UNIVERSITY OF TURKU
Turku School of Economics

LSTM-Based Stock Index Forecasting

Prediction Accuracy, Trading Performance, and Interpretability

Finance,
Department of Accounting and Finance
Master's thesis

Author:
Anton Salonen

Supervisor:
Prof. Luis Alvarez

26.3.2026
Turku

Student’s statement regarding the use of Artificial Intelligence (AI) for preparing and/or writing this thesis:

☐ **I have not used any AI-based tools.**

☒ **I have used AI-based tools**. Their use is documented in the Appendix. The AI tools were used in a way that complies with academic integrity guidelines.

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin Originality Check service.

Master's thesis

Subject: Finance
Author: Anton Salonen
Title: LSTM-Based Stock Index Forecasting: Prediction Accuracy, Trading Performance, and Interpretability
Supervisor: Prof. Luis Alvarez
Number of pages: 59 pages (+ appendices 8 pages)
Date: 26.3.2026

Abstract

This study examines the effectiveness of Long Short-Term Memory (LSTM) networks in forecasting stock index prices and evaluates whether such forecasts can be translated into an economically meaningful trading strategy. Using daily data from the OMXH25 index from 2010–2024 and a set of macroeconomic variables, a LSTM model is trained to predict next-day closing prices. The performance of the LSTM model is compared against a traditional autoregressive integrated moving average (ARIMA) benchmark, both in terms of statistical forecasting accuracy and economic performance in a rule-based trading framework.

The empirical results indicate that the LSTM model does not outperform the ARIMA benchmark in forecasting accuracy. Furthermore, while the LSTM-based trading strategy generates higher returns than a buy-and-hold benchmark in a transaction cost free setting, the advantage disappears once transaction costs are introduced. Under such conditions, the strategy fails to produce economically meaningful excess returns in comparison to the buy-and-hold benchmark. The findings suggest that applying machine learning models into financial modelling does not necessarily translate into profitable trading opportunities, particularly when transaction costs are considered.

To enhance interpretability, SHapley Additive exPlanations (SHAP) are applied to the trained LSTM model. The analysis reveals that the model relies primarily on recent lagged values of the index, while the additional macroeconomic variables provide limited incremental predictive value. This indicates that the model predominantly captures short-term autoregressive patterns rather than complex relationships across explanatory variables.

Overall, the results suggest that, in the context of the OMXH25 index, the increased complexity of LSTM models does not yield superior economic performance compared to simpler benchmark models. The study highlights the importance of evaluating forecasting models within realistic trading frameworks and contributes to the literature by combining forecasting performance, trading evaluation, and model interpretability within a unified empirical setting.

Keywords: Long Short-Term Memory, LSTM, Financial forecasting, SHAP, Predictability of the stock market

Pro gradu -tutkielma

Oppiaine: Rahoitus
Tekijä: Anton Salonen
Otsikko: LSTM-Pohjainen Osakeindeksin Ennustaminen: Ennustetarkkuus, Kaupankäynti ja Tulkittavuus
Ohjaaja: Prof. Luis Alvarez
Sivumäärä: 59 sivua (+ liitteet 8 sivua)
Päivämäärä: 26.3.2026

Tiivistelmä

Tässä tutkimuksessa tarkastellaan Long Short-Term Memory (LSTM) -neuroverkkojen tehokkuutta osakeindeksien hintojen ennustamisessa sekä arvioidaan, voiko sijoittaja hyödyntää kyseisiä ennusteita luomalla niiden pohjalta yksinkertaisen kaupankäyntistrategian. Tutkimuksessa hyödynnetään päivittäistä OMXH25-indeksin hintadataa vuosilta 2010–2024 sekä joukkoa makrotaloudellisia muuttujia ja luodaan LSTM-malli, jota opetetaan ennustamaan seuraavan päivän päätöskurssia. LSTM-mallin suorituskykyä verrataan perinteiseen autoregressivie integrated moving average (ARIMA) -malliin sekä ennustetarkkuuden että kaupankäyntistrategian tuottavuuden avulla.

Tulokset osoittavat, että LSTM-malli ei saavuta ARIMA-mallia parempaa ennustetarkkuutta, eikä täten pärjää ARIMA pohjaiselle kaupankäyntistrategialle, kun kaupankäyntikustannuksia ei huomioida. LSTM-pohjaisen kaupankäyntistrategian havaitaan tuottavan korkeampia tuottoja ja riskikorjattuja tuottoja kuin “osta ja pidä” -strategia tilanteessa, jossa kaupankäyntikustannuksia ei huomioida. Ilmiö kuitenkin katoaa, kun tarkastelussa huomioidaan kustannukset, jolloin strategia ei kykene tuottamaan taloudellisesti merkittävää etua verrattuna “osta ja pidä” -strategiaan. Tulokset viittaavat siihen, että koneoppimismallien soveltaminen rahoitusmarkkinoilla ei välittömästi johda kannattaviin kaupankäyntistrategioihin, erityisesti kun tarkastelussa huomioidaan kaupankäyntikustannukset.

Mallin tulkittavuuden parantamiseksi tutkimuksessa hyödynnetään SHapley Additive exPlanations (SHAP) menetelmää. Analyysi osoittaa, että LSTM-malli perustaa ennusteensa pääasiassa osakeindeksin tuoreimpiin arvoihin, kun taas tarkastelun muiden muuttujien tuoma lisäarvo ennustamisessa jää hintadataan verratessa vähäiseksi. Tämä viittaa siihen, että malli oppii ensisijaisesti lyhyen aikavälin rakenteita hintadatassa sen sijaan, että se löytäisi monimutkaisempia muuttujien välisiä riippuvuuksia.

Kokonaisuutena tulokset osoittavat, että OMXH25-indeksin ja tutkimuksen muiden olosuhteiden tapauksessa LSTM-mallien kyky mallintaa monimutkaisia muuttujien välisiä riippuvuuksia ei johda parempaan taloudelliseen suorituskykyyn verrattuna yksinkertaisempiin vertailumalleihin. Tämä tutkimus korostaa ennustemallien arvioinnin merkitystä realistisissa kaupankäyntiolosuhteissa ja tarjoaa kirjallisuuteen oman lisänsä yhdistämällä ennustetarkkuuden, kaupankäynnin ja mallien tulkittavuuden saman aikaisesti empiirisessä tarkastelussa.

Avainsanat: Long Short-Term Memory, LSTM, Taloudellinen ennustaminen, SHAP, Osakemarkkinoiden ennustettavuus

TABLE OF CONTENTS

FIGURES

TABLES

1 Introduction

1.1 Background and motivation

The ability to accurately predict future stock prices has long been a central pursuit for individual investors, financial professionals, and academics alike (Malkiel, 1973, 83). However, historical evidence highlights the volatility and apparent unpredictability of equity returns, raising the question of whether historical market data can reliably serve as an indicator of future performance (Bodie et al., 2021, 148–149). In principle, accurate prediction would enable substantial economic gains, as the ability to anticipate price movements could allow investors to systematically achieve abnormal returns (Malkiel, 1973, 83). The extent to which stock prices are predictable is therefore not only an academically interesting question, but also one of major practical importance for portfolio management, trading strategy design, and risk management.

Questions of market predictability have intrigued researchers for over a century, beginning with Louis Bachelier's theory of speculation in 1900. According to the English translation of Bachelier's paper by Davis and Ethridge (2006, 31–60), Bachelier proposed that stock prices follow a random walk, implying that price changes are unpredictable by definition. Later, Eugene Fama's work on the efficient market hypothesis reached similar conclusions, suggesting that stock prices cannot be forecasted with consistent success because available information is rapidly incorporated into market prices. These findings implied that systematic price prediction would be futile unless compelling evidence to the contrary could be established. (Fama, 1965; 1970;)

Nevertheless, empirical research in the late 1980s began to challenge this strict view of market efficiency. Lo & MacKinlay (1988), Poterba & Summers (1988), and Fama & French (1988) all presented evidence of return predictability through deviations from the random walk hypothesis. Advances in econometrics and empirical analysis have since shown that certain patterns of predictability may exist in stock returns, particularly over specific horizons and market conditions (Campbell et al., 2012, 80). As a result, the question has shifted from whether markets are perfectly unpredictable to whether predictability exists at a level that can be exploited in practice after accounting for real-world trading frictions.

To exploit potential predictability, a wide range of forecasting approaches has been applied, ranging from traditional econometric models to more speculative methods (Malkiel, 1973, 83). Classical econometric approaches are generally designed to capture linear dependencies in financial time series, leaving nonlinear relationships largely unmodelled. However, financial markets are influenced by

complex interactions, suggesting that nonlinear dynamics may play an important role. To account for such nonlinear dependencies, machine learning methods such as artificial neural networks have increasingly been considered. (Campbell et al., 2012, 467–468).

Nonlinear dynamics have become more prominent in research on the behaviour of financial time series since the 2000s (Campbell et al., 2012, 524–525). Building on this development, the present study investigates the use of long short-term memory (LSTM) networks for forecasting stock indices and the application of these forecasts in rule-based trading strategies. LSTM networks, first introduced by Hochreiter & Schmidhuber (1997), are a specialised form of recurrent neural network designed to learn long-term dependencies in sequential data. This makes them well suited for financial time series forecasting, where dependencies may extend across long horizons and may not be captured by traditional linear models (Fischer & Krauss, 2018).

While individual stocks are vulnerable to idiosyncratic risks, stock indices capture the overall performance of markets and economies. This makes index forecasting particularly relevant for portfolio allocation, risk management, and the design of trading strategies. (Bodie et al., 2021, 5–9; 42–45) At the same time, forecasting an index provides a cleaner setting for evaluating predictive models, as index dynamics are less affected by single-firm events such as earnings surprises or takeover announcements. This motivates the focus of this thesis on index forecasting and the evaluation of whether LSTM-based strategies provide practical value relative to traditional benchmark approaches.

1.2 Objective of the study

The primary objective of this study is to evaluate the performance of LSTM networks in forecasting index prices in the Finnish equity market, using one of Finland's main stock indices, the OMX Helsinki 25. In particular, the study examines whether forecasts generated by LSTM models can be translated into a rule-based trading strategy that yields economically meaningful excess returns relative to commonly used benchmark approaches.

This study builds on the findings forecasting framework developed by Bhandari et at. (2022), where promising results in forecasting the prices of the S&P500 index were found. This study applies a similar model framework and dataset in the Finnish market in an attempt to replicate similar forecasting results and a trading and interpretability application to further their approach.

To assess the practical relevance of the forecasts, the performance of the LSTM-based strategy is compared against two benchmark strategies: a passive buy-and-hold strategy and an ARIMA-based

forecasting strategy representing a traditional econometric approach. This comparison provides a framework for evaluating whether the additional complexity of LSTM networks, including their ability to model nonlinear dependencies, leads to improved forecasting accuracy and, more importantly, superior trading performance under realistic market conditions.

A secondary objective of the study is to examine which input variables contribute most to the predictive behaviour of the LSTM model. In addition to index price information, the dataset includes volatility index VSTOXX, the European Central Bank main refinancing rate, the EUR/USD exchange rate, consumer confidence, and the unemployment rate. The study therefore investigates the relative importance of these variables and evaluates whether they provide incremental predictive information beyond past price movements. In doing so, particular attention is given to the interpretability of the model, addressing the commonly cited "black box" nature of LSTM networks.

Building on these objectives, the study aims to provide empirical evidence through the following research questions:

1. Can forecasts generated by LSTM networks be used to construct trading strategies that yield excess returns relative to a buy-and-hold benchmark after accounting for realistic trading conditions?

This question is addressed by implementing a rule-based trading strategy based on the predicted returns of the LSTM model. Trading signals are generated when predicted returns exceed predefined thresholds, and the resulting strategy performance is evaluated using cumulative returns, risk-adjusted measures, and trading activity metrics. To ensure realism, transaction costs are incorporated at multiple levels, allowing the sensitivity of the strategy's profitability to trading frictions to be assessed. The performance is then compared to a passive buy-and-hold benchmark.

2. Do LSTM forecasts outperform those generated by the ARIMA model, and to what extent do differences in model structure and adaptability explain any observed performance differences?

This question is examined by comparing the out-of-sample forecasting performance of the LSTM model with that of ARIMA benchmark models using RMSE, MAPE, and correlation, as well as the economic performance of the corresponding trading strategies. The comparison also reflects differences in model structure and adaptability. The LSTM model captures nonlinear patterns and long-term dependencies but is applied in a static manner after training, making it less responsive to new information. In contrast, the ARIMA model follows a linear structure but is re-estimated in a rolling one-step-ahead framework, allowing it to adapt continuously to recent data. Thus,

performance differences can be attributed to a trade-off between LSTM's flexibility in modeling complex patterns and ARIMA's ability to adjust to changing market conditions.

3. What is the relative contribution of each input variable to the LSTM model's forecasts, and do these variables provide meaningful predictive information beyond past prices?

This question is addressed using SHAP (SHapley Additive exPlanations) values to interpret the predictions of the trained LSTM model. The analysis evaluates both feature importance and the contribution of different time lags within the input sequences. By examining the relative importance of the explanatory variables compared to past price information, the study assesses whether the included macroeconomic and financial variables provide incremental predictive value.

Overall, the study seeks to assess not only the statistical performance of LSTM models, but also their economic usefulness and interpretability within a realistic financial forecasting setting. It contributes to the existing literature on stock market forecasting by combining insights from prior research in computer science on LSTM model architectures, such as Bhandari et al. (2022) and Yadav et al. (2020), with an application to trading strategies to evaluate whether these models can generate economically meaningful returns. Similar approaches at the individual stock level have been employed by Fischer & Krauss (2018) and Akita et al. (2016), who report promising results. Building on this framework, the present study also incorporates model interpretability through SHAP analysis, aiming to shed light on the underlying drivers of the forecasts and to address the "black box" nature of LSTM models, thereby enhancing transparency in the decision-making process.

1.3 Structure of the thesis

Chapter 1 introduces the study by presenting the background and motivation, outlining the research objectives, and formulating the research questions. It also discusses the contribution of the study and its relevance within the existing literature.

Chapter 2 provides the theoretical background and literature review. It begins by discussing key theories related to stock market predictability, including the random walk theory and the efficient market hypothesis, followed by empirical evidence suggesting the existence of predictable patterns in financial markets. The chapter then introduces artificial neural networks, progressing from basic models to more advanced architectures, with a particular focus on Long Short-Term Memory (LSTM) networks. Finally, prior research on the application of LSTM models in financial forecasting and trading strategies, as well as approaches to model interpretability, is reviewed.

Chapter 3 presents the data and methodology employed in the study. It describes the dataset, the selection and preprocessing of input variables, and the normalization procedures. The forecasting framework is then introduced, including the configuration and training of the LSTM model, as well as the specification of the ARIMA benchmark models. In addition, the chapter outlines the evaluation metrics used to assess forecasting accuracy and introduces the trading strategy framework used to evaluate the economic performance of the forecasts. The implementation of SHAP values for model interpretation is also described.

Chapter 4 reports the empirical results of the study. It first presents the forecasting performance of the models, followed by the results of the trading strategies under different transaction cost assumptions. The chapter also includes an analysis of the SHAP values to provide insight into the behaviour of the LSTM model and the relative importance of the input variables.

Finally, Chapter 5 concludes the thesis by summarizing the main findings and discussing their implications for both theory and practice. The chapter also highlights the limitations of the study and proposes directions for future research.

1.4 Limitations

Despite the insights provided, this study is subject to several limitations that should be acknowledged. First, the LSTM model is estimated once and remains fixed throughout the test period, which may limit its ability to adapt to changing market conditions compared to models that are continuously updated. As a result, part of the observed performance may reflect limitations in model adaptability rather than purely differences in model structure.

Second, the analysis is restricted to the dataset and financial instruments included in this study. In particular, the focus on a single market index, the OMXH25, may limit the generalizability of the findings to other markets, asset classes, or time periods.

Third, the selection of input variables and the chosen model configuration may influence the results. Alternative feature sets, model architectures, or hyperparameter choices could yield different outcomes, and the study does not exhaustively explore all possible configurations due to the computational constraints associated with such analysis. Furthermore, the methodological focus is limited to LSTM networks, and other machine learning or deep learning approaches are not examined.

Fourth, while the trading strategy evaluation provides insight into the economic usability of the forecasts, the analysis is conducted in a simulated environment. Consequently, it does not fully

capture real-world trading conditions, such as liquidity constraints, market impact, or taxation, which may affect actual investment performance.

Finally, although SHAP values are employed to improve the interpretability of the LSTM model, the explanations provided by this method should be interpreted with caution. SHAP values reflect the contribution of input variables to model predictions but do not imply causal relationships between the variables and the underlying financial processes.

The inherent difficulty of forecasting nonstationary time series such as stock index prices should also be noted. Despite this, the study aims to build on the promising results reported by Bhandari et al. (2022) and extend their approach by incorporating a trading strategy and SHAP-based interpretability, thereby providing additional insight into both the economic relevance and underlying drivers of the forecasts.

