CHARLES UNIVERSITY
FACULTY OF SOCIAL SCIENCES
Institute of Economic Studies

Predictability of financial returns across horizons using Deep Learning

Master's thesis

Author: Martin Nedvěd

Study program: Economics and Finance

Supervisor: doc. PhDr. Jozef Baruník, Ph.D.

Year of defense: 2024

Declaration of Authorship

The author hereby declares that he or she compiled this thesis independently, using only the listed resources and literature, and the thesis has not been used to obtain any other academic title.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis in whole or in part and agrees with the thesis being used for study and scientific purposes.

Prague, July 30, 2024

Martin Nedved

Abstract

This thesis explores the predictability of financial returns across hourly, daily, weekly, and monthly horizons using Long Short-Term Memory (LSTM) networks. Despite advancements in machine learning, its application in finance faces unique challenges, such as small datasets and low signal-to-noise ratios. Our research aims to address the limitations of existing studies, which predominantly focus on the daily horizon and although some studies analyze different horizons, direct comparisons are challenging due to the varied methodologies and datasets employed. By utilizing a consistent dataset and methodology, we enable a direct comparison of models' performance across various horizons. We enhance predictive models by incorporating fractionally differentiated series to retain memory in financial data and realized volatility from high-frequency data to capture market fluctuations. Our study also extends beyond equities to include futures markets. The key takeaway of our research is that LSTM networks are particularly effective for short-term financial return predictions at hourly and daily horizons. Their performance decreases for longer horizons, such as weekly and monthly, possibly due to fewer market inefficiencies to exploit. Furthermore, the inclusion of futures data does not enhance model performance but reveals interesting trends in feature selection.

JEL Classification	C52, C45, C53, G17, C22
Keywords	Deep Learning, Financial Returns, Fractionally Differentiated Series, Time Horizons
Title	Predictability of financial returns across horizons using Deep Learning

Abstrakt

Tato práce zkoumá předvídatelnost finančních výnosů na hodinovém, denním, týdenním a měsíčním horizontu pomocí Long Short-Term Memory (LSTM) sítí. Navzdory pokrokům ve strojovém učení se jeho aplikace ve finančním sektoru potýká s problémy, jako jsou malé datasety a nízký poměr signálu k šumu. Naším cílem je překonat omezení stávajících studií, které se převážně zaměřují na denní horizont. I když některé studie analyzují různé časové horizonty, přímé srovnání je komplikované kvůli různým metodikám a datasetům. Použitím jednotného datasetu a metodiky umožňujeme přímé srovnání úspěšnosti modelů

napříč různými horizonty. Modely navíc rozšiřujeme integrací frakčně diferencovaných řad pro zachování paměti a dále realizované volatility vypočítané z vysokofrekvenčních dat, aby bylo možné zachytit tržní fluktuace. Naše studie také překračuje rámec akcií a zahrnuje i trhy s futures. Klíčovým zjištěním je, že LSTM sítě jsou nejefektivnější pro krátkodobé předpovědi finančních výnosů na hodinovém a denním horizontu. Jejich výkonnost se snižuje pro delší horizonty, jako jsou týdenní a měsíční, pravděpodobně kvůli menšímu počtu tržních neefektivit, které lze využít. Rozšíření datového souboru o futures nezvyšuje účinnost modelu, ale odhaluje zajímavé trendy ve výběru vstupních parametrů.

Klasifikace JEL	C52, C45, C53, G17, C22
Klíčová slova	Hluboké učení, Finanční výnosy, Frakčně diferencované řady, Časové horizonty
Název práce	Predikovatelnost finančních výnosů na různých horizontech pomocí hlubokého učení

Acknowledgments

I would like to express my appreciation and gratitude to the supervisor doc. PhDr. Jozef Baruník, Ph.D. for his invaluable suggestions, guidance, and useful comments. I am deeply thankful to my supportive girlfriend, my family, and all those close to me for their support during the process of writing and during my whole studies. A special thanks to my mum for her delicious pancakes.

Typeset in FSV LaTeX template with great thanks to prof. Zuzana Havrankova and prof. Tomas Havranek of Institute of Economic Studies, Faculty of Social Sciences, Charles University.

Bibliographic Record

Nedvěd, Martin: *Predictability of financial returns across horizons using Deep Learning.* Master's thesis. Charles University, Faculty of Social Sciences, Institute of Economic Studies, Prague. 2024, pages 75. Advisor: doc. PhDr. Jozef Baruník, Ph.D.

Contents

List of Tables

List of Figures

Acronyms

ML Machine learning

LSTM Long Short-Term Memory

ARIMA Autoregressive integrated moving average

SVM Support vector machine

RNN Recurrent neural network

FNN Feedforward neural network

GRU Gated Recurrent Unit

PT Pesaran-Timmermann

Master's Thesis Proposal

Author	Martin Nedvěd
Supervisor	doc. PhDr. Jozef Baruník, Ph.D.
Proposed topic	Predictability of financial returns across horizons using Deep Learning

Motivation Predicting financial time series returns is a classical and highly challenging problem due to the presence of complex relationships, a low signal-to-noise ratio, and the continuous evolution of markets. Despite these difficulties, researchers are motivated by the potential rewards of identifying investment opportunities, improving risk management, and gaining deeper insights into the workings of the market.

While traditional approaches, such as ARIMA and GARCH models, have been widely used for modeling financial time series, recent years have witnessed significant advancements in machine learning methods across various domains. Notably, generative models and large language models like ChatGPT have gained substantial attention. Researchers have increasingly employed these methods in the context of financial time series due to their ability to capture non-linear relationships, uncover complex patterns, and adapt to the dynamic nature of financial markets. One notable advantage of machine learning techniques is their flexibility in handling different prediction horizons. Financial markets show different characteristics at various time horizons, while short-term returns may be influenced by high-frequency market noise, longer-term returns might, on the other hand, be influenced by macroeconomic or fundamental factors. The choice of prediction horizon is a crucial consideration that depends on the specific application. For portfolio management, predicting daily or even monthly returns might be more relevant, while for algorithmic trading, 1-minute predictions could be of greater value. This thesis aims to investigate the ability of machine learning models to predict returns at various horizons.

Furthermore, this thesis will compare the behavior of machine learning models across different market types to assess their robustness and generalizability. Specifically, we will examine how these models perform in international stock markets and

determine whether their predictive capabilities extend to commodity markets and cryptocurrency markets. By conducting such comparisons, we can gain insights into the applicability and limitations of the machine learning models in different market contexts.

Overall, this thesis seeks to contribute to the field of financial time series prediction by evaluating the effectiveness of machine learning models across different horizons and diverse market types. The findings from this study will provide valuable insights for investors, risk managers, and algorithmic traders, aiding them in making informed decisions and developing effective strategies in various financial market scenarios.

Hypotheses Our main hypothesis that we want to address in the thesis: 1. Can deep learning explain the horizon predictability? 2. Does the horizon predictability differ across different markets (e.g. international stock markets, commodity markets, and cryptocurrency markets)?

Methodology For this thesis, we will use historical financial data from stocks, Bitcoin, and commodities. Using 1-minute, 10-minute, hourly, daily, and monthly data allows us to measure the performance of the models across various prediction horizons. In order to predict returns in the next period(s) we could use only historical prices as input features, Jiang (2021) analyze 124 papers and reports that almost 36% choose this approach. Further, another 25% of analyzed studies utilize historical prices as well as technical indicators. Other input features could be for example macroeconomics data, fundamental data, or text data such as news and social media. Selecting the appropriate features and determining the optimal data length for different horizons will be a demanding yet crucial task in our research.

There are many different deep learning architectures available, we will primarily consider the family of Recurrent neural networks (RNN) models which excel in managing the sequence of input data. Jiang (2021) found that Long Short-Term Memory (LSTM) models were the most frequently utilized in the analyzed papers. Hence, following this direction by incorporating LSTM models in our research seems justified. By employing LSTM models and considering various input features, including historical prices and potentially other relevant data sources, we aim to evaluate their performance in predicting financial time series returns across different horizons. These models have shown promise in capturing complex patterns and relationships in financial data, making them well-suited for our research objectives.

Expected Contribution The main goal of the thesis is to contribute to the field of financial time series prediction by exploring the predictive capabilities of deep

learning models for various financial time series across different horizons. In general, predicting returns is very challenging, yet the rewards are significant The findings from this research will have practical implications for market participants, providing guidance in selecting suitable models based on the needed prediction horizon.

The better the model the higher the potential to enhance risk management and the overall profitability. Further, the comparison across different markets could provide valuable insights into market-specific patterns. This analysis can also identify the model's strengths and weaknesses in different market scenarios, highlighting their ability to adapt to varying data characteristics and underlying market dynamics. Finally, the recent macroeconomic events such as the Covid-19 pandemic, the war in Ukraine, and high inflation offer great challenges that our models will have to account for. By incorporating these real-world events into the analysis, we can evaluate the models' ability to capture and adapt to sudden changes in market conditions.

Outline

1. Introduction
2. Literature review
3. Methodology
4. Data
5. Results
6. Conclusion

Bibliography

Lezmi, E., & Xu, J. (2023). Time Series Forecasting with Transformer Models and Application to Asset Management. Available at SSRN 4375798.

Jiang, W. (2021). Applications of deep learning in stock market prediction: recent progress. Expert Systems with Applications, 184, 115537.

Thakkar, A., & Chaudhari, K. (2021). A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions. Expert Systems with Applications, 177, 114800.

Rezaei, H., Faaljou, H., & Mansourfar, G. (2021). Stock price prediction using deep learning and frequency decomposition. Expert Systems with Applications, 169, 114332.

Yu, P., & Yan, X. (2020). Stock price prediction based on deep neural networks. Neural Computing and Applications, 32, 1609-1628.

Author

Supervisor

Chapter 1

Introduction

Machine learning techniques have significantly advanced various fields, including image recognition, natural language processing, and game playing (Krizhevsky *et al.* 2012; Mnih *et al.* 2015; Brown *et al.* 2020). These techniques excel at uncovering hidden patterns and interpreting high-dimensional data. However, their application in finance presents unique challenges, such as small dataset issues, a low signal-to-noise ratio, and the necessity for model interpretability (Israel *et al.* 2020).

