UNIVERSITY OF OSLO

Master's thesis

Forecasting the Oslo Stock Exchange All-Share Index with Deep Learning and Economic Data

A Machine Learning Regression Approach

Ulrik Johan Vedde Tranvåg

Informatics: Robotics and Intelligent Systems
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Autumn 2023

Ulrik Johan Vedde Tranvåg

Forecasting the Oslo Stock Exchange All-Share Index with Deep Learning and Economic Data

A Machine Learning Regression Approach

Supervisor:
Jim Tørresen

Abstract

Succeeding in accurately forecasting stock market indices is a sought-after capability for investors, traders, and policymakers. However, many financial researchers regard financial assets, including stock market indices, as unforecastable. Others have attempted to challenge these views, traditionally by fundamental or technical analysis of the indices' historical data. Within the technical approaches, there has been a growing interest in machine learning (ML), a field within computer science. Within the ML methodologies, deep learning (DL), which consists of complex model designs inspired by brain neurons, has lately attracted the most attention. Several studies where DL is used for forecasting stock market index developments present impressive forecasting accuracies, but they seldom benchmark against models based on financial theories such as the random walk and efficient market hypotheses.

This study proposes a framework to evaluate ML regression models against these financial theories. As benchmarks, the framework includes two random walk hypothesis-based models: the naive seasonal and naive drift. It is also examined if utilizing economic indicators increases forecasting performance. For experiments, the Oslo Stock Exchange, situated within the small, open, and oil-price-dependent economy of Norway, presents an interesting environment. The Oslo Stock Exchange all-share index is chosen as the target variable, and Norwegian economic data is gathered, resulting in a 72-feature dataset stretching back to 1988 with a daily frequency. Two state-of-the-art DL models are evaluated: long short-term memory networks and the temporal fusion transformer. The models are assessed by forecasting the target variable's next-day closing valuation and then compared against the benchmark models.

From the results, it is found that none of the DL models outcompete the random walk-based models. The most accurate DL implementation, the long short-term memory networks, has a 42,75 % higher error rate than the benchmarks. It is also found that including additional features beyond the target index's price history in the training data leads to decreased performance. Several factors may cause these results. There may be too much noise or irrelevant information in the additional data, causing the models to overfit. Stock markets are dynamic, and since the dataset stretches back to the 1980s, it may learn the forecasting model patterns that are outdated in today's market.

Foreword and acknowledgments

This thesis written on ML-applied stock market forecasting represents not just an academic endeavor but also a journey of personal and professional growth. Using ML models and economic data to forecast future valuations of stock and stock indices is an area that has long intrigued me, stemming from my long interest in programming, ML, economics, and finance. This journey has thus been immensely rewarding, allowing me to explore these four fields in depth.

I would like to express my deepest gratitude to my supervisor, Professor Jim Tørresen, who has allowed me to follow these interests when writing my thesis and whose expertise and guidance have been invaluable.

My greatest gratitude to my family for their unwavering love, patience, and encouragement. I am equally grateful to my partner and her family for such immense support and guidance while writing this work.

Thanks to my fellow students and friends for abundant valuable input and suggestions. Your companionship and intriguing discussions have greatly enriched my years at the University of Oslo.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This introductory chapter presents the motivation behind this thesis research and gives a general overview of its contents. The motivation is presented in Section 1.1, followed by the problem statements in Section 1.2. To highlight the focus of the thesis, Section 1.3 explains the delimitation and scope of the research. Thereafter, ethical considerations will be discussed in Section 1.4. The main contributions from the thesis research will be presented in Section 1.5 Finally, the full contents of the thesis are outlined in Section 1.6

1.1 Motivation

Stock market indices, such as the Standard and Poor's 500 (S&P 500) and the Oslo Børs all-share index (OSEAX), are useful tools for measuring the valuation of selected stock market segments and benchmarking asset allocation performance [77]. Additionally, such indices indicate their respective stock market states, reflecting their investors' confidence and thus may indicate their situated economy's overall health [62]. Stock indices may also function as investable assets traded as "index funds."

Succeeding in accurately forecasting stock indices may, therefore, be considered an important task for investors, traders, and policymakers. Viewing the possible future development of one's economy and its investors' sentiment may greatly aid in making informed decisions regarding asset allocation, risk management, and market trends. Not to mention the possibilities of using this knowledge in building trading strategies [55] to extract excessive risk-adjusted returns.

In recent years, there has been a growing interest in using machine learning (ML) techniques for time series forecasting, particularly in the context of forecasting stock valuations and economic indicators (econometrics), including stock market indices [77, 50]. Compared to traditional, explicitly programmed stock market forecasting models, ML algorithms have some advantages:

1. They easily identify trends and patterns that are not inherently apparent to humans [47].
2. They may provide an advantage when dealing with multidimensional data, as they can recognize relationships that haven't previously been appreciated [48].
3. ML algorithms can continuously and autonomously learn from and adapt to new information without human intervention [2].
4. They may finish tasks faster than humans and are unaffected by emotions [47].

Given these factors, the technology might add depth to traditional forecasting methods or even serve as a cheaper, less labor-intensive alternative with the potential for higher accuracy. Most recently, state-of-the-art ML techniques utilizing deep learning (DL), such as the long short-term memory networks (LSTM), have shown promise in improving the accuracy of such models [9, 57, 50]. LSTMs are well-suited for handling the large, noisy, and complex datasets associated with economic data, achieving good forecasting accuracies [32].

Recently, a new emergent DL model has been gaining attention. The temporal fusion transformer (TFT) is a novel deep learning attention-based architecture that makes accurate forecasts on time series data [52]. It is an extension of the popular transformer architecture, which has shown great success in natural language processing and other applications. The TFT is designed to capture patterns over different time scales or frequencies, making it well-suited for time series forecasting tasks. Within a financial forecasting application, the TFT has successfully been used to forecast and explain future stock valuation [40].

Indices representing smaller economies are underrepresented [50]. The Norwegian stock market, represented by the Oslo Stock Exchange (OSE), is a small, open, and oil-price-dependent economy, making it an interesting candidate to test further the capabilities of ML-based financial time-series forecasting.

The recent research literature on stock market forecasting presents many good results with high accuracies, including stock index forecasting applications. These studies present intricate, novel algorithms that outperform previous and more common ML forecasting models. In their introduction chapter, most studies tackle the problems faced with stock market forecasting, highlighting the financial theories of the random walk and the efficient market hypotheses, two commonly accepted frameworks that explain these difficulties [29, 59]. However, many do not include benchmarking forecasting models based on these financial theories [50]. Including these models as benchmarks may give a very accurate indication of the real-world performance of the main ML implementations. For example, the random walk hypothesis can be modeled with a naive baseline model, repeating the last observed value as its forecasts.

1.2 Problem statements

This thesis aims to develop and examine the forecasting capabilities of the two state-of-the-art DL models, LSTM and TFT, on long- and short-term future composite OSE valuation. The challenge addressed is demonstrating whether these models trained on different technical and economic indicators outperform a random walk-inspired forecasting model. The motivation is to research if the newly emergent DL technology can *challenge the random walk hypothesis* on the OSE. Summarizing the main goal, the following research question is formed:

> (RQ-1) *Are the state-of-the-art DL models **TFT** and **LSTM** capable of achieving higher accuracies than the **random walk model** when forecasting **long and short-term OSE valuations**?*

Furthermore, this thesis aims to research how including additional indicators to a univariate, historical valuation-only data series alters the selected DL models' performance in this environment. The motivation is to establish whether these models benefit from adding extra economic-based features to their training data. Additionally, if adding such supplementary data enhances performance, which economic and technical indicators have the most significant impact? This is summarized in the second research question:

> (RQ-2) *For the **TFT** and **LSTM** DL models forecasting future valuation of the OSE: To what extent does **including economic and technical indicators enhance performance** for these algorithms in this environment?*

During recent years, LSTMs have emerged as a dominant technique for financial forecasting [50, 9]. The TFT attention-based architecture has shown promising results forecasting noisy, tabular data [52], making it a good contender against the LSTM architecture in a financial forecasting setting. Except for Hu's Google valuation forecasting [40], there has been little research conducted on the TFT's performance at stock and stock-marked index forecasting. Benchmarking the two models in the OSE environment can give insight and further establish the TFT's capabilities on such data. Therefore, this thesis also seeks to establish if the TFT can be considered a dominant forecasting technique in this environment compared to the LSTM. This leads to the third research question:

> (RQ-3) *Is the **TFT** architecture capable of **outperforming** the **LSTM** when forecasting future valuations of the compound OSE?*

1.3 Delimitations

When tackling problems within the cross-disciplinary universe of ML, economics, and finance, various options for research and possible experiments are available. Therefore, some limits must be set for the scope of this thesis:

1. This thesis will not attempt to directly investigate the capabilities of the selected algorithms to generate excessive returns when deployed with trading strategies into the financial markets. The theories regarding the possibility of generating excess risk-adjusted returns with historical data are a complete field on their own, and researchers are divided on the matter. This thesis will stick to achieving high forecasting accuracy and disregard return generating and trading strategies, which is better suited for further research.
2. DL models are time-consuming and computationally expensive. Therefore, this study has been limited to a 1-day forecasting horizon only. Longer forecasting horizons are motivated as future work.
3. The data utilized for forecasting in this research is limited to variables that are easily accessible and with sufficient length and quality. Variables that do not fulfill these requirements are excluded, which means some central indicators, for example, the Chicago Board Options Exchange's volatility index (VIX) and the Norwegian consumer confidence index.

1.4 Ethical considerations

The field of asset forecasting ML applications raises several ethical concerns that must be addressed. Many of these relate not directly to this thesis but to the potential of further use of its technology. The following considerations may also apply to ML applied to finance and economics. These include:

Risk of malicious use: Actors with malicious intent can potentially utilize the methods researched in this thesis. For example, AI may be capable of learning and conducting trade manipulation on the asset market, harming other fair-playing individuals. It is important to be aware of this risk and motivate future work to investigate solutions to prevent such incidences.

Risk of faulty use: The technology research in this thesis can potentially be used erroneously, especially by actors with limited knowledge of the fields of ML and finance. Such an example is the overestimation of ML's capabilities, resulting in permanent capital losses through investments motivated by the technology. Related to this study, investors may try to use the model's forecasts as a net present value (NPV) calculation of the market and further invest in the forecasted index according to their price relative to this NPV. It is important to clarify the risks associated with such investing techniques. Additionally, it is important to highlight that even though an algorithm may produce accurate forecasts or outperform indices when modeling portfolios, it may underperform when put into real-world financial markets. This is due to costs and other frictions occurring in real-world investing that are hard to predict or account for in theoretical models.

Job safeguarding: Due to AI-driven financial technology, Wells Fargo estimates that about 200,000 banking jobs will be replaced by robots this decade [85]. It is important to consider the impact on these people's lives and to motivate further research into preventing this from becoming an excessive problem. This could be done through, for example, wealth distribution management through tax-financed re-education and through AI taxing.

Environmental impact: As mentioned in Section 1.3, DL models are quite computing resource intensive, thereby having a high demand for electricity and hardware built of rare precious metals. It is important to highlight the potential environmental impact the induced demand for these resources can have.

1.5 Main contributions

While conducting this research, this thesis provides several new contributions. These contributions are:

- Presenting a preprocessing, RF-FS feature selection, hyperparameter tuning, and performance evaluation framework for thoroughly evaluating stock index (and other financial assets) regression-based forecasting models against the random walk hypothesis using larger economic datasets.
- Provide the first identified attempt at introducing the TFT architecture at forecasting an OSE-based index.
- Prove that the financial hypotheses of RW and EMH remain uncontested in a 1-day OSEAX valuation forecasting environment.
- Present an optimal dataset composition, architecture, and training parameters for both the LSTM and TFT in such an environment.
- Gathered and built an extensive 1-day granularity, 72-feature economic dataset consisting of OSEAX and Norwegian economy-related indicators as features, spanning over 35 years.
- Provided evidence that including additional features in the covered forecasting approaches degrades performance: training on a series of the target OSEAX variable alone produces the best results.

1.6 Thesis outline

This section presents an overview of the thesis contents and structure. The thesis has been divided into 7 chapters. In addition, there are two appendixes at the end of the thesis: one for larger tables and one for additional information like data sources. The main summary of the results can be found under Section 5.4 on page 84. The chapters, appendixes, and their contents are structured as follows:

Chapter 1 - Introduction: Presents the thesis motivation, research questions, ethical considerations, and this overview of its contents.

Chapter 2 - Background and Related Works: Explains the central concepts on which this thesis is built, in addition to the most recent relevant research.

Chapter 3 - Methodology: Presents the implementations and other methods used by this thesis to produce the experiment and its results.

Chapter 4 - The Economic Dataset: Introduces the collected dataset, elaborates on the preprocessing and feature selection results, and then presents the final feature-reduced datasets used in the experiments.

Chapter 5 - Experiments and Results: Gives an overview of the experiments and presents their results.

Chapter 6 - Discussion: Discusses the results obtained from the experiments in the previous chapter.

Chapter 7 - Conclusions: Concludes based on the results and discussion by summarizing the thesis and formally answering the research questions. The thesis's main contribution and possible future work will also be elaborated.

Chapter 2

Background and Related Works

The purpose of this chapter is to provide the reader with an understanding of the theoretical and practical background relevant to the research topic of this thesis, including the history and evolution of related fields, key concepts and definitions, and current research and trends. Section 2.1 introduces the field of stock market forecasting and explains its most central concepts. Following that, Section 2.2 will give an introduction to the world of ML. Section 2.3 introduces DL and recent developments in related algorithms. Section 2.4 gives a short overview of feature selection methodologies. Finally Section 2.5 presents the most relevant recent research and trends within stock market forecasting applied ML.

2.1 Stock market forecasting

The field of *stock market forecasting* focuses on developing and employing different approaches to forecast stock or stock index valuations accurately. The main goal is to use historical and real-time data to foresee future market valuations, which can then be utilized through trading strategies by investors to achieve high returns [7]. Stock market forecasting is generally recognized as one of the most relevant but, at the same time, highly challenging fields within financial research. The latter is due to controversies about to which extent its goals are achievable, which is still being debated to this day [50].

2.1.1 Stock markets and exchanges

A *Stock* is a certificate representing ownership of a share of a corporation's assets (share capital). There are two main types of stocks: common stocks and preferred stocks [22]. Owners of stocks, called shareholders, also claim a proportion of the corporation's future profits. Profits are rewarded to the shareholders through paying out dividends [51]. These certificates can be traded, with the buyers and sellers often referred to as investors. *Stock markets* are places where stocks are bought and sold between investors. Most of these transactions take place on stock exchanges [75].

Stock exchanges serve as physical meeting places and communication facilities where investors can buy and sell their stocks and many other financial assets. Stock exchanges operate with specific rules and regulations, and most transactions are done digitally. When a stock is tradeable on a given exchange, it is referred to as "listed on" that exchange [75].

The *Oslo Stock Exchange* (OSE), or Oslo Børs, is the main stock exchange in Norway. The OSE also facilitates the trading of other financial instruments, such as bonds and derivatives. It was established by law in 1819, with trading commencing the year after [25]. The OSE lists various companies but is most heavily weighted in the energy sector. Other prominent sectors are maritime (energy service and shipping) and seafood [26].

2.1.2 Stock market indices

A stock market index is a measure used to track and compare the performance of a stock market or its segments. Investors and analysts keep a close eye on the indexes not only to monitor the economic activity but also to evaluate the performance of individual companies [75]. By comparing a stock's price history against a market index, investors can assess the performance of that stock against the market the index represents (most usually the stock's respective market). The same can be done for portfolios consisting of multiple stocks, whereof now the portfolio is compared against the market index. Figure 2.1 shows the daily closing prices of the MSCI world index, an index covering the 23 developed markets countries [65].

Stock market indices are computed from several selected stocks, often organized by the exchange in which they are listed or to a specific sector. Most commonly, they are calculated from the same-day total market capitalization of the selected stocks. [51] This makes stock market indices a tool for measuring the current and historic compound valuation of their market. Stock market indices can be nationally bound, representing the valuation of

Figure 2.1
MSCI world index daily closing prices between 1988 and 2022

their respective countries' stock markets. Two other commonly known examples of market indices include the S&P 500 and the Dow Jones industrial average. The S&P 500 tracks the performance of 500 large companies listed on US stock exchanges, while the Dow Jones Industrial Average measures the performance of 30 significant US industrial companies [75].

2.1.3 Traditional forecasting approaches

In the past, stock market forecasting has been explored with a mixture of economic theories, statistical analysis, human psychology, and financial understanding. In overview, there are two distinct traditional methodologies:

1. **Fundamental analysis**: A method focusing on evaluating the *intrinsic value* of stocks, which is the value that can be justified based on the assets, earnings, and dividends of the company. The intrinsic value is found by analyzing financial statements, industry trends, and economic factors, among others. Fundamental analysis is conducted quantitatively and qualitatively and is more frequently used over longer investment horizons [82].

2. **Technical analysis**: A technique that utilizes price data, trading volumes, and other asset pricing indicators to forecast valuations based on historical performance and patterns. The main difference between

technical and fundamental analysis is that technical analysis assumes that market prices result from the interplay between supply and demand driven by investor behavior. In other words, technical analysis focuses on market trends and patterns rather than the intrinsic value of the security and is traditionally used for shorter horizon forecasts [82].

Historically, the evolution of traditional approaches in stock market forecasting has been a journey from qualitative assessments towards more quantitative, data and model-driven methodologies, resulting in today's state-of-the-art forecasting techniques. These more recent approaches are presented in the next subsection.

2.1.4 Modern forecasting approaches

Recently, advancements in computer technology have opened a new era of stock market forecasting. Most previous studies have moved more towards applying statistical *time series forecasting*, a technical approach that analyzes time-ordered historical data to find patterns, trends, and cyclic behaviors [51]. These are then used to forecast the future values of a stock or stock market index. Models like ARIMA (Autoregressive integrated moving average) are often used for this purpose [16, 24]. From these techniques, combined with the most recent advancements in data analytics, computer science, and computational power, even more sophisticated and increasingly accurate forecasting models have spawned. The introduction of artificial intelligence has particularly led to these.

Machine learning in stock market forecasting

With the introduction of *artificial intelligence* (AI), an even bigger interest in time series forecasting applied to stock market forecasting has followed. The quantitative methodology of machine learning (ML) is a key contributor to this progress and has been gaining increased attention [50]. ML, more elaborately explained in Section 2.2, involves training autonomous, self-learning algorithms on historical data to make predictions on new, unseen data [8]. Unlike traditional, explicitly programmed quantitative methodologies, ML models are adaptable and recognize relationships that haven't previously been appreciated [48]. ML models are also better equipped to handle the chaotic, nonlinear, noisy, and complex data associated with stock markets. This enables them to make more accurate predictions [18], explaining their increase in popularity. Between 2000 and 2020, the number of published

articles regarding stock market forecasting applied ML has increased exponentially [50]. During this period, three ML methods stand out from the literature:

1. **Artificial neural networks (ANN)**: Models inspired by the interconnected neuron structure of brains. These models consist of layers of neurons (called nodes) [4]. ANNs are more elaborately explained in Subsection 2.2.4.
2. **Support vector machines (SVM)**: Models dividing input data into classes using a hyperplane [19]. They also exist as regressors called *support vector regressors* (SVR) [50].
3. **Fuzzy theory-based techniques**: Models incorporating principles from fuzzy logic to handle uncertainties [49]. Fuzzy logic extends classical (binary) logic: an event is not only true or false but can be partially true or false (for example, 0.8 true and 0.2 false).

Recent trends in modern forecasting approaches

Most recently, deep learning (DL), ensemble learning, and feature selection approaches have become more prominent topics within the field [50]. DL, to be discussed in Section 2.3, is a subset of ML and exceedingly capable of identifying hidden, nonlinear relationships from complex and noisy data [50]. Ensemble learning is a method in which multiple ML models are combined to improve the overall forecasting performance. Random Forest is an example of an ensemble learning method and will be discussed in Subsection 2.4.2. Feature selection, to be discussed in section 2.4, concerns reducing the variables (features) of a dataset to filter out irrelevant ones.

2.1.5 A challenging domain

The underlying premise in stock market forecasting is that past market data behaviors, coupled with current market conditions, can inform forecasts about asset valuations. However, it is a highly challenging domain due to the stochastic nature of financial markets, influenced by many factors such as economic indicators, political events, company performance, and investor sentiment, among others.

Chaotic systems

As defined by historian Yuval Noah Harari [36], *chaotic systems* may be partly attributed to the problems faced by stock market forecasting. These

systems are neither deterministic nor simple to predict, as they are affected by many factors. They can be divided into two forms:

1. **Level one chaotic systems**: Chaotic systems that do not react to predictions about themselves. The weather is an example of such a system. It is complex and hard to predict, but it does not get influenced or alter its course when metrologists present their weather forecasts.
2. **Level two chaotic systems**: Level two systems, on the other hand, are chaotic systems that are reactive to any prediction about them. Markets and politics are examples of such systems.

Stock markets fall under the type two category and its associated forecast-responding behavior. For instance, if a highly regarded stock analyst predicts that a stock valued at 100 NOK will be worth 200 NOK in a week, investors will immediately adjust their prices to reflect this new valuation. As a result, the stock is valued at 200 NOK seven days before the analyst's target date, with the remaining time only posing an uncertainty risk. This poses a challenge if the forecasting results, method, or data is publicly available and known. However, chaotic systems may only explain parts of the associated problems with forecasting financial assets. Additionally, a forecasting methodology, data, and results may be kept away from competing actors, preventing them from affecting the system.

Within finance, two important main hypotheses try to explain why precise forecasting of assets and achieving excess returns that are higher than the market average: the *random walk hypothesis*, outlined in Subsection 2.1.6, and the *efficient market hypothesis*, discussed in Subsection 2.1.7.

2.1.6 Random walk hypothesis

The Random walk hypothesis (RW) suggests that stocks and other financial assets' prices evolve randomly and are unpredictable [30]. This implies that historical price movements do not hold any predictive power over future price directions. Put simply: past prices cannot predict future prices [30]. More formally, the RW states that:

1. The future changes in the valuation of a stock are independent of its past price history.
2. A stocks price changes conform to a probability distribution.

Burton G. Malkiel popularized the RW in 1973 with his book *A Random Walk Down Wall Street*. He argues that attempting to "time" (buying and

selling at the right time) or in any other form of outperforming the market is a futile effort that often results in underperformance. This also applies to utilizing fundamental or technical analysis to forecast stock prices. Instead, Malkiel suggests that investors should invest in and hold a diverse stock index fund to follow the general market returns [60]. The RW is a foundational hypothesis upon which the *efficient market hypothesis* is built, which will be discussed in the following subsection.

2.1.7 Efficient market hypothesis

In 1970, Eugene F. Fama published a defining paper, establishing a central part of Financial theory, the *efficient market hypothesis* (EMH) [29]. EMH states that financial markets are *informationally efficient*, meaning all publicly available information is already reflected in the market price. Therefore, it is impossible to achieve returns higher than the average market through any systematic strategy consistently. In other words, future price changes are unpredictable, as the prices of financial assets already reflect all publicly available information. There is no ground for getting an informational advantage that gives a more accurate "fair value estimate" to a specific asset. Also, the asset prices are subjected to noise, making them "randomly walk" around their anchor point [30], as explained in the previous subsection.

The EMH can be divided into three forms:

1. **Weak form**: Prices reflect all information regarding past prices and returns.
2. **Semi-strong Form**: Prices reflect all publicly available information, including past prices.
3. **Strong form**: Prices reflect absolutely all information, both public and private.

In the weak-form environment, using an asset's price history to forecast its future valuation is impossible, making technical analysis ineffective in trying to beat or accurately forecast the market [30]. On the other hand, weak-form environments theoretically allow for fundamental analysis. This means it is viable to analyze assets using publicly available fundamental data like company-specific fundamentals, financial reports, and macroeconomic variables.

In the semi-strong form environment, there is no opportunity for the latter as the current market prices already reflect all publicly available informa-

tion. The only way of getting a hold of the market is through possessing and analyzing private, non-public data, also addressed as "insider information." In such an environment, any attempts at forecasting financial assets to achieve above-market returns are considered futile unless such insider information is possessed.

Lastly, the strong form of EMH prevents any investor or algorithm from achieving higher returns than the broader market. Since all information, both public and private, is reflected in the current stock price, not even insiders may achieve an advantage, and stock market returns are effectively non-forecastable. In such a case, using any form of analytics for stock market forecasting is considered ineffective.

Since its inception, both the EMH and RW theories have been widely debated among financial researchers and economists [81, 28, 20, 53, 60]. Even Fama himself has stated that it would be unexpected if private information would not bring benefits for an insider [29], thereby implying that the strong form of EMH is somewhat radical. The EMH and other traditional financial theories have proven well-suited to aiding in making calculated financial decisions, but they cannot explain stock market disruptions [46]. Examples of such disruptions are stock market bubbles, momentum, market overreaction, and market under-reaction.

2.2 Machine learning

Machine learning (ML) is a field within computer science that is fundamentally concerned with studying, developing, and applying algorithms that can recognize, learn, and improve without being explicitly programmed. ML is regarded as a subset of AI, as illustrated in Figure 2.2.