2 Theoretical background

2.1 Predictability of the stock market

Classical financial theories, such as the random walk theory and the efficient market hypothesis, suggest that stock prices are largely unpredictable and that available information is rapidly incorporated into market prices. Certain predictable patterns may however exist in financial markets, creating potential opportunities for forecasting.

2.1.1 Random walk theory

The random walk theory provides one of the earliest and most influential foundations for understanding how stock prices behave. It originates with Bachelier (1900), who proposed that asset prices evolve according to a stochastic process resembling what is now known as Brownian motion. This introduced the idea that price changes are essentially unpredictable. The theory received further empirical support from Kendall (1953), who examined time series of commodity and stock prices and found that price changes exhibited almost no serial dependence. Based on this near-random behaviour, Kendall (1953) concluded that there appeared to be "no hope" of reliably predicting short-term movements in financial markets, reinforcing the conceptual basis of the random walk.

A further theoretical contribution was provided by Fama (1965), who examined whether stock prices follow a random walk using data from the constituent stocks of the Dow Jones Industrial Average. Fama (1965) concluded that the behaviour of stock prices is broadly consistent with the random walk hypothesis, implying that attempts to systematically forecast price movements are unlikely to yield persistent success. He later integrated these insights into a broader theoretical framework through the Efficient Market Hypothesis (Fama, 1970).

From the random walk perspective, the implications for forecasting are straightforward: if price changes follow a truly random process, no forecasting model should be able to consistently outperform simple naive benchmarks such as assuming tomorrow's price will equal today's. This view dominated financial economics for much of the 20th century. However, by the early 21st century, confidence in the possibility of at least some degree of stock market predictability had grown, supported by accumulating empirical evidence challenging previous beliefs. (Malkiel, 2003)

2.1.2 Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH) forms one of the most notable theories in modern finance and provides a theoretical framework for understanding how information is incorporated into asset

prices (Malkiel, 2003). First formally introduced by Fama (1970), the EMH proposes that financial markets are efficient in the sense that all available information is fully reflected in the prices. In such a market, no trading strategy can systematically earn abnormal returns.

Fama (1970) famously classifies market efficiency into three forms: weak, semi-strong, and strong, based on the information set reflected in prices. In the weak form, prices reflect all information contained in past price data. In the semi-strong form, prices reflect all publicly available data. In the strong form, prices fully incorporate all information, both public and private, implying that even insiders cannot consistently earn abnormal returns.

For true efficiency to exist, Fama (1970) outlines three conditions to be met. First, there must be no transaction costs. Second, all relevant information must be freely and simultaneously available to all market participants. Third, all participants must agree on how new information affects asset prices. While these assumptions do not hold in real-world markets, Fama (1970) emphasizes that a market can still be considered efficient even if these ideal conditions are only partially met.

A substantial amount of research has since been made regarding the EMH. Early studies around the time when Fama first proposed the theory, studies focused on testing whether market prices reflect information as implied by the weak and semi-strong forms of the EMH. These studies found little evidence of predictable patterns in stock prices, reinforcing the belief that abnormal returns cannot be systematically earned through information available to the public.

The weak form of the EMH received early support from Fama (1965), as discussed earlier in Chapter 2.1.1 of this study. Similarly, Fama & Blume (1966) examined the profitability of trading strategies and focused particularly on a method known as the filter rule, where trading decisions are triggered by chosen percentage movements in past prices. They found that, once transaction costs were considered, filter rules failed to outperform a simple buy-and-hold strategy, providing further evidence to the support of the weak form of the EMH.

Ball & Brown (1968) demonstrated that earnings announcements are incorporated into the stock prices rapidly, with a substantial portion of the information reflected in prices prior to the official announcement. As a result, little opportunity remains for investors to earn abnormal returns by trading on earnings news alone, providing empirical support to the semi-strong form of the EMH. This finding was reinforced by Fama et al. (1969), who showed that stock prices adjust to stock split news with an almost immediate effect after the announcement date. Fama et al. (1969) conclude that stock prices adjust very rapidly to new information and thus can be seen as being efficient.

Additional empirical support for the EMH has been provided through studies of mutual fund performance. Jensen (1968) analysed the performance of 115 mutual funds and found no evidence that any individual fund consistently overperformed a buy-and-hold benchmark in a manner that could not be attributed to chance. Importantly, these findings held even before accounting for management fees. Consequently, even if an investor were able to replicate the portfolio holdings of these funds, they would not achieve abnormal returns, a result supporting the EMH.

Later studies have reinforced these conclusions. Malkiel (1995) examined the performance of all existing U.S. equity mutual funds from 1971 to 1991 and very few funds achieved returns exceeding market benchmarks, and that such performance could largely be explained by chance rather than managerial skill. Malkiel (1995) therefore concludes that there was no evidence to contradict the EMH.

Overall, the literature supporting the EMH indicates that publicly available information is rapidly incorporated into market prices, that historical prices offer little predictive power, and that even professional investors struggle to outperform passive benchmarks. However, the emergence of documented anomalies and return predictability in later decades has prompted further investigation. With that said, it is important to keep in mind that evidence of predictability is not evidence against the EMH. A key implication of the EMH is the joint hypothesis problem: market efficiency can only be tested together with a specific asset pricing model. Observed predictability or abnormal returns do not necessarily imply inefficiency, as they may reflect compensation for risk as expected returns and thus also prices depend on both available information and the assumed pricing model. (Fama, 2017, 55–56)

2.1.3 Empirical evidence of predictability

Beginning in the late 1980s, researchers have increasingly identified predictable components in stock prices across different time horizons, asset classes, and market conditions. One of the earliest challenges to the random walk theory and EMH was provided by Lo & MacKinlay (1988), who applied variance ratio tests to U.S. equity returns from 1962 to 1985 and rejected the hypothesis that stock prices follow random walks. Their findings indicated statistically significant positive serial correlation in weekly and monthly returns, suggesting that deviations from randomness could be detected in stock prices.

Evidence of predictability has also been documented over longer investment horizons. De Bondt & Thaler (1985) showed that stocks listed on the New York Stock Exchange between 1926 and 1982,

exhibiting past positive or negative extreme performance tend to experience reversals on the long-term. Poterba & Summers (1988) found evidence of positive serial correlation in stock returns over short periods and negative correlation over longer periods. Their findings were based on data from the NYSE equal- and value-weighted returns from 1926 to 1985 and from 17 other countries to corroborate their findings. These findings contradict the assumption that price changes are independent over time and suggest that predictable features could be found in the behaviour of stock prices.

Over shorter horizons, a contrasting form of predictability has been observed. Jegadeesh & Titman (1993) demonstrated that over the 1965 to 1989, stocks that performed well or poorly over the past 3-to-12-month period tend to continue similar performance in the subsequent period. By conducting a trading strategy buying well performing stocks and selling poorly performing ones, they were able to generate abnormal returns, with best results coming from a 6-month observation and holding periods. This phenomenon is called momentum, and evidence of it has been found globally across asset classes (see for example Asness et al., 2013)

A form of predictability has been identified in the direction of returns rather than their magnitude. Christoffersen & Diebold (2006) show that even when stock returns themselves are difficult to predict, the sign of returns can exhibit systematic dependence, partly driven by time-varying volatility. Building on this, Gu & Peng (2019) provide further empirical evidence that the direction of stock returns is, to some extent, predictable. Using a time-varying probability density framework applied to the Chinese stock market, they find statistically and economically significant out-of-sample predictive ability for return direction. These findings have given way for machine learning based methods to attempt directional forecasting, instead of forecasting the exact price.

Advances in computational power and machine learning have enabled research to explore nonlinear and more intricate forms of predictability that traditional econometric models may fail to capture. Gu et al. (2020) demonstrated that machine learning methods such as neural networks outperform linear benchmarks in return prediction. This suggests that neural network-based methods may be capable of extracting subtle but economically meaningful predictive signals from time series.

Overall, the literature indicates that while financial markets may be efficient in many respects, they are not completely unpredictable. Instead, stock returns exhibit patterns that can give rise to limited amounts of predictability. This study examines whether that predictability is something a Long Short-Term Memory based trading strategy can take advantage of.

2.2 Neural networks

Artificial neural networks are machine learning techniques inspired by the human brain and how it works. In the brain, neurons are connected by pathways that strengthen or weaken as we learn from experiences. Artificial neural networks, mimic this process using computational units called neurons that are connected by adjustable weights. These weights determine how strongly one neuron influences another, allowing the network to process information and identify patterns in data. (Aggarwal, 2023, 15)

2.2.1 The perceptron

The history of artificial neural networks starts with McCulloch & Pitts (1943), who mathematically modelled how natural neurons could function. They proposed neurons as simple binary devices, which could signal either all or nothing. Rosenblatt (1958) expanded on this idea by introducing the perceptron, an artificial neuron capable of learn from data. A perceptron can adjust its weights based on the training examples it receives enabling it to learn logical functions like AND and OR.

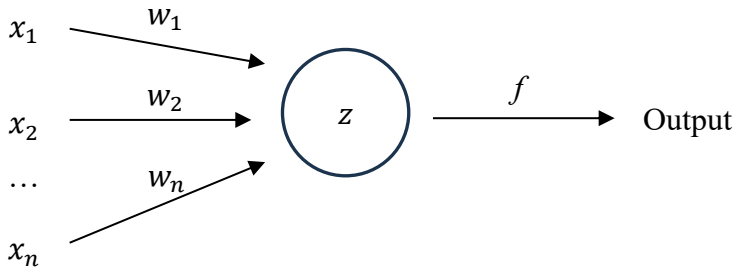


Figure 1. Perceptron model

As illustrated in Figure 1, the perceptron model has multiple binary inputs $x_1, x_2, \dots, x_n$, where n, can be any positive integer. Weights $w_1, w_2, \dots, w_n$ are real numbers, representing the importance the individual input has to the output, determined based on the training data. The neuron then produces a binary output with the activation function f based on the weighted sum of inputs z based on a comparison to the bias value b, determined based on the training data. The perceptron's process can be presented algebraically as:

$$\mathrm{z} = \sum_{j=1}^{n} w_j x_j + b,$$

$$output = f(\mathrm{z}) = \begin{cases} 0\,, \mathrm{z} \leq 0 \\ 1\,, \mathrm{z} > 0 \end{cases}.$$

The perceptron model is not commonly used today as it is not sophisticated enough to model complex problems, but it is the basis which later more complex models are built on (Nielsen, 2015, Chapter 1).

2.2.2 The multilayer perceptron

To overcome the perceptron's inability to model non-linear dependencies, researchers developed the multilayer perceptron, utilizing hidden layers. These are the layers of neurons in the model that are neither the input nor the output layer. Contrary to what its name suggests, the multilayer perceptron does not use the original perceptron as its neuron model. Instead, neurons with alternative activation functions are preferred.

One common choice is the sigmoid neuron, a neuron utilizing the sigmoid activation function. The sigmoid neuron's architecture is otherwise like that of the perceptron's, but both the input and output values can take any value between 0 and 1. The sigmoid neuron also uses a different activation function, the sigmoid function σ. According to Nielsen (2015, Chapter 1), the sigmoid neurons process can be presented algebraically as:

$$z = \sum_{j=1}^{n} w_j x_j + b,$$

$$output = \sigma(z) = \frac{1}{1+e^{-z}}.$$

Another possibility is to utilize the hyperbolic tangent (tanh) activation function in the multilayer perceptron. It is like the sigmoid function but differs in the possible outputs, producing values between -1 and 1. Like the sigmoid, the tanh function is smooth and differentiable rather than binary, allowing neural networks to model continuous, non-linear dependencies. Algebraically, the tanh function can be expressed as: (Bishop, 1995, 126–127)

$$output = tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}.$$

While the multilayer perceptron is a powerful tool for learning non-linear relationships in data, it is designed for static inputs, where observations are treated as independent of one another. However, the focus of this study is on models suitable for forecasting stock index data. Index data is naturally presented as a time series, which evolves over time and is not static. To address problems such as this, recurrent neural networks were developed (Goodfellow et al., 2016, Chapter 10).

2.2.3 Recurrent neural networks

A recurrent neural network (RNN) is a type of neural network designed to process sequential data. Unlike feedforward networks such as the perceptron and multilayer perceptron, which treat each input as independent, RNNs include recurrent connections that allow information to flow across time steps. At each point in time, the network takes the current input and combines it with information stored in its hidden state, which acts as a memory of previous inputs. The hidden state is then updated and stored to be used again at the next step in time. This mechanism allows the RNNs to learn temporal dependencies within data, making RNNs particularly suitable for time series prediction. (Goodfellow et al., 2016, Chapter 10)

RNNs are trained using a method called backpropagation through time, which is an adaptation of the learning process used in standard neural networks. In a feedforward network, the model learns by comparing its predicted outputs with the correct values and then adjusting the parameters to reduce the overall error. In an RNN, the idea is extended to sequences of data by unfolding the network over time. Each time step is treated as one stage in a chain, where the same set of weights is applied repeatedly. During training, the network examines the development of errors over the sequence and adjusts its weights accordingly. This process allows the model to learn how information evolves over time, but it also introduces challenges such as the vanishing and exploding gradient problem when learning long-term relationships. (Goodfellow et al., 2016, Chapter 10)

The vanishing and exploding gradient problem occur during the learning process as the network adjusts the weight parameters over many time steps. Because the same parameters are repeatedly updated with the backpropagation through time method, the gradients indicating how much each weight should change can either become very small or large. On one hand, when the gradients become very small, or vanish, the weight adjustments become negligible, preventing the learning process from effectively learning long-term relationships. On the other hand, when the gradients become very large, or explode, the weight updates become unstable, again disrupting the learning process. These limitations of RNNs motivated researchers to develop more advanced models, capable of learning long-term dependencies. This study utilizes the long short-term memory, which is a model designed to overcome these problems.

2.2.4 Long short-term memory

The long short-term memory, or LSTM network, introduced by Hochreiter & Schmidhuber (1997), was developed to address the limitations of previously developed RNNs. The LSTM is designed to maintain a constant flow of error information over time, thereby preventing the vanishing and

exploding gradient problems. This is achieved through a memory cell structure containing a set of gating mechanisms. These include the input gate, forget gate, and output gate, which together control how information is retained, discarded, or passed forward in time within the network. (Hochreiter & Schmidhuber, 1997)

A LSTM network consists of a series of memory cells arranged in one or more layers. Each memory cell maintains two types of memory simultaneously: a long-term memory, known as the cell state, and a short-term memory, known as the hidden state. The cell state carries information across long time intervals, while the hidden state captures the more immediate patterns from recent inputs, allowing the network to learn both short- and long-term dependencies in sequential data. (Goodfellow et al., 2016, Chapter 10)

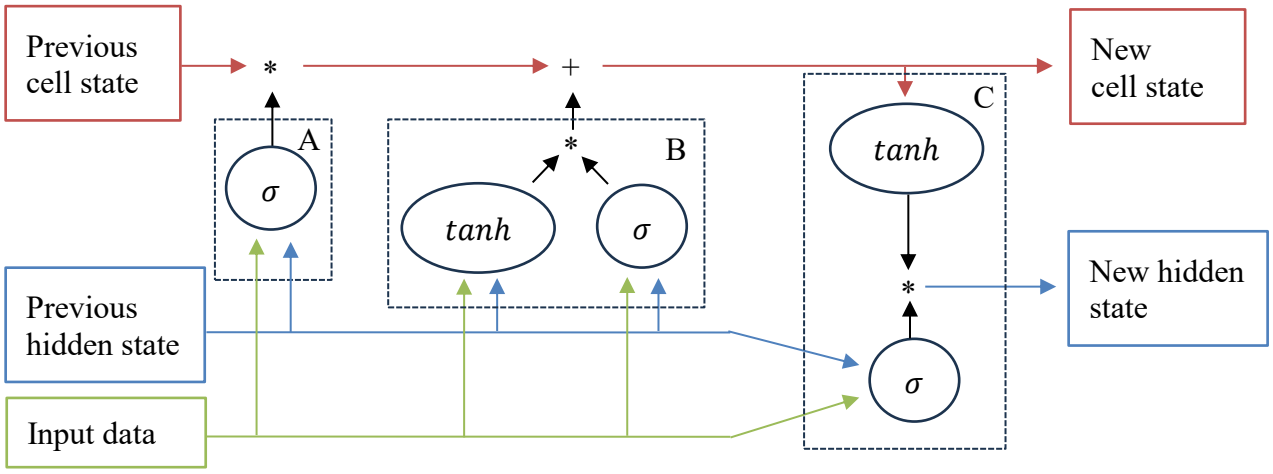


Figure 2. LSTM memory cell

As illustrated in Figure 2, the forget gate (A) is a sigmoid neuron that takes as input both the previous hidden state and the current input data. The input gate (B) consists of two components: a sigmoid neuron and a tanh neuron, both receiving the same inputs as the forget gate. Similarly, the output gate (C) includes a sigmoid neuron with the same inputs, while its tanh neuron operates on the cell state.