Most studies in financial return prediction focus on the daily horizon, with only a few exceptions exploring longer horizons, such as five or ten days (Jiang 2021). This focus on a single timescale limits the demonstration of the generalizability of machine learning algorithms across various market conditions (Akyildirim *et al.* 2021). Different prediction horizons capture distinct aspects of market behavior. Short-term predictions, such as hourly forecasts, are influenced by intraday trading activities and market microstructure, while longer-term predictions, like weekly or monthly forecasts, reflect broader economic trends. Although there are studies that examine each horizon separately, comparing the results is challenging due to differences in methodologies and datasets. This thesis aims to bridge this gap by using a consistent methodology and dataset to compare the predictive ability of deep learning models across four different horizons: hourly, daily, weekly, and monthly.

This thesis builds on the foundation laid by previous studies in financial forecasting using machine learning. Specifically, we are following the methodologies of Krauss *et al.* (2017) and the subsequent study by Fischer & Krauss (2018), who extend the original work. These studies compare various model architectures and find recurrent neural networks to perform the best, which is

why we are employing this family of models as well. Additionally, these studies analyze different study periods (years), allowing us to test how the models behave over time and across different market conditions. This temporal analysis is crucial for evaluating the robustness and adaptability of the models in varying financial environments.

Moreover, this thesis introduces several methodological advancements over previous studies. First, it incorporates fractionally differentiated series as features, following the insights of Prado (2018), to better handle the memory removal from price data to achieve stationarity. Second, it includes realized volatility, calculated from high-frequency 5-minute data, as an additional feature to capture market fluctuations more accurately. Third, the analysis extends beyond traditional equity markets to include futures, offering a broader perspective on different markets.

Following this introduction, the thesis is further divided into the following chapters. Chapter 2 presents a literature review that provides an overview of the fundamental concepts of machine learning and its applications in finance. This chapter also reviews key studies in the field and discusses the challenges associated with financial return prediction. Chapter 3 outlines the research methodology, including feature engineering, data preparation, model selection, and evaluation metrics. It details the deep learning architectures used in the study and describes the software implementation. Chapter 4 describes the data sources, preprocessing steps, and criteria for selecting stocks and futures for analysis. It also presents the characteristics of the dataset used in the empirical analysis. Chapter 5 presents the study's findings, comparing the performance of different models and features across various time horizons. Finally, Chapter 6 summarizes the key findings, discusses the implications of the results, and suggests directions for future research.

Chapter 2

Literature Review

Machine learning (ML) systems have shown remarkable achievements in a variety of fields. Notable accomplishments include breakthroughs in image classification, as demonstrated by the work of Krizhevsky *et al.* (2012). In the area of natural language processing, large language models like those developed by Brown *et al.* (2020) have achieved exceptional performance across a spectrum of tasks without needing specific training for each. Additionally, the field of video gaming has seen ML reach human-competitive levels, a milestone marked by Mnih *et al.* (2015).

These examples not only showcase the versatility of machine learning, covering supervised, unsupervised, and reinforcement learning but also illustrate its effectiveness in big data environments characterized by high signal-to-noise ratios, a measure of how much predictability is present in a system, as Israel *et al.* (2020) explain. However, within the finance sector, these features often present challenges.

In this chapter, we explain the fundamentals of machine learning. Then we present the most important applications in finance. Finally, review in depth the literature regarding returns predictions and its limitations.

2.1 Fundamentals of Machine Learning

Mitchell (1997) in his book famously defines machine learning as the process through which computer programs can independently enhance their performance by gaining experience. What sets machine learning apart is its inherent flexibility. Dixon *et al.* (2020) contrast machine learning with many conventional statistical methods, emphasizing its unique capability to discern patterns

from data without relying on predefined assumptions about the data's underlying structure. Gu *et al.* (2020) propose an even more detailed definition:

"The definition of "machine learning" is inchoate and is often context specific. We use the term to describe (a) a diverse collection of high-dimensional models for statistical prediction, combined with (b) so-called '*regularization*' methods for model selection and mitigation of overfit, and (c) efficient algorithms for searching among a vast number of potential model specifications."

The authors further elaborate that these high-dimensional models provide more flexibility than traditional econometric prediction methods, enhancing the ability to approximate complex data processes, such as those underlying equity risk premiums. However, they also note that this increased flexibility raises the risk of overfitting. To address this, machine learning integrates specific strategies aimed at ensuring stable performance in unseen data, thereby actively preventing overfit. Moreover, considering the sheer number of predictors and potential model combinations, machine learning utilizes advanced tools to effectively and efficiently approximate the optimal model specification without exhaustive computational efforts.

In the introduction of this chapter, we mentioned three examples of successful machine learning applications, on these examples we will explain the types of machine learning.

Image classification represents a classic example of supervised learning, where the algorithm is trained on a dataset that contains labeled images. Each label indicates the class or category of the image, and the algorithm's task is to learn to predict these categories for new, unseen images. The breakthrough achievement by Krizhevsky *et al.* (2012) in image classification, specifically with their development of the AlexNet model, revolutionized this field. They utilized a deep convolutional neural network that could automatically and accurately identify and categorize images, a task that was previously challenging for machines. This success marked a significant advancement in computer vision and highlighted the potential of deep learning, a subset of machine learning.

In contrast, the large language models developed by Brown *et al.* (2020), such as GPT-3, are examples of unsupervised learning, particularly in the realm of natural language processing. These models learn to generate coherent and contextually relevant text by being exposed to a vast corpus of text data without specific task-oriented labeling. They are trained to predict the next word in a sentence, learning the structure and nuances of language in the process. This approach enables the model to perform a variety of language-related tasks with-

out being explicitly trained for each one, showcasing the versatility and power of unsupervised learning in handling complex and unstructured data.

The domain of video gaming, particularly the achievement of Mnih *et al.* (2015) in training algorithms to play Atari 2600 video games, is an exemplar of reinforcement learning. In this type of learning, an agent learns to make decisions by performing actions within an environment to achieve a goal, receiving feedback in the form of rewards or penalties. The agent learns the best actions to take in various situations to maximize its rewards, effectively learning a strategy for the game. This method, which mimics the way humans and animals learn through interaction with their environment, has broad applications beyond gaming, including robotics, autonomous vehicles, and more complex decision-making tasks in finance.

These examples illustrate the broad applicability of machine learning in various fields. In the context of this thesis, our focus is directed toward structured and labeled data, positioning our study within the domain of supervised learning.

Brief history of feedforward neural networks

The origins of feedforward neural networks can be traced back to 1958 with the introduction of the perceptron model by Rosenblatt (1958). This early model was an attempt to simulate the functioning of biological neurons, giving rise to the term *neural networks*. However, the perceptron had its limitations, which were not fully addressed until the mid-1980s. It was during this period that Rumelhart *et al.* (1986) introduced the backpropagation algorithm, a pivotal development that allowed for the effective training of multi-layer neural networks, thereby overcoming some of the perceptron's initial constraints.

This historical context demonstrates that while machine learning concepts have existed for some time, it is the advent of new algorithms and the significant increase in computational power that have truly unlocked the potential of these techniques. These advancements have enabled the widespread application and rapid progress in machine learning that we witness today.

2.2 Machine Learning in Finance

Machine learning's application in the financial sector is diverse, encompassing areas such as fraud detection, risk evaluation, financial text mining, and sen-

timent analysis. Studies covering this diverse range of topics are thoroughly explored by Ozbayoglu *et al.* (2020). The potential of machine learning to enhance credit card fraud detection is exemplified in the work of Jurgovsky *et al.* (2018). Additionally, Luo *et al.* (2017) study the effectiveness of deep belief networks in credit risk scoring models, particularly for credit default swaps datasets, showcasing their superiority over traditional models like logistic regression. The use of financial news information in improving bank distress classifiers is discussed by Cerchiello *et al.* (2018). Li *et al.* (2017) incorporate investor sentiment from forum posts to analyze the irrational component of stock price to further improve stock prediction performance. These examples illustrate the extensive range of financial issues to which machine learning techniques have been applied. The final topic, focusing on predicting stock returns, directly relates to our research area. We will explore this area in more depth in the following section.

2.3 Predicting financial returns

By some authors return prediction is the most important task for portfolio construction problems (Israel *et al.* 2020). The interest in applying machine learning to this problem is driven by the increasing availability of data and the advancement of computational techniques among others (Dixon *et al.* 2020). Given the vast number of new studies emerging annually (Sezer *et al.* 2020), this thesis concentrates on those most relevant to our specific research question. We begin by addressing the challenges inherent in predicting financial returns. Subsequently, we examine some significant papers in the field, followed by an exploration of studies that forecast returns over various time horizons.

2.4 Challenges

As we discussed at the beginning of this chapter, a high signal-to-noise ratio is key to the success of machine learning in areas like image recognition, language models, and video games. But finance is unfortunately much more difficult. Israel *et al.* (2020) explain that when it comes to return prediction task we deal with small data problem. This problem doesn't come from a lack of regressors. Prado (2018) categorizes data into four categories. First is fundamental data, encompassing sales, costs, earnings, and macroeconomic

variables. Second, market data includes prices, volumes, and dividends. Third, analytics data involves news sentiment, credit ratings, and analysts' recommendations. The final category, alternative data, taps into satellite imagery, Google searches, social media feeds, and more. However, the core problem is the number of observations, for example, working with daily data we have only approximately 220-240 observations every year. In comparison with other machine learning tasks such as image recognition with datasets in hundreds of thousands of images, this is a really small dataset. Another problem mentioned by Israel *et al.* (2020) is the low signal-to-noise ratio, this is not a coincidence and is in check with the efficient market hypothesis proposed by Fama (1970). It makes returns prediction however extremely challenging. We already mentioned various categories of data, some mainly alternative and high-frequency data that are challenging to process on the other hand as Prado (2018) argues the more challenging the dataset the less chance someone already exploited the information. Other limitations of machine learning in finance mentioned by Israel *et al.* (2020) are for instance need for interpretability and evolving markets, the first is more important for regulated asset managers who prefer interpretability of the model so they can explain their investment decisions. Evolving markets lead to the need for constant rolling of new models and algorithms to beat the market, Prado (2018) explains in detail how this is done in practice. Prado (2018) further emphasizes the need for proper backtesting of proposed strategies which many studies lack.

