_j$ . The log-likelihood is preferred to the likelihood function itself because by transforming the products of probabilities into sums it simplifies the derivatives and we avoid having to compute multiplications that are computationally more expensive. The goal of the MLE method is to maximize the log-likelihood function, to obtain the best estimate of the parameters  $\theta_{MLE}$  as in eq 4.15.

$$\theta_{MLE} = \arg \max_{\theta} \mathcal{L}(\theta) \tag{4.15}$$

To maximize  $\mathcal{L}(\theta)$ , we compute the first partial derivative with respect to each parameter  $\theta_k$ , where  $k \in \theta$  and  $\theta_k$  is the  $k$ -th element of the set  $\theta$ , and set it to 0:

$$\begin{aligned}
\frac{\partial \mathcal{L}(\theta)}{\partial \theta_k} &= 0 \quad \forall k \\
\frac{\partial}{\partial \theta_k} \left( \sum_{i=1}^n \log P(X_j = x_{ij} \mid X_{Pa(j)} = x_{Pa(j)i}, \theta) \right) &= 0 \quad \forall k
\end{aligned} \tag{4.16}$$

As an example for the family of the frequentist approaches, let's now introduce a well-known algorithm called EM that comprises of two main phases that are executed iteratively. The first step - also called the E-step where E stands for Expectation - given the observed data and the current estimates of the parameters, computes the expected value of the log-likelihood function of the complete data (that includes observed and latent data). In the context of Bayesian Networks, the EM algorithm is a prime example of how to deal with partial data. When incomplete data is used as part of the learning process, some variables will not have observed values in the network. To solve the issue, the algorithm infers the most likely states of these hidden variables, given the observed data and the network's structure. Following, once the E-step is computed, the M-step - Maximization step - is used to find the parameter values such that the log-likelihood function obtained from the E-step is maximized. The output of this process is then used as an input to the E-step for the following iteration. This algorithm requires a threshold to be set and the process stops when the increase in likelihood falls below a certain threshold, the estimated parameters at the end of this process will then become the model's parameters.

Bayesian approaches differ in the process from the frequentists. While in the frequentist approach parameters are considered to be fixed quantities and the focus is more on the

likelihood of the data, in Bayesian inference, both data and parameters are treated as random variables. In addition, those methods allow for prior knowledge to be incorporated into the network through the use of prior distributions, which is a fundamental aspect of Bayesian Networks (BNs) since it allows for experts knowledge to be incorporated in the network itself. This prior distribution represents our beliefs about the parameters before observing any data, incorporating prior beliefs in a network, means manually defining the prior probabilities given our knowledge of the distribution. Despite this being an advantage of Bayesian inference, appearances can sometimes be deceiving. The reason is that it is highly influenced by the subjectivity of the definition of the prior distribution since local maxima are obtained in the learning process. Hence, starting from a different prior distribution also leads to reaching a different posterior distribution, which in substance means reaching different conclusions.

Let's now introduce the Maximum A Posteriori (MAP) to understand this process. At the foundation of the Bayesian approach, we find the Bayes theorem, defined as in 4.17. In this formula  $P(\theta|D)$  represents the posterior probability,  $P(D|\theta)$  the likelihood of observing the dataset  $D$  given the parameters  $\theta$  and  $P(\theta)$  the prior probability of the parameters  $\theta$

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \quad (4.17)$$

The goal of the MAP approach is to find the parameters  $\theta$  that maximize the posterior probability, resulting in what is called point estimation. In this project, since the dataset  $D$  is fixed, the denominator  $P(D)$ , which represents the marginal likelihood, is a constant. However, because it does not depend on  $\theta$ , it does not affect the optimization of the parameters. Therefore,  $P(D)$  can be omitted from the optimization process, simplifying the formula to maximize the product of the likelihood  $P(D|\theta)$  and the prior  $P(\theta)$  as in 4.18. This simplification reduces the complexity and speeds up the computation.

$$\theta_{MAP} = \arg \max_{\theta} (P(D|\theta) \cdot P(\theta)) \quad (4.18)$$

In the context of a full Bayesian learning approach, the denominator  $P(D)$  would normally be defined as:

$$P(D) = \int P(D|\theta) \cdot P(\theta), d\theta \quad (4.19)$$

Integrating the product of the likelihood and the prior over all possible parameters can be challenging, especially with a large number of parameters.

### 4.2.5 Partitioned Dynamic Bayesian Networks

Partitioned Dynamic Bayesian Networks (PDBNs) offer an advanced method for non-homogeneous modeling processes, which are processes that vary over time, unlike traditional Bayesian Networks (BNs), which treat the entire period uniformly. While the approach of BNs simplifies the modeling process, it fails to capture the intricacies of temporal data, leading to models that do not accurately reflect the underlying process dynamics. This is particularly problematic in fields like the medical one, where data sparsity and dynamic changes are common. PDBNs leverage limited data more efficiently than BNs and their extension DBNs.

DBNs improve temporal modeling by duplicating random variables (nodes) over time, with each time frame having an instance of the variables, and edges between different time frames used to represent conditional (in)dependencies between nodes as time progresses. This peculiarity makes DBNs better suited to capture the temporal domain when compared to more classic models. However, this comes with its own limitations. The limitation of these kinds of networks resides in their inability to model changes in the structure of the networks, the idea behind it is to model a dynamic system where probabilistic dependencies between variables evolve over time, but the structure of the graphical model remains unchanged.

Partitioned Dynamic Bayesian Networks (PDBNs), on the other hand, address this limitation by following a different approach. In the original paper by Marcos Bueno [11], a structure learning approach was proposed where a homogeneous model first captures the overall temporal pattern, and sub-models are subsequently incorporated for discrete intervals. This technique permits an increase in model complexity only when a proposed partitioning of a sub-model yields better predictive accuracy across both training and testing datasets. It's important to notice that this partitioning is applied uniformly across all features at the point in time where the new partition was created, ensuring that there is a common division at each "junction".

In essence, PDBNs dissect the timeline into chunks or periods, each of which is modeled distinctly. For each time slice, a separate DBN model is created, which fits that period within the larger sequence. This is represented in figure 4.7b where it is possible to notice that each submodel represents a Dynamic Bayesian Network (DBN). The original paper on PDBN introduces the term "distribution cuts" to describe the points in time in which the process is segmented. A network with "k" distribution cuts, denoted as PDBN- $k$  divides the timeline into  $k + 1$  separate parts, where each part is modeled independently. It follows that a PDBN with a single cut - PDBN-1 - is equivalent to having a cut at T where T represents the full extent of the timeline, as represented in figure 4.7a. A PDBN with a single cut, is effectively the same as a standard Dynamic Bayesian Network (DBN); no "partitioning" is created since only one model is present over the entire timeline. In substance, a PDBN-1 covers all the timeframes of the DBN whose range is  $\{0, \dots T\}$ .



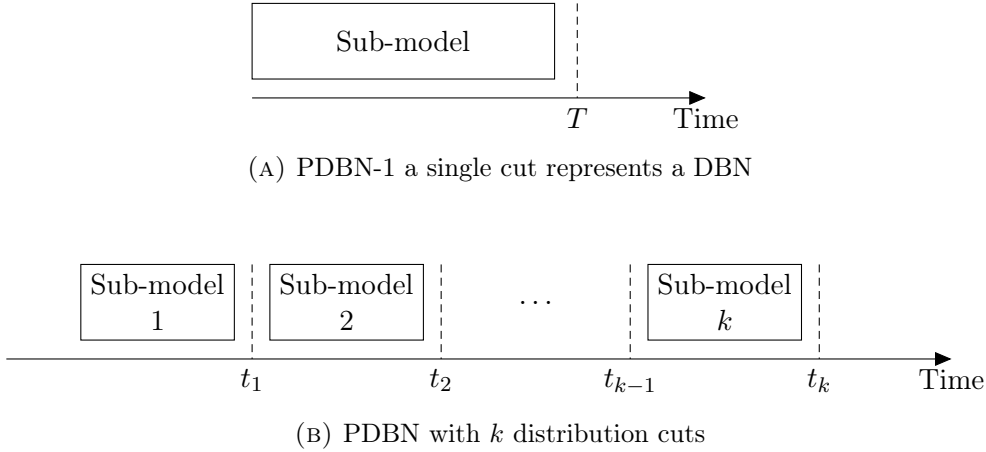


FIGURE 4.7: Comparative illustration of PDBNs with different numbers of distribution cuts.

From the given description of the network, we can derive the **joint probability distribution** which plays a crucial role in capturing the evolving dynamics. The joint probability distribution empowers the calculation of the probability of future events given current observations, allowing inference on past states or future predictions. The joint probability distribution formulation as defined by Bueno [11] is expressed in Equation 4.20, where the equation represents the joint probability over all variables  $\mathbf{X}^{(0:T)}$  from time  $t = 0$  to  $t = T$ .

$$P(\mathbf{X}^{(0:T)}) = \prod_{i=1}^n P_0 \left( X_i^{(0)} \mid \pi(X_i, \mathcal{B}_0) \right) \prod_{r=1}^k \prod_{t=t_{r-1}}^{t_r-1} \prod_{i=1}^n P_r \left( X_i^{(t+1)} \mid \pi(X_i, \mathcal{B}_r) \right) \quad (4.20)$$

As visible in the formulation, there is a clear distinction between the components of the function. It can be considered as split into two main parts, where the first part, displayed in Equation 4.21 represents the initial probability distribution at time  $t = 0$ , while the second part accounts for the transitions between subsequent time steps.

$$\prod_{i=1}^n P_0 \left( X_i^{(0)} \mid \pi(X_i, \mathcal{B}_0) \right) \quad (4.21)$$

This term describes the probability of each variable  $X_i$  at the initial time step  $t = 0$ , conditioned on its parents  $\pi(X_i, \mathcal{B}_0)$  where  $\mathcal{B}_0$  represents the structure of the initial Dynamic Bayesian Network (DBN).

The second term in the formula represents the transitions between different cutsets. For each cutset  $r$ , representing a different DBN, the joint distribution over the networks is computed as in Equation 4.22.

$$\prod_{r=1}^k \prod_{t=t_{r-1}}^{t_r-1} \prod_{i=1}^n P_r \left( X_i^{(t+1)} \mid \pi(X_i, \mathcal{B}_r) \right) \quad (4.22)$$

This term iterates over the different time slices  $t_1, t_2, \dots, t_k$ , available in each cutset, the product computes the probability of each variable  $X_i$  at time  $t + 1$ , given its parents  $\pi(X_i, \mathcal{B}_r)$  and the Dynamic Bayesian Network (DBN) structure  $\mathcal{B}_r$ .

This part of the equation indicates that the state of a variable at the next time step ( $t + 1$ ) depends on the current state of its parents. To provide an example, it follows that the price of the S&P 500 depends on the previous day price of the companies composing the index.

When dealing with sparse data, Partitioned Dynamic Bayesian Networks (PDBN) show their full strength sustaining a high accuracy and generally low possibility of overfitting. A PDBN can construct a model that adapts its complexity progressively, which is a good way to mitigate the risk of overfitting. Learning the network structure happens through an algorithm designed to partition a dataset sequentially in time. Initially, the algorithm considers the entire timeline as a single segment (as previously defined, this corresponds to a PDBN-1). It then introduces distribution cuts to create partitions, thereby segmenting the timeline into discrete, smaller models that describe different periods. Each of these cuts are methodologically placed based on a scoring function which has a primary role in the construction of the network. The goal of the scoring function is to reward simpler models while maximizing accuracy. This process continues until no further cuts improve the model, indicating that the current complexity is adequate.

When moving from one segment to the next, the final state of the variables at the end of one segment serves as the initial state for the next. This process is also called conditioning. The CPDs of the initial state at time  $t + 1$  are conditioned on the final state of the previous segment at time  $t$ . Through this process the model captures temporal dependencies across different segments. To use an analogy, we can think of it like a relay race. In this case, each runner represents a cut and has to pass the baton to the next runner where the baton represents the state of the system between the two runners that represent two different cuts. Using this analogy, it is easy to understand that the way the first runner finishes influences how the second runner starts his part of the race.

In substance, when moving to a new segment, you don't start from scratch, instead, you start based on where the last segment left off. Hence, the initial state of the next segment's CPDs depends on the final state of the previous segment's CPDs.

# Chapter 5

## Methodology

### 5.1 Data Extraction and preparation

In this research, the focus was placed on the predictive analysis of the Standard and Poor's (S&P) 500 index whose data is highly available and easy to access. Given the strategic importance of this index, it is an ideal candidate for the application of predictive models. Comprising 500 of the largest companies listed on the United States stock exchanges, the S&P 500 is a good indicator of the health and trends in the American economy. By delving into the stock prices of the companies that constitute the index, the research aims to capture the intricate dynamics and relationships that drive the overall performance of the index, to assess how the amount of historical data affects the overall prediction. The data used for this analysis can be categorized into four distinct groups: the composition of the companies included in the index, the historical prices of the S&P 500, the sectors to which each of these companies belong, and the daily price change of the companies composing the S&P 500 index itself. Let's now discuss how each was obtained.

#### 5.1.1 Retrieval of the S&P 500 constituents

Ticker symbols are a unique combination of letters and, sometimes, numbers allowing for quick identification of publicly traded companies in financial markets. For instance, the company Apple Inc. is identified with the symbol "APPL". These ticker symbols assume high importance when retrieving financial data since they are used as a primary key to identify each company in the financial markets. To understand how the S&P 500 index composition works, let's now introduce some concepts. As previously stated, the S&P 500 index consists of the 500 most valuable companies listed in the American stock exchanges. The value of these companies fluctuates over time, influenced by market conditions and individual company performance, and the fluctuations are reflected in the companies' stock prices, which in turn affect their market capitalization.

Market capitalization, often referred to as "market cap", is a measure of a company's total value in general related to the stock market but applicable to every ambiance. It can be calculated using the formula as in eq 5.1.

$$\text{Market Capitalization} = \text{Number of Outstanding Shares} \times \text{Share Price} \quad (5.1)$$

The term "Number of Outstanding Shares" refers to the total number of shares of a company that are currently held by all of its shareholders - including institutional investors. On the other hand, "Share Price" denotes the current trading price of a single share of

the company's stock. This is the price at which a share can be bought or sold in the open market at any given time. This calculation helps the investors determine the companies' size. In the context of the S&P 500, the index is mainly weighted by market capitalization, meaning that companies with larger market capitalization have a greater impact on the index's performance. Such a weighting mechanism ensures that the index reflects the market dynamics giving more importance where it's due. Several factors are taken into account to assess whether a company is eligible to be part of this index [39] where market capitalization is the primary factor.

This is an important concept due to the index being periodically reviewed to ensure it remains representative of the market's current state. Companies whose market capitalization grows significantly may be added to the index, replacing those with lower market values. The consequence of this process is an always-evolving and changing index where the constituents retrieved on a specific day will not be the same retrieved the following year, enforcing the need to retrieve all the constituents in different periods. This is further proven by the number of companies retrieved which is a total of 914 different ticker symbols although the index is only composed of circa 500 companies at each period in time. Retrieving the ticker symbols was a tedious process that involved obtaining the composition of the S&P 500, whose data was sourced from the iShares website [1]. iShares is a company active in the financial sector that allows the trading of Exchange-Traded Funds (ETFs), which are funds that try to replicate the performances of various indices. For a private investor, buying ETFs is a big advantage since acquiring a portion of an index is the equivalent of acquiring a portion of all the companies that compose the index without the need to buy each individual company the index is composed of. The iShares website offers technical and financial information about all the funds managed by them. Together with the financial data, they also provide the daily composition of the indexes, which, however, is not already available in the form of a dataset, but is accessible on their platform through the form of a table.

To retrieve this data from their website [1], a Python script was put in place to automate the extraction process. Selenium [4] is a Python library that enables the user to easily interact with web pages. In this case, the library was employed to execute a web scraping process designed to iterate across different dates, capturing all the available information on the index's composition. It's important to notice that the retrieved data not only includes the list of constituent companies but also details like the market value and the number of shares of each company. This step allowed the retrieval of data regarding the ticker symbols of all the companies that compose the index per day that the company was available in the index itself.

### 5.1.2 Retrieval of historical data

Once the composition of the S&P 500 index was determined for various historical periods, the next phase of the research involved the collection of historical price data for each constituent company. Given the full list of companies, APIs were used to access historical data. This was done by identifying the first and last dates of a company's presence within the index, thereby defining a specific period for which historical price data was needed. The APIs then provided daily stock prices for these companies, covering the entire span of their inclusion in the index. This type of data retrieval approach ensured that the analysis would include the precise time frames relevant to each company's impact on the S&P 500.

The data collection process utilized two distinct APIs: the "Nasdaq Data Link" provided by the National Association of Securities Dealers Automated Quotations (NASDAQ) stock exchange [29], and the Yahoo Finance API [46]. The NASDAQ is an American stock

exchange ranked second only behind the NYSE, in terms of market capitalization, calculated as the sum of the market capitalization of the individual companies that are listed in the exchange itself. The exchange can be viewed as a market that facilitates the trading of securities where securities are tradable financial assets that have an intangible value, like stocks and bonds among the most famous ones. Any type of entity that allows the exchange of companies' stocks, like banks and trading platforms, needs an intermediary. It is here that stock exchanges come into play, setting the rules and boundaries to be followed for a company's stock to be traded. The NASDAQ stock exchange, in addition to offering a platform for the exchange of stocks, also serves as an important hub for financial data where datasets of various types are available. In this platform, a large amount of financial data is available, including direct trading activities and companies' stock prices, through a digital marketplace where both the NASDAQ and third-party vendors can sell their market-related data services. The NASDAQ's extensive repository of datasets and APIs, although mostly accessible upon payment, contains many resources for real-time data analysis and offers some datasets for free for research purposes. Accessibly via an academic account, the Nasdaq Data Link grants access to stock price data. Similarly, Yahoo Finance API not only offers historical price data but also provides access to additional information for publicly traded companies. While the dataset initially comprised of 914 companies identified as constituents of the index, successful data retrieval was achieved for only 701 companies using the Nasdaq API. Several factors contributed to this discrepancy in terms of the amount of data retrieved through their API. Firstly, some companies might have been delisted from the exchange due to financial difficulties or bankruptcy. Secondly, mergers and acquisitions could have resulted in companies becoming private and not listed on the stock market anymore. Lastly, a company might change its stock ticker symbol that can happen for the above-mentioned factors (delisting, mergers, and acquisitions) as well as being a common occurrence when a company decides to change its name, which usually triggers a change in the ticker symbol to maintain user recognition and association. To address the shortfall in data collection, a second approach using the Yahoo Finance API was employed. By focusing solely on those tickers for which the Nasdaq API retrieved no data, 71 additional companies were identified increasing the total of companies for which the data was retrieved to 772 out of 914.

After employing the mentioned APIs, the data retrieved was in the form of Open, High, Low, Close, Volume (OHLCV). Starting from this dataset, a decision had to be made on which specific metric to use. After analyzing the data, we noticed that the APIs were already providing adjusted data, that is data where stock splits and dividends are already taken into account. In financial markets, there are times when companies decide that increasing or decreasing the number of shares available can profit the company. This so-called "stock split" does not change the overall market capitalization of a company, however, it changes the intrinsic value of each share. Moreover, companies can decide to release dividends, that are profits that are released to investor as a return for their investments. It is in this situations that adjusted data comes to our aid providing prices that are independent of how many or what kind of splits happened throughout the years and independently from the dividends released, providing a clearer vision of how companies' prices changed over the years. The final decision in terms of choice of source of truth for the daily market price of companies, went for the adjusted close price.

Once the adjusted close price was retrieved for each company, this was transformed into a percentage representing the daily price increase when compared to the previous day and discretized to be better handled by the network. Considering that each company can have a different behavior, which is also dependent on the liquidity of the companies's shares -

that is the number of shares available for each company - the discretization was computed per company. Figure 5.1 displays the ranges used for discretizing the data.

Range	Label
Below 10th percentile	High Decrease
10th to 30th percentile	Decrease
30th to 45th percentile	Low Decrease
45th to 55th percentile	No Change
55th to 70th percentile	Low Increase
70th to 90th percentile	Increase
Above 90th percentile	High Increase

TABLE 5.1: Percentile Ranges and Corresponding Labels

Given a company’s daily price change as a percentage of the company’s total value, percentiles were computed and the prices discretized for all the companies, where the price of the S&P 500 index, followed the same approach.

### 5.1.3 Retrieval of sectors data

To further enhance the predictive capabilities of the network, sectors data were included in the dataset. The Global Industry Classification Standard (GICS) [41] a widely recognized framework within the financial sector, categorizes companies into specific sectors, providing investors with a clearer understanding of the companies’ primary focus. In addition to classifying by each sector, the standard further divides companies into additional subgroups, offering a more in-depth view of the different types of business. For this research, companies were assigned to their respective primary sectors. After retrieving a subset of the data spanning a 10 years period comprising of a total of 596 companies, the Yahoo Finance API was initially employed to retrieve each company’s sector. However, as mentioned in the previous section, this approach encountered limitations, which led to the absence of sector information for some companies. Consequently, sector information for only 484 companies was successfully retrieved. For the remaining 112 companies, sectors were assigned manually by analyzing each ticker’s historical data to determine the corresponding company and subsequently classifying it into the correct sector. Financial websites, that provide financial news or historical stock price data like Marketwatch [2], Yahoo Finance [5], Nasdaq [3], and CNBC [6] were used as a primary source for this research.

Overall, the companies were divided into the eleven sectors defined by GICS. Table 5.2 provides an overview of the sectors and the number of companies identified within each category.

## 5.2 Application of Partitioned Dynamic Bayesian Networks

Let’s now introduce how the Partitioned Dynamic Bayesian Networks (PDBN) network [15] was constructed in this project. As defined in Chapter 4 Section 2.5, the development of the network involved the construction of multiple Dynamic Bayesian Networks (DBNs),

Sector	Number of companies	Percentage (%) of the total
Financial Services	87	14.59 %
Industrials	80	13.42 %
Technology	75	12.57 %
Healthcare	73	12.24 %
Consumer Cyclical	72	12.08 %
Energy	45	7.54 %
Consumer Defensive	41	6.87 %
Basic Materials	34	5.70 %
Real Estate	33	5.53 %
Utilities	29	4.87 %
Communication Services	27	4.53 %

TABLE 5.2: Number of companies by sector and their percentage share

as many as the defined number of cutsets. This development process happened through the use of the library `bnlearn` [37] and the programming language R.

The phases detailed below, highlight, step-by-step, the process followed for the construction of the model. Starting with the foundational steps that involved data preparation and cleaning, the process advanced through the manual and automated construction of each Dynamic Bayesian Networks (DBNs). Subsequently, starting from the newly defined DBNs, the structures were unrolled and the CPTs constructed for each network's node. The final phase involved learning the parameters given the provided dataset.

The development of the network can be logically divided into different phases as follows:

1. **Data preparation:** Initial organization and cleaning of data.
2. **Manual construction of Dynamic Bayesian Networks**
3. **Structure Learning:** determining the structure of each Dynamic Bayesian Network from the data
4. **Partitioned Dynamic Bayesian Network construction - Structure Unrolling:** Assembling the PDBN by extending the structure of individual DBNs
5. **Partitioned Dynamic Bayesian Network construction - Parameters Unrolling:** Construct the Conditional Probability Tables (CPTs) for the unrolled PDBN
6. **Parameter Learning:** Learn the network parameters from the data

Each of these phases is further detailed in the following sections to provide a comprehensive understanding of the network development process.

### Data Preparation

After downloading the S&P components and their corresponding sectors as defined in chapter 5.1.1, the initial step involved a preparation process to structure the data. Despite this careful preparation phase, it quickly became evident that an additional step was indispensable to align the data with the requirements of our network. The network's structure can significantly vary based on the change in the data and based on the specific cutset chosen for the analysis. To tackle this challenge, we began by loading the data

for each company, which includes the daily price changes. This data is then merged with the sectors associated with each company, employing a row-wise combination approach. Through this process, we create a structure where, for each day and each company, a single row represents the daily price change of that particular company. The structure resulting from this process is illustrated in table 5.3.

<b>Ticker</b>	<b>Sector</b>	<b>Change</b>
APPL	Technology	Increase
AMZN	Technology	Decrease
ABT	Healthcare	Low Increase
AMP	Financial Services	No Change

TABLE 5.3: Example of the processed data structure before the subdivision in time frames

The process just described is necessary for the creation of differently large networks. Specifically, when defining the transitions within each DBN we need to take an extra step to ensure they accurately reflect the transitions in stock price movements. This involves collecting data in a quantity proportional to the number of divisions or *cutsets* in the network. For example, if we are working with a network divided into five different DBNs, like a PDBN with five cuts (PDBN-5), we would use 20% of the dataset for training the structure of each segment (or cut). This collected data is again divided into two parts, where each of the two parts identifies one side of the transition of the DBN. This is crucial to ensure that each node in the network accurately models the transition between different phases of prices transitioning from one day to the following.

After gathering the data, the combined data (as in table 5.3) undergoes a structuring phase. This is where the data is prepared for further analysis, to facilitate the unrolling and learning of the network's parameters. The data is organized so that changes in company prices and the S&P 500 index, are split into columns. The data within these columns is distributed in proportion to the network's segments, as described above. For instance, if a PDBN-5 has a total of 50 transitions and we focus on the first 10 transitions, we will make use of 20% of the available data for this purpose. This extracted subset is then utilized to learn the parameters of the first five transitions, an example structure can be seen in table 5.4.

<b>Ticker</b>	<b>Sector</b>	<b>Change_1</b>	<b>Change_2</b>	<b>SNP_Price_1</b>	<b>SNP_Price_2</b>
APPL	Technology	Increase	Low Decrease	High Increase	Low Increase
ABT	Healthcare	Low Increase	Increase	Low Increase	High Decrease
AMP	Financial Services	No Change	Low Decrease	Increase	No Change
APPL	Technology	Decrease	Low Increase	Low Decrease	Increase
ABT	Healthcare	Increase	Low Increase	Low Increase	Low Decrease
AMP	Financial Services	High Increase	Decrease	Decrease	Low Decrease

TABLE 5.4: Example of the processed data structure divided in time frames



### Manual Construction of Dynamic Bayesian Networks

Constructing the structure of a BN is a challenging task, with a lot of research that has been done over the years, mostly involving the use of structure learning, which, as explained in chapter 4.2.4, is a significant component of building a Bayesian Network. Unlike manual structure construction, which relies heavily on expert knowledge and domain understanding, structure learning enables the creation of networks directly from data, avoiding the need for human intervention in specifying the network topology.

Although from an external perspective structure learning seems to have many advantages and seems to always play a crucial role, this does not come without drawbacks. The complexity of the structure learning problem cannot be, in fact, understated. As demonstrated by Chickering (1996) [12], this problem is NP-hard, indicating its computational difficulties and the absence of polynomial-time algorithms capable of solving all instances of the problem. For small networks involving few nodes, creating all possible combinations is computationally feasible, however, as the number of nodes increases, so does the number of permutations which increases exponentially. In cases where the domain is vast, automated methods may become impractical or fail to produce a model in a reasonable time frame. As a consequence, optimization algorithms have to be employed, to approximate solutions by choosing a network structure that offers improved performances compared to the vast search space of possible structures.

One of the advantages of Bayesian Networks is the ability to incorporate experts' knowledge via the manual creation of networks, this manual creation can be beneficial when dealing with a scarcity of data that can lead to a sparse representation when learning the network's structure. In an ideal scenario, the structure learning algorithm would find the optimal structure that maximizes the prediction accuracy and log-likelihood. However, in most cases, the algorithm reaches a local optima, a point in which the network is unable to improve any further given the current configuration, impeding improved performances.

To effectively exploit both the available data and the domain knowledge, a hybrid methodology was implemented. This approach began by constructing a manual structure based on the domain knowledge and subsequently fine-tuned the network architecture through structure learning techniques. When defining the approach, the final one adopted for representing the data ensured that the variety of nodes within the network remains limited. The dataset, as previously described, comprises company stock price changes, fundamental company information (such as ticker symbols and sectors), and variations in the S&P 500 index price. This streamlined dataset allowed for the manual construction of the network by following basic intuition rules.

Given the interdependent nature of the variables within the dataset, constructing the network manually proved relatively straightforward. For instance, a company's sector is inherently linked to the company itself (represented with the ticker symbol), likewise, the impact of company performances on the S&P index fluctuations is a natural correlation due to the S&P 500 representing the overall movement of its constituents.

This manual network construction created the foundations for further improvements through structure learning. Using the constructed data as in table 5.4, the network was trained making use of the Hill Climbing (HC) algorithm. The Hill Climbing algorithm [32] is part of the family of score-based methods, as described in 4.2.4 this family of methods assigns a score to each candidate Bayesian Network and tries to maximize this score. The difference between score-based methods lies in how each candidate is found and based on the chosen scoring function utilized to evaluate the model. This algorithm computes what is defined as a greedy search by looking at its neighboring models. The search is defined as greedy since the algorithm chooses the most immediate, or locally optimal, choice at each

step of the process to obtain the best outcome.

The basic hill climbing algorithm works as in Algorithm 1.

---

**Algorithm 1** Hill Climbing Algorithm

---

```

1: Evaluate the initial state
2: while not termination condition do
3:   Select an operator not yet applied to the current state (insertion, deletion, or reversing an edge)
4:   Apply the operator to generate a new state
5:   Evaluate the new state
6:   if new state is a final state then
7:     return new state as the solution
8:   else if new state is better than the current state then
9:     Set current state to new state
10:  end if
11: end while
12: return current state

```

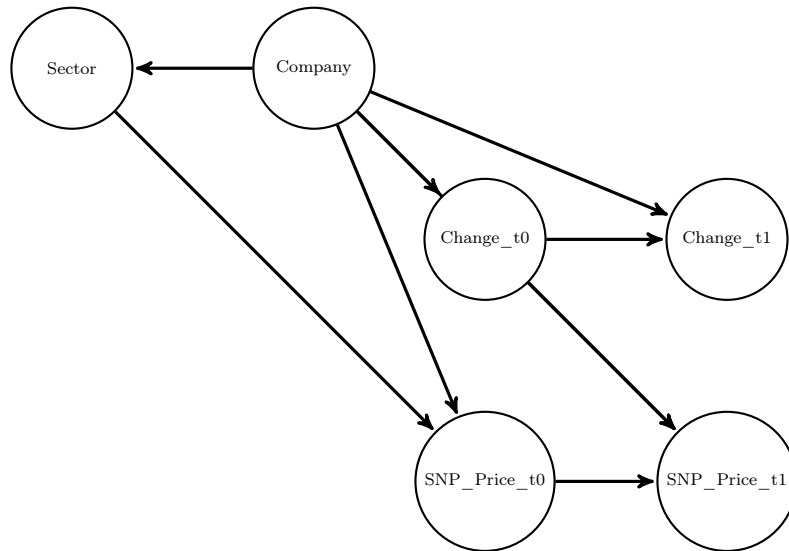
---

The bnlearn library, employed extensively throughout the project, supports the usage of various scoring functions such as BIC and AIC, defined in chapter 4.2.4. By default, the library uses BIC as the scoring function. While bnlearn is not designed with specific functions for training Dynamic Bayesian Networks (DBNs) - which differ in the structure to Bayesian Networks, primarily in their edge directions restricted in "forward" movements across time periods - this limitation can be overcome. The library allows modifications to the network structure through the use of *whitelist* and *blacklist* passed as parameters to the function used for the structure learning. Both the parameters contain lists of tuples representing edges. The *whitelist* comprises of edges that should always be included in the network, thereby unremovable. The *blacklist*, instead, comprises of edges that should not appear in the network, guiding the structure learning algorithms to avoid undesirable connections. Through the usage of those two parameters one can learn the structure of a Dynamic Bayesian Network. The blacklist was used to prevent edges that go "backwards", for each node at time  $t + 1$ , the blacklist includes an edge linking the node at time  $t + 1$  to the node at time  $t$  ensuring that only forward temporal connections are possible. The whitelist can also play a crucial role in the learning of the structure. In fact, one of the experiments conducted involved its use. After the manual network structure is defined, the edges involved in the network are passed through the use of this parameter, resulting in creating a network that retains the defined manual edges, while still incorporating additional edges learned through the algorithm.