The operation of a LSTM memory cell proceeds as follows. In area A, the forget gate determines how much of the previous cell state to retain, based on the previous hidden state and current input. In area B, the input gate processes the same inputs through both a sigmoid and a tanh neuron; their outputs are multiplied to control how much new information is added to the cell state. This contribution is then combined with the retained information to form the updated cell state. Finally,

in area C, the output gate generates the new hidden state by applying a tanh transformation to the cell state and scaling it with the output of the sigmoid neuron. (Goodfellow et al. 2016, Chapter 10)

In this study, data beyond the historical the index price is considered in forecasting future index values. This requires the use of a multivariate LSTM network, which is able to process multiple input variables simultaneously. This way each time step contains a vector of inputs rather than a single value, allowing the network to learn dependencies not only over time but across different variables as well. (Bhandari et al., 2022)

In summary, a LSTM network is a type of neural network designed to learn patterns in sequences, such as time series data, by remembering important information over time while discarding what is less relevant. It does this using a memory cell that keeps track of both long-term information (the cell state) and short-term information (the hidden state). Three gates control this process: the forget gate decides what past information to keep or discard, the input gate determines what new information should be added, and the output gate decides what information to pass on as the result for the current step. By continuously updating and filtering information in this way, the LSTM can capture both short-term changes and long-term trends in the data. In this study, a multivariate LSTM is used, meaning that multiple input variables are processed at each time step, allowing the model to learn relationships not only over time but also between different variables.

2.3 LSTM architecture in financial forecasting

Early applications of LSTM networks to financial forecasting demonstrated their ability to capture dependencies in stock market data. Nelson et al. (2017) applied a LSTM network to predict short-term directional movements in the Brazilian stock market using 15-minute interval data. Their network used multivariate inputs composed of historical prices and technical indicators, outperforming traditional machine learning methods such as the multilayer perceptron and random forests. When the forecasts were applied to trading strategies, the LSTM-based strategy was able to outperform the buy-and-hold benchmark strategy by producing higher returns with lower risk. The authors noted that their results were promising, as the LSTM performed strongly in comparison to other machine learning methods, but it did not consistently outperform all baseline investment strategies in terms of financial returns.

Later research explored how network architecture influences forecasting performance. Yu & Yan (2020) compared LSTM architectures with varying numbers of hidden layers while forecasting six major stock indices in a pure prediction setting. They found that a LSTM with three hidden layers

produced the best predictive performance when compared to architectures ranging from one to eight hidden layers, as measured by directional accuracy and error-based metrics. The authors cautioned that increasing model depth beyond this point led to overfitting, reducing the model's ability to generalize to unseen data and highlighting the importance of careful architectural tuning in LSTM-based financial forecasting.

Similarly, Yadav et al. (2020) examined LSTM architectures in the context of the Indian stock market by varying the number of hidden layers from one to seven. They found that a single hidden layer achieved the best predictive performance, as measured by root mean square error, while also offering easier training and a reduced risk of overfitting due to unnecessary model complexity. The authors noted, however, that for more complex prediction tasks deeper LSTM architectures may still be advantageous in extracting additional predictive structure from the data.

Bhandari et al. (2022) further extended this line of research by applying LSTM networks to forecast the next-day closing price of the S&P 500 index using a multivariate dataset that combined fundamental data, macroeconomic variables, and technical indicators. They compared single-layer and multilayer LSTM architectures and evaluated performance using root mean square error, mean absolute percentage error, and correlation metrics. Across all evaluation measures, the single-layer LSTM consistently outperformed deeper architectures, providing more stable and accurate forecasts. The authors attributed the inferior performance of deeper models to increased model complexity, which reduced out-of-sample generalization and offered no additional predictive benefit in this setting.

Similar to the approach of this study, Siami-Namini et al. (2018) compared the performance of ARIMA models and LSTM networks in forecasting a range of financial and economic time series, including major stock indices. They however, utilized a rolling forecasting framework and evaluated performance with RMSE, finding that LSTM models consistently outperformed ARIMA across all datasets. In particular, the LSTM achieved substantial improvements in forecast accuracy, reducing error rates by approximately 84–87% on average. The authors attribute this superior performance to LSTM's ability to capture nonlinear relationships and long-term dependencies in the data, which are not adequately modelled by the linear structure of the ARIMA model.

In contrast, Kobiela et al. (2022) provide evidence that ARIMA models can outperform LSTM in certain settings. Using NASDAQ stock data and evaluating performance with MAPE and MSE, they find that ARIMA consistently delivers more accurate forecasts than LSTM when only historical price data is used as input, particularly over longer forecasting horizons. The authors attribute this

difference to the limited feature set, noting that LSTM's more complex architecture requires additional input variables to learn effectively, whereas ARIMA is better suited to univariate time series based solely on past prices.

2.4 Applications of LSTM Networks in Trading Strategies

While many studies evaluate LSTM models primarily in terms of predictive accuracy, a smaller body of research has examined whether these forecasts can be translated into economically profitable trading strategies. Several studies have examined how LSTM-based forecasts can be translated into practical trading strategies. One of the most notable studies in this area is by Fischer & Krauss (2018), who applied LSTM networks to predict daily directional movements of all S&P 500 constituent stocks between 1992 and 2015. Their results showed that the LSTM network was able to extract meaningful patterns from stock price data by exploiting temporal dependencies in return sequences and outperformed alternative methods including random forests, deep neural networks, and logistic regression. When translated into a long–short trading strategy, the LSTM-based approach outperformed the market from 1992 to 2009. However, the authors noted that after 2010 the excess returns appeared to have been arbitraged away, rendering the strategy unprofitable after transaction costs and suggesting that the predictive advantage of the model diminished as markets adapted.

Another approach is presented by Akita et al. (2016), who propose a deep learning framework that combines numerical stock market data with textual information from financial news to predict stock prices. The model was evaluated using data from 50 companies listed on the Tokyo Stock Exchange together with news articles from the Nikkei newspaper. In a market simulation based on predicted closing prices, the LSTM-based model generated higher trading profits than several baseline methods, demonstrating that combining textual and numerical information can improve the performance of LSTM-based stock prediction and trading strategies.

Chalvatzis & Hristu-Varsakelis (2020) propose a LSTM-based trading framework that combines price prediction with an event-based trading strategy. The model predicts next-day stock index prices using historical price data within a rolling training window. Instead of relying on simple directional predictions, trading decisions are based on the distribution of predicted returns. Predicted returns are divided into percentile-based bins, and positions are taken only when the predicted return falls within bins that historically generated positive profits. The model is retrained daily using recent observations, and the trading strategy determines both entry and exit conditions based on the predicted return distribution. Tested on major U.S. stock indices, the approach achieved higher cumulative returns than a buy-and-hold benchmark over the 2010–2018 period.

Overall, the reviewed studies demonstrate that LSTM-based models are capable of capturing meaningful temporal patterns in financial time series and often outperform traditional machine learning approaches and classical econometric approaches in terms of predictive accuracy. However, the findings also indicate that improved predictive performance does not necessarily translate into persistent economic profitability, as excess returns may erode over time due to market adaptation and transaction costs. Similar conclusions are reported by Gu et al. (2020), who conduct a large-scale comparative analysis of machine learning methods for predicting equity risk premiums. Using a dataset of nearly 30,000 U.S. stocks and a large set of predictive variables, the authors show that machine learning models, including neural networks, can improve return prediction relative to traditional econometric approaches. However, the authors note that these predictive improvements primarily reflect better measurement of expected returns rather than a direct identification of the underlying economic mechanisms driving asset prices.

Much of the existing research on LSTM-based forecasting in financial markets focuses on evaluating whether such models can extract meaningful information from historical data and achieve accurate predictions. While these studies primarily assess predictive accuracy or trading outcomes, comparatively less attention has been given to understanding the nature of the temporal dependencies learned by these models. This study therefore seeks not only to evaluate predictive performance, but also to identify and interpret the temporal patterns captured by the LSTM and to examine whether these learned dependencies can be meaningfully explained in an economic or market-based context.

A growing strand of research seeks to improve the interpretability of machine learning models used in financial forecasting. Studies such as Kumar et al. (2017) and Giudici et al. (2024) develop methods for explaining the predictions of deep learning models applied to financial time series, while other studies, such as Carbó & Gorjón (2024), integrate interpretability techniques such as SHAP values with LSTM models to identify which variables drive model predictions. These approaches aim to address the “black box” nature of deep learning models and to assess whether the relationships identified by such models correspond to economically meaningful patterns in financial markets.

3 Data and methodology

The selection of explanatory variables in this study follows the choices made by Bhandari et al. (2022), who utilized LSTM networks in forecasting the S&P 500 index. Their framework combines market-based price information with macroeconomic indicators in order to capture both short-term market dynamics and broader economic conditions. A similar approach is adopted in this study for the Finnish equity market.

3.1 Data

All utilized data are listed in Table 1 and motivated after the table in the following subsections.

Table 1. Data

Data	Source
Index price variables	
Opening price	LSEG
Closing price	LSEG
Macroeconomic variables	
Volatility index	STOXX
European Central Bank main refinancing rate	European Central Bank
EUR/USD exchange rate	Bank of Finland
Consumer confidence	Statistics Finland
Unemployment rate	Statistics Finland

A set of price and macroeconomic variables are utilized in the forecasts of this study. The price data is extracted from the OMXH25 index, while the macroeconomic variables are included to provide the model with information on the broader economic and financial environment influencing the behaviour of the stock market. The sample period spans from 1 January 2010 to 31 December 2024.

3.1.1 Price data

The first two variables in Table 1 represent the price data of the OMXH25 index. The opening price corresponds to the index level recorded at the beginning of each trading day, while the closing price represents the level recorded at the end of each trading day. The closing price is used as the target variable in the forecasts of this study.

3.1.2 Macroeconomic data

The second set of variables in Table 1 represent the macroeconomic data, which are intended to capture the broader economic and financial environment influencing stock market behaviour. These variables provide the model with contextual information beyond the index's own price development, enabling it to account for a wider set of economic factors.

The first macroeconomic variable considered in this study is the Euro Stoxx 50 Volatility index VSTOXX. According to Whaley (2009), volatility indices such as the VIX in the United States capture investors' expectations of future price changes and indicate the level of stability in the market.

The second macroeconomic variable is the European Central Bank (ECB) main refinancing rate, which represents the key monetary policy rate in the Eurozone. Ioannidis & Kontonikas (2008) found that monetary policy decisions by European central banks have significant impact on stock market returns, highlighting the relevance of an interest rate variable in forecasting the stock market.

The third macroeconomic variable is the EUR/USD exchange rate, representing the relative value of the euro against the U.S. dollar. While Bhandari et al. (2022) used the U.S. Dollar index to represent global currency dynamics in their S&P 500 forecasts, the EUR/USD rate serves a similar role for Eurozone markets. Previous research, such as, Phylaktis & Ravazzolo (2005) supports a close relationship between exchange rate fluctuations and stock market movements. As an interest rate variable and a currency exchange rate are both used as data, it is of importance to note the interest rate parity condition. condition states that differences in interest rates between countries are reflected in exchange rate movements, implying a close relationship between interest rates and exchange rates (Frenkel & Levich, 1975), and thus might offer overlapping information to the model.

The fourth macroeconomic variable considered is the unemployment rate, which reflects labour market conditions and broader economic activity. Prior studies have shown labour market indicators to be significantly related to stock market movements (see for example Boyd et al., 2005).

The fifth and final macroeconomic variable considered in this study is the Consumer Confidence Indicator, or CCI, published by Statistics Finland. The index measures consumers' confidence toward financial markets and the overall economy. Similar indicators have been found to contain predictive information for equity returns (see for example Fisher & Statman, 2003).

Non-daily data such as ECB main refinancing rate, unemployment rate, and CCI are converted to daily frequency using forward filling to align them with daily financial data so that there is a daily

input value to the LSTM model for all features in the dataset. This measure is additionally made to ensure that the forecasts in this study do not utilize data that would not be available on said day in a real forecasting scenario.

3.1.3 Data selection for final forecasts

All input variables contribute some level of information in the prediction of the index closing price. To avoid redundant information, Pearson correlations between all variables are calculated. If two variables exhibit correlations above 0.80, one of them is excluded from the model. This approach follows Bhandari et al. (2022), who note that highly correlated variables provide overlapping information and may reduce model efficiency without improving predictive performance. A correlation matrix of all input variables is presented in Figure 3.

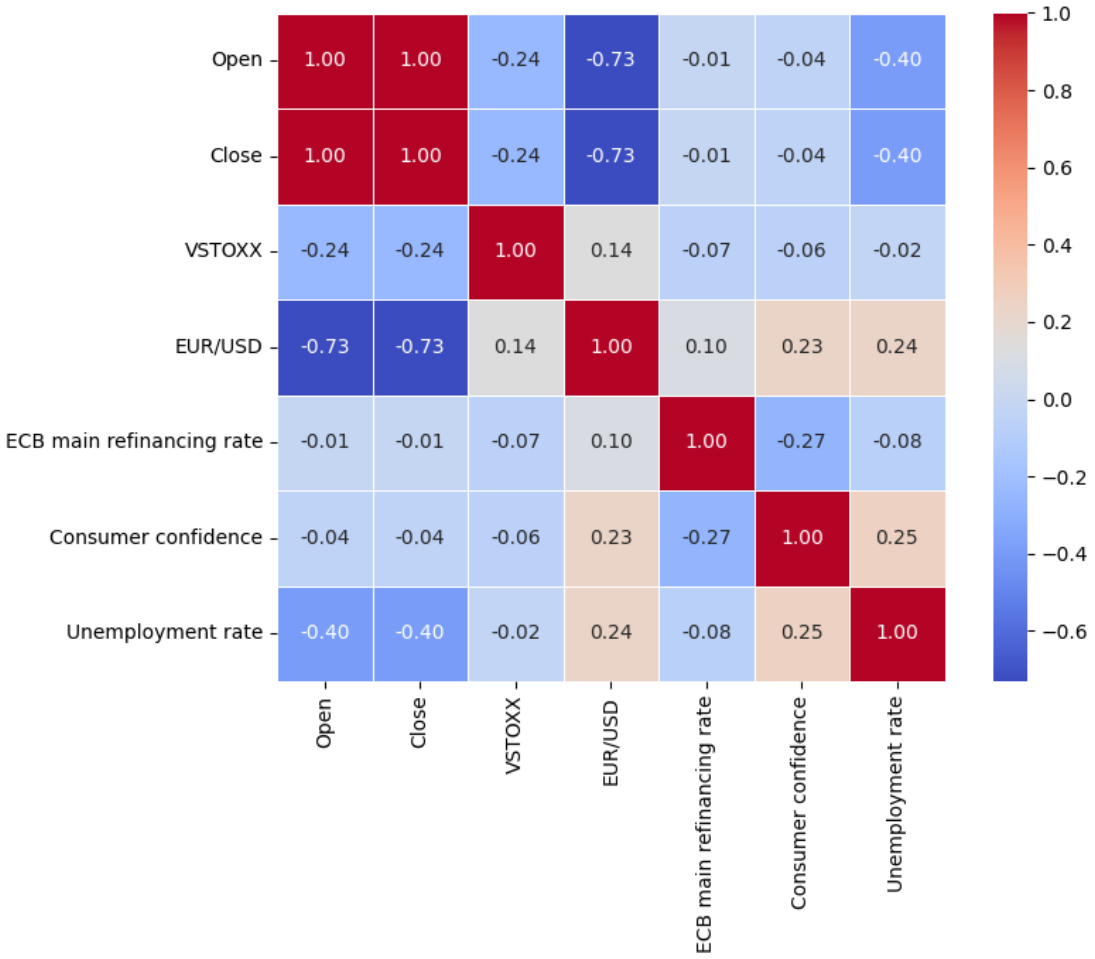


Figure 3. Correlation matrix

As shown in Figure 3, the Pearson correlation between the opening (Open) and closing price (Close) is equal to 1. Consequently, the opening price is removed from the final set of explanatory variables.

After excluding the opening price, none of the remaining variable pairs exhibit correlations exceeding the threshold of 0.80. The final set of input variables used in the forecasts is reported in Table 2.

Table 2 Input data

Data
Index price variables
Closing price
Macroeconomic variables
Volatility index
European Central Bank main refinancing rate
EUR/USD exchange rate
Consumer confidence
Unemployment rate

3.1.4 Data normalization

The data used in this study exhibits substantial differences in scale across variables. To ensure stable training of the LSTM networks, all input variables are scaled using min–max normalization, following the methodology of Bhandari et al. (2022). Min–max normalization is defined as

$$z_t = \frac{x_t - x_{min}}{x_{max} - x_{min}},$$

where x_t denotes the original value of the variable at time t, and x_{min} and x_{max} are the minimum and maximum values computed from the training dataset. This method rescales all variables to the interval [0,1], which helps prevent variables with larger numerical ranges from dominating the learning process.