2.5 Key studies

Krauss *et al.* (2017) analyze the application of deep neural networks, gradient-boosted trees, and random forests in financial statistical arbitrage. They developed a model to generate daily trading signals for stocks in the S&P 500 Index, aiming to outperform the market. Specifically, these models are used to predict the directional movements of S&P 500 stocks. Stocks are ranked daily by their likelihood of upward movement, the top k are bought and the bottom k sold short, forming portfolios of varying sizes based on these predictions. Their findings suggest that an ensemble of these methods achieved higher average raw returns compared to individual models. The importance of this study for our research is twofold, first, it highlights the importance of the ensemble technique, and second, it provides a benchmark that can be used for comparison with our results. Fischer & Krauss (2018) build upon the work of Krauss *et al.*

(2017) and deploy Long Short-Term Memory (LSTM) networks. The authors demonstrate that these models outperform memory-free classification methods such as logistic regression, random forests, and even deep neural networks. These findings are important for our study as LSTM shows promising results in comparison with other models. Ghosh *et al.* (2022) extend the work of Krauss *et al.* (2017) and Fischer & Krauss (2018) by employing both random forests and LSTM networks to forecast directional movements of S&P 500 stocks. They introduce a multi-feature setting, including returns related to closing, opening prices, and intraday returns. Their methodology enhances the predictive accuracy beyond the single-feature methods used in the earlier studies. Again, for our research, these discoveries demonstrate that potential improvements can be achieved with more complex input features rather than only closing prices. We reviewed these papers in detail because they provide reasonable benchmarks that later studies try to overcome. This illustrates which methods and features can enhance the overall performance.

The importance of benchmarking is stressed by Lago *et al.* (2021), in this case for the electricity markets but the message applies to the field of our research as well. The authors highlight that the use of non-public datasets, and short and market-limited test samples leads to difficulties in benchmarking new methods against established models. Hence without proper benchmarks, it is complicated or even impossible to compare the performance of various models.

In machine learning, the value of benchmarking is for example illustrated by the achievements of AlexNet, developed by Krizhevsky *et al.* (2012) which we already explained at the beginning of this chapter. This model emerged from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where researchers compete on standardized datasets to deliver the best performance. Popular datasets such as CIFAR-10, CIFAR-100, or MNIST are commonly utilized for computer vision problems.

In finance, the use of standardized datasets is not as straightforward due to the sector's distinctive challenges. One good example of a standardized challenge is the M6 Financial Forecasting Competition, where the objective was to explain above-average financial returns. Its structure, spanning a year and utilizing live data for evaluation, was specifically designed to counter the look-ahead bias, a frequent obstacle in financial forecasting. Staněk (2023) explains briefly the rules of the competition but more importantly his winning forecasting methods. For our research, this highlights the importance of not using data that were not seen at the time which is also strongly stressed by

Prado (2018).

Until now we only briefly reviewed that machine learning delivers better performance than classical methods. Gu *et al.* (2020) highlight that machine learning forecasts in some cases double the performance of leading regression-based strategies. The authors find that allowing for nonlinearities and interactions among predictors is crucial for improving accuracy which is missed by linear methods. They further stress the importance of momentum, liquidity, and volatility as dominant predictive signals. Li *et al.* (2020) acknowledge the limitations of traditional models like Autoregressive integrated moving average (ARIMA) when handling high-frequency data. To account for this the authors extend ARIMA with ML models, in this case Support vector machine (SVM) and LSTM. They argue that by this extension the model retains the theoretical basis of ARIMA but at the same time leverages machine learning in handling non-linear relationships. They conclude the superior performance of the ARIMA-LSTM model compared with both standalone ARIMA and SVM-ARIMA.

Sezer *et al.* (2020) systematically review literature focused on the application of deep learning, a branch of machine learning. Their findings suggest that LSTM models and their variations are predominantly used in financial time series forecasting out of various other deep learning models. Furthermore, they highlight Python and its libraries as the preferred tools for modeling in this domain. In a similar vein, Thakkar & Chaudhari (2021) also observe the extensive use of LSTM in various financial forecasting studies, though they abstain from concluding which method is the most utilized. Jiang (2021) provides another detailed survey on the application of deep learning in stock market predictions, highlighting Recurrent neural network (RNN) as the most frequently used models. The author also notes that historical prices and technical indicators are the most common input features, with daily prediction horizons being the most favored. This finding aligns with Sezer *et al.* (2020), who also observe daily prediction horizons as a common approach in the studies they analyzed. Moving forward, the next and final section will explore research papers that focus on predicting returns over various horizons.

2.6 Various horizons

Most of the studies reviewed in the previous section consider only a single timescale. This is also noted by Akyildirim *et al.* (2021) in the context of cryptocurrencies. They argue that focusing solely on a single timescale misses

the opportunity to demonstrate the generalizability of machine learning algorithms across various market conditions. In their research, Akyildirim *et al.* (2021) examine the predictability of major cryptocurrencies over four different horizons, specifically 15min, 30min, 60min, and daily. This multi-timescale analysis is significant as it provides a more comprehensive understanding of the predictive capabilities of these algorithms under different trading frequencies and conditions. The study employs a combination of market data and technical indicators constructed from these data as input features for a binary classification problem, centered on predicting price movement. Zhu *et al.* (2008) further elaborate on the concept of multi-timescale analysis in their investigation of stock index increments. They emphasize the importance of incorporating trading volume as an additional input feature in neural network models, thereby enhancing the prediction accuracy across various time horizons. Their research encompasses daily, weekly, and monthly financial data from major stock indices which allows them to assess the generalizability and robustness of their neural network models. The results suggest short-term horizon performance does not improve by adding additional information, in this case, trading volume. The authors argue that this might be due to higher noise levels and high volatility of the daily data, for longer horizons, however, trading volume improves the performance. Orimoloye *et al.* (2020) compare the effectiveness of deep learning against shallow architectures, such as SVM or one-layer neural networks, across different time horizons for predicting stock price indices. The authors utilize data from 34 financial indices across 32 countries over daily, hourly, minute, and tick-level horizons. Their findings reveal that while more complex architectures show an advantage over shallow architectures in some contexts, such as with minute-level data, this superiority is not consistently observed across all time horizons. Particularly, at the tick level, the performance edge of deep neural networks diminishes, suggesting that the complexity and noise inherent in highly granular data can challenge even advanced models. The review of literature on varying time horizons in financial market prediction reveals a consensus on the importance of multi-timescale analysis. These studies highlight the varied performance of machine learning and deep learning models across different time frames. These insights suggest that while deep learning offers advantages in certain contexts, its efficacy is not uniform across all time horizons, particularly in highly granular and noisy data environments.

This concluding section wraps up the literature review chapter. Initially, we broadly explored machine learning, using three exemplary cases of machine

learning applications beyond the financial sector to illustrate the concepts of supervised, unsupervised, and reinforcement learning. Subsequently, our focus shifted to the domain of finance, where we showcased practical uses ranging from fraud detection to sentiment analysis. In the final part, we examined research related to predicting financial returns, investigating the challenges in this area and highlighting the importance of proper benchmarking. The studies indicate a preference for LSTM models in this field, however, multi-horizon analysis reveals that more complex models do not always yield superior performance.

Chapter 3

Methodology

Our research is based on the study of Krauss *et al.* (2017). We focus on forecasting the next horizon directional movements of financial instruments, framing this as a classification problem where we predict probabilities of whether the instrument will go up or down. We analyze four different prediction horizons, hourly, daily, weekly, and monthly. Based on these forecasts, we then construct our trading strategy.

In this chapter, we outline our research methodology. The first step involves feature engineering and determining the target variables. This is followed by the creation of distinct datasets for training, validation, and testing purposes. Subsequently, we explore basic machine learning models and then advance to a detailed examination of recurrent neural networks. We also cover the details of regularization and optimization processes. In the final sections, we outline the evaluation metrics and portfolio construction strategies, and provide a brief overview of the software implementation aspects. This chapter's theoretical basis draws from the books by Prado (2018) and Dixon *et al.* (2020), which provide key insights into the application of machine learning in finance. To complement these financial perspectives with a foundational understanding of machine learning methodologies, we also reference the work of Goodfellow *et al.* (2016).

3.1 Feature engineering and target selection

Input Features

Based on the findings of Hsu *et al.* (2016), technical indicators do not provide substantial predictive value in forecasting financial returns, thus, we have

chosen to exclude them from our analysis.

Our dataset includes 5-minute interval data (open, close, low, high, and volume). Following established practices in the literature, we incorporate past returns as features. This approach is supported by Fischer & Krauss (2018) and Ghosh *et al.* (2022), who utilized a sequence of past returns spanning 240 days to predict next-day market movements. The primary rationale for using returns is to deal with the nonstationarity inherent in price data. We have previously discussed the low signal-to-noise ratio characteristic of financial time series. Furthermore, as Prado (2018) notes, integer differentiation, commonly used to transform the series, tends to diminish its memory, suggesting that returns may not represent the optimal transformation of price data. The author proposes using fractional differentiation as a more effective method. To extend the existing literature, we employ fractionally differentiated series and compare their performance with traditionally used logarithmic returns. In addition to leveraging past returns, we introduce another extension over our benchmark studies (Krauss *et al.* 2017; Fischer & Krauss 2018; Ghosh *et al.* 2022) by incorporating realized volatility, calculated from 5-minute data. The third extension we introduce is the inclusion of futures as potential additional data.

Now we have 3 potential features that can be used for training our models. We first construct sequences of various lengths based on the prediction horizon. For daily data, we follow Fischer & Krauss (2018) and create sequences using the past 240 days, or approximately 1 year. For hourly we use the past 160 hours, or about 1 trading month. For weekly we opt for 100 weeks, or around 2 years, and finally, for monthly, we choose 60 months, or 5 years. During training, we try combinations of using only returns (either logarithmic or fractionally differentiated) and adding realized volatility based on data availability. In other words we either have only 1 input feature or 2 input features.