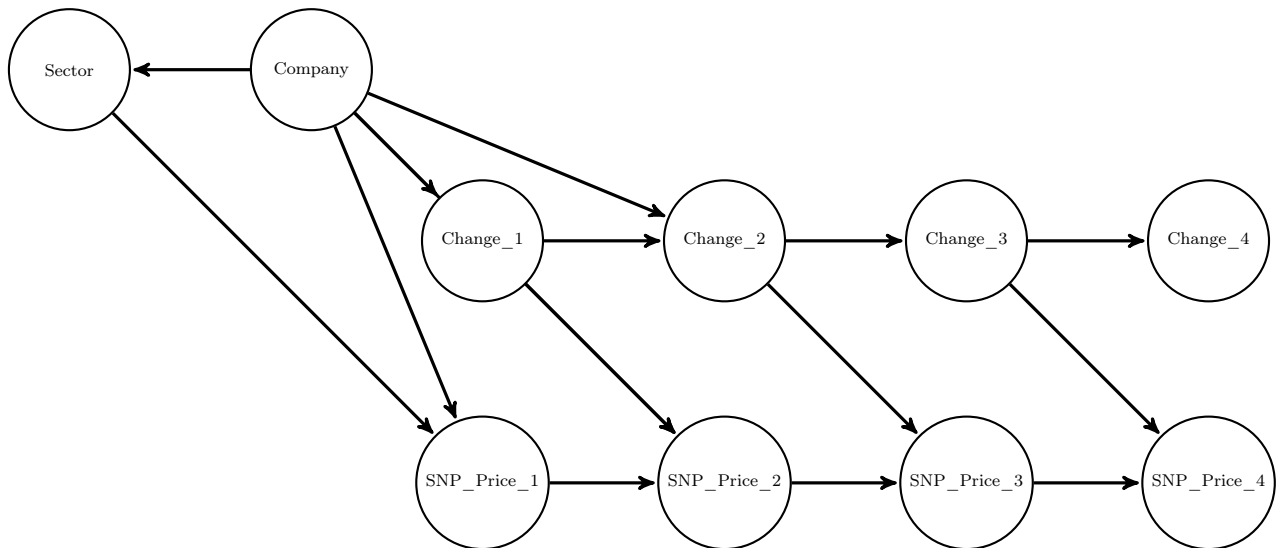
### Partitioned Dynamic Bayesian Network construction - Structure Unrolling

For building the *Partitioned Dynamic Bayesian Networks (PDBN)*, the process of constructing the Dynamic Bayesian Network (DBN), previously described, has to be repeated as many times as the number of cuts present in the PDBN + 1. This creates multiple DBNs where each potentially has a different structure based on the distribution of the batch of data utilized for its construction. The library bnlearn, although a powerful library under various aspects, does not natively support the use of PDBNs as a mean of a sequence of Dynamic Bayesian Networks. To overcome this limitation, a common technique called *structure unrolling* has to be employed. The goal of this technique is to transform a DBN into a standard Bayesian Network by explicitly representing each time slice. In Figure 5.1

we can see a representation of this process for a Dynamic Bayesian Network with 4 time slices, where Figures 5.1a and 5.1b respectively represent the network before and after unrolling the structure. After obtaining the structure of each sub-network, the following process involved connecting these sub-networks into a unique network that forms the Partitioned Dynamic Bayesian Network. To make sure that links were added to connect two different Dynamic Bayesian Networks, the edges in a network at time  $t + 1$  were used as a bridge between the two networks at time  $t + 1$  and  $t$ . Once this process was completed, the resulting network is a full Partitioned Dynamic Bayesian Network.



(A) DBN before unrolling its structure



(B) DBN after unrolling its structure

FIGURE 5.1: Example representation of the unrolling of a Dynamic Bayesian Network

# Chapter 6

## Results

In this chapter, an analysis of the results of the experiments conducted is presented. This involved extensive testing and evaluation across multiple network structures, structure learning techniques, and varying amounts of data. The objective was to rigorously assess the performance of the models and the impact of the amount of data under different configurations and data scales.

To this end, various network structures and cutset configurations were explored. The specific cutset tested are as in table 6.1.

Cutset Configuration	Model Description
{5, 10, 15}	PDBN-3 (Three cuts at time transitions 5, 10, 15)
{5, 10, 15, 20}	PDBN-4 (Four cuts at time transitions 5, 10, 15, 20)
{5, 10, 15, 20, 25}	PDBN-5 (Five cuts at time transitions 5, 10, 15, 20, 25)
{5, 10, 15, 20, 25, 30}	PDBN-6 (Six cuts at time transitions 5, 10, 15, 20, 25, 30)
{15, 30, 45}	PDBN-3 (Three cuts at time transitions 15, 30, 45)

TABLE 6.1: Cutset Configurations and Corresponding Models

These configurations represent different variations of the proposed structure, where, for instance, {5, 10, 15} corresponds to a *PDBN*-3, which is a Partitioned Dynamic Bayesian Networks (PDBN) model that incorporates three distinct cuts at the specified time transitions. Each cutset was chosen to evaluate the effects that the structure learning algorithm has on the model’s performance and the impact of different temporal granularities.

The temporal granularity is particularly relevant when training the network with different training data sizes to assess the impact on the network’s prediction capabilities. The experiments were conducted with varying dataset sizes, ranging from 500 to 2450, partitioned according to each cutset configuration, to evaluate how different data volumes and partitioning affect the model’s performance. Additionally, to analyze changes in market behavior and the impact of different prediction intervals, the metrics were tracked every 50 predictions, up to a maximum of 500 predictions. The specific cutset configurations, dataset sizes, and the corresponding amount of days of data used per each time slice are summarized in Table 6.2.

Tables 6.3 and 6.4 present the results and the associated metrics for the experiments conducted on models constructed using structure learning and not, respectively.

Cutset Configuration	Dataset Size	Days of Data per Time Slice
{5, 10, 15}	500	33
{5, 10, 15}	750	50
{5, 10, 15}	1000	67
{5, 10, 15}	1250	83
{5, 10, 15}	1500	100
{5, 10, 15}	1750	117
{5, 10, 15}	2000	133
{5, 10, 15}	2250	150
{5, 10, 15}	2450	163
{5, 10, 15, 20}	500	25
{5, 10, 15, 20}	750	38
{5, 10, 15, 20}	1000	50
{5, 10, 15, 20}	1250	63
{5, 10, 15, 20}	1500	75
{5, 10, 15, 20}	1750	88
{5, 10, 15, 20}	2000	100
{5, 10, 15, 20}	2250	113
{5, 10, 15, 20}	2450	123
{5, 10, 15, 20, 25}	500	20
{5, 10, 15, 20, 25}	750	30
{5, 10, 15, 20, 25}	1000	40
{5, 10, 15, 20, 25}	1250	50
{5, 10, 15, 20, 25}	1500	60
{5, 10, 15, 20, 25}	1750	70
{5, 10, 15, 20, 25}	2000	80
{5, 10, 15, 20, 25}	2250	90
{5, 10, 15, 20, 25}	2450	98
{5, 10, 15, 20, 25, 30}	500	17
{5, 10, 15, 20, 25, 30}	750	25
{5, 10, 15, 20, 25, 30}	1000	33
{5, 10, 15, 20, 25, 30}	1250	42
{5, 10, 15, 20, 25, 30}	1500	50
{5, 10, 15, 20, 25, 30}	1750	58
{5, 10, 15, 20, 25, 30}	2000	67
{5, 10, 15, 20, 25, 30}	2250	75
{5, 10, 15, 20, 25, 30}	2450	82
{15, 30, 45}	500	11
{15, 30, 45}	750	17
{15, 30, 45}	1000	22
{15, 30, 45}	1250	28
{15, 30, 45}	1500	33
{15, 30, 45}	1750	39
{15, 30, 45}	2000	44
{15, 30, 45}	2250	50
{15, 30, 45}	2450	54

TABLE 6.2: Cutset Configurations and Days per Node for Each Dataset Size

# Executions with Structure Learning

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Exact Accuracy	Weighted Accuracy	Exact Prediction	Near Miss
500	500	15_30_45	1.865732	2.210292	-1070.512858	-1.755748	0.194	0.380944	97	132
	750	15_30_45	1.982211	2.261116	-1179.792355	-1.744432	0.188	0.368470	94	132
	1000	15_30_45	2.040465	2.354330	-1111.017423	-1.824024	0.194	0.350461	97	133
	1250	15_30_45	2.082159	2.359415	-1095.791854	-1.892081	0.190	0.327777	95	126
	1500	15_30_45	1.973429	2.222980	-938.967996	-1.888283	0.204	0.355178	102	133
	1750	15_30_45	1.770959	2.012041	-841.771214	-1.878328	0.248	0.384650	124	144
	2000	15_30_45	1.826911	2.049640	-945.065500	-1.896317	0.192	0.349001	96	143
	2250	15_30_45	1.987005	2.210760	-1088.518929	-1.885416	0.218	0.334374	109	126
	2450	15_30_45	2.052136	2.292538	-1123.743773	-1.845320	0.194	0.337652	97	108
	500	5_10_15	1.822328	2.062688	-1062.899750	-1.750503	0.252	0.390117	126	126
	750	5_10_15	1.933162	2.115694	-1190.078756	-1.740421	0.198	0.372663	99	134
	1000	5_10_15	2.085298	2.379187	-1109.786655	-1.814685	0.200	0.344056	100	116
	1250	5_10_15	2.086942	2.351504	-1150.170113	-1.895525	0.184	0.325094	92	109
	1500	5_10_15	1.952572	2.196272	-964.777628	-1.888587	0.208	0.352300	104	122
	1750	5_10_15	1.836646	2.075793	-843.440990	-1.893769	0.218	0.370687	109	148
	2000	5_10_15	1.846394	2.065205	-958.323813	-1.901557	0.222	0.348680	111	134
	2250	5_10_15	1.916873	2.134556	-1049.310887	-1.877415	0.210	0.348633	105	139
	2450	5_10_15	2.020900	2.249550	-1145.306781	-1.858285	0.190	0.335753	95	135
	500	5_10_15_20	1.796897	2.047302	-1052.363127	-1.747373	0.224	0.392660	112	128
	750	5_10_15_20	1.934442	2.129304	-1225.739039	-1.744698	0.254	0.372497	127	115
	1000	5_10_15_20	2.085707	2.389248	-1115.392463	-1.821467	0.184	0.341943	92	101
	1250	5_10_15_20	2.080624	2.354345	-1124.363169	-1.893655	0.192	0.327825	96	118
	1500	5_10_15_20	1.957230	2.203632	-958.035916	-1.890334	0.210	0.352895	105	137
	1750	5_10_15_20	1.841321	2.081039	-855.280591	-1.895130	0.220	0.368195	110	149
	2000	5_10_15_20	1.845762	2.066433	-955.386661	-1.898447	0.198	0.350807	99	164
	2250	5_10_15_20	1.912864	2.130891	-1050.678802	-1.873591	0.224	0.352653	112	120
	2450	5_10_15_20	2.008341	2.236363	-1138.114800	-1.853656	0.214	0.341275	107	113
	500	5_10_15_20_25	1.783975	2.045089	-1017.828267	-1.751546	0.216	0.392853	108	140
	750	5_10_15_20_25	1.934690	2.160582	-1196.562960	-1.744036	0.216	0.374907	108	117
	1000	5_10_15_20_25	2.076004	2.387042	-1117.755384	-1.820042	0.210	0.344062	105	114

Continued on next page

TABLE 6.3: Execution Results and Metrics: 500 predictions with structure learning (continued)

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Exact Accuracy	Weighted accuracy	Exact Prediction	Near Miss
1250	5	10_15_20_25	2.097467	2.373850	-1122.133270	-1.894705	0.192	0.325256	96	128
1500	5	10_15_20_25	1.957388	2.209263	-954.987225	-1.893982	0.194	0.351464	97	145
1750	5	10_15_20_25	1.819415	2.060949	-850.194729	-1.890791	0.214	0.372280	107	140
2000	5	10_15_20_25	1.836981	2.059868	-944.056772	-1.891174	0.194	0.352187	97	163
2250	5	10_15_20_25	1.926692	2.151140	-1054.467796	-1.875209	0.206	0.349247	103	147
2450	5	10_15_20_25	2.011312	2.239885	-1137.456634	-1.852128	0.186	0.340685	93	127
500	5	10_15_20_25_30	1.848348	2.154630	-1039.961795	-1.755147	0.226	0.384152	113	139
750	5	10_15_20_25_30	1.958190	2.189915	-1180.587658	-1.747305	0.244	0.371686	122	115
1000	5	10_15_20_25_30	2.078447	2.393738	-1110.557259	-1.821765	0.184	0.344690	92	130
1250	5	10_15_20_25_30	2.089168	2.371296	-1122.756183	-1.893381	0.188	0.328081	94	123
1500	5	10_15_20_25_30	1.973405	2.225978	-959.964686	-1.896693	0.196	0.351670	98	134
1750	5	10_15_20_25_30	1.813873	2.053255	-849.991615	-1.888172	0.226	0.374398	113	145
2000	5	10_15_20_25_30	1.823130	2.046540	-938.098492	-1.889542	0.222	0.353021	111	147
2250	5	10_15_20_25_30	1.940443	2.162325	-1060.095667	-1.877816	0.198	0.345469	99	139
2450	5	10_15_20_25_30	2.038922	2.274503	-1133.625889	-1.850515	0.208	0.337207	104	105
End of the table										

TABLE 6.3: Execution Results and Metrics: 500 predictions with structure learning (end)

# Executions without Structure Learning

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Exact Accuracy	Weighted Accuracy	Exact Prediction	Near Miss
500	500	15_30_45	1.003860	1.783121	-598.455081	-1.430377	0.254	0.676417	127	202
	750	15_30_45	1.182149	1.949456	-721.149239	-1.441832	0.258	0.622169	129	175
	1000	15_30_45	1.240127	2.000143	-711.614552	-1.456160	0.268	0.607780	134	190
	1250	15_30_45	1.251528	1.984559	-741.262985	-1.466726	0.222	0.599737	111	185
	1500	15_30_45	1.252617	1.964543	-713.068213	-1.461650	0.244	0.595009	122	167
	1750	15_30_45	1.163691	1.810748	-656.640681	-1.433268	0.282	0.608998	141	185
	2000	15_30_45	1.192097	1.802263	-688.222492	-1.445674	0.240	0.593797	120	194
	2250	15_30_45	1.401071	2.045905	-845.025705	-1.501595	0.262	0.540820	131	172
	2450	15_30_45	1.546524	2.222936	-954.528388	-1.529078	0.202	0.509827	101	157
	500	5_10_15	1.000672	1.775279	-585.874903	-1.428318	0.334	0.678012	167	162
	750	5_10_15	1.174124	1.933820	-729.842386	-1.440118	0.260	0.623280	130	162
	1000	5_10_15	1.275999	2.042256	-727.811903	-1.461322	0.252	0.598189	126	171
	1250	5_10_15	1.259280	1.990962	-766.485579	-1.466463	0.252	0.598268	126	172
	1500	5_10_15	1.232461	1.941322	-702.429296	-1.458755	0.260	0.600158	130	176
	1750	5_10_15	1.218870	1.881911	-673.846362	-1.454753	0.294	0.595416	147	165
	2000	5_10_15	1.192608	1.805656	-692.302117	-1.444125	0.300	0.594656	150	168
	2250	5_10_15	1.332997	1.964654	-798.319349	-1.481043	0.278	0.558525	139	171
	2450	5_10_15	1.521165	2.189246	-956.945824	-1.527951	0.216	0.513039	108	157
	500	5_10_15_20	0.996833	1.767818	-583.328464	-1.427680	0.298	0.678315	149	193
	750	5_10_15_20	1.169432	1.926131	-732.951444	-1.440701	0.272	0.624027	136	178
	1000	5_10_15_20	1.272719	2.037206	-725.524251	-1.461708	0.294	0.598682	147	159
	1250	5_10_15_20	1.251278	1.981806	-756.954103	-1.465633	0.242	0.600219	121	189
	1500	5_10_15_20	1.234078	1.941893	-706.067276	-1.460016	0.270	0.599064	135	168
	1750	5_10_15_20	1.215934	1.874975	-671.019822	-1.453090	0.236	0.595637	118	196
	2000	5_10_15_20	1.193193	1.805532	-685.291083	-1.443624	0.276	0.595485	138	189
	2250	5_10_15_20	1.338484	1.971922	-809.143372	-1.481993	0.282	0.557199	141	168
	2450	5_10_15_20	1.516373	2.183911	-951.537216	-1.526412	0.244	0.514161	122	154
	500	5_10_15_20_25	0.995290	1.765190	-584.418517	-1.427930	0.372	0.678173	170	186
	750	5_10_15_20_25	1.166645	1.925571	-718.414778	-1.439983	0.278	0.624910	139	170
	1000	5_10_15_20_25	1.268464	2.030482	-719.746151	-1.460244	0.242	0.599831	121	176

Continued on next page

TABLE 6.4: Execution Results and Metrics: 500 predictions without structure learning (continued)



Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Exact Accuracy	Weighted accuracy	Exact Prediction	Near Miss
1250	5	10_15_20_25	1.251035	1.984003	-745.812097	-1.465449	0.260	0.600409	130	170
1500	5	10_15_20_25	1.239391	1.946627	-708.998569	-1.462271	0.292	0.597481	146	161
1750	5	10_15_20_25	1.206401	1.861678	-670.816302	-1.449747	0.276	0.597365	138	182
2000	5	10_15_20_25	1.188123	1.799993	-685.904966	-1.441226	0.390	0.596418	133	195
2250	5	10_15_20_25	1.347470	1.983980	-814.493280	-1.485908	0.244	0.554917	122	187
2450	5	10_15_20_25	1.516362	2.183854	-946.249634	-1.527007	0.222	0.514380	111	160
500	5	10_15_20_25_30	1.000525	1.776164	-582.484077	-1.429221	0.352	0.677437	154	176
750	5	10_15_20_25_30	1.172594	1.936522	-717.232988	-1.441371	0.242	0.624415	121	172
1000	5	10_15_20_25_30	1.261875	2.023666	-718.998029	-1.459552	0.248	0.601708	124	170
1250	5	10_15_20_25_30	1.253450	1.986978	-748.302426	-1.465262	0.248	0.599589	124	165
1500	5	10_15_20_25_30	1.251744	1.961588	-716.253209	-1.463692	0.276	0.594897	138	190
1750	5	10_15_20_25_30	1.193727	1.847542	-665.922985	-1.445291	0.282	0.601954	141	178
2000	5	10_15_20_25_30	1.183880	1.794923	-687.621372	-1.440808	0.268	0.597027	134	187
2250	5	10_15_20_25_30	1.359063	1.997024	-821.231622	-1.490161	0.252	0.551301	126	170
2450	5	10_15_20_25_30	1.530083	2.202827	-949.085000	-1.527199	0.204	0.512623	102	171

End of the table

TABLE 6.4: Execution Results and Metrics: 500 predictions without structure learning (end)

These tables focus solely on the most extensive prediction setting, which is the one where the number of predictions considered reached the maximum of 500. The tables include a variety of metrics, crucial for evaluating the effectiveness and better analyzing the model, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared, Log-Likelihood, Exact accuracy, Weighted Accuracy, the number of Exact Predictions and the number of Near Misses. The entries highlighted in red in each table denote the best values for each computed metric across all tested configurations (within the table content). The MAE is a metric that evaluates the accuracy of a model by measuring the absolute difference between the prediction and the actual value. Unlike other metrics, only the magnitude is taken into account and not the direction. This property ensures that the metric is not affected by over and under estimations, as shown in eq 6.1

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (6.1)$$

where  $N$  represents the total number of predictions,  $y_i$  denotes the actual value for the  $i$ -th observation, and  $\hat{y}_i$  is the predicted value for the  $i$ -th observation.

In this context, the labels that were created, representing the change in direction of the price of a specific company compared to the previous day, were used to compute this metric. To achieve this, these changes in direction were categorized into seven distinct classes: High Decrease (1), Decrease (2), Low Decrease (3), No Change (4), Low Increase (5), Increase (6), High Increase (7). Each label was assigned a numerical value enabling the calculation of the distance between actual and predicted observations.

A similar approach was used to compute the weighted accuracy, obtained by taking into account the distance between the prediction and the actual value. A lower value was given where the distance from the prediction to the actual value was higher. This approach was taken into account only for neighboring predictions, that is, those where the distance between the prediction and the actual value was not greater than 1, giving a maximum score for a correct prediction and 0.5 for a neighboring prediction. Let  $y_i$  and  $\hat{y}_i$  respectively be the actual and predicted labels for  $i$ -th observation, and  $N$  the total number of observations, the weighted score is defined as:

$$w_i = \begin{cases} 1 & \text{if } |y_i - \hat{y}_i| = 0, \\ 0.5 & \text{if } |y_i - \hat{y}_i| = 1, \\ 0 & \text{otherwise.} \end{cases} \quad (6.2)$$

The weighted accuracy  $W$  is computed as the average of these weighted scores as:

$$W = \frac{1}{N} \sum_{i=1}^N w_i \quad (6.3)$$

For a deeper exploration of all tested configurations, including varying numbers of predictions and different temporal granularities, a complete dataset containing all the conducted tests is provided in Table A.1 in the appendix for models where structure learning was applied, and in Table B.1 for model where structure learning was not applied.

After analyzing the results of the various runs, confusion matrices were created to evaluate the distribution of the predictions. Tables 6.5 and 6.6 summarize the predictions

across all models when structure learning was applied and when it was not, respectively. These tables show the cumulative counts of predictions for each scenario. To provide a clearer view of the prediction distribution, Tables 6.7 and 6.8 show the accuracy of the predictions per label.

		Actual Value							Total
		High Decrease	Decrease	Low Decrease	No Change	Low Increase	Increase	High Increase	
Predicted Value	High Decrease	7007291	3681890	2982437	4621016	3859630	3659964	2477005	28289233
	Decrease	354803	481027	279099	245550	142030	113730	76586	1692825
	Low Decrease	433228	462117	795219	417494	256832	314823	185581	2865294
	No Change	4757189	2882308	3154565	7650639	4066708	4315533	3064766	29891708
	Low Increase	1417555	622486	608871	1309980	1624589	1132879	828973	7545333
	Increase	471674	232812	409571	651679	473289	786394	368899	3394318
	High Increase	13055	5204	10529	13747	12288	6517	14899	76239
Total		14454795	8367844	8240291	14910105	10435366	10329840	7016709	73754950

TABLE 6.5: Combined Predictions for all tested models with Structure Learning

		Actual Value							Total
		High Decrease	Decrease	Low Decrease	No Change	Low Increase	Increase	High Increase	
Predicted Value	High Decrease	8797817	1986810	478467	1409045	2125071	1521610	1338885	17657705
	Decrease	1143981	4887271	174311	312147	447960	274352	236486	7476508
	Low Decrease	395366	142130	4488305	1113185	363279	580396	284673	7367334
	No Change	1339732	369640	1932557	9045120	1577925	2240044	1392193	17897211
	Low Increase	1393689	523004	359699	1128976	4448972	1076285	907961	9838586
	Increase	875759	279090	594729	1355324	970370	4154589	801228	9031089
	High Increase	508451	179899	212223	546308	501789	482564	2055283	4486517
Total		14454795	8367844	8240291	14910105	10435366	10329840	7016709	73754950

TABLE 6.6: Combined Predictions for all tested models without Structure Learning

		Actual Value						
		High Decrease	Decrease	Low Decrease	No Change	Low Increase	Increase	High Increase
Predicted Value	High Decrease	48.48	44.00	36.19	30.99	36.99	35.43	35.30
	Decrease	2.45	5.75	3.39	1.65	1.36	1.10	1.09
	Low Decrease	3.00	5.52	9.65	2.80	2.46	3.05	2.64
	No Change	32.91	34.45	38.28	51.31	38.97	41.78	43.68
	Low Increase	9.81	7.44	7.39	8.79	15.57	10.97	11.81
	Increase	3.26	2.78	4.97	4.37	4.54	7.61	5.26
	High Increase	0.09	0.06	0.13	0.09	0.12	0.06	0.21

TABLE 6.7: Accuracy of models constructed with Structure Learning

		Actual Value						
		High Decrease	Decrease	Low Decrease	No Change	Low Increase	Increase	
Predicted Value	High Decrease	60.86	23.74	5.81	9.45	20.36	14.73	19.08
	Decrease	7.91	58.41	2.12	2.09	4.29	2.66	3.37
	Low Decrease	2.74	1.70	54.47	7.47	3.48	5.62	4.06
	No Change	9.27	4.42	23.45	60.66	15.12	21.69	19.84
	Low Increase	9.64	6.25	4.37	7.57	42.63	10.42	12.94
	Increase	6.06	3.34	7.22	9.09	9.30	40.22	11.42
	High Increase	3.52	2.15	2.58	3.66	4.81	4.67	29.29

TABLE 6.8: Accuracy of models constructed without Structure Learning

The Pearson correlation coefficient measures the strength and the direction of the relationship between two datasets. Figures 6.1 and 6.2, use this coefficient to understand how the predictive capabilities of the network are affected when predicting older data points. The first figure focuses on models that make use of structure learning, while the second examines models without. The tables show a variety of correlations between the number of predictions in the past — representing the amount of data that was predicted, the more the predictions, the further in the past it was predicted — and several performance metrics such as the Weighted Accuracy, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-Squared, and Log-Likelihood, where each is categorized by the number of predictions that were executed for each batch.

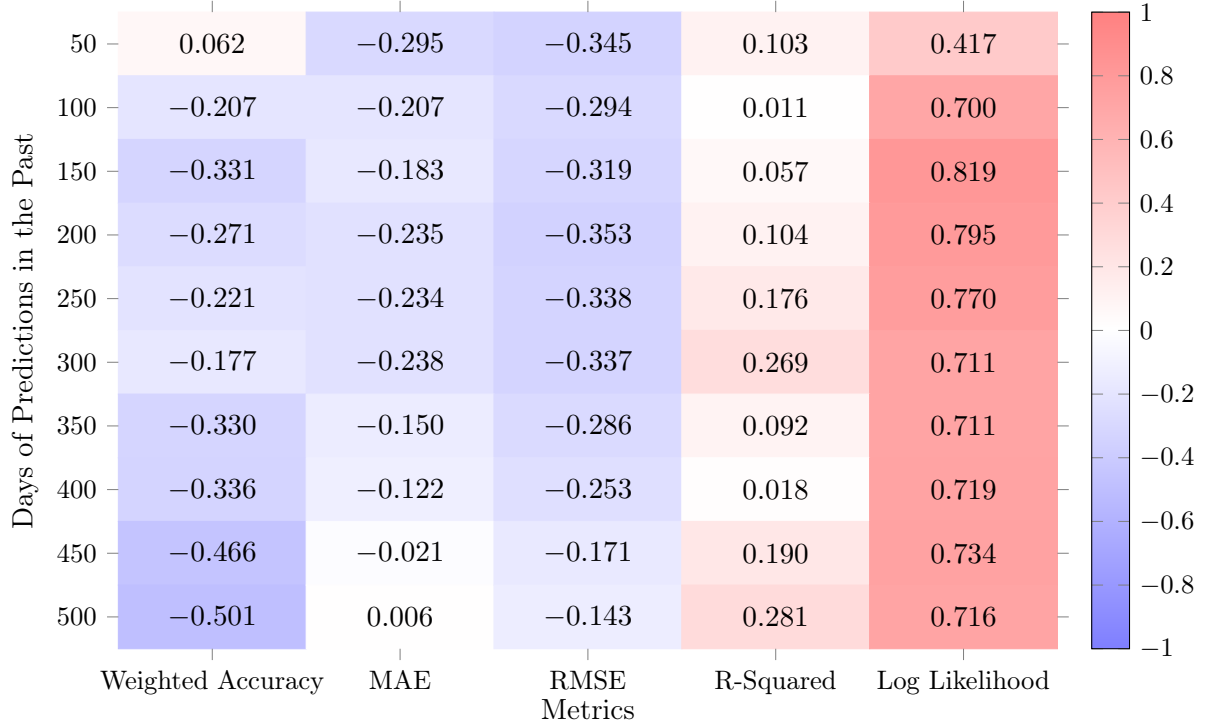


FIGURE 6.1: Pearson correlation between training sample size and metrics, grouped by the number of days of historical data used for predictions with structure learning

In statistics, when there is a high variance in the correlation, relying solely on the

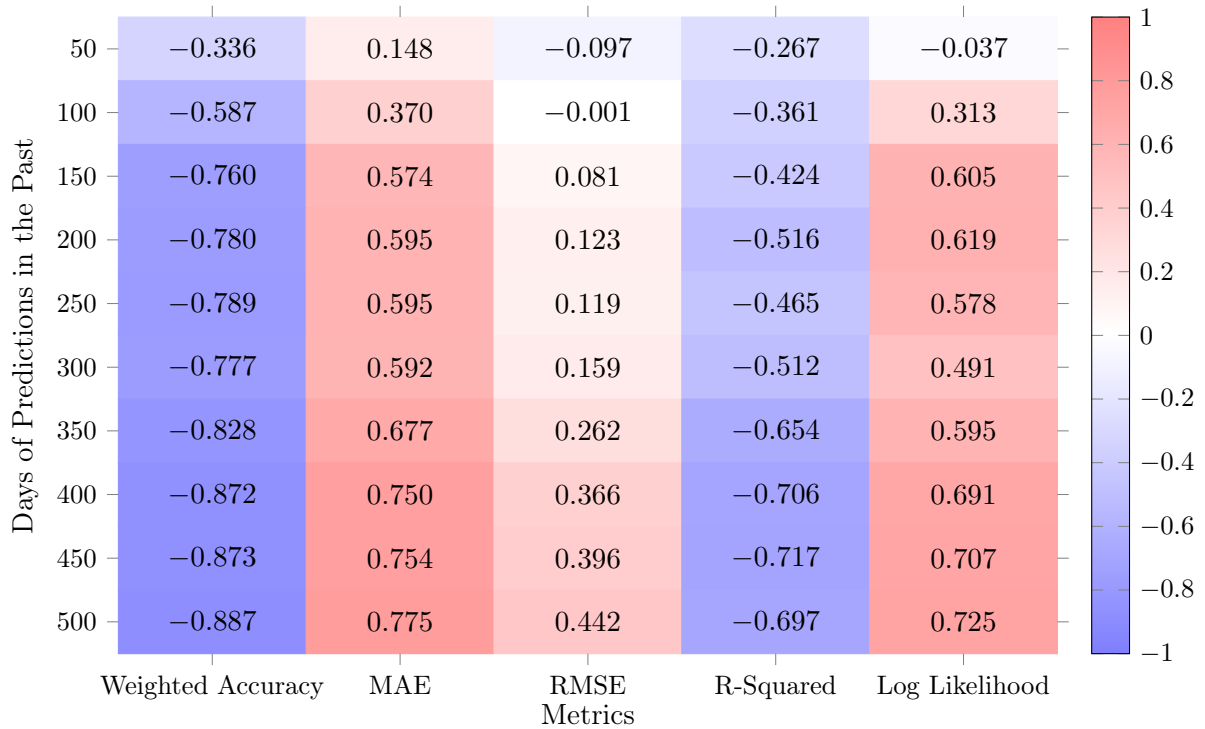


FIGURE 6.2: Pearson correlation between training sample size and metrics, grouped by the number of days of historical data used for predictions when no structure learning is applied

Pearson correlation coefficient without accounting for a third influencing random variable can result in misleading conclusions. In such cases, partial correlation provides more accurate results. Partial correlation helps to remove the effect of this third variable — in this case, the number of predictions — isolating the relationship to the variables we are more interested in.