The field embodies many methods ranging from data analysis, pattern recognition, statistical modeling, and computational learning theories. ML's computational characteristic is to generalize experience obtained through training to output a hypothetically estimated target function. The goal is to obtain a target function that is generalized well and precisely predicts future outcomes unknown to the computer [61]. ML is generally categorized into three types: supervised learning, unsupervised learning, and reinforcement learning:

Supervised learning: When an ML model is categorized under supervised learning, it is trained using a dataset of *labeled* data. This dataset contains the input (training) data and a corresponding correct response. When train-

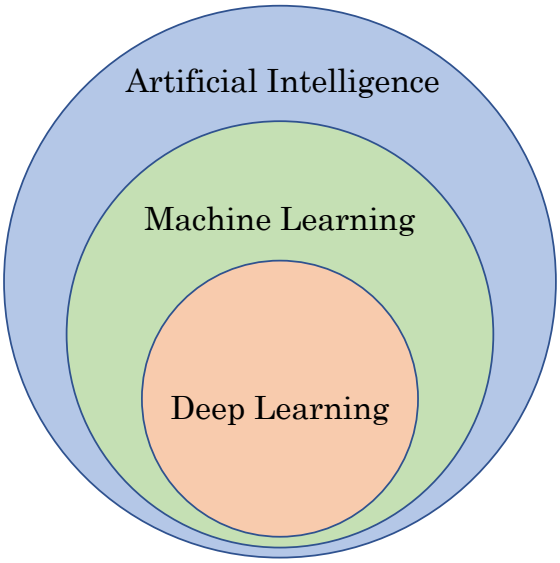

Figure 2.2
Illustration of the AI definition-hierarchy [56].

ing, the ML model's predictions are compared and corrected with the "true" labeled data [8]. The goal is to train, correct, and adjust the models, utilizing the labeled data, to predict the output for new, unseen inputs accurately. A properly trained model does this by generalizing to respond correctly to the input features [61]. Examples of supervised learning applications include image classification, speech recognition, and sentiment analysis [8]. Supervised learning is generally used in solving two types of problems [61]:

1. **Regression**: To find a mathematical function (model) that fits a curve as close to the target data points as possible.
2. **Classification**: Finding a model that most precisely assigns the input data to a predefined number of classes.

Unsupervised learning: Unlike supervised learning, unsupervised learning deals with unlabeled data, meaning correct responses are not provided. Unsupervised learning algorithms try to identify patterns and structures in the input data and to group instances without any "supervision" from prespecified labeled data. Such ML systems generally learn by rejecting pure unstructured noise [8] and categorizing inputs with common attributes [61]. Clustering and dimensionality reduction are common unsupervised learning techniques used in various applications [8], such as data compression and anomaly detection.

Reinforcement learning: Algorithms based on reinforcement learning (RL) learn by live-interacting with an environment to optimize a predefined cumulative reward or other goal. The learner (RL-agent) receives feedback through rewards and penalties triggered by observable changes in the environment state. The RL agent learns by trial and error by exploring and adapting a sequence of actions to maximize the reward [8].

Other ML learning categories exist, such as semi-supervised learning, transductive learning, and inductive inference [4].

2.2.1 Data

A robust ML implementation is dependent on sufficient amounts of high-quality data. From the literature, it is well established that an ML implementation's performance is upper bound by the data quality [45]. First of all, a sufficient amount of data is needed. More data may give a more comprehensive representation of the environment and potential underlying patterns. ML algorithms require significant amounts of data, but too much data may increase the computational cost too much [61]. Secondly, the data must be relevant to the problem and contain features that are potentially informative for predicting the target. Domain awareness is key here [61]. Additionally, there are five important quality dimensions to assess when ensuring quality data for ML tasks [12]:

1. **Consistent representation**: None of the dataset's features have identical values with different compositions (unique but semantically equal values) [12]. For example, both "25 years" and "25 y" convey the same meaning, but have different composition.
2. **Completeness**: No values are missing.
3. **Feature accuracy**: The accuracy of a feature in a given dataset indicates the extent to which its values match their corresponding true values [12].
4. **Target accuracy**: Absence of incorrectly valued data in the labeled target data; for example, differences in using commas and dots in floats.
5. **Uniqueness**: Ensuring unique features and avoiding redundancies.

To ensure these requirements are satisfied, proper *data preprocessing* and *feature selection* have to be performed as a part of the ML process pipeline [61].

2.2.2 Loss

Loss is the difference between a model's prediction output and the actual *true* target values in the training data. In essence, it measures the model's performance by quantifying how far off the model's predictions are from the true values. A *loss function* is a mathematical expression that calculates the loss as a single scalar value. An ML algorithm's main goal is to minimize this function. Mean squared error is an example of a widely used loss function.

Mean squared error

Mean squared error (MSE) is a common loss function used in regression problems within ML and statistics and is a measure of absolute error. It calculates the average of the squares of the errors between the predicted values by the model and the actual target values in the labeled dataset. The MSE is often used as a performance measure when doing time series forecasting [91]. A lower MSE value indicates that the model is accurate, while a higher value signifies poor performance. If a model's forecast is perfect, meaning no deviations between predicted and true values, the MSE will be zero. Mathematically, the formula for MSE is expressed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

where:

n = number of observations in the dataset
y_i = true value of the target variable for the ith observation
$\hat{y}_i$ = models target variable forecast for the ith observation

The MSE can be sensitive to outliers, as the squaring can give too much weight to outliers if they are present in the data. In addition, the MSE error value does not come in the same unit as the labeled data, which might be unintuitive for interpretation.

2.2.3 Overfitting

A common challenge within the field of ML is overfitting. When a model "overfits," it has learned noise and other unimportant details in the training data to the extent that it performs poorly on new, unseen data. Its internal

structure has become too specialized for the training data, making it unable to generalize. Overfitting can occur if the model is trained too long on the datasets, making its parameter (over)fit to the data [61]. Several techniques can help to prevent or mitigate overfitting, including:

- **Cross-validation**: Partitioning the data into three subsets: training validation and testing. The validation set is used to measure how well the model is generalizing.
- **Early stopping**: Stopping the training process when the model's generalizing performance measured on the validation set starts to degrade.
- **Dropout regularization**: Involves randomly freezing nodes within ANN-based models, forcing the neural net to learn additional routes to prevent node codependence.
- **Training on more data**: Increasing the amount of training data can, given it is diverse enough, help mitigate overfitting by providing a broader basis for learning.
- **Feature selection**: Selecting the features that best represent the problem while removing irrelevant or redundant features mitigates the overfitting risk.
- **Hyperparameter tuning**: Careful tuning of the model's hyperparameters, including the learning rate and the architectural complexity, can help avoid overfitting.

2.2.4 Artifcial neural networks

Artificial neural networks (ANN) is a type of ML algorithm consisting of many interconnected nodes, called neurons, organized into layers. Its design is inspired by the structure and function of biological neurons in brains [4]. The standard structure consists of an input layer, a "hidden" layer, and an output layer and is illustrated in Figure 2.3. All neurons have their own individual constant value, called "bias." Each neuron is interconnected with every neuron in its neighboring layers. These connections are referred to as "edges." Each connecting edge also has an individual weighting factor called "weights." The weights decide how much information is passed from the edges of the starting neuron to the next neuron. Both weights and bias can be tuned for each individual neuron. The goal of the ANN is to obtain the set of weights and biases that make the model predict as precisely as possible when given new unseen data, known as the test dataset.

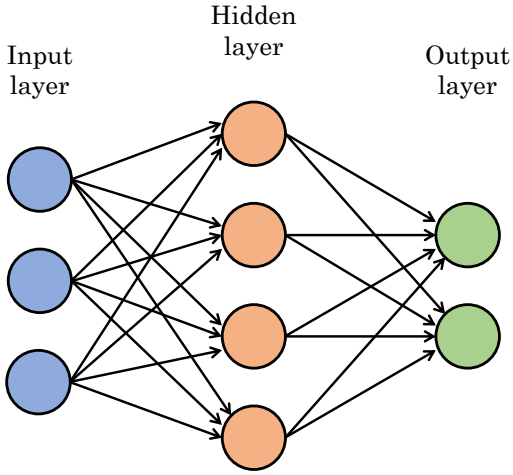

Figure 2.3
A simple ANN and the organization of its layers [13].

An ANN learns by passing information from the input layer through its hidden layers by the interconnected neurons. The value(s) resulting in its final output neuron(s) is the model's prediction. This is also called a "forward pass." In a supervised regression setting, the output value is compared with its corresponding true value from the labeled dataset. This is done through the loss function, as discussed in Subsection 2.2.2. The model wants to minimize the difference (error) between its output and the true value, or in other words, minimize its loss function. To achieve this, the model performs "backpropagation," or a "backward pass." This involves computing the partial derivatives of the loss function with respect to each weight, showing how much a change in weight would affect the loss. Essentially, it's a measure of responsibility: how much did each weight contribute to the error?

After computing the gradients, the network's weights are updated using a gradient descent optimization algorithm. The gradient descent adjusts each weight in the opposite direction of its gradient, which reduces the loss. The strength of the weight updates is controlled by a parameter called the *learning rate*. A smaller learning rate means the network will slowly adjust its weights, while a larger learning rate leads to faster learning but can overshoot the optimal values.

ANNs are the foundation of DL. When increasing the number of hidden layers, the definition of the model changes to *deep artificial neural networks* [4]. This will be further discussed in the next section.

2.3 Deep learning

Deep learning (DL) is a more advanced form of ML and is considered a subpart of the ML umbrella, as seen in the AI definition-hierarchy illustrated in Figure 2.2. As with the ANN discussed in Subsection 2.2.4, DL utilizes "hidden" layers between the input and output layers. The main difference lies in the number of hidden layers. DL models use *multiple* hidden layers, whereas traditional "shallow" ML only uses one or two. This structure makes them demanding, both computationally and in required amounts of data, to learn efficiently. This has limited the research and application of the technology [4]. Recently, as computing power has become more affordable, the technology has been utilized with great success in multiple fields. The "deep" version of ANN is referred to as *deep neural newtworks* (DNN).

2.3.1 Recurrent neural networks

Recurrent neural networks (RNN) are a subclass of DNNs adept at recognizing patterns in data sequences. By utilizing feedback loops, the RNN algorithm can use previous outputs as inputs to its current computation, allowing it to memorize past inputs [88]. They are highly suitable for tasks where previous information is required to generate accurate outputs. RNNs, generally DNNs, can suffer from *exploding or vanishing gradients*, making training deeper networks challenging [4]. When training on longer data sequences, this results in inaccurate results or even a total stall during the training process.

2.3.2 Long short-term memory networks

Long short-term memory networks (LSTM) are a more advanced form of RNN architecture. LSTMs solve the vanishing gradient problem by introducing a series of memory cells that can selectively store and forget information from previous time steps [39]. By utilizing gating mechanisms of input, forget, and output gates to control the flow of information, LSTMs are capable of tackling the vanishing gradient problem. This technique also enables it to handle long-term dependencies [4]. The gates decide what information to store, discard, and output, allowing LSTMs to maintain or forget information over extended sequences, making them capable of memorizing input information over longer periods. Figure 2.4 shows the architecture of the LSTM and how the three gates update the memory cell containing the long-term memory and the hidden state, which contains the short-term memory of the calculations from the previous time step.

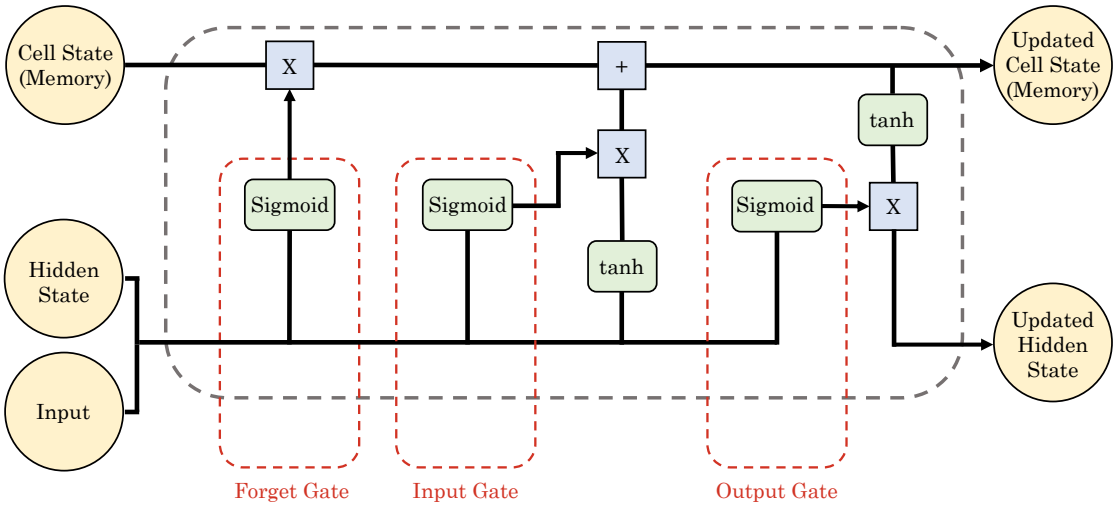

Figure 2.4
Overview of the LSTM architecture [21]

2.3.3 Transformer models

Transformers are attention-based models initially developed for natural language processing. By moving away from the recurrent structure of RNNs and LSTM, they have significantly improved the handling of sequential data. Using an encode-decode architecture combined with a self-attention mechanism, transformers can process input sequences in parallel rather than sequentially [90]. Compared to the LSTM, this architecture gives the transformer the capability to capture long-range (global) dependencies in data, as well as reduce its training times [91].

Within the transformer architecture, an encoder processes the input data while a decoder generates the outputted forecasts. The encoder and decoder are built of multiple self-attention and feed-forward neural networks. The central self-attention mechanism allows each value within the input sequence to focus on other parts of the sequence. This enables the model to learn dependencies between different values, regardless of their position of the sequence [90].

2.3.4 Temporal fusion transformers

Temporal fusion transformers (TFT) is a transformer-based technique that uses several mechanisms from other DL models. It utilizes the gated layers from LSTMs and the attention-based mechanism used in transformer models [52]. The TFT's architecture, which is depicted in Figure 2.5 on

page 24, is designed for handling time-series data. It fuses temporal (time-dependent) and static (time-invariant) data in a unified framework. TFTs employ the transformer-based architecture to capture temporal relationships, combined with the LSTM layers gating networks to fuse the temporal and static data effectively [52]. One key feature is the model's capability of taking in *future covariates*, which are features that can be known in the future. Examples are dates (it is known which dates will be occurring in the future), holidays, and, to some degree, the weather.

Like the original transformer model, the TFT uses an encoder to process input data and a decoder to generate forecasts. In addition, it introduces components like temporal self-attention, static covariate attention, and variable selection. These unique architectures enable the TFT to handle the intricacies of time series better. Like the original transformer, the temporal self-attention mechanism helps to learn long-term (global) dependencies in the time-series data. The LSTM gating networks aid in selectively using or ignoring certain features, improving the model's interpretability and performance in time-series forecasting tasks [52].

2.4 Feature selection

This section will briefly explain the workings and rationale behind the feature selection process. Subsection 2.4.1 will give an introductory rundown on the feature selection methodology. Thereafter, Subsection 2.4.2 will elaborate on the technique of random forest-based feature selection.

2.4.1 An overview

As discussed in Subsection 2.2.3, *feature selection* selects the most informative features from a dataset by removing redundant or irrelevant features [8]. The goal is to improve model accuracy, reduce the computational cost, and provide insight into the data-generating process while at the same time minimizing information loss [35]. Accuracy improvements are achieved by mitigating overfitting risk and defying the *curse of dimensionality*: avoiding the number of data dimensions and, thereby, the data mass from becoming too large. The risk of overfitting is mitigated by reducing the model complexity: fewer features result in a smaller input layer (input window). Feature selection is especially important in cases of high-dimensional data to alleviate the curse of dimensionality, and it's pivotal in domains where interpretability and model simplicity are essential.

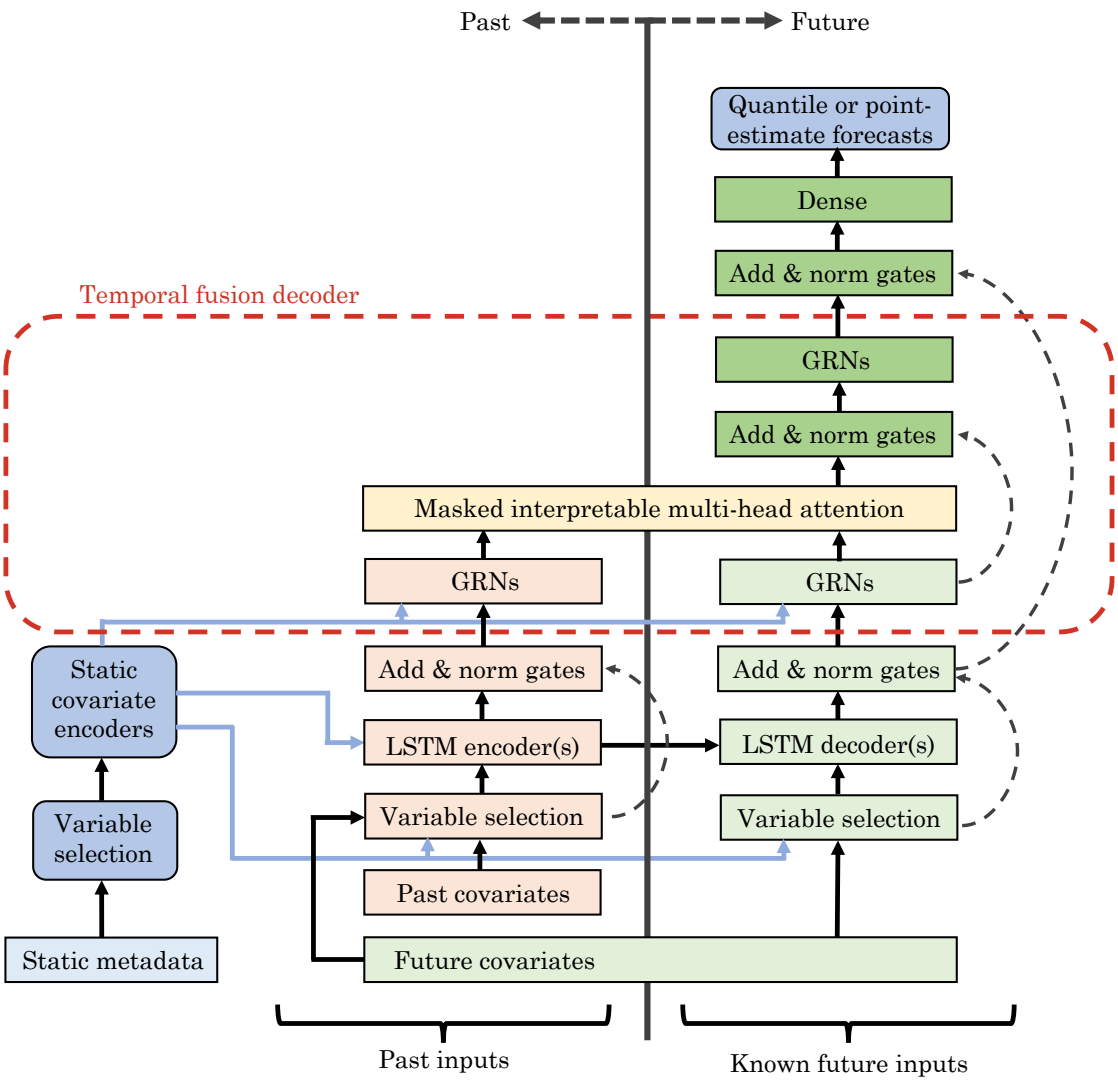

Figure 2.5
A simplified overview of the TFT model architecture [52]

Primary Methods

There are three primary methods for feature selection: filter methods, wrapper methods, and embedded methods:

- **Filter methods** evaluate features individually based on statistical characteristics, such as mutual information or correlation coefficients.
- **Wrapper methods**, on the other hand, assess feature subsets by training models and evaluating their performance. Recursive feature elimination is a popular technique used in wrapper methods.
- **Embedded methods** intertwine feature selection with model training. This means the model itself selects the most relevant features during training.

2.4.2 Random forest based feature selection

Random forest-based feature selection (RF-FS) is a method that uses the ensemble learning ML technique *random forest* (RF) to evaluate the importance of and then select features from a dataset. The RF is utilized for feature selection by using the by-products of its learning process (the feature importance) to identify which features are most influential in predicting the target variable. To achieve this, an RF regressor is trained on the data, and feature importances are derived from the model. Feature importance is determined by the average impurity reduction a feature brings to the trees in the forest, typically measured by the **gini impurity** or *entropy*. The *out-of-bag error* (OOB) can also be utilized to estimate feature importance by observing how the error changes as the values of each feature are permuted. Based on their importance scores, the features are ranked, and a subset of features can be selected. RF-FS is a versatile and reliable method for feature selection as it does not assume any specific form of data distribution [10].

2.5 Related works

This section presents the recent related works on which this thesis research has been inspired by and built upon.

An important study is the work of Kumbure et al. [50], which gives an extensive overview of the recent history and trends within the field of ML stock market forecasting. In addition, their paper presents a thorough overview of which types of features, forecasting models, and benchmarking techniques have been used in different research. From this research, it is noticed that few *regression* ML stock market *index* forecasting applications utilize financial theory-based baseline models. Studies attempting to achieve excess returns with trading strategies often use a "buy and hold" approach as a baseline. However, this is not as straightforward when using regression alone without attempting such strategies.

DL integrations gaining traction

As discussed in Subsection 2.1.4, the field of stock market forecasting is growing fast. Between 2000 and 2020, the number of yearly published articles regarding stock market forecasting applied ML has grown exponentially. This period has been dominated by ANN, SVM, and Fuzzy Theory-based models. In the past five years, "shallow" ANNs and SVMs have lost ground to DL models, feature selection processes, and classifier ensembles. DL models and feature selection techniques, in particular, have been rapidly gaining traction and becoming more prominent topics within the field [50].

2.5.1 LSTM applications

Some DL-based methodologies that have recently gained popularity are RNN-based models, particularly LSTMs [50]. LSTMs are not only popular as a standalone configuration: good results have also been achieved by utilizing different LSTM ensemble model configurations, particularly in combination with convolutional neural nets (CNN) and feature selection models [50].

In 2018, Fischer and Krauss did the first study utilizing LSTM for stock market forecasting [32]. By forecasting the out-of-sample directional movements of the S&P 500 constituent stocks, the authors attempted to achieve excess risk-adjusted returns. The LSTM implementation outperformed all the following benchmark models: Logistic Regression, standard DNNs, and RF. The LSTM also shows resilience during market friction, which are periods of financial turmoil and stock market downturns. A highly interesting

finding is that the LSTM's trading performance statistically challenged the semi-strong EMH until 2010. Past this, excess returns seem to have been arbitraged away, meaning that all possibilities of using LSTMs and similar models for buying and selling stocks to generate an above-market average income have already been exploited by others. The latter may indicate that ML implementations or other more efficient quantitative methodologies have been increasingly deployed to the S&P 500 stock market around this time.

The same experiment has been conducted on the OSE: In the same year, Lund and Løvås attempted a similar approach to the methodology developed by Fischer and Krauss on the OSE [57]. When experimenting with various LSTM architectures coupled with historical valuations and some technical and economic indicators, the authors achieved promising results forecasting next-day out-of-sample directional movements of stock included in the *Oslo Stock Exchange benchmark index* (OSEBX). Their results proved the LSTM outperformed the benchmarked Logistic Regression, RF, and SVM models. However, when deploying their algorithms to achieve excess risk-adjusted returns, the results were unsatisfactory.

2.5.2 Transformer and TFT applications

Researchers have recently successfully applied transformer and TFT models to stock market forecasting problems.

Wang et al. applied a transformer model to forecast multiple stock market indices, including the S&P 500 and CSI 300 (300 larger companies in China) [91]. Their research shows that the transformer model outperforms all benchmarked models, including LSTMs and CNNs. The authors utilized a univariate dataset, as the models were trained on the indices' historical prices only.

Hu experimented with a TFT implementation for stock and index forecasting and compared it against an LSTM and an SVR [40]. The target variables were the stock price of Google and the valuation of the S&P 500 index. The data consisted of high, low, open, and close prices, in addition to traded volume and adjusted closing prices. When benchmarking the models' forecasting results, the TFT was found to be outperforming both LSTM and SVR contenders. Analyzing the TFTs encoder variable importance, it was found that the model highly favored the last five days' closing price before any other feature.

2.5.3 Utilization of wider economic datasets

Some studies have explored methodologies involving feature selection from larger, more diverse datasets. The authors of said studies emphasize the importance of including such approaches when conducting stock market forecasting.

Gaspareniene et al. achieved an accuracy of 97,68% (R^2) over one-month forecasts, using the Random Forest algorithm trained on carefully selected US. economic indicators. Utilizing manual statistical feature selection, they found the *3-month treasury bill*, *West Texas Intermediate* (a global oil price benchmark), and *personal savings* to be the most important variables. The study only examined one simpler DL algorithm, *deep learning feed-forward neural networks* (DL FFNN) - a DNN. Although outperformed by the RF, the dataset utilized was a smaller size of 2444 data points for each of the 4 variables. The authors claim that macroeconomic environments have a significant impact on their respective compound stock market [33], highlighting the importance of their inclusion. Additionally, the research also leaves room for experimenting with more sophisticated DL models on more extensive datasets.

Chapter 3

Methodology

This chapter will introduce and explain the methods used to produce the results of this thesis, presenting the chosen implementations and the reasons why they have been included. First, Section 3.1 will give an overview of the experiment design. Following, Section 3.2 will explain the data research and retrieval process. The preprocessing procedures of the collected data are presented in Section 3.3. Section 3.4 will elaborate on the feature selection procedure. The forecasting model implementations will be presented in Section 3.5. Thereafter, the training procedures will be explained in Section 3.6. Finally, the hyperparameter optimization search method will be elaborated in Section 3.7

3.1 Experiment design

To test the LSTM and TFT model's capabilities of forecasting the compound OSE valuation, a system allowing for this experiment must be developed. This section presents the environment in which this thesis experiments will be conducted.

3.1.1 Formalizing the problem-solving framework

In overview, the problems regarding the research questions in Section 1.2 can be formalized into a problem environment. In brief, this environment can be described as the following:

- **Time series forecasting**: The data is time ordered.
- **Supervised learning**: There is labeled target data available.

- **Regression**: Instead of classification, point estimation will be conducted.