In the empirical analysis of this study, min-max normalization is implemented using the MinMaxScaler class from the sklearn.preprocessing Python package. The scaling parameters are estimated using the training data following the methodology of Bhandari et al. (2022), to ensure no future information leaks into the training process or forecasting.

3.2 Forecasting framework

Following Bhandari et al. (2022), the dataset is divided into training and testing sets using a time-ordered split, which preserves the chronological structure of the time series. Specifically, the first 80% of the observations are used for model training, while the remaining 20% are reserved for out-of-sample testing. In addition, the last 10% of the training portion is set aside as a validation dataset used for model monitoring during training. This produces a setting where the model is trained on data from 1.1.2010 to 30.12.2021, and the forecasts are produced from 31.12.2021 to 31.12.2024.

This approach ensures that the model is always trained on past observations and evaluated on future observations, thereby preventing information leakage from future data into the training process. As a result, the experimental setup reflects a realistic forecasting scenario, where predictions are made using only information that would have been available at the time of forecasting.

For clarity, Figure 4 illustrates the step-by-step implementation of the LSTM forecasting framework used in this study.

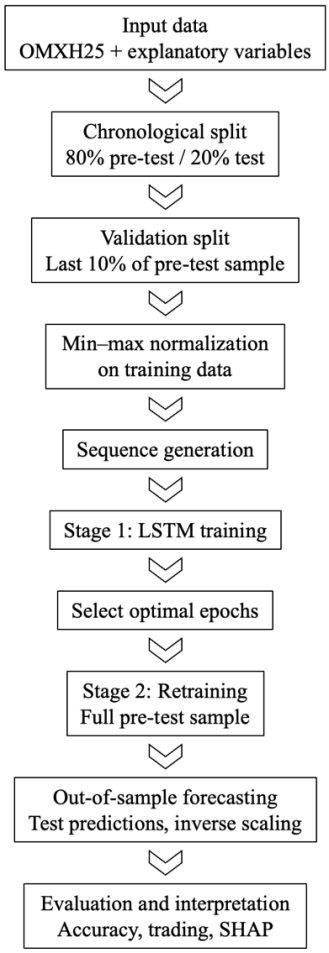


Figure 4. Implementation of the LSTM framework in this study

Figure 4 summarizes the forecasting pipeline implemented for the LSTM model. The procedure begins with the OMXH25 dataset and associated explanatory variables, which are divided into a pre-test sample and an out-of-sample test sample using a chronological split. Within the pre-test sample, the final portion is reserved for validation, ensuring that model training and monitoring are conducted using only past information.

The data are subsequently normalized using min–max scaling based on the training sample and transformed into overlapping sequences of fixed length, which serve as inputs to the LSTM model. Model training is carried out in two stages: first, the optimal number of training epochs is determined using validation data and early stopping, and second, the model is retrained on the full pre-test sample using the selected number of epochs. Finally, the trained model is used to generate out-of-sample forecasts for the test period, which are then evaluated and further analysed.

3.2.1 LSTM configuration

The final LSTM model configuration is selected following the empirical findings of Bhandari et al. (2022). In their study, several single-layer LSTM models were compared using different combinations of neurons, optimizers, learning rates, and batch sizes. The results indicate that a single-layer LSTM with 150 neurons performs best among the tested configurations in terms of root mean square error (RMSE), mean absolute percentage error (MAPE), and correlation.

Since the present study utilizes data similar to that used by Bhandari et al. (2022), their optimal configuration was adopted as the baseline model. Accordingly, the LSTM architecture consists of a single layer with 150 neurons and was trained using the Adagrad optimizer with a learning rate of 0.01 and a batch size of 16. As this configuration produced unstable results with the dataset used in this study, the learning rate was reduced from 0.01 to 0.001 improve training stability.

The number of neurons determines the model's capacity to capture temporal patterns in the input data as it determines the number of LSTM memory cells included in the model (Géron, 2022, 350). The optimizer and learning rate control how the network parameters are updated during training and therefore influence training stability (Géron, 2022, 351; 379). The batch size specifies how many observations are processed in each parameter update and can affect both computational efficiency and predictive performance (Géron, 2022, 352).

In addition to these parameters, the sequence length, number of training epochs, and early stopping settings must be specified. As LSTM networks operate on sequences of observations rather than individual data points, the training dataset is transformed into overlapping sequence windows. Each

sequence in this study is 20 consecutive observations long, corresponding approximately to one trading month, and is used to predict the subsequent observation (Géron, 2022, 552). This provides recent historical information while avoiding excessively long sequences that could increase model complexity. To ensure that predictions can be generated immediately at the beginning of the test period, the sequence for the first test observation includes the final observations from the training dataset.

The number of epochs determines how many full passes through the training data are performed during the learning process. To prevent overfitting, early stopping is applied as a regularization technique that halts training when validation performance no longer improves. The patience parameter specifies how many epochs the model is allowed to continue without improvement before training is stopped. (Géron, 2022, 311; 391)

The parameters related to early stopping are selected following Fischer & Krauss (2018). The patience parameter is set to 10 epochs, and the maximum number of epochs is set to 1000 to allow early stopping to determine the optimal stopping point during training. To implement early stopping and mitigate overfitting, the training data are further divided into training and validation subsets. Specifically, the last 10% of the training sequences are reserved as a validation set, which is used to monitor validation loss and determine when training should be halted.

Early stopping is implemented to mitigate overfitting by monitoring the evolution of the training loss, following Bhandari et al. (2022) and Fischer & Krauss (2018), who terminate training when the validation loss does not improve over several consecutive epochs. The validation loss represents the prediction error on validation data, which is not used for parameter estimation during training and therefore provides an estimate of the model's generalization performance. However, these studies consider any marginal improvement sufficient to continue training. This study utilizes a minimum improvement threshold of 10^{-4} to prevent prolonged training due to negligible changes in loss.

After training the model on the initial training subset, the optimal number of epochs is determined based on validation performance. The model is then re-estimated using the full pre-test sample for the selected number of epochs, and final forecasts are generated on the test set.

Table 3. The final LSTM configuration

LSTM configuration	
LSTM layers	1
Neurons	150
Learning rate	0.001

Batch size	16
Sequence length	20
Epochs	Maximum of 1000
Patience	10

3.2.2 Shapley additive explanations

Understanding how complex neural network models, such as LSTM networks used for stock index forecasting, generate their predictions is both challenging and essential. In financial applications, model transparency is particularly important, as making investment decisions based on opaque models may be difficult to justify. Despite their strong predictive performance, deep learning models are often considered "black boxes" because their internal decision-making processes are not directly interpretable. (Shrikumar et al., 2017, 1)

To address this limitation, one widely adopted approach is Shapley Additive exPlanations (SHAP). SHAP is an explanation method, which quantifies the contribution of each input feature to an individual prediction. This is done by computing Shapley values, which measure the average marginal contribution of a feature across all possible subsets of features. In simpler terms, SHAP measures how much each feature contributes to a prediction by comparing the model's output with and without that feature. It does this across multiple different combinations of the other features and then takes the average of those effects. This way, the final contribution reflects the feature's overall impact on the prediction in a fair and systematic manner. (Lundberg & Lee, 2017, 1–3)

Lundberg and Lee (2017) show that SHAP is an attribution method that satisfies three desirable properties: local accuracy, missingness, and consistency. Local accuracy means that the explanation model's output must exactly match the output of the original model being explained. This can mathematically be expressed as:

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i,$$

where f denotes the original model (in this study, the LSTM model), and g denotes the explanation model. The vector x represents the actual input data used by the LSTM model to produce a specific forecast. The vector x' is a simplified binary representation of the inputs used by the explanation model. The term ϕ_0 represents the expected value of the model output, which is the average LSTM model prediction across the dataset. M denotes the number of input features, and ϕ_i is the SHAP value for feature i, measuring how much a that feature contributes to moving the prediction away

from the baseline value ϕ_0. Missingness means that if a feature is not present in the simplified input used by the explanation model, then its SHAP value must be zero. Consistency means that if a feature becomes more important in the model, its SHAP value should not decrease. (Lundberg & Lee, 2017, 4)

To satisfy the previously mentioned properties, SHAP is written by Lundberg & Lee (2017) as:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \backslash i)],$$

where previously introduced notation holds. The vector z' is a binary input vector indicating whether the feature is present (1) or absent (0). The term $|z'|$ represents the number of nonzero elements in z', or equivalently, the number of features included in the subset. The expression $f_x(z')$ represents the model output when only the features included in subset are considered, while $f_x(z' \backslash i)$ denotes the model output when feature is removed from that subset. The difference therefore measures the marginal contribution of feature i to subset z'. The SHAP value is thus obtained by averaging these marginal contributions across all possible subsets. The formula measures the average contribution of each feature by evaluating how the model's prediction changes when the feature is added to different combinations of other features and the final value represents how much the feature contributes to the prediction overall. (Lundberg & Lee, 2017, 4)

In the empirical analysis of this study, SHAP values were computed using the GradientExplainer implementation from the SHAP Python package for a subset of the LSTM input sequences. Since each input observation consists of a 20-step sequence with six features, the resulting SHAP values form a three-dimensional structure consisting of samples, time steps, and features.

To obtain a measure of global feature importance, the mean absolute SHAP values were computed by averaging the absolute SHAP values across both the sample and time dimensions. This aggregation provides an overall indication of how strongly each feature influences the forecasts produced by the LSTM model, irrespective of the direction of the effect.

In addition to global feature importance, mean absolute SHAP values were also aggregated across the time dimension to examine which lags within the input sequence were most influential for the model's predictions. This lag-level analysis provides insight into whether the LSTM relies primarily on recent observations or longer historical patterns when generating forecasts. To complement the magnitude-based analysis, the SHAP values were also examined to assess the directional influence of features on the model's predictions. This analysis makes it possible to determine whether specific

features tend to increase or decrease the predicted values, thereby providing additional insight into the model's learned relationships.

3.2.3 ARIMA benchmark

Forecast accuracy is compared against a traditional linear time-series model. Specifically, the autoregressive integrated moving average (ARIMA) model is employed as a forecasting benchmark, as it is widely used in financial time-series analysis (Tsay, 2010, pp. 67–68). The ARIMA model serves as a linear reference model against which the nonlinear LSTM forecasts can be evaluated.

ARIMA models capture linear dependencies in nonstationary time series data and have been commonly applied in financial forecasting (Tsay, 2010, 67–68). Comparing the performance of LSTM-based forecasts with those produced by an ARIMA model highlights the potential advantages of neural networks in modelling nonlinear relationships in financial markets.

ARIMA models extend the autoregressive moving average (ARMA) framework by incorporating differencing to address nonstationarity. An ARMA(p,q) model expresses the current value of a stationary time series as a linear function of its past values and past forecast errors (Enders, 2015, 50–53):

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-1} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t,$$

where y_t is the value of the time series at time t, c is a constant term, ϕ_i are the autoregressive parameters, θ_j are the moving average parameters, and ε_t is an error term. ARMA models are appropriate for the modelling of stationary time series. (Enders, 2015, 50–53)

However, financial time series such as stock indices are typically nonstationary. To address this limitation, the ARIMA model was introduced by incorporating a differencing step into the ARMA model. The integration component transforms a nonstationary time series into a stationary one by differencing the data d times before applying the ARMA structure. An ARIMA(p,d,q) model therefore introduces differencing order d to achieve stationarity. (Enders, 2015, 189)

In this study the ARIMA model was estimated using daily closing prices of the OMXH25 index from 1.12.2010 to 31.12.2024. Logarithmic prices were used so that first differences correspond to continuously compounded returns, which are time-additive and therefore convenient for time-series modelling (Campbell et al., 2012, 11). The dataset was divided chronologically into a training set (first 80% of observations) and a test set (remaining 20%), consistent with the LSTM implementation to ensure comparability.

Following the Box–Jenkins methodology (Enders, 2015, 76), model identification was conducted using the training sample only. First differences of the log price series were examined to determine the integration order. The augmented Dickey–Fuller test strongly rejected the presence of a unit root in the differenced series, indicating that first differencing $(d = 1)$ was sufficient to achieve stationarity. Inspection of the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the differenced log series revealed no statistically significant autocorrelation at conventional lags. This suggested that no additional autoregressive or moving average terms were required. Accordingly, an ARIMA(0,1,0) specification was selected as the primary candidate model.

Two alternative versions of the ARIMA(0,1,0) model were estimated: a pure random walk without drift and a random walk with drift. Model selection was conducted using the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), which provide penalized likelihood measures for comparing competing time series models with parsimony requirements (Enders, 2015, 78). Consistent with Box–Jenkins methodology, model adequacy was further evaluated using residual diagnostics, including the Ljung–Box Test and inspection of residual autocorrelations (Enders, 2015, 78–79)

While the AIC marginally favoured the ARIMA(0,1,0) specification with drift, the difference in AIC values between the two models was negligible. In contrast, the BIC favoured the more parsimonious specification without drift. Given the minimal improvement in AIC and the stronger penalty imposed by BIC for additional parameters, the ARIMA(0,1,0) model without drift was selected as the benchmark specification. Residual diagnostics were conducted using the Ljung–Box test to assess the presence of serial correlation in the residuals. The test statistics at lags 10, 20, and 30 all produced p-values equal to 1.0, indicating that the null hypothesis of no autocorrelation cannot be rejected. This suggests that the residuals behave as white noise and that the ARIMA(0,1,0) model adequately captures the linear dependence structure present in the data. The selected model can be written as:

$$log(P_t) = log(P_{t-1}) + \varepsilon_t,$$

which corresponds to a random walk model.

To ensure comparability with the LSTM model, benchmark forecasts were constructed using a naive random walk specification corresponding to an ARIMA(0,1,0) model without drift. Under this specification, the optimal one-step-ahead forecast equals the most recently observed price. Accordingly, benchmark forecasts were generated by shifting the observed price series by one period,

such that the forecast for day $t$equals the realized price observed on day $t-1$. This approach is equivalent to the one-step-ahead forecast produced by an ARIMA(0,1,0) model.

The ARIMA(0,1,0) model corresponds to a random walk in logarithmic prices, implying that the optimal one-step-ahead forecast equals the most recently observed price in expectation. When applied to the trading rule used in this study, this forecast implies a predicted price change of zero. Since the trading strategy only enters positions when the predicted price change exceeds the assumed transaction costs, the ARIMA(0,1,0) model would never generate trading signals as it predicts the change in price to be zero. Consequently, the model would effectively produce a "no-trade" outcome, and it is therefore used only as a benchmark for forecast accuracy rather than as a benchmark trading strategy.

To address this limitation and to construct a benchmark trading strategy based on a traditional econometric model, an ARIMA(1,1,1) model is additionally considered. While the ARIMA(0,1,0) model is preferred based on the Box–Jenkins methodology, it produces a setting, where trading signals cannot be generated under the strategy employed in this study.

The ARIMA(1,1,1) model introduces minimal additional structure through both autoregressive and moving average components, allowing for non-zero return forecasts while maintaining a relatively simple model configuration. Although it is not the optimal model in terms of in-sample fit, it provides a benchmark for evaluating whether a standard linear time-series model can support a viable trading strategy in comparison to the LSTM model.

While the LSTM model is estimated using a static train-test split, the ARIMA(1,1,1) model is implemented using a rolling one-step-ahead forecasting scheme. A static implementation of ARIMA would produce multi-step forecasts that do not reflect realistic forecasting conditions and would severely limit the model's ability to generate meaningful trading signals. However, this does give the ARIMA(1,1,1) model an advantage in comparison to the LSTM model, as it is re-estimated using newly observed data at each forecasting step, thereby benefiting from an expanding information set and the ability to adapt to evolving market conditions, whereas the LSTM model remains fixed over the evaluation period.

3.2.4 Evaluation metrics

Evaluation of the LSTM forecasts is conducted by comparing their predictive performance against the benchmark ARIMA(0,1,0) model. Forecast accuracy is assessed using three commonly employed error metrics in RMSE, MAPE, and correlation (R), following the approach of Bhandari et al. (2022).

These metrics provide complementary perspectives on forecast performance, capturing both the magnitude of forecast errors and the strength of the linear relationship between predicted and actual values.

In addition, the Diebold–Mariano test is applied to evaluate whether differences in forecasting accuracy between the LSTM model and the ARIMA benchmark are statistically significant, following Fischer & Krauss (2018). The Diebold–Mariano test examines the null hypothesis that two competing forecasting models exhibit equal predictive accuracy (Diebold & Mariano, 1995), thereby providing a formal statistical comparison of forecast performance. However, while Fisher & Krauss (2018) apply the test to classification errors, due to their focus on directional forecasting, this study evaluates forecast accuracy based on continuous price forecasts. Accordingly, the test is applied to measure forecast error with the objective of minimizing deviations between predicted and realized prices.