To compute the partial correlation between two variables while controlling for the effect of a third variable, we first need to regress each variable (the metrics and the number of training days) on the third variable (the number of predictions) and obtain their residuals. The residuals represent the portion of the variables that cannot be explained by the third variable, allowing us to examine their direct relationship.

Regressing a variable  $Y$  on another variable  $X$  means modeling  $Y$  as a function of  $X$ , to understand the relationship between them. Mathematically, this means fitting a linear equation of the form:

$$Y = \beta_0 + \beta_1 X + \varepsilon \quad (6.4)$$

In this regression equation,  $Y$  is the dependent variable we want to explain, such as a model's *metric* (e.g. weighted accuracy).  $X$  is the independent variable used to explain changes in  $Y$ , such as the *number of predictions* made. The term  $\beta_0$  is the intercept, representing the value of the metric when the number of predictions is zero. The coefficient  $\beta_1$  is the slope, indicating how much the metric  $Y$  changes with a one-unit increase in  $X$  (the number of predictions). For example, a positive  $\beta_1$  suggests that more predictions lead to higher accuracy.

The error term,  $\varepsilon$ , represents the residuals — the part of  $Y$  not explained by  $X$ . By examining these residuals, we can isolate the effect of the number of predictions and better understand how other factors, like training days, independently affect the metric.

Considering the computed metrics (denoted as *metric*) and the number of days of data used for the training (denoted as *days*), the variable representing the number of predictions (denoted as *predictions*) is treated as the control variable. The following step is to regress the *metric* and *days* on the number of predictions using Ordinary Least Squares (OLS) regression to obtain the residuals.

Finally, the *partial correlation* is obtained by computing the Pearson correlation coefficient given the residuals  $\varepsilon_{days}$  and  $\varepsilon_{metrics}$  as in eq 6.5.

$$\begin{aligned} \text{metric} &= \beta_0 + \beta_1 \cdot \text{predictions} + \varepsilon_{\text{metric}} \\ \text{days} &= \alpha_0 + \alpha_1 \cdot \text{predictions} + \varepsilon_{\text{days}} \\ r_{\text{partial}} &= \text{Corr}(\varepsilon_{\text{days}}, \varepsilon_{\text{metrics}}) \end{aligned} \tag{6.5}$$

In addition to this approach, a weighted Pearson correlation coefficient was also calculated. In this method, each entry from Figures 6.1 and 6.2 was weighted based on the number of predictions computed in each batch, represented by the Y axis, and averaged over the total number of tests executed. The results of both the partial and weighted correlation analyses are presented in Table 6.9 which gathers both approaches (partial and weighted correlation).

	With Structure Learning		Without Structure Learning	
	Partial Correlation	Weighted Correlation	Partial Correlation	Weighted Correlation
Weighted Accuracy	-0.208437	-0.208132	-0.703073	-0.698492
MAE	-0.174929	-0.174534	0.519402	0.515491
RMSE	-0.274799	-0.274154	0.133663	0.132440
R-Squared	0.107454	0.105765	-0.491547	-0.480551
Log Likelihood	0.694201	0.689369	0.428697	0.426119

TABLE 6.9: Comparison of Partial and Weighted Pearson Correlations with and without Structure Learning

The log-likelihood is a fundamental statistical function used to measure how well a particular model explains a given set of data. In the context of this thesis, it was employed in all the tests conducted, as presented in Tables 6.3 and 6.4, and in Figures 6.1 and 6.2.

In a Bayesian Network (BN) we consider a set of random variables  $X = \{X_1, X_2, \dots, X_N\}$  where each  $X_i$  corresponds to a node, and  $N$  is the total number of nodes in the network. Each  $\theta = \{\theta_1, \theta_2, \dots, \theta_N\}$  is a set of parameters that represents the CPTs of a node. Specifically, for each node  $X_i$ , there is an associated set of conditional probabilities  $\theta_i$  given every possible configuration of its parents  $Pa(X_i)$ .

The likelihood function, denoted as  $L(\theta; \mathcal{D})$ , tells us how plausible our observed data  $\mathcal{D}$  is, given the parameters  $\theta$  of the model. In the context of a BN, it essentially quantifies how well the network’s structure, with its current parameters, explains the observed data.

Given a dataset  $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_M\}$  consisting of  $M$  independent observations, the likelihood function is expressed as in eq 6.6.

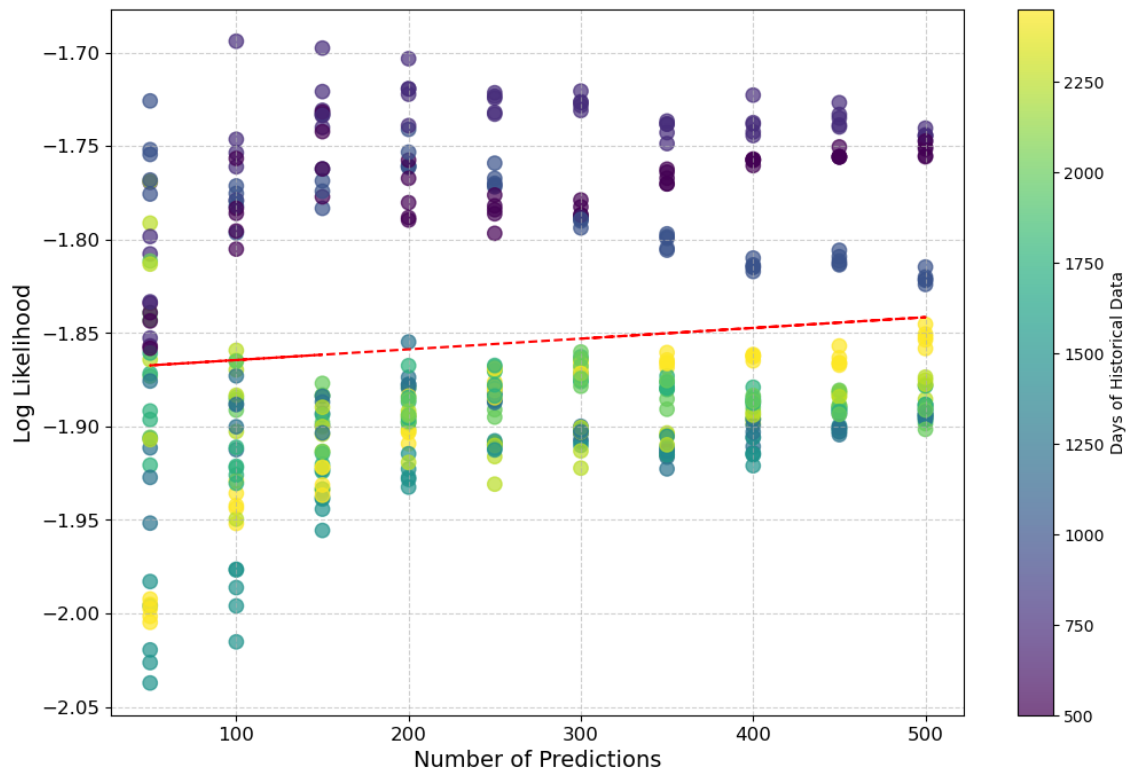
$$L(\theta; \mathcal{D}) = P(\mathcal{D} | \theta) = \frac{1}{M} \prod_{j=1}^M P(\mathbf{d}_j | \theta) \quad (6.6)$$

where  $P(\mathbf{d}_j | \theta)$  is the probability of observing the  $j$ -th data instance  $\mathbf{d}_j$  given the model parameters. However, directly computing the likelihood often involves multiplying small probabilities multiple times. Since probabilities are values between 0 and 1, their product can become extremely small. This repeated multiplication can lead to a numerical issue called underflow. Underflow occurs when the product of a multiplication is smaller than the smallest value a computer can represent with its floating-point precision. As a consequence, the really small values are approximated to zero, which leads the overall likelihood computation to also become zero.

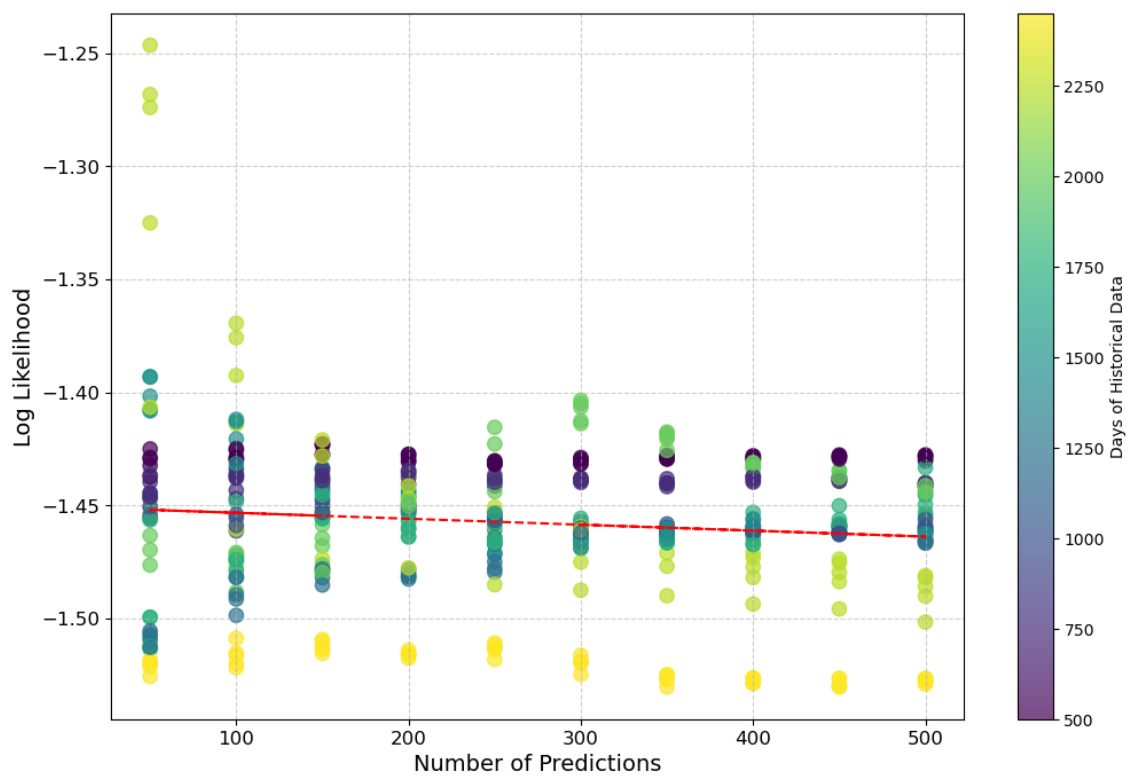
To avoid underflow and simplify computations, the logarithm of the likelihood function is taken, which leads to converting the product of probabilities into a sum of logarithms. This results in the log-likelihood, which is then averaged over the number of predictions as defined in equation 6.7.

$$\ell(\theta | \mathcal{D}) = \frac{1}{M} \sum_{j=1}^M \log P(\mathbf{d}_j | \theta) \quad (6.7)$$

Figures 6.3a and 6.3b, show the log-likelihood values plotted against the number of predictions on the x-axis. Figure 6.3a displays the results for models that employed structure learning, while Figure 6.3b presents the results for models without structure learning. Different models, trained using increasing amounts of historical data, are distinguished by color coding while the red line represents the linear regression fitted across all the data points to highlight the overall trend as we predict more into the past.



(A) Log Likelihood when structure learning is applied



(B) Log Likelihood when no structure learning is applied

FIGURE 6.3: Log Likelihood for models with and without structure learning applied, grouped by the number of predictions and colour coded by the amount of historical data used for training.



To better understand how the log-likelihood evolves as we predict further in the past, the log-likelihood was additionally plotted for each batch of predictions. Figure 6.4 shows the results when no structure learning is applied, whereas Figure 6.5 displays the results when structure learning is applied.

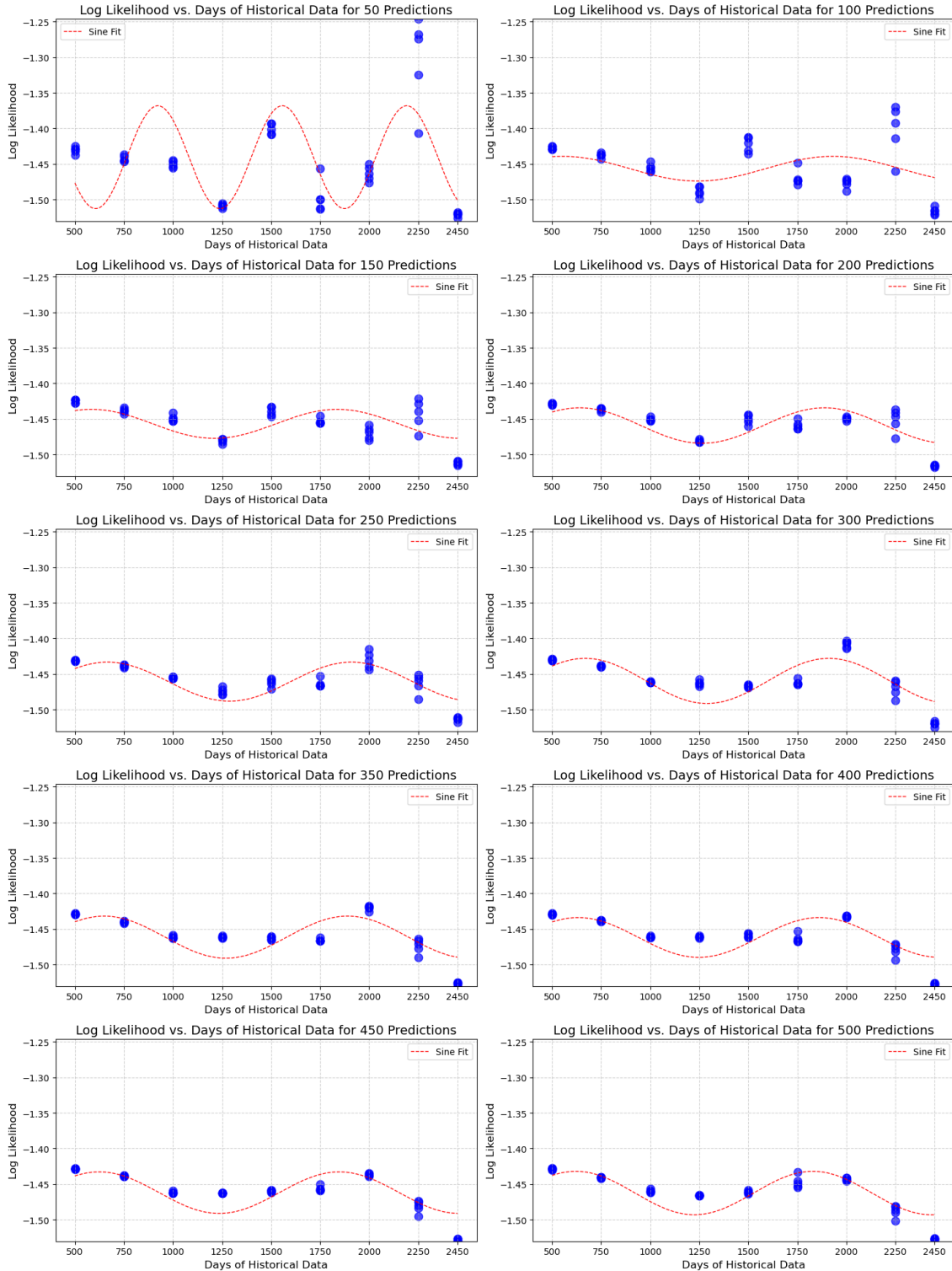


FIGURE 6.4: Log Likelihood vs amount of historical data used for training each model, grouped by the number of predictions, when no structure learning is applied

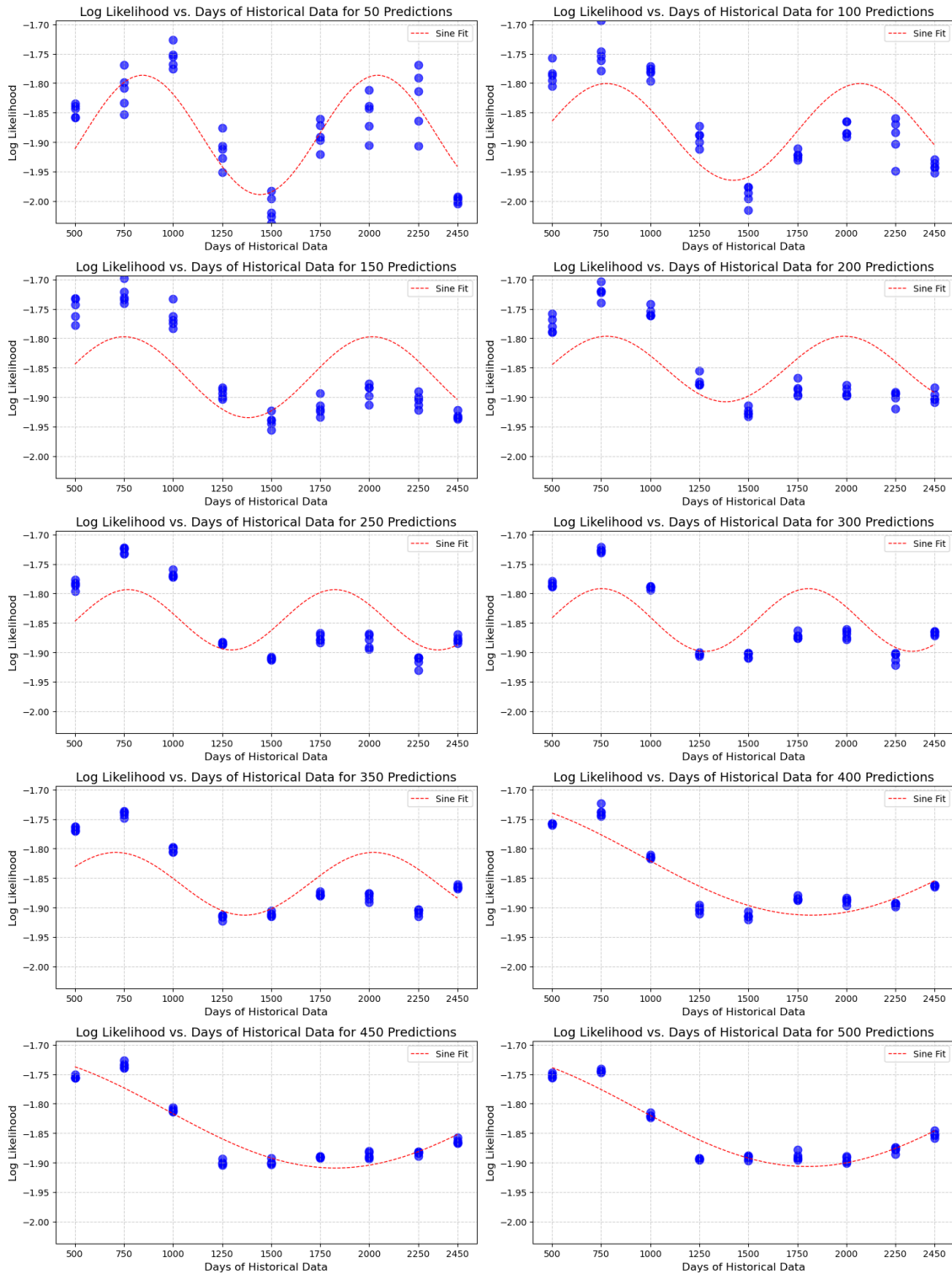
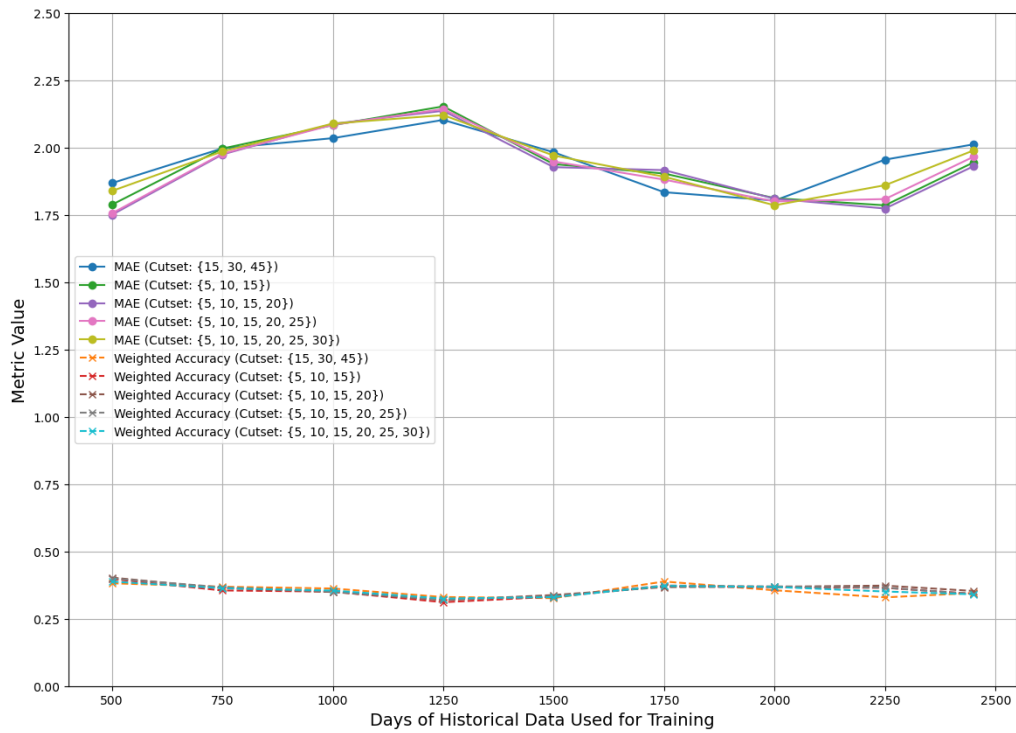
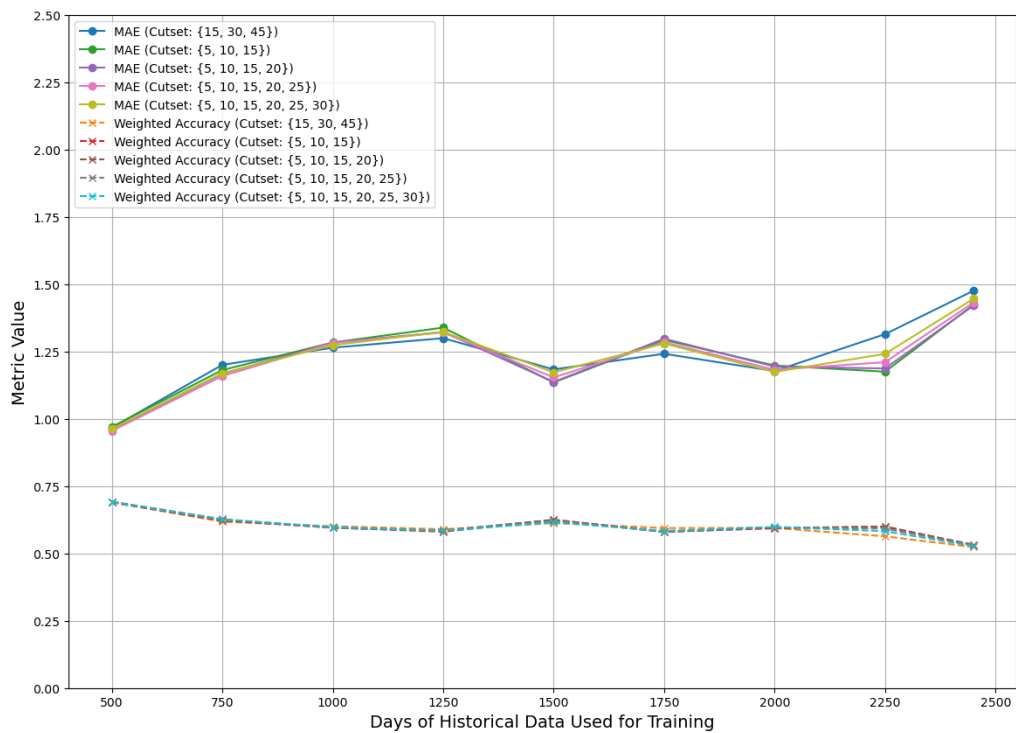


FIGURE 6.5: Log Likelihood vs amount of historical data used for training each model, grouped by the number of predictions, with structure learning applied

In addition to the log-likelihood, the MAE and the computed weighted accuracy were also visualized. Figure 6.6a presents these metrics for models that employed structure learning, while Figure 6.6b shows the results for models without structure learning. Both metrics are plotted together to highlight their indirect relationship, whereas the MAE improves (i.e. decreases), the weighted accuracy is expected to increase.



(A) MAE and Weighted Accuracy Over Time with Structure Learning Applied.

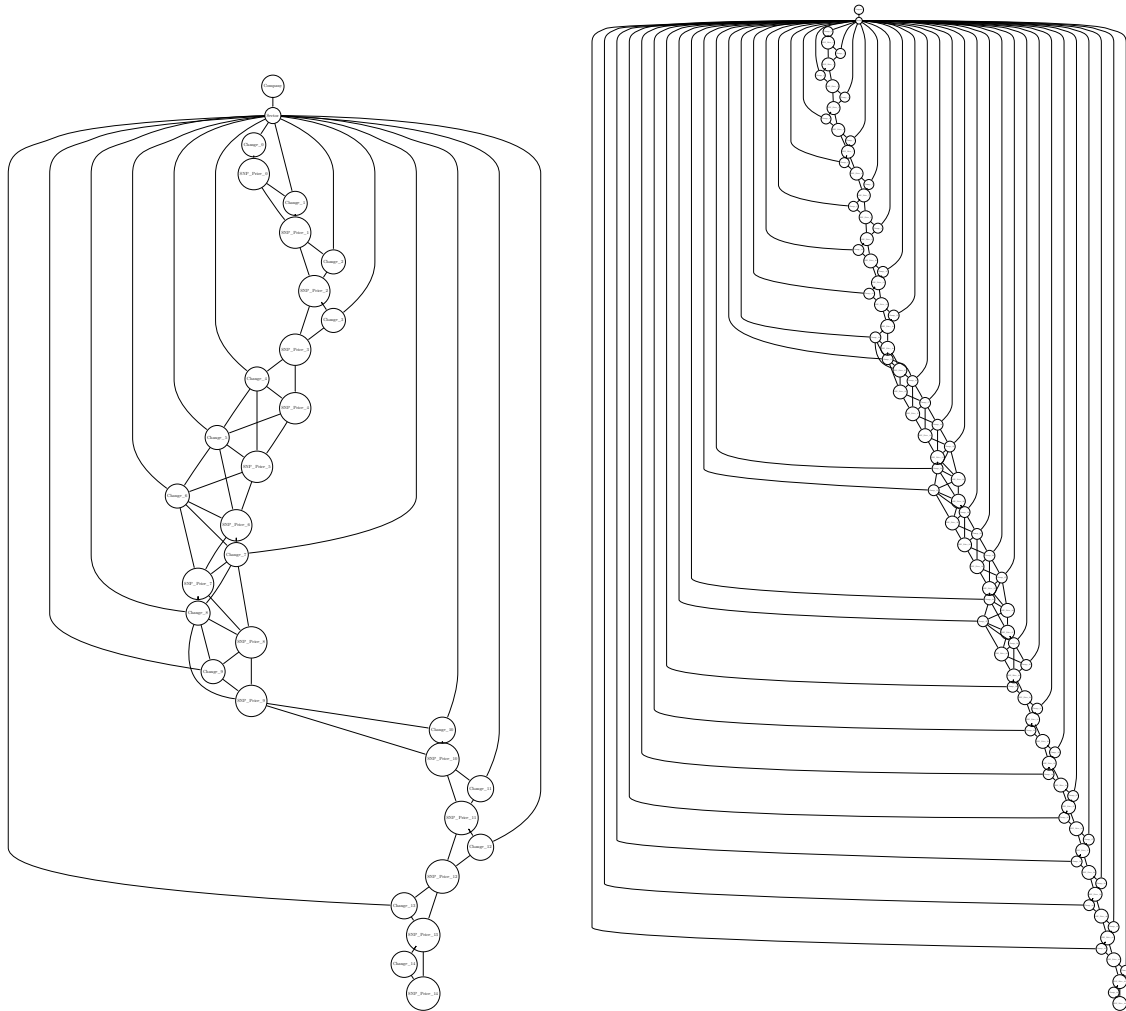


(B) MAE and Weighted Accuracy Over Time without Structure Learning Applied.

FIGURE 6.6: MAE and Weighted Accuracy Over Time for models with and without structure learning applied, with linear regression to indicate the trend direction

Figures 6.7a and 6.7b illustrate the structure of a Partitioned Dynamic Bayesian Net-

works (PDBN). Figure 6.7a represents a PDBN-3 with a cutset of  $\{5, 10, 15\}$ , while Figure 6.7b shows a PDBN-3 with cutset of  $\{15, 30, 45\}$ . These figures are presented to highlight the differences in size between two networks when the same number of cutsets is used, but with each cut spanning 5 and 15 time transitions, respectively.



(A) 3-PDBN - Partitioned Dynamic Bayesian Network 5, 10, 15 with 3 cuts - structure learning applied

(B) 3-PDBN - Partitioned Dynamic Bayesian Network 15, 30, 45 with 3 cuts - structure learning applied

FIGURE 6.7: Comparison of two Partitioned Dynamic Bayesian Networks (PDBNs) with different sizes and structures.

To compare the impact that the structure learning algorithm has on the network, Figure 6.7a is further analysed in Figure 6.8, where the network's structure is fully represented, allowing each node to be viewed in detail. In figure 6.9, the same network (a PDBN-3 with cutset  $\{5, 10, 15\}$ ) is displayed, to highlight the differences in the construction of the network. In the first case, it can be seen how each cutset has a different number of connections, reflecting the influence of the structure learning algorithm. In the second case, the constructed network is equivalent to a Dynamic Bayesian Network (DBN) with two time-invariant nodes, represented by the nodes "Sector" and "Company".