Stock market data and associated features come as time-ordered data, meaning all values have an associated timestamp telling when the data is recorded. Stock market valuations are also a time-dependent forecasting problem, making this a *time series forecasting problem*. This time-ordered data can easily be used as a labeled target dataset by shifting it with H forecasting horizon-days. This makes the problem *solvable through supervised learning*, which is also the most common for stock market forecasting [50]. The latter point, regression, is implemented by choice in this thesis' methodology. Many stock market forecasting implementations use classification to forecast directional movements (for example, up, down, flat), but this method can be limited due to the lack of differencing between directional strength (how *much* does the valuation rise or fall) [50]. Therefore, the problem is solved as a *regression problem* in this experiment.

3.1.2 Target variable: Oslo Børs all-share index

To be able to time series forecast the valuation of the compound OSE with supervised learning regression models, a target variable and a corresponding labeled true-values dataset are required. To achieve this, the compound valuation of the OSE must be quantified into data that satisfy these requirements and can be used for model experiments.

Oslo Børs all-share index

The Oslo Børs all share index (OSEAX) is a stock market index reflecting the performance of all shares listed on the (Euronext) OSE. At the end of 2022, the index consists of 196 individual component stocks, with a composite market capitalization of 362.41 billion EUR. Since its inception in 1995, the annualized return has been 6.38 %. The 30-day volatility is 14.93 %. [74] Figure 3.1 shows the OSEAX's daily closing prices from 1988 to 2022. The price is denominated in index points, starting at 100 at the end of 1995 for the OSEAX.

The index is full market capitalization weighted, meaning each stock within is weighted relative to its total market capitalization compared with the index. This causes larger companies with higher market capitalization to have a greater impact on the index's fluctuations. For example, the oil and gas company of Equinor, the largest in Norway, makes up 29.57 % of the index at the end of 2022. This gives a more realistic representation of the

Figure 3.1
OSEAX index daily closing prices between 1988 and 2022

compound OSE valuation and the Norwegian economy in general.

The OSEAX also serves as an underlying recipe for funds, exchange-traded funds (ETFs), and other products, making a respective forecasting model's outputs applicable for integration into systems trading these related assets in future works. The index composition is revised daily for corporate actions: events initiated by companies that could impact their stock price, such as stock splits, dividends, or mergers. Additional semi-annual reviews are conducted in January and July [74].

Since the OSEAX represents the performance of all the stocks listed on the OSE, it is selected as the feature for representing the OSE valuation in this research. More specifically, the gross return version of the OSEAX is chosen (OSEAX GI). The GI (Gross return index) version includes any additional yields, such as dividends, which gives a more realistic projection of the index returns and its compound valuation, making the models trained in this thesis better suited for future works related to profit estimations. Other OSE indices exist, such as the OSEBX, but the OSEAX is wider in its representation.

3.1.3 Data granularity and frequency

For time series data, the **granularity** (frequency) is the time unit in which each element within the series is sampled (recorded) individually, and the time steps between each sample are fed to the forecasting model. For example, the price of a stock can be recorded each minute, hour, daily, and so on. A finer granularity provides more data samples (a longer dataset) but is more computationally expensive. In large, diverse datasets, it is common for different features to be updated at different intervals. In such cases, its underlying features will not update with the same frequency as the granularity. For example, some macroeconomic indicators are updated quarterly, while stock prices can be recorded every five minutes.

With this in mind, the data in this experiment will be processed to have a one-day granularity. This is the most standard practice within the field [50] and may provide the best common ground between the number of training data samples and granularity that fits the update frequency of stock market indices, its technical indicators, and economic data. It is also a compromise between common practices of studies utilizing DL, which often get good results from univariate five-minute-granularity data, and studies focusing on general ML trained on wider macroeconomic datasets, which often use monthly granularity [50, 33].

The daily granularity will be on a business daily frequency. A *business day* (b-day) is commonly every weekday, except for holidays or other days when financial institutions are required by law to be closed. This is also the most common practice [50]. With a b-daily frequency, the dataset gets even shorter as non-b-day samples are removed. This may improve performance, as including weekends and other days without new updates to the target variable increases computational cost without bringing any new useful information to the table (the stock exchange is closed). It can also make a false assumption of better performance, as metrics may look better on a full-week/all-day dataset because of many days with a static, easily predictable target variable.

3.1.4 Forecasting horizons

The forecasting horizon is the number of days ahead the model is trained to forecast. The labeled data is shifted by the horizon H (b-days) before being fed to the algorithm's loss function for comparison. Every horizon, dataset, and hyperparameter combination in this thesis experiment environment will require a uniquely trained model. Therefore, experimenting

with many time spans will be too time-consuming and computationally expensive. To reduce this cost, the forecasting experiments are confined to the horizon span of 1 b-day. A 1-day horizon is the most common practice in ML-based stock index forecasting [50], allowing cross-experiment comparison. It is also simple and allows for examining immediate, short-term fluctuations forecasts.

3.1.5 Software

The main programming language utilized in this thesis is **Python**, chosen due to its large extent of relevant ML libraries and its simple and efficient high-level language [89]. Mainly, three Python libraries will be used for training and optimizing models and organizing, analyzing, and interpreting the data. In overview, these are:

- **PyTorch**: PyTorch is a Python ML library combining efficiency and user-friendliness. It offers many different ML models [76].
- **Darts**: A Python library focusing on time series forecasting. Offers many different ML and non-ML models, tools, and objects for such projects. Most of the ML models are wraparounds of pytorch [38].
- **Optuna**: A hyperparameter optimization search Python library. Optuna offers tools for and simple integration of state-of-the-art optimization techniques [5].

3.1.6 System, hardware, and setup

When training forecasting models, sufficient computing resources are required. Computing ML models, especially for DL, are computationally expensive. In this thesis, computation will be conducted on different resources: a simple, local system and a high-performance computing cluster (HPC).

Local resource

For smaller, less computing-intensive tasks, a simpler local system is used. These are performed on a Lenovo Yoga Slim 7 laptop computer. Since this system does not have a CUDA-compatible NVIDIA GPU, only CPU resources are utilized. For heavier task, which involves all DL implementations, more power is required. Using the local system is insufficient for these models because of the computational time associated with DL models.

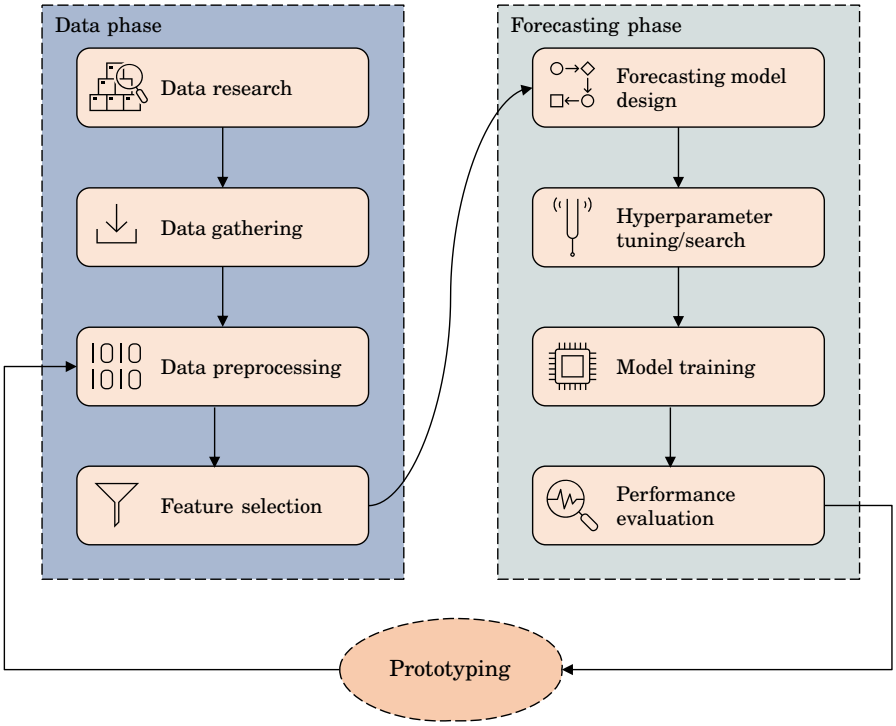

Figure 3.2
Overview of the experiment framework

Fox HPC cluster

The Fox HPC cluster is a powerful computing resource available for users of the Educloud Research infrastructure, a project-oriented self-service platform for research projects [73].

Fox offers different system configurations. In this experiment environment, the *ifi_accel* nodes will be used, as they are more available and the GPUs more compatible with the software.

3.1.7 An overview of the final experiment framework

With the general experiment environment defined, a general experiment framework is presented to provide an interpretable overview of the operation. This thesis experiment structure is inspired by the ML design process outlined by Marshland [61]. An overview of the process is illustrated in Figure 3.2. The system has been built through many iterations, as illustrated by the "prototyping" process in the figure. It has developed through six stages or versions. For each iteration, the following were the most im-

portant lessons that led to this final system:

1. Non-stationary data produced inaccurate results: some features, including the target variable, must be modified to become stationary.
2. Utilization of a daily frequency gives more training data but makes the results look artificially good and is not standard within the field: b-daily frequency is used instead.
3. Using the TFT for feature selection (originally intended) is too time-consuming: integrating other feature selection approaches, including an RF-based mechanism, instead.
4. Calculating and including technical indicators to the dataset, as they are known to be short-term indicators.
5. The TFTs perform badly under manual hyperparameter tuning. Therefore, an advanced automated hyperparameter optimization step is integrated into the framework, greatly improving the TFTs results.
6. Adding an additional RW-based forecasting model that includes long-term growth may be important for the framework for future works with longer forecasting horizons. Such a model is built and benchmarked.

3.2 Data research and retrieval

Data collection is a critical aspect of any research study. In this thesis, time series data is collected from various sources. This section will explain the process behind the research conducted to identify the desired data and the related selection criteria. It will also elaborate on the retrieval process. For a full overview of all the selected features, please see Table 4.1 under Chapter 4 on page 57.

3.2.1 Selection criteria

Before starting the research, some selection criteria are set to ensure relevance and delineate the available data. In overview, variables to be included in this thesis dataset must:

1. Be relevant to the OSEAX and the Norwegian economy.
2. Show a scientific potential to be utilized in stock market forecasting.
3. Be available publicly or through university resources.

4. Have a granularity between 1 day and one quarter.
5. Contain data back to 1988 and until the end of 2022.

Point 1 indicates that variables must be relevant to the compound OSEAX and the stock market in Norway, which means that the feature can potentially contain important information for future OSEAX valuations. This point is fulfilled if there is research, or generally known, that an indicator has or has a probability of correlating with the compound stock marked. Point 2 sets a boundary to only include features that have shown potential for relevance in ML stock market forecasting applications through research. Point 3 is there to ensure data is available for reproducibility. Point 4 is included to avoid having too infrequent updates within features in this thesis's daily granularity dataset. Point 5 is included for a number of reasons:

- **DL requires lots of data**: A longer data set will give the algorithms more data to train on, increasing the chances of learning important patterns and reducing the possibility of overfitting.
- **Ensuring environmental diversity**: Starting within the 1980s, the algorithms get a chance to learn patterns during high inflation and interest rate periods. Many research projects only train on the last 5 or 10 years of data [50], meaning they miss out on what may be multiple important economic periods.
- **Recency and relevance**: By waiting until the end of 2022, the most recent and relevant data can be collected, allowing the algorithms to be tested on as recent data as possible.
- **Availability**: As the oldest identifiable OSEAX valuation data starts on 03.01.1980 [71], it is desirable to gather data that is as close as possible to this date and, at the same time, not discard other important features that may be regarded as essential for this research. The requirement of having data back to 1987 is set as a compromise because multiple potentially important features do not have accessible data before this year.

With the selection criteria defined, the research to identify the desired economic indicator features can commence. In the next subsection, this process will be explained.

3.2.2 Research

The search for viable indicators revolves around reviewing literature research. Many papers have been written about technical and economic indicators and other stock market forecasting viable features. Kumbure et al.'s literature review paper on ML stock market forecasting techniques and associated data [50] has been a central piece when navigating this territory. The findings in Gaspareniene et al.'s macroeconomic indicator-centered approach discussed in Subsection 2.5.3 have also been considered. [33]. Findings in these studies have been projected onto the OSE and OSEAX environment, whereof the relevant and available sources will be listed in the next subsection. The resulting economic indicators chosen as features for the main full dataset will be outlined in Section 4.1. A full overview is provided in Table 4.1 on page 57.

3.2.3 Data sources

Due to the targeted length, diversity, and granularity of the features, the data set has to be collected from multiple sources. The most important sources are SSB [84], OECD [72], and Norges Bank [68]. For a complete list and introduction to the sources, please refer to the Appendix A.1 on page 113. Another central source for this research is Bern Arne Ødegaard's database [71], which contains OSEAX-index daily returns data all the way back from 1980. As the OSEAX we index know today was formally initiated at the end of 1995, most of the available dataset only contains data back to this year [74]. Thanks to Ødegaard's data, the OSEAX index valuation can be backward projected, giving potential for an additional 15 years of experiment data.

3.2.4 Retrieval

With the desired features and their respective sources identified, the data can be downloaded and organized for preprocessing. Since the sources are quite diverse in their data-delivering standards and procedures, it is decided to download and save the data as comma-separated values (CSV) files instead of using APIs and web scraping. The only exception is the date and time features, which are extracted through Darts API later. For a full overview of all the selected features, see Table 4.1 under Chapter 4 on page 57.

General data retrieval with CSVs

The data have to be consolidated into one dataset. This is done using Microsoft Excel. An Excel sheet is initiated and filled with all dates between 01.01.1980 and 31.12.2022. Each feature from each CSV file is then joined into this Excel sheet by date, with each feature having its own column. Some incomplete features must be extended or built from different sources to cover the desired time frame. Features built from multiple sources can be seen in Table 4.1 with multiple citations. With the data consolidated, the Excel file is converted into a Pandas data frame in Python.

Date and time feature data retrieval with Darts API

The next step in the retrieval procedure is to gather all date and time-related features. These are easily accessible through Darts' API, with which they are added to the data frame using the `add_datetime_attribute()` function. These features are grouped under the category "Time and date" at the end of Table 4.1, which starts on page 57.

Calculating features

The final data retrieval step involves calculating desired features that are obtainable by calculating them from the current dataset. One prominent group is the OSEAX technical indicators features, which can be computed from the retrieved OSEAX valuation feature. These are:

- OSEAX MACD histogram (MACD)
- OSEAX relative strength index (RSI)
- OSEAX 50d moving average (50d MA)
- OSEAX 200d moving average (200d MA)

Since no OSEAX volume data fulfilling the selection criteria were found during the data collection phase, no volume-based technical indicator features are included. In addition, the *10y/3m rate spread NOR* is calculated from the *10y G-bond* and *3m T-bill* features. After being computed, they are added as separate features to the data frame. The final Pandas data frame is ready for preprocessing and feature selection, which will be handled in the next section.

3.3 Data preprocessing

Due to the difference in the above sources' data values, the dataset must be preprocessed before it can be considered applicable for ML training. Data preprocessing transforms raw data into a clean, organized, and understandable format suitable for analysis. It involves cleaning, formatting, imputing (filling), and transforming the data to make it suitable for analysis. ML implementations learn much more efficiently when the data is preprocessed accordingly [61].

This study uses preprocessing techniques, such as forward-filling, data imputation, and normalization, to ensure these standards are met, enhancing performance and reducing the probability of overfitting. This section will elaborate on the preprocessing steps conducted in this research. The following subsections will present each step individually, ordered by how the preprocessing pipeline takes place. All preprocessing steps have been done using integrated Darts and Pytorch tools, in addition to Pandas, a flexible data analysis and manipulation tool for Python [63].

3.3.1 Forward filling and frequency handling

The first preprocessing step is to handle missing values and to set the data frequency for the data, now organized in a Pandas DataFrame. One problem that emerges when dealing with time series data is missing values. Missing values are a commonality and may be caused by multiple reasons. When dealing with stock indices, this problem materializes as missing data points during weekends and holidays, as stock exchanges are only open on working days. Many of the financial data sources that are utilized are incomplete due to downtime during recording, systems faults, or payment walls. Additionally, some values, such as the Norwegian GDP, are only set on specific dates, with months in between, leaving only a single data point for that period. To make the algorithm able to read the data set and to prevent fault during calculation or in the results, the lacking data has to be managed and standardized.

Forward filling imputation, trimming, and data type conversion

Forward filling is a technique that fills all empty values in a time series ordered dataset with the last observed value. It is also called "last observation carried forward (LOCF), as it is carrying on the last observed value until a new value is available. In the data frame, forward filling is conducted using

the Pandas `ffill()` method.

This method is chosen since it best represents the information gained from the financial markets in a realistic manner. The models will always have the latest, true recorded data points, in contrast to interpolation, which fills missing data with modified values. Forward filling is also chosen because it does not pose any threat of "data leaks" instances where the forecasting model gains access to information before its true occurrence date.

Ensuring b-daily frequency

Subsection 3.1.3 discussed the definition of b-daily frequency and why using it is attractive in stock market forecasting. To convert the forward-filled dataset to n-daily frequency, the following code snippet is used:

```
# 1. Get b-daily-range from the data frame's date column:
bd_range = pd.bdate_range(
    start=dataframe.index.min(),
    end=dataframe.index.max())
# 2. filtering dataframe:
dataframe = dataframe[dataframe.index.isin(bd_range)]
```

This script first extracts the b-daily data range as dates. Then, the data frame is filtered by those dates, effectively removing any date that is not a b-daily date. This method will not remove Norwegian holidays, only dates falling within weekends.

Removing empty elements and setting data type

With the dataset forward-filled and on b-daily frequency, there are still some empty elements that will need to be removed. This is because the ffill process does not fill any *forwardly* missing values, leaving columns of empty elements before the first recording for some features. These are removed by using the Pandas method `dropna()` on the data frame, which removes any empty values still present. Finally, the dataset is converted to a data type that is optimal for ML models: `np.float32`. This is done using the Numpy `astype(np.float32)` method on the data frame.

Adding a time buffer

Due to the uncertainty of the time of the announcement, all the macroeconomic indicators updated monthly or less frequently have been shifted backward by one to three months (22-66 b-days). This is quite a large buffer, but the thought is that the state of the macroeconomic indicators, although

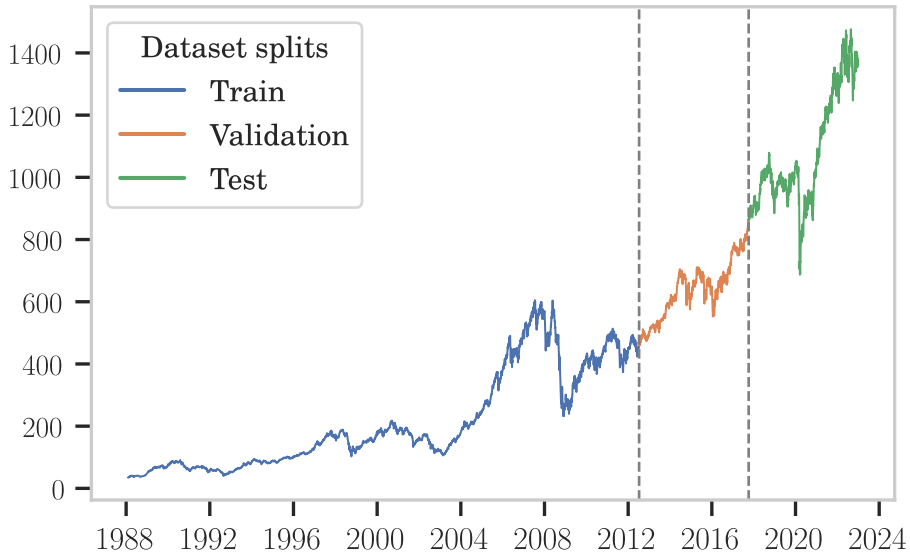

Figure 3.3
Lineplot visualizing the dataset splits on the target variable

lagged, can have informational value for the algorithm. This step is taken to prevent the algorithm from having a look-ahead bias by gaining information before it was publicly available and thereby performing better than what is possible. This may potentially harm performance, but in the first instance of this system built in this thesis, security precautions are taken. Using a non-buffered macroeconomic dataset is better suited for future work.

After this, the initial preprocessing steps are done. The dataset is now ready for outlier detection and handling, which will be elaborated in the next subsection.

3.3.2 Outlier detection

Outliers, also called anomalies, are values within the dataset significantly deviating from other values [3]. They are another common occurrence in datasets and can introduce noise and hamper performance. To detect any possible outliers, a visual inspection is conducted. The results from the inspection are presented later in the dataset chapter; Chapter 4 under Subsection 4.2.1

3.3.3 Splitting

To properly train, cross-validate, and test the forecasting algorithms, the dataset has to be split into partitions. A train-validation-test-split is carried out to achieve this, a common and important practice in ML. By using the Darts `split_after()` method, the dataset is split into the following three sets [61]:

- **(70 %) Training**: For training the forecasting model.
- **(15 %) Validation**: To monitor model performance while training.
- **(15 %) Test**: For final evaluation of model performance on unseen data.

A visualization of the dataset splits can be seen in Figure 3.3. The 70-15-15 split is chosen to ensure both the train and test periods include a "drawdown period," during which the OSEAX experiences a sudden decline. As seen in Figure 3.3, this split lets the training data experience multiple such periods, including the 2008-2009 financial crisis, while the test set falls within the timeframe of the 2020 COVID-19 recession. Preferably, the validation set should also include such a period, but that would have to be at the cost of a smaller training set or the exclusion of a drawback period in the test set. Both are undesirable. A smaller training set can hamper algorithmic performance while excluding the COVID-19 recession from the test set may lead to artificially good-looking results when evaluating forecasting performance.

3.3.4 Managing stationarity

Stationarity, in this case, *weak-form stationarity*, or “covariance stationarity,” are time series that have constant, mean, variance, and covariance between identically distanced periods [94]. When dealing with time series forecasting, ensuring that the data has stationary properties is important for some algorithms. If a feature is non-stationary, it can be differentiated (in an attempt) to make it stationary. To identify non-stationary indicators, an augmented Dickey-Fuller test is administered.

Augmented Dickey-Fuller test

The *augmented Dickey-Fuller test* (ADF) is a statistical test developed by David Dickey and Wayne Fuller used to determine whether a time series is stationary [23, 94]. The test checks if there’s a unit root in the time series, suggesting that the series is non-stationary. An ADF implementation is available from Python’s `statsmodels` library as the `adfuller()` function,

which is further utilized in this methodology. For experiment integrity, the test is only carried out on the training set. This is to avoid data leaks from the test set. The results from the test and the following feature (differentiating) modifications are presented later under Subsection 4.2.2 on page 59 under the Dataset chapter.

3.3.5 Normalization

Normalization is a technique used to change the values of numeric values of a feature in a dataset to a common scale without distorting differences in the ranges of values or losing information. This is important in datasets with features that vary in magnitudes, units, and range, something which this thesis' dataset does (to be elaborated in Chapter 4). Additionally, many ML algorithms, especially ANN-based ones, perform poorly when numerical input variables have different scales, making normalization necessary.

Normalization is performed on the training, validation, and testing data frame using the Darts `darts.dataprocessing.transformers.Scaler` object. This scaler sets all individual feature values between 0 and 1. First, the scaler is fitted to the training set using `Scaler.fit(training_data)` method. Fitting the scaler means adjusting its range to the feature value ranges in this data set. Then, the scaler is applied to all data set splits, transforming their values to the range set from the training data. This is done using the `Scaler.fit(<dataset_split>)` method. Some models will not work optimally with normalized data. Therefore, a pre-normalization instance of the data set splits is kept for later.

The dataset preprocessing procedure is now completed, and the data is ready for the next step in the experiment pipeline. This step is to perform data analytics, followed by feature selection, to be elaborated in the next section.

3.4 Feature Selection

This section will elaborate on the feature selection processes administered in this thesis. These selection processes are included for several reasons. A dataset of 72 features may be too much for an algorithm to handle. Training time may take too long, and overfitting is a threat. Reducing the amount of features through feature selection will result in a smaller and possibly more informative dataset. Three methods have been chosen: Manual selection based on research, the RF-FS method, and a hyperparameter tuning-integrated method.

3.4.1 Research-based manual feature selection

In this thesis, the manual research-based feature selection process will involve choosing a set of features based on their proven relevance or potential for relevance. To decide which features should be considered, relevant literature is analyzed. This feature selection methodology is included to provide a baseline against the other dataset instances. The resulting dataset instance is presented in Subsection 4.3.2 on page 62.

3.4.2 Random forest-based feature selection

Random forest feature selection (RF-FS) is the feature selection method based on the RF ML algorithms feature importance system discussed in Subsection 2.4.2. This method has been included due to its simple yet proven accurate way of selecting suitable features. RF-FS has been successfully utilized in many applications [55, 69, 37].

This thesis RF-FS methodology involves training an RF regressor on the training set and extracting the resulting features and feature-importances. For this, the `sklearn SelectFromModel` meta transformer class and the `RandomForestRegressor` class are utilized. The former extracts the features based on the feature importance results from training the latter. This is done by running the following script:

```
# Building selector (meta transformer) using RF feature importance:
selector = SelectFromModel(
    estimator=RandomForestRegressor(
        n_estimators=100,
        random_state=42,
        n_jobs=6,
        verbose=0))
# Training selector:
selector = selector.fit(
    X=dataset,
    y=target_series)
# Getting results:
seleted_features = selector.get_feature_names_out()
n_selected = len(seleted_features)
# Getting feature importances:
feature_scores = pd.Series(
    selector.estimator_.feature_importances_,
    index=dataset.columns).sort_values(ascending=False)
```

Due to the RF regressor's low computational complexity, the script is executed on the local system. As for hyperparameters, the number of trees in

the forest (`n_estimators`) is set to 100. A number 1000 was considered due to possible performance gains, but some research has found that RF models overfit for some noisy datasets [79]. To ensure reproducibility, the model seed is fixed by setting `random_state` to 42.