Target

Following the approach of Krauss *et al.* (2017), we create a binary target variable for each financial instrument. This target is assigned a value of 1 if the next horizon return exceeds the median return calculated across the entire dataset, and a value of 0 otherwise. This procedure is designed to predict if an instrument's performance in the next period will be above or below the dataset's median return. Further, this approach results in a balanced dataset. For example, if we made our target equal to 1 if the returns are positive and 0

otherwise then in the bull runs our dataset would contain more targets equal to 1 than 0 targets which could lead to poorer performance of our model.

3.2 Generation of training, validation, and test data

In machine learning, it is common practice to split the dataset into training, validation, and test subsets. The training set is employed for fitting the model, during this phase, the algorithm acquires patterns and establishes relationships within the data. A challenge with complex models, particularly in deep learning, is their propensity to overfit. Overfitting occurs when a model learns the training data too well, compromising its ability to generalize. To mitigate this, the validation set is used. This subset aids in fine-tuning and hyperparameter adjustments, ensuring the model performs optimally on an independent dataset. The validation process is instrumental in selecting the most effective model. Lastly, the test set serves as the final benchmark for evaluating the model's performance. It comprises real-world data that has not been used in training or tuning, thus providing an unbiased assessment of the model's efficacy in practical scenarios.

An important distinction between this application and other machine learning domains, like image classification, lies in the treatment of time-ordered data. In financial modeling, particularly when predicting returns using deep learning, the chronological sequence of the data is crucial. It is imperative to handle this time ordering meticulously to avoid *look-ahead bias*. Look-ahead bias occurs when a model uses information that would not have been available at the time of prediction, leading to misleadingly optimistic performance. Ensuring that the model is trained and validated without violating the temporal order of data is essential for its reliability and accuracy in real-world scenarios.

In a similar manner to Krauss *et al.* (2017), we construct study periods consisting of a training period, validation period, and testing, or trading, period, chronologically ordered. Here we would like to note the difference between our approach and Krauss *et al.* (2017) as the authors split the training period randomly into training and validation, hence they do not keep chronological order with training and validation data. For the reasons mentioned above, we decided not to follow this approach.

The testing period is always 1 year long and is non-overlapping, in other

words, we have a rolling window with 1-year step. This method allows us to evaluate performance across different periods, thus enhancing the robustness of our results compared to selecting a single period. Given the different horizons we are dealing with, we have varying training periods for each horizon. Specifically, for daily and hourly horizons, we use a 2-year training period, for weekly horizons, we use a 3-year training period, and for monthly horizons, we use a 5-year training period. The validation period is consistently 1 year for all horizons, matching the duration of the testing period.

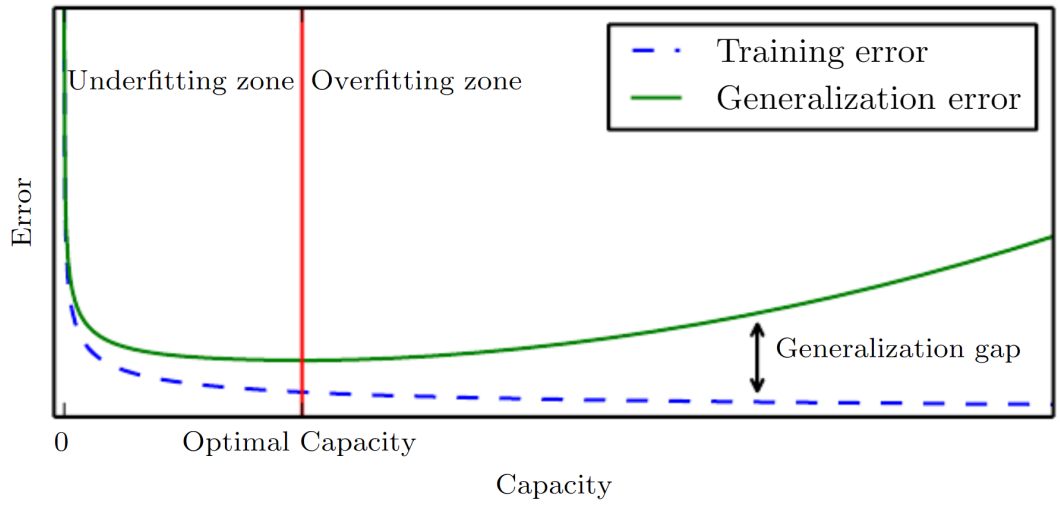
Additionally, unlike Krauss *et al.* (2017), who use only the study period data for generating past returns, we do not limit ourselves in this manner and we use returns before the start of the study period. This does not create any issues as it only extends the length of our training period in comparison with Krauss *et al.* (2017).

3.3 Models

This subsection describes the basic concepts behind machine learning (and deep learning) models. Although reinforcement learning offers significant potential in financial applications, including option pricing, market making, and portfolio optimization (Hambly *et al.* 2023). Our study will focus on supervised learning due to its direct applicability within financial return prediction.

It's essential to highlight the distinction from traditional econometric models, which rely on metrics like t-statistics, R^2, and statistical significance. In machine learning, the absence of these conventional metrics raises the question of model evaluation. The answer lies in out-of-sample forecasting, demonstrating the model's performance on unseen data. A critical consideration is a balance between model complexity and the risk of overfitting. This concept is visually depicted in Figure 3.1, illustrating the crucial equilibrium machine learning models must achieve. To find the balance we use regularization techniques which serve as a protection against overfitting by penalizing model complexity. We discuss these methods in greater detail later in the text. The complexity of a model is significantly influenced by its architecture. We begin by introducing Feedforward neural network (FNN) and then proceed to discuss the more advanced RNN.

Figure 3.1: Relationship between capacity and error



Source: Figure 5.3 of "Deep Learning" book, `https://www.deeplearningbook.org`

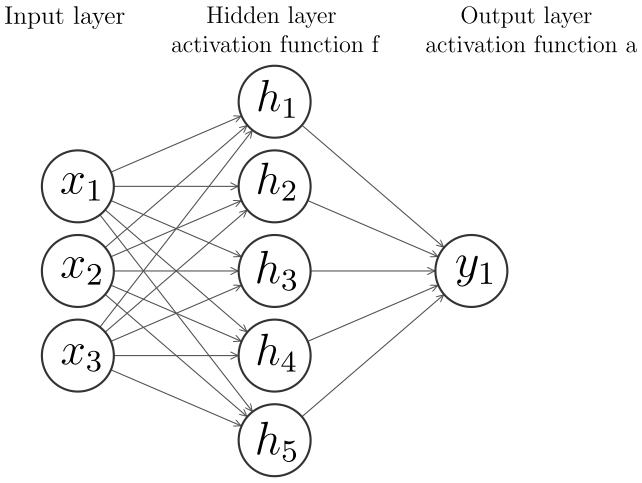
3.3.1 Feedforward neural networks

Feedforward neural networks, as the name suggests, are defined by a one-way flow of data. Hence there are no cycles within the architecture. Although this architecture is quite simple it is the cornerstone of many machine learning structures.

Figure 3.2 displays a straightforward neural network consisting of an input layer, a single hidden layer, and an output layer. The dimensionality of the input layer corresponds to the number of features in the input data. To illustrate, let's consider a height prediction model using three features: age, gender, and birth country. Consequently, the input layer will have three units, one for each feature.

After the input layer, there are one or more hidden layers. Each hidden layer has some number of units, this is then called a width. In this example, we've selected one densely connected hidden layer with five units. A densely connected layer in this context means that each unit is linked to every unit in the preceding layer. It's important to note that neural networks can contain multiple hidden layers, contributing to the network's depth. This capability to increase depth by adding more layers is what gives rise to the term *deep learning*. Each neuron in a hidden layer transforms the values from the previous layer with a set of weights and adds a bias. This process is linear as it is a basic linear combination of weights and input features. To introduce non-

Figure 3.2: Example of a fully connected feedforward neural network



linearity in the network we apply a non-linear function which is called the activation function. Without activation functions the number of layers would not matter because it would still behave like a single-layer network since it would be just a linear combination of linear functions. We describe different activation functions later in the text.

The architecture completes with an output layer. Our example features a single unit, tailored to our goal of forecasting a single quantity: the height of an individual. For binary classification, we would also have a single neuron, for multi-class classification we would have multiple neurons. Again activation function can be applied depending on the task, in our example, we would not apply any activation function as we are doing regression. However, for classification tasks, it is crucial to apply the activation function.

In the simple example, we explained how forward propagation works. More formally this process can written as:

$$h_i = f\left(\sum_j x_j w_{j,i}^{(h)} + b_i^{(h)}\right) \tag{3.1}$$

$$y_i = a\left(\sum_j h_j w_{j,i}^{(y)} + b_i^{(y)}\right) \tag{3.2}$$

Given that $x_1, x_2, \ldots, x_n$ are input features, the model can be described with the following components:

- f: the activation function of the hidden layer,

- h_i: the i-th node in the hidden layer,
- $w_{j,i}^{(h)}$: the weight from the j-th input node to the i-th hidden layer node,
- $b_i^{(h)}$: the bias of the i-th hidden layer node,
- y_i: the i-th output node,
- $w_{j,i}^{(y)}$: the weight from the j-th hidden layer node to the i-th output node,
- $b_i^{(y)}$: the bias of the i-th output node,
- a: the activation function of the output layer.

When we train the neural network we want to learn the weights and biases. First, we have to define a loss function that measures how far the networks's predictions are from the actual values. The goal of the network is to minimize this loss. The loss function depends on the problem we are solving, we will get back to this later in the text. The learning algorithm is called backpropagation and it involves calculating the gradient of the loss function with respect to each weight using the chain rule. It is called backpropagation because we are moving backward from the output layer to the input layer. A simple interpretation of this process is that we first calculate the loss in the forward propagation and then by backpropagation, we are adjusting the weights to decrease the loss. We can use various optimization algorithms which we discuss later.