3.3 Trading strategies

The trading strategy in this study is based on the forecasts produced by the LSTM model. The strategy simulates a setting in which an investor decides whether to take a long or short position in the OMXH25 index for the following trading day. If the forecasted next-day closing price exceeds the current day's closing price by more than the assumed transaction costs, the strategy enters a long position, representing the purchase of the index. Conversely, if the forecasted next-day closing price is lower than the current closing price by more than the assumed transaction costs, the strategy enters a short position, meaning that the investor sells the index short in order to profit from the expected decline. When the forecasted price difference does not exceed the assumed transaction costs, no position is taken, as the expected return is insufficient to justify trading. Transaction costs ranging from 0 to 0.25% are considered, in increments of 0.05 percentage points, to evaluate the performance of the trading strategies under both an idealized cost-free setting and a more realistic setting that accounts for transaction costs.

Each position is opened at the close of the current trading day and liquidated at the close of the following trading day. After the position is closed, a new forecast becomes available, and the trading decision is updated accordingly. This daily rebalancing framework follows the methodological approach of Fischer & Krauss (2018), who applied a similar trading setup when evaluating LSTM-based trading strategies.

In addition to the LSTM-based strategy, a benchmark trading strategy utilizing the ARIMA(1,1,1) forecasts is implemented. This allows for a direct comparison between a nonlinear machine learning

approach and a traditional linear time-series model within an identical trading framework. The ARIMA-based strategy follows the same ruleset as the LSTM strategy, ensuring that any differences in performance can be attributed to the underlying forecasting models rather than differences in trading design.

Additionally, the LSTM-based trading strategy is compared with a buy-and-hold benchmark. The buy-and-hold strategy represents a passive investment approach, where the investor holds the underlying investment for the whole investing period without making any trades. This benchmark provides a simple reference point for assessing whether the active trading strategy based on LSTM forecasts is able to generate superior investment performance.

3.3.1 Evaluation metrics for the trading strategy

The performance of the trading strategy is evaluated out-of-sample and compared against the buy-and-hold benchmark. The evaluation focuses on both profitability and risk characteristics of the resulting return series. Following the methodological approach of Fischer & Krauss (2018), performance is assessed using a combination of return characteristics, and risk-adjusted performance metrics.

To compare strategies on a risk-adjusted basis, annualized risk-return metrics are computed. The Sharpe ratio is used as the primary risk-adjusted performance measure, as it scales excess return by the volatility of returns and can be interpreted as the return earned per unit of risk (see Sharpe, 1966). In addition, the Sortino ratio is reported as a complementary measure that penalizes downside volatility only, thereby distinguishing between downside risk and upside volatility (see Sortino & van der Meer, 1991). In the calculation of both Sharpe and Sortino ratio, the risk-free rate is proxied by the 3-month Euribor, which is quoted in annualized terms. An average of the Euribor over the sample period from 2022 to 2024 is used, reflecting prevailing short-term interest rate conditions during the study period. The 3-moth Euribor data is sourced from the Bank of Finland.

Additionally, the directional accuracy of the LSTM model is considered. Rather than considering all predictions, trades are only executed when the predicted return exceeds a predefined threshold accounting for transaction costs. Within this framework, directional accuracy is defined as the proportion of executed trades that correctly predict the direction of price movement. This measure reflects the model's ability to generate economically meaningful signals, as only predictions with sufficient expected return are translated into trading decisions.

4 Results

This chapter presents the results of the forecasting models applied in this study. Additionally, the results of the trading strategies are presented and finally the interpretation of the LSTM forecasts with the SHAP values are presented.

4.1 Forecasting

When comparing the two forecasting methods utilized in this study, the empirical results indicate that the benchmark ARIMA(0,1,0) and ARIMA(1,1,1) models outperform the LSTM model across all evaluated metrics. As shown in Table 4, the benchmark models achieve substantially lower RMSE and MAPE values, indicating both lower absolute and relative forecasting errors. Furthermore, the ARIMA benchmarks exhibit a higher correlation with the observed values of the OMXH25 index during the forecasting period, suggesting better tracking of the underlying time series.

Table 4. Forecasting error metrics

Error metrics			
Metric	LSTM	ARIMA(0,1,0)	ARIMA(1,1,1)
RMSE	97.507412	49.278328	49.230255
MAPE (%)	1.572089	0.780350	0.781970
R	0.944779	0.984511	0.984547

The results indicate that both ARIMA specifications provide nearly identical forecasting performance, with only marginal differences between the random walk benchmark and the ARIMA(1,1,1) model. This suggests that the additional autoregressive and moving average components do not materially improve predictive accuracy in this context.

The Diebold–Mariano test results in Table 5 further confirm that the difference in predictive accuracy between the LSTM model and both ARIMA benchmarks is statistically significant, leading to a rejection of the null hypothesis of equal predictive accuracy between the LSTM and ARIMA models. The positive test statistics indicate that the ARIMA models consistently produce lower forecast errors than the LSTM model.

Table 5. Results of the Diebold–Mariano tests

The Diebold–Mariano tests		
	LSTM vs ARIMA(0,1,0)	LSTM vs ARIMA(1,1,1)
DM statistic	10.595245	10.571484
p-value	0.000	0.000

These findings are consistent with the near random walk behaviour commonly observed in the stock market, where simple random walk models are often difficult to outperform (Fama, 1970). It should be noted, however, that the ARIMA(1,1,1) model is implemented using a rolling one-step-ahead forecasting scheme, allowing it to incorporate newly available information at each time step. In contrast, the LSTM model is estimated once on the training data and remains fixed throughout the test period. This difference in implementation provides the ARIMA(1,1,1) model with an advantage, as it benefits from an expanding information set and the ability to adapt to evolving market conditions.

The visual comparison of actual and forecasted prices in Figure 5 further supports the quantitative results. The ARIMA(0,1,0) forecasts closely track the observed price series, with only minor deviations, reflecting its strong alignment with short-term price movements. The ARIMA(1,1,1) model exhibits similar behaviour. In contrast, the LSTM forecasts display larger deviations from the observed values, particularly during periods of rapid price changes, indicating weaker short-term tracking performance. Overall, the figure reinforces the conclusion that the ARIMA models provide a more accurate representation of short-term price dynamics in this setting.

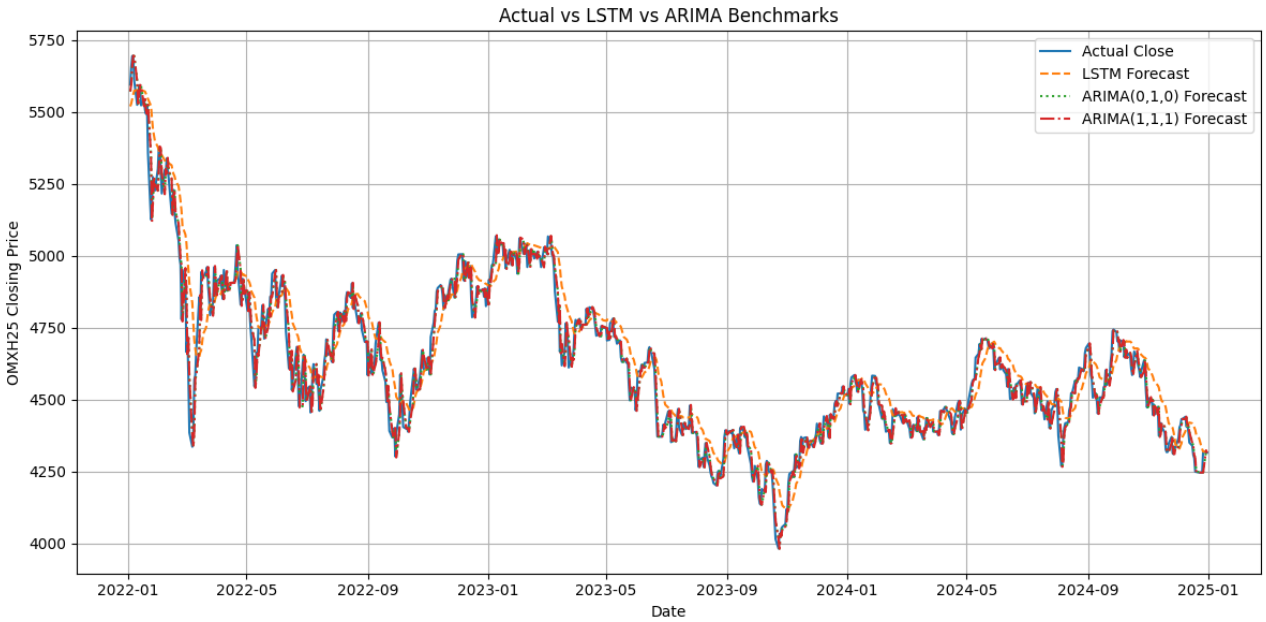


Figure 5. Actual and forecasted prices

4.2 Trading

While the previous section evaluates the forecasting performance of the models, this section examines their economic value in a trading context. The trading strategies derived from the forecasts are assessed under varying transaction cost assumption, enabling a comparison between frictionless and more realistic settings. Performance is evaluated using total returns and risk-adjusted measures, as well as metrics of trading activity and accuracy.

Figure 6 presents the cumulative returns of the trading strategies under a 0% transaction cost assumption. This setting represents an idealized benchmark, allowing the performance of the strategies to be evaluated without the influence of trading frictions. The results indicate that both the ARIMA- and LSTM-based strategies outperform the buy-and-hold benchmark under these conditions. However, the ARIMA-based strategy achieves superior performance relative to the LSTM strategy, further reinforcing the earlier finding that the LSTM model underperforms compared to the ARIMA models considered in this study.

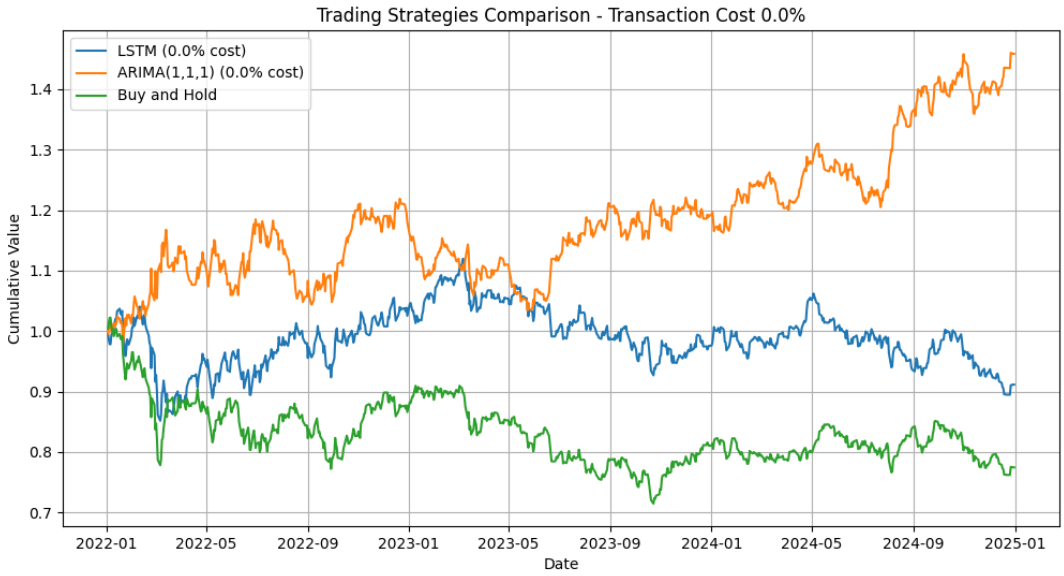


Figure 6. Cumulative returns without transaction costs

To further examine the performance of the trading strategies, Figure 7 presents the total returns as a function of transaction costs. This allows for an assessment of how sensitive the profitability of the strategies is to increasing trading frictions. As transaction costs increase, the profitability of both strategies declines. The returns of the ARIMA-based strategy quickly converge towards zero, while the returns of the LSTM-based strategy decrease more gradually across all considered transaction cost levels. This pattern suggests that the ARIMA-based strategy becomes inactive at higher

transaction cost levels, as the trading rule requires the expected return to exceed transaction costs for a position to be taken.

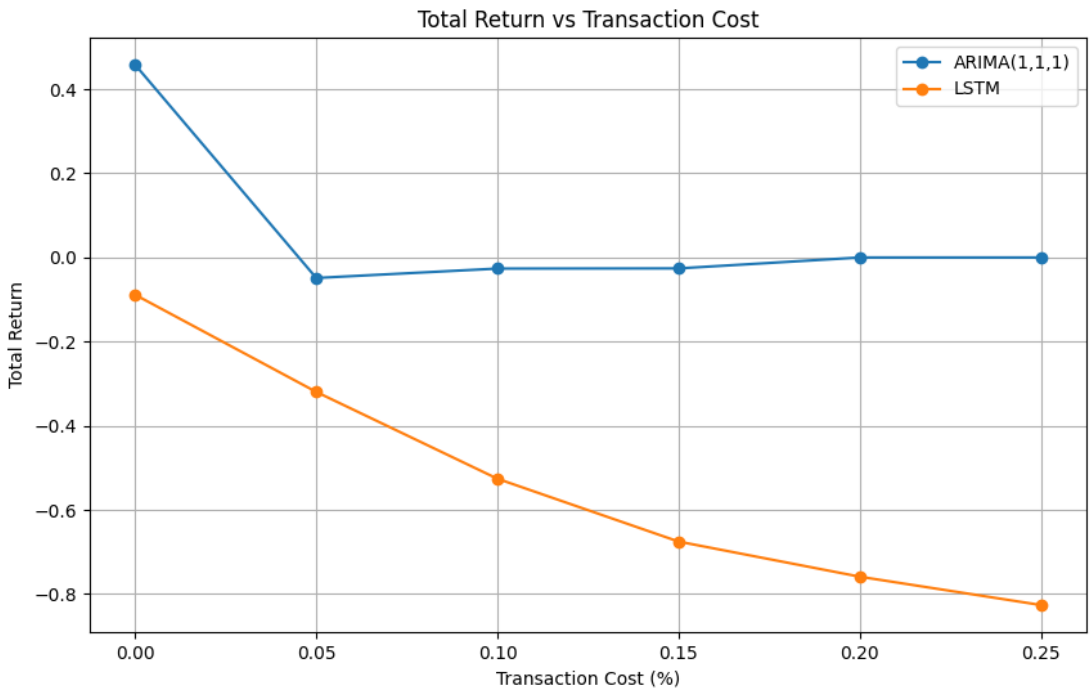


Figure 7. Total returns at varying transaction cost levels

This observation is reinforced by Figure 8, which presents the number of trades as a function of transaction costs. As transaction costs increase, trading activity declines, particularly for the ARIMA-based strategy, which becomes completely inactive at the 0.2% transaction cost level. This pattern, together with the forecasting performance indicators, suggests that the ARIMA(1,1,1) model behaves very similarly to the random walk benchmark ARIMA(0,1,0). In particular, the forecasts closely track the index by remaining near the previous day's price, resulting in limited exploitable signals once transaction costs are considered.

Although the ARIMA-based strategy exhibits strong performance under zero transaction costs, it becomes effectively unusable in more realistic settings, as it fails to generate trading signals consistently under the ruleset of the trading strategy utilized in this study. Additionally, the hit rate of both models should be considered, meaning the ratio of correctly taken positions by each trading strategy. From here, it is evident that the LSTM creates much larger signals for the trading strategy than those from the ARIMA model.

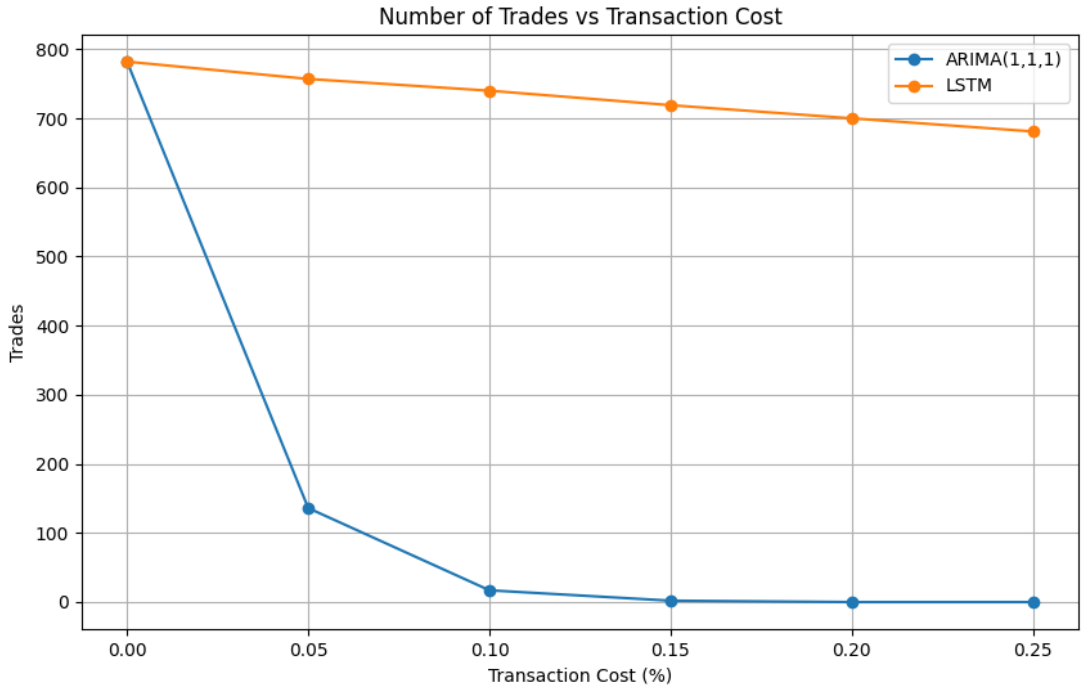


Figure 8. The number of trades at varying transaction cost levels

Table 6 presents the hit rates of the trading strategies across different transaction cost levels for each of the considered strategies. The results indicate that the LSTM-based strategy achieves a hit rate close to 50% across all transaction cost levels, suggesting that its directional predictions are only marginally better than random guessing. Similarly, the ARIMA-based strategy exhibits a hit rate around 50% at the transaction cost level of zero, further indicating limited directional predictive power. As transaction costs increase, the hit rate for the ARIMA-based strategy is no longer reported, reflecting the absence of trading activity at higher cost levels. Once transaction costs are considered, the LSTM model is able to beat the ARIMA(1,1,1) in directional accuracy.