Two dataset instances are built according to these selected features and their importance ratings. The resulting datasets are presented in the upcoming Subsection 4.3.3 on page 62.

3.4.3 Hyperparameter optimization feature selection

As later explained in Section 3.7, this thesis will perform a hyperparameter optimization search on the LSTM and TFT models. Due to Darts' support of simple inline inclusion of date and time features during model initialization, a third form of feature selection will be performed. This will be executed as a part of the hyperparameter optimization search process, explained in Subsection 3.7, and will only regard the date and time features. The results from this hyperparameter selection process will not result in another dataset instance but will be included as part of the hyperparameters for the tuned LSTM and TFT.

Including this method will allow further experimenting if the models prefer to utilize date and time features. This is especially important for the TFT, which supports date and time features as future known inputs (future covariates). If the RF-FS method is unjustly disfavoring them, the features can show their real potential within this process. The results from the hyperparameter optimization search are presented under Section 5.1

Finally, with the final dataset instances' building methods explained, the implementation of the selected forecasting models will be presented in the next section.

3.5 Forecasting models

This section explains the inner workings, selection rationale, and integration of the selected forecasting models. As discussed in the Background section, the LSTM and TFT models pose promising candidates for this experiment. In addition, two RW-based models have also been selected to compare the DL models forecast against the RW. All models have been implemented using the Darts library in Python.

3.5.1 Random walk model

As discussed in Subsection 2.1.6, the RW states that prices develop independently of pricing history and are bound to a probability distribution. This implies that prices are non-forecastable. To test whether the DL models can challenge this, a forecasting implementation based on the RW is built: a *random walk model* (RWM).

Modeling random walks

Forecasting the outcome of a random walk is difficult due to its unpredictable nature. The best estimate that can be made is to use the observation from the previous time step as the expected outcome for the next time step. It's simple but surprisingly hard to beat; repeating the last day's value is an accurate approach for short-term forecasting. This type of forecasting is commonly referred to as a naive seasonal forecast [11], and it has been shown to be optimal when data follow a random walk pattern [42]. Preceding DL models should be capable of outperforming this model to be considered well-performing.

Take this example while assuming an equal probability distribution of price changes: If a stock has an equal chance of falling or increasing within a set symmetric range of NOK tomorrow, the two probabilities cancel each other out and only leave the last recorded price in the equation.

An RWM can be modeled as a naive seasonal forecast. Formally, the RWM can written as the following equation [67]:

$$\hat{y} = y_{t-H} \tag{3.1}$$

where:

$\hat{y}$ = models price forecast
y_{t-H} = price observed H forecasting horizon time step ago

From a trading perspective, this can simulate a buy-and-hold strategy. If fully invested in a financial asset, a forecast repeating today's value (the price stays flat) for the asset motivates one to hold it. Selling takes time and energy and also induces transaction costs. This is given that this example is only observed as a delimited case, excluding the possibility that the investor has other, better assets in his or her overview that are a better investment opportunity.

Implimentation

To implement the RWM, the Darts `NaiveSeasonal` model class is used. This class works as the naive seasonal methodology described above: the last observed value is used as the forecast. The model is univariate, considering only the OSEAX price history and no other features. It is evaluated on a non-differentiated OSEAX test set, as it works best when forecasting full index values. Using a differentiated OSEAX data set may falsely indicate worse performance.

3.5.2 Random walk with drift

An RWM implementation can also include a constant underlying growth. This is called a *random walk with drift* (RWD). In this setting, the model includes a drift component. The drift component represents a consistent upward or downward trend in the data, gradually and short-term unpredictably pushing the random walk in the trend direction. The following equation expresses this as [67]:

$$\hat{y} = y_{t-H} + \alpha \tag{3.2}$$

where:

$\hat{y}$ = models price forecast
y_{t-H} = price observed H forecasting horizon time step ago
α = drift component

In this case, it can be modeled as the underlying economic growth that, over time, lifts stock indices. This is useful for the longer-term one-year forecasts. Where the standard RWM will only repeat the same price recorded last year, the RWG does the same, but with an extra upward push by the growth trend, potentially improving forecast accuracy. Such a model is important for benchmarking the DL models' long-term forecasting capabilities.

Implimentation

Like the RWM, the RWD is implemented using the Darts `NaiveSeasonal` model. In addition, the mean H growth is added to the `NaiceSeasonal` forecast. The mean growth is calculated from all H time spans in the test set. It is evaluated on a non-differentiated univariate version of the OSEAX test set.

3.5.3 LSTM

In Subsection 2.3.2, the workings of the LSTM model were discussed. The model has been chosen based on its good performance in similar studies and has been described as ideal for learning advanced patterns from the immense and noisy data sets associated with economic data [50, 9, 32]. The LSTM technique has been deployed successfully for forecasting daily out-of-sample directional movements of the constituent stocks of the OSEBX index, a similar index to the OSEAX [57]. The model is included based on:

- Its proven performance makes it a benchmark for the TFT experiments to see if it can achieve good accuracies.
- It is a good benchmark to compare against the RWM and RWD models.

Implementation

The LSTM model is built using the `BlockRNNModel` class, a Darts implementation for training sliding window RNN models. To modify the class to train as an LSTM, the hyperparameter `model='LSTM',` is set when initiating the model object. The hyperparameters set as standard by Darts are used for the non-tuned version of the model. To allow the algorithm to capture inter-feature patterns, the input window size is set to 22 b-days, corresponding to a business month.

3.5.4 TFT

The TFT model, discussed in Subsection 2.3.4, is a state-of-the-art time series forecasting technique and is therefore included in this research. It has already shown promising performance forecastsing other financial assets [40]. No research applying the TFT to the OSE has been identified during the literature search. The TFT is included to observe if it can outperform the LSTM in this domain, further proving DL's potential in stock market forecasting applications.

Implementation

Implementation of the TFT is done using the Darts `TFTModel` class. This class is based on the paper published by Lim et al. [52]. Their research found a set of hyperparameters that were generally well-performing at multiple tasks. These hyperparameters are used for the standard non-tuned version of the TFT in the experiments, while the input window size is set to the same as for the LSTM: a sliding type of 22 b-days.

Table 3.1
DL models training settings

Setting	Value
Models trained per instance	30
Max epochs	1000
Stopper patience	50
Loss function	MSE
System	Fox HPC

3.6 Training and evaluation

This section explains the training procedure chosen for this experiment. Because of their different workings, the RW-based models and the DL models require different training procedures. The next subsections will describe these.'

3.6.1 RW models

The RW models are both simple and computationally inexpensive. Both are, therefore, trained on the local system. Additionally, the RW models are deterministic, meaning their forecasts will be the same for every same-model run. These models only take the univariate OSEAX data, and none require training. Therefore, only one instance is evaluated for each model on the OSEAX test set. Results are presented in Subsection 5.3.1.

3.6.2 DL models

DL models are stochastic, meaning the same model architecture will perform different forecasts for each unique instance. They also take all the different dataset instances and require training. All DL models are implemented using a fixed-size sliding window and outputting a single point H forecasting horizon b-days ahead of time. DL models are also quite computationally expensive, requiring more robust hardware. Therefore, a more sophisticated training procedure is designed and then uploaded and executed on the FOX HPC. For each DL algorithm, feature-reduced dataset, and hyperparameter set combination, training is conducted as follows:

1. **Train 30 models**: 30 model instances with different weights are trained for the DL algorithms.

(a) **Fixed seeds**: To ensure reproducibility, the 30 models are trained with the same 30 fixed seeds.

(b) **Early stopping**: During training, the validation performance is monitored. If the model shows no improvement in the validation loss after a certain number of epochs (stopper patience), the training is halted, and the next step is performed.

(c) **Backtesting**: After training, each model is evaluated on the training set by backtesting and calculating metrics, using the best model weights discovered during training.

i. **Conversion to full index values**: To compare the DL models with the RW models, the DL forecasts are converted to the same representation used by the RW ones: full, non-differentiated, non-normalized OSEAX index valuations denominated in index points.

2. **Mean metrics**: When all models have been trained, the mean metric scores are calculated from the 30 models.

Running 30 different models is performed to calculate mean forecasts, metrics, and standard deviation for each model instance. Early stopping is included to prevent overfitting and reduce the overall training time and computational expense. The max epoch, early stopper patience, loss function, and training system settings used for training are summarized in Table 3.1. The max epoch setting is chosen because DL models are known to require hundreds of epochs to learn the dataset patterns completely [91]. When evaluating the models on the test set, selected metrics are used. These are discussed in the next subsection.

3.6.3 Evaluation metrics

When evaluating the models' performances on the training set, three evaluation metrics are used. These are basically other types of loss functions, such as the MSE discussed under Subsection 2.2.2, which is used during training. They are quantitative methods of measuring the accuracy of the forecasting models. Multiple metrics are used to get a diverse picture of the models' performances. The metrics selection in this thesis is inspired by Bhandari et al.'s research [9] Additionally, these three are common in stock market forecasting regression applications [50]. The MSE is not used as a final evaluation metric due to its lesser-informational format.

Mean absolute percentage error

Mean absolute percentage error (MAPE) is a measure of forecasting accuracy, measured as a percentage. This percentage is the average deviation of the model's forecast from the true values. MAPE is defined by the following formula:

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{3.3}$$

where:

n = number of observations in the dataset
y_i = the true value of the ith observation
$\hat{y}_i$ = the models forecasted value of the ith observation

Root mean square error

Root mean square error (RMSE) measures the average difference between a model's forecasted values and the true values. RMSE is denominated in the unit of the forecasted value and is therefore included in this thesis. For instance, if a forecast of a NOK-denominated stock price has an RMSE = 2, it means that the model, on average, misses by NOK 2 from the true value. RMSE is defined by the following formula:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3.4}$$

where:

n = number of observations in the dataset
y_i = the true value of the ith observation
$\hat{y}_i$ = the models forecasted value of the ith observation

Coefficient of determination

The coefficient of determination, also known as R-squared (R^2), is a metric that measures the proportion of variance for a dependent variable (a forecasting model's output) explained by its independent values (the input features). It is commonly used in regression analysis, including regression ML models, where it measures the goodness of fit, which is how well the model

explains the forecasted value in relation to the true data. The R^2 score generally ranges from 0 to 1 but can also be negative if the forecasting model is severely flawed. An R^2 of 0 indicates that the model explains none of the variability in the data. An R^2 of 1 indicates that the model forecasts the target variable perfectly. Mathematically, the R^2 can be expressed as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{3.5}$$

where:

n = number of observations in the dataset
y_i = the true value of the ith observation
$\hat{y}_i$ = the models forecasted value of the ith observation
$\bar{y}$ = the mean of the true y_i values

3.7 Hyperparameter optimization search

There are many different parameters to set for the LSTM and TFT models. Some are related to architecture, like the number of nodes or hidden layers, while others are related to training, like learning rate and optimizers. The best combinations of hyperparameters for these models in the OSEAX forecasting environment are unknown. Therefore, a hyperparameter optimization search is conducted to explore the possible combinations.

3.7.1 Tuning machine learning models

The hyperparameter optimization search, from now on referred to as *tuning*, is the process of exploring a given range of hyperparameters to find the most optimal set for a given ML model in a given environment. Traditionally, tuning has been performed with *grid search*, explicitly defining all parameter combinations to be tested and systematically running through them. Although thorough, this method is slow and computationally expensive. In recent times, more sophisticated and efficient tuning methodologies have been developed. Bayesian optimization, in particular, is one of these. Most recently, the Bayesian optimization method of Tree-Structured Parzen Estimators (TPE) has gained much attention due to its good performance [93]. The tuning framework in this thesis is chosen to be built around the TPE, as it has been shown to be more efficient and precise in applications with many tunable hyperparameters.

Table 3.2
DL tuning settings

Setting	Value
Number of trials	500
Models trained per instance	10
Max epochs	1000
Stopper patience	20
Loss function	MSE
System	Fox HPC

3.7.2 Implementation

For the tuning procedure, the Python library Optuna is chosen. Optuna offers flexibility and modularity within a simple interface. It also offers tools for increasing efficiency and accuracy [5]. The most desirable methods included in Optuna are:

- **TPE support**: Optuna uses TPE as default.
- **Warm starts**: The possibility to instruct the tuning algorithm to train and learn on a predetermined set of hyperparameters.
- **Pruner**: A pruner for stopping unpromising experiments early so they don't waste resources.

All of the above are utilized in this framework.

3.7.3 Overview of the tuning process

Each DL algorithm and dataset instance combination is tuned in case different data work better on different hyperparameters. The tuning procedure starts before the training experiments described in Section 3.6 to obtain the tuned hyperparameters before the final main experiment. An overview of the settings used in the tuning procedure can be viewed in Table 3.2. The number of trials (unique hyperparameter experiments) is set to 500 to give the tuning algorithm enough combinations to explore. The number of trained models per model-type dataset instance has been decreased to 10 to reduce runtime. Stopper patience is reduced to 20 for the same reason. The process of tuning can be summarized as follows:

1. **Initialize trial**: If the number executed of trials is below 500, a new trial is initiated.

2. **Selecting hyperparameters**: The optimization algorithm chooses a set of hyperparameters within the instructed ranges for the trial.
    (a) **Warm starts**: For all models, the first five sets of hyperparameters are so-called "warm starts." These are hyperparameters manually set beforehand to give the algorithm a good ground to start off on. The warm-start hyperparameter sets are derived from Darts' standard hyperparameters for the respective models, as well as from other research papers with similar models.
3. **Train 10 models**: 10 model instances with the selected hyperparameters, all with different weights, are initiated and trained. The number of trained models is reduced during tuning to save computational resources and rescue training time.
    (a) **Fixed seeds**: To ensure reproducibility, the 10 models are trained with the same 10 fixed seeds.
    (b) **Pruning**: To reduce training time, a pruner is utilized to monitor the validation loss and stop promising trials. This is done by comparing the current trial with the past executed trial.
    (c) **Early stopping**: Stopping of models when learning deteriorates is also included in the tuning procedure, but the patience is set to 20 to reduce time and computational expenditures.
    (d) **Backtesting**: After stopping or reaching the max epoch, the best model instance is loaded and backtested on the validation set to calculate MSE loss (the test set is not utilized during tuning due to data leak/look-ahead bias).
4. **Mean MSE reported**: The mean of all the final validation loss from all models is calculated and then reported to the optimization algorithm. Then the algorithm returns to step 1.

During tuning, the optimization algorithm records the best-performing hyperparameters. When the models are finished after the 500 trials, the Optuna study is saved to a file in which the best parameters are obtainable. These can then be extracted and used in the main experiment.

It is important to mention that sophisticated tuning procedures run the risk of overfitting to the validation set. Therefore, all hyperparameter-tuned models are compared with models trained with a fixed set of standard hyperparameters.

Chapter 4

The Economic Dataset

This chapter presents an overview of the dataset collected for the experiments. It will briefly present its composition in Section 4.1, discuss the results from the preprocessing procedure in Section 4.2, and finally present the process of attaining the four final feature-reduced datasets through the selected feature selection processes in Section 2.4.

4.1 Composition

This section will give an introduction to how the dataset is comprised. A full overview of the dataset's features, their respective abbreviation, and sources are presented in Table 4.1 on page 57.

4.1.1 Length, size, and span

After the initial processing steps, which were discussed in Section 3.3, the total length of the dataset is 9110 b-days. The average number of b-days per year is 260.93 in the dataset. This is rounded to 261 and selected as the formal number of b-days per year in this thesis and is the basis of the 261 b-days long one-year forecasting horizon. This corresponds to nearly 35 years of b-daily data, ranging from 01.02.1988 to 31.12.2022.

4.1.2 Categorization

Including the target variable, the dataset consists of 72 unique features. The features have been grouped into 12 categories:

1. **Stock indices**: Includes the target OSEAX index and its belonging technical indicators, as well as two foreign indices relevant to the OSE: MSCI-W and WILL5000.
2. **General macroeconomic indicators**: Features related to general Norwegian macroeconomic data.
3. **Consumption indicators**: Indicators reflecting consumer, retail, and government consumption.
4. **Residential indicators**: Features regarding the housing market.
5. **Foreign currency exchange rates (forex)**: The exchange rates between the Norwegian Krone (NOK) and relevant forex, such as the USD and GBP.
6. **Commodities**: Features reflecting the prices of important basic raw materials.
7. **Interest rates**: Indicators of the interest rates set by NB and on different assets.
8. **Monetary aggregates**: Feature representing measures of the amount of money in circulation within Norway.
9. **Debt indicators**: Features representing the debt burden within different sectors.
10. **Economic health and sentiment indicators**: Indicators often used to gauge the overall economic and stock market sentiments, both globally and in Norway.
11. **Geopolitical risk indices**: Included as a substitute for sentiment analysis data. These risk-measuring indices, developed by Dario Caldara and Matteo Iacoviello, work by analyzing sentiment through large, global-spanning newspapers [15].
12. **Time and date**: Features representing time and date-related information.

The features in the full overview in Table 4.1 are ordered by these categories.

Table 4.1
Overview of the features, their abbreviations, and sources

Category	Variable	Abbreviation	Source(s)
Stock indices	Oslo Børs all-share index	OSEAX	[71, 87, 31]
	OSEAX MACD histogram	MACD	Computed
	OSEAX relative strength idx	RSI	Computed
	OSEAX 50d moving average	50d MA	Computed
	OSEAX 200d moving average	200d MA	Computed
	MSCI world index	MSCI-W	[80]
	Wilshire 5000 full cap index	WILL5000	[6]
General macro-economic indicators	Gross domestic product	GDP	[84]
	Current account balance	CAB	[84]
	Net international trade	Trade	[72]
	Consumer price index	CPI	[84]
	Producer price index	PPI	[72]
	Industrial production	Production	[72]
	Wages in manufacturing	Wages	[72]
	Unemployed persons	Unemployed	[72]
	Unfilled job vacancies	Vacancies	[72, 84]
Consumption indicators	Government consumption	C govt.	[72]
	Private consumption	C priv.	[72]
	Retail trade index	RTI	[72]
	New car registrations	New cars	[72]
Residential indicators	Residential property prices	Home price	[43]
	Residential build costs	Build costs	[72]
	Residential build starts	Home starts	[72]
Foreign currency exchange rates	NOK trade-weighted index	NOK-TWI	[68]
	USD/NOK exchange rate	USD/NOK	[68]
	GBP/NOK exchange rate	GBP/NOK	[68]
	SEK/NOK exchange rate	SEK/NOK	[68]
	DKK/NOK exchange rate	DKK/NOK	[68]
	JPY/NOK exchange rate	JPY/NOK	[68]
	CAD/NOK exchange rate	CAD/NOK	[68]
Commodities	Brent crude oil price	Crude oil	[27]
	Natural gas price	Natural gas	[44]
	Gold price	Gold	[44]
	Copper price	Copper	[58]
	Lumber price	Lumber	[58]

Table 4.1 continued from previous page

Category	Variable	Abbreviation	Source(s)
Interest	Key policy rate	Policy rate	[68]
	Money market rate	3m NIBOR	[72]
	3-month treasury bill rate	3m T-bill	[87, 68]
	6-month treasury bill rate	6m T-bill	[87, 68]
	12-month treasury bill rate	12m T-bill	[87, 68]
	3-year government bond rate	3y G-bond	[68]
	5-year government bond rate	5y G-bond	[68]
	10-year government bond rate	10y G-bond	[68]
Monetary aggregates	Base money supply	M0	[84, 68]
	Narrow money supply	M1	[84, 72]
	Intermediate money supply	M2	[84, 72]
	Broad money supply	M3	[72]
	International reserves	I-reserves	[84]
Debt indicators	Debt general public	D public	[84]
	Debt corporations	D corporate	[84]
	Debt private households	D households	[84]
	Debt central government	D cent. govt.	[84]
	Debt municipal government	D mun. govt.	[84]
Economic health and sentiment indicators	Consumer confidence idx EU	CCI EU	[72]
	Business confidence idx NOR	BCI NOR	[72]
	10y/3m rates spread NOR	10y/3m NOR	Computed
	10y/3m rates spread U.S.	10y/3m U.S.	[83]
	Number of bankruptcies	Bankruptcies	[84]
Geopolitical risk indices	Geopolitical risk index global	GPR	[14]
	GPR 7d moving average	GPR 7d MA	[14]
	GPR 30d moving average	GPR 30d MA	[14]
	GPR acts risk index global	GPRA	[14]
	GPR threats risk index global	GPRT	[14]
	GPR index NOR recent	GPR NOR	[14]
	GPR index NOR historical	GPRH NOR	[14]

Table 4.1 continued from previous page

Category	Variable	Abbreviation	Source(s)
Time and date	Year	Year	Darts
	Month of year	Month	Darts
	Day of month	Day	Darts
	Week of year	Week	Darts
	Day of week	Day/week	Darts
	Holiday binary indicator	Holidays	Darts
	Datetime value	Datetime	Darts

4.2 Preprocessing results

This section will present any important results from the preprocessing operations, which were described in Section 3.3.

4.2.1 Detecting and handling outliers

To detect any possible outliers, a visual inspection is conducted by plotting and inspecting all features individually. From this, only one possible troublesome anomaly is detected in the T-Bill features. This is in the short-term interest rate indicators 3-M T-Bill, 6-M T-Bill, 12-M T-Bill, and 10Y-3M NOR. In 1990, the short-term Norwegian Treasury Bills dropped from around 10-12 % down to 0 % and then surged back up to initial levels.

Some research into this drop has been conducted. Although no direct proof of the drop was identified, it occurred during the Norwegian banking crisis [64]. This may indicate that the drop is not an anomaly. Nevertheless, it is decided to remove the anomaly by forward filling.

4.2.2 Managing non-stationary features

This subsection presents the results of the ADF tests and which variables are transformed to make them stationary. To prevent data leakage, the test is conducted on the test set only.

ADF test results

From the ADF test results, presented in Table B.2 on page 115, it is seen that only 23 of the 72 features are stationary. The features failing the test will undergo a data transformation process to make them stationary.

Data transformation

Features not fulfilling the ADF test get all their individual data points differentiated to a 1-year delta. The choice of using year-to-year deltas is because of the smoother, less noisy data. The features are transformed by either:

(a) **Percentage change year-over-year**: Each individual data point is transformed by

$$\Delta\%(t) = \frac{x_t - x_{t-261}}{x_{t-261}} \tag{4.1}$$

where $\Delta\%(t)$ is the function for calculating the year-on-year change as a fraction at time point t, X_t is the value of the time series at time t, and X_{t-261} is the value of the time series one dataset adjusted business year (261 business days) before time t. Some features are on forms that risk creating infinity values when differentiated on a percentage basis. For such features, option (b) is performed instead:

(b) **Change in actual value year-over-year**: Alternative transformation if the percentage differentiation method in (a) runs the risk of resulting in infinity values. For these variables, each data point is transformed by

$$\Delta f(t) = x_t - x_{t-261} \tag{4.2}$$

where $\Delta f|(t)$ is the function for calculating the year-on-year change in terms of absolute value at time point t, X_t is the value of the time series at time t, and X_{t-261} is the value of the time series one year (or 261 days) before time t.

Having two transformation methods resolves two problems:

1. **Risk of infinity values for the percentage year-on-year form**: The problem of infinity values arises for interest rates and other features denominated in percentage or any other features crossing into zero-territory. When these variables go from 0 to any other value, the percent change becomes +infinite or -infinite.
2. **Change in intensity**: If an original pre-transformation series has a trend, the intensity of the changes in its value may increase over time. This may incur problems for some ML algorithms, as the future values outside its training set may be outside the bounds it is trained to forecast.

In addition to this, some exceptions require attention: Some values are considered statistically stationary by the ADF test but are still converted to year-to-year percentage change. This is so that all variables within each

Table 4.2
Overview of the feature-reduced datasets

Univariate	RF strict	RF top-7	Selected
OSEAX	OSEAX	OSEAX	OSEAX
Datetime	Datetime	Datetime	Datetime
	50d MA	50d MA	WILL5000
	MSCI-W	MSCI-W	CPI
	RSI	RSI	Production
		200d MA	3m NIBOR
		GBP/NOK	10y G-bond
		MACD	Crude oil
		Gold	USD/NOK

category are standardized or because the variables could qualitatively be regarded as having a time-dependent trend (not stationary) despite not being caught by the ADF test. This counts for M0, House Starts, Lumber, Bankruptcies, DKK/NOK and CAD/NOK. Due to their nature, these variables may fundamentally increase over time due to growth in population, the economy, and or inflation. This underlying trend may not have been caught during the variable's data time span or caused by extraordinary events. None of the time and date features are converted, as their underlying information will get altered to a state that probably does not contain any information valuable to the algorithms.

4.3 Feature selection

The results from the feature selection process and its resulting datasets are presented in this section. From this, four feature-reduced datasets are built: the nine-feature *selected* dataset based on earlier studies, the *RF strict*, which is a minimal RF-FS dataset build, the *RF top-7*, which is a larger nine-feature RF-FS build, and the *univariate* dataset which contains the target variable only (in addition to a Datetime value). The maximal number of features is set to nine to allow 1000 samples per feature. An overview of the final feature-reduced datasets is presented in Table 4.2.

4.3.1 A target feature baseline

To assess if adding additional features to a stock index forecasting approach enhances its performance, a baseline is needed for comparison. Because of this, the *univariate* dataset is built. This baseline dataset consists of the OSEAX target index, but to make it able to run with the TFT, the *datetime* value is also included since the TFT requires at least one future covariate feature. Due to this, the dataset is not strictly univariate, however, it was chosen to keep the name for simplicity's sake.

4.3.2 Research based selection

Within the research-based selection process, multiple papers and theses regarding which indicators may influence the OSE have been analyzed. From this, nine features are selected. First of all, the OSEAX target variable is included, as the target feature is also the most common feature to use for input data. In addition, the datetime feature is included to have a time-related feature within the dataset. This is important because the TFT requires at least one feature to be used as a future covariate.