Activation Functions

Activation functions are key for neural networks because they introduce non-linearity to the system. This allows the neural network to capture more complex patterns in the data. For output layers, we usually use the following activation functions:

1. Linear: $f(x) = x$, used for regression tasks.
2. Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$, used for binary classification.
3. Softmax: $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$, used for multiclass classification.

For hidden layers, the most popular activation functions are:

1. Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$.

2. Hyperbolic Tangent (tanh): $f(x) = 2\sigma(2x) - 1$ where $\sigma(x)$ is the sigmoid function.

3. Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$.

but there are much more for example Gaussian Error Linear Unit (GELU), Exponential linear unit (ELU), or Leaky rectified linear unit (Leaky ReLU). Choosing the proper activation function for hidden layers has an important effect on the model's ability to learn the data. Selecting the output layer's activation function is a more straightforward process as it depends on the task.

Loss Function

The loss function allows the model to measure the distance between the actual and predicted values. Usually, loss functions are derived using the maximum likelihood framework and the selection of the loss function is related to the task. For regression, we often use well-known mean squared error (MSE) or mean absolute error (MAE). Our problem is however binary classification hence we will use binary cross-entropy loss, also known as log loss. Log loss is a fundamental loss function for binary classification tasks as it quantifies the difference between two probability distributions. Formally we define log loss as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right) \tag{3.3}$$

where N is the total number of observations, y_i, and p_i are the actual value and predicted probability for the i-th observation, respectively. The interpretation of log loss is quite intuitive: when $y = 1$ (in our case, the return is positive) then the loss is equal to $-\log(p)$, hence as p approaches 1 the loss approaches 0. At the same time, as p approaches 0, then the loss increases, thus penalizing the wrong prediction. The same holds when $y = 0$.

Optimization

Now that we defined the loss function we can finally describe the optimization process. Put simply optimization algorithms are used to update weights and biases in a way that minimizes the loss function. The selection of an optimization algorithm plays a crucial role in machine learning as it can significantly impact both the speed and the quality of the training process. The backbone of most

optimization algorithms is gradient descent. The core principle of gradient descent is iteratively adjusting the weights and biases in the opposite direction of the gradient. The algorithm starts with an initial set of weights and biases (parameters). These are usually initialized randomly or set to 0. Then the gradient of the loss function is calculated. We can interpret the gradient as a slope of the loss function with respect to each parameter. Then parameters are adjusted in the opposite direction of the gradient by some step. The size of this step is controlled by a hyperparameter called learning rate. The learning rate can of course change during optimization, usually, we want to decrease the learning rate during training. Such a sequence of learning rates denoted as α_i should fulfill the following conditions in order to guarantee convergence to the local optimum:

$$\forall i : \alpha_i > 0, \quad \sum_i \alpha_i = \infty, \quad \sum_i \alpha_i^2 < \infty.$$

The algorithm repeats the steps of calculating the gradient and updating the parameters until a stopping criterion is met. This is another hyperparameter that is chosen before training. It can be either a number of iterations, also known as a number of epochs. Or a threshold below which improvements are no longer significant, we will call this hyperparameter early stopping. There are various variants of gradient descent, but the most notable are the following.

(Minibatch) Stochastic Gradient Descent (SGD): Using the whole training data to compute the gradient can be in some cases very computationally demanding and slow. For this reason, minibatch SGD calculates the gradient using smaller subsets of the training data. The size of these subsets is called batch size and is another hyperparameter that needs to be set before training.

SGD with Momentum: This variant of SGD not only considers the current gradient in its updates but also incorporates a fraction of the previous gradients. This approach helps in smoothing out the updates, reducing oscillations and potentially speeding up convergence.

RMSProp: This method is particularly effective in dealing with the issue of drastically different learning rates for different parameters, which can be a problem in SGD. RMSProp modifies the learning rate for each parameter by dividing it by an exponentially decaying average of the squared gradients. This adaptive learning rate approach helps to stabilize the updates and prevents the learning rate from diminishing too quickly.

ADAM: This algorithm combines concepts from both Momentum and RMSProp. Essentially, it maintains an exponentially decaying average of past gradients (similar to momentum) and uses squared gradients to scale the learning rate (like RMSProp). ADAM is usually quite robust to the choice of hyperparameters due to its bias corrections to the first two moments to counteract their initialization which helps in the early stages of training.

All of these algorithms are heavily used in machine learning, however, there is no consensus on which algorithm is the best. For this reason, we have to try different and select the best for our problem based on our results.

Batch Normalization

We conclude the Optimization subsection with Batch Normalization first introduced by Ioffe & Szegedy (2015). The primary motivation for this method is to counteract the internal covariate shift that happens during training when the distribution of each layer's inputs changes due to updates in the parameters of preceding layers, thus complicating the training dynamics. To mitigate this, Batch Normalization normalizes the inputs for each mini-batch to ensure consistent mean and variance across the input layers. This technique not only speeds up the training process but also brings stability to it, allowing for higher learning rates and less strict requirements for initialization. Further, the authors argue that in some cases it can act as a regularizer which brings us to the next subsection.

Regularization

In machine learning, overfitting is a common challenge where models perform well on training data but poorly on unseen data. To address this, regularization techniques are employed to decrease the validation error, sometimes at the cost of an increased training error. This subsection outlines several key regularization strategies used in this thesis to enhance model generalization.

L1, L2 parameter regularization

L1 regularization, commonly referred to as Lasso regression, and L2 regularization, known as Ridge regression, are foundational techniques in machine learning that modify the loss function by adding a penalty proportional to the magnitude of the model parameters. Lasso regression adds a penalty equivalent to the absolute value of the coefficients' magnitudes. One of the distinctive

outcomes of L1 regularization is its tendency to produce sparse models. Sparsity occurs because the L1 penalty can force some of the weight coefficients to become exactly zero, effectively performing feature selection. On the other hand Ridge regression, adds a penalty equal to the square of the magnitude of the coefficients. Unlike L1, the L2 approach does not zero out coefficients but rather shrinks them uniformly. This helps in enhancing prediction accuracy by maintaining smaller model weights, which prevents overfitting.

Early stopping

We have already briefly mentioned early stopping, however because of its importance in the training process we want to explain it in greater detail. This method is crucial not only for preventing overfitting but also for improving computational efficiency. Early stopping works by continuously observing the model's performance on a validation set at each epoch during training. If the performance, as measured by a chosen metric, such as validation loss or accuracy, fails to improve for a designated number of consecutive epochs, known as the patience period, the training is stopped. This approach assumes that the validation set offers an unbiased assessment of the model's capability to generalize to new, unseen data.

Dropout

Dropout is another widely utilized regularization technique that offers a computationally efficient way to mitigate overfitting in neural networks. This strategy randomly deactivates a subset of neurons in the network during each training iteration, reducing the network's complexity temporarily. Such random deactivation prevents neurons from becoming overly dependent on the specific patterns of the training data. When a neuron is deactivated, it does not participate in the forward pass nor is its weight updated during backpropagation. This introduction of randomness acts as noise, adding robustness to the training process. The dropout rate hyperparameter is set beforehand and it is a fraction of neurons in a layer that are set to zero at each training step.

Ensemble

According to Prado (2018), machine learning models typically exhibit three types of errors: bias, variance, and noise. Noise, which is inherent and random, cannot be mitigated by models. However, strategies exist to reduce bias and

variance errors effectively. High bias occurs when a model overlooks significant patterns in the data, resulting in underfitting. Conversely, high variance indicates overfitting, where slight variations in the training data can cause drastic changes in predictions.

One common method to decrease variance is model averaging. This technique involves training multiple models independently and then averaging their outputs. Even using the same model with different initial conditions can yield more stable results, as these models are less likely to replicate identical errors on a test set. Additionally, employing various models can further enhance this effect.

Another technique, bootstrap aggregating, also known as bagging, involves creating several datasets from the original by sampling with replacement. Each model is then trained on a different dataset, which helps in reducing variance by diversifying training data. These are methods that can effectively reduce the variance in forecasts hence addressing overfitting (Prado 2018).

Boosting is another technique that has the power to reduce both bias and variance in predictive models. But it also carries a heightened risk of overfitting. In finance, however, we usually have to deal with overfitting rather than underfitting (Prado 2018) hence we will not use boosting. A well-known example of a boosting algorithm is AdaBoost.

3.3.2 Recurrent Neural Networks

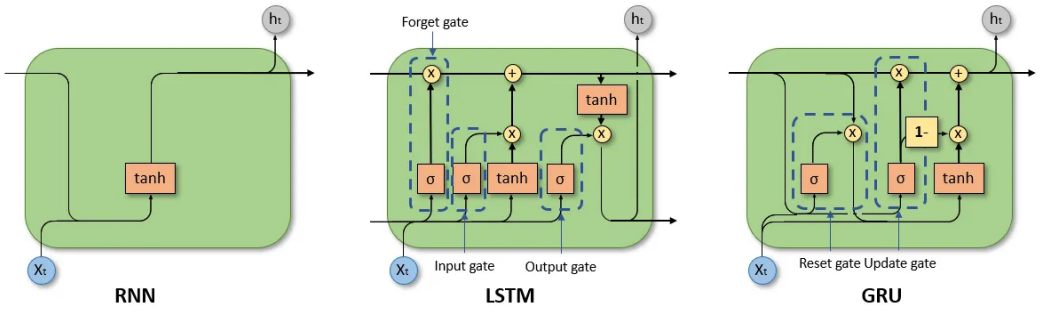
In previous sections, we explained the fundamentals of machine learning focusing on feedforward neural networks and how they are trained including various optimization and regularization techniques. Now we can move to more complex architecture which became popular in the context of financial returns predictions (Jiang 2021; Thakkar & Chaudhari 2021) and that are RNN. Unlike FNN where the flow of information is one-directional, RNN have connections that loop backward, effectively allowing these networks to maintain a form of memory that is well-suited for time series analysis (Dixon *et al.* 2020). Moreover, RNN are flexible with respect to sequence length, which proves beneficial in various applications, particularly in natural language processing.