Overall, the hit rate results suggest that neither model demonstrates strong directional forecasting ability. This reinforces the earlier findings that the profitability of the strategies is not driven by consistent predictive accuracy, but rather by occasional favourable movements that are insufficient to sustain performance once transaction costs are introduced. However, it should be noted that the LSTM model is able to outperform the ARIMA(1,1,1) in terms of hit rate across all transaction cost levels and achieves its highest hit rate of 0.5014 at 0.2% transaction cost level.

Table 6. Hit rates

Hit rates across strategies and transaction cost levels		
Transaction cost (%)	Strategy	Hit rate
0	LSTM	0.4949
0	ARIMA(1,1,1)	0.4834
0.05	LSTM	0.4993
0.05	ARIMA(1,1,1)	0.4632
0.10	LSTM	0.5000
0.10	ARIMA(1,1,1)	0.3529
0.15	LSTM	0.4993
0.15	ARIMA(1,1,1)	0
0.20	LSTM	0.5014
0.20	ARIMA(1,1,1)	–
0.25	LSTM	0.4993
0.25	ARIMA(1,1,1)	–

To further evaluate the performance of the trading strategies, risk-adjusted measures are considered. In particular, the Sharpe and Sortino ratios provide insight into the returns achieved relative to the level of risk undertaken. Table 7 presents the Sharpe and Sortino ratios across different transaction cost levels and trading strategies.

The results indicate that the LSTM-based strategy exhibits negative risk-adjusted performance even under zero transaction costs, suggesting that its returns are insufficient to compensate for the level of risk undertaken. However, the Sharpe ratio of -0.2407 and Sortino ratio of -0.3285 achieved by the LSTM under zero transaction costs indicate that the is able to outperform the buy-and-hold strategy, indicating better though still negative risk adjusted performance. This outcome may be partly explained by the fact that the underlying index generated negative returns during the investment period, which makes it inherently difficult for any strategy with limited predictive accuracy to achieve strong positive performance. In such an environment, even if the LSTM model captures certain patterns in the data, these signals may not be strong or consistent enough to overcome the prevailing downward trend and generate economically meaningful positive returns. In comparison, the ARIMA-based strategy achieves positive risk-adjusted performance in the frictionless setting, suggesting that it may be better suited to exploiting short-term price dynamics under these market conditions.

As transaction costs increase, the risk-adjusted performance of the LSTM-based strategy deteriorates further, with both Sharpe and Sortino ratios becoming increasingly negative across all cost levels.

This highlights the sensitivity of the LSTM-based strategy to trading frictions and its inability to sustain performance once realistic market conditions are considered. The ARIMA-based strategy also experiences a rapid decline in risk-adjusted performance, becoming inactive at higher transaction cost levels.

Table 7. Sharpe and Sortino ratios

Sharpe and Sortino ratios across strategies and transaction cost levels			
Transaction cost (%)	Strategy	Sharpe ratio	Sortino ratio
	Buy-and-Hold	-0.5421	-0.7732
0	LSTM	-0.2407	-0.3285
0	ARIMA(1,1,1)	0.7224	1.2123
0.05	LSTM	-0.7793	-1.0377
0.05	ARIMA(1,1,1)	-0.4251	-0.2966
0.10	LSTM	-1.3870	-1.8296
0.10	ARIMA(1,1,1)	-0.7132	-0.2134
0.15	LSTM	-1.9785	-2.5464
0.15	ARIMA(1,1,1)	-3.1397	-13.4078
0.20	LSTM	-2.3901	-3.0437
0.20	ARIMA(1,1,1)	–	–
0.25	LSTM	-2.8117	-3.5174
0.25	ARIMA(1,1,1)	–	–

These findings highlight that, in this context, the utilized LSTM model configuration fails to deliver practical economic value despite its capacity to model complex patterns. This underscores the importance of evaluating machine learning models not only based on statistical or in-sample performance, but also on their out-of-sample economic viability when applied to financial markets.

4.3 Interpretation of the LSTM forecasts

While the theoretical framework of SHAP explains how feature contributions can be computed, it is equally important to interpret what these contributions reveal about the behaviour of the LSTM forecasting model used in this study. The SHAP values produced by the trained LSTM model provide insight into how different input variables influence the forecasts of the OMXH25 index and how the model combines information from the historical input sequence when generating predictions. Under conditions where the model produces reasonably accurate forecasts and the explanatory features exhibit meaningful SHAP values, this approach can be used to better understand the internal decision-making of the LSTM model. Furthermore, SHAP can serve as an exploratory tool for forming

hypotheses about potential relationships in financial markets, although such interpretations should be treated with caution, as they do not imply causal effects.

The computed global SHAP values reveal a ranking of the explanatory variables in terms of their influence on the LSTM forecasts. Since the LSTM model is trained on min–max normalized target values, the SHAP values are expressed in the normalized output space. Although these values can be linearly rescaled to approximate contributions in original index units, their interpretation remains local and model-dependent and therefore should be treated as indicative rather than exact measures of economic impact. Accordingly, the SHAP values indicate the relative contribution of each feature to the scaled prediction, not its direct effect in index points.

Table 8 Global SHAP values

Global SHAP feature importance	
Feature	Mean absolute SHAP value
Close	0.052968
Unemployment rate	0.000854
EUR/USD rate	0.000554
Consumer confidence	0.000454
ECB main refinancing rate	0.000142
VSTOXX	0.000140

Table 8 presents the mean absolute SHAP values for each input variable. The results indicate that the closing price variable overwhelmingly dominates the model's predictions, with a mean absolute SHAP value that is over 60 times greater than that of all other variables. This finding suggests that the LSTM model relies primarily on the historical values of the index itself, effectively behaving as a strongly autoregressive model. Such behaviour is consistent with financial time-series dynamics, where past values of an index often contain substantial predictive information due to serial dependence (Tsay, 2010, 43).

Among the remaining variables, Unemployment rate, EUR/USD rate, and Consumer confidence exhibit the next highest SHAP importance, although their contributions remain very small in comparison to the Close variable. This indicates that macroeconomic and financial indicators provide only marginal additional information, slightly adjusting the forecasts derived from past index movements. The remaining two variables ECB main refinancing rate and VSTOXX show negligible SHAP contributions, suggesting that their influence on the model's predictions is minimal.

Overall, the results reported in Table 8 indicate that the LSTM model primarily behaves as an autoregressive forecasting model, in which the historical values of the stock index dominate the predictive signal. The additional variables appear to play a limited supporting role.

Following the analysis of global feature importance, the SHAP values were examined across the time dimension to understand how the model utilizes information from different points in the input sequence.

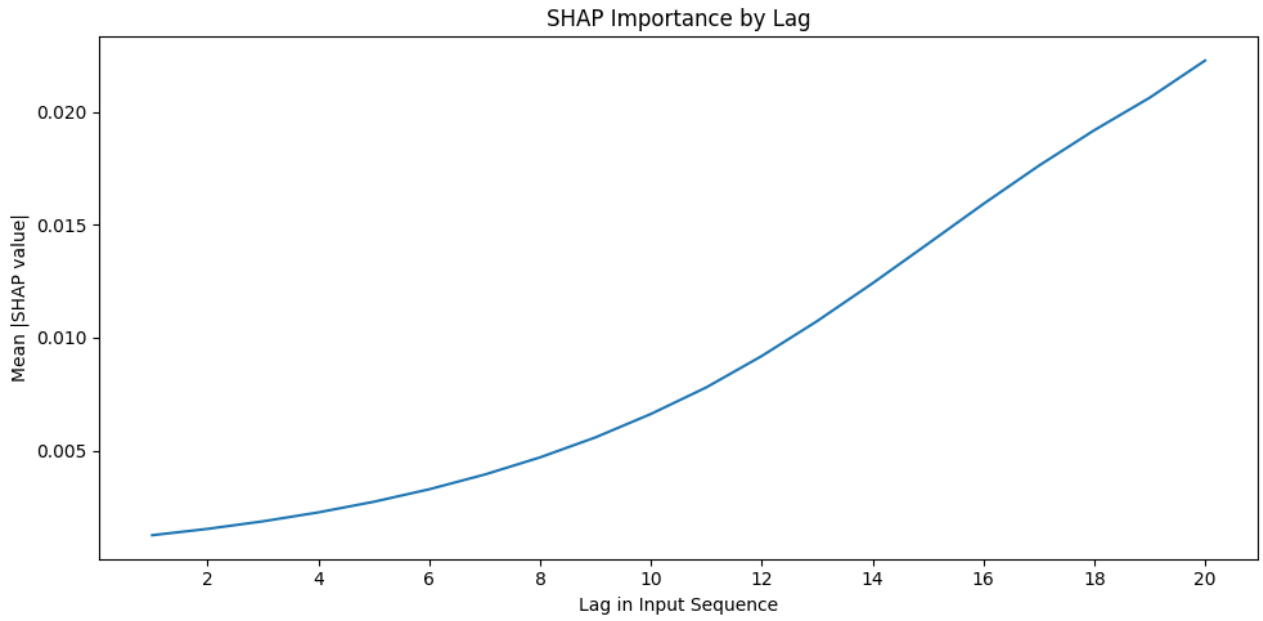


Figure 9. Mean absolute SHAP value per lag

Figure 9 illustrates the mean absolute SHAP value across lags and features. A clear and monotonic pattern emerges: the importance of observations increases steadily toward the most recent time steps. The final lags of the input sequence exhibit the largest SHAP values, indicating that the model places the greatest weight on the most recent observations when forming predictions.

This pattern suggests that the LSTM's effective memory is strongly concentrated near the end of the sequence, with earlier observations contributing progressively less to the forecast. Such behaviour is consistent with short-term dependencies typically observed in financial time series, where recent market information tends to dominate predictive signals (Tsay, 2010, 38–40).

While mean absolute SHAP values provide information about the magnitude of feature importance, they do not capture the direction of influence. To address this, the regular SHAP values were examined. Figure 11 presents the distribution of SHAP values for each feature. The results show that the Close variable exhibits a wide spread of positive SHAP values, indicating that it consistently

contributes to increasing the predicted values, although the magnitude of this contribution varies across observations and time steps. This suggests that higher recent values of the index are associated with upward adjustments in the model's forecasts, reflecting a pronounced persistence effect in the underlying time series. In other words, recent movements of the index tend to carry forward into the near future, and the model captures this behaviour.

The remaining variables display SHAP values that are closely clustered around zero, further enforcing the notion of their relatively weak influence on the model's predictions. Although slight variation is present, particularly at more recent lags, these effects are small compared to those of the Close variable.

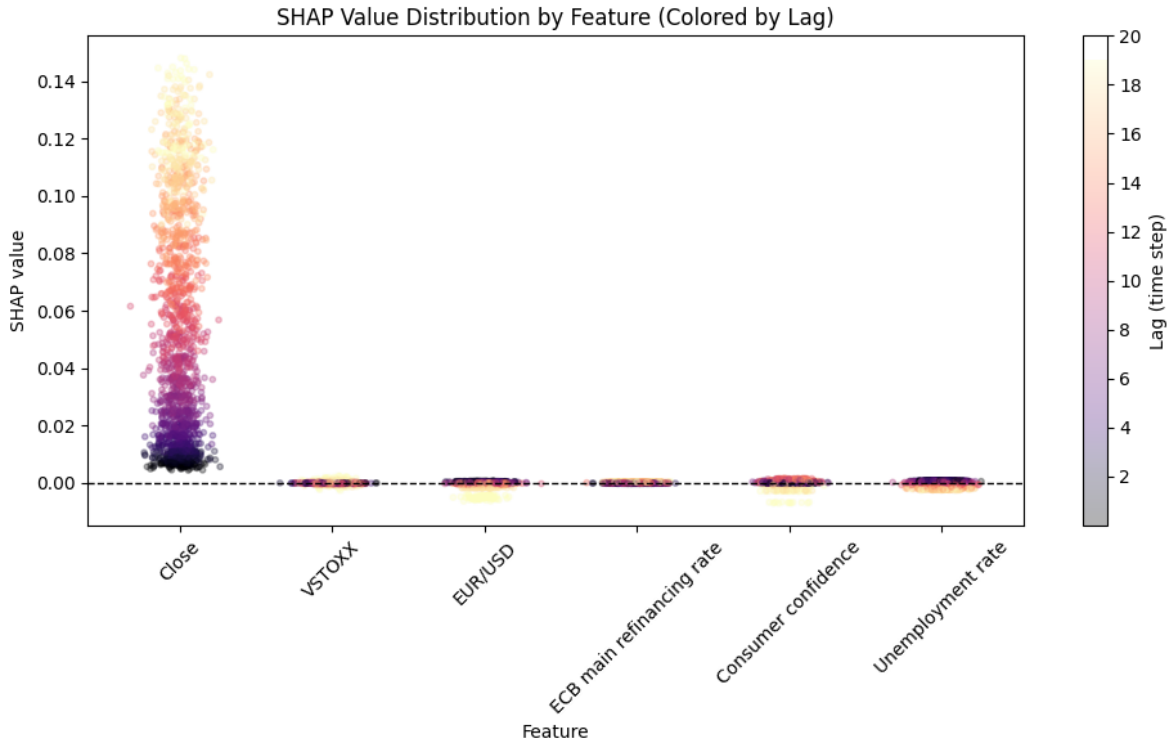


Figure 10 SHAP values by feature

Taken together, the SHAP analysis suggests that the LSTM model primarily captures short-term autoregressive patterns in the OMXH25 index. The historical values of the index dominate both the magnitude and direction of predictions, while the other variables contribute only marginally. This may indicate that the model is unable to effectively learn relationships between the OMXH25 closing price and the additional explanatory variables, that these relationships are unstable and change between the training and forecasting periods, or that the included variables possess limited predictive power for the index's price movements.

The temporal analysis further indicates that the model's predictive power is concentrated in the most recent price observations, with limited reliance on longer-range historical information. This behaviour suggests that the model effectively learns short-term dependencies rather than complex long-term temporal structures.

5 Conclusions

The purpose of this study was to examine whether a LSTM-based trading strategy can provide investors with an economically meaningful opportunity to earn excess returns relative to a buy-and-hold benchmark. In addition, the forecasting performance of the LSTM model was compared to that of a classical econometric model, namely the ARIMA model. This comparison is particularly relevant, as LSTM models are computationally intensive and their forecasting process is inherently less transparent.

Accordingly, the study aims to assess whether, under the conditions considered, the additional complexity of training a LSTM model is justified, or whether a simpler autoregressive model provides superior predictive performance. Furthermore, given that LSTM models are often characterized as "black boxes," the study also considers interpretability using additional explanatory methods in SHAP values.

This study aims to add to the current literature around stock market forecasting methods, and the utilization of the LSTM for this purpose by combining findings of studies in the field of computer science searching for optimal LSTM model architectures such as Bhandari et al. (2022) and Yadav et al. (2020), while applying the model to a trading strategy, to find out whether these models are able to produce economically meaningful returns. Similar approaches on an individual stock level have been applied for example by Fischer & Krauss (2018) and Akita et al. (2016), who found promising results in their studies. Additionally, this study aims to further their framework by attempting to explain how the model produces the forecasting results in an attempt to tackle the black box nature of these models and make the trading process more transparent.