The dataset of research-selected data is called the *selected* dataset. It contains the following features, with their respective inclusion justifying studies/theses cited behind: *OSEAX* [50], *WILL5000* [41, 78], *CPI* and *production* [34, 66, 78], *3m NIBOR* [34, 78], *10y G-bond* [86, 57], *crude oil* [86, 34, 41], *USD/NOK* [57, 9, 78], and *datetime* (due to TFT compatibility).

4.3.3 RF-FS based selection

Two dataset instances are built using the RF-FS procedure. The RF-FS is fed the full 72 feature dataset, and a H shifted target OSEAX data series. After executing the RF-FS procedure, it returns a list of all the features it considers optimal for this forecasting environment. The result is interesting, as the only feature the implementation finds suitable is the target OSEAX series alone. The RF-FS algorithm chooses to go univariate, meaning it only selects the OSEAX and discards all the remaining 71 features. This indicates the algorithm finds no important information in the other features for a one-day horizon.

It is decided to alter the approach. As a *univariate* dataset already exists, two RF-FS datasets will be built using its outputted *feature scores* instead. These differ from the original approach in that all features are listed by their scores instead of just the best being returned in a list (without scores).

These are built to test if there is any information in the top-ranked features that may outperform the *univariate* or *selected* datasets. The two RF-FS are named *RF strict* and *RF top-7*, and their contents are presented in Table 4.2.

The *RF strict* is built with the 3 top ranked features from the RF-FS, in addition to the target variable and datetime feature. It was originally considered a "features strictly selected by the RF-FS" representation, as the RF-FS can return multiple features. But with the results only choosing the target feature, it is decided to make this dataset consist of the top 3 ranked features by extracting the algorithms feature ranking scores. The *RF top-7* is built on the same basis as the RF strict ended with but with the top seven best-ranked features. By having the n-best approach, it can be tested if using RF-FS ranked features can be better than just the single feature originally chosen by the algorithm. It also allows for testing a broader amount of features from the data set.

Chapter 5

Experiments & Results

This chapter presents the results obtained through this thesis methodologies. First, Section 5.1 presents the hyperparameters found by the DL model tuning procedures. Following this, the experiment and process will be explained in Section 5.2. This includes all final forecasting models, feature-reduced datasets, and hyperparameter combinations. Finally, Section 5.3 will present the final result obtained from the experiments.

5.1 Hyperparameter tuning results

This section presents the results from the tuning procedures and their subsequent hyperparameter sets for each model. An overview of the tuned parameters for each dataset is presented in Table 5.1 on page 67 for the LSTM and Table 5.2 on page 69 for the TFT.

5.1.1 LSTM

From the results presented in Table 5.1 (page 67), it can be observed each of the 1-day horizon LSTM dataset instances adopt quite different architectures and training parameters, except for the optimizer and the date and time feature inclusion. "Month" is the most popular feature used by two model-dataset instances: Univariate and Selected, the two most dissimilar datasets. The latter also chooses the "Holiday" indicator, while all the remaining features remain unchosen. All instances prefer the Adam optimizer in this experiment. Most models adopt a single-layered architecture for the number of hidden layers, except for the RF Top-7 instance, which is more

optimal with two. The smallest Univariate and the largest selected dataset instances adopt about the same input window sizes and dropout rates.

Some trends can be identified. Firstly, the RF feature selection-based datasets prefer a more significant number of total nodes: around 335. The nodes are organized in either one layer or spread over two. Secondly, the RF-based dataset shares the same batch size and learning rate. Thirdly, the RF-based dataset instances prefer a smaller input window depth and dropout rates than their manually built peers. More specifically, for each dataset, the following is noted: Finally, the two most extensive datasets prefer the warm restating cosine annealing LR scheduler strategy. Both the RF top-7 and the selected datasets tuning results in the CosAWR scheduler (and also with very similar scheduler parameters). The next paragraphs will look at each LSTM dataset combination separately:

Univariate : After finishing the tuning, it is observed through the validation loss log that the univariate instance performs poorly. In the tunings resulting parameters, the algorithm has chosen to use the StepLR learning rate scheduler with a step size S of just 3. With the predefined fixed gamma γ of 0.1, the learning rate is reduced by 90% after every third epoch with this setting, which is quite steep. It is decided to use a more fitting step size in the final experiment, so S is set to 40. With this, the validation losses improve to about the same level as for the standard univariate LSTM. For the other hyperparameters, it can be seen that the algorithm has chosen the highest possible LR of 0.1, probably due to utilizing an LR scheduler. The univariates-instance ends up with the largest batch size of the four while at the same time adopting a high dropout rate of 0.7. It also chooses quite a large input window depth compared to the RF-based sets.

RF Strict : When introducing the technical indicators selected by the RF, it is evident that the optimization algorithm prefers none of the date and time features. The total amount of nodes, on the other hand, increases while the input windows narrow compared to the univariate results.

RF Top-7 : With the RF Top-7 dataset's additional four features, the optimization search does not result in any inclusion of extra time and date features. It has reduced its input window size compared to its 5-feature RF Strict counterpart. The total amount of hidden layer nodes is the same as for its sibling but is now spread over two layers instead of just one. The RF Top-7 dataset instance is the only one adopting a two-layered hidden architecture of all four LSTM dataset instances. Lastly, this instance is one of the

two that adapt the CosAWR LR scheduler.

Selected : Finally, the selected dataset instance search results in quite a different set of hyperparameters than the second largest RF Top-7. It adopts the same optimizer and LR scheduler (and nearly the same LR scheduler parameters), but the similarities end here. Architecturally, it is similar to the most optimal LSTM model found by Bhandari et al., which is also trained with economic indicators.

5.1.2 TFT

The TFT tuning results are presented in Table 5.2 on page 69. In the overview, the following can be spotted: Most of the TFT models chose to include multiple date and time features. The *RF strict* trained TFT is the only model not utilizing any. There is a lot of variation between the models regarding hidden layers, nodes, and continuous layer nodes. Like the LSTM, all models adopt the Adam optimizer. Also, TFT models use a narrow input depth (window) of 7-14 days, a small batch size, and prefer a low or no dropout. The number of attention heads seems to increase with dataset size. LR is mostly set to be fixed, with all models choosing a starting LR of 0.001 or lower. More specifically, for each TFT dataset instance, the following is observed:

Univariate : The tuning findings include all time and date features, except *day*/*week*, as the most optimal. Architecturally, the *input depth* is set to seven days, with a single hidden layer of 247 nodes. The univariate TFT is the only model utilizing an LR scheduler.

RF Strict : This model goes for a no-time-feature approach as well as consisting of a double-layered 37-node architecture. It has the largest hidden continuous layer and is the only model utilizing dropout.

RF Top-7 : The larger RF-FS dataset TFT adopts a very different approach by including most date and time features, having a larger input depth, and a large single-layered architecture of 307 nodes. The number of continuous nodes is 19, while the number of attention heads has increased by many folds to eight (compared to one for the *RF strict*).

Selected : Finally, the selected TFT adopts something that seems like a common ground between the hyperparameters of the univariate and RF

Table 5.1
Hyperparameter tuning results for the LSTM models

Type	Parameter	Dataset			
		Univariate	RF Strict	RF Top-7	Selected
Inclusion of extra date and time features	Year				
	Month	✓			✓
	Day/month				
	Week				
	Day/week				
	Holiday				✓
Architecture	Input width	3 $_{features}$	5 $_{features}$	9 $_{features}$	11 $_{features}$
	Input depth	161	68	19	178
	Layers $_{hidden}$	1	1	2	1
	Nodes/layer	31	336	167	149
	Dropout	0.7	0.0	0.4	0.7
Training	Batch size	16	4	4	8
	Optimizer	Adam	Adam	Adam	Adam
	LR	0.1	0.001	0.001	0.01
	Scheduler	StepLR	None	CosAWR	CosAWR
		$S = 40$*		$T_0 = 18$	$T_0 = 24$
		$\gamma = 0.1$		$T_m = 4$	$T_m = 3$

LR = learning rate.
S = step size.
γ = gamma - a factor by which the LR is reduced by multiplication.
T_0 = iterations before the first restart.
T_m = A T_0 multiplication factor increasing the n-iterations after each restart.

* Set to 3 by the search, but set to 40 due to bad performance

top-7. It uses three of six time and date features. Architecturally, this model has an input depth of 12, coupled with a single 176-node hidden layer and a 72-node continuous layer. Attention head count is at 5.

5.2 Experiment procedure

This section summarizes the final models, feature-reduced datasets, and hyperparameter set combinations, as well as their respective rationales for inclusion. Additionally, it presents the experiments based on these combinations and how they are conducted.

5.2.1 Reiteration of final components and rationale

The experiment training and evaluation procedure is built around the selected forecasting models, feature selection, and hyperparameter sets.

Models: There are four models: RWM, RWD, LSTM, and TFT. The TFT is the main "novel" model in this research. The RWM and RWD models are included to test the DL models against the RW, while the LSTM is included due to its proven performance in stock market forecasting and thereby as a benchmark against the TFT.

Datasets: For the LSTM and TFT, there are four feature-reduced datasets: the *univariate*, *RF strict*, *RF top-7*, and *selected*. The *univariate* is a benchmark to see if including more sophisticated features enhances performance. The RF-FS-based feature-reduced datasets are included to test if using the RF-FS n-feature ranking procedure improves performance. Finally, the *selected* dataset also serves as a benchmark against the other sets and a measure of how a "manually" selected feature-reduced dataset fares in this environment.

Hyperparameters: In addition, each DL model and feature-reduced dataset combination can be trained with the *standard* and *tuned* hyperparameter sets. Including one experiment with the *standard*, generally well-performing hyperparameters are done to benchmark against the *tuned* hyperparameters. Using two sets of hyperparameters per DL model is also done to compare and monitor if the tuned models are overfitting.

Table 5.2
Hyperparameter tuning results for the TFT models

Type	Parameter	Dataset			
		Univariate	RF Strict	RF Top-7	Selected
Inclusion of extra time features	Year	✓		✓	
	Month	✓			✓
	Day/month	✓		✓	✓
	Week	✓		✓	
	Day/week			✓	✓
	Holiday	✓		✓	
Architecture	Input width	7 $_{features}$	7 $_{features}$	14 $_{features}$	12 $_{features}$
	Input depth	247	37	66	176
	Hidden layers	1	2	1	1
	Nodes/layer	72	111	307	77
	Nodes cont.	99	110	19	72
	Dropout	0.0	0.1	0.0	0.0
	Att. heads	1	1	7	5
	Full attention	True	False	True	False
Training	Batch size	2	4	2	2
	Optimizer	Adam	Adam	Adam	Adam
	LR	0.001	0.001	0.0001	0.001
	Scheduler	RPlatau $\gamma = 0.1834$ $P = 6$	None	None	None

LR = learning rate.
γ = gamma - a factor by which the LR is reduced by multiplication.
P = number of epochs with no improvement before the LR gets reduced.

5.2.2 Procedure

With the available combinations described in the previous section, the experiments which will be conducted are the following:

- **RW-models**: One experiment per RW-based model is conducted, both with a strictly univariate dataset, totaling two experiments. Since the RW-based implementations are deterministic, only one run is required per model. The results are presented in Subsection 5.3.1 for the RHM and in Subsection 5.3.2 for the RWD.
- **LSTM-models**: In total, eight experiments are conducted, one for each feature-reduced dataset and hyperparameter combination. Each experiment trains and compares the results of 30 sub-models. The results are presented under Subsection 5.3.3.
- **TFT-models**: The same procedure as for the LSTMs: Eight experiments are conducted, one for each feature-reduced dataset and hyperparameter combination. Each experiment trains and compares the results of 30 sub-models. The results are presented under Subsection 5.3.4.

These model-dataset-hyperparameter combinations are trained and evaluated in accordance with their respective procedures described under Section 3.6

5.2.3 Results presentation

For each experiment, the results will be presented in different measures to give a complete and thorough picture of the different combinations of performances. Each model and tuning combination result is presented by:

- **Tables**: To present the absolute accuracies and precisions, the MAPE, RMSE, and R^2 metrics discussed in Subsection 3.6.3 are presented in a table accompanied by their standard deviations in parentheses. In addition, the full runtime of the 30-model experiment will be included as a measure of efficiency. All the above four measures are presented for each feature-reduced dataset.
- **Line plots**: Included to give a visualization of the model forecast against the actual values. The plots are focused on the period of the COVID-19 stock market crash between the dates of 14.02.2022 and 16.04.2020 to visualize the forecasts during a challenging drawdown period. For the DL models, the forecast line plot is the average of the

30 trained sub-models and includes error bands for indicating standard deviation. To preserve space, only the most accurate model-tuning-dataset-instance is plotted for each DL model-tuning version experiment.

- **Box plots**: These plots are included to visualize the internal differences between the different feature-reduced datasets for each model-hyperparameter instance. Therefore, they will only be included in the DL models. All measures included in the tables will also be presented in the box plots, but the total runtime will be presented as *mean* runtime (over the 30 sub-models) instead. This is to visualize if the sub-models differ in train time, which can signal instability.

In this experiment, MAPE is considered as the "main" metric. This is because it is relative and unaffected by the size of the OSEAX stock index valuation. For example, the RMSE can differ in magnitude when the base value of the index changes: 1 % of 100 is 1, while for 1000, 1 % it is 10. Summarizing the results, a sorted consolidation of all the forecasting models MAPE scores is presented in Figure 5.11 on page 84.

5.3 Forecasting results

This section presents the results from the forecasting experiments. Initially, the baseline RW-based models' results will be presented, with the classic RWM in Subsection 5.3.1 and then the "drifting" RWD in Subsection 5.3.2. Following this, the DL models' results will be presented in Subsection 5.3.3 for the LSTMs and Subsection 5.3.4 for the TFTs.

5.3.1 RWM

Table 5.3 outlines the RWM experiments metrics. As the model is deterministic, no standard deviations are included. Despite being a straightforward approach, the RWM achieves relatively high accuracy with a MAPE score of 0.8071 and an R^2 value of 0.9953 with a total runtime of just 20 seconds on the local system. The model's mean day-to-day forecasting error is 12.3756 index points, as shown by its RMSE value. The line plot in Figure 5.1 shows that the simple repeating pattern forecast is closely intertwined with the true values. If weekends were to be included, the RWMs forecast would be a perfect 1-day forward-shifted copy of the true OSEAX values. This is not the case here as the dataset only represents b-days.

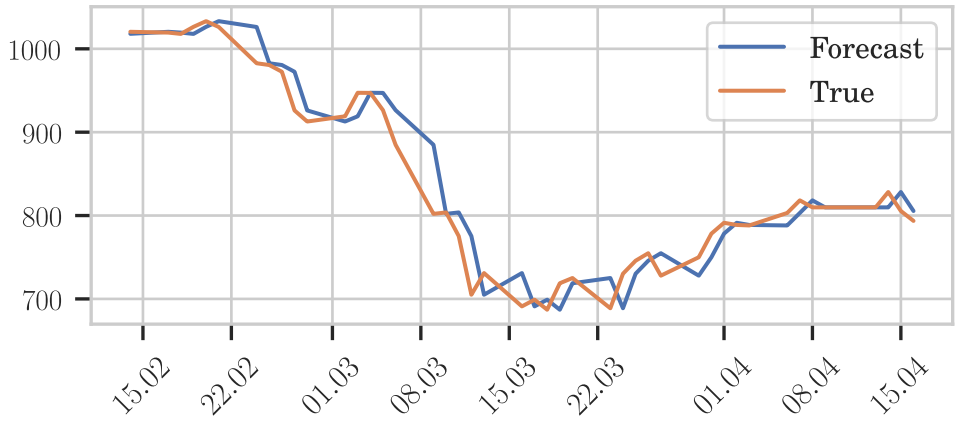

Figure 5.1
Line plot - RWM forecast

Table 5.3
Metrics - RWM and RWD forecasts

Model	Metric			
	MAPE	RMSE	R^2	Runtime
RWM	0.8072	12.3757	0.9953	00:00:20
RWD	0.8074	12.3736	0.9953	00:00:20

5.3.2 RWD

Outlined in Table 5.3, the resulting metrics from the RWD model's forecast do not differ much from the simpler RW model in Subsection 5.3.1. Despite having to compute the mean 1-day growth over the training set additionally, the RWD finishes within the same 20 seconds of runtime as its simpler, no-drift version. With a MAPE of 0.8073, it underperforms the RW with a mere 0.0247% accuracy-wise. For the RMSE, it underperforms with 0.0170%. When rounded to four digits, the R^2 value is identical. In the line plot in Figure 5.2, the forecast looks identical to its RWM sibling.

5.3.3 LSTM

The first of the two DL implementation results to be presented is the LSTM. Its results are split into two subsections. The first contains the results from the standardized, fixed hyperparameters experiments, and the second from the (tuned) models trained with hyperparameters found through the hyper-

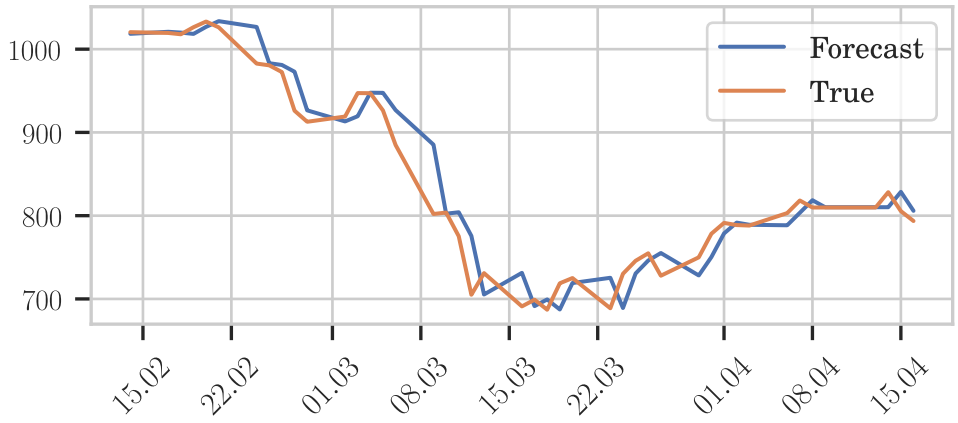

Figure 5.2
Line plot - RWD forecast

Table 5.4
Metrics - standard LSTM forecasts

Dataset	Metric			
	MAPE	RMSE	R^2	Runtime
Univariate	**1.1523 (0.0072)**	**17.4580 (0.0873)**	**0.9907 (9.3e-5)**	02:11:41
RF strict	1.1639 (0.0120)	17.6283 (0.1411)	0.9905 (1.5e-4)	02:10:19
RF top-7	1.1818 (0.0228)	17.8911 (0.2773)	0.9902 (3.1e-4)	02:12:58
Selected	1.3086 (0.0664)	19.9013 (1.1693)	0.9878 (0.0015)	**02:00:25**

Mean metric values (standard deviation) over 30 unique models for each LSTM-dataset instance trained with standard (non-tuned) hyperparameters on 1-day forecasting. The best values and standard deviations for each metric are in bold.

parameter optimization search.

Standardized hyperparameter forecasts

The results from the standard (non-tuned) LSTM experiments are presented in Table 5.4 as mean metrics and in Figure 5.4 as box plots. In summary, the standard LSTM performs best on the *univariate* dataset, achieving a MAPE score of 1.1523 and a standard deviation of 0.0072. The same can be observed in the remaining metrics, excluding runtime, where the *univariate* model spends about the same time as the other models, about over 2 hours. From the line plot in Figure 5.3, a good fit can be observed, and no error bands are visible. In second to the *univariate* 2-feature dataset performance

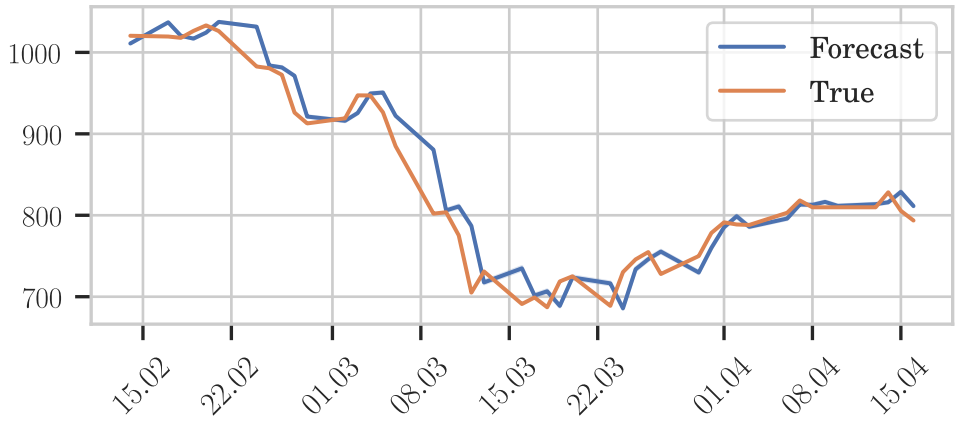

Figure 5.3
Line plot - LSTM standard univariate average forecast

Table 5.5
Metrics - tuned LSTM forecasts

Dataset	Metric			
	MAPE	RMSE	R^2	Runtime
Univariate	1.2113 (0.0229)	18.3565 (0.3322)	0.9900 (3.7e-4)	02:10:36
RF strict	**1.1685 (0.0162)**	**17.8504 (0.1912)**	**0.9902 (2.1e-4)**	03:55:05
RF top-7	1.1973 (0.0227)	18.0686 (0.2435)	0.9900 (2.7e-4)	02:58:36
Selected	1.2445 (0.0295)	18.7615 (0.4183)	0.9896 (4.6e-4)	**02:09:52**

Mean metric values (standard deviation) over 30 unique models for each LSTM-dataset instance trained with optimized hyperparameters on 1-day forecasting. The best values and standard deviations for each metric are in bold.

comes the 5-feature *RF strict*.

Both the nine feature *RF top-7* and *selected* datasets perform the worst, with the manually selected *selected* dataset being the decidedly inferior. When training the LSTM at 1-day forecasting with nine features, RF-FS outmatches the manual. This is evident in the Figure 5.4 box plots, where the selected dataset's poor performance, both in accuracy and precision, skews all the other boxes. None of the standard 1-day forecasting LSTM dataset instances achieves metrics better than the RW models while at the same time requiring more computing resources for a longer duration of time.

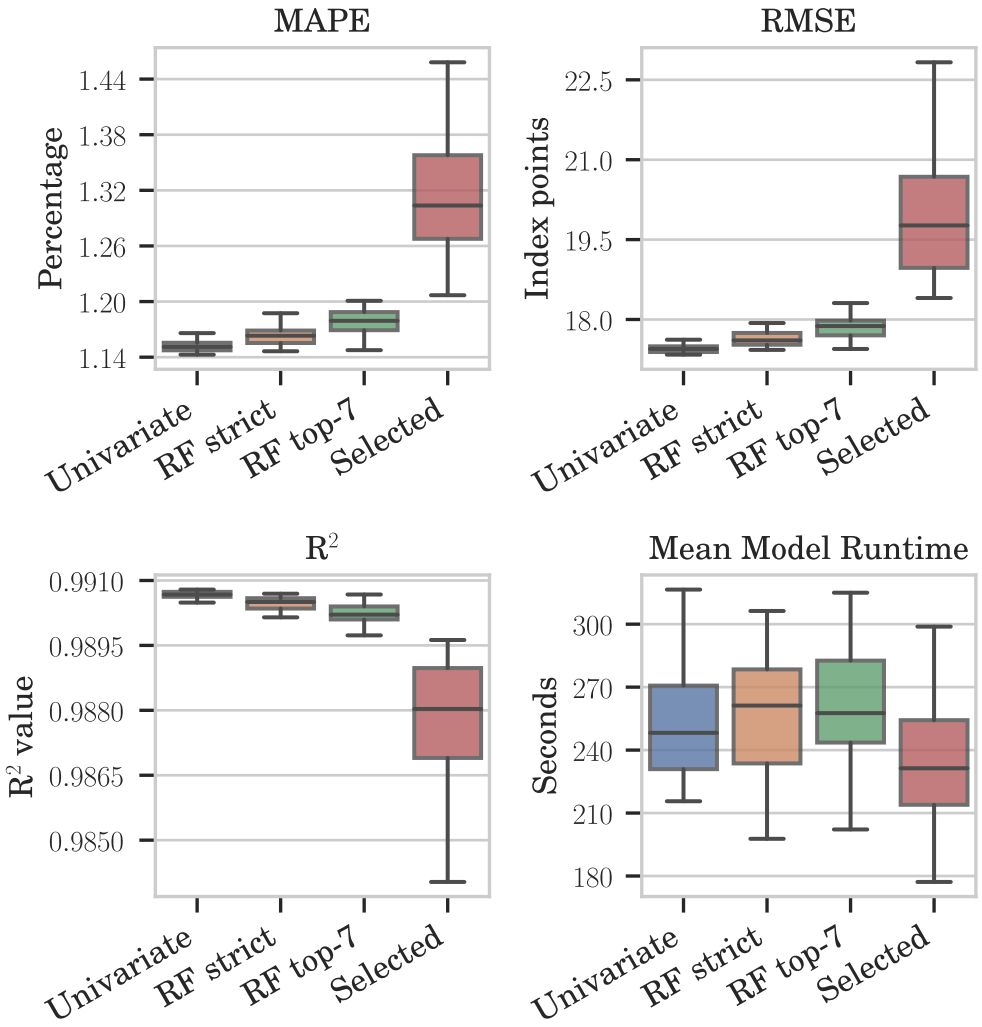

Figure 5.4
Box plots - standard LSTM forecasts

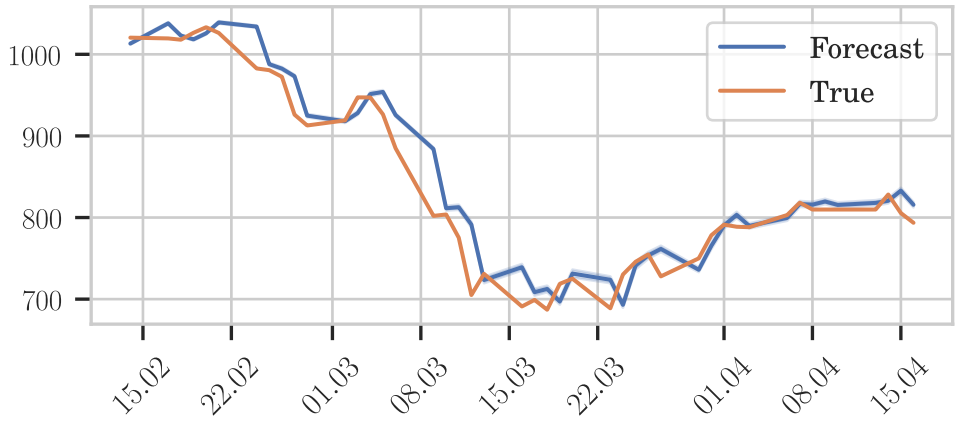

Figure 5.5
Line plot - LSTM tuned RF strict average forecast

Tuned hyperparamterer forecasts

Mean metrics are presented in Table 5.5, and their distributions by box plots in Figure 5.6.