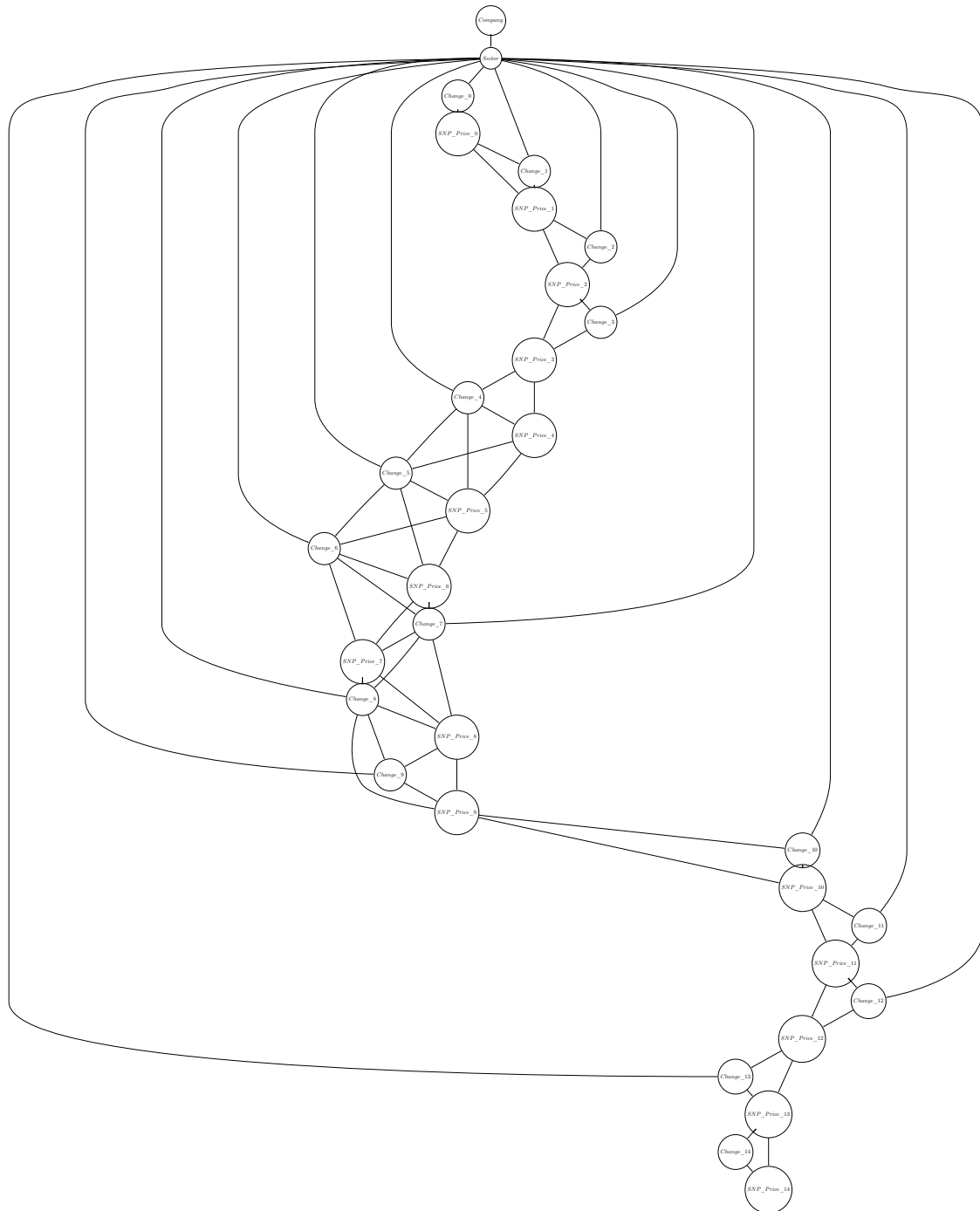


FIGURE 6.8: 3-PDBN - Partitioned Dynamic Bayesian Network 5, 10, 15 with 3 cuts - structure learning applied

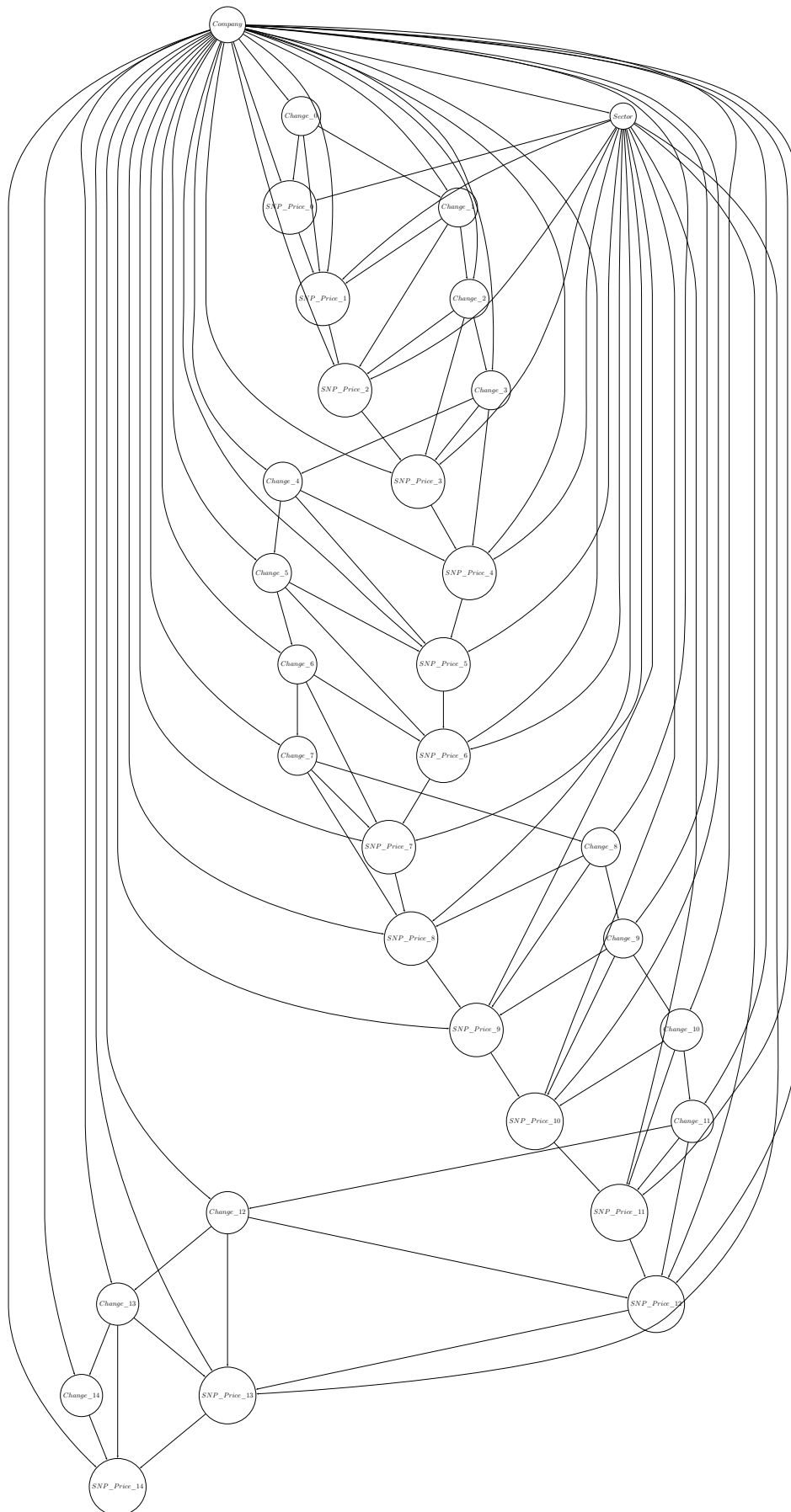


FIGURE 6.9: 3-PDBN - Partitioned Dynamic Bayesian Network 5, 10, 15 with 3 cuts - no structure learning applied

## Chapter 7

# Discussion

### 7.1 Tool Selection and Transition to R

Developing the PDBN involved different steps, during which several challenges were encountered, particularly due to limitations in the libraries initially chosen. The project began with the use of Python to maintain consistency with the data preparation phase. However, the need for a more suitable programming environment led to a shift towards using R.

Initially, the `PGMPY` [7] and `pomegranate` [35] libraries were selected for their full focus on Bayesian networks, and their ability to rapidly implement networks, which also allowed the representation of continuous variables through the usage of normal distributions. Despite these advantages, the libraries proved insufficiently flexible for the construction of a PDBN, especially due to their inability to link multiple Dynamic Bayesian Networks (DBNs) together. This capability is crucial for the successful construction of the network, which consequently meant a reassessment of the tools employed moving from Python to R where the library `bnlearn` [37] was used instead.

### 7.2 Data Handling and Factor Representation

One of the issues faced during the development involved limitations brought by the programming language and the complexity of the problem. Learning the parameters meant having to divide the dataset into multiple parts where each was fed to learn the parameters of each node of the network. This division meant the use of so called "factors" which are a type of data structure used in R for categorical data. This type of data structure differs from lists, which are capable of storing every type of data. Factors can be associated with the more classic enum that can be found in different programming languages like C, C++, and Java. Enum (which stands for "enumeration") is a data type composed of a set of constants where each constant is associated with an integer. Similarly, factors define categorical types of data. In this project, the labels represent the daily price changes, where the so-called "levels" correspond to these labels, whereas the values inside are represented by integers. In an ideal scenario, the factors would be structured as follows:

```
{High Decrease = 1, Decrease = 2, Low Decrease = 3, No Change = 4, Low Increase = 5, Increase = 6, High Increase = 7}
```

When splitting the data into smaller subsets, the number of factor levels can become inconsistent across different time transitions if one of the seven defined labels is absent in the created subset. Additionally, since each label is associated to an integer value, the order in which this levels are stored is meaningful when combining different factors.



In particular, when using the employed library `bnlearn`, it was necessary to establish a standardized level for each of the subsets created.

### 7.3 Tested Modeling Approaches

The project’s development had different phases, the initial approach saw the use of one node per company, and the idea was evaluating whether each company carried enough information in its price to be able to infer relationships between companies and the price of the S&P 500. Unfortunately, the results obtained were not convincing, it led to low prediction accuracy, high error rate, and mostly an unfeasible amount of time for the structure and parameter learning of the networks. Learning the structure of the network requires the companies’ data to be more informative, as the number of companies — random variables — increases, so does the complexity of the calculations to identify relationships between different nodes meaning that a larger amount of data is required.

To reduce complexity and increase the information on companies, sector data were incorporated. As described in 5.1.3, each company was assigned to its corresponding GICS sector. The aim was to model the PDBN by grouping companies being part of the same sector, constructing a model where each random variable — or node — represented the distribution of the daily stock price changes from day  $t$  to day  $t + 1$ . However, this approach was also found to be not optimal. Assigning all companies to a single node without accounting for individual characteristics about the company resulted in treating each company with the same importance, thereby losing any unique property that might influence price movements.

To address this, “Company” was included as an external entity, influencing sector-level dependencies. However, the S&P 500 is an index where companies are weighted based on their market capitalization, where larger companies constitute 6 – 7% of the index each while the smaller companies represent as little as 0.01%. This significant difference in weight, combined with the sector-based structure means that some sectors might be considered more important than others, not only due to their weight in the index but also because of the uneven distribution of companies across sectors, as shown in Table 5.2. Moreover, attempts to apply any kind of structure learning technique led to the incorrect assumption of conditional independence between sectors and the overall S&P 500 price changes. Since every sector — although in different amounts — contributes to the final index price, this led to incorrect assumptions.

### 7.4 Final Approach and Computational Improvements

The final approach followed is as defined in Chapter 4 Section 2.5 where one random variable represents the price change of the companies and two different nodes (per time transition) are used to define the company and sectors. This approach allows to continue modeling both the individual companies and the sectors they belong to, while also capturing the connections between them. By using separate nodes for “Company” and “Sector”, the model keeps the unique behavior of each company, while still taking advantage of the broader patterns seen in the sectors. By structuring the model this way, the complexity is reduced compared to the original approach of treating each company separately. Using sectors helps to include more information in each variable, making it easier to learn the connections in the network. Adding the company as an external factor solves the earlier issue of treating all companies the same, ensuring that larger companies, that have more impact on the market, are given the right importance. This approach also fixes the problem

with the sectors-only model, where all companies in a sector are treated equally, leading to poor predictions. Now, the model accounts for both sector-level influences and individual company differences, which makes it more accurate in predicting stock price changes.

## 7.5 Prediction Methods and Execution Time

During the development of the first approach where each company was represented as a separate random variable, a revision of how the parameters were constructed was found to be necessary. The original method presented in [11] was used as a starting point, however, this method was soon found to be computationally intensive not just for learning the structure or obtaining predictions, but for the construction of the CPTs themselves. For this reason, the approach was revised to speed up the computation. While the revisions did result in improvements, the process still required a significant amount of time for its execution. To further optimize it, the construction of the Conditional Probability Tables (CPTs) was parallelized which significantly reduced the execution time, transforming a process that was previously taking hours into one that now takes just a few minutes.

When working with the library `bnlearn` in R to make predictions using Bayesian Networks, there are several methods that can be used, each with its own strengths and trade-offs in terms of computational efficiency and accuracy. One of the challenges faced when predicting future values in these networks, particularly for complex or large-scale problems, is the time it takes to compute these predictions. A single prediction was initially taking several hours, which was impractical to execute all the tests needed. To address this issue, predictions were also run in parallel, which significantly reduced the computation time. The library `bnlearn` provides three main types of prediction methods: `parents`, `exact` and `bayes-lw`. Each of these method approaches the problem differently. The `parents` method is the simplest and fastest of the three, for this reason it was initially employed in the case where each node represented a company. This method uses the parent nodes of the node we would like to predict and uses the local probability distribution of the target node to compute the prediction. This means that it does not consider the broader network or any indirect relationship between the nodes, instead, it focuses solely on the direct dependencies and distribution of the data in its parent's node. The advantage of this approach is the speed, with the drawback of not being able to capture the full complexity of the network, nor taking advantage of all the historical data fed as observations for the computation of the prediction. The `exact` method, takes a different approach by using exact inference techniques. This involves constructing a junction tree and performing belief propagation, allowing the method to compute the maximum a posteriori estimates given all the observed evidence. Finally, the `bayes-lw` method uses a technique known as Likelihood Weighting [25], which is a form of Monte Carlo simulations [36], for approximate inference. The algorithm is used to approximate posterior probabilities and is particularly useful when no evidence is provided for certain nodes. In these cases, their values are sampled based on their conditional probability distributions. In the case of this research, observations are provided for each node except the one that needs to be predicted, as those observations represent all the data prior to the node we would like to predict. The prediction process becomes largely deterministic since there is no need for sampling for the observed nodes. Only the target node (the one we want to predict) requires sampling. Specifically, for each node with known evidence, we use the observed data to directly calculate the likelihood of that node's observed value, conditioned on the values of its parents (as defined by the Conditional Probability Tables (CPTs)). The likelihoods of the observed nodes are combined by multiplying them to get the total likelihood of the entire set of evidence. Since there

is no randomness for the observed nodes, the counts are updated directly using this likelihood. For the target node (which has no evidence since it represents the node we would like to predict), its value is randomly chosen based on the probabilities determined by its parent nodes' observed values. After sampling, the algorithm updates the counts for the target node's predicted values, using the likelihood of the evidence to weight the counts. This process is repeated  $N$  times (in this case,  $N = 500$ ). After each round of sampling, the count for the target node's sampled values is updated, weighted by the overall likelihood of the evidence. Finally, the posterior probability of the target node is computed by dividing the weighted count of the target node's sampled values by the total weighted count of the evidence. The final choice in the project went for this prediction method, which, although it can be computationally intensive based on the number of samples that are generated, for large networks, provides big improvements over the exact inference computed by the exact method.

## 7.6 Influence of Historical Trends on Predictions

After describing the ways the predictions were obtained, let's now focus on the results. The confusion matrices of the model (tables 6.5, 6.6, 6.7, 6.8) reveal that the predictions are generally correct, predicting the correct labels more frequently than they misclassify them. This observation indicates that the models achieve an accuracy that is higher than that of a random guess. When no structure learning technique was used, the models did a better job at predicting the correct datapoints. This is shown in the highlighted (red) cells where the labels predicted were often incorrect in the case of models making use of structure learning. This is the case for the actual labels being: **Decrease**, **Low Decrease**, **Low Increase**, **Increase**, and **High Increase**, which represents the majority of the cases. Figures 6.1 and 6.2 show the correlation between the metrics and amount of training data used for the prediction. The correlations were computed by grouping each instance based on the number of predictions made. It is evident that the number of predictions highly affects these metrics due to the inherent nature of the predicted data, where predicting more data points means predicting data points further in the past. The Pearson correlation indicates that the metrics generally worsen as the of predicted data points increases. This pattern is visible for most of the computed metrics, regardless of whether structure learning is applied or not. Specifically, weighted accuracy shows a strong negative correlation, while the MAE display a strong positive correlation. This suggests that as the historical data used for training increases, the weighted accuracy declines, whereas the error rate, measured by the MAE, rises. On the other hand, the Log-Likelihood improves as the number of predictions increases. This improvement can be attributed to the growing amount of historical data available for training, which enhances the model's understanding of older data distributions. When the number of predictions is low (e.g. 50) the Pearson coefficient does not display any correlation. However, as the number of predictions increases — which corresponds to predicting data points further in the past — the correlation becomes more clear. More historical data allows the predictor nodes (and their parent nodes) with better knowledge of past data distributions, which in substance, enables the model to more accurately explain this older periods. From these correlations, two main insights emerge: first, predicting past datapoints using nodes trained on more recent data reduces the prediction accuracy, likely due to a shifting market behaviour over time. The second aspect that can be inferred is that, increasing the amount of historical data can, at times, benefit the model, improving its ability to interpret unexpected scenarios.

Although a powerful indicator, the Pearson correlation coefficient assumes linearity

in the data. However, financial data, and the computed metrics, often violates this assumption displaying non-linear patterns and non-uniform distributions. Therefore, it is also necessary to interpret the results from a different perspective to account for multiple aspects.

In Figure 6.3, the log-likelihood was plotted to display how differently it behaves depending on the amount of historical data used. In both conditions — whether with or without structure learning — a distinction can be noticed where using more historical data results in a worse log-likelihood score. However, it is worth noticing that this distinction is highly attenuated when no structure learning technique is applied. When analyzing the same metric from a different perspective, evaluating how it changes over time, like in Figures 6.4 and 6.5, a different kind of behavior is noticeable. The log-likelihood seems to follow more of a wave-like pattern where increasing the amount of historical data consequently adds more information. Depending on the market behavior in the considered period, it can improve or worsen the predictions.

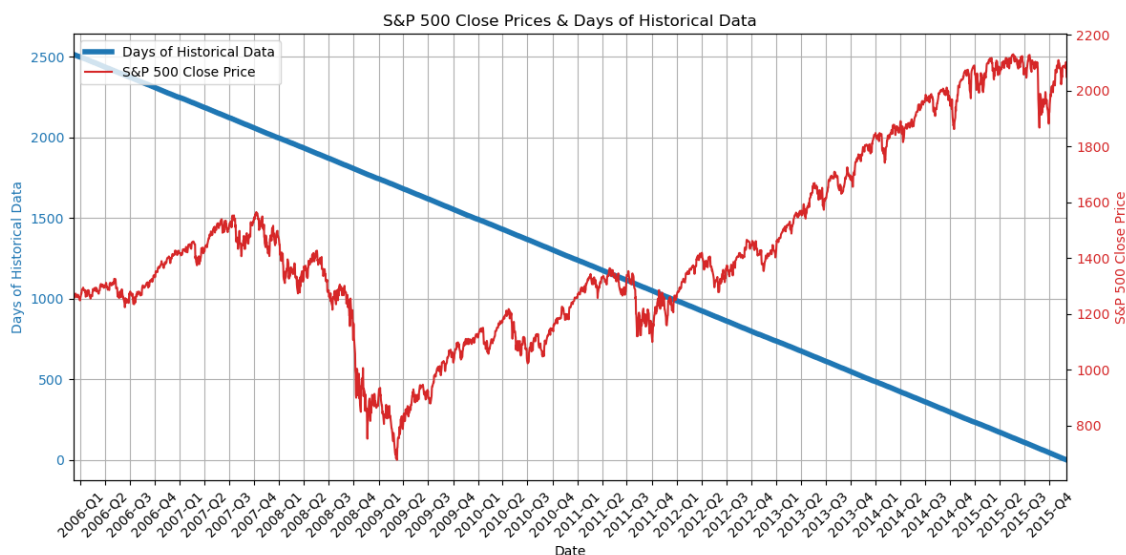


FIGURE 7.1: S&P 500 and number of predictions

To further understand how the market direction affects the predictions, we need to understand the behaviour in the period under consideration. Figure 7.1 presents two key pieces of information: the closing price of the S&P 500 (red line) over the 10 years from January 2006 to December 2015, and the number of days of historical data used in the model's training (blue line).

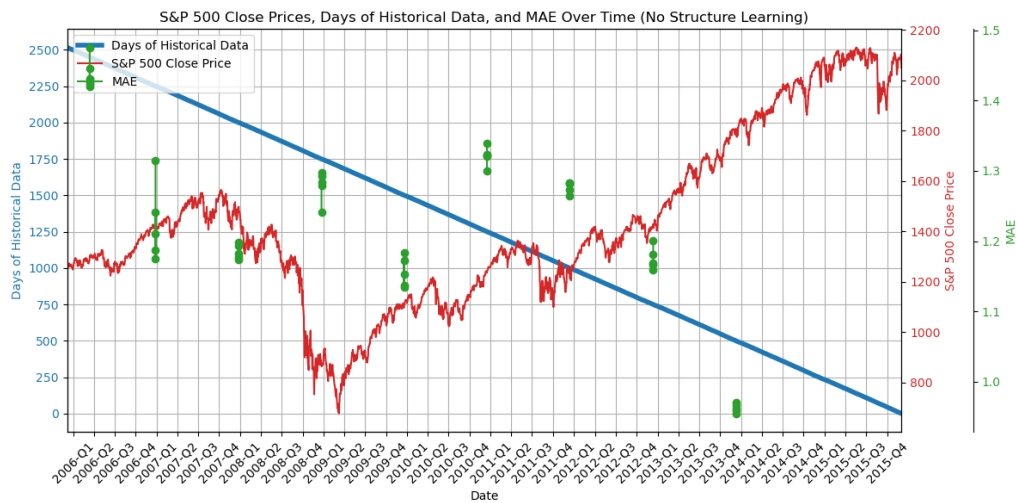
The red line displays the price changes in the S&P 500 index where the index daily closing price was used. From the plot, we can notice how relevant events, such as the 2008 financial crisis, are clearly visible. This financial crisis represented a negative period for the American economy resulting in a drop in the index price, which was followed by a period of recovery and growth reaching its peak towards the end of the dataset, showing the overall upward trend after 2012.

On the other hand, the blue line represents the accumulation of training data with a total of 2500 days of data available. As more and more training data is used, we move further in the past where data is added to the training set. For instance, we can see that the 1000th day of historical data in the dataset corresponds to Q4 (4th quarter) 2011. This representation helps us understand how trends in the S&P 500 index influenced the

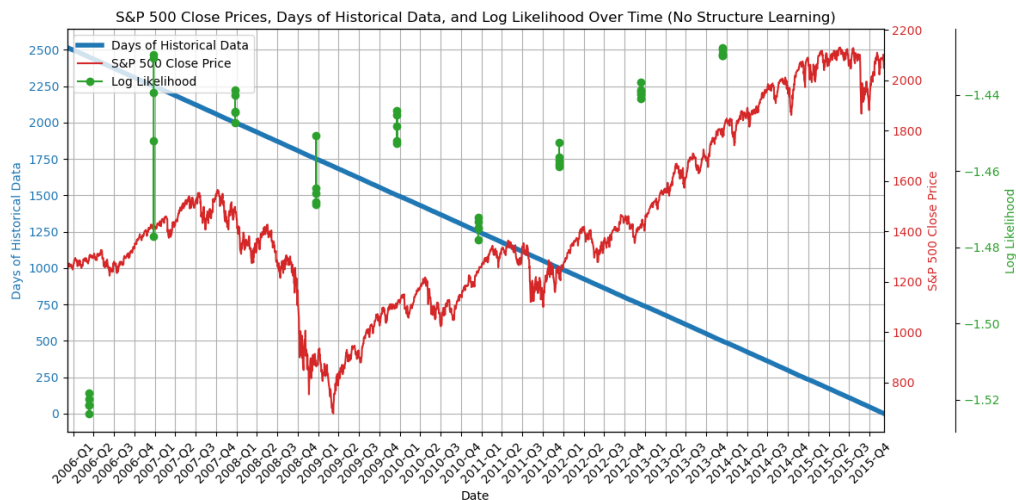
network predictions as more historical data were incorporated into the prediction. The direction of the amount of training data is opposite to the time direction. The reason why this line has a different trend is that increasing the amount of data used for training means increasing the amount of historical data while leaving the previously used data unchanged.

To understand how trends affected the predictions, the S&P 500 was overlapped to the previously computed Mean Absolute Error (MAE) and Log-Likelihood across all the tested models, shown in Figures 6.6 and 6.3.

Figures 7.2 and 7.3 display this comparison where no structure learning and structure learning were used, respectively. Both metrics (MAE and log-likelihood) were averaged over the cutset and the number of days of historical data used, minimizing unnecessary noise and leading to a mean that does not take into account the number of predictions computed per each model to focus more on the variance of the different cutsets.

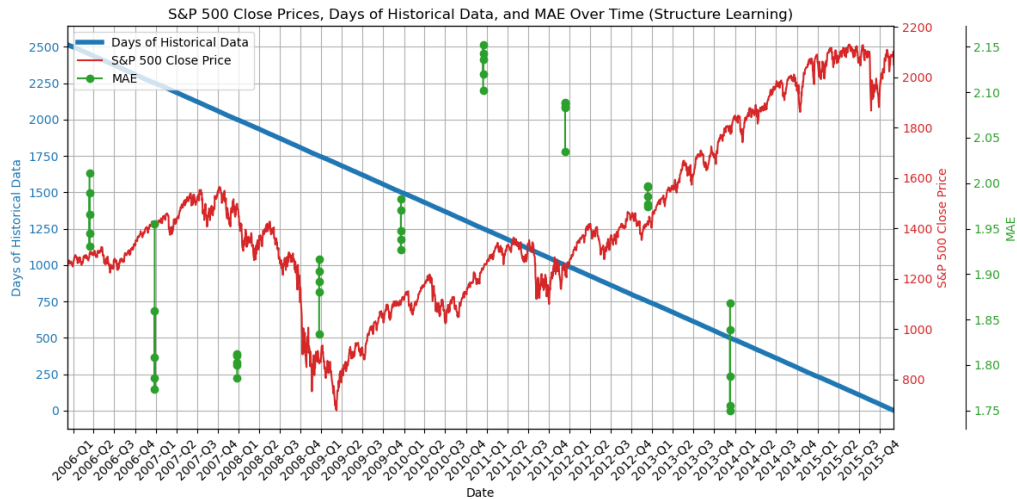


(A) S&P 500, Mean Absolute Error (MAE) over time (No Structure Learning Applied)

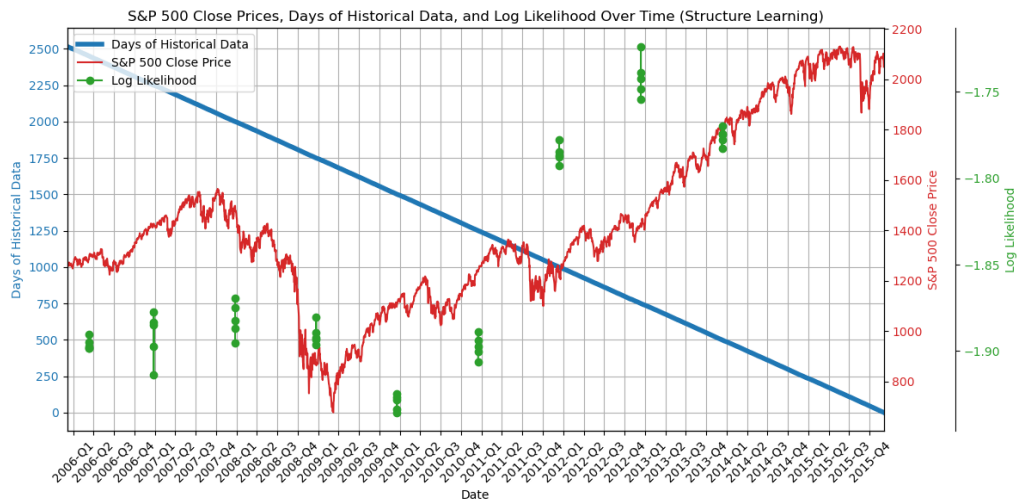


(B) S&P 500, Log-Likelihood over time (No Structure Learning Applied)

FIGURE 7.2: S&P 500 and Days of Historical Data compared to metrics (No Structure Learning Applied)



(A) S&P 500, Mean Absolute Error (MAE) over time (Structure Learning Applied)



(B) S&P 500, Log-Likelihood over time (Structure Learning Applied)

FIGURE 7.3: S&P 500 and Days of Historical Data compared to metrics (Structure Learning Applied)

This comparison is key to understanding how increasing the amount of training data impacts the predictions and model performance. Both metrics were placed at the point in time that represents the amount of historical data used for obtaining that prediction. It in fact represents the value of the metric computing a prediction where the amount of data used is equals to all the data from the last day of data available until the date where the metric result is placed. When looking at the period post 2011 until the end of the dataset (2016), an increase in the amount of training data used for the training (that means going further in the past) corresponded to a rise in the MAE and a decline in the log-likelihood, with a consequent decrease in the weighted accuracy (as visible in Figure 6.6). The period 2011-2016 is characterized by strong and continuous long-term growth. Given the mono-tonal trend, this suggests that increasing the amount of historical data likely led to overfitting, where the model has more likely seen upward movements compared to downwards.



The period going from late 2007 to 2011, saw a different kind of trend. This is the period hit by the financial crisis and the following recovery, which meant a total change in the market behaviour with steep declines and rises. The impact that implementing this period had on the predictions differed between the approaches followed. When no structure learning was applied, the MAE remained quite stable with a slight improvement when looking at the log-likelihood. On the other hand, when structure learning was applied, the MAE saw improvements with a log-likelihood that slightly worsened.

The behavior visible in these figures seems to also point towards a wave-like movement of the metrics. However, before reaching any conclusions, we must point out that this behaviour is not the direct consequence of adding the exact historical periods as mentioned above. While it is true that incorporating certain periods of data influences the predictions, the placement in time of the metrics on the plots, reflects the data used to train the full extent of the network without taking into account the Markov order employed during the actual predictions. The Markov order represents the depth in the dependencies of the three for the variables involved in the decision process of the network (e.g. a First order Markov dependency indicates that only the direct parents of a node are involved in the computation of the actual predictions). Although the data used to train the network is quite extensive and spans from the end of the graph (April 2015) to the day where the metrics are placed (represented by the x-axis), the effects are conditioned only on the most recent data due to the First-order Markov assumption used by the model. As previously explained in section 7.5, despite attempting to use all the available methods, the final one used for the results displayed in this research, involved the usage of Likelihood Weighting. This technique makes use of approximate inference where a first-order Markov assumption is employed.

In Figures 7.5 and 7.4, a more focused explanation of the data employed for training the nodes involved in the prediction process is displayed. In this case the focus went for the nodes used under the first-order Markov assumption, which involves the node we would like to predict and its direct parents.

The plot illustrates how the S&P 500 Close Price, displayed through the blue line, evolves over time. The period shown varies based on the amount of data used for training the network. Fluctuations in the S&P 500 prices help to understand how the model reacts to different market conditions and how training data contributes to those predictions.

The number of training days per time slice is one of the key focus, this number - represented by green dots scattered across the timeline - represents the amount of data used for training each node for the specific cutset defined on the graphs' title. Not only does this represent the amount of training data per node, but the green dot specifically represents the period of data that was used to train the predictor node. By tracing a line from the right end of the graph, which coincides approximately with December 2015, to the green dot, the S&P 500 price changes in this area represents the data used for its prediction. The reason why this was plotted is to help understand the amount of historical data the prediction relied on and the following log-likelihood derived from this prediction. For instance, as more training data is used, we expect that the model is better able to generalize, obtaining better predictions and improved log-likelihood, though this may not always translate into higher accuracy due to factors such as model complexity or market volatility.