The empirical findings of this study suggest that the LSTM model does not provide economically meaningful advantages in the current setting of this study. In terms of forecasting accuracy, the ARIMA benchmarks outperform the LSTM model across all evaluated metrics, producing lower forecasting errors and stronger correlation with the underlying index. This indicates that the computational resources required to train and implement the LSTM model do not translate into improved predictive performance for the OMXH25 index. Comparing the results of this study to that of Bhandari et al. (2022), their LSTM model exhibits significantly lower RMSE of 40.4574, and MAPE of 0.7989 while in this study the respective values are 97.507412 and 1.572089. Similarly, their model's forecasts exhibit a correlation of 0.9976 with the underlying index while in this study a correlation of 0.944779 is reached. While these results are not directly comparable due to differences

in datasets, the overall evidence suggests that the model used by Bhandari et al. (2022) outperforms the one employed here in terms of forecasting accuracy. Whether this difference arises from superior data characteristics or more effective model specification remains an open question for further research.

These differences in predictive performance naturally raise the question of whether they translate into economically meaningful trading outcomes. The trading results further reinforce the earlier conclusions. Although the LSTM-based strategy was able to outperform the buy-and-hold benchmark in terms of both returns and risk-adjusted performance under zero transaction cost conditions, the advantage disappears once even modest transaction costs of 0.05% per transaction are considered. Thus, under more realistic market conditions, the LSTM-based strategy is unable to outperform the benchmark. Moreover, the hit rate analysis indicates that the directional accuracy of the LSTM model is close to that of random guessing at around 50% accuracy, limiting its ability to generate reliable trading signals.

However, this does not unambiguously mean that the model is unusable in a trading setting. Chalvatzis & Hristu-Varsakelis (2020) achieved a similar directional accuracy of around 50% when forecasting the S&P 500, DJIA, NASDAQ, and R2000 indices, while achieving returns well over the buy-and-hold strategy by utilizing a more sophisticated trading strategy than the one utilized in this study. The question then raises, whether the LSTM configuration utilized in this study could consistently provide improved returns by simply applying a more sophisticated trading strategy, which remains an avenue for further research. For example, restricting trades to periods with stronger predicted signals or adjusting position sizes based on the magnitude of expected returns could potentially enhance performance. However, such approaches would require additional investigation and are beyond the scope of this study.

Beyond the design of the trading strategy itself, another possible explanation for the observed performance relates to the stability of the relationships learned by the model. The relationships captured by the LSTM may be unstable and difficult to generalize over time. While more frequent retraining, such as updating the model daily, could potentially improve adaptability to changing market conditions, it would further complicate the identification and interpretation of underlying relationships. In contrast, the ARIMA model benefits from its implementation as a rolling one-step-ahead model, where parameters are effectively updated using newly available information at each time step. This allows the ARIMA-based approach to adapt more quickly to evolving market conditions, providing an inherent advantage over the static LSTM model used in this study.

However, this comparison is not entirely symmetric. The LSTM model is estimated once and remains fixed throughout the test period, whereas the ARIMA model is continuously updated. Consequently, part of the observed performance difference may reflect differences in model updating procedures rather than purely differences in model structure. More broadly, while hyperparameter tuning or the inclusion of additional explanatory variables may improve statistical performance, it remains an open question whether such improvements would translate into economically meaningful gains under the conditions of this study.

Further insight is provided by the SHAP analysis conducted in this study. The results indicate that the LSTM model places substantial importance on recent lagged values of the price series. This suggests that the model primarily relies on short-term information that closely reflects current market conditions, rather than extracting strong predictive signals from the broader set of explanatory variables. As a result, the additional variables included in the model may contribute limited incremental predictive power beyond what is already contained in past prices.

It remains unclear whether the limited predictive contribution of the additional features is due to a lack of informational content with respect to the OMXH25 index during the sample period, changes in the relationships between variables across the training and forecasting periods, or the inability of the LSTM model to effectively learn and utilize such relationships. Distinguishing between these explanations is challenging, as both data limitations and model specification may influence the results. However, the findings suggest that, within the context of this study, the selected variables provide only limited incremental value beyond the information already embedded in past prices. An additional possibility is that the considered LSTM construction is overly complex and thus is leading into overfitting during the training process, this phenomenon of overfitting is widely documented by other studies such as Yadav et al (2020) and Yu & Yan (2020).

In summary, the findings of this study provide answers to the research questions. First, a LSTM-based forecast does not yield excess returns relative to a buy-and-hold benchmark once realistic transaction costs are considered in the setting of this study. Second, the ARIMA model outperforms the LSTM model in both forecasting accuracy and trading performance, with part of this difference attributable to its ability to adapt dynamically to new information in the setting of this study. Third, the SHAP analysis indicates that the LSTM model relies primarily on past price information, leaning mostly on the most recent observations, while additional explanatory variables contribute limited incremental predictive value in the setting of this study.

Despite the insights provided, this study is subject to several limitations. First, the LSTM model is estimated once and remains fixed throughout the test period, which may limit its ability to adapt to changing market conditions compared to models that are continuously updated. Second, the selection of input variables and model configuration may influence the results, and alternative feature sets or architectures could yield different outcomes. Finally, the analysis is conducted on a single market index, which may limit the generalizability of the findings to other asset classes or time periods.

Future research could extend this analysis in several directions. First, using the same model architecture and dataset, the performance of LSTM models could be evaluated within a rolling retraining framework to assess whether more frequent updates improve adaptability and economic performance. Second, alternative model architectures, hyperparameter tuning strategies, and additional explanatory variables could be explored to enhance predictive accuracy. In particular, further research could examine whether the limited predictive contribution of the explanatory variables observed in this study reflects insufficient informational content in the data or limitations in the model's ability to capture such relationships, for example by developing more optimized model configurations through systematic hyperparameter tuning.

Third, alternative trading rule designs, such as threshold-based strategies or dynamic position sizing, could be investigated to better exploit potential predictive signals, as similar approaches have yielded positive results in prior studies, such as Chalvatzis & Hristu-Varsakelis (2020). Finally, extending the analysis to other markets could provide additional insight into the broader applicability of machine learning models in financial forecasting.

Overall, the findings of this study suggest that, in the context of financial markets, the complexity of machine learning models does not guarantee superior economic performance. Instead, simple models and realistic trading considerations remain critical in determining practical usefulness.

References

Aggarwal, C. C. (2023) *Neural Networks and Deep Learning: A Textbook*, Second edition, Springer. https://doi.org/10.1007/978-3-031-29642-0

Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016) Deep learning for stock prediction using numerical and textual information, *2016 IEEE/ACIS 15th International Conference on Computer and Information Science*. https://doi.org/10.1109/ICIS.2016.7550882

Asness, C. S. – Moskowitz, T. J. – Pedersen, L. H. (2013) Value and Momentum Everywhere, *The Journal of Finance*, Vol. 68 (3), 929–985. https://doi.org/10.1111/jofi.12021

Bachelier, L. (1900) Théorie de la speculation. *Annales scientifiques de l'École Normale Supérieure*, Vol. 17, 21–86. https://doi.org/10.24033/asens.476

Bachelier, L. – Davis, M. H. A. – Etheridge, A. (2006) *Louis Bachelier's theory of speculation: the origins of modern finance*, course book, Princeton University Press. https://doi.org/10.1515/9781400829309

Ball, R. – Brown, P. (1968) An Empirical Evaluation of Accounting Income Numbers, *Journal of Accounting Research*, Vol. 6 (2), 159–178. https://doi.org/10.2307/2490232

Bhandari, H. N. – Rimal, B. – Pokhrel, N. R. – Rimal, R. – Dahal, K. R. – Khatri, R. K. C. (2022). Predicting stock market index using LSTM. *Machine Learning with Applications*, Vol 9, 100320. https://doi.org/10.1016/j.mlwa.2022.100320

Bishop, C. M. (1995) *Neural Networks for Pattern Recognition*, Oxford: Clarendon.

Bodie, Z. – Kane, A. – Marcus, A. J. (2021) *Investments*, twelfth edition, international student edition. New York: McGraw-Hill Education.

Boyd, J. H. – Hu, J. – Jagannathan, R. (2005) The Stock Market's Reaction to Unemployment News: Why Bad News Is Usually Good for Stocks, *The Journal of Finance (New York)*, Vol. 60 (2), 649–672. https://doi.org/10.1111/j.1540-6261.2005.00742.x

Campbell, J. Y. – Lo, A. W. – MacKinlay, C. A. (2012) *The Econometrics of Financial Markets*, New Jersey: Princeton University Press. https://doi.org/10.1515/9781400830213

Carbó, J. M. – Gorjon, S. (2024) Determinants of the price of bitcoin: An analysis with machine learning and interpretability techniques. *International Review of Economics & Finance*, Vol. 92, 123–140. https://doi.org/10.1016/j.iref.2024.01.070

Chalvatzis, C., – Hristu-Varsakelis, D. (2020) High-performance stock index trading via neural networks and trees, *Applied Soft Computing*, Vol 96, https://doi.org/10.1016/j.asoc.2020.106567

Christoffersen, P. F. – Diebold, F. X. (2006) Financial Asset Returns, Direction-of-Change Forecasting, and Volatility Dynamics. *Management Science*, Vol. 52 (8), 1273–1287. https://doi.org/10.1287/mnsc.1060.0520

De Bondt, W. F. M. – Thaler, R. (1985) Does the Stock Market Overreact, *The Journal of Finance*, Vol. 40 (3), 793–805. https://doi.org/10.1111/j.1540-6261.1985.tb05004.x

Diebold, F. X. – Mariano, R. S. (1995) Comparing Predictive Accuracy, *Journal of Business & Economics Statistics*, Vol. 13 (3), 253–263. https://doi.org/10.2307/1392185

Enders, W. (2015) *Applied econometric time series*, 4th edition, Hoboken, New Jersey, John Wiley and Sons.

Fama, E. F. (1965) The Behaviour of Stock-Market Prices, *The Journal of Business*, Vol. 38 (1), 34–105. https://doi.org/10.1086/294743

Fama, E. F. (1970) Efficient Capital Markets: A Review of Theory and Empirical Work, *The Journal of Finance*, Vol. 25 (2), 383–417. https://doi.org/10.2307/2325486

Fama, E. F. (2017) *The Fama Portfolio: Selected Papers of Eugene F. Fama*, edited by J. H. Cochrane and T. J. Moskowitz, University of Chicago Press.

Fama, E. F. – Blume, M. E. (1966) Filter Rules and Stock-Market Trading, *The Journal of Business*, Vol. 39 (1), 226–241. https://doi.org/10.1086/294849

Fama, E. F. – Fisher, L. – Jensen, M. C. – Roll, R. (1969) The Adjustment of Stock Prices to New Information, *International Economic Review*, Vol. 10 (1), 1–21. https://doi.org/10.2307/2525569

Fama, E. F. – French, K. F. (1988) Permanent and Temporary Components of Stock Prices, *The Journal of Political Economy*, Vol. 96 (2), 246–273. https://doi.org/10.1086/261535

Fischer, T. – Krauss, C. (2018) Deep learning with long short-term memory networks for financial market predictions, *European Journal of Operational Research*, Vol. 270 (2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

Fisher, K. L. – Statman, M. (2003) Consumer Confidence and Stock Returns, *The Journal of Portfolio Management*, Vol. 30 (1), 115–127. https://doi.org/10.3905/jpm.2003.319925

Frenkel, J. A. – Levich, R. M. (1975) Covered Interest Arbitrage: Unexploited Profits?, *Journal of Political Economy*, Vol. 83 (2), 325–338. https://doi.org/10.1086/260325

Géron, A. (2022) *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*, Third edition, Sebastopol, CA: O'Reilly Media, Inc.

Giudici, P. – Piergallini, A. – Recchioni, M. C. – Raffinetti, E. (2024) Explainable Artificial Intelligence methods for financial time series. *Physica A*, *655* https://doi.org/10.1016/j.physa.2024.130176

Goodfellow, I. – Bengio, Y. – Courville, A. (2016) *Deep Learning*, Cambridge, Massachusetts; The MIT Press.

Gu, S. – Kelly, B. – Xiu, D. (2020) Empirical Asset Pricing via Machine Learning, *The Review of Financial Studies*, Vol. 33 (5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

Gu, W. – Peng, Y. (2019) Forecasting the market return direction based on a time-varying probability density model, *Technological Forecasting & Social Change*, Vol 148, 1–8. https://doi.org/10.1016/j.techfore.2019.119726

Hochreiter, S.– Schmidhuber, J. (1997) Long Short-Term Memory, *Neural Computation*, Vol.9 (8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Ioannidis, C. – Kontonikas, A. (2008) The Impact of Monetary Policy on Stock Prices, *Journal of Policy Modeling*, Vol. 30 (1), 33–53. https://doi.org/10.1016/j.jpolmod.2007.06.015

Jegadeesh, N. – Titman, S. (1993) Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency, *The Journal of Finance*, Vol. 48 (1), 65–91. https://doi.org/10.1111/j.1540-6261.1993.tb04702.x

Jensen, M. C. (1968) The Performance of Mutual Funds in the Period 1945–1964, *The Journal of Finance*, Vol. 23 (2), 389–416. https://doi.org/10.1111/j.1540-6261.1968.tb00815.x

Kendall, M. G. (1953) The Analysis of Economic Time-Series-Part I: Prices, *Journal of The Royal Statistical Society. Series A (General)*, Vol. 116 (1), 11–34. https://doi.org/10.2307/2980947

Kobiela, D. – Krefta, D. – Król, W. – Weichbroth, P. (2022) ARIMA vs LSTM on NASDAQ stock exchange data, *Procedia Computer Science*, Vol. 207, 3830–3839. https://doi.org/10.1016/j.procs.2022.09.445

Kumar, D. – Taylor, G. W. – Wong, A. (2017) *Opening the Black Box of Financial AI with CLEAR-Trade: A CLass-Enhanced Attentive Response Approach for Explaining and Visualizing Deep Learning-Driven Stock Market Prediction.* https://doi.org/10.48550/arxiv.1709.01574

Lo, A. W. – MacKinlay, C. (1988) Stock Market Prices Do Not Follow Random Walks: Evidence from a Simple Specification Test, *The Review of Financial Studies*, Vol. 1 (1), 41–66. https://doi.org/10.1093/rfs/1.1.41

Lundberg, S – Lee, S. I. (2017) *A Unified Approach to Interpreting Model Predictions*, https://doi.org/10.48550/arxiv.1705.07874.

Malkiel, B. G. (1973) *A random walk down Wall Street: Including a life-cycle guide to personal investing*.

Malkiel, B. G. (1995) Returns from Investing in Equity Mutual Funds 1971 to 1991, *The Journal of Finance*, Vol. 50 (2), 549–572. https://doi.org/10.1111/j.1540-6261.1995.tb04795.x

Malkiel, B. G. (2003) The Efficient Market Hypothesis and Its Critics, *The Journal of Economic Perspectives*, Vol. 17 (1), 59–82. https://doi.org/10.1257/089533003321164958

McCulloch, W. S. – Pitts, W. (1943) A Logical Calculus of the Ideas Immanent in Nervous Activity, *Bulletin of Mathematical Biology*, Vol. 52 (1), 99–115. https://doi.org/10.1007/BF02478259

Nelson, D. M. Q. – Pereira, A. C. M. – de Oliveira, R. A. (2017) Stock Market's Price Movement Prediction with LSTM Neural Networks, *Proceedings of International Joint Conference on Neural Networks,* 1419–1426. https://doi.org/10.1109/IJCNN.2017.7966019

Nielsen, M. A. (2015) *Neural Networks and Deep Learning*, Determination Press.

Phylaktis, K. – Ravazzolo, F. (2005) Stock Prices and Exchange Rate Dynamics, *Journal of International Money and Finance*, Vol. 24 (7), 1031–1053. https://doi.org/10.1016/j.jimonfin.2005.08.001

Rosenblatt, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, Vol. 65 (6), 386–408. https://doi.org/10.1037/h0042519

Sharpe, W. F. (1966) Mutual Fund Performance, *The Journal of Business*, Vol. 39 (1), 119–138. http://dx.doi.org/10.1086/294846

Shrikumar, A – Greenside, P – Shcherbina, A – Kundaje, A (2017) *Not Just a Black Box: Learning Important Features Through Propagating Activation Differences*, https://doi.org/10.48550/arxiv.1704.02685.

Siami-Niamini, S. – Tavakoli, N. – Siami Namin, A. (2018) A Comparison of ARIMA and LSTM in Forecasting Time Series, *17th IEEE International Conference on Machine Learning and Applications*. https://doi.org/10.1109/ICMLA.2018.00227

Sortino, F. A. – van der Meer, R. (1991) Downside Risk: Capturing What's at Stake in Investment Situations. *Journal of Portfolio Management*, Vol. 17 (4), 27-31. https://doi.org/10.3905/jpm.1991.409343

Tsay, R. S. (2010) *Analysis of financial time series*, 3rd edition, Hoboken, New Jersey: Wiley.