Table 5.5 shows that all LSTM-dataset instances, except for *selected*, have obtained worse accuracies than the standardized LSTM. The *selected* dataset is still the least performing of the four but no longer skews the figure. Regarding MAPE, RMSE, and R^2 value, the *RF strict* has now dethroned the *univariate* dataset, albeit both have slightly worse accuracies than before. The LSTM tuned's forecast line plot in Figure 5.5 is similar to the standard LSTM's best forecasting model. Figure 5.6 shows that the models' metrics score distribution has become more equalized than the standard LSTM. Looking at runtime, both RF-FS-based datasets have increased their time expenditure by quite a lot.

The now-dethroned *univariate* dataset suffers the most and performs worse than both RF-FS-based datasets. Compared to the standardized *univariate* LSTM model, the error has increased by 5.12% for MAPE and 5.15% for RMSE. For the R^2 value, the difference is relatively slight, with a 0.08% reduction in variance explainability.

None of the LSTM implementations have managed to outperform any of the baselines. So far, the best-performing model is the most basic RWM. Of the LSTMs, the model trained with standard parameters on the *univariate* dataset performs the best. The results of the TFT model will be presented and examined in the next subsection.

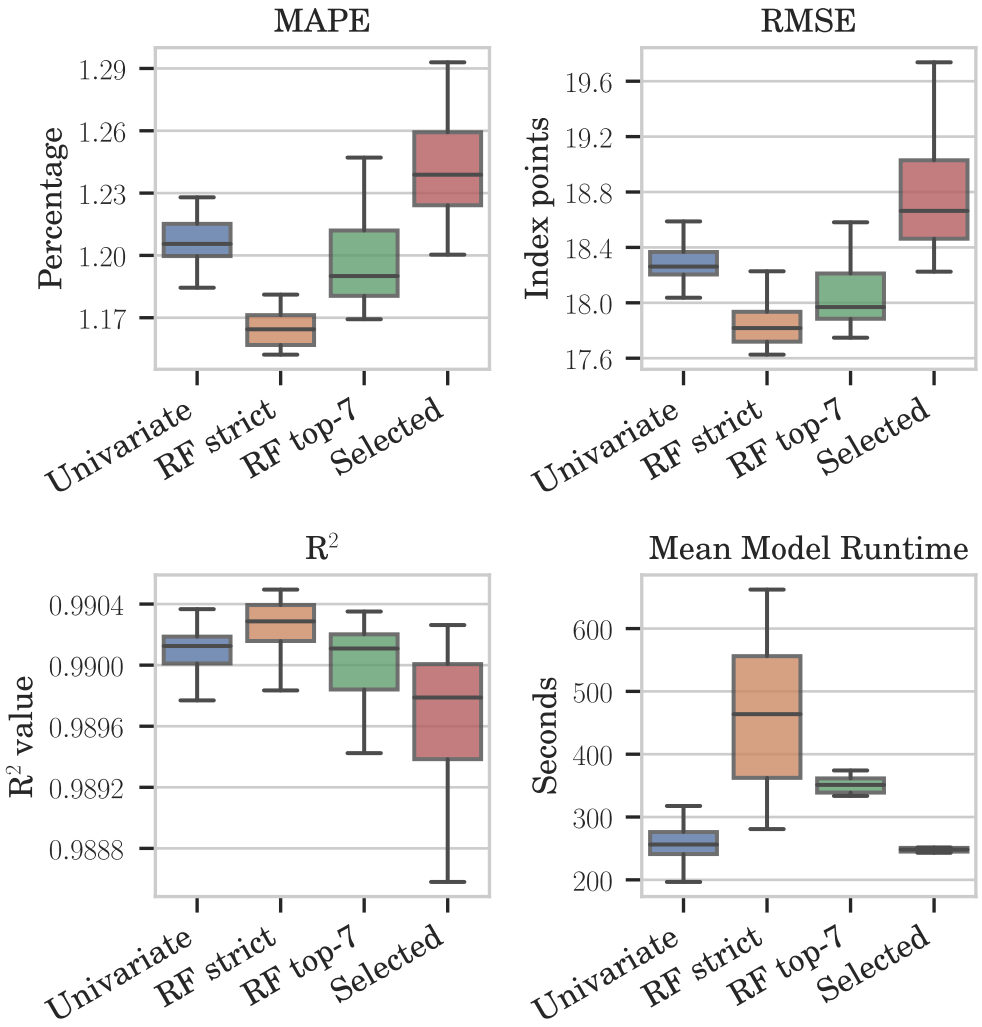

Figure 5.6
Box plots - tuned LSTM forecasts

Table 5.6
Metrics - Standard TFT forecasts

Dataset	Metric			
	MAPE	RMSE	R^2	Runtime
Univariate	2.4530 (0.6574)	36.9051 (15.1442)	0.9514 (0.0690)	04:28:50
RF strict	2.6110 (1.0731)	37.7847 (14.0406)	0.9504 (0.0549)	**05:24:53**
RF top-7	**2.4374** (0.2361)	**35.4544** (3.6894)	**0.9611** (0.0083)	06:10:13
Selected	2.4747 **(0.1835)**	36.0224 **(2.5656)**	0.9600 **(0.0059)**	05:26:13

Mean metric values (standard deviation) over 30 unique models for each TFT-dataset instance trained with standard (non-tuned) hyperparameters on 1-day forecasting. The best values and standard deviations for each metric are in bold.

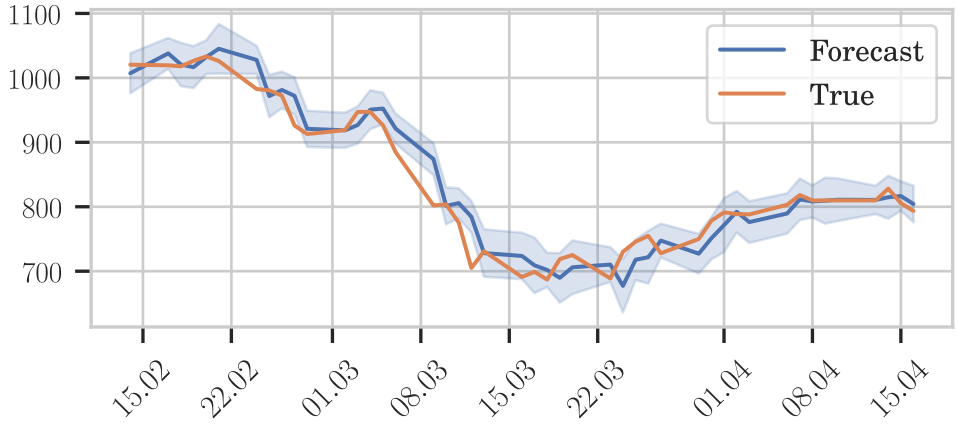

Figure 5.7
Line plot - TFT standard RF top-7 average forecast

5.3.4 TFT

Like the LSTM, the TFT's results are split into two sections, one for the model trained with standardized hyperparameters and one for the model trained with tuned hyperparameters.

Standardized hyperparameter forecasts

Starting with the standardized hyperparameter TFT, the mean metrics are presented in Table 5.6. The metrics' distribution is illustrated with box plots in Figure 5.8. From analyzing the metrics in Table 5.6, the following can be observed: The standard TFT model trained on the *RF top-7* obtains all the best error-related metrics but not the lowest standard deviations. Looking at

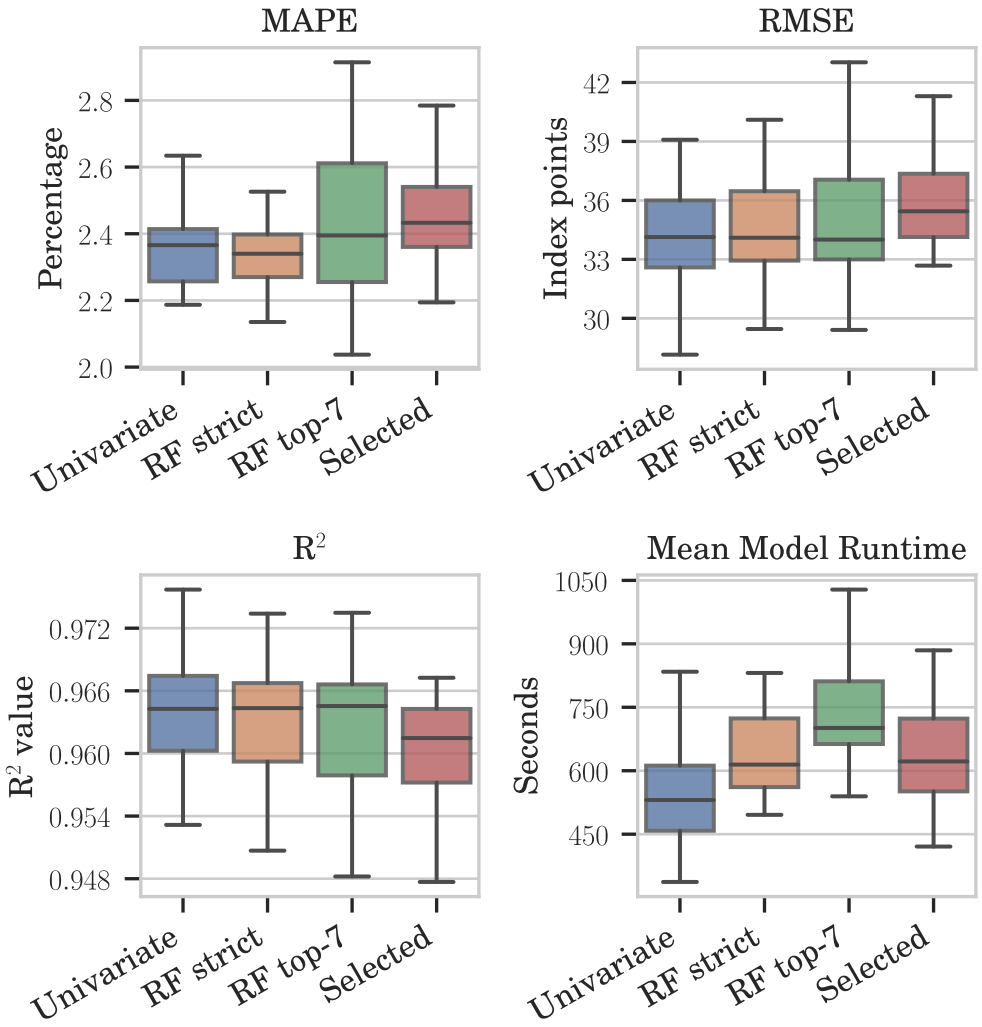

Figure 5.8
Box plots - standard TFT forecasts

MAPE, the *RF Top-7* instance has the lowest MAPE value of 2.4374%, making it the best performer of the datasets. For RMSE, it also scores the best, obtaining the lowest value of 35.4544. For the R^2 value, it scores 0.9611. Large standard deviations can be observed in the line plot in Figure 5.7 through the error bands around the average forecast.

For the remaining three datasets, the results are a bit mixed. Regarding standard deviations, the *selected* dataset is the most precise, scoring the lowest standard deviation across all metrics. The *selected* dataset also performs second best in RMSE and R^2 score. But when looking at MAPE, the *univariate* dataset takes second place with its 2.4530% rate. For the standard TFT model, the larger datasets seem to give the most accurate results.

The metrics distribution in Figure 5.8 shows that the different data sets are more equal in performance compared to the LSTM experiments. The plots are not as skewed, and the boxes are more aligned. The top performing *RF top-7* seems to have a much larger distribution of MAPE scores than the others. This is also observed to a lesser degree in its other metrics.

Among the different datasets, the TFT model trained on the *RF top-7* dataset scores the best accuracy, while the model trained on the *selected* dataset has the highest precision. The former makes, on average, the best predictions, while the latter is more confident and decisive in its forecast.

Compared to the LSTM and RWM implementations, the standard TFT underperforms quite a bit. The best TFT and dataset combination, *RF top-7*, underperforms the RW with a 201.9574% increase in error by MAPE. For RMSE, the increase in error is 186.4840%, and by R^2 value 3.4361%. Looking at the runtime scores, it also consumes more than double the computing resources, in some cases triple, compared to the LSTM models.

Tuned hyperparameter forecasts

The tuned TFT experiment's mean metrics are presented in Table 5.7, while their distributions are visualized as box plots in Figure 5.10.

Unlike the LSTM, the tuning procedure has drastically improved all TFT and dataset combinations. Looking at Table 5.7, the MAPE and RMSE errors have been more than halved compared to the standard TFTs result in Table 5.6. The R^2 values have also been improved. It is also evident that all dataset-model-instances are very similar in performance over every measured metric. This excludes total runtime, as all instances have very dissimilar time expenditures. The runtimes differ greatly from previous

Table 5.7
Metrics - Tuned TFT forecasts

Dataset	Metric			
	MAPE	RMSE	R^2	Runtime
Univariate	1.2215 **(0.0053)**	18.5077 **(0.0542)**	**0.9904 (5.5e-5)**	26:32:32
RF strict	**1.2069** (0.0324)	**18.1231** (0.4209)	0.9899 (4.7e-4)	**14:15:52**
RF top-7	1.2170 (0.0353)	18.2562 (0.4874)	0.9898 (5.5e-4)	58:21:00
Selected	1.2222 (0.0219)	18.4233 (0.2730)	0.9900 (3.0e-4)	41:03:48

Mean metric values (standard deviation) over 30 unique models for each TFT-dataset instance trained with optimized hyperparameters on 1-day forecasting. The best values and standard deviations for each metric are in bold.

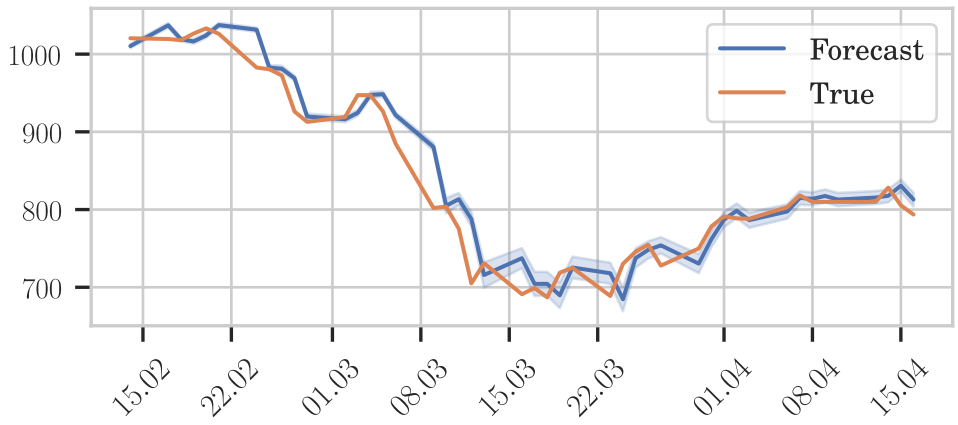

Figure 5.9
Line plot - TFT tuned RF strict average forecast

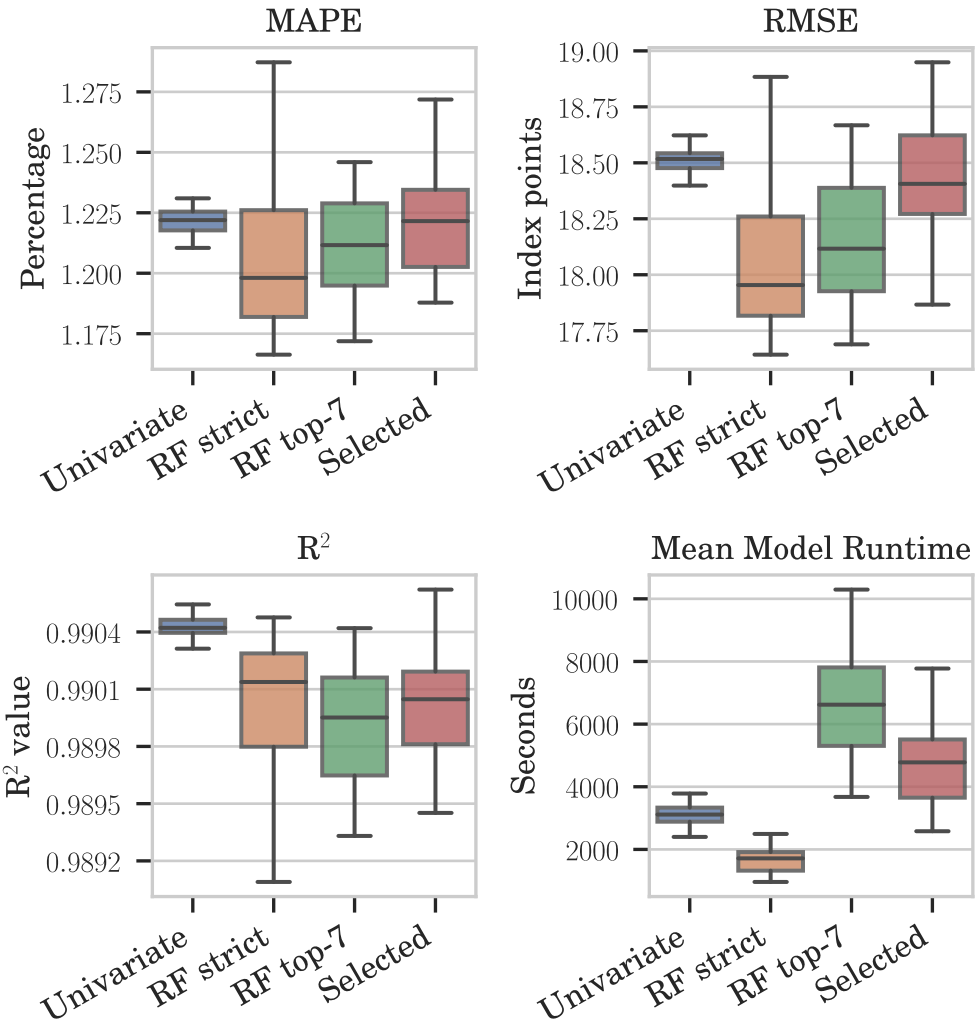

Figure 5.10
Box plots - tuned TFT forecasts

experiments: the tuned TFT models now may take over one day to train, compared to a range of 2-6 hours for all previous experiments.

With tuned hyperparameters, the TFT has advanced closer to the performance territory of the best LSTM models. Two dataset instances stand out. The *univariate* dataset gets the best results over every measured standard deviation, making it the more precise model. It also achieves the best R^2 with a value of 0.9904, which is better than most models in the 1-day forecasting experiment. Measured by MAPE, RMSE, and runtime, the *RF strict* dataset performs the best. With its 1.2069% MAPE, the error has been reduced by 53.7763% compared to the MAPE achieved by the same dataset on the standard TFT. In Figure 5.9, the wide error bands present in the standard model have now shrunk dramatically.

Measuring total runtime, the *RF strict* trained model is the most time-efficient of the four tuned TFTs, with a runtime of 14:15:52. The TFTs trained with the *RF top-7* and *selected* perform close to the two others, but their runtimes of 58:21:00 and 41:03:48, respectively, are of quite another dimension. The former's runtime extends beyond two days on the FOX HPC cluster.

From the box plots in Figure 5.10, the distribution is no longer as even for the standardized TFT. All non-univariate datasets seem to have a wider distribution than the *univariate*, also evident in Table 5.7's standard deviations. It is interesting to note the distribution difference between the two most accurate dataset-model instances. The *univariate* dataset is, as mentioned, the most precise instance and also has the lowest distribution. At the same time, the *RF strict* instance has the widest distribution (albeit not standard deviation) of all the models. Cross-model forecasting volatility does not reduce the MAPE and RMSE accuracy scores for the *RF strict* compared to the other instances. The TFT can use features that are possible to be aware of in the future as future covariates. In this case, these features are the date and time variables. With this capability, the TFT can know if the forecast date is a holiday or which weekday it is, something it can utilize in its calculations.

Compared to the best-performing LSTM and baseline RW-based models, neither the tuned TFT *univariate* nor the *RF strict* datasets achieves a better accuracy. Focusing on MAPE, the tuned TFT trained with the *RF strict* dataset has a 4,7383% higher error rate than the standard *univariate*-trained LSTM. Against the RW model, the increase in MAPE error is as high as 49,4798%.

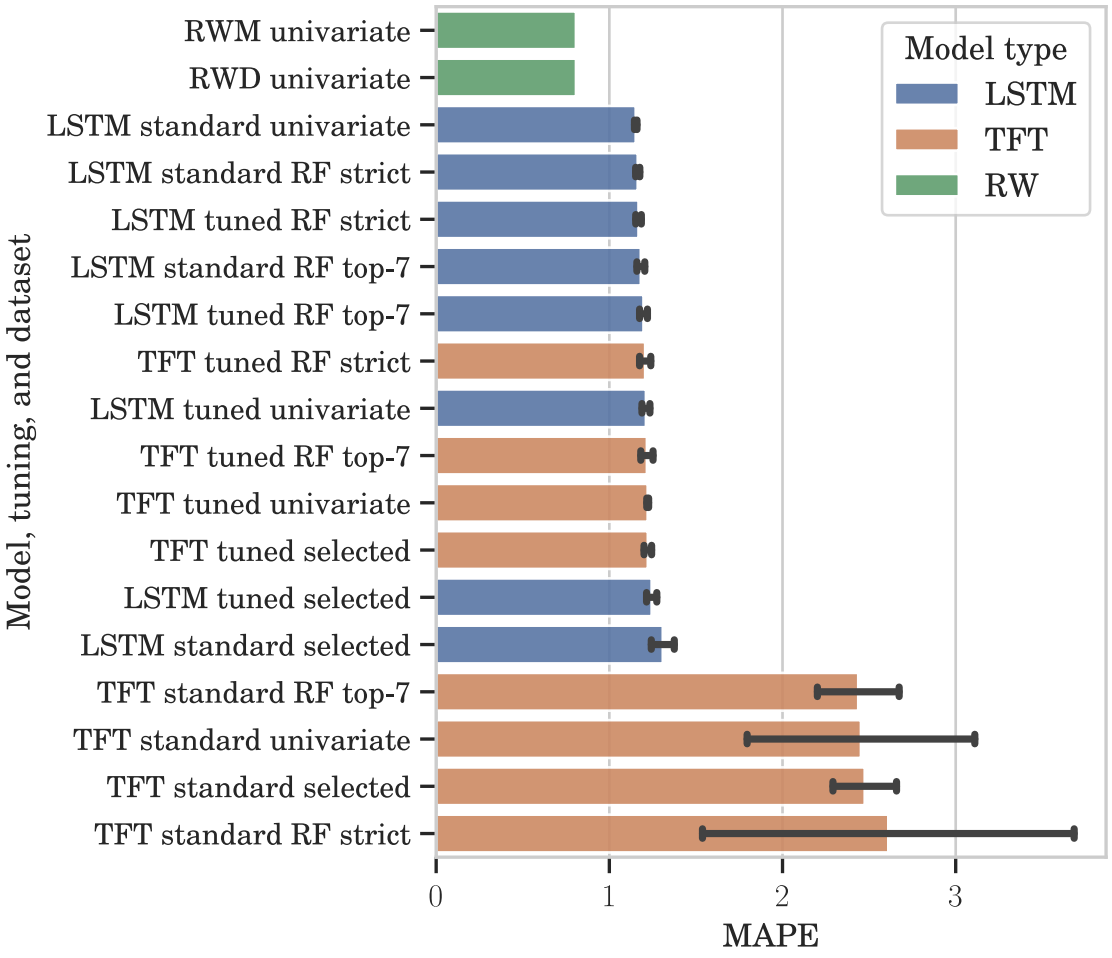

Figure 5.11
Summarized results for all models by MAPE

5.4 Summary

With all forecasting experiments finished, the results are summarized in this section. As MAPE is the primary evaluation metric, the summary will be based on it. A general overview is presented with a descending bar plot in Figure 5.11, including standard deviations for the DL models. For a more absolute view of the final mean MAPEs in numbers, please see Table 5.8.

Table 5.8
MAPE scores for all forecasting experiments

Model	Parameters	Dataset			
		Univariate	RF Strict	RF Top-7	Selected
LSTM	Standard	1.1523	1.1639	1.1818	1.3086
	Optimized	1.2113	1.1685	1.1973	1.2445
TFT	Standard	2.4530	2.6110	2.4374	2.4747
	Optimized	1.2215	1.2069	1.2170	1.2222
RWM		**0.8072**			
RWD		0.8074			

Mean MAPE values over 30 unique sub-models by model type and dataset. Best model's MAPE in **bold**.