The network combines the current input with the previously stored hidden state, this stored information acts as the network's memory. In mathematical terms, we have:

$$H_t = f(W_{hh}H_{t-1} + W_{hx}X_t + b),$$

where H_t is the current hidden state, W_{hh} and W_{hx} are the weight matrices, H_{t-1} is the previous hidden state, X_t is the input, and b is the bias. This recursive formula is central to the RNN's ability to handle data where context and order matter. Unfortunately, there are also challenges such as vanishing or exploding gradients. These issues occur when gradients diminish or increase exponentially across many layers, due to the repetitive application of the same function through the network's depth. To mitigate these problems, more sophisticated variants of RNNs, like LSTM units and Gated Recurrent Unit (GRU), have been developed. LSTM use gates (input, forget, and output) to regulate the information flow, helping to maintain stable gradients over time. GRU simplify this design by combining gates and state updates, allowing for efficient learning with fewer parameters. Figure 3.3 illustrates the differences between simple RNN, LSTM, and GRU architecture.

Figure 3.3: RNN, LSTM, and GRU cells



Source: https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4

3.4 Evaluation metrics and portfolio construction

We already described that we are predicting stock price movements, that is whether the price will go up or down in the next horizon. Hence binary classification problem. The literature suggests the following evaluation metrics for this kind of problem: accuracy, precision, recall, F1 score, area under the ROC curve, or confusion metrics. Accuracy is a simple metric representing the proportion of correct predictions out of all predictions made, as we are not working with an unbalanced dataset this metric is reasonable for our use case. Our model is better than a random guess if the accuracy is larger than 50% at a reasonable significance level.

Further, we follow Krauss *et al.* (2017) approach to portfolio construction. We buy k instruments with the highest probability of positive returns in the next period and short-sell k instruments with the lowest probability of positive results. We acknowledge the simplicity of this approach but following the methodology of previous studies allows us to compare the results. We expect the accuracy of this portfolio to be larger than that of the whole sample because we are selecting only the instruments with the highest (lowest) probability.

3.5 Software implementation

We use Python and TensorFlow for data handling and modeling. Python's extensive libraries and tools are well-suited for data analysis, machine learning, and financial modeling. TensorFlow, as an open-source library, efficiently implements neural network architectures. The literature further supports Python as the preferred language for our research field (Sezer *et al.* 2020). Google Colab is our primary environment for training models, offering access to powerful GPUs and TPUs without requiring local computational resources.

Chapter 4

Data

In the empirical section of our thesis, we utilize 5-minute adjusted price data for stock splits and dividends, obtained from Kibot. To ensure comparability with our benchmark studies (Krauss *et al.* 2017; Fischer & Krauss 2018; Ghosh *et al.* 2022), we focus on companies that were part of the S&P 500 index at the beginning of each study period. We sourced the historical constituents of the S&P 500 index annually from 1998 through 2022 using the Reuters Refinitiv Data Platform API.

Additionally, we expand our analysis to include futures as potential instruments in our trading strategy. We utilize continuous contracts to accommodate various maturities and select 20 different futures, matching them to the trading hours of our stocks. Our dataset includes stock data starting from 1998 or from the company's initial public offering date, while futures data begins around 2009.

Further, we select only such stocks and futures that have at least 5 years of available data. Although we have data back to 1998 we can not construct a fractionally differentiated series for other than hourly data right from 1998 hence we start with the first study period in 2003 and the last one in 2020. Thus, at most, we have 18 study periods in total, but for weekly and monthly horizons, the number is lower due to data availability. For each study period, we use all the stocks (and futures) that were part of the S&P 500 index at the beginning of the study period. Using all the stocks together rather than selecting a few stocks and training individual models on them brings undeniable benefits. Most importantly, the models are trained on much larger datasets, allowing for more robust and generalizable conclusions as this approach leads to less biased results caused by outliers or overfitting.

Table 4.1: Number of available stocks (futures)

Horizon type		*Hourly*			*Daily*			*Weekly*		*Monthly*
d	0.30	0.50	1.00	0.50	0.60	1.00	0.65	0.80	1.00	1.00
2003	232 (0)	323 (0)	326 (0)	259 (0)	290 (0)	321 (0)	0 (0)	0 (0)	312 (0)	0 (0)
2004	297 (0)	335 (0)	337 (0)	286 (0)	311 (0)	332 (0)	0 (0)	0 (0)	326 (0)	0 (0)
2005	303 (0)	344 (0)	347 (0)	302 (0)	324 (0)	341 (0)	0 (0)	276 (0)	333 (0)	292 (0)
2006	292 (0)	360 (0)	364 (0)	296 (0)	327 (0)	359 (0)	0 (0)	259 (0)	329 (0)	316 (0)
2007	213 (0)	380 (0)	383 (0)	236 (0)	314 (0)	379 (0)	0 (0)	289 (0)	355 (0)	343 (0)
2008	313 (0)	417 (0)	422 (0)	317 (0)	367 (0)	398 (0)	0 (0)	320 (0)	381 (0)	357 (0)
2009	345 (0)	428 (0)	432 (0)	307 (0)	371 (0)	426 (0)	261 (0)	343 (0)	394 (0)	364 (0)
2010	417 (0)	439 (0)	461 (20)	385 (0)	401 (0)	438 (0)	264 (0)	342 (0)	424 (0)	379 (0)
2011	416 (0)	460 (20)	463 (20)	391 (0)	427 (0)	461 (20)	328 (0)	382 (0)	437 (0)	389 (0)
2012	354 (17)	459 (20)	462 (20)	394 (0)	431 (0)	460 (20)	320 (0)	381 (0)	458 (20)	390 (0)
2013	424 (16)	468 (20)	468 (20)	411 (0)	433 (0)	466 (20)	328 (0)	389 (0)	460 (20)	423 (0)
2014	412 (13)	469 (20)	473 (20)	410 (0)	446 (19)	468 (20)	350 (0)	414 (0)	463 (20)	420 (0)
2015	446 (20)	479 (20)	479 (20)	444 (19)	457 (20)	477 (20)	347 (0)	425 (0)	470 (20)	441 (20)
2016	352 (16)	478 (20)	485 (20)	428 (18)	454 (19)	480 (20)	349 (0)	423 (0)	474 (20)	444 (20)
2017	440 (18)	489 (20)	492 (20)	456 (19)	475 (20)	489 (20)	373 (0)	455 (20)	484 (20)	460 (20)
2018	434 (17)	492 (20)	495 (20)	461 (17)	478 (18)	491 (20)	390 (0)	457 (20)	488 (20)	456 (19)
2019	438 (14)	501 (20)	502 (20)	476 (18)	489 (18)	501 (20)	409 (0)	467 (20)	497 (20)	477 (20)
2020	332 (13)	499 (20)	501 (20)	479 (19)	494 (20)	500 (20)	416 (0)	476 (20)	500 (20)	481 (19)

Note: The table shows a number of stationary stocks and futures (in parentheses) available for each horizon and differentiation factor d. The year in the first column denotes the start of the training period.

As outlined in Chapter 3, we extend the existing literature by incorporating realized volatility data and fractionally differentiated time series as potential input features, rather than relying solely on logarithmic returns. However, we encounter certain limitations with different prediction horizons. For hourly data, calculating realized volatility from only 12 points (since we are using 5-min data) is insufficient, thus it is not used. Similarly, for monthly data, we face constraints with fractional differentiation, as its calculation would significantly reduce our sample size.

We calculate various fractionally differentiated series (differentiation factor d) for each prediction horizon and test for the stationarity of these series as well as for logarithmic returns ($d = 1$) for each training period. Note that we do not test stationarity for validation and test periods, as this would involve looking at data that we could not have seen during training, potentially leading to look-ahead bias.

Table 4.1 shows the number of stationary stocks and futures for each study period. We observe that the lower the d value, the fewer stocks (and futures) remain stationary for each study period. Interestingly, the stationarity of the majority of instruments is maintained even for relatively low d values. For instance, with hourly data, a d as low as 0.3 still results in most instruments being stationary. This suggests that logarithmic returns ($d = 1$) might not be

optimal, as they may unnecessarily remove too much information.

For daily data, a d of 0.5 is sufficient for maintaining stationarity in most instruments. For weekly data, a higher d is required due to the loss of data when calculating these series, but a d of 0.65 offers a good balance between stationarity and the number of available instruments. For monthly data, we decided not to use fractionally differentiated series due to data availability constraints, thus we only report the number of stationary instruments for logarithmic returns ($d = 1$).

During training, we always use the fractionally differentiated series with the lowest d value and traditional logarithmic returns. Specifically, for hourly horizons, we use $d = 0.3$ and $d = 1$, for daily horizons, $d = 0.5$ and $d = 1$, for weekly horizons, $d = 0.65$ and $d = 1$, and for monthly horizons, only $d = 1$.

Figure 4.1 further contrasts fractionally differentiated series and logarithmic returns for Microsoft on hourly, daily, weekly, and monthly horizons. We show the optimal d values that we described previously. We observe that fractionally differentiated series exhibit more memory, especially for lower d values. These series are somewhat similar to the actual price series, showing long-term dependencies evident from an upward trend, but they remain stationary. At higher d values, such as $d = 0.65$ displayed on the weekly horizon, most of the memory seems to diminish.

Figure 4.2 then shows realized volatility for Microsoft over daily, weekly, and monthly horizons. We observe periods of higher volatility, particularly in the early 2000s during and after the dot-com bubble, followed by a relatively calm period until the Great Financial Crisis in 2008, and then increased volatility during the COVID-19 crisis. The aggregation of volatility data in the weekly and monthly plots smooths the intensity and frequency of these spikes, but the overall pattern remains consistent, capturing the broader impacts of market events.