Additionally, the red dots represent the log-likelihood of the model when the amount of data the node was trained with was equal to the quantity defined on the red axis on the top part of each plot. To better display what portion of the data was used to train this node, dotted lines connecting the green and red dots were drawn. The area between those

two points, represents the portion of the data the parent node was trained on. Numerical values were included to display how much data was used for training, not only the specific parent node but also each node in which the node was part of, offering a clear visual of how training data allocations vary across nodes as the amount data for training varies.

The log-likelihood, shown on the right-hand axis, measures how well the model fits the observed data. As the amount of training data increases, the log-likelihood typically improves, indicating that the model is becoming more confident in its predictions. In the case of this project, this correlation is not always visible.

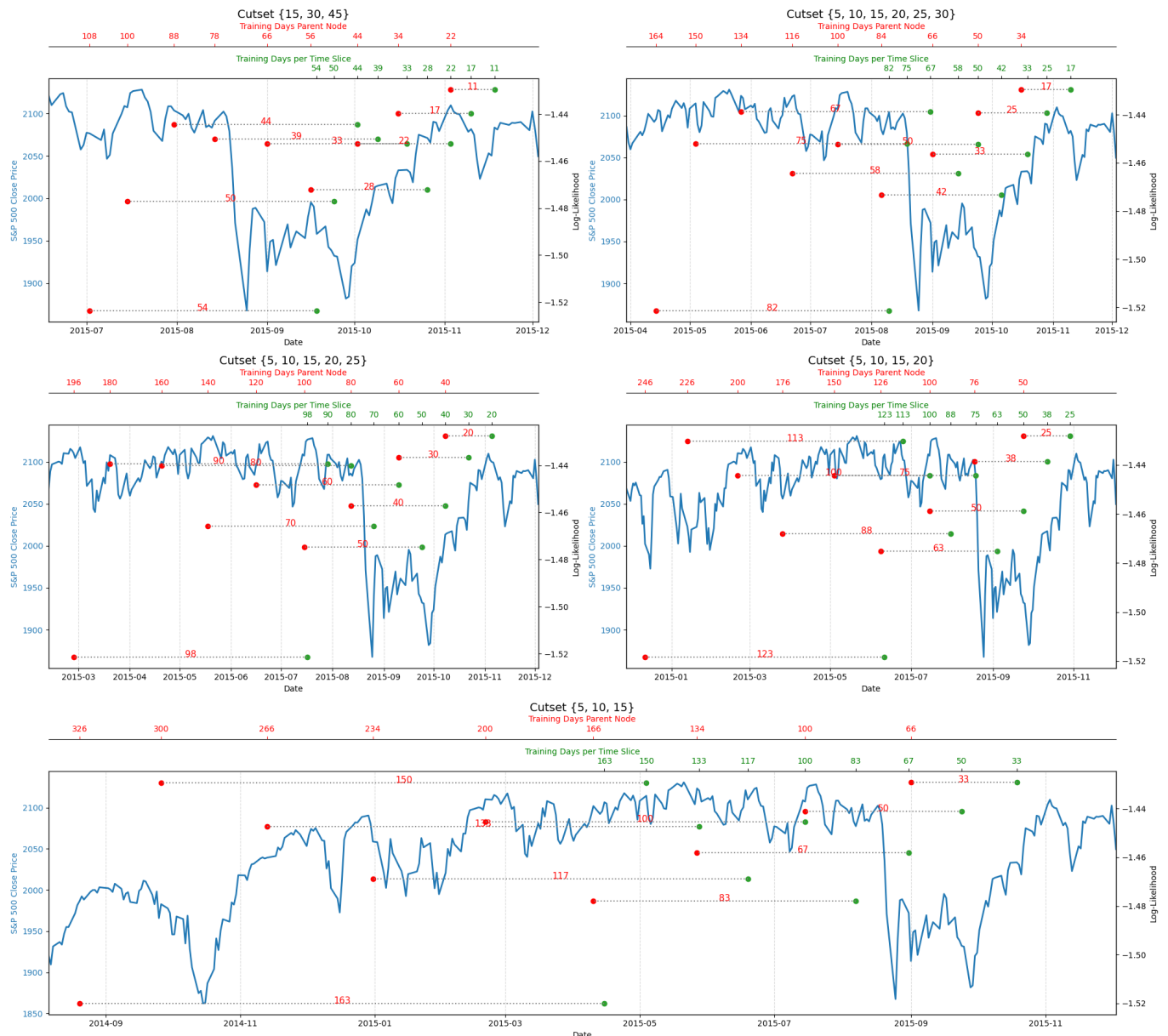


FIGURE 7.4: Likelihood per cutset in Relation to the Amount of Data Used to train each node (No Structure Learning Applied). The area between the green dots and the right end of the plot represents the data used for training the predictor node and the area between the red and green dots represents the data used for training the parent nodes.



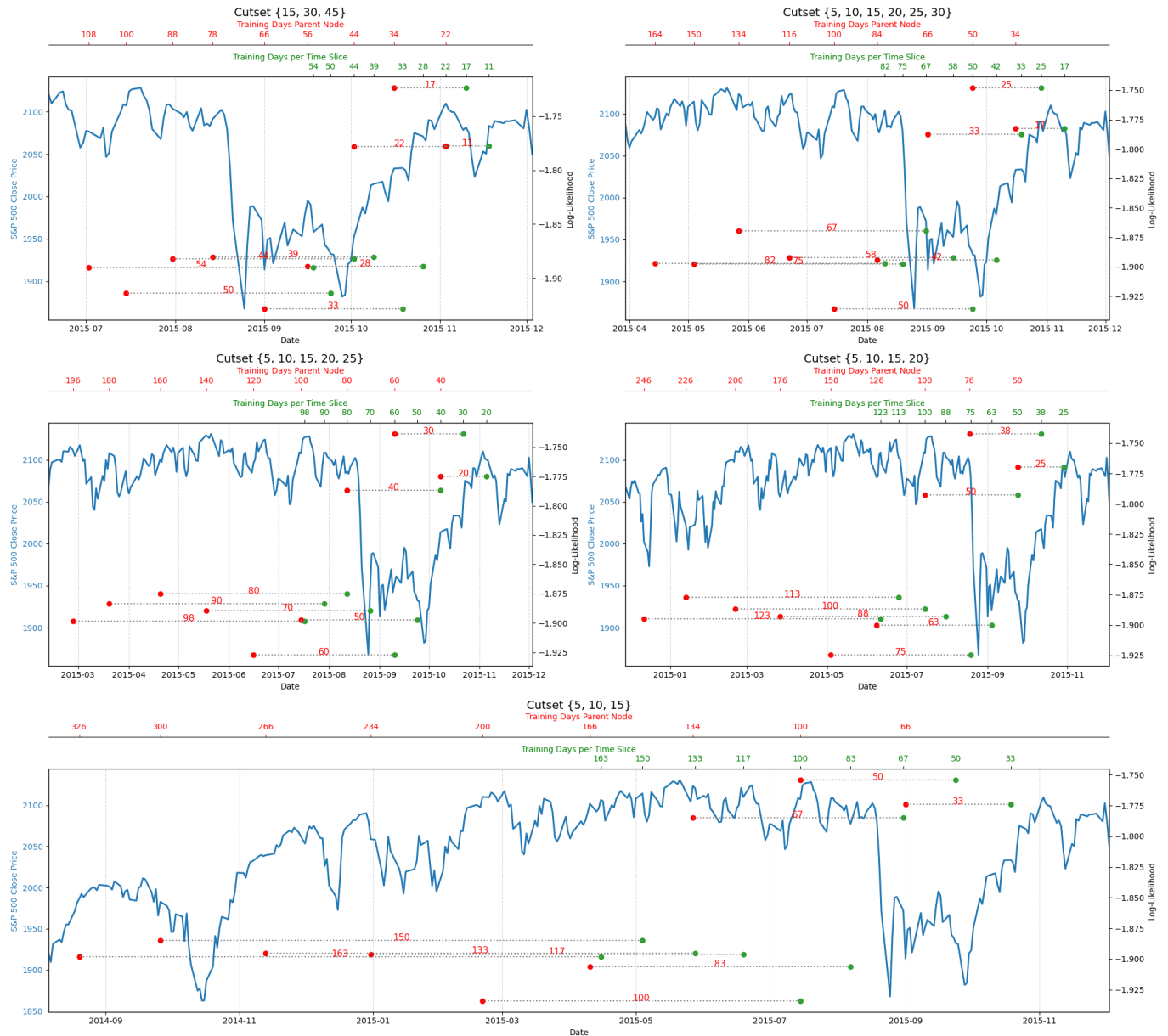


FIGURE 7.5: Likelihood per cutset in Relation to the Amount of Data Used to train each node (With Structure Learning Applied). The area between the green dot and the right end of the plot represents the data used for training the predictor node.

## Chapter 8

# Conclusions and Future Developments

In this section, we will summarise what was discovered during the development of the project. This thesis aimed to model price changes in the S&P 500 index by leveraging Partitioned Dynamic Bayesian Networks (PDBNs). The goal was not to obtain the best predictive model possible when employing PDBNs but to understand the relationship between the amount of historical data used and the prediction accuracy of a model.

Our journey through this complex task involved a series of challenges faced during the construction of the model that ultimately shaped its final version. In Chapter 5 we discussed how the data was retrieved and processed. Data handling was a big obstacle that was faced in multiple circumstances, both during the retrieval process and during the later transformations when constructing the network. This issue became recurrent as multiple approaches were tested during the evolution of the project, each of which involved different data processing steps.

In Chapter 7 Section 3 the different modeling approaches tested were discussed. The initial approach involved the creation of one node per company being part of the S&P 500 to evaluate their relationships and possibly decrease the number of connections through the effect of conditional independence given by structure learning techniques. We soon realised the unfeasibility of following this approach, which was also found to be limiting when concerning the creation of manual structures. The main reason for this approach being unsuccessful is the need for a huge amount of data that grows with the growth of the number of companies involved.

The second approach, which involved the construction of the network making use of sectors as the main source of truth, is not a suggested approach. With each sector having a different number of companies as part of it, it makes it highly unbalanced. Depending on the library or programming language used, this approach could also have additional limitations. Some libraries do not allow for a different amount of data to be used per node, which due to the unbalanced dataset, would be the case when following this approach. An additional consequence of this highly unbalanced dataset is that the network might try to predict the overall trend making use of one or two sectors among the ones that it detects as the most relevant ones. This leads to disregarding companies that might instead be more relevant in different periods and also disregarding individual companies' characteristics leading to a low prediction accuracy.

## 8.1 Final Model and Prediction Approach

To address the previously found limitations, as described in [Chapter 4 Section 2.5](#) and [Chapter 7 Section 4](#) changes to the model finally struck a balance between complexity and specificity by employing separate nodes for "Company" (representing the price change of each company) and "Sector" across time transitions. In this way, we were able to preserve the individual behavior of companies given by the trends available in the individual stock price changes while still capturing the broader patterns given by the sectors. This not only decreased the complexity compared to the company-specific approach but also resolved issues with overweighting specific sectors compared to the sectors-specific approach. Despite this approach having significantly improved various aspects of the model, the impact of adding the sector component on the predictions remains unclear and was not further investigated, particularly on how their probabilities shifted and the nature of those changes.

In [Chapter 7 Section 5](#) we discussed the various methods available and the one that was ultimately employed. The final approach involved the use of the Likelihood Weighting prediction method, which was chosen out of necessity due to the impossibility of computing the exact inference because of the computational complexity. While approximation methods often yield good results, they can, at times, be inaccurate. These methods offer multiple advantages, including greater memory efficiency and better scalability as the networks grow larger and more complex. On the other hand, approximation methods introduce a degree of error. In contrast to the exact inference which is deterministic, approximation methods generate different results depending on the sampled data. In the case of Likelihood Weighting, the convergence of those approaches to the prediction computed from the exact inference improves with an increased number of samples. However, with this improvement comes also an increase in computational complexity. Additionally, when dealing with discrete variables, like the labels involved in the development of this project, an increase in their number leads to less reliability in the approximation which would require a higher number of samples.

## 8.2 Results Analysis

Our analysis conducted in [Chapter 7 Section 6](#) revealed that models that made use of structure learning generally under-performed their counterparts. The confusion matrices in [Tables 6.5, 6.7, 6.6](#) and [6.8](#) confirmed this aspect, with the prediction accuracy indicating a higher rate of correct predictions and the Mean Absolute Error (MAE) in [Figure 6.6](#) indicating a lower error rate in the absence of structure learning.

This outcome suggests that the employed hill-climbing algorithm was not able to improve the structure. The hypothesis is that the variance in the data is too high, while the amount of data too low, for the algorithm to better understand the patterns. Moreover, the employed random variables, defined as: "Company", "Sector", "Company Price Change" and "S&P Price Change" are all relevant variables for the prediction of the S&P price, the removal of a single variable in this context will hardly achieve a similar accuracy as the accuracy deriving from the PDBN built without structure learning and would instead introduce additional biases and incorrect assumptions.

The effects of historical data on the predictions were varied. We observed different patterns where increasing the amount of historical training data had both positive and negative effects. In general, the predictions followed a wave-like pattern, where, based on the historical period included in the training data, the prediction improved or worsened. While more historical data enhanced the model's ability to explain older data distribu-

tions and consequently improved the generalization of the model, it often led to decreased weighted accuracy and increased mean absolute error (MAE) when predicting data points further in the past. This is reflected in the improved log-likelihood scores and in figure 6.3 where (mostly in the case of a model where structure learning is applied as in figure 6.3a), we can see that increasing the amount of data improves the accuracy of older predictions.

Our findings seem to indicate that the market behavior evolves over time, and models trained with a large amount of data, although might behave better in terms of generalization, might not correctly capture fast market movements during volatile periods which are less likely to happen. This underscores the importance of incorporating mechanisms within the model to account for shifting market conditions like Partitioned Dynamic Bayesian Networkss (PDBNs).

### 8.3 Limitations

During the development of this project, some compromises had to be made to be able to obtain results in a reasonable time frame. Two are the main limitations that we are now going to discuss.

#### Companies Weights

When computing the predictions through the library `bnlearn`, predictions were made by providing the observations row-wise, where each row represented a company, its sector, and the data observations up to the day before the prediction day target. With this kind of approach, for each company, one S&P 500 prediction was generated at each iteration. The predictions were then downsampled to produce a single prediction which was then compared to the actual value of the S&P 500 on the day we wanted to predict. The process of downsampling was executed by fitting a normal distribution to the sampled values, from which a single prediction value was extracted.

As explained in [Chapter 5 Section 1.1](#) the S&P 500 is a weighted index, meaning that its constituents have different weights, primarily based on their market capitalization when compared to the total capitalization of the companies composing the index. The weight of each company in the index changes daily depending on the growth that the company sustains. Due to the absence of weights data in the constructed dataset and the challenges in retrieving it, we made the incorrect assumption that each company had the same weight during the development process.

Additionally, the index consists of the largest companies listed in the United States stock exchanges. As companies grow in market capitalization, they become eligible to be included in this index, and vice versa, companies that shrink in size are removed and substituted by larger ones. As a consequence, companies are not permanently part of the index. Due to the challenges in retrieving data, the specific dates when companies were added or removed from the index were not considered. As a result, some companies in the dataset may no longer have been part of the index in the period analyzed, or they may not have been included yet. This limitation decreased the quality of the dataset, which included periods where companies were not part of the index at that specific time.

#### Look-ahead Bias

An additional limitation involved executing predictions for nodes that were trained with a set of data that included the exact data point we aimed to predict. This kind of approach introduces a look-ahead bias, where the model has access to future information that should

have otherwise been hidden at prediction time. The limited amount of data available for this project had left little room for the creation of separate training and test sets. Additionally, including different companies for executing the prediction or even using only a portion of the available companies would have defeated the purpose of this project since we aimed to analyze the companies part of the S&P 500 index. Ideally, the network would have been trained with data from the first available day (in the past) up to the day prior to the prediction day, and the prediction would have been executed for the following day. This approach would have avoided the network interaction with the target data point. However, it would have meant retraining the network as many times as the number of predictions computed (500 per each model that was tested). Following such an approach would have been prohibitive and it would have limited the number of predictions performed for each tested model.

## 8.4 Future Work

As described in section 8.3 above, one of the major limitations encountered was the absence of a fully detailed dataset. The consequence of this absence meant the use of incorrect assumptions to be able to proceed with the project. A key aspect for future development would be to incorporate the actual weights assigned to each company. After computing the prediction, each company's prediction would then be weighted based on the company's weight in the S&P 500, resulting in a more representative prediction. However, it is important to note that while this approach may better reflect the actual behavior of the index where each company is weighted accordingly, it might not necessarily lead to an improved model accuracy. One more step towards an improved dataset would be including the dates detailing when companies were included or removed from the index. By including those dates, we would be able to remove companies no longer part of the index, in each time slice, removing unnecessary noise and decreasing the computational complexity.

Additional future work may focus on enhancing how the predictor node is trained, trying to improve the look-ahead bias described in 8.3. The idea here is to design a cross-validation approach where the predictor node would be re-trained multiple times and at each time a different set of companies is used to train the node to predict the companies that were not included instead. By following this process multiple times, we would end up with a set of predictions that can then be averaged to obtain the actual prediction. This process would not require training the full network multiple times, in this way we can speed up the time required for the network construction. Instead of learning the full network, after a change in the predictor node, only the parameters and the structure of the last Dynamic Bayesian Network (DBN) would be reprocessed.

One additional approach that would be interesting to test to improve the predictions is the impact of higher-order Markov chains on the predictions. The method explained in Chapter 5 Section 2 made use of the Likelihood Weighting approach where first-order Markov chains are employed. In the case of the model built in this project, where the network was constructed without the use of structure learning. The first-order Markov chains meant that only the predictor node and the previous time slice were used for the prediction, disregarding the rest of the network and the historical data the network was trained with. Employing higher-order Markov chains might reveal more aspects of how the historical data employed helps in predicting further in the future.

Finally, as highlighted in Chapter 7 Section 6, all the employed variables are equally important and provide additional information at any given time. Since employing structure learning did not seem to improve the network's performance, adding more company-related

---

information that also evolves over time — such as the P/E ratio or the daily trading volume — could not only improve the prediction accuracy but also help structure learning algorithms better identify relationships between random variables. This approach could help maintaining only important variables, taking better advantage of the conditional independence property of Bayesian Networks (BNs).

# Appendix A

## Results of Executions with Structure Learning

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
50	500	15_30_45	1.822944	2.170368	-742.525470	-1.839098	0.396986	6	24
	750	15_30_45	2.122886	2.371014	-1387.353508	-1.769414	0.340076	10	9
	1000	15_30_45	2.056110	2.449696	-764.641740	-1.725850	0.376106	14	10
	1250	15_30_45	2.125300	2.386628	-886.667240	-1.875704	0.334508	11	12
	1500	15_30_45	1.744130	1.882248	-792.223452	-2.037184	0.322030	11	16
	1750	15_30_45	1.815004	2.069740	-914.974178	-1.860742	0.395570	16	9
	2000	15_30_45	1.816038	2.097036	-1050.260530	-1.843026	0.361940	8	15
	2250	15_30_45	1.816408	1.996834	-1118.406510	-1.907016	0.358240	10	15

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15	30_45	2.004998	2.217572	-979.323588	-1.997498	0.335020	11	12
500	5	10_15	1.657914	1.905992	-604.047936	-1.856908	0.424982	14	12
750	5	10_15	2.141112	2.304770	-1621.760446	-1.852698	0.309278	7	14
1000	5	10_15	2.117618	2.451478	-826.260396	-1.754486	0.364866	11	12
1250	5	10_15	2.342222	2.592902	-1462.219448	-1.951684	0.252938	6	9
1500	5	10_15	1.767382	1.913824	-843.705900	-2.019488	0.321104	13	13
1750	5	10_15	2.040032	2.239352	-837.721072	-1.920670	0.339736	11	12
2000	5	10_15	1.836946	2.124210	-623.544304	-1.905772	0.402696	14	13
2250	5	10_15	1.198826	1.384046	-599.400174	-1.813142	0.482152	14	20
2450	5	10_15	1.934560	2.131186	-1312.018522	-1.995176	0.303808	9	13
500	5	10_15_20	1.540204	1.799918	-661.506636	-1.834064	0.449684	11	20
750	5	10_15_20	2.049428	2.227982	-1776.217476	-1.807680	0.338996	15	7
1000	5	10_15_20	2.207142	2.531666	-749.918094	-1.775664	0.354998	8	9
1250	5	10_15_20	2.260236	2.532094	-1144.064168	-1.927228	0.285052	7	9
1500	5	10_15_20	1.673492	1.834638	-774.221960	-1.982922	0.354162	9	19
1750	5	10_15_20	2.057346	2.256170	-846.222344	-1.891716	0.330280	9	12
2000	5	10_15_20	1.835830	2.123620	-808.309480	-1.873094	0.415984	14	16
2250	5	10_15_20	1.091336	1.291444	-478.105242	-1.768806	0.523190	12	25

Continued on next page



TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	1.919360	2.137220	-1212.122646	-1.992282	0.340268	11	20
500	5	10_15_20_25	1.541294	1.807332	-648.234782	-1.843594	0.441604	15	15
750	5	10_15_20_25	2.052282	2.239774	-1630.088440	-1.798404	0.346720	12	10
1000	5	10_15_20_25	2.202186	2.549852	-786.823698	-1.768162	0.352174	9	14
1250	5	10_15_20_25	2.184062	2.445712	-1023.828400	-1.911424	0.309534	9	13
1500	5	10_15_20_25	1.728724	1.891688	-832.617136	-1.995868	0.348520	9	19
1750	5	10_15_20_25	1.905338	2.108318	-942.473136	-1.871688	0.355062	9	14
2000	5	10_15_20_25	1.829098	2.122762	-850.514814	-1.839272	0.410868	12	16
2250	5	10_15_20_25	1.199768	1.391444	-565.688226	-1.791234	0.494868	18	17
2450	5	10_15_20_25	2.009222	2.205388	-1219.434654	-2.001522	0.304262	7	16
500	5	10_15_20_25_30	1.687316	1.999842	-696.099248	-1.858162	0.420082	15	12
750	5	10_15_20_25_30	2.093218	2.280186	-1354.133550	-1.833300	0.327282	10	13
1000	5	10_15_20_25_30	2.169294	2.534122	-757.175222	-1.751928	0.371158	9	14
1250	5	10_15_20_25_30	2.119300	2.397120	-953.664922	-1.906446	0.317282	7	14
1500	5	10_15_20_25_30	1.786574	1.945680	-863.654876	-2.026278	0.332868	13	15
1750	5	10_15_20_25_30	1.959232	2.158250	-979.687578	-1.896294	0.336492	8	18
2000	5	10_15_20_25_30	1.725972	2.044548	-874.883238	-1.811528	0.413136	12	16
2250	5	10_15_20_25_30	1.425602	1.601536	-656.212816	-1.864280	0.430952	9	22

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
100	2450	5_10_15_20_25_30	2.000876	2.193760	-1143.002820	-2.004746	0.311054	14	9
	500	15_30_45	1.813872	2.171767	-808.991720	-1.756627	0.404706	15	37
	750	15_30_45	2.043223	2.317726	-1398.515161	-1.693932	0.366347	18	21
	1000	15_30_45	1.992082	2.337531	-733.661499	-1.771315	0.372037	21	26
	1250	15_30_45	2.179765	2.454559	-1032.867545	-1.888301	0.325169	19	26
	1500	15_30_45	1.967934	2.180081	-841.954529	-1.995978	0.301770	18	27
	1750	15_30_45	1.876965	2.117795	-743.051692	-1.925928	0.370956	25	26
	2000	15_30_45	1.944879	2.188178	-1166.592713	-1.891268	0.342588	16	30
	2250	15_30_45	2.049178	2.241023	-1022.227951	-1.949530	0.292229	18	29
	2450	15_30_45	2.031710	2.264051	-997.047134	-1.942241	0.348973	24	22
	500	5_10_15	1.734696	1.986135	-682.411676	-1.795430	0.409190	24	29
	750	5_10_15	2.098875	2.266019	-1628.145382	-1.779008	0.328172	18	25
	1000	5_10_15	2.170721	2.484319	-807.872893	-1.775323	0.334941	21	22
	1250	5_10_15	2.267605	2.534192	-1153.827512	-1.912390	0.289320	16	18
	1500	5_10_15	1.749363	1.938444	-773.619060	-1.986182	0.355780	26	29
	1750	5_10_15	1.994111	2.216941	-760.371987	-1.921248	0.351686	23	26
	2000	5_10_15	1.834360	2.107037	-992.953000	-1.884309	0.377657	26	23
	2250	5_10_15	1.658070	1.843850	-906.816647	-1.869694	0.384670	25	32

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.905049	2.132651	-1079.118447	-1.935745	0.339994	21	29
500	5	10_15_20	1.693693	1.956853	-710.613740	-1.786006	0.418994	23	30
750	5	10_15_20	2.022651	2.207226	-1657.265582	-1.753588	0.351864	27	14
1000	5	10_15_20	2.161405	2.484775	-739.051168	-1.782087	0.332647	13	20
1250	5	10_15_20	2.218106	2.495462	-988.921026	-1.900147	0.306115	15	20
1500	5	10_15_20	1.743170	1.945161	-820.204044	-1.976349	0.361207	21	36
1750	5	10_15_20	1.999580	2.234355	-788.300402	-1.911132	0.351389	22	23
2000	5	10_15_20	1.859644	2.128372	-990.398385	-1.885741	0.370019	20	30
2250	5	10_15_20	1.630815	1.827168	-825.255080	-1.859259	0.400958	20	42
2450	5	10_15_20	1.858890	2.091776	-1035.040890	-1.928937	0.360153	22	30
500	5	10_15_20_25	1.697794	1.973203	-695.307453	-1.783325	0.421675	21	36
750	5	10_15_20_25	2.048654	2.253947	-1577.113129	-1.746372	0.351402	21	22
1000	5	10_15_20_25	2.140875	2.467889	-707.966974	-1.779311	0.335854	22	21
1250	5	10_15_20_25	2.250521	2.523129	-967.342700	-1.887970	0.307310	15	23
1500	5	10_15_20_25	1.818153	2.023275	-880.969081	-1.976755	0.346173	20	32
1750	5	10_15_20_25	1.970171	2.204540	-813.577082	-1.922140	0.349835	21	24
2000	5	10_15_20_25	1.873202	2.147799	-1081.042793	-1.865115	0.368118	16	34
2250	5	10_15_20_25	1.726192	1.918982	-913.589906	-1.883266	0.375509	23	29

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	5_10_15_20_25	1.930450	2.159158	-1033.253762	-1.943732	0.346316	18	26
	500	5_10_15_20_25_30	1.844480	2.160820	-649.658494	-1.805126	0.396264	27	27
	750	5_10_15_20_25_30	2.045886	2.258047	-1407.337080	-1.761069	0.349326	27	20
	1000	5_10_15_20_25_30	2.129512	2.471361	-692.745624	-1.796427	0.345605	18	25
	1250	5_10_15_20_25_30	2.181041	2.467552	-936.708354	-1.872929	0.328573	15	26
	1500	5_10_15_20_25_30	1.882549	2.087309	-896.884910	-2.015181	0.327299	22	31
	1750	5_10_15_20_25_30	1.980321	2.209425	-804.592845	-1.930392	0.348623	20	31
	2000	5_10_15_20_25_30	1.910317	2.184096	-1174.038092	-1.864728	0.354075	17	32
	2250	5_10_15_20_25_30	1.834128	2.022704	-1012.828147	-1.902689	0.346481	16	34
	2450	5_10_15_20_25_30	2.006007	2.234708	-1009.783576	-1.951881	0.333609	24	20
150	500	15_30_45	1.885916	2.246567	-780.568171	-1.777147	0.387438	22	49
	750	15_30_45	2.038859	2.308967	-1310.986791	-1.697615	0.356048	27	33
	1000	15_30_45	1.953991	2.297909	-896.306035	-1.732797	0.385714	33	39
	1250	15_30_45	2.163469	2.463009	-973.351165	-1.883824	0.325415	27	37
	1500	15_30_45	2.014361	2.246337	-1030.856621	-1.923241	0.308160	27	36
	1750	15_30_45	1.875249	2.116945	-768.229388	-1.893760	0.384903	37	37
	2000	15_30_45	1.813597	2.046899	-1115.158505	-1.883647	0.354661	30	43
	2250	15_30_45	2.026377	2.248803	-940.446858	-1.921908	0.313319	28	37

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15	30_45	1.922707	2.159245	-988.105563	-1.921814	0.360833	36	35
500	5	10_15	1.725023	1.981575	-727.331791	-1.733247	0.423403	36	42
750	5	10_15	2.018353	2.188913	-1465.772890	-1.740067	0.346874	31	40
1000	5	10_15	2.067820	2.375679	-855.806669	-1.774165	0.354031	34	36
1250	5	10_15	2.220329	2.504281	-1073.488645	-1.903445	0.306898	23	29
1500	5	10_15	1.949967	2.166844	-923.873949	-1.955598	0.313519	33	36
1750	5	10_15	1.877564	2.097030	-763.830845	-1.933465	0.361097	31	42
2000	5	10_15	1.905167	2.151588	-1074.210677	-1.913425	0.356301	34	37
2250	5	10_15	1.842742	2.036594	-901.873012	-1.900053	0.347111	35	43
2450	5	10_15	1.889363	2.109189	-992.230071	-1.931307	0.350864	36	47
500	5	10_15_20	1.717706	1.981809	-740.773067	-1.731781	0.422673	34	40
750	5	10_15_20	2.023231	2.204051	-1507.695667	-1.733718	0.351658	38	28
1000	5	10_15_20	2.055065	2.375705	-867.770159	-1.783315	0.353969	26	29
1250	5	10_15_20	2.208045	2.504427	-980.444766	-1.899532	0.313082	22	30
1500	5	10_15_20	1.947527	2.174291	-942.889265	-1.944216	0.319703	27	42
1750	5	10_15_20	1.891161	2.115159	-771.818921	-1.924135	0.360265	31	42
2000	5	10_15_20	1.890569	2.139365	-1086.342911	-1.897878	0.361165	28	48
2250	5	10_15_20	1.853575	2.057542	-864.413690	-1.889715	0.353033	26	50

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	1.873522	2.097149	-971.910502	-1.934460	0.363226	30	45
500	5	10_15_20_25	1.752611	2.023427	-736.758799	-1.742299	0.415855	32	47
750	5	10_15_20_25	2.018072	2.229297	-1399.276867	-1.720867	0.357183	33	32
1000	5	10_15_20_25	2.044529	2.378733	-868.377441	-1.768632	0.361157	35	36
1250	5	10_15_20_25	2.201881	2.503534	-953.075788	-1.892997	0.316811	27	36
1500	5	10_15_20_25	1.966665	2.203211	-932.580834	-1.938801	0.314221	24	42
1750	5	10_15_20_25	1.881584	2.105413	-793.263833	-1.920245	0.369730	36	39
2000	5	10_15_20_25	1.863889	2.116263	-1078.749045	-1.883227	0.364778	28	48
2250	5	10_15_20_25	1.905078	2.113787	-869.544869	-1.904141	0.344545	31	39
2450	5	10_15_20_25	1.924330	2.137221	-1082.127853	-1.936783	0.345243	27	40
500	5	10_15_20_25_30	1.866331	2.190147	-784.665285	-1.762198	0.395791	36	46
750	5	10_15_20_25_30	2.046521	2.260791	-1341.434845	-1.730878	0.346687	43	26
1000	5	10_15_20_25_30	2.033211	2.376179	-855.206561	-1.762273	0.370039	27	35
1250	5	10_15_20_25_30	2.192803	2.501715	-969.529566	-1.886919	0.318339	22	35
1500	5	10_15_20_25_30	1.975143	2.215461	-1006.469265	-1.938073	0.317125	28	41
1750	5	10_15_20_25_30	1.891991	2.113675	-800.798358	-1.914471	0.371751	33	45
2000	5	10_15_20_25_30	1.849093	2.101248	-1080.755052	-1.876931	0.362885	32	46
2250	5	10_15_20_25_30	1.959441	2.168561	-912.736607	-1.913276	0.330747	21	47