5.4.1 Top five most accurate models

By this MAPE comparison, none of the DL implementations achieves better accuracies than the baseline RW models in this experiment's (1-day horizon) environment. By MAPE score, the five best models are ranked as follows:

1. RWM 0.8072
2. RWD 0.8074 (+ 00,02% error/best)
3. LSTM standard univariate 1.1523 (+ 42,75% error/best)
4. LSTM standard RF strict 1.1639 (+ 44,19% error/best)
5. LSTM optimized RF strict 1.1685 (+ 44,76% error/best)

Chapter 6

Discussion

This chapter is dedicated to discussing the results obtained through the experiments from the previous chapter. Section 6.1 will first discuss the feature selection results. Following this, Section 6.2 will discuss tuning procedure results. The main experiment forecasting results will be talked about in Section 6.3. Finally, the most important observations will be discussed in Section 6.4

6.1 Feature selection results

Before presenting the main forecasting experiments results, results from the pre-occurring procedures will be discussed. This section discusses the results from the feature selection procedure, mainly the RF-FS results.

6.1.1 A preference for the target variable

An interesting discovery from the RF-FS feature selection process is that it favors the same approach as the already built *univariate* dataset. It selects the target variable only, disregarding all other 71 features, favoring the same univariate dataset used by the RW-based models. In essence, the *univariate* is the true *strict-form* of the RF-FS dataset, as the originally intended RF-FS strict approach was only to use whichever features the selection algorithm returned.

The OSEAX preference indicates that the RF algorithm finds the most informational value in the target feature. As presented in the RWM results in Subsection 5.3.1, the last observed target value explains 99.53 % of the

next day's true value (as read by R^2 value). In essence, the point that only 0.47 % is left of the next-day valuation is to be attributed to the remaining 71 features or any other influence outside the scope of this thesis. The RF-FS results may indicate that there are no more potential forecasting performance gains to be retrieved from the remaining features or that it is so small that it cannot be identified.

6.1.2 RF-FS promoting technical indicators

When altering the RF-FS implementation to return the n-highest ranked features instead, it becomes clear that it prefers technical indicators for this 1-day forecasting environment. The *RF top-7* dataset includes all available technical indicators: the *50d MA*, *200d MA*, *RSI*, and *MACD*. Although not explicitly selecting them (only the OSEAX got strictly selected), this can be an explanation for the findings of Kumbure et al., stating that 62 % of features used in stock market forecasting applications consist of technical indicators [50]. They might be better indicators in the short term, supporting that short-term fluctuations in the stock markets are mostly forecasted with technical indicators [17].

6.2 Tuning results

This section discussed the results from the tuning process that commenced on the LSTM and TFT models before training. Several observations are made from the results.

6.2.1 Overfitting LSTMs

From the results presented in Table 5.1, it can be observed each of the LSTM instances adopts quite different architectures and training parameters. The exception is for the optimizer and the date and time feature inclusion, where the latter shows that the LSTM does not benefit much from adding date and time values to its training set in this environment. This is not unexpected, as the `BlockRNNModel`'s (Block) LSTM architecture can not utilize the time and date features as future covariates, which reduces their informational purposes.

It is observed that for the LSTMs trained with the RF-FS datasets, the number of nodes increases while the input depth narrows compared to the *univariate* and *selected* parameters. Considering that the model now has more features to compare, it may not be as necessary to learn from intra-feature

patterns as the univariate dataset, explaining the narrowing input depth. Another explanation may be that the models get overwhelmed by a too-large total input window when additional features increase the input width. Learning patterns over multiple features may also require more nodes, explaining the increment in nodes per layer.

Considering the results for the tuned LSTM implementations, the tuning shows signs of overfitting. This may be due to the too many complex variables and maybe a too small validation set. In hindsight, a less complex tuning procedure would probably be more beneficial in this first instance of the system, while the more advanced tuning scheme used in this thesis would be better suited to future works.

6.2.2 An improved TFT

As seen by the results in Subsection 5.3.4, the tuning greatly improves the TFT's performance. Unlike the LSTM, it does not overfit, at least not to the same extent. There is some consensus over the different TFT-dataset combinations The TFT algorithm tuning results point toward them preferring the same training parameters: smaller batch sizes of 2-4, the Adam optimizer, and a starting learning rate of 0.001 0.0001, which is mostly fixed (no use of LR schedulers except for the *univariate* set).

Regarding date and time features, the different models seem to "make it or break it": the models either chose to include 3-5 or completely exclude them all. The latter is done by the *RF strict* TFT, which is also the worst-performing of the tuned TFT models in the forecasting experiments. This may indicate that it overfits or that the tuning algorithm did not get enough trials to discover the relationship between the time and date features.

Architecturally, there is a lot of variance. The same trend of a deeper input window for the *univariate* model, while it is much more shallow for the datasets with more features. Most models adopt a single-layered architecture, but the number of nodes per hidden layer varies widely. It is possible the tuning algorithms have struggled to find a global optimum and have rather settled in the first encountered local ones. A simpler approach, letting the tuning choose from a set of predefined node counts (e.g. 8, 16, 32, 64), might have been a better approach in this first implementation. The same goes for the continuous processing layer nodes and can possibly be attributed to this implementation not using static covariates. Attention head count seems to increase as the number of features exceeds seven, possibly explained by the more features, the more attention capacity is required.

6.3 Forecasting results

This section discusses the results obtained from the experiments, highlighting the key findings.

6.3.1 RWM: Performs the best

As summarized in Table 5.8 on page 85, the RWM is the best-performing forecasting model in this experiment. This shows that the RW still holds for this short 1-day forecasting environment. The DL implementations do not even come close to their accuracy, achieved with a simple strategy. It also underlines the strength of a buy-and-hold strategy: if nothing can forecast the price, but you know things will increase (unpredictably) over time, it is better to buy and hold the (OSEAX) index [59].

From the RWM results in Subsection 5.3.1, there are several observations. First and foremost, the model's high R^2 value. Since the RW model repeats today's observed valuation, the R^2 value indicates that today's valuation accounts for 99.53% of the next-day OSEAX valuation variation. This leaves just 0.47% for other variables and noise. The high R^2 value is not unexpected, as the day-to-day variation within the OSEAX's valuation is quite tiny, relatively close to yesterday's value. This shows why this simple model can be so hard to beat. An exemplary ML forecasting implementation must be capable of discovering this relationship to forecast over shorter-term horizons accurately, especially over a 1-day horizon. This concludes that the RW is a good benchmark for assessing the DL models' performances and should be included in any respectable ML stock market forecasting approach.

Secondly, by making direct MAPE comparisons, the RW model outperforms Wang et al.s Transformer implementation on the CSI 300 Index (MAPE of 0.9549) with an 18.31% lower error rate [91]. Bhandari et al.'s single-layered LSTM implementation outperforms the RW with a MAPE of 0.7989 when forecasting the S&P 500 index [9], a performance with 1.02% less error than the RW on the OSEAX. A study that vastly outperforms the RW is Wang et al.s' Reservoir Computing models [92], achieving MAPEs between 0.5500-0.7600 for their models, all of which are below the RW's MAPE. In this environment, the simple RW implementation seems capable of competing with heavy-weighting DL models. However, it is essential to note that none of the above-listed studies can directly speak for the OSE and the OSEAX, as they have been conducted on other indices in different periods.

6.3.2 RWD: Drift does not improve accuracy

From the results in Subsection 5.3.2, the RWD slightly underperforms the RW model. These results may indicate that the underlying growth in the OSEAX index may happen too randomly and abruptly to be modeled by stepwise adding, considering a 1-day horizon. In summary, gradually adding mean growth over every step for this experiment's short-term forecasting environment degrades the model's performance. However, this does not indicate worse performance over longer time horizons, which might be more viable applications for this model.

6.3.3 LSTM standard: Most accurate DL model

As seen from the results in Subsection 5.3.3, the LSTM implementation performs the best on the *univariate* feature-reduced dataset. This is also the best DL model-dataset-tuning combination, achieving the third-best MAPE, as seen in Subsection 5.4. Still, it underperforms the RWM with a whole 42,75 % larger MAPE error.

A trend indicating decreased performance per added feature is observed: In second place to the *univariate* 2-feature dataset forecasts comes the 5-feature *RF strict*'s performance. The *RF strict* forecasts being more inaccurate than the univariate trained model indicates that neither extra OSEAX-target variable-related technicals nor the MSCI-W index enhances accuracy. This is interesting, as the MSCI-W could give the model a little "foresight" into the next day. The possible foresight could be expected to happen because of the OSEAX having an earlier closing time, while at the same time, the two are somewhat correlated. The movement in the MSCI-W after the OSEAX has closed could give the model an edge over the RW model by providing the LSTM the possibility of baking the post-close movement of the MSCI-W into the next-day forecast. Nevertheless, this is not being captured (enough) by the model to give it an edge in this experiment.

None of the dataset instances manages to score MAPEs as low as Bhandari et al.'s best LSTM implementation [9] conducted on the S&P 500 index, but again, this is in another index tested over a different time span. It is also noted that both nine feature datasets perform the worst, with the manually selected *Selected* dataset being the decidedly inferior. Referring to the boxplots in Figure 5.4 on page 75, where the selected dataset's poor performance, both in accuracy and precision, skews all the other boxes.

When training the LSTM for a 1-day forecast with nine features, RF-based feature selection outmatches the manual one. This may be explained by the

Selected dataset's high content of longer-term macro variables. At the same time, the *RF top-7* set is comprised of shorter-term indicators, including all the available technicals. This shows the strength of the RF-based feature selection and that it is better suited to identify the most important features in this instance. Further fortifying this is that the RF feature selection only chose the OSEAX target series for the 1-day forecast horizon. With the *univariate* dataset included in the best performing DL model, the standard *univariate* LSTM, it shows that the RF feature selection method selected the best possible feature.

By the total runtime, the *selected* trained model scores the best, probably caused by the model's validation loss reduction tapering off faster. As the model does not find more patterns to learn from, it gets stopped by the pruner earlier than the other dataset-model instances.

In summary, none of the standard LSTM-dataset combinations score metrics better than the RW-based models when 1-day forecasting. At the same time, they require heavier computing resources for a longer duration of time. For this 1-day horizon LSTM experiment, adding additional features, both manually and machine learning selected, makes the accuracy worse. Neither technical nor macroeconomic indicators positively affect the model's accuracy results compared to a univariate data set.

6.3.4 LSTM tuned: Overfits

As discussed in Subsection 6.2.1, it looks like the tuning is overfitting the models to the validation set. Compared to the standard LSTM, the hyperparameter optimization process has pulled the best-performing models backward and improved the previous worst performer instead of enhancing performance overall. When looking at the R^2 value, the difference is relatively slight. This can be attributed to the forecasted target OSEAX index's slight variance over a one-day horizon.

It is not very unexpected to find that the tuning has reduced the performance of most of the forecasts. There are a lot of tunable parameters available in this implementation, posing a risk of overfitting. This is caused by either the 15%-share validation set being too small or the search being too thorough. Also, the algorithms may get stuck in local optima, or the number of parameters may be too many, making the search space too complex. Lastly, there may be too much noise, misguiding the algorithm, resulting in the search algorithm optimizing for noise rather than an underlying pattern.

6.3.5 TFT standard: Suboptimal architecture

In overview, the standard parameter-trained TFT greatly underperforms all the other forecasting models. As seen in Table 5.8, it mostly has more than double the error rate of the LSTM implementations. Among the different datasets, the standard TFT trained on the *RF top-7* dataset scores the best accuracy, while the model trained on the *selected* dataset has the highest precision. The former makes, on average, the best predictions, while the latter is more confident and decisive in its forecast. Both the mentioned datasets contain the most features of the 4, indicating that in this experiment, the TFT prefers more complex datasets.

Being a more advanced model, the TFT may overfit to noise or find patterns in the data that were relevant in the 1990s or 2000s but not in the period of the training set. Most probably, the architecture and training hyperparameters of the standard TFT are suboptimal for this environment because of the improved result obtained with the tuned model, as discussed in the next subsection.

6.3.6 TFT tuned: Closing in on the LSTMs

From the results in Subsection 5.3.4, a drastic improvement can be observed over the standard TFT. In summary, the hyperparameter optimization search has greatly improved all instances. This may indicate that the standard hyperparameters for the TFT, as mentioned, are a mismatch for this environment, hence highlighting the importance of the tuning procedure. The search has not overfitted, at least to the same notifiable extent as for the LSTM models. As seen in Table 5.8, the tuned TFT closes in on the accuracy performance of the LSTM models but does not quite get there. Compared to the RW-based models, it is miles behind, with the best configuration, the tuned TFT-*RF strict* having a 49.48 % higher MAPE. The TFT is not capable of outperforming the LSTM and RW-based models in the 1-day horizon environment.

An interesting observation is that the metrics across the different tuned TFT configurations are quite similar. With all the model-dataset instances scoring such equal metric values, it may indicate that the TFT is capable of extracting the most relevant information while at the same time filtering out irrelevant noise. With the *univariate* and *RF strict* dataset performing the best, it is possible that the features strictly chosen by the RF, or just the OSEAX alone, are the most important in this setting. The extra features added in the *RF top-7* and *selected* dataset may contain noise, but the model

may be capable of identifying this and ignoring these features. This may be attributed to the attention mechanism, and it seems like the tuning has allowed it to show its full potential. Still, this is not enough to lift the TFT's performance to beat the LSTM, let alone compete with the RW.

Finally, the model architectures found by the search are quite computationally expensive. This may be caused by the small batch sizes, the many parameters, and all the required calculations within the complex model network.

6.4 Summary

This section summarizes the findings from the above-discussed results. The findings are consolidated and further discussed to establish why the results came out as they did.

6.4.1 The random walk remains undefeated

The results demonstrate that despite being the simplest model and calculating for about 20 seconds on a simple system, the RWM and RWD stand uncontested in the 1-day OSE/OSEAX forecasting environment. Complex models with long and short-term memory and attention mechanisms get well above 40% more error on their forecasts. Sophisticated hyperparameter searches and "foresight" through future covariates and indices that close later than the target OSEAX neither help. This illustrates how noisy and complex stock index forecasting and economic data can be.

If hypothetically assuming the RW approach is the most accurate forecasting technique possible, one could expect that the DL models' performance would be at a tangent with the RW-based models, as the RW follows a very simple "repeat last days value" instruction. However, the models do not seem capable of learning this pattern.

6.4.2 Why the deep learning models are inaccurate

There may be several reasons why this is happening. Due to the type of data and the problems associated with stock market forecasting discussed in Subsections 2.1.5-2.1.7, the models may overfit to noise in the data. There may be too much noise and too little graspable information for the models to identify the "repeat last" pattern. Additionally, the new information may get

priced into the OSEAX before any algorithms reach the information through the datasets, hence the EMH.

It is possible the patterns of the RW-based models, which may seem obvious to a human reader, are hard to grasp for an ML model in the noisy dataset. To learn the RW's "repeat last observed" method, the models must receive positive feedback if forecasting the same as the same-day target value. Since the OSEAX valuation must be differentiated before being fed to the DL implementations, it becomes much harder to grasp this relationship. One would preferably not differentiate this data, but the early experiments showed that differentiating is necessary for these implementations. Additionally, the pattern may be hard for the model to discover since the data is so noisy, and at the same time, seemingly rewarding patterns emerge elsewhere from this noise during training. These patterns may not necessarily be noise but also outdated patterns that do not foresee the same future OSEAX index movements in the test set as in the training set. Also, the model may forecast something that deviates from the RW method and, at the same time, get a better score than the RW would have for some instances.

6.4.3 Arbitrage and noise

An important note is that the potential for forecasting stock markets may have been arbitraged out of the market, leaving only unpredictable noise. Many studies have found signs of the LSTM (or more capable algorithms) already being used in the markets since the 2000s, with an explosion after 2010 [32]. Maybe all possibilities of achieving good accuracies on stocks and stock indices are priced out by actors in the market. The result is tapering excessive profits for the algorithm utilizing these models in trading strategies. An important question is if this "pricing out" just leaves noise in the markets, making further forecasts harder (or less effective) for the same models. If so, some of the deficiencies of the DL models may be attributed to this.

6.4.4 Additional reasons for the observed performance

In the previous two subsections, the general attributes of why the DL models fail to outcompete the RW implementations have been discussed. However, there may also be more experiment-specific explanations for the poor DL results. These are the following:

The dataset is too small: As DL models are data-intensive, the dataset may still be too short for the selected models. Increasing the length or the granularity may result in better performance.

The dataset is too long: A completely opposite causation may be that the information in the long dataset is outdated, hampering the DL models when forecasting in today's market environment. Markets are dynamic and noisy, so one type of correlation can mean a completely different thing during different time periods. Training on a smaller dataset or splitting a large dataset into many smaller parts, as done by Fischer and Krauss [32], may increase performance. Many well-performing studies discussed in the Related Works (Section 2.5) use a smaller, more recent dataset.

Overfitting tuning procedure: As observed in the LSTM tuning experiment, it shows signs of overfitting. The complicated tuning procedure may cause this. Although the tuning gives the TFT a great performance increase, it may still be overfitting. A simpler, more explicitly defined tuning implementation may increase the performance of the tuned model experiments.

The large buffer: As the exact publication time of some different macroeconomic features is hard to estimate, it was decided to add a large buffer to the data to prevent "look-ahead" bias. This buffer may have been too large, making the DL models incapable of efficiently utilizing its informational value regarding 1-day horizon OSEAX forecasts. In hindsight, it would have been better to remove the buffer and only add and increase it if the results were unexpectedly accurate.

Loss function: This experiment has been limited to the MSE loss. Other loss functions may be more appropriate, such as the MAPE and SMAPE, noted by Hu's paper on the TFT S&P implementation [40]. These loss functions may allow DL models to achieve better results with non-differentiated data as well, due to their size indifference and relativeness (as discussed in Subsection 5.2.3.

Inclusion of datetime in the univariate datasets: It is possible that including the datetime feature does not add any informational value and decreases the models' performances by introducing noise.

Choice of models: This thesis focuses on the DL spectrum within the field of ML. However, it might be that other types of ML implementations may

be better suited for this application, such as tree-based ensemble models. Additionally, many variations of DL implementations may function much better, such as the convolutional neural net-LSTM ensemble models, which have gained lots of attention lately [50].

Missing features: Many features considered important for forecasting recent market environment valuations are not included because of their shorter data history or no availability. Some of these are the *Norwegian consumer confidence index*, *Chicago Board Options Exchange's volatility index*, and the *EUR/NOK exchange rate*.

Chapter 7

Conclusions

This chapter will conclude with the results obtained from the research. Section 7.1 will summarize what was performed in this study. Section 7.2 will formally answer the research questions that were formulated in Section 1.2. Following that, Section 7.3 will elaborate on the main contribution this thesis has given to the field of stock market forecasting. Finally, Section 7.4 will discuss the future work that has emerged from this research.

7.1 Summary

This thesis explores if the state-of-the-art stock market forecasting techniques can outperform the RW when forecasting the OSEAX index, representing the compound valuation of the stocks listed at the OSE. To achieve this, two DL models have been set up against each other: the highly prominent LSTM-based architecture and the recently emergent TFT. In addition, this thesis seeks to test whether said algorithms perform better when training on an extensive and diverse economic dataset.

To explore the selected models' capabilities within this defined system, this thesis presents a new framework for testing OSE-related indices with ML based on a comprehensive 72-feature economic dataset. The framework includes an ADF-stationarity detection routine, an RF-based feature selection process, and a Bayesian TPE-based hyperparameter optimization technique. Finally, it also includes two baseline evaluation models based on the central financial theories of the RW (and EMH) to test forecasting capabilities thoroughly.

Utilizing this framework, the experiment results indicate that neither the

LSTM nor the TFT comes close to the RW-based baseline models. The most precise RW-based model, the RWM, achieves the lowest MAPE of 0.8072. Meanwhile, the best DL model, the LSTM trained on an OSEAX target variable-only dataset, achieves a MAPE of 1.1523, a 42,75 % higher error rate. The research also finds that the TFT is not capable of dethroning the LSTM within this environment. Additionally, experiments show that adding additional features from the large dataset does not increase forecasting performance.

7.2 Answering the research questions

Finally, the research questions formulated in Section 1.2 will be formally answered based on the results and the discussion.

Research question 1

> (1) *Are the state-of-the-art DL models **TFT** and **LSTM** capable of achieving higher accuracies than the **random walk model** when forecasting **short-term OSE valuations**?*

In the 1-day horizon forecasting experiments conducted in this thesis, neither the LSTM nor the TFT can outperform the RWM nor the RWD. The results indicate that the RW and EMH still remain uncontested within this environment for a 1-day horizon.

Research question 2

> (2) *For the **TFT** and **LSTM** DL models forecasting future valuation of the OSE: To what extent does include **economic and technical indicators** enhance performance for these algorithms in this environment?*

For a 1-day forecasting horizon, the results show that the more additional features are added, the worse performance gets. This is another win for the RWM and EMH because of the following:

1. The univariate dataset performs the best in most cases and on all the best models. This shows that, for this environment, the most valuable data to use for 1-day forecasts are past prices - all the others add noise and reduce accuracies. With no other data than the historical prices, strong and semi-stong from EMH remains uncontested.

2. Since none of the DL models can outperform the random walk on the univariate (historical prices) dataset, the weak-form EMH remains unchallenged.

The final answer to the research question is: For a 1-day forecasting horizon, neither economic, technical, nor any other indicators enhance performance; they degrade it.

Research question 3

> (3) *Is the **TFT** architecture capable of **outperforming** the **LSTM** when forecasting future 1-day valuations of the compound OSE?*

The LSTM generally outperforms the TFT when forecasting over a 1-day horizon. There are a few instances where the TFT can beat the LSTM on certain dataset variations, but in most cases, the LSTM implementations have the best accuracy. The best-performing models of the DL algorithms are all based on the LSTM architecture. To answer the question, no, the TFT is not capable of outperforming the LSTM when forecasting future valuations of the compound OSE for a 1-day horizon in this environment.

7.3 Main contributions

During this thesis research, multiple novel contributions have been made to the stock market forecasting field. These are:

- Presenting a preprocessing, RF-FS feature selection, hyperparameter tuning, and performance evaluation framework for thoroughly evaluating stock index (and other financial assets) regression-based forecasting models against the random walk hypothesis using larger economic datasets.
- Provide the first identified attempt at introducing the TFT architecture at forecasting an OSE-based index.
- Prove that the financial hypotheses of RW and EMH remain uncontested in a 1-day OSEAX valuation forecasting environment.
- Present an optimal dataset composition, architecture, and training parameters for both the LSTM and TFT in such an environment.
- Gathered and built an extensive 1-day granularity, 72-feature economic dataset consisting of OSEAX and Norwegian economy-related indicators as features, spanning over 35 years.

- Provided evidence that including additional features in the covered forecasting approaches degrades performance: training on a series of the target OSEAX variable alone produces the best results.

7.4 Future Work

Stock market forecasting is a broad field accompanied by multiple other research areas. Because of this, there are multiple routes to explore further. In the setting of this thesis, the most notable are the following future work:

Applying other ML approaches to the framework: There exists a myriad of different algorithms that remain to be implemented and experimented with, counting both as standalone and in stacked ensemble models. The most interesting approaches may be experimenting with ensemble learning models like the RF and Extreme Gradient Boosting or stacked DL models like a CNN-LSTM system [50].

Longer or shorter forecasting horizons: Experiment with different forecasting horizons other than one day is an alternative route. For shorter ones, high-frequency trading can be applicable, while the longer forecasting horizons can explore the relationship between economic indicators and future OSE index valuations. For this thesis data and framework, the longer-spanning horizons are the most appealing future works. Especially a one-year horizon due to the possible macroeconomic indicator relationships.

Longer or shorter datasets: As discussed in Subsection 6.4.4, the lackluster DL model performance may be caused by the dataset being too long or too short. A too-long dataset may let the models learn old, outdated patterns, while a too-small dataset may have too few samples and lack diversity. Further research into this problem may be conducted to see if it hampers performance. For example, a shorter dataset span can be thoroughly explored by the "study period" implementation (splitting the large dataset into smaller parts) done by Fischer and Krauss [32].

Including other features: As mentioned in Subsection 6.4.4, some features that may be considered important were excluded from the dataset due to a too short time span or being unavailable. This can be solved by conducting further research on how to attain these indicators or by shortening

the span of the overall dataset so that shorter history features like the EUR/NOK rate, VIX, and Norwegian CCI can be included.

A simpler tuning regime: The complex tuning procedure utilized in this thesis may have resulted in overfitting. Simplifying and restraining the search may prevent overfitting and increase performance for the tuned models.

Removing macro data time buffers: A large buffer was added to the less frequently updated macroeconomic data to prevent look-ahead bias. This buffer may have been too large, making the DL models incapable of efficiently utilizing macroeconomic features' informational value for a shorter 1-day horizon OSEAX forecast. Experiment without the buffer is an appealing future work to test if and how much it hampers performance.

Quantile regression: The TFT and many other ML implementations have the ability to perform probabilistic forecasts, or "quantile regression." Conducting experiments with this technique instead of standard point estimation regression or classification may be an interesting approach for further work.

Applying other preprocessing techniques: Using different preprocessing techniques may enhance performance. For example, many stock market forecasting implementations use discrete wavelet transforms to denoise the data prior to model training [9, 57, 50], a method which was not included in this thesis implementation. In addition, using a different loss function may alleviate the need to differentiate the data, as discussed in Subsection 6.4.4.

Bibliography

[1] *About EIA*. U.S. Energy Information Administration. URL: https://www.eia.gov/about/ (visited on 23.3.2023).

[2] *Advantages and Disadvantages of Machine Learning Language*. DataFlair. 2021. URL: https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/ (visited on 6.4.2023).

[3] Charu C. Aggarwal. "An Introduction to Outlier Analysis." In: *Outlier Analysis*. Cham: Springer International Publishing, 2017, pp. 1–34. ISBN: 978-3-319-47578-3. DOI: 10.1007/978-3-319-47578-3_1. URL: https://doi.org/10.1007/978-3-319-47578-3_1.