Whaley, R. E. (2009) Understanding the VIX, *Journal of Portfolio Management*, Vol. 35 (3), 98–105. https://doi.org/10.3905/JPM.2009.35.3.098

Yadav, A. – Jha, C. K. – Sharan, A. (2020) Optimizing LSTM for time series prediction in Indian stock market, *Procedia Computer Science*, Vol. 167, 2091–2100. https://doi.org/10.1016/j.procs.2020.03.257

Yu, P. – Yan, X. (2020) Stock Price Prediction Based on Deep Neural Networks, *Neural Computing & Applications*, Vol. 32 (6), 1609–1628. https://doi.org/10.1007/s00521-019-04212-x

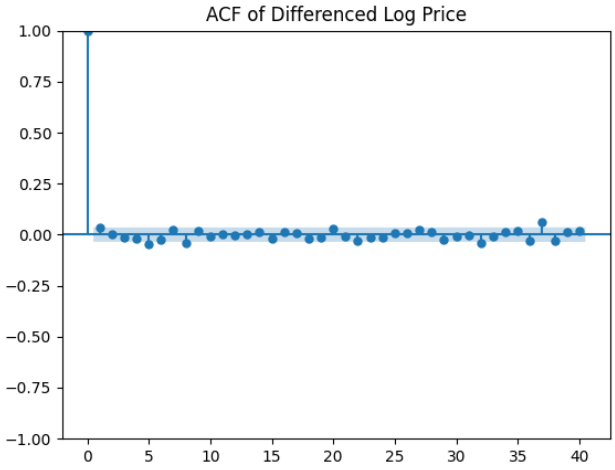
The tables and figures shown in this study were created by the author

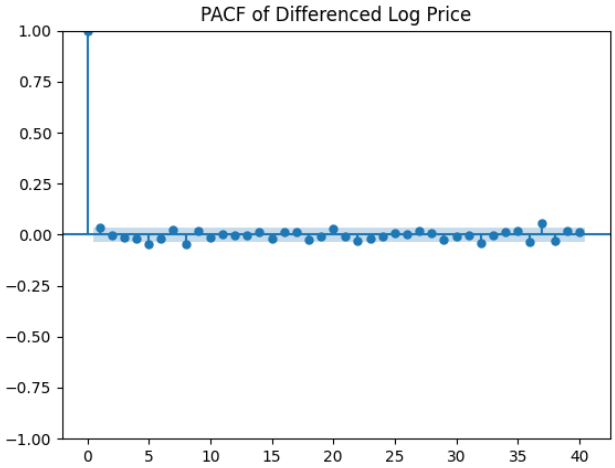
The code for the methodology of this study is stored in the following GitHub repository:
https://github.com/antonsalonen/Master-s-thesis-code

Appendices

Appendix 1 ARIMA(0,1,0) following Box–Jenkins

ARIMA(0,1,0) comparison		
	AIC	BIC
Model without drift	-18555.618	-18549.569
Model with drift	-18555.707	-18543.610



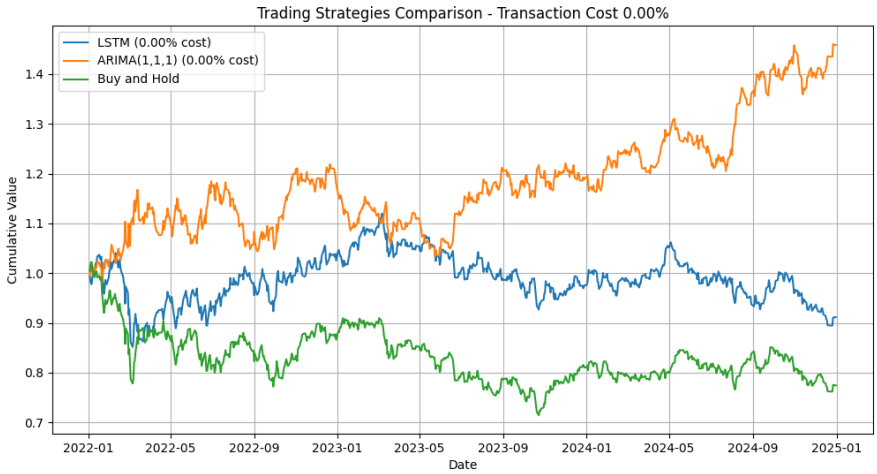


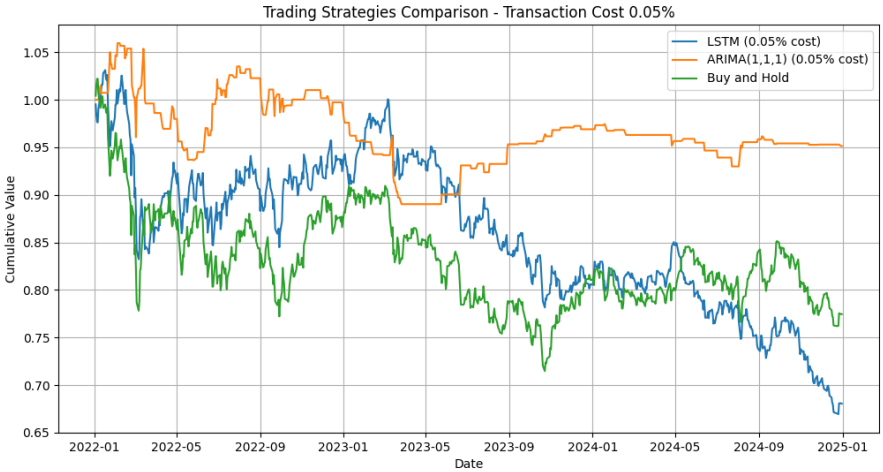
Ljung–Box test results		
Lag	LB statistic	p-value
10	0.066783	1.0
20	0.108044	1.0
30	0.182694	1.0

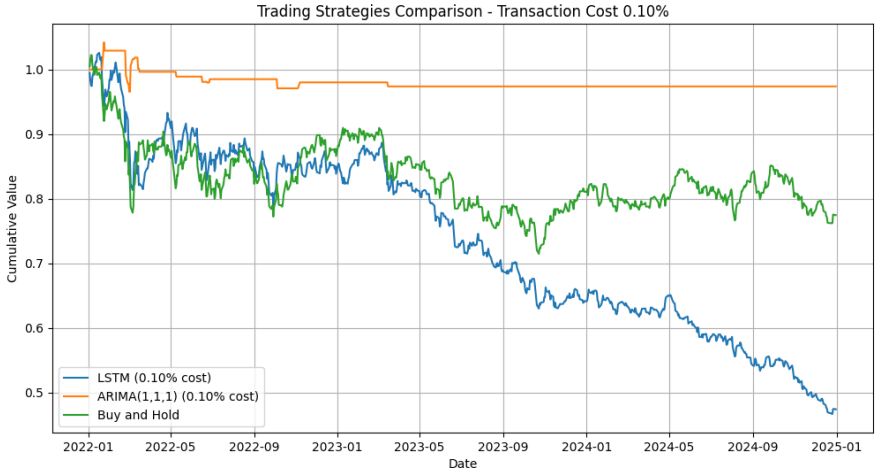
Appendix 2 Trading strategy returns, volatilities, and trades

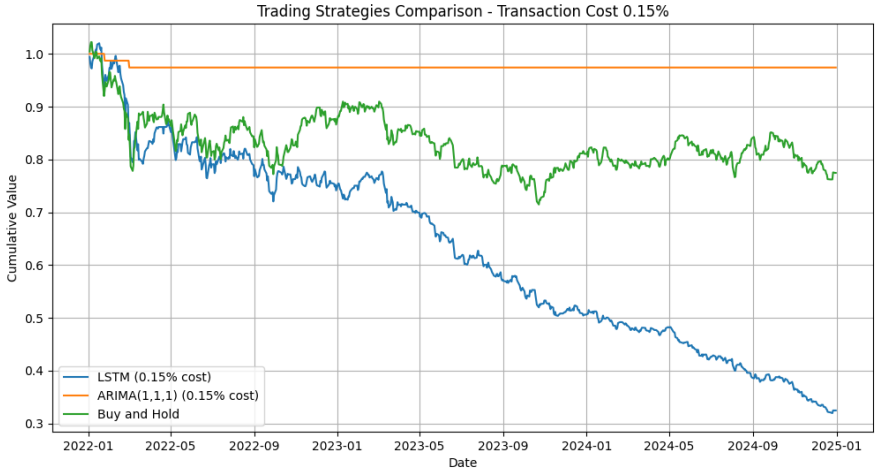
Total returns, volatilities, and trades across strategies and transaction cost levels				
Transaction cost	Strategy	Total return (%)	Annual volatility (%)	Trades
	Buy-and-Hold	-0.2254	0.1670	–
0	LSTM	-0.0884	0.1671	782
0	ARIMA(1,1,1)	0.4583	0.1669	782
0.05	LSTM	-0.3195	0.1655	757
0.05	ARIMA(1,1,1)	-0.0487	0.0865	136
0.10	LSTM	-0.5259	0.1641	740
0.10	ARIMA(1,1,1)	-0.0263	0.0449	17
0.15	LSTM	-0.6753	0.1614	719
0.15	ARIMA(1,1,1)	-0.0259	0.0105	2
0.20	LSTM	-0.7588	0.1606	700
0.20	ARIMA(1,1,1)	0	0	0
0.25	LSTM	-0.8261	0.1594	681
0.25	ARIMA(1,1,1)	0	0	0

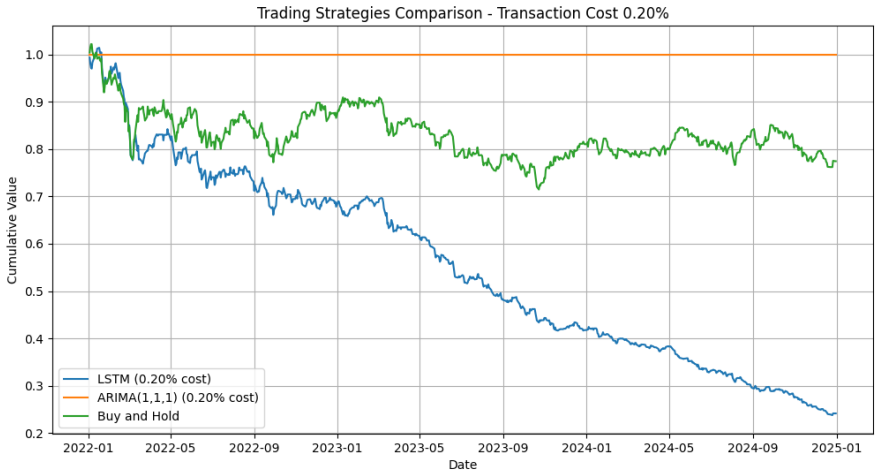
Appendix 3 Cumulative return plots

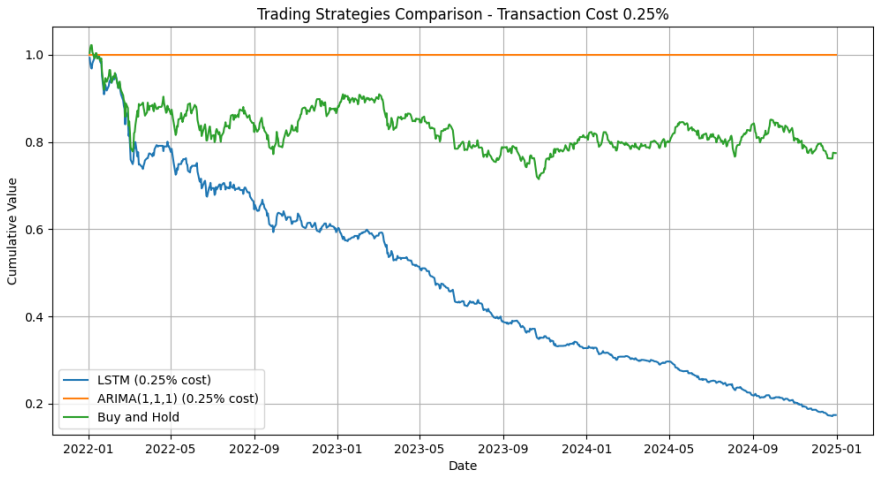







Trading Strategies Comparison - Transaction Cost 0.15%
Cumulative Value
Date
LSTM (0.15% cost)
ARIMA(1,1,1) (0.15% cost)
Buy and Hold


Trading Strategies Comparison - Transaction Cost 0.20%
Cumulative Value
Date
LSTM (0.20% cost)
ARIMA(1,1,1) (0.20% cost)
Buy and Hold


Trading Strategies Comparison - Transaction Cost 0.25%
Cumulative Value
Date
LSTM (0.25% cost)
ARIMA(1,1,1) (0.25% cost)
Buy and Hold

Appendix 4 A simplified step-by-step explanation of what the utilized codes do

The code LSTM_Training_Forecasting_SHAP, stored in the GitHub repository linked to this study contains the LSTM training, forecasting, and SHAP analysis. Below, is a simplified step-by-step explanation of the code:

- Loads market and macroeconomic data from an Excel file
- Uses Close variable as the prediction target
- Selects the variables of this study as input features
- Splits the data into training, and test sets
- Creates a separate validation set from the tail end of the training period
- Scales inputs and target values with MinMax normalization
- Builds fixed-length input sequences of 20 observations for the LSTM
- Trains a LSTM model with early stopping to find the best number of epochs
- Retrains the same LSTM structure on the full pre-test dataset using that best epoch count
- Predicts Close values on the test set
- Transforms predictions and actual values back to the original price scale
- Uses SHAP to interpret the trained LSTM
- Computes global feature importance from SHAP values
- Computes lag importance to see which time steps matter most
- Draws a SHAP distribution plot by feature, coloured by lag
- Saves actual and forecasted prices to a separate CSV file for further analysis

The code ARIMA is used in the estimation of the optimal ARIMA model with Box-Jenkins methodology and forecasting. Below, is a simplified step-by-step explanation of the code:

- Loads price data from Excel
- Uses the Close price series
- Converts prices to log scale

- Splits the data into training and test sets
- Examines the differenced training series with ACF/PACF
- Compares two benchmark random-walk models as a result of ACF and PACF inspection:
    - ARIMA(0,1,0) without drift
    - ARIMA(0,1,0) with drift
- Chooses one based on AIC/BIC
- Checks residual autocorrelation with Ljung–Box
- Introduces a rolling ARIMA(1,1,1) forecast
- Converts forecasts back to price scale
- Saves them to a separate CSV file for further analysis

The code LSTM_ARIMA_COMPARISON is used in the comparison of the LSTM and ARIMA in terms of forecasting accuracy. Below, is a simplified step-by-step explanation of the code:

- Loads the LSTM and ARIMA(1,1,1) forecast files
- Merges the LSTM and ARIMA forecast series by date
- Creates an ARIMA(0,1,0) benchmark forecast
- Computes forecast evaluation metrics for each model:
    - RMSE
    - MAPE
    - Correlation with actual values
- Runs Diebold–Mariano tests comparing LSTM against both ARIMA benchmarks
- Saves the results to a CSV file
- Plots actual prices against all three forecasted series

The code Trading_simulation is used in the the simulation of the trading strategies and their evaluation. Below, is a simplified step-by-step explanation of the code:

- Loads the forecast files

- Merges both forecast series on the same date index
- Converts forecasted prices into predicted daily returns
- Computes the actual daily return from observed prices
- Builds a buy-and-hold benchmark return series
- Loops over different transaction cost levels from 0.00% to 0.25%
- For each transaction cost, creates trading signals for LSTM and ARIMA(1,1,1)
- Turns those signals into gross daily strategy returns
- Subtracts transaction costs to get net daily strategy returns
- Builds cumulative equity curves for each strategy and each cost level
- Computes summary trading statistics such as number of trades, hit rate, and total return
- Computes additional performance measures such as Sharpe, and Sortino ratio
- Prints the summary and performance tables
- Plots cumulative equity curves for each transaction cost level
- Plots total return as a function of transaction cost
- Plots number of trades as a function of transaction cost

Appendix 5 Explanation of the use of AI

Artificial intelligence tools have been utilized in a manner allowed by the UTU guidelines during the writing of this thesis. Specifically, AI has been used to improve the clarity, structure, and overall quality of the written language. This includes refining grammar, enhancing academic tone, and improving the coherence and flow of the text.

In addition, AI tools have been used to assist with coding tasks, such as debugging, improving code structure, and providing guidance on the implementation of statistical and machine learning methods. AI was also utilized to support the identification of relevant academic literature, including books and research articles related to the topic of this thesis. Furthermore, these tools were also used to support the organization and structure of the thesis, including suggestions for chapter layout and logical flow.

All substantive research decisions, including the selection of methodology, data, model configuration, interpretation of results, and conclusions, were made independently by the author. AI-generated suggestions were critically evaluated and, where necessary, modified to ensure their accuracy and alignment with academic standards.

Below is a list of example prompts utilized during the thesis process:

- Make this paragraph more concise and remove repetition.
- Fix grammar and improve flow in this text.
- Help me improve the logical flow between these two paragraphs.
- Check whether the sequence construction for the validation and test sets in my Python code avoids data leakage when extending the datasets with previous observations.