Finally, we construct sequences for various prediction horizons using realized volatility and returns (or fractionally differentiated series). Before constructing these sequences, we standardize our data to have a mean of 0 and a standard deviation of 1, more formally:

$$X_{\text{scaled}} = \frac{X - \mu_{\text{train}}}{\sigma_{\text{train}}}$$

Where X represents either realized volatility, returns, or fractionally differentiated series for an individual stock or a futures contract for a specific

Figure 4.1: Logarithmic returns and Fractional differentiated series - Microsoft

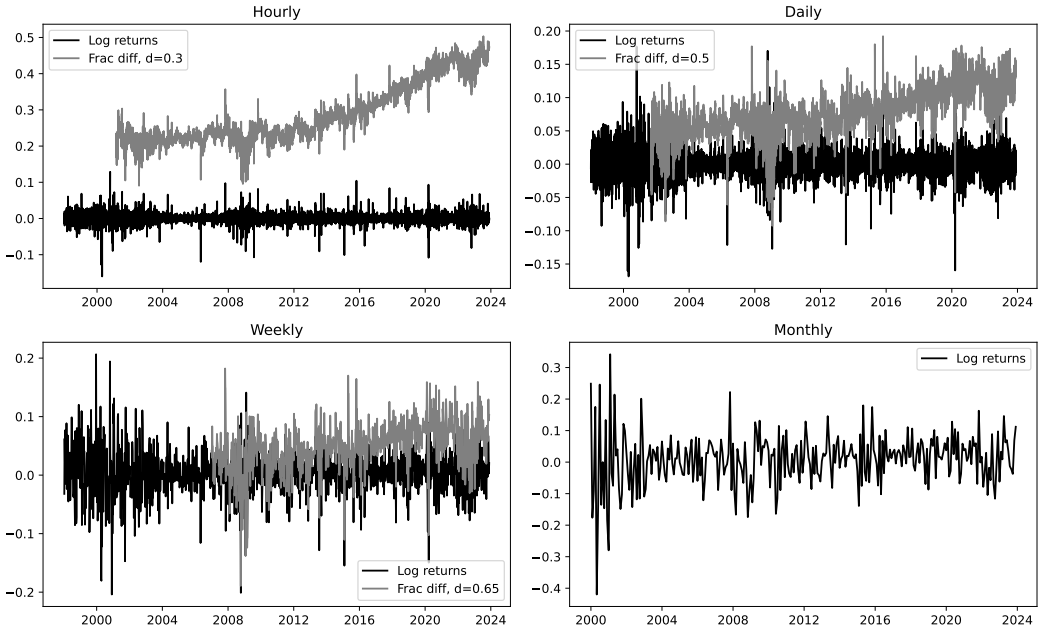
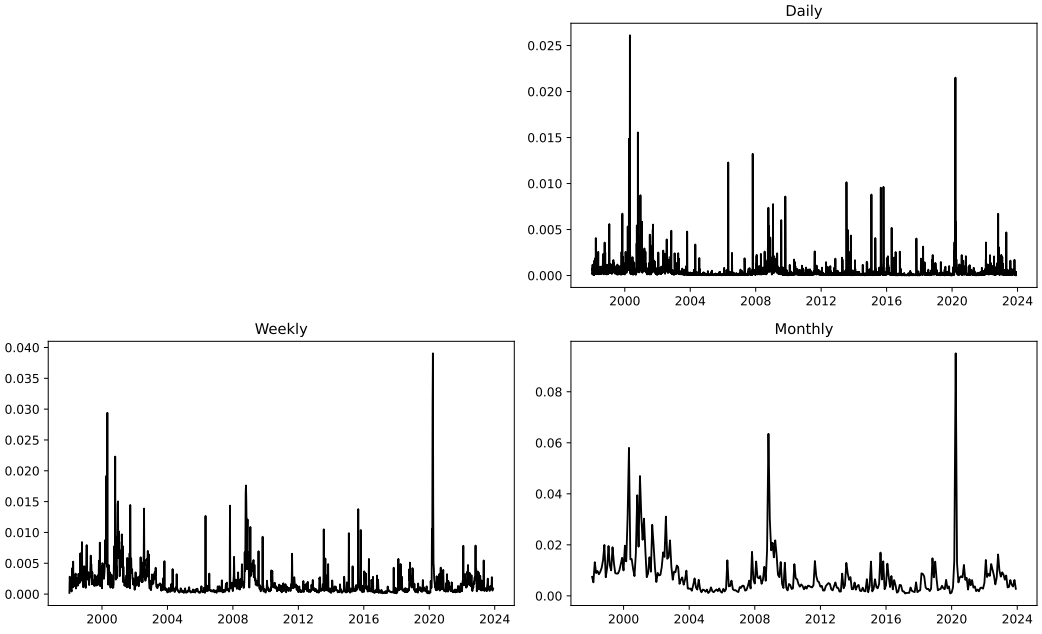


Figure 4.2: Realized volatility - Microsoft



date and time. Further $\mu_{\text{train}}, \sigma_{\text{train}}$ denote the mean and standard deviation of the series calculated on the training data, respectively. We use only training data for calculating the mean and standard deviation to avoid look-ahead bias, which could occur if we used the validation or test data for standardization. However, as a result, the standardized validation and test data might not have a mean and standard deviation exactly equal to 0 and 1.

Chapter 5

Results

In this chapter, we first describe the training process and comment on the results of individual prediction horizons. Specifically, we discuss metrics such as accuracy, mean returns, financial performance, and the impact of incorporating futures into our models. We then compare these metrics across multiple models for each horizon, based on the number and type of input features. Comparing various models serves two purposes. First, it allows us to evaluate the effect of different input features on the final outcomes. Second, it helps ensure the robustness of our results. By testing multiple models, we reduce the risk that our findings are merely a result of random chance or overfitting specific to a single model, thereby strengthening the validity and reliability of our conclusions.

After analyzing individual horizons, we compare the results across all prediction horizons, focusing on accuracy, annualized returns, and annualized standard deviation. Given that the daily horizon is the most commonly used in the literature and our benchmark studies, we begin with a detailed examination of the daily horizon.

5.1 Daily horizon

Training

In Chapter 4, we discussed various regularization techniques and model architectures. After hyperparameter tuning, we selected the following configuration: an input layer with either 1 feature (returns or fractionally differentiated series) or 2 features (adding realized volatility), with an input length of 240. The LSTM layer consists of 28 units, batch normalization, and a 10% dropout rate. For the 2-feature setting, features are concatenated. This is followed by a fully

connected layer with 8 units and ReLU activation function, and an output layer with sigmoid activation for binary classification. We use the ADAM optimizer with a learning rate of 0.001, a batch size of 1024, and 100 epochs. Early stopping after 20 epochs based on validation loss restoring the best weights, with a warm-up period of 10 epochs. We avoid L1 and L2 regularization as our other techniques effectively prevent overfitting. After extensive testing of different configurations and models, including LSTM, GRU, and simple RNN, we found this architecture best suited to our data.

Similarly to our benchmark studies, we use the same architecture for all the study periods. We could of course test different architectures for every study period, but instead, we choose to study the feature importance also highly stressed by Prado (2018). For each of the 18 study periods, we trained up to 8 different models based on input features. These models were trained using returns or fractionally differentiated series (we will denote fractionally differentiated series as $Returns_d$), further with either a single feature or a two-feature setting that included realized volatility. Additionally, starting in 2015, we incorporated futures data, training models on both stock data alone and combined stock and futures data. For every input specification, we trained 3 models with different random seeds and then ensembled these to reduce overfitting. In total, we trained 104 ensembled models, which took approximately 34 hours using Google Colab's T4 GPU.

Evaluation

In this section we evaluate 4 different models based on their input features. DM1 uses only *Returns*, hence it has only one input feature. DM2 uses *Returns* and realized volatility (RV), thus incorporating two input features. DM4 and DM5 use fractionally differentiated returns ($Returns_{0.5}$) with either one or two input features.

Additionally, we construct 5 ensembles. DM9 includes all four models. DM3 and DM6 combine models with the same returns type (DM1 and DM2 for *Returns*, DM4 and DM5 for $Returns_{0.5}$). DM7 and DM8 combine models with the same number of input features: DM7 ensembles DM1 and DM4, and DM8 ensembles DM2 and DM5.

These models predict whether the next day's returns will be above or below the median. Based on these probabilities, we select 10 stocks (or stocks and futures) with the highest probability of exceeding the median returns and 10

Table 5.1: Daily accuracy

Returns type		*Returns*			$Returns_{0.5}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	$DM1$	$DM2$	$DM3$	$DM4$	$DM5$	$DM6$	$DM7$	$DM8$	$DM9$
2006	54.442***	54.163***	54.402***	53.506***	52.47***	52.829***	53.665***	53.805***	54.522***
2007	52.171***	52.888***	52.61***	52.908***	52.131***	52.41***	51.873***	51.594**	51.673***
2008	52.391***	53.162***	53.083***	52.569***	52.846***	52.569***	52.885***	52.846***	53.498***
2009	51.508**	51.567**	52.262***	52.52***	51.31**	51.349**	51.627**	51.746***	51.19**
2010	51.976***	52.579***	52.956***	51.468**	51.548**	51.746***	49.802	50.298	50.238
2011	51.825***	51.19**	52.262***	51.508**	52.164***	51.925***	51.369**	51.845***	51.726***
2012	51.64**	50.56	51.82***	52.0***	52.78***	52.28***	50.76	51.58**	51.44**
2013	50.714	52.123***	51.448**	51.389**	51.627**	52.004***	51.25**	51.389**	51.885***
2014	52.103***	51.19**	50.675	51.825***	51.29**	50.774	51.587**	51.548**	51.27**
2015	51.012*	51.865***	51.528**	51.25**	52.817***	53.016***	51.548**	51.746***	52.738***
2016	49.107	52.163***	50.496	50.417	51.171**	50.317	50.694	51.667***	50.952*
2017	52.45***	50.697	51.653***	51.514**	51.315**	51.076*	51.574**	50.159	50.837
2018	49.183	49.303	49.92	51.554**	50.737	51.454**	50.757	49.641	49.442
2019	51.706***	50.317	50.417	52.758***	50.139	51.925***	52.063***	49.683	51.369**
2020	53.202***	52.138***	52.984***	52.095***	52.47***	52.391***	52.866***	51.561**	52.609***
2021	50.437	51.071*	50.278	49.881	51.885***	51.885***	51.032*	51.27**	51.31**
2022	50.518	50.219	51.275**	50.797	50.876	50.916*	50.339	50.02	49.88
2023	51.991***	52.345***	52.146***	52.367***	51.151*	52.721***	52.81***	52.168***	52.522***
Whole period	51.574***	51.638***	51.788***	51.792***	51.71***	51.861***	51.577***	51.361***	51.613***

Note: The table shows the daily accuracy in percentage (%) of the k = 10 portfolio for each study period and for all model variants based on the number and type of input features. A binomial test was used to compare the accuracy against a 50% random chance, with statistical significance indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. The year in the first column denotes the study period.

with the lowest probability. This selection is referred to as the k portfolio, with $k = 10$ chosen for comparability with our benchmark studies.