[4] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. 1st. Springer Publishing Company, Incorporated, 2018. ISBN: 3319944622. DOI: 10.1007/978-3-319-94463-0.

[5] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework." In: *CoRR* abs/1907.10902 (2019). arXiv: 1907.10902. URL: http://arxiv.org/abs/1907.10902.

[6] Wilshire Associates. *Wilshire 5000 Price Index (WILL5000PR)*. Accessed: 2023-04-22. 2023. URL: https://fred.stlouisfed.org/series/WILL5000PR.

[7] George S. Atsalakis and Kimon P. Valavanis. "Surveying stock market forecasting techniques – Part II: Soft computing methods." In: *Expert Systems with Applications* 36.3, Part 2 (2009), pp. 5932–5941. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2008.07.006. URL: https://www.sciencedirect.com/science/article/pii/S0957417408004417.

[8] Mariette Awad and Rahul Khanna. "Machine Learning." In: *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015, pp. 1–18. ISBN: 978-1-4302-5990-9. DOI: 10.1007/978-1-4302-5990-9_1. URL: https://doi.org/10.1007/978-1-4302-5990-9_1.

[9] Hum Nath Bhandari et al. “Predicting stock market index using LSTM.” In: *Machine Learning with Applications* 9 (2022), p. 100320. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2022.100320. URL: https://www.sciencedirect.com/science/article/pii/S2666827022000378.

[10] Leo Breiman. “Random forests.” In: *Machine learning* 45 (2001), pp. 5–32. ISSN: 1573-0565. DOI: https://doi.org/10.1023/A:1010933404324.

[11] Jason Brownlee. *A Gentle Introduction to the Random Walk for Times Series Forecasting with Python*. Accessed: 2023-11-06. 2020. URL: https://machinelearningmastery.com/gentle-introduction-random-walk-times-series-forecasting-python/.

[12] Lukas Budach et al. “The effects of data quality on machine learning performance.” In: *arXiv preprint arXiv:2207.14529* (2022). DOI: https://doi.org/10.48550/arXiv.2207.14529. arXiv: 2207.14529 [cs.DB].

[13] Colin M.L. Burnett. *An example artificial neural network with a hidden layer*. Wikimedia Foundation. Feb. 22, 2011. URL: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg (visited on 8.4.2023).

[14] Dario Caldara and Matteo Iacoviello. *Geopolitical Risk Index (GPR) database*. Accessed: 2023-04-16. 2023. URL: https://www.matteoiacoviello.com/gpr.htm.

[15] Dario Caldara and Matteo Iacoviello. “Measuring Geopolitical Risk.” In: *American Economic Review* 112.4 (Apr. 2022), pp. 1194–1225. DOI: 10.1257/aer.20191823. URL: https://www.aeaweb.org/articles?id=10.1257/aer.20191823.

[16] Chris Chatfield. *Time-series forecasting*. 1st ed. New York: Chapman and Hall/CRC, 2000. ISBN: 9780429126352. DOI: 10.1201/9781420036206.

[17] James Chen. *Technical Indicator: Definition, Analyst Uses, Types and Examples*. Accessed: 2023-11-06. 2021. URL: https://www.investopedia.com/terms/t/technicalindicator.asp.

[18] Yingjun Chen and Yongtao Hao. “A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction.” In: *Expert Systems with Applications* 80 (2017), pp. 340–355. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2017.02.044. URL: https://www.sciencedirect.com/science/article/pii/S0957417417301367.

[19] Corinna Cortes and Vladimir Vapnik. “Support-vector networks.” In: *Machine learning* 20 (1995), pp. 273–297. DOI: https://doi.org/10.1007/BF00994018.

[20] Kent Daniel, David Hirshleifer, and Avanidhar Subrahmanyam. “Investor Psychology and Security Market under- and Overreactions.” In: *The Journal of Finance* 53.6 (1998), pp. 1839–1885. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/117455.

[21] DatabaseCamp. *LSTMs*. https://databasecamp.de/en/ml/lstms. Accessed: 13-Nov-2023. 2023.

[22] Pasquale De Luca. *Corporate finance: fundamentals of value and price*. eng. Springer texts in business and economics. Cham, Switzerland: Springer, 2023. ISBN: 9783031183003. DOI: 10.1007/978-3-031-18300-3. URL: https://ideas.repec.org/b/spr/sptbec/978-3-031-18300-3.html.

[23] David A. Dickey and Wayne A. Fuller. “Distribution of the Estimators for Autoregressive Time Series With a Unit Root.” In: *Journal of the American Statistical Association* 74.366 (1979), pp. 427–431. ISSN: 01621459. URL: http://www.jstor.org/stable/2286348 (visited on 7.11.2023).

[24] Paul Dix. *Time series forecasting methods*. InfluxData. Aug. 25, 2021. URL: https://www.influxdata.com/time-series-forecasting-methods/ (visited on 10.4.2023).

[25] Lucas Downey. *Oslo Stock Exchange (OSL): Meaning, History, Associated Markets*. Ed. by Thomas Brock. https://www.investopedia.com/terms/o/oslobors.asp. Accessed: 2023-11-04. 2022. (visited on 4.11.2023).

[26] Euronext. *Euronext Oslo Børs*. https://www.euronext.com/en/markets/oslo. Accessed: 2023-11-03. 2023.

[27] *Europe Brent Spot Price FOB*. U.S. Energy Information Administration. 2023. URL: https://www.eia.gov/dnav/pet/hist/RBRTED.htm (visited on 30.1.2023).

[28] Eugene F Fama. “Market efficiency, long-term returns, and behavioral finance.” In: *Journal of financial economics* 49.3 (1998), pp. 283–306. ISSN: 0304-405X. DOI: https://doi.org/10.1016/S0304-405X(98)00026-9. URL: https://www.sciencedirect.com/science/article/pii/S0304405X98000269.

[29] Eugene F. Fama. “Efficient Capital Markets: A Review of Theory and Empirical Work.” In: *The Journal of Finance* 25.2 (1970), pp. 383–417. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2325486 (visited on 6.2.2023).

[30] Eugene F. Fama. “The Behavior of Stock-Market Prices.” In: *The Journal of Business* 38.1 (1965), pp. 34–105. ISSN: 00219398, 15375374. URL: http://www.jstor.org/stable/2350752 (visited on 6.2.2023).

[31] Yahoo Finance. *OSEAX historical data*. https://finance.yahoo.com/quote/%5EOSEAX/history?period1=1362528000&period2=1672444800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true. Accessed on March 17, 2023. 2022.

[32] Thomas Fischer and Christopher Krauss. “Deep learning with long short-term memory networks for financial market predictions.” In: *European Journal of Operational Research* 270.2 (2018), pp. 654–669. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2017.11.054. URL: https://www.sciencedirect.com/science/article/pii/S0377221717310652.

[33] Ligita Gaspareniene et al. “Modelling of S&P 500 Index Price Based on U.S. Economic Indicators: Machine Learning Approach.” In: *Engineering Economics* 32 (Oct. 2021), pp. 362–375. DOI: 10.5755/j01.ee.32.4.27985.

[34] Øystein Gjerde and Frode Sættem. “Causal relations among stock returns and macroeconomic variables in a small, open economy.” In: *Journal of International Financial Markets, Institutions and Money* 9.1 (1999), pp. 61–74. ISSN: 1042-4431. DOI: https://doi.org/10.1016/S1042-4431(98)00036-5. URL: https://www.sciencedirect.com/science/article/pii/S1042443198000365.

[35] Isabelle Guyon and André Elisseeff. “An introduction to variable and feature selection.” In: *Journal of machine learning research* 3 (Jan. 2003), pp. 1157–1182. DOI: 10.1162/153244303322753616.

[36] Yuval Noah Harari. *Sapiens: A brief history of humankind*. Harper, 2015. ISBN: 9780062316103. URL: https://books.google.no/books?id=FmyBAwAAQBAJ.

[37] Md Al Mehedi Hasan et al. “Feature selection for intrusion detection using random forest.” In: *Journal of information security* 7.3 (2016), pp. 129–140. DOI: http://dx.doi.org/10.4236/jis.2016.73009.

[38] Julien Herzen et al. “Darts: User-Friendly Modern Machine Learning for Time Series.” In: *Journal of Machine Learning Research* 23.124 (2022), pp. 1–6. URL: http://jmlr.org/papers/v23/21-1177.html.

[39] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory.” In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[40] Xiaokang Hu. “Stock Price Prediction Based on Temporal Fusion Transformer.” In: *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*. 2021, pp. 60–66. DOI: 10.1109/MLBDBI54094.2021.00019.

[41] Jørgen Husby. “Hvordan påvirkes aksjene notert på Oslo Børs av den amerikanske- og den europeiske sentralbankens kvantitative lettelser i perioden 2005-2015?” MA thesis. Nord University, 2019. URL: https://nordopen.nord.no/nord-xmlui/handle/11250/2619750?show=full.

[42] Rob J Hyndman and George Athanasopoulos. *Some Simple Forecasting Methods*. Accessed: 2023-11-06. 2023. URL: https://otexts.com/fpp2/simple-methods.html.

[43] Bank for International Settlements. *BIS Residential Property Price database*. Accessed: 2023-04-24. Copyright, 2016, Bank for International Settlements (BIS). Retrieved from FRED, Federal Reserve Bank of St. Louis. 2023. URL: http://www.bis.org/statistics/pp.htm.

[44] Investing.com. *Commodities database*. Accessed: 2023-04-17. 2023. URL: https://www.investing.com/commodities/.

[45] Abhinav Jain et al. “Overview and Importance of Data Quality for Machine Learning Tasks.” In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 3561–3562. ISBN: 9781450379984. DOI: 10.1145/3394486.3406477. URL: https://doi.org/10.1145/3394486.3406477.

[46] Sujata Kapoor and Jaya M. Prosad. “Behavioural Finance: A Review.” In: *Procedia Computer Science* 122 (2017). 5th International Conference on Information Technology and Quantitative Management, ITQM 2017, pp. 50–54. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.11.340. URL: https://www.sciencedirect.com/science/article/pii/S1877050917325693.

[47] Ku Chhaya A Khanzode and Ravindra D Sarode. "Advantages and disadvantages of artificial intelligence and machine learning: A literature review." In: *International Journal of Library & Information Science (IJLIS)* 9.1 (2020), p. 3. DOI: https://doi.org/10.17605/OSF.IO/GV5T4.

[48] Kathleen M. Kingsmore et al. "An Introduction to Machine Learning and Analysis of Its Use in Rheumatic Diseases." In: *Nature Reviews Rheumatology* 17.12 (2021), pp. 710–730. DOI: 10.1038/s41584-021-00708-w. URL: https://doi.org/10.1038/s41584-021-00708-w.

[49] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Vol. 4. Prentice Hall New Jersey, 1995. URL: https://api.semanticscholar.org/CorpusID:46622061.

[50] Mahinda Mailagaha Kumbure et al. "Machine learning techniques and data for stock market forecasting: A literature review." In: *Expert Systems with Applications* 197 (2022), p. 116659. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.116659. URL: https://www.sciencedirect.com/science/article/pii/S0957417422001452.

[51] Jonathan Law. *A Dictionary of Finance and Banking*. Oxford University Press, 2018. ISBN: 9780191831430. DOI: 10.1093/acref/9780198789741.001.0001. URL: https://www.oxfordreference.com/view/10.1093/acref/9780198789741.001.0001/acref-9780198789741.

[52] Bryan Lim et al. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting." In: *International Journal of Forecasting* 37.4 (2021), pp. 1748–1764. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0169207021000637.

[53] Kian-Ping Lim and Robert Brooks. "The evolution of stock market efficiency over time: A survey of the empirical literature." In: *Journal of Economic Surveys* 25.1 (2011), pp. 69–108. DOI: https://doi.org/10.1111/j.1467-6419.2009.00611.x.

[54] Craiyon LLC. *AI Generated Image*. Accessed: 2023-04-09. 2023. URL: https://www.craiyon.com/.

[55] Christoph Lohrmann and Pasi Luukka. "Classification of intraday S&P500 returns with a Random Forest." In: *International Journal of Forecasting* 35.1 (2019). Special Section: Supply Chain Forecasting, pp. 390–407. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2018.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0169207018301481.

[56] Lollixzc. *AI hierarchy*. Wikimedia Foundation. Apr. 3, 2023. URL: https://commons.wikimedia.org/wiki/File:AI_hierarchy.svg (visited on 8.4.2023).

[57] Henrik Lund and Jonas Løvas. "Employing Deep Learning for Stock Return Prediction on the Oslo Stock Exchange." MA thesis. Norwegian School of Economics, 2018. URL: http://hdl.handle.net/11250/2586263.

[58] Macrotrends. *Commodities database*. Accessed: 2023-04-16. 2023. URL: https://www.macrotrends.net/charts/commodities.

[59] B.G. Malkiel. *A Random Walk Down Wall Street: The Time-tested Strategy for Successful Investing*. Business book summary. W.W. Norton, 2007. ISBN: 9780393062458. URL: https://books.google.no/books?id=_0LM5sH5FhEC.

[60] Burton Malkiel, Sendhil Mullainathan, and Bruce Stangle. "Market efficiency versus behavioral finance." In: *Journal of Applied Corporate Finance* 17.3 (2005), pp. 124–136. URL: https://ssrn.com/abstract=3396133.

[61] Stephen Marsland. *Machine Learning: An Algorithmic Perspective, Second Edition*. 2nd. Chapman & Hall/CRC, 2014. ISBN: 1466583282. DOI: https://dl.acm.org/doi/10.5555/2692349.

[62] J.B. Maverick. *Key Indicators for Following the Stock Market and Economy*. Investopedia. Sept. 18, 2022. URL: https://www.investopedia.com/ask/answers/032415/what-are-most-common-market-indicators-follow-us-stock-market-and-economy.asp (visited on 6.4.2023).

[63] Wes McKinney. "Data Structures for Statistical Computing in Python." In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[64] Thorvald G. Moe, Jon A. Solheim, and Bent Vale. *The Norwegian Banking Crisis*. Accessed: 2023-11-06. Norges Bank, 2004. URL: https://www.norges-bank.no/contentassets/ed5dd397dce345338046a22c7e07f959/hele_heftet.pdf?v=03/09/2017122240.

[65] MSCI. *MSCI World Index*. Accessed: 2023-11-06. 2023. URL: https://www.msci.com/documents/10199/178e6643-6ae6-47b9-82be-e1fc565ededb.

[66] Randi Næs, Johannes A Skjeltorp, and Bernt Arne Ødegaard. *What factors affect the Oslo Stock Exchange?* Norges Bank, 2009. ISBN: 978-82-7553-530-4. URL: http://hdl.handle.net/11250/2497617.

[67] Robert Nau. *Statistical forecasting: notes on regression and time series analysis*. Accessed: 2023-11-06. 2023. URL: https://people.duke.edu/~rnau/411rand.htm.

[68] *Norges Bank's datatorg*. Norges Bank. 2023. URL: https://app.norges-bank.no/query/index.html#/no/ (visited on 24.3.2023).

[69] Kofi O Nti, Adebayo Adekoya, and Benjamin Weyori. "Random forest based feature selection of macroeconomic variables for stock market prediction." In: *American Journal of Applied Sciences* 16.7 (July 2019), pp. 200–212. DOI: 10.3844/ajassp.2019.200.212.

[70] Bernt Arne Odegaard. *Bernt Arne Odegaard's Website*. 2023. URL: https://ba-odegaard.no/ (visited on 2.3.2023).

[71] Bernt Arne Odegaard. *Norwegian Financial Data from B.A. Odegaard*. https://ba-odegaard.no/financial_data/index.html. Accessed: 02.03.2023.

[72] OECD. *Main Economic Indicators - complete database*. Accessed: 2023-04-22. Copyright, 2016, OECD. Reprinted with permission. Retrieved from FRED, Federal Reserve Bank of St. Louis. 2023. DOI: https://dx.doi.org/10.1787/data-00052-en.

[73] University of Oslo. *Fox – High Performance Computing cluster for Educloud Research users*. Accessed: 2023-11-06. 2023. URL: https://www.uio.no/english/services/it/research/hpc/fox/.

[74] *Oslo Børs All-Share Index Factsheet*. Euronext Oslo. 2022. URL: https://live.euronext.com/sites/default/files/documentation/index-fact-sheets/Oslo_Bors_All-share_Index_Factsheet.pdf (visited on 22.2.2023).

[75] Robert Parrino. *Fundamentals of corporate finance*. eng. 2nd ed. Place of publication not identified: Wiley, 2012. ISBN: 9781118213759. URL: https://books.google.no/books?id=cewbAAAAQBAJ.

[76] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *CoRR* abs/1912.01703 (2019). arXiv: 1912.01703. URL: http://arxiv.org/abs/1912.01703.

[77] Theophilos Papadimitriou Periklis Gogas. "Machine Learning in Economics and Finance." In: *Computational Economics* 57 (2021), pp. 1–4. DOI: https://doi.org/10.1007/s10614-021-10094-w.

[78] Alexander Andersen Sandvik and Lars Røberg Følgesvold. "Causal relations between stock market returns and macroeconomic variables: cointegration evidence from the Norwegian stock market." MA thesis. NHH Norwegian School of Economics, 2016. URL: https://openaccess.nhh.no/nhh-xmlui/handle/11250/2432672.

[79] Mark R Segal. "Machine learning benchmarks and random forest regression." In: *Technical Report, Center for Bioinformatics Molecular Biostatistics, University of California, San Francisco* (May 2003). URL: https://www.researchgate.net/publication/228861739_Machine_Learning_Benchmarks_and_Random_Forest_Regression.

[80] Bloomberg Professional Services. *Data*. Accessed: 2023-11-06. 2023. URL: https://www.bloomberg.com/professional/product/data/.

[81] Martin Sewell. "History of the efficient market hypothesis." In: *Rn* 11.04 (2011), p. 04. URL: http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/images/Research_Student_Information/RN_11_04.pdf.

[82] Shveta Singh and Surendra S Yadav. *Security Analysis and Portfolio Management: A Primer*. eng. 1st Edition 2021. Classroom Companion: Business. Singapore: Springer, 2021. ISBN: 9789811625190. URL: https://link.springer.com/book/10.1007/978-981-16-2520-6.

[83] Federal Reserve Bank of St. Louis. *FRED - Federal Reserve Bank of St. Louis Economic database*. Accessed: 2023-04-10. 2023. URL: https://fred.stlouisfed.org/categories.

[84] *Statistics Norway's StatBank database*. Statistics Norway. 2023. URL: https://www.ssb.no/en/statbank/ (visited on 27.3.2023).

[85] Paweł Stężycki. *Using AI in Finance? Consider These Four Ethical Challenges*. Netguru. June 15, 2021. URL: https://www.netguru.com/blog/ai-in-finance-ethical-challenges (visited on 6.4.2023).

[86] Øyvind Svarttjernet and Joachim Ulsrud. "Makroøkonomiske faktorers påvirkning på Oslo Børs." Accessed: 2023-08-22. MA thesis. NHH Norwegian School of Economics, 2016. URL: https://openaccess.nhh.no/nhh-xmlui/bitstream/handle/11250/2407191/masterthesis.PDF?sequence=1.

[87] TITLON Financial Database. *OSEAX historical data*. https://titlon.uit.no/download.php?p=getdata&d=equityindex&t=stocks. Accessed: 02.03.2023.

[88] Unadkat, Ciocoiu, and Medsker. "Chapter 1: Introduction." In: *Recurrent Neural Networks: Design and Applications*. CRC Press, 1999. ISBN: 9781420049176. URL: https://books.google.no/books?id=ME1SAkN0PyMC.

[89] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697. URL: https://dl.acm.org/doi/book/10.5555/1593511.

[90] Ashish Vaswani et al. "Attention is All you Need." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[91] Chaojie Wang et al. "Stock market index prediction using deep Transformer model." In: *Expert Systems with Applications* 208 (2022), p. 118128. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118128. URL: https://www.sciencedirect.com/science/article/pii/S0957417422013100.

[92] Wei-Jia Wang et al. "Stock market index prediction based on reservoir computing models." In: *Expert Systems with Applications* 178 (2021), p. 115022. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115022. URL: https://www.sciencedirect.com/science/article/pii/S0957417421004632.

[93] Shuhei Watanabe. "Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance." In: *arXiv preprint arXiv:2304.11127* (2023). URL: https://doi.org/10.48550/arXiv.2304.11127.

[94] J.M. Wooldridge, M. Wadud, and J. Lye. *Introductory Econometrics: Asia Pacific Edition with Online Study Tools 12 Months*. Cengage Learning Australia, 2016. ISBN: 9780170350839. URL: https://books.google.no/books?id=wXdLDwAAQBAJ.

Appendix A

Additional Information

A.1 Data sources

TITLON Financial Database: A financial database for Norwegian academic institutions. It contains large amounts of data on stocks, bonds, indices, and more [87].

Yahoo Finance: A website that provides financial news, data, and insights for investors and traders. It offers a wide range of financial information and covers a variety of financial markets [31].

Bernt Arne Ødegaard: A Norwegian finance professor at the University of Stavanger. His financial research and related data are openly available through his personal website [70].

U.S. Energy Information Administration: A U.S. government agency responsible for energy information. They openly provide a wide range of related data on their website [1].

Norges Bank: The Central Bank of Norway. It is responsible for promoting economic stability within the country through monetary policy. They offer data through their database *Norges Banks datatorg* [68].

Statsistisk Sentralbyrå (SSB): The Norwegian central government agency for official statistics and the country's national statistical institute. SSB's data is available through their database *Statbank* [84].

Bloomberg: Bloomberg Professional Services provides an extensive financial database, processing 200 billion data points daily [80].

OECD: The Organization for Economic Cooperation and Development's iLibrary contains seven billion data points across 44 databases, including economic, employment, and financial data [72].

Dario Caldara and Matteo Iacoviello: These two researchers provide an open database for their geopolitical risk indices that reads the sentiment from the electronic archives of 10 global-spanning newspapers [14].

Macrotrends: A research platform that contains data for stocks, commodities, precious metals, oil, gas, and global metrics [58].

Investing.com: Investing.com is a financial markets platform that offers real-time data, quotes, charts, financial tools, breaking news, and analysis across 250 exchanges worldwide [44].

BIS: The Bank for International Settlements (BIS) is an international financial institution that aims to ensure monetary and financial stability through international cooperation [43].

Wilshire: Wilshire is a global advisory company specializing in investment solutions, consulting services, technology solutions, and market indexes [6].

FRED: The Federal Reserve Economic Data (FRED) is an online database that contains economic time series data from numerous national, international, public, and private sources [83].

A.2 Front page illustration:

The front page illustration image is generated with Craiyon, an AI image-generating model related to the DALL-E mini from OpenAI [54].

Date generated: 09.04.2023

Prompt: "Something that depicts deep learning technology deployed in the stock markets. Include a candlestick chart."

Appendix B

Additional Tables

Table B.1
ADF critical values

	1%	5%	10%
Critical Values	-3.43	-2.86	-2.57

Table B.2
ADF test results

Feature	ADF statistic	P-value	Stationary
OSEAX	-0.75	0.83	false
MSCI-W	-1.60	0.48	false
WILL5000	-1.25	0.65	false
GDP	1.93	1.00	false
CAB	-1.31	0.63	false
Trade	-1.49	0.54	false
CPI	-0.80	0.82	false
PPI	2.09	1.00	false
Production	-2.37	0.15	false
Wages	2.39	1.00	false
Unemployed	-1.98	0.30	false
Vacancies	-0.77	0.83	false
C govt.	2.30	1.00	false
C priv.	2.02	1.00	false
RTI	0.56	0.99	false

Table B.2 continued from previous page

Feature	ADF statistic	P-value	Stationary
New cars	-1.64	0.46	false
M0	-3.06	0.03	true
M1	1.19	1.00	false
M2	2.94	1	false
M3	2.85	1	false
I-reserves	-1.02	0.74	false
Policy rate	-2.20	0.20	false
3m NIBOR	-2.21	0.20	false
3m T-bill	-2.23	0.20	false
6m T-bill	-2.23	0.20	false
12m T-bill	-2.30	0.17	false
3y G-bond	-2.17	0.22	false
5y G-bond	-2.08	0.25	false
10y G-bond	-2.01	0.28	false
D public	6.37	1	false
D corporate	3.69	1	false
D households	6.67	1	false
D cent. govt.	-0.46	0.90	false
D mun. govt.	5.65	1	false
Home price	2.31	1.00	false
Build costs	3.48	1	false
Home starts	-4.30	0.00	true
Crude oil	-1.22	0.67	false
Natural gas	-2.61	0.09	false
Gold	2.56	1.00	false
Copper	-1.16	0.69	false
Lumber	-3.56	0.01	true
CCI EU	-3.23	0.02	true
BCI NOR	-5.71	0	true
10y/3m NOR	-2.41	0.14	false
10y/3m U.S.	-2.26	0.19	false
Bankruptcies	-5.47	0	true
GPR	-7.03	0	true
GPR 7d MA	-7.11	0	true
GPR 30d MA	-6.33	0	true
GPRA	-7.13	0	true
GPRT	-6.62	0	true
GPR NOR	-7.83	0	true

Table B.2 continued from previous page

Feature	ADF statistic	P-value	Stationary
GPRH NOR	-8.48	0	true
NOK-TWI	-3.01	0.03	true
USD/NOK	-2.09	0.25	false
GBP/NOK	-1.65	0.45	false
SEK/NOK	-2.11	0.24	false
DKK/NOK	-2.90	0.05	true
JPY/NOK	-1.97	0.30	false
CAD/NOK	-3.50	0.01	true
50d MA	-1.05	0.74	false
200d MA	-1.31	0.63	false
RSI	-10.15	0	true
MACD	-14.29	0	true
Year	-0.21	0.94	false
Month	-8.29	0	true
Day	-21.54	0	true
Week	-8.54	0	true
Day/week	-2.16M	0	true
Holidays	-11.11	0	true
Datetime	35.31	1	false