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
200	2450	5_10_15_20_25_30	1.926991	2.145696	-1059.146208	-1.933475	0.350085	36	36
	500	15_30_45	1.875333	2.227281	-842.596749	-1.789594	0.376064	31	62
	750	15_30_45	1.962391	2.241351	-1406.156118	-1.703279	0.379581	41	48
	1000	15_30_45	1.988430	2.316216	-1049.746054	-1.741073	0.378620	38	54
	1250	15_30_45	2.159037	2.455942	-1073.587268	-1.854777	0.332248	36	50
	1500	15_30_45	2.047675	2.274548	-1130.435675	-1.914581	0.314813	35	49
	1750	15_30_45	1.814792	2.064289	-810.483884	-1.867456	0.400460	55	50
	2000	15_30_45	1.808478	2.028332	-972.583196	-1.897148	0.349464	42	58
	2250	15_30_45	1.935731	2.164808	-1020.429346	-1.919219	0.319768	42	49
	2450	15_30_45	1.940126	2.166921	-1172.873572	-1.883773	0.363034	45	51
	500	5_10_15	1.763564	2.015452	-773.926380	-1.757557	0.408517	50	54
	750	5_10_15	2.024352	2.193109	-1506.704522	-1.739093	0.349374	42	49
	1000	5_10_15	2.003900	2.307477	-984.588736	-1.761363	0.368447	41	49
	1250	5_10_15	2.162395	2.456282	-1073.692785	-1.879290	0.316771	34	36
	1500	5_10_15	1.993927	2.221940	-1000.487340	-1.932432	0.314704	43	50
	1750	5_10_15	1.932688	2.157650	-808.880675	-1.895498	0.372672	43	54
	2000	5_10_15	1.801913	2.037595	-990.343839	-1.897724	0.365729	43	58
	2250	5_10_15	1.864524	2.081205	-965.415733	-1.893972	0.344718	44	55

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	5_10_15	1.855857	2.078012	-1077.422780	-1.909191	0.363392	49	62
	500	5_10_15_20	1.751243	2.010255	-775.850119	-1.767275	0.405805	43	54
	750	5_10_15_20	1.973925	2.163205	-1495.808335	-1.719390	0.367087	51	39
	1000	5_10_15_20	1.996012	2.312425	-995.541814	-1.760847	0.370558	37	40
	1250	5_10_15_20	2.165647	2.466537	-1043.078782	-1.877611	0.322701	33	44
	1500	5_10_15_20	1.991518	2.223546	-998.748936	-1.922775	0.319336	37	52
	1750	5_10_15_20	1.952834	2.179531	-842.415514	-1.897980	0.368185	46	51
	2000	5_10_15_20	1.802918	2.038469	-982.164878	-1.894020	0.369210	43	65
	2250	5_10_15_20	1.872631	2.088623	-952.077854	-1.893629	0.342874	38	58
	2450	5_10_15_20	1.853245	2.077122	-1058.846829	-1.903284	0.370570	47	55
	500	5_10_15_20_25	1.770668	2.037230	-786.409453	-1.780277	0.399585	38	66
	750	5_10_15_20_25	1.974058	2.192261	-1435.634047	-1.719349	0.367794	49	38
	1000	5_10_15_20_25	2.019229	2.343213	-1046.414352	-1.753286	0.368073	48	47
	1250	5_10_15_20_25	2.195931	2.493950	-1077.024280	-1.877545	0.317868	33	48
	1500	5_10_15_20_25	2.007830	2.246162	-1013.603609	-1.927172	0.315729	32	57
	1750	5_10_15_20_25	1.905952	2.139282	-834.823276	-1.886883	0.378670	50	50
	2000	5_10_15_20_25	1.784386	2.023105	-959.320319	-1.885829	0.370860	37	70
	2250	5_10_15_20_25	1.871787	2.099185	-960.291224	-1.891696	0.345770	43	57

Continued on next page



TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
250	2450	5_10_15_20_25	1.883416	2.101489	-1142.575779	-1.902662	0.363339	39	55
	500	5_10_15_20_25_30	1.855035	2.171438	-779.773931	-1.788241	0.386373	51	54
	750	5_10_15_20_25_30	1.955402	2.184492	-1349.730694	-1.722290	0.371308	58	37
	1000	5_10_15_20_25_30	2.038809	2.364537	-1040.442404	-1.760582	0.367970	34	55
	1250	5_10_15_20_25_30	2.188812	2.488755	-1071.000634	-1.873727	0.319504	31	44
	1500	5_10_15_20_25_30	2.039696	2.282606	-1039.880238	-1.928257	0.313809	39	49
	1750	5_10_15_20_25_30	1.890897	2.125158	-827.603119	-1.884402	0.382953	47	54
	2000	5_10_15_20_25_30	1.779118	2.017275	-972.370507	-1.879581	0.364742	40	63
	2250	5_10_15_20_25_30	1.886276	2.112913	-975.703227	-1.901310	0.339473	36	57
	2450	5_10_15_20_25_30	1.909521	2.133267	-1157.445481	-1.895411	0.362791	50	48
	500	15_30_45	1.882597	2.229282	-969.089532	-1.796684	0.371205	43	74
	750	15_30_45	1.976690	2.250446	-1298.529212	-1.721486	0.372263	49	60
	1000	15_30_45	2.052542	2.376597	-1116.747211	-1.770769	0.357426	50	63
	1250	15_30_45	2.089215	2.371145	-1051.556524	-1.882870	0.337862	49	64
	1500	15_30_45	2.066772	2.288996	-1078.974254	-1.909732	0.312782	42	67
	1750	15_30_45	1.842148	2.098712	-780.226742	-1.867338	0.401683	69	64
	2000	15_30_45	1.693139	1.910482	-897.215365	-1.869940	0.372810	53	81
	2250	15_30_45	1.948926	2.175899	-1009.838457	-1.930839	0.323206	53	65

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15_30_45		1.950388	2.173792	-1148.545441	-1.868978	0.362289	56	61
500	5_10_15		1.793374	2.037587	-816.289918	-1.776189	0.393754	61	66
750	5_10_15		1.931434	2.109355	-1419.896905	-1.732947	0.372240	54	66
1000	5_10_15		2.030112	2.330527	-1015.979470	-1.759163	0.363643	55	54
1250	5_10_15		2.168970	2.446933	-1160.910418	-1.886142	0.319151	43	46
1500	5_10_15		2.010079	2.233847	-1107.865484	-1.911625	0.322152	52	62
1750	5_10_15		1.881572	2.121188	-792.011022	-1.870471	0.384655	57	66
2000	5_10_15		1.792132	2.015704	-901.431558	-1.894985	0.361304	54	72
2250	5_10_15		1.875378	2.088328	-1029.440771	-1.910067	0.337624	51	70
2450	5_10_15		1.894691	2.115909	-1168.923665	-1.884617	0.362515	58	77
500	5_10_15_20		1.784860	2.032242	-827.016276	-1.781980	0.389056	51	70
750	5_10_15_20		1.930750	2.121356	-1450.369208	-1.723312	0.375033	63	52
1000	5_10_15_20		2.039768	2.346997	-1063.400176	-1.771830	0.359256	45	49
1250	5_10_15_20		2.163174	2.449496	-1113.962771	-1.887172	0.321393	45	51
1500	5_10_15_20		2.003860	2.229156	-1100.988395	-1.907689	0.325697	47	64
1750	5_10_15_20		1.899482	2.138394	-788.554030	-1.877752	0.380926	61	59
2000	5_10_15_20		1.787798	2.009858	-928.169999	-1.891005	0.358848	54	84
2250	5_10_15_20		1.879346	2.098137	-998.064552	-1.908652	0.343210	49	67

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	1.872541	2.092044	-1149.764053	-1.878885	0.369989	58	63
500	5	10_15_20_25	1.795527	2.050984	-859.372555	-1.786222	0.383774	45	76
750	5	10_15_20_25	1.930219	2.148315	-1381.261820	-1.724346	0.377977	61	53
1000	5	10_15_20_25	2.035922	2.359087	-1083.864483	-1.767317	0.360720	56	60
1250	5	10_15_20_25	2.159845	2.445232	-1103.294794	-1.886002	0.323894	45	61
1500	5	10_15_20_25	2.029122	2.259954	-1088.911905	-1.909642	0.319180	38	74
1750	5	10_15_20_25	1.874526	2.118899	-794.205789	-1.879716	0.385802	63	59
2000	5	10_15_20_25	1.758246	1.983249	-903.464542	-1.878437	0.365611	46	89
2250	5	10_15_20_25	1.893986	2.116078	-1005.122643	-1.908499	0.339433	53	70
2450	5	10_15_20_25	1.896994	2.115645	-1144.175806	-1.880738	0.363433	49	64
500	5	10_15_20_25_30	1.842960	2.149036	-897.507633	-1.784065	0.381863	61	68
750	5	10_15_20_25_30	1.941019	2.164498	-1323.109140	-1.731995	0.372570	68	50
1000	5	10_15_20_25_30	2.073759	2.399230	-1096.502620	-1.769922	0.357501	43	68
1250	5	10_15_20_25_30	2.130338	2.416623	-1093.898490	-1.882041	0.327940	42	58
1500	5	10_15_20_25_30	2.038011	2.274369	-1088.367167	-1.912510	0.319900	51	62
1750	5	10_15_20_25_30	1.898140	2.139208	-803.604576	-1.883701	0.385286	60	66
2000	5	10_15_20_25_30	1.722434	1.948902	-892.503579	-1.868190	0.371162	55	80
2250	5	10_15_20_25_30	1.905337	2.125196	-1002.881525	-1.916100	0.335034	45	74

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
300	2450	5_10_15_20_25_30	1.914274	2.139438	-1134.130236	-1.876212	0.362504	62	59
	500	15_30_45	1.886979	2.228346	-1018.161494	-1.786952	0.369735	56	82
	750	15_30_45	1.982767	2.264879	-1196.365063	-1.720605	0.373488	60	71
	1000	15_30_45	2.039774	2.357621	-1080.766308	-1.787820	0.357944	65	74
	1250	15_30_45	2.026025	2.294072	-1009.735608	-1.906979	0.336211	56	84
	1500	15_30_45	2.023747	2.255936	-1053.881476	-1.901375	0.328156	53	83
	1750	15_30_45	1.832964	2.095054	-823.170561	-1.862693	0.396453	79	81
	2000	15_30_45	1.712430	1.922729	-929.710143	-1.874320	0.370588	64	98
	2250	15_30_45	1.968121	2.200769	-1008.742662	-1.922261	0.329970	61	80
	2450	15_30_45	2.011666	2.236541	-1147.351039	-1.863704	0.345906	62	68
	500	5_10_15	1.836358	2.078547	-931.974070	-1.786834	0.383928	74	76
	750	5_10_15	1.952868	2.130818	-1309.518667	-1.730996	0.370563	60	83
	1000	5_10_15	2.082823	2.376045	-1102.681922	-1.790085	0.346047	61	64
	1250	5_10_15	2.083594	2.343161	-1094.766611	-1.903614	0.325913	56	61
	1500	5_10_15	2.019730	2.243856	-1069.418973	-1.910120	0.323695	60	77
	1750	5_10_15	1.875605	2.124752	-762.391151	-1.875371	0.387834	69	81
	2000	5_10_15	1.693947	1.911233	-852.508727	-1.878489	0.381123	71	90
	2250	5_10_15	1.876121	2.098250	-997.055853	-1.902910	0.343872	64	79

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.939811	2.160561	-1114.739671	-1.871601	0.352597	67	88
500	5	10_15_20	1.792262	2.041984	-939.644630	-1.778850	0.389856	61	79
750	5	10_15_20	1.967136	2.156406	-1321.398571	-1.726303	0.371960	76	64
1000	5	10_15_20	2.068678	2.370772	-1086.013391	-1.793777	0.348209	56	55
1250	5	10_15_20	2.074095	2.342621	-1048.245898	-1.899920	0.329208	55	72
1500	5	10_15_20	2.017177	2.243619	-1062.735134	-1.901045	0.325351	58	75
1750	5	10_15_20	1.890714	2.139626	-790.471787	-1.875977	0.386561	71	80
2000	5	10_15_20	1.673707	1.891929	-854.610599	-1.867825	0.385373	69	106
2250	5	10_15_20	1.866228	2.092237	-971.526504	-1.900587	0.349078	64	77
2450	5	10_15_20	1.917554	2.138029	-1123.427722	-1.864753	0.361758	68	77
500	5	10_15_20_25	1.792704	2.051773	-934.407060	-1.782540	0.388732	58	90
750	5	10_15_20_25	1.965518	2.184404	-1265.317921	-1.726653	0.374104	72	67
1000	5	10_15_20_25	2.053507	2.366070	-1076.750138	-1.788291	0.353383	68	70
1250	5	10_15_20_25	2.087987	2.358815	-1058.237081	-1.903556	0.328377	59	80
1500	5	10_15_20_25	2.014889	2.246153	-1071.799601	-1.901768	0.324006	49	89
1750	5	10_15_20_25	1.872525	2.125938	-804.609276	-1.872631	0.388552	72	72
2000	5	10_15_20_25	1.666816	1.886312	-847.933477	-1.860136	0.386289	58	112
2250	5	10_15_20_25	1.890866	2.121598	-977.001031	-1.902947	0.346911	66	84

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	5_10_15_20_25	1.945202	2.162725	-1130.254098	-1.868917	0.353282	58	77
	500	5_10_15_20_25_30	1.851594	2.154472	-943.832153	-1.788708	0.379885	69	87
	750	5_10_15_20_25_30	1.959882	2.188569	-1213.333180	-1.728347	0.375052	81	60
	1000	5_10_15_20_25_30	2.067964	2.386608	-1072.531459	-1.789250	0.352316	56	74
	1250	5_10_15_20_25_30	2.068413	2.341394	-1052.415147	-1.902863	0.330777	55	77
	1500	5_10_15_20_25_30	2.030380	2.264462	-1078.126211	-1.908216	0.321380	59	79
	1750	5_10_15_20_25_30	1.872316	2.126149	-815.552654	-1.871332	0.388782	72	78
	2000	5_10_15_20_25_30	1.672790	1.890793	-852.302471	-1.864112	0.381084	70	97
	2250	5_10_15_20_25_30	1.929486	2.157760	-981.054397	-1.913048	0.337031	56	85
	2450	5_10_15_20_25_30	1.960725	2.184772	-1131.410451	-1.866227	0.351392	70	66
350	500	15_30_45	1.879429	2.225628	-1038.956254	-1.762360	0.375714	69	91
	750	15_30_45	1.966486	2.247151	-1123.684974	-1.736639	0.373493	67	88
	1000	15_30_45	2.069179	2.380842	-1095.989519	-1.804665	0.350735	72	86
	1250	15_30_45	2.057825	2.335989	-1005.571338	-1.915164	0.329144	64	95
	1500	15_30_45	2.011817	2.246046	-985.852079	-1.915092	0.331511	64	93
	1750	15_30_45	1.861045	2.117257	-880.630503	-1.879009	0.386303	85	96
	2000	15_30_45	1.791455	2.002959	-929.913523	-1.890779	0.351824	71	107
	2250	15_30_45	1.931639	2.165542	-1001.724320	-1.915303	0.337786	76	93

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	15_30_45	2.060747	2.283012	-1242.715832	-1.860449	0.335059	69	75
	500	5_10_15	1.847271	2.086813	-1001.812291	-1.770223	0.379663	86	86
	750	5_10_15	1.980358	2.157534	-1231.038363	-1.742813	0.360952	66	97
	1000	5_10_15	2.088095	2.377574	-1067.411832	-1.799337	0.344224	71	79
	1250	5_10_15	2.038839	2.300168	-1041.283981	-1.916373	0.333165	65	80
	1500	5_10_15	2.003647	2.233611	-1015.709169	-1.911827	0.331897	70	86
	1750	5_10_15	1.863913	2.116891	-847.422682	-1.873011	0.384351	80	100
	2000	5_10_15	1.755392	1.968451	-902.791383	-1.885324	0.367528	77	104
	2250	5_10_15	1.875030	2.094815	-970.210895	-1.903458	0.348182	75	98
	2450	5_10_15	1.978064	2.195420	-1169.418610	-1.868428	0.344159	74	99
	500	5_10_15_20	1.798803	2.049565	-985.649265	-1.763951	0.386703	73	88
	750	5_10_15_20	1.969531	2.160537	-1229.529397	-1.738176	0.364726	84	79
	1000	5_10_15_20	2.076872	2.375513	-1050.364124	-1.805705	0.345500	63	66
	1250	5_10_15_20	2.036551	2.309667	-1027.381563	-1.912824	0.335075	67	87
	1500	5_10_15_20	1.995141	2.232724	-1016.829161	-1.905273	0.336682	70	87
	1750	5_10_15_20	1.883580	2.135903	-843.717906	-1.879546	0.379427	80	92
	2000	5_10_15_20	1.746547	1.961984	-899.669851	-1.879879	0.370039	77	122
	2250	5_10_15_20	1.876747	2.097510	-955.052948	-1.904623	0.351259	76	91

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	1.985971	2.200333	-1202.287130	-1.865793	0.344958	76	87
500	5	10_15_20_25	1.802121	2.062714	-978.752097	-1.767187	0.383599	68	101
750	5	10_15_20_25	1.968589	2.187180	-1169.115269	-1.738299	0.367856	82	78
1000	5	10_15_20_25	2.079685	2.388455	-1052.818471	-1.798409	0.347861	78	79
1250	5	10_15_20_25	2.062518	2.336548	-1042.287829	-1.913163	0.331317	69	99
1500	5	10_15_20_25	2.006449	2.247538	-1014.906591	-1.910716	0.330969	63	98
1750	5	10_15_20_25	1.872859	2.129313	-875.439437	-1.876151	0.381462	80	83
2000	5	10_15_20_25	1.749261	1.966278	-902.542832	-1.876438	0.368453	69	125
2250	5	10_15_20_25	1.896398	2.120494	-1006.027345	-1.904907	0.345777	71	105
2450	5	10_15_20_25	1.995820	2.207264	-1211.711917	-1.866329	0.340369	64	90
500	5	10_15_20_25_30	1.855695	2.161635	-988.025401	-1.770212	0.378818	82	99
750	5	10_15_20_25_30	1.970393	2.198070	-1126.187793	-1.748677	0.367331	90	72
1000	5	10_15_20_25_30	2.080983	2.396943	-1047.402493	-1.796953	0.353157	68	89
1250	5	10_15_20_25_30	2.059546	2.334291	-1038.289865	-1.922673	0.329467	61	94
1500	5	10_15_20_25_30	2.014736	2.256852	-1013.342420	-1.914630	0.331077	69	93
1750	5	10_15_20_25_30	1.890302	2.144509	-896.658827	-1.880250	0.378337	80	96
2000	5	10_15_20_25_30	1.754649	1.971306	-922.979769	-1.875431	0.364668	79	108
2250	5	10_15_20_25_30	1.908745	2.132431	-1005.579665	-1.909785	0.343121	72	99

Continued on next page



TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
400	2450	5_10_15_20_25_30	2.035046	2.254537	-1217.301989	-1.864423	0.335918	76	72
	500	15_30_45	1.894807	2.234578	-1054.140407	-1.757466	0.371032	77	104
	750	15_30_45	1.939803	2.223992	-1127.917856	-1.722761	0.381032	79	105
	1000	15_30_45	2.079361	2.393469	-1072.998548	-1.814568	0.345834	79	97
	1250	15_30_45	2.065123	2.349094	-1065.489859	-1.896046	0.328387	74	104
	1500	15_30_45	2.000499	2.236085	-966.514090	-1.905929	0.339732	80	101
	1750	15_30_45	1.826582	2.077762	-899.417258	-1.879009	0.383829	97	112
	2000	15_30_45	1.819645	2.040638	-909.450791	-1.887350	0.352609	79	116
	2250	15_30_45	1.939493	2.167784	-1088.427340	-1.898315	0.341808	89	102
	2450	15_30_45	2.065427	2.297841	-1219.018519	-1.861487	0.335298	76	88
	500	5_10_15	1.842080	2.080588	-1010.156657	-1.757650	0.381558	96	104
	750	5_10_15	1.953417	2.133128	-1188.911075	-1.744380	0.366207	80	109
	1000	5_10_15	2.098170	2.389450	-1052.363055	-1.809905	0.343373	78	91
	1250	5_10_15	2.075129	2.341592	-1068.858305	-1.910702	0.322331	72	88
	1500	5_10_15	1.962096	2.191869	-991.335259	-1.920988	0.336880	80	102
	1750	5_10_15	1.881121	2.129986	-887.247310	-1.887525	0.375839	89	113
	2000	5_10_15	1.819483	2.032477	-902.211693	-1.896148	0.355446	85	109
	2250	5_10_15	1.861365	2.081392	-1015.076723	-1.895794	0.354618	87	114

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.983814	2.206598	-1194.247758	-1.862434	0.343547	84	113
500	5	10_15_20	1.813584	2.060463	-997.310344	-1.757261	0.384704	85	98
750	5	10_15_20	1.948553	2.143004	-1210.329528	-1.742424	0.368055	99	95
1000	5	10_15_20	2.100506	2.402174	-1054.498154	-1.817092	0.341162	74	75
1250	5	10_15_20	2.076357	2.354706	-1053.807982	-1.905727	0.324684	71	98
1500	5	10_15_20	1.960539	2.194670	-988.376126	-1.914062	0.340107	82	110
1750	5	10_15_20	1.887846	2.138575	-890.112341	-1.886364	0.374442	87	114
2000	5	10_15_20	1.820671	2.036867	-904.701351	-1.889652	0.356752	83	131
2250	5	10_15_20	1.869222	2.089848	-999.913850	-1.893129	0.356349	89	101
2450	5	10_15_20	1.978450	2.200134	-1198.229662	-1.862536	0.346073	89	96
500	5	10_15_20_25	1.805865	2.064435	-964.789586	-1.756947	0.384619	82	110
750	5	10_15_20_25	1.944507	2.167936	-1162.406207	-1.737468	0.374067	91	93
1000	5	10_15_20_25	2.096889	2.408704	-1054.450928	-1.813565	0.342598	83	92
1250	5	10_15_20_25	2.091284	2.372288	-1052.262660	-1.903001	0.322545	76	109
1500	5	10_15_20_25	1.974553	2.213183	-989.538282	-1.915086	0.338236	79	114
1750	5	10_15_20_25	1.867658	2.120643	-894.440128	-1.884532	0.377072	88	98
2000	5	10_15_20_25	1.821076	2.039818	-888.649217	-1.886426	0.357203	78	136
2250	5	10_15_20_25	1.882136	2.108410	-1043.106917	-1.891781	0.354747	82	120

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
450	2450	5_10_15_20_25	2.006665	2.225501	-1219.910840	-1.864903	0.337886	73	104
	500	5_10_15_20_25_30	1.868032	2.171123	-993.572747	-1.760450	0.377441	92	109
	750	5_10_15_20_25_30	1.939512	2.170930	-1122.304985	-1.738366	0.375725	102	89
	1000	5_10_15_20_25_30	2.108847	2.427597	-1061.294128	-1.814938	0.343270	77	100
	1250	5_10_15_20_25_30	2.076478	2.360483	-1076.169893	-1.899554	0.326675	77	100
	1500	5_10_15_20_25_30	1.983267	2.224232	-993.567640	-1.914801	0.339315	81	110
	1750	5_10_15_20_25_30	1.873897	2.124231	-897.969949	-1.885804	0.376605	90	114
	2000	5_10_15_20_25_30	1.817218	2.037960	-895.149278	-1.883436	0.356446	89	116
	2250	5_10_15_20_25_30	1.902833	2.127228	-1058.149550	-1.893913	0.350746	82	114
	2450	5_10_15_20_25_30	2.024509	2.251022	-1245.843541	-1.862167	0.337651	86	84
	500	15_30_45	1.868505	2.215253	-1100.157518	-1.755785	0.381648	89	115
	750	15_30_45	1.952423	2.232757	-1167.160444	-1.726809	0.378638	87	124
	1000	15_30_45	2.074237	2.390473	-1108.759682	-1.805738	0.347608	90	111
	1250	15_30_45	2.076951	2.357425	-1117.208751	-1.893310	0.329202	84	115
	1500	15_30_45	1.972535	2.214124	-968.566889	-1.892378	0.349675	90	118
	1750	15_30_45	1.824034	2.068658	-861.705966	-1.890136	0.377423	108	128
	2000	15_30_45	1.803671	2.027462	-942.745182	-1.888649	0.350887	88	128
	2250	15_30_45	1.947112	2.172536	-1084.174274	-1.888704	0.343041	100	114

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15_30_45		2.075666	2.309026	-1163.263931	-1.856937	0.330449	83	97
500	5_10_15		1.851625	2.088347	-1059.970429	-1.755736	0.381316	109	111
750	5_10_15		1.927042	2.110168	-1204.044585	-1.739789	0.373056	90	123
1000	5_10_15		2.085290	2.378464	-1062.654822	-1.809176	0.343532	91	101
1250	5_10_15		2.081017	2.349745	-1095.363873	-1.904388	0.322299	83	94
1500	5_10_15		1.974666	2.208656	-986.593649	-1.902955	0.345034	94	109
1750	5_10_15		1.850102	2.092873	-874.997649	-1.891037	0.373958	96	136
2000	5_10_15		1.833466	2.055477	-925.032156	-1.893505	0.353841	99	122
2250	5_10_15		1.883872	2.102747	-1070.468969	-1.884087	0.355179	98	126
2450	5_10_15		2.043081	2.266302	-1196.950480	-1.865948	0.330397	87	120
500	5_10_15_20		1.809658	2.059722	-1037.322304	-1.750465	0.388409	98	111
750	5_10_15_20		1.920204	2.118057	-1222.423415	-1.735020	0.376448	115	106
1000	5_10_15_20		2.087713	2.391469	-1078.725301	-1.813561	0.343326	86	87
1250	5_10_15_20		2.080036	2.359182	-1072.821922	-1.900576	0.324658	83	105
1500	5_10_15_20		1.981020	2.218334	-996.895557	-1.901219	0.345910	92	123
1750	5_10_15_20		1.856240	2.100447	-868.346657	-1.891929	0.374203	100	132
2000	5_10_15_20		1.834754	2.055846	-933.539980	-1.890861	0.352274	92	145
2250	5_10_15_20		1.881269	2.099486	-1055.930766	-1.881122	0.357422	102	111

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	2.035036	2.256869	-1185.459442	-1.863234	0.333072	94	102
500	5	10_15_20_25	1.811266	2.071701	-1033.210593	-1.755792	0.384524	95	121
750	5	10_15_20_25	1.933447	2.158682	-1201.153446	-1.733110	0.377290	100	105
1000	5	10_15_20_25	2.092924	2.404078	-1101.487598	-1.811473	0.342512	90	103
1250	5	10_15_20_25	2.093003	2.374662	-1082.081759	-1.902642	0.323442	87	121
1500	5	10_15_20_25	1.976326	2.220209	-987.656821	-1.899518	0.345494	86	134
1750	5	10_15_20_25	1.837374	2.084576	-863.356211	-1.890111	0.376535	98	118
2000	5	10_15_20_25	1.814826	2.040956	-918.272577	-1.882142	0.357437	88	152
2250	5	10_15_20_25	1.892566	2.119139	-1054.339739	-1.881890	0.355424	95	130
2450	5	10_15_20_25	2.050996	2.271101	-1184.513852	-1.866029	0.327203	81	114
500	5	10_15_20_25_30	1.864902	2.173983	-1047.338893	-1.755836	0.380574	101	122
750	5	10_15_20_25_30	1.943302	2.174205	-1171.350212	-1.738767	0.376172	111	102
1000	5	10_15_20_25_30	2.107961	2.423656	-1097.357765	-1.812892	0.342055	84	114
1250	5	10_15_20_25_30	2.096451	2.381736	-1096.985176	-1.900379	0.324933	83	112
1500	5	10_15_20_25_30	1.977412	2.223497	-986.051897	-1.901113	0.347068	89	123
1750	5	10_15_20_25_30	1.843592	2.088881	-873.681930	-1.889190	0.374753	101	129
2000	5	10_15_20_25_30	1.800358	2.028245	-919.072909	-1.880614	0.358096	101	130
2250	5	10_15_20_25_30	1.907364	2.131306	-1054.503988	-1.883582	0.351892	91	130

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
500	2450	5_10_15_20_25_30	2.072479	2.297684	-1189.008507	-1.867189	0.324498	94	88
	500	15_30_45	1.865732	2.210292	-1070.512858	-1.755748	0.380944	97	132
	750	15_30_45	1.982211	2.261116	-1179.792355	-1.744432	0.368470	94	132
	1000	15_30_45	2.040465	2.354330	-1111.017423	-1.824024	0.350461	97	133
	1250	15_30_45	2.082159	2.359415	-1095.791854	-1.892081	0.327777	95	126
	1500	15_30_45	1.973429	2.222980	-938.967996	-1.888283	0.355178	102	133
	1750	15_30_45	1.770959	2.012041	-841.771214	-1.878328	0.384650	124	144
	2000	15_30_45	1.826911	2.049640	-945.065500	-1.896317	0.349001	96	143
	2250	15_30_45	1.987005	2.210760	-1088.518929	-1.885416	0.334374	109	126
	2450	15_30_45	2.052136	2.292538	-1123.743773	-1.845320	0.337652	97	108
	500	5_10_15	1.822328	2.062688	-1062.899750	-1.750503	0.390117	126	126
	750	5_10_15	1.933162	2.115694	-1190.078756	-1.740421	0.372663	99	134
	1000	5_10_15	2.085298	2.379187	-1109.786655	-1.814685	0.344056	100	116
	1250	5_10_15	2.086942	2.351504	-1150.170113	-1.895525	0.325094	92	109
	1500	5_10_15	1.952572	2.196272	-964.777628	-1.888587	0.352300	104	122
	1750	5_10_15	1.836646	2.075793	-843.440990	-1.893769	0.370687	109	148
	2000	5_10_15	1.846394	2.065205	-958.323813	-1.901557	0.348680	111	134
	2250	5_10_15	1.916873	2.134556	-1049.310887	-1.877415	0.348633	105	139

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	2.020900	2.249550	-1145.306781	-1.858285	0.335753	95	135
500	5	10_15_20	1.796897	2.047302	-1052.363127	-1.747373	0.392660	112	128
750	5	10_15_20	1.934442	2.129304	-1225.739039	-1.744698	0.372497	127	115
1000	5	10_15_20	2.085707	2.389248	-1115.392463	-1.821467	0.341943	92	101
1250	5	10_15_20	2.080624	2.354345	-1124.363169	-1.893655	0.327825	96	118
1500	5	10_15_20	1.957230	2.203632	-958.035916	-1.890334	0.352895	105	137
1750	5	10_15_20	1.841321	2.081039	-855.280591	-1.895130	0.368195	110	149
2000	5	10_15_20	1.845762	2.066433	-955.386661	-1.898447	0.350807	99	164
2250	5	10_15_20	1.912864	2.130891	-1050.678802	-1.873591	0.352653	112	120
2450	5	10_15_20	2.008341	2.236363	-1138.114800	-1.853656	0.341275	107	113
500	5	10_15_20_25	1.783975	2.045089	-1017.828267	-1.751546	0.392853	108	140
750	5	10_15_20_25	1.934690	2.160582	-1196.562960	-1.744036	0.374907	108	117
1000	5	10_15_20_25	2.076004	2.387042	-1117.755384	-1.820042	0.344062	105	114
1250	5	10_15_20_25	2.097467	2.373850	-1122.133270	-1.894705	0.325256	96	128
1500	5	10_15_20_25	1.957388	2.209263	-954.987225	-1.893982	0.351464	97	145
1750	5	10_15_20_25	1.819415	2.060949	-850.194729	-1.890791	0.372280	107	140
2000	5	10_15_20_25	1.836981	2.059868	-944.056772	-1.891174	0.352187	97	163
2250	5	10_15_20_25	1.926692	2.151140	-1054.467796	-1.875209	0.349247	103	147

Continued on next page

TABLE A.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20_25	2.011312	2.239885	-1137.456634	-1.852128	0.340685	93	127
500	5	10_15_20_25_30	1.848348	2.154630	-1039.961795	-1.755147	0.384152	113	139
750	5	10_15_20_25_30	1.958190	2.189915	-1180.587658	-1.747305	0.371686	122	115
1000	5	10_15_20_25_30	2.078447	2.393738	-1110.557259	-1.821765	0.344690	92	130
1250	5	10_15_20_25_30	2.089168	2.371296	-1122.756183	-1.893381	0.328081	94	123
1500	5	10_15_20_25_30	1.973405	2.225978	-959.964686	-1.896693	0.351670	98	134
1750	5	10_15_20_25_30	1.813873	2.053255	-849.991615	-1.888172	0.374398	113	145
2000	5	10_15_20_25_30	1.823130	2.046540	-938.098492	-1.889542	0.353021	111	147
2250	5	10_15_20_25_30	1.940443	2.162325	-1060.095667	-1.877816	0.345469	99	139
2450	5	10_15_20_25_30	2.038922	2.274503	-1133.625889	-1.850515	0.337207	104	105

End of the table



## Appendix B

# Results of Executions Without Structure Learning

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
50	500	15_30_45	0.827744	1.599234	-388.616850	-1.429436	0.734854	15	21
	750	15_30_45	1.240404	2.012672	-733.447168	-1.446908	0.609886	11	13
	1000	15_30_45	1.318558	2.125340	-615.662262	-1.444556	0.597420	15	16
	1250	15_30_45	1.491206	2.244324	-778.146118	-1.509678	0.538434	11	17
	1500	15_30_45	0.839198	1.481826	-481.339836	-1.393024	0.706440	15	20
	1750	15_30_45	1.267274	1.972194	-729.776522	-1.456500	0.589852	11	21
	2000	15_30_45	1.242784	1.898828	-959.353704	-1.476410	0.572430	6	22
	2250	15_30_45	1.075068	1.646386	-699.416416	-1.406542	0.620762	19	19

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15	30_45	1.412984	2.033200	-693.787966	-1.525646	0.537752	12	19
500	5	10_15	0.911048	1.678688	-381.404884	-1.437316	0.704712	15	19
750	5	10_15	1.130708	1.885472	-814.488622	-1.444470	0.637296	11	16
1000	5	10_15	1.290634	2.073266	-556.322040	-1.450598	0.598916	13	19
1250	5	10_15	1.519498	2.271380	-1084.160920	-1.512732	0.534316	17	12
1500	5	10_15	0.856584	1.497382	-437.084728	-1.407678	0.693300	15	24
1750	5	10_15	1.438024	2.124512	-715.487538	-1.513534	0.539308	12	15
2000	5	10_15	1.287652	1.961114	-590.845100	-1.469702	0.578778	13	17
2250	5	10_15	0.707758	1.217166	-445.835430	-1.268274	0.723356	15	26
2450	5	10_15	1.318016	1.924044	-874.460834	-1.521450	0.547176	10	19
500	5	10_15_20	0.851430	1.597846	-369.231134	-1.432418	0.721578	13	22
750	5	10_15_20	1.090804	1.823296	-867.056730	-1.439548	0.642574	11	19
1000	5	10_15_20	1.310642	2.087844	-508.571264	-1.455652	0.592082	11	16
1250	5	10_15_20	1.482654	2.237886	-914.824566	-1.505514	0.543828	12	14
1500	5	10_15_20	0.840770	1.489978	-441.388830	-1.393276	0.702350	16	18
1750	5	10_15_20	1.447376	2.131652	-743.883832	-1.512228	0.534498	8	21
2000	5	10_15_20	1.299704	1.975078	-682.137764	-1.463308	0.579604	14	18
2250	5	10_15_20	0.680602	1.196954	-434.096122	-1.246466	0.735520	21	22

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	1.335996	1.948576	-828.599540	-1.519472	0.549952	14	13
500	5	10_15_20_25	0.778918	1.508240	-370.716280	-1.425038	0.741640	17	16
750	5	10_15_20_25	1.077380	1.809028	-759.006696	-1.436578	0.645736	11	25
1000	5	10_15_20_25	1.323024	2.105748	-570.139008	-1.454066	0.587868	12	16
1250	5	10_15_20_25	1.487312	2.231710	-817.361716	-1.506878	0.538594	10	19
1500	5	10_15_20_25	0.872788	1.527664	-488.474618	-1.401714	0.693576	20	13
1750	5	10_15_20_25	1.360832	2.025172	-770.855814	-1.499438	0.550396	15	18
2000	5	10_15_20_25	1.247886	1.914036	-681.097506	-1.455714	0.586900	13	15
2250	5	10_15_20_25	0.737758	1.261710	-460.130004	-1.273968	0.716828	16	26
2450	5	10_15_20_25	1.336876	1.930234	-810.865544	-1.519944	0.546948	16	19
500	5	10_15_20_25_30	0.784096	1.524074	-363.885058	-1.428736	0.739948	15	16
750	5	10_15_20_25_30	1.117652	1.859186	-705.363756	-1.446036	0.635890	11	17
1000	5	10_15_20_25_30	1.288078	2.072666	-532.961226	-1.446432	0.598992	16	17
1250	5	10_15_20_25_30	1.526474	2.281278	-822.009626	-1.507886	0.531626	9	12
1500	5	10_15_20_25_30	0.887688	1.538986	-524.674610	-1.408232	0.691456	23	18
1750	5	10_15_20_25_30	1.369172	2.036546	-735.187560	-1.499526	0.552852	14	17
2000	5	10_15_20_25_30	1.223948	1.894914	-787.280374	-1.450632	0.589392	12	11
2250	5	10_15_20_25_30	0.863692	1.390356	-496.137572	-1.325042	0.677600	13	24

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
100	2450	5_10_15_20_25_30	1.343120	1.946732	-817.653046	-1.518198	0.548762	15	16
	500	15_30_45	0.935624	1.716269	-458.926067	-1.425543	0.702116	26	45
	750	15_30_45	1.244179	2.023193	-838.358323	-1.436529	0.608455	22	28
	1000	15_30_45	1.237520	2.015035	-558.831234	-1.446995	0.609174	31	30
	1250	15_30_45	1.347268	2.089942	-716.094123	-1.481714	0.577764	27	36
	1500	15_30_45	1.064898	1.759794	-501.643782	-1.435960	0.645791	27	33
	1750	15_30_45	1.241336	1.936039	-600.189201	-1.448340	0.599255	28	37
	2000	15_30_45	1.320702	1.964899	-936.962618	-1.488225	0.557153	18	38
	2250	15_30_45	1.292818	1.907014	-729.513124	-1.460402	0.573135	32	35
	2450	15_30_45	1.425455	2.064998	-704.249570	-1.516088	0.537835	19	37
	500	5_10_15	0.916330	1.687491	-399.750031	-1.429484	0.707633	33	33
	750	5_10_15	1.203476	1.967205	-857.309780	-1.443359	0.619041	20	30
	1000	5_10_15	1.329559	2.114931	-629.793545	-1.461296	0.586545	24	39
	1250	5_10_15	1.452300	2.206178	-822.699420	-1.498647	0.551733	29	27
	1500	5_10_15	0.945693	1.615477	-478.973165	-1.412893	0.679271	43	32
	1750	5_10_15	1.337165	2.027711	-615.704290	-1.479126	0.572229	27	26
	2000	5_10_15	1.287214	1.938680	-867.739413	-1.477798	0.568746	27	33
	2250	5_10_15	0.983041	1.538436	-576.652886	-1.369459	0.647951	28	39

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.353623	1.984218	-786.494335	-1.515383	0.543078	22	34
500	5	10_15_20	0.911701	1.674689	-385.498436	-1.428629	0.707482	30	42
750	5	10_15_20	1.187131	1.947892	-902.051250	-1.437306	0.623344	24	38
1000	5	10_15_20	1.315945	2.098422	-593.023919	-1.458892	0.588431	22	32
1250	5	10_15_20	1.407605	2.158014	-731.459438	-1.491410	0.562809	22	33
1500	5	10_15_20	0.948522	1.619297	-507.009128	-1.411944	0.678741	32	34
1750	5	10_15_20	1.340118	2.039799	-598.307993	-1.471665	0.574640	20	40
2000	5	10_15_20	1.289047	1.946590	-870.751733	-1.474785	0.571394	29	36
2250	5	10_15_20	1.020475	1.586966	-574.323568	-1.375834	0.640193	36	37
2450	5	10_15_20	1.332430	1.966737	-757.916222	-1.508837	0.549993	22	32
500	5	10_15_20_25	0.912746	1.677758	-384.574432	-1.424852	0.707658	36	34
750	5	10_15_20_25	1.177617	1.940072	-861.953792	-1.433795	0.625726	26	38
1000	5	10_15_20_25	1.297570	2.077971	-546.762974	-1.453974	0.593613	29	33
1250	5	10_15_20_25	1.403371	2.146136	-682.953614	-1.489529	0.562075	20	40
1500	5	10_15_20_25	0.988192	1.660835	-524.406045	-1.420615	0.667548	33	34
1750	5	10_15_20_25	1.332627	2.023583	-632.224790	-1.473808	0.573198	31	31
2000	5	10_15_20_25	1.283992	1.940091	-893.449842	-1.470835	0.571762	30	30
2250	5	10_15_20_25	1.077368	1.654824	-665.130914	-1.392562	0.627289	23	48

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
150	2450	5_10_15_20_25	1.375149	2.003655	-733.390695	-1.519810	0.540706	24	37
	500	5_10_15_20_25_30	0.939230	1.717667	-348.882542	-1.429659	0.702507	29	36
	750	5_10_15_20_25_30	1.197111	1.968366	-817.370590	-1.437967	0.622461	26	32
	1000	5_10_15_20_25_30	1.276890	2.055256	-536.150639	-1.455496	0.599353	30	36
	1250	5_10_15_20_25_30	1.394868	2.139858	-682.894132	-1.481856	0.565812	18	29
	1500	5_10_15_20_25_30	1.034400	1.710364	-539.708741	-1.431562	0.655660	37	38
	1750	5_10_15_20_25_30	1.331200	2.024010	-623.230123	-1.474060	0.575686	27	36
	2000	5_10_15_20_25_30	1.287577	1.940301	-930.858225	-1.472757	0.568722	28	25
	2250	5_10_15_20_25_30	1.140487	1.722848	-712.621744	-1.414091	0.609556	27	39
	2450	5_10_15_20_25_30	1.404414	2.041046	-725.033313	-1.521773	0.535932	26	32
	500	15_30_45	0.973772	1.762631	-435.979172	-1.427631	0.690805	36	65
	750	15_30_45	1.226760	2.003225	-812.080880	-1.438148	0.610956	36	46
	1000	15_30_45	1.227282	1.991755	-655.129317	-1.440931	0.608672	48	49
	1250	15_30_45	1.360204	2.110877	-695.260987	-1.478118	0.575265	36	58
	1500	15_30_45	1.164363	1.867228	-681.754970	-1.446834	0.617587	41	46
	1750	15_30_45	1.240321	1.928032	-639.600973	-1.445313	0.599509	45	52
	2000	15_30_45	1.197257	1.813879	-828.996273	-1.458534	0.584773	33	58
	2250	15_30_45	1.343690	1.974725	-689.080505	-1.473630	0.560943	41	55

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15_30_45		1.404027	2.040403	-781.264306	-1.515469	0.538876	28	58
500	5_10_15		0.957877	1.740517	-429.345565	-1.422776	0.697041	52	46
750	5_10_15		1.214980	1.976998	-871.937951	-1.442575	0.611867	31	47
1000	5_10_15		1.262984	2.031654	-625.345562	-1.453083	0.600485	36	55
1250	5_10_15		1.405605	2.158931	-755.155920	-1.485230	0.566465	42	46
1500	5_10_15		1.068396	1.759758	-533.469017	-1.433379	0.643473	49	52
1750	5_10_15		1.248995	1.940337	-659.165187	-1.455306	0.597306	34	53
2000	5_10_15		1.287271	1.928097	-843.992866	-1.479776	0.567256	41	51
2250	5_10_15		1.172138	1.764943	-630.478845	-1.421031	0.603847	36	55
2450	5_10_15		1.353612	1.983752	-739.438633	-1.510219	0.547019	36	46
500	5_10_15_20		0.959839	1.735120	-426.410090	-1.422792	0.694804	40	65
750	5_10_15_20		1.191935	1.950689	-864.558358	-1.439119	0.618231	37	58
1000	5_10_15_20		1.250743	2.016131	-638.816615	-1.453247	0.602648	40	49
1250	5_10_15_20		1.385071	2.139588	-698.529154	-1.481910	0.571946	34	51
1500	5_10_15_20		1.088143	1.784378	-577.658377	-1.433395	0.639777	45	46
1750	5_10_15_20		1.257587	1.947041	-665.802311	-1.455563	0.593540	31	62
2000	5_10_15_20		1.277270	1.919887	-836.794475	-1.476260	0.570337	46	56
2250	5_10_15_20		1.205853	1.804987	-636.558550	-1.428109	0.595827	46	51

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	5_10_15_20	1.346385	1.975233	-734.892449	-1.509332	0.549515	35	52
	500	5_10_15_20_25	0.962165	1.738564	-428.370931	-1.423463	0.693433	52	53
	750	5_10_15_20_25	1.174005	1.934645	-799.897503	-1.434369	0.623182	37	54
	1000	5_10_15_20_25	1.251031	2.014709	-617.193517	-1.450308	0.602837	41	53
	1250	5_10_15_20_25	1.382147	2.138595	-671.964705	-1.479606	0.572917	38	57
	1500	5_10_15_20_25	1.121635	1.822476	-597.689331	-1.438930	0.631608	50	50
	1750	5_10_15_20_25	1.261080	1.944421	-680.634407	-1.455959	0.592562	43	48
	2000	5_10_15_20_25	1.253945	1.891515	-828.907385	-1.467935	0.574960	40	55
	2250	5_10_15_20_25	1.235716	1.840468	-628.232807	-1.439456	0.587995	37	61
	2450	5_10_15_20_25	1.366721	1.991173	-811.157047	-1.512915	0.543805	37	56
	500	5_10_15_20_25_30	0.975014	1.760240	-433.414196	-1.427474	0.690198	37	56
	750	5_10_15_20_25_30	1.185031	1.953913	-787.409478	-1.436865	0.621501	32	55
	1000	5_10_15_20_25_30	1.236081	2.001700	-619.563372	-1.448125	0.607200	40	57
	1250	5_10_15_20_25_30	1.379912	2.135761	-683.627798	-1.478227	0.572599	32	45
	1500	5_10_15_20_25_30	1.145015	1.844402	-653.569209	-1.443928	0.624619	49	55
	1750	5_10_15_20_25_30	1.257165	1.940026	-670.587001	-1.454145	0.594855	39	55
	2000	5_10_15_20_25_30	1.239246	1.872659	-840.919268	-1.464489	0.577821	46	46
	2250	5_10_15_20_25_30	1.265247	1.876637	-638.388642	-1.451581	0.579185	36	52

Continued on next page



TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
200	2450	5_10_15_20_25_30	1.384809	2.013154	-811.862427	-1.513573	0.541090	38	50
	500	15_30_45	0.970969	1.755470	-462.079652	-1.430742	0.690348	50	85
	750	15_30_45	1.188543	1.952622	-827.441812	-1.437596	0.618617	52	64
	1000	15_30_45	1.234773	1.987716	-686.058189	-1.446184	0.604599	63	69
	1250	15_30_45	1.379037	2.133621	-782.185255	-1.478216	0.571029	47	71
	1500	15_30_45	1.223542	1.924063	-751.091826	-1.460547	0.599213	52	65
	1750	15_30_45	1.247942	1.937328	-704.006638	-1.449415	0.594967	59	66
	2000	15_30_45	1.176995	1.782362	-719.878522	-1.449515	0.592929	46	73
	2250	15_30_45	1.304991	1.932942	-751.460736	-1.477596	0.564528	57	76
	2450	15_30_45	1.440630	2.084168	-937.841099	-1.517652	0.529812	38	77
	500	5_10_15	0.974119	1.752903	-452.154223	-1.427442	0.689468	73	62
	750	5_10_15	1.191488	1.949775	-843.160839	-1.440711	0.617569	48	59
	1000	5_10_15	1.245383	2.002559	-688.546059	-1.451912	0.601661	48	72
	1250	5_10_15	1.392643	2.145066	-791.984145	-1.481392	0.567777	60	58
	1500	5_10_15	1.138938	1.838043	-628.782361	-1.443464	0.624358	56	66
	1750	5_10_15	1.286837	1.973170	-685.011362	-1.463958	0.585561	52	69
	2000	5_10_15	1.202216	1.826209	-763.119384	-1.452887	0.589809	60	68
	2250	5_10_15	1.214689	1.821495	-674.113087	-1.436245	0.592886	56	69

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.379043	2.009753	-851.767961	-1.515763	0.538812	51	58
500	5	10_15_20	0.966714	1.738455	-441.523683	-1.427387	0.690087	59	82
750	5	10_15_20	1.173767	1.932294	-835.511557	-1.436158	0.623493	51	78
1000	5	10_15_20	1.247107	2.003885	-693.792946	-1.451268	0.600983	56	61
1250	5	10_15_20	1.388929	2.142696	-763.142733	-1.480095	0.569459	44	69
1500	5	10_15_20	1.147298	1.846875	-642.393240	-1.444432	0.621756	55	62
1750	5	10_15_20	1.292972	1.980228	-693.707228	-1.463843	0.583821	39	79
2000	5	10_15_20	1.200712	1.823379	-748.220779	-1.450388	0.592303	53	85
2250	5	10_15_20	1.234306	1.846623	-686.386700	-1.441412	0.588251	53	70
2450	5	10_15_20	1.375368	2.008419	-837.043097	-1.513977	0.541158	50	64
500	5	10_15_20_25	0.965457	1.736721	-443.872802	-1.428720	0.689907	69	75
750	5	10_15_20_25	1.166059	1.927219	-797.157867	-1.434459	0.625599	44	74
1000	5	10_15_20_25	1.251375	2.006333	-696.622075	-1.450592	0.599629	50	71
1250	5	10_15_20_25	1.394537	2.146377	-752.165918	-1.482586	0.567012	48	76
1500	5	10_15_20_25	1.167434	1.866153	-672.882869	-1.449901	0.615631	62	66
1750	5	10_15_20_25	1.280994	1.966422	-702.499392	-1.459729	0.586443	59	66
2000	5	10_15_20_25	1.193439	1.814015	-738.847725	-1.448325	0.592786	54	74
2250	5	10_15_20_25	1.245186	1.864571	-695.252437	-1.446942	0.585350	49	81

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
250	2450	5_10_15_20_25	1.382332	2.010374	-862.182514	-1.516134	0.538878	49	70
	500	5_10_15_20_25_30	0.975640	1.756789	-427.056195	-1.430254	0.688902	56	74
	750	5_10_15_20_25_30	1.166156	1.933934	-783.088730	-1.435107	0.626595	43	77
	1000	5_10_15_20_25_30	1.250265	2.002788	-694.178409	-1.452651	0.599775	49	69
	1250	5_10_15_20_25_30	1.396175	2.147845	-750.269102	-1.481445	0.566232	45	60
	1500	5_10_15_20_25_30	1.191370	1.893334	-696.465529	-1.453838	0.609818	62	77
	1750	5_10_15_20_25_30	1.268127	1.952208	-690.560336	-1.457075	0.590299	51	74
	2000	5_10_15_20_25_30	1.176658	1.791345	-738.819596	-1.446398	0.594981	59	68
	2250	5_10_15_20_25_30	1.258809	1.878250	-700.500855	-1.456464	0.579411	52	68
	2450	5_10_15_20_25_30	1.400578	2.032513	-886.560310	-1.515376	0.535834	46	72
	500	15_30_45	0.967862	1.747265	-533.112799	-1.431915	0.689047	60	106
	750	15_30_45	1.195096	1.961132	-794.826213	-1.441402	0.617522	62	85
	1000	15_30_45	1.266384	2.024388	-717.730718	-1.453402	0.598108	72	93
	1250	15_30_45	1.288880	2.030770	-744.824843	-1.467287	0.593613	58	93
	1500	15_30_45	1.272681	1.977475	-769.006328	-1.471132	0.586747	59	85
	1750	15_30_45	1.272320	1.961304	-667.042320	-1.453344	0.589154	69	87
	2000	15_30_45	1.083268	1.671544	-659.443936	-1.415452	0.618367	67	94
	2250	15_30_45	1.321940	1.948564	-736.743382	-1.485076	0.560775	68	96

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	15_30_45	1.440155	2.088190	-929.501454	-1.518123	0.528996	48	94
	500	5_10_15	0.976798	1.753070	-475.178253	-1.430228	0.688218	90	82
	750	5_10_15	1.181600	1.939856	-851.398543	-1.439724	0.620040	60	77
	1000	5_10_15	1.267029	2.025253	-671.225022	-1.454149	0.596636	57	92
	1250	5_10_15	1.368746	2.117629	-825.517104	-1.479550	0.573141	70	75
	1500	5_10_15	1.206439	1.905808	-736.371594	-1.456885	0.604876	66	79
	1750	5_10_15	1.298390	1.988638	-674.974270	-1.464861	0.581849	68	84
	2000	5_10_15	1.170605	1.783106	-684.493641	-1.443892	0.597480	74	90
	2250	5_10_15	1.230766	1.837865	-713.467992	-1.450750	0.585476	72	87
	2450	5_10_15	1.396604	2.037180	-914.366320	-1.512020	0.537423	64	75
	500	5_10_15_20	0.976547	1.748770	-473.410737	-1.430574	0.686920	71	103
	750	5_10_15_20	1.172342	1.927920	-848.070190	-1.437975	0.622966	67	91
	1000	5_10_15_20	1.269236	2.024341	-694.817933	-1.456515	0.594850	76	71
	1250	5_10_15_20	1.355136	2.102980	-796.669122	-1.477892	0.576642	57	84
	1500	5_10_15_20	1.206416	1.904835	-748.294650	-1.458146	0.604292	67	77
	1750	5_10_15_20	1.304465	1.994237	-673.618158	-1.466380	0.579570	53	99
	2000	5_10_15_20	1.153537	1.761136	-681.004221	-1.439337	0.601960	65	104
	2250	5_10_15_20	1.250014	1.863166	-710.751384	-1.454646	0.582129	76	85

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	5_10_15_20	1.388429	2.028588	-921.046434	-1.510804	0.539191	66	79
	500	5_10_15_20_25	0.981419	1.754223	-502.380034	-1.431663	0.684735	86	95
	750	5_10_15_20_25	1.161285	1.916686	-808.462094	-1.436589	0.625573	58	89
	1000	5_10_15_20_25	1.268604	2.024866	-700.941496	-1.455553	0.595606	62	87
	1250	5_10_15_20_25	1.339870	2.086240	-773.444632	-1.474910	0.580206	58	91
	1500	5_10_15_20_25	1.220786	1.922857	-743.112981	-1.460860	0.601269	78	78
	1750	5_10_15_20_25	1.302722	1.988241	-683.344086	-1.466684	0.579053	71	84
	2000	5_10_15_20_25	1.134113	1.737267	-669.041128	-1.430851	0.606916	65	98
	2250	5_10_15_20_25	1.258283	1.873525	-718.653453	-1.459005	0.579729	61	101
	2450	5_10_15_20_25	1.401799	2.041910	-902.015646	-1.513728	0.536356	60	88
	500	5_10_15_20_25_30	0.979291	1.755693	-504.949674	-1.431739	0.685828	77	90
	750	5_10_15_20_25_30	1.172166	1.934102	-801.454684	-1.439285	0.623443	53	94
	1000	5_10_15_20_25_30	1.267542	2.024638	-702.334475	-1.455880	0.597154	60	86
	1250	5_10_15_20_25_30	1.327846	2.073223	-769.892675	-1.471692	0.583093	60	81
	1500	5_10_15_20_25_30	1.237754	1.941943	-759.888257	-1.464133	0.596483	74	95
	1750	5_10_15_20_25_30	1.303857	1.989463	-677.223937	-1.466068	0.580490	64	87
	2000	5_10_15_20_25_30	1.111923	1.711588	-670.174231	-1.422806	0.612582	75	95
	2250	5_10_15_20_25_30	1.271121	1.888070	-720.947023	-1.466056	0.574854	64	85

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
300	2450	5_10_15_20_25_30	1.410059	2.051790	-915.081098	-1.513292	0.534499	55	91
	500	15_30_45	0.983989	1.764651	-550.636177	-1.431550	0.684099	72	122
	750	15_30_45	1.199326	1.972468	-753.443869	-1.439863	0.618699	73	102
	1000	15_30_45	1.288525	2.047943	-719.054028	-1.459994	0.592464	88	107
	1250	15_30_45	1.206628	1.935066	-701.076538	-1.457146	0.615024	72	111
	1500	15_30_45	1.267554	1.976193	-763.184033	-1.468667	0.588830	73	102
	1750	15_30_45	1.262090	1.948699	-710.349651	-1.455685	0.588443	87	104
	2000	15_30_45	1.077813	1.662440	-661.769157	-1.412629	0.619835	81	112
	2250	15_30_45	1.338815	1.970513	-731.373589	-1.487538	0.557693	79	112
	2450	15_30_45	1.489100	2.147519	-939.077413	-1.524739	0.518696	65	106
	500	5_10_15	0.973658	1.748719	-508.084258	-1.429819	0.688512	106	100
	750	5_10_15	1.182999	1.945302	-792.254122	-1.439232	0.621516	77	92
	1000	5_10_15	1.293487	2.054850	-729.388411	-1.461693	0.590997	72	105
	1250	5_10_15	1.274341	2.006323	-757.102632	-1.467691	0.596197	82	97
	1500	5_10_15	1.237212	1.939226	-731.763052	-1.464779	0.596678	77	98
	1750	5_10_15	1.292451	1.983432	-658.354312	-1.464425	0.583170	83	98
	2000	5_10_15	1.091204	1.686524	-644.419181	-1.413800	0.619223	94	111
	2250	5_10_15	1.254984	1.870301	-709.505673	-1.459428	0.578503	80	106

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.434022	2.084391	-903.873232	-1.518665	0.528933	72	90
500	5	10_15_20	0.966445	1.734401	-511.992377	-1.428798	0.688687	82	127
750	5	10_15_20	1.180296	1.940246	-782.908726	-1.438835	0.622407	74	111
1000	5	10_15_20	1.291197	2.050368	-726.480745	-1.462014	0.590663	88	91
1250	5	10_15_20	1.259565	1.992278	-733.042063	-1.464582	0.600863	76	108
1500	5	10_15_20	1.237942	1.937589	-744.417394	-1.465592	0.595234	77	92
1750	5	10_15_20	1.298733	1.988631	-670.791649	-1.464886	0.581098	68	114
2000	5	10_15_20	1.070771	1.661447	-635.116336	-1.406569	0.625447	84	127
2250	5	10_15_20	1.261046	1.881182	-705.637429	-1.460908	0.577430	89	106
2450	5	10_15_20	1.427858	2.077423	-914.441715	-1.516284	0.530630	87	92
500	5	10_15_20_25	0.972736	1.741457	-525.678688	-1.429497	0.686319	100	114
750	5	10_15_20_25	1.173984	1.934914	-758.571399	-1.437812	0.623607	77	103
1000	5	10_15_20_25	1.288943	2.046366	-713.121854	-1.461206	0.591064	66	107
1250	5	10_15_20_25	1.253194	1.986879	-724.083030	-1.464175	0.601991	77	106
1500	5	10_15_20_25	1.237988	1.937796	-747.197254	-1.466422	0.594995	92	92
1750	5	10_15_20_25	1.291193	1.976655	-680.358545	-1.464061	0.581440	81	100
2000	5	10_15_20_25	1.064501	1.654357	-633.384999	-1.403504	0.626470	81	122
2250	5	10_15_20_25	1.282881	1.905824	-708.100172	-1.467757	0.572600	71	119

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
350	2450	5_10_15_20_25	1.436169	2.084742	-909.416815	-1.519357	0.528405	69	106
	500	5_10_15_20_25_30	0.977921	1.750773	-520.885901	-1.431516	0.685056	93	109
	750	5_10_15_20_25_30	1.177751	1.944883	-748.697787	-1.438197	0.623629	73	106
	1000	5_10_15_20_25_30	1.294758	2.054650	-714.766326	-1.461711	0.590801	70	105
	1250	5_10_15_20_25_30	1.244285	1.975786	-723.478171	-1.462188	0.604247	76	96
	1500	5_10_15_20_25_30	1.254046	1.956798	-756.640478	-1.468992	0.591203	88	112
	1750	5_10_15_20_25_30	1.284892	1.971052	-691.667383	-1.462498	0.584043	79	105
	2000	5_10_15_20_25_30	1.067016	1.655252	-638.914803	-1.404962	0.624870	92	117
	2250	5_10_15_20_25_30	1.301097	1.926913	-705.600396	-1.475067	0.567161	73	101
	2450	5_10_15_20_25_30	1.449753	2.101279	-932.240850	-1.519766	0.525732	66	106
	500	15_30_45	0.999469	1.782039	-580.996326	-1.429489	0.679498	86	139
	750	15_30_45	1.181014	1.951979	-711.810698	-1.440169	0.622772	91	119
	1000	15_30_45	1.268409	2.026747	-704.238253	-1.458207	0.598770	102	126
	1250	15_30_45	1.211003	1.943477	-672.621007	-1.459643	0.613365	83	130
	1500	15_30_45	1.253747	1.964196	-719.724070	-1.464833	0.593840	83	122
	1750	15_30_45	1.273565	1.954261	-733.082512	-1.461532	0.583950	98	124
	2000	15_30_45	1.138260	1.734660	-675.334221	-1.425673	0.607471	93	131
	2250	15_30_45	1.337805	1.969337	-754.514896	-1.489955	0.556504	93	126

Continued on next page



TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15_30_45		1.526789	2.187688	-1018.554337	-1.530317	0.510889	73	122
500	5_10_15		0.988965	1.765642	-539.764295	-1.429304	0.683612	120	118
750	5_10_15		1.192585	1.959069	-766.382455	-1.441692	0.619523	88	106
1000	5_10_15		1.292802	2.056458	-700.362755	-1.463092	0.592085	90	119
1250	5_10_15		1.235015	1.964578	-724.662303	-1.462801	0.607774	99	116
1500	5_10_15		1.232063	1.936576	-701.894711	-1.461483	0.599324	93	116
1750	5_10_15		1.285169	1.970006	-720.703007	-1.466258	0.581775	97	118
2000	5_10_15		1.111359	1.709093	-652.412090	-1.420617	0.613881	107	130
2250	5_10_15		1.265844	1.882927	-698.323581	-1.463451	0.576226	97	122
2450	5_10_15		1.479709	2.134130	-966.538847	-1.525103	0.518697	81	109
500	5_10_15_20		0.984637	1.757312	-547.075846	-1.427880	0.683726	103	141
750	5_10_15_20		1.182391	1.946526	-754.532893	-1.440093	0.622135	91	130
1000	5_10_15_20		1.289191	2.049668	-698.224529	-1.463259	0.592082	108	103
1250	5_10_15_20		1.223442	1.951273	-705.847793	-1.461067	0.610690	92	129
1500	5_10_15_20		1.231918	1.938997	-704.655091	-1.460180	0.599918	90	108
1750	5_10_15_20		1.291292	1.975914	-720.719389	-1.466615	0.580167	81	135
2000	5_10_15_20		1.108393	1.704165	-644.716511	-1.419379	0.615302	98	141
2250	5_10_15_20		1.273917	1.893879	-702.359771	-1.466237	0.574283	106	121

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	1.479172	2.132153	-976.884949	-1.524841	0.518790	96	106
500	5	10_15_20_25	0.989580	1.763096	-561.452533	-1.428236	0.681824	126	127
750	5	10_15_20_25	1.172240	1.936199	-717.642177	-1.438197	0.624665	92	118
1000	5	10_15_20_25	1.285379	2.043299	-686.842676	-1.461629	0.593177	79	126
1250	5	10_15_20_25	1.223030	1.952195	-696.539174	-1.461361	0.610211	92	127
1500	5	10_15_20_25	1.239804	1.946476	-710.259273	-1.463134	0.597288	106	113
1750	5	10_15_20_25	1.289992	1.972361	-735.060232	-1.466071	0.579692	92	123
2000	5	10_15_20_25	1.110877	1.707990	-662.719796	-1.417533	0.615136	91	139
2250	5	10_15_20_25	1.288989	1.910676	-740.433522	-1.471087	0.570072	85	137
2450	5	10_15_20_25	1.485695	2.136624	-990.687846	-1.526752	0.517009	79	118
500	5	10_15_20_25_30	0.993496	1.771992	-552.090416	-1.429452	0.681491	111	121
750	5	10_15_20_25_30	1.176318	1.945340	-710.175649	-1.440397	0.624203	88	121
1000	5	10_15_20_25_30	1.281534	2.041287	-686.796769	-1.460045	0.595250	84	118
1250	5	10_15_20_25_30	1.227171	1.956183	-694.771497	-1.462201	0.608939	91	114
1500	5	10_15_20_25_30	1.255692	1.964697	-722.344584	-1.465758	0.593430	105	130
1750	5	10_15_20_25_30	1.288255	1.969690	-738.386349	-1.466154	0.580903	89	121
2000	5	10_15_20_25_30	1.115251	1.711486	-679.049046	-1.418139	0.613697	102	136
2250	5	10_15_20_25_30	1.302911	1.926940	-738.101362	-1.476965	0.565949	83	120

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
400	2450	5_10_15_20_25_30	1.501226	2.157590	-1000.030647	-1.527096	0.515121	77	117
	500	15_30_45	1.006267	1.789217	-589.207549	-1.429973	0.676872	99	159
	750	15_30_45	1.172552	1.940428	-718.313197	-1.437191	0.624368	103	140
	1000	15_30_45	1.275164	2.036113	-693.114574	-1.459295	0.597688	110	143
	1250	15_30_45	1.222991	1.956719	-711.993673	-1.459450	0.609251	90	146
	1500	15_30_45	1.247801	1.957247	-720.390848	-1.462282	0.596067	93	142
	1750	15_30_45	1.230666	1.898336	-717.552402	-1.453035	0.592114	112	141
	2000	15_30_45	1.172265	1.778159	-663.581205	-1.433875	0.600167	103	150
	2250	15_30_45	1.359884	1.994936	-825.647037	-1.493677	0.550640	105	144
	2450	15_30_45	1.529069	2.197587	-996.304256	-1.528850	0.512690	87	132
	500	5_10_15	0.999686	1.776012	-562.539867	-1.429132	0.679385	135	132
	750	5_10_15	1.172492	1.934394	-739.096562	-1.439717	0.624141	105	123
	1000	5_10_15	1.286542	2.052724	-687.056843	-1.461520	0.595511	101	133
	1250	5_10_15	1.236884	1.968486	-708.416929	-1.462599	0.606027	108	133
	1500	5_10_15	1.208131	1.915002	-707.948111	-1.455927	0.606880	107	139
	1750	5_10_15	1.279011	1.957997	-727.735575	-1.467372	0.581491	110	131
	2000	5_10_15	1.165458	1.773186	-660.757362	-1.432848	0.602294	124	137
	2250	5_10_15	1.284484	1.903528	-752.656123	-1.471130	0.569931	111	138

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.494674	2.153099	-987.053079	-1.526349	0.515929	87	129
500	5	10_15_20	0.992971	1.765057	-560.873826	-1.428324	0.680215	118	156
750	5	10_15_20	1.163216	1.922944	-742.253211	-1.439075	0.626787	109	145
1000	5	10_15_20	1.286553	2.050947	-684.600273	-1.462306	0.594950	121	122
1250	5	10_15_20	1.233074	1.965211	-700.884475	-1.461115	0.607499	99	150
1500	5	10_15_20	1.208227	1.914205	-712.606336	-1.456419	0.606199	103	134
1750	5	10_15_20	1.281000	1.958750	-723.458004	-1.467182	0.580465	92	155
2000	5	10_15_20	1.164931	1.771006	-655.148058	-1.432720	0.602691	112	159
2250	5	10_15_20	1.295491	1.918005	-751.138897	-1.473197	0.567554	121	133
2450	5	10_15_20	1.494326	2.152529	-1003.203474	-1.526329	0.516291	105	123
500	5	10_15_20_25	0.994490	1.767327	-562.295573	-1.428061	0.679446	135	151
750	5	10_15_20_25	1.157333	1.917457	-710.532570	-1.437254	0.628050	107	137
1000	5	10_15_20_25	1.285701	2.049133	-677.095188	-1.461137	0.595603	96	141
1250	5	10_15_20_25	1.235805	1.970611	-697.934033	-1.461096	0.606739	106	138
1500	5	10_15_20_25	1.219809	1.925701	-719.922833	-1.459091	0.603229	124	128
1750	5	10_15_20_25	1.273198	1.947129	-729.205173	-1.465161	0.581331	103	141
2000	5	10_15_20_25	1.166375	1.772378	-649.446603	-1.431342	0.602346	100	160
2250	5	10_15_20_25	1.305248	1.929662	-773.114566	-1.477233	0.564977	101	152

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
	2450	5_10_15_20_25	1.500059	2.157395	-998.501767	-1.528348	0.514957	92	130
	500	5_10_15_20_25_30	0.997460	1.775579	-559.137588	-1.429258	0.679327	123	141
	750	5_10_15_20_25_30	1.162211	1.927841	-708.404870	-1.437920	0.627599	100	136
	1000	5_10_15_20_25_30	1.285159	2.049190	-682.992978	-1.461390	0.595865	100	133
	1250	5_10_15_20_25_30	1.237177	1.970885	-712.897715	-1.460755	0.605932	102	129
	1500	5_10_15_20_25_30	1.231517	1.939733	-726.526805	-1.460602	0.600496	112	154
	1750	5_10_15_20_25_30	1.266074	1.939052	-727.524708	-1.463315	0.584158	106	137
	2000	5_10_15_20_25_30	1.164951	1.771410	-654.316970	-1.431029	0.602398	114	149
	2250	5_10_15_20_25_30	1.318130	1.944138	-779.884582	-1.481926	0.561155	92	138
	2450	5_10_15_20_25_30	1.506106	2.166651	-1013.086260	-1.527307	0.514453	84	135
450	500	15_30_45	0.999527	1.778318	-603.698184	-1.429524	0.677770	114	181
	750	15_30_45	1.172026	1.936793	-716.351771	-1.438642	0.624198	117	158
	1000	15_30_45	1.283345	2.047470	-725.794792	-1.459411	0.596111	121	165
	1250	15_30_45	1.234714	1.966340	-733.059327	-1.462852	0.604326	100	167
	1500	15_30_45	1.246196	1.956764	-736.654849	-1.461712	0.595898	107	156
	1750	15_30_45	1.215417	1.874497	-682.382284	-1.450150	0.595800	121	164
	2000	15_30_45	1.171219	1.778846	-690.985386	-1.439049	0.598412	111	175
	2250	15_30_45	1.366266	2.004410	-832.393351	-1.495770	0.548511	116	162

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	15_30_45		1.540135	2.211402	-964.815437	-1.530186	0.510491	94	144
500	5_10_15		1.000276	1.775212	-570.792393	-1.428702	0.678693	148	148
750	5_10_15		1.164776	1.923228	-747.700465	-1.439142	0.625282	115	148
1000	5_10_15		1.288381	2.055187	-708.270572	-1.461760	0.594811	113	148
1250	5_10_15		1.242029	1.974952	-727.730924	-1.462835	0.604052	117	152
1500	5_10_15		1.224620	1.930165	-712.222459	-1.458504	0.601925	118	156
1750	5_10_15		1.240310	1.910772	-707.425835	-1.459268	0.590776	127	150
2000	5_10_15		1.183051	1.795884	-674.430519	-1.437645	0.598374	137	153
2250	5_10_15		1.302592	1.927455	-794.307427	-1.473775	0.566158	125	157
2450	5_10_15		1.502995	2.166834	-969.061986	-1.526539	0.516165	98	147
500	5_10_15_20		0.993875	1.765834	-566.691670	-1.427666	0.679688	136	174
750	5_10_15_20		1.159224	1.916193	-743.253837	-1.438100	0.626830	124	162
1000	5_10_15_20		1.290135	2.055921	-708.760112	-1.462369	0.594158	132	144
1250	5_10_15_20		1.236572	1.968415	-714.123710	-1.462003	0.605252	108	173
1500	5_10_15_20		1.225128	1.930896	-719.591724	-1.458852	0.601704	121	149
1750	5_10_15_20		1.244908	1.913417	-698.212898	-1.458436	0.589915	105	177
2000	5_10_15_20		1.180632	1.792107	-672.600966	-1.437250	0.599230	125	173
2250	5_10_15_20		1.309444	1.936014	-802.675492	-1.475487	0.564036	131	153

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20	1.505142	2.168492	-965.906169	-1.526563	0.515763	114	137
500	5	10_15_20_25	0.997261	1.770626	-575.473710	-1.428116	0.678632	150	171
750	5	10_15_20_25	1.159203	1.916999	-725.799162	-1.437523	0.626529	125	152
1000	5	10_15_20_25	1.293439	2.056898	-708.839448	-1.462673	0.593146	109	158
1250	5	10_15_20_25	1.236799	1.969478	-718.035739	-1.462559	0.604694	119	154
1500	5	10_15_20_25	1.227962	1.934934	-723.008738	-1.459561	0.601014	134	144
1750	5	10_15_20_25	1.240889	1.906214	-700.412901	-1.458048	0.589592	116	165
2000	5	10_15_20_25	1.176382	1.789302	-675.114568	-1.434585	0.600300	114	175
2250	5	10_15_20_25	1.322275	1.952683	-801.851925	-1.479715	0.561140	113	166
2450	5	10_15_20_25	1.512729	2.174911	-956.550686	-1.529396	0.513871	102	145
500	5	10_15_20_25_30	0.997400	1.775702	-569.663562	-1.428412	0.679220	138	159
750	5	10_15_20_25_30	1.166384	1.928292	-722.471048	-1.439372	0.625288	107	158
1000	5	10_15_20_25_30	1.292165	2.056050	-707.032329	-1.462545	0.593638	111	147
1250	5	10_15_20_25_30	1.241588	1.974557	-723.895973	-1.462554	0.603399	114	149
1500	5	10_15_20_25_30	1.236609	1.944922	-728.473528	-1.460917	0.598748	125	174
1750	5	10_15_20_25_30	1.232308	1.895614	-695.488547	-1.456256	0.592212	126	157
2000	5	10_15_20_25_30	1.170692	1.781511	-676.475113	-1.434680	0.600746	123	168
2250	5	10_15_20_25_30	1.329825	1.961176	-802.563285	-1.483739	0.558215	113	152

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
500	2450	5_10_15_20_25_30	1.525693	2.190673	-968.034974	-1.530219	0.511468	91	154
	500	15_30_45	1.003860	1.783121	-598.455081	-1.430377	0.676417	127	202
	750	15_30_45	1.182149	1.949456	-721.149239	-1.441832	0.622169	129	175
	1000	15_30_45	1.240127	2.000143	-711.614552	-1.456160	0.607780	134	190
	1250	15_30_45	1.251528	1.984559	-741.262985	-1.466726	0.599737	111	185
	1500	15_30_45	1.252617	1.964543	-713.068213	-1.461650	0.595009	122	167
	1750	15_30_45	1.163691	1.810748	-656.640681	-1.433268	0.608998	141	185
	2000	15_30_45	1.192097	1.802263	-688.222492	-1.445674	0.593797	120	194
	2250	15_30_45	1.401071	2.045905	-845.025705	-1.501595	0.540820	131	172
	2450	15_30_45	1.546524	2.222936	-954.528388	-1.529078	0.509827	101	157
	500	5_10_15	1.000672	1.775279	-585.874903	-1.428318	0.678012	167	162
	750	5_10_15	1.174124	1.933820	-729.842386	-1.440118	0.623280	130	162
	1000	5_10_15	1.275999	2.042256	-727.811903	-1.461322	0.598189	126	171
	1250	5_10_15	1.259280	1.990962	-766.485579	-1.466463	0.598268	126	172
	1500	5_10_15	1.232461	1.941322	-702.429296	-1.458755	0.600158	130	176
	1750	5_10_15	1.218870	1.881911	-673.846362	-1.454753	0.595416	147	165
	2000	5_10_15	1.192608	1.805656	-692.302117	-1.444125	0.594656	150	168
	2250	5_10_15	1.332997	1.964654	-798.319349	-1.481043	0.558525	139	171

Continued on next page



TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15	1.521165	2.189246	-956.945824	-1.527951	0.513039	108	157
500	5	10_15_20	0.996833	1.767818	-583.328464	-1.427680	0.678315	149	193
750	5	10_15_20	1.169432	1.926131	-732.951444	-1.440701	0.624027	136	178
1000	5	10_15_20	1.272719	2.037206	-725.524251	-1.461708	0.598682	147	159
1250	5	10_15_20	1.251278	1.981806	-756.954103	-1.465633	0.600219	121	189
1500	5	10_15_20	1.234078	1.941893	-706.067276	-1.460016	0.599064	135	168
1750	5	10_15_20	1.215934	1.874975	-671.019822	-1.453090	0.595637	118	196
2000	5	10_15_20	1.193193	1.805532	-685.291083	-1.443624	0.595485	138	189
2250	5	10_15_20	1.338484	1.971922	-809.143372	-1.481993	0.557199	141	168
2450	5	10_15_20	1.516373	2.183911	-951.537216	-1.526412	0.514161	122	154
500	5	10_15_20_25	0.995290	1.765190	-584.418517	-1.427930	0.678173	170	186
750	5	10_15_20_25	1.166645	1.925571	-718.414778	-1.439983	0.624910	139	170
1000	5	10_15_20_25	1.268464	2.030482	-719.746151	-1.460244	0.599831	121	176
1250	5	10_15_20_25	1.251035	1.984003	-745.812097	-1.465449	0.600409	130	170
1500	5	10_15_20_25	1.239391	1.946627	-708.998569	-1.462271	0.597481	146	161
1750	5	10_15_20_25	1.206401	1.861678	-670.816302	-1.449747	0.597365	138	182
2000	5	10_15_20_25	1.188123	1.799993	-685.904966	-1.441226	0.596418	133	195
2250	5	10_15_20_25	1.347470	1.983980	-814.493280	-1.485908	0.554917	122	187

Continued on next page

TABLE B.1: Execution Results and Metrics

Predictions	Days	Cutset	MAE	RMSE	R Squared	Log-Likelihood	Weighted Accuracy	Exact Prediction	Near Miss
2450	5	10_15_20_25	1.516362	2.183854	-946.249634	-1.527007	0.514380	111	160
500	5	10_15_20_25_30	1.000525	1.776164	-582.484077	-1.429221	0.677437	154	176
750	5	10_15_20_25_30	1.172594	1.936522	-717.232988	-1.441371	0.624415	121	172
1000	5	10_15_20_25_30	1.261875	2.023666	-718.998029	-1.459552	0.601708	124	170
1250	5	10_15_20_25_30	1.253450	1.986978	-748.302426	-1.465262	0.599589	124	165
1500	5	10_15_20_25_30	1.251744	1.961588	-716.253209	-1.463692	0.594897	138	190
1750	5	10_15_20_25_30	1.193727	1.847542	-665.922985	-1.445291	0.601954	141	178
2000	5	10_15_20_25_30	1.183880	1.794923	-687.621372	-1.440808	0.597027	134	187
2250	5	10_15_20_25_30	1.359063	1.997024	-821.231622	-1.490161	0.551301	126	170
2450	5	10_15_20_25_30	1.530083	2.202827	-949.085000	-1.527199	0.512623	102	171

End of the table

# Bibliography

- [1] iShares Core S&P 500 ETF | IVV. URL: <https://www.ishares.com/us/products/239726/ishares-core-sp-500-etf>.
- [2] MarketWatch: Stock Market News - Financial News - MarketWatch. URL: <https://www.marketwatch.com/>.
- [3] Nasdaq: Stock Market, Data Updates, Reports & News. URL: <https://www.nasdaq.com/>.
- [4] Selenium with Python — Selenium Python Bindings 2 documentation. URL: <https://selenium-python.readthedocs.io/>.
- [5] Yahoo Finance - Stock Market Live, Quotes, Business & Finance News. URL: <https://finance.yahoo.com/>.
- [6] CNBC Investing, January 2012. URL: <https://www.cnbc.com/investing/>.
- [7] Ankur Ankan and Johannes Textor. pgmpy: A Python Toolkit for Bayesian Networks, April 2023. arXiv:2304.08639 [cs, stat]. URL: <http://arxiv.org/abs/2304.08639>, doi:10.48550/arXiv.2304.08639.
- [8] Adebisi A. Ayodele, Adewumi O. Adewumi, and Charles K. Ayo. Stock Price Prediction Using the ARIMA Model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112, March 2014. URL: <https://ieeexplore.ieee.org/abstract/document/7046047>, doi:10.1109/UKSim.2014.67.
- [9] Louis Bachelier. *Théorie de la Spéculation*. PhD thesis, University of Paris, 1900. URL: <http://www.numdam.org/item/10.24033/asens.476.pdf>.
- [10] Ray Ball. Anomalies in relationships between securities' yields and yield-surrogates. *Journal of Financial Economics*, 6(2):103–126, June 1978. URL: <https://www.sciencedirect.com/science/article/pii/0304405X78900260>, doi:10.1016/0304-405X(78)90026-0.
- [11] Marcos L. P. Bueno, Arjen Hommersom, Peter J. F. Lucas, Martijn Lappenschaar, and Joost G. E. Janzing. Understanding disease processes by partitioned dynamic Bayesian networks. *Journal of Biomedical Informatics*, 61:283–297, June 2016. doi:10.1016/j.jbi.2016.05.003.
- [12] David Maxwell Chickering. Learning Bayesian Networks is NP-Complete. In Doug Fisher and Hans-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer, New York, NY, 1996. doi:10.1007/978-1-4612-2404-4\_12.

- [13] Peter K. Clark. A Subordinated Stochastic Process Model with Finite Variance for Speculative Prices. *Econometrica*, 41(1):135–155, 1973. Publisher: [Wiley, Econometric Society]. URL: <https://www.jstor.org/stable/1913889>, doi:10.2307/1913889.
- [14] J. Contreras, R. Espinola, F.J. Nogales, and A.J. Conejo. ARIMA models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020, August 2003. Conference Name: IEEE Transactions on Power Systems. URL: <https://ieeexplore.ieee.org/document/1216141>, doi:10.1109/TPWRS.2002.804943.
- [15] Verdecchia Cristian. Application of Partitioned Dynamic Bayesian Networks for Stock Price Prediction, July 2024. URL: <https://github.com/crisgrin/PDBN-Stock-Prediction>.
- [16] Paul Dagum and Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, March 1993. URL: <https://www.sciencedirect.com/science/article/pii/000437029390036B>, doi:10.1016/0004-3702(93)90036-B.
- [17] Tiegang Duan. Auto Regressive Dynamic Bayesian Network and Its Application in Stock Market Inference. In Lazaros Iliadis and Ilias Maglogiannis, editors, *Artificial Intelligence Applications and Innovations*, pages 419–428, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-44944-9\_36.
- [18] Jeremy Eberhardt. Bayesian Spam Detection. *Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal*, 2(1), March 2015. URL: <https://digitalcommons.morris.umn.edu/horizons/vol2/iss1/2>, doi:10.61366/2576-2176.1024.
- [19] Eugene F. Fama. Random Walks in Stock Market Prices. *Financial Analysts Journal*, 21(5):55–59, 1965. Publisher: CFA Institute. URL: <https://www.jstor.org/stable/4469865>.
- [20] Eugene F. Fama. Market efficiency, long-term returns, and behavioral finance1. *Journal of Financial Economics*, 49(3):283–306, September 1998. URL: <https://www.sciencedirect.com/science/article/pii/S0304405X98000269>, doi:10.1016/S0304-405X(98)00026-9.
- [21] Nathalia Costa Fonseca and João Vinicius de França Carvalho. Analysis of financial contagion among economic sectors through dynamic bayesian networks. *Anais Do XLV Encontro Da ANPAD*, pages 1–16, 2021. URL: <https://anpad.com.br/uploads/articles/114/approved/ca8155f4d27f205953f9d3d7974bdd70.pdf>.
- [22] Helmut Herwartz. Stock return prediction under GARCH — An empirical assessment. *International Journal of Forecasting*, 33(3):569–580, July 2017. URL: <https://www.sciencedirect.com/science/article/pii/S0169207017300079>, doi:10.1016/j.ijforecast.2017.01.002.
- [23] O. Jangmin, Jae Won Lee, Sung-Bae Park, and Byoung-Tak Zhang. Stock Trading by Modelling Price Trend with Dynamic Bayesian Networks. In Zheng Rong Yang, Hujun Yin, and Richard M. Everson, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2004*, Lecture Notes in Computer Science, pages 794–799, Berlin, Heidelberg, 2004. Springer. doi:10.1007/978-3-540-28651-6\_118.

- [24] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, July 2009.
- [25] Kevin B. Korb and Ann E. Nicholson. *Bayesian Artificial Intelligence*. CRC Press, Boca Raton, 2 edition, December 2010. doi:10.1201/b10391.
- [26] Xisuo Liu. Stochastic Process and its Role in The Development of the Financial Market: Celebrating Professor Chow's Long and Successful Career. *Communications on Stochastic Analysis*, 13(3), September 2019. URL: <https://repository.lsu.edu/cosa/vol13/iss3/7>, doi:10.31390/cosa.13.3.07.
- [27] Yue Liu, Haoyuan Feng, and Kun Guo. The Dynamic Relationship between Macroeconomy and Stock Market in China: Evidence from Bayesian Network. *Complexity*, 2021(1):2574267, 2021. \_eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1155/2021/2574267>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/2574267>, doi:10.1155/2021/2574267.
- [28] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. John Wiley & Sons, New York, NY, 1st edition, 2018. URL: <https://www.wiley.com/en-ie/Advances+in+Financial+Machine+Learning-p-9781119482109>.
- [29] Nasdaq. S&P 500 (SPX) Historical Data | Nasdaq, February 2024. URL: <https://www.nasdaq.com/market-activity/index/spx/historical>.
- [30] Rangsan Nochai and Titida Nochai. ARIMA Model Forecasting Oil Palm Price. January 2006.
- [31] Shraddha Parab and Supriya Bhalerao. Choosing statistical test. *International Journal of Ayurveda Research*, 1(3):187–191, 2010. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2996580/>, doi:10.4103/0974-7788.72494.
- [32] Stuart J. Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Pearson, 2016. URL: <https://thuvienso.hoasen.edu.vn/handle/123456789/8967>.
- [33] Luciana S. Malagrino, Norton T. Roman, and Ana M. Monteiro. Forecasting stock market index daily direction: A Bayesian Network approach. *Expert Systems with Applications*, 105:11–22, September 2018. Publisher: Pergamon. URL: <https://www.sciencedirect.com/science/article/pii/S0957417418301854>, doi:10.1016/j.eswa.2018.03.039.
- [34] Santosh Kumar Sahu, Anil Mokhade, and Neeraj Dhanraj Bokde. An Overview of Machine Learning, Deep Learning, and Reinforcement Learning-Based Techniques in Quantitative Finance: Recent Progress and Challenges. *Applied Sciences*, 13(3):1956, January 2023. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute. URL: <https://www.mdpi.com/2076-3417/13/3/1956>, doi:10.3390/app13031956.
- [35] Jacob Schreiber. Pomegranate: fast and flexible probabilistic modeling in python. *The Journal of Machine Learning Research*, 18(1):5992–5997, January 2017.
- [36] Marco Scutari. bnlearn - Predicting new observations from a Bayesian network. URL: <https://www.bnlearn.com/examples/predict/>.

- [37] Marco Scutari and Jean-Baptiste Denis. *Bayesian Networks with Examples in {R}*. Chapman and Hall, Boca Raton, 2nd edition, 2021.
- [38] Ethan Z. Shen and Cole R. Winstanley. Modeling prediction markets with dynamic Bayesian networks. Technical report, Stanford University, 2019.
- [39] S&P Global. S&P U.S. Indices Methodology | S&P Dow Jones Indices, February 2024. URL: <https://www.spglobal.com/spdji/en/methodology/article/sp-us-indices-methodology/>.
- [40] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. The MIT Press, January 2001. URL: <https://direct.mit.edu/books/book/2057/Causation-Prediction-and-Search>, doi:10.7551/mitpress/1754.001.0001.
- [41] Standard & Poor's and MSCI. Global Industry Classification Standard, 2023. URL: <https://www.msci.com/gics>.
- [42] Stephen J. Taylor. Tests of the Random Walk Hypothesis Against a Price-Trend Hypothesis. *Journal of Financial and Quantitative Analysis*, 17(1):37–61, March 1982. Publisher: Cambridge University Press. URL: <https://www.cambridge.org/core/journals/journal-of-financial-and-quantitative-analysis/article/abs/tests-of-the-random-walk-hypothesis-against-a-pricetrend-hypothesis/C07C80CBE0B0F14665316C3E33AFDBAD>, doi:10.2307/2330928.
- [43] Richard H. Thaler. *Advances in Behavioral Finance*, volume 1. Russell Sage Foundation, 1993.
- [44] Chi-Chen Wang. A comparison study between fuzzy time series model and ARIMA model for forecasting Taiwan export. *Expert Systems with Applications*, 38(8):9296–9304, August 2011. URL: <https://www.sciencedirect.com/science/article/pii/S0957417411000352>, doi:10.1016/j.eswa.2011.01.015.
- [45] Xiaohua Wang, P.K.H. Phua, and Weidong Lin. Stock market prediction using neural networks: Does trading volume help in short-term prediction? In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 4, pages 2438–2442 vol.4, July 2003. ISSN: 1098-7576. doi:10.1109/IJCNN.2003.1223946.
- [46] Yahoo Finance. APIs - Yahoo Developer Network, February 2024. URL: <https://developer.yahoo.com/api/>.
- [47] Gili Yen and Cheng-few Lee. Efficient Market Hypothesis (EMH): Past, Present and Future. *Review of Pacific Basin Financial Markets and Policies (RPBFMP)*, 11:305–329, June 2008. doi:10.1142/S0219091508001362.
- [48] Yi Zuo and Eisuke Kita. Stock price forecast using Bayesian network. *Expert Systems with Applications*, 39(8):6729–6737, June 2012. URL: <https://www.sciencedirect.com/science/article/pii/S0957417411017064>, doi:10.1016/j.eswa.2011.12.035.

# List of Acronyms

- AIC** Akaike Information Criterion. [21](#)
- AR** Auto Regressive. [8, 9](#)
- AR-DBN** Auto-Regressive Dynamic Bayesian Network. [10](#)
- ARCH** Autoregressive Conditional Heteroskedasticity. [9](#)
- ARIMA** AutoRegressive Integrated Moving Average. [8, 9](#)
- ARMA** AutoRegressive Moving Average. [9](#)
- ATS** Automated Trading System. [6, 7](#)
- BIC** Bayesian Information Criterion. [21, 38](#)
- BN** Bayesian Network. [1, 3, 4, 8–11, 14, 15, 17, 20, 21, 27, 28, 37, 50, 74](#)
- CPD** Conditional Probability Distribution. [19, 24, 30](#)
- CPT** Conditional Probability Table. [15–19, 24, 35, 50, 62](#)
- DAG** Directed Acyclic Graph. [14, 15, 17, 19, 22](#)
- DBN** Dynamic Bayesian Network. [1, 3, 4, 8–11, 14, 17–19, 28, 29, 34–36, 38, 57, 60, 73](#)
- DJI** Dow Jones Industrial Average. [9, 11](#)
- EM** Expectation Maximisation. [10, 26](#)
- EMH** efficient market hypothesis. [2, 5, 6, 8](#)
- ETF** Exchange-Traded Funds. [32](#)
- GARCH** Generalized Autoregressive Conditional Heteroskedasticity. [8, 9](#)
- GBN** Gaussian Bayesian Network. [9](#)
- GDP** Gross Domestic Product. [9](#)
- GICS** Global Industry Classification Standard. [34, 61](#)
- HC** Hill Climbing. [37](#)
- HFT** High-Frequency Trading. [6, 7](#)

- HHMM** Hierarchical Hidden Markov Model. [10](#)
- HMM** Hidden Markov Model. [10](#), [14](#)
- IAV** Industrial Added Value. [9](#)
- MA** Moving Average. [8](#), [9](#)
- MAE** Mean Absolute Error. [46](#), [48](#), [55](#), [63](#), [65–67](#), [71](#)
- MAP** Maximum A Posteriori. [27](#)
- MLE** Maximum Likelihood Estimation. [25](#), [26](#)
- NASDAQ** National Association of Securities Dealers Automated Quotations. [32](#), [33](#)
- NSE** Nigeria Stock Exchange. [8](#)
- NYSE** New York Stock Exchange. [8](#), [33](#)
- OHLCV** Open, High, Low, Close, Volume. [8](#), [33](#)
- OLS** Ordinary Least Squares. [50](#)
- P/E** Price/Earnings. [11](#), [74](#)
- PC** Peter Clark. [22](#)
- PDBN** Partitioned Dynamic Bayesian Networks. [1–4](#), [8](#), [14](#), [28](#), [30](#), [34–36](#), [38](#), [40](#), [56](#), [60](#), [61](#), [70–72](#)
- PMF** probability mass function. [24](#)
- RMSE** Root Mean Squared Error. [46](#), [48](#)
- RWH** Random Walk Hypothesis. [2](#), [5](#)
- S&P** Standard and Poor's. [1](#), [2](#), [10](#), [11](#), [17](#), [30–32](#), [34–36](#), [61](#), [64](#), [65](#), [67](#), [71–73](#)
- TRIX** Triple Exponential Average. [10](#), [11](#)
- VWAP** Volume Weigthed Average Price. [13](#)
- YOY** Year Over Year. [13](#)