Accuracy

Table 5.1 presents the accuracy of the k portfolio for all test years, focusing on models that predict only stocks. Accuracy is measured by the proportion of correctly predicted directions, where 50% indicates random predictions. Higher accuracy indicates better performance. To determine if the predictions are better than random chance, we perform a binomial test. The null hypothesis for this test is that the accuracy is 50%, meaning the predictions are no better than random. If we reject the null hypothesis, we can conclude that the accuracy is statistically significantly different from random chance.

Additionally, following Krauss *et al.* (2017), we also conducted the Pesaran-Timmermann (PT) test, which evaluates whether the predictions and actual outcomes are independently distributed. Since the results from the binomial and PT tests indicate the same significance brackets, we only report the binomial test for clarity.

Our findings suggest that, over the years, most models show statistically

significant predictive accuracy. Each year, at least one model demonstrates significant accuracy, although no single model consistently performs significantly across all years. During periods of high market volatility, such as the Great Financial Crisis in 2008 and the COVID-19 pandemic in 2020, the models exhibited higher predictive accuracy. This shows the models' ability to adapt and perform well under turbulent market conditions, capturing extreme market movements effectively. Conversely, certain years show weaker predictive accuracy, suggesting that the models struggled to identify clear patterns or trends. This trend is especially notable in 2016, 2018, and 2022. This lower performance could be attributed to a relatively stable market with fewer distinct patterns for the models to capture.

When analyzing model performance across all years, several trends emerge. Ensemble models generally outperform their components, often achieving higher or comparable accuracy rates. The highest accuracy over the entire period from 2006 to 2023 of 51.86%, comes from an ensemble that combines fractionally differentiated returns ($Returns_{0.5}$) with both one and two input features (DM6). Another strong performer over the same period, with an accuracy of 51.79%, is an ensemble that combines models using *Returns* with both one and two input features (DM3).

When comparing by year, the *Returns* ensemble (DM3) consistently outperforms the $Returns_{0.5}$ ensemble (DM6) before 2012. However, post-2012, $Returns_{0.5}$ models deliver higher predictive accuracy. Based on this observation our preferred ensemble is the combination of $Returns_{0.5}$ with one input feature model and $Returns_{0.5}$ with two input features model (DM6).

Ensembles involving both *Returns* and $Returns_{0.5}$ models (DM7-DM9) perform worse than the aforementioned ensembles.When comparing one-input feature models to two-input feature models, there is no clear advantage as the *Returns* model benefits from two input features, whereas the $Returns_{0.5}$ model performs better with one feature.

It is important to note that while we compare different models, their predictions are often very similar. This means that, on average, the differences in their predictive performance may not be statistically distinguishable.

Financial performance

Table 5.2 shows the mean daily returns before transaction costs along with an indication of whether the returns are significantly greater than zero, with

Table 5.2: Mean daily returns

Returns type		*Returns*			*Returns*$_{0.5}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	*DM*1	*DM*2	*DM*3	*DM*4	*DM*5	*DM*6	*DM*7	*DM*8	*DM*9
2006	0.288***	0.266***	0.27***	0.175***	0.099*	0.11*	0.216***	0.236***	0.25***
2007	0.09	0.151*	0.138*	0.145**	0.148**	0.12*	0.068	0.108	0.089
2008	0.315	0.166	0.321*	0.271**	0.333*	0.241*	0.512**	0.285*	0.424**
2009	-0.058	0.092	0.098	0.107	-0.132	-0.011	-0.083	0.083	0.025
2010	-0.001	0.076	0.033	0.096*	0.045	0.106*	0.037	0.086	0.11*
2011	0.209**	0.164**	0.237***	0.067	0.24***	0.148**	0.128*	0.235***	0.22***
2012	0.181***	0.004	0.088	0.204***	0.086	0.054	0.151**	0.039	0.058
2013	0.024	0.083*	0.064	0.093**	0.062	0.108**	0.072	0.07	0.106*
2014	0.099	0.053	0.037	0.11**	0.142***	0.114**	0.054	0.118**	0.097**
2015	0.002	0.058	0.037	0.214**	0.269***	0.329***	0.227***	0.207***	0.318***
2016	-0.051	0.162	0.165	-0.16	0.074	-0.073	-0.041	0.106	0.118
2017	0.148**	-0.044	0.012	0.108*	0.039	0.08	0.08	0.012	0.044
2018	-0.127*	-0.169**	-0.103	0.107*	0.005	0.038	0.012	-0.071	-0.079
2019	0.133	0.047	0.098	0.218***	0.123	0.106	0.255***	0.137	0.221**
2020	0.206*	0.024	0.09	0.249*	0.213*	0.218*	0.294**	0.091	0.266**
2021	0.023	0.033	-0.002	-0.055	0.137	0.135	0.018	0.015	0.021
2022	0.073	-0.024	0.084	0.042	-0.017	-0.027	-0.061	-0.03	-0.059
2023	0.242**	0.166**	0.213**	0.085	0.136	0.187*	0.358***	0.182**	0.251**
Whole period	0.099***	0.072***	0.104***	0.116***	0.111***	0.11***	0.127***	0.106***	0.137***

Note: The table shows the mean daily returns in percentage (%) of the k = 10 portfolio for each study period and for all model variants based on the number and type of input features. Statistical significance is indicated as follows: *p $<$ 0.1; **p $<$ 0.05; ***p $<$ 0.01. These values are calculated using Newey-West standard errors with a one-lag correction. The year in the first column denotes the study period.

critical values calculated using Newey-West standard errors with a one-lag correction.

The results are less robust compared to accuracy metrics, yet all models deliver statistically significant returns over the entire period. The ensemble model DM3 improves upon the *Returns* models (DM1 and DM2), whereas DM6 does not enhance the *Returns*$_{0.5}$ models (DM4, DM5). The clear winner over the whole period is ensemble DM9, comprising all four core models, with a mean daily return of 0.137%. DM7, an ensemble of DM1 and DM3, also shows promising average returns of around 0.127%. Additionally, the *Returns*$_0$.5 family models (DM4-DM6) outperformed the *Returns* family models (DM1-DM3).

When examining individual years, most models achieve positive returns in the majority of years, but these returns are generally insignificant, with some exceptions.

Table 5.3 details financial performance before and after transaction costs over the entire period. To ensure comparability with benchmark studies, we apply a transaction cost of 0.05% for both buying and selling. This results in a daily transaction fee of 0.2%, as four transactions are made each day: buying

Table 5.3: Financial performance

Returns type		*Returns*			$Returns_{0.5}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	$DM1$	$DM2$	$DM3$	$DM4$	$DM5$	$DM6$	$DM7$	$DM8$	$DM9$
Before transaction costs									
Mean returns (%)	0.099***	0.072***	0.104***	0.116***	0.111***	0.11***	0.127***	0.106***	0.137***
Buy mean returns (%)	0.083***	0.065**	0.081***	0.086***	0.074***	0.074***	0.103***	0.08***	0.092***
Sell mean returns (%)	0.016	0.007	0.023	0.03	0.037	0.036	0.024	0.026	0.045*
Minimum (%)	-15.982	-17.369	-15.946	-12.697	-19.414	-12.578	-15.144	-13.777	-14.132
Quartile 1 (%)	-0.840	-0.778	-0.836	-0.642	-0.732	-0.706	-0.736	-0.741	-0.720
Median (%)	0.078	0.075	0.097	0.107	0.093	0.115	0.109	0.107	0.126
Quartile 3 (%)	1.008	0.942	1.037	0.878	0.954	0.917	0.980	0.976	0.984
Maximum (%)	21.065	14.738	20.985	14.550	14.137	14.385	14.214	13.510	12.477
Standard dev. (%)	2.052	1.802	1.995	1.659	1.809	1.752	1.886	1.751	1.813
Skewness	0.462	-0.071	0.359	-0.040	-0.057	0.202	0.027	0.121	0.048
Excess Kurtosis	8.725	4.279	6.732	5.986	6.623	4.469	5.343	3.371	3.370
Annualized									
Returns (%)	21.531	15.036	23.438	28.993	26.705	26.695	31.251	25.393	35.340
Standard dev. (%)	32.580	28.599	31.671	26.343	28.720	27.819	29.940	27.801	28.780
Sharpe Ratio	0.661	0.526	0.740	1.101	0.930	0.960	1.044	0.913	1.228
After transaction costs									
Mean returns (%)	-0.101	-0.128	-0.096	-0.084	-0.089	-0.090	-0.073	-0.094	-0.063
Buy mean returns (%)	-0.017	-0.035	-0.019	-0.014	-0.026	-0.026	0.003	-0.020	-0.008
Sell mean returns (%)	-0.084	-0.093	-0.077	-0.070	-0.063	-0.064	-0.076	-0.074	-0.055
Minimum (%)	-16.182	-17.569	-16.146	-12.898	-19.614	-12.778	-15.344	-13.977	-14.332
Quartile 1 (%)	-1.040	-0.978	-1.036	-0.842	-0.932	-0.906	-0.936	-0.941	-0.920
Median (%)	-0.122	-0.125	-0.103	-0.093	-0.107	-0.085	-0.091	-0.093	-0.074
Quartile 3 (%)	0.808	0.742	0.837	0.678	0.754	0.717	0.780	0.776	0.784
Maximum (%)	20.865	14.538	20.785	14.350	13.937	14.185	14.014	13.310	12.277
Standard dev. (%)	2.052	1.802	1.995	1.659	1.809	1.752	1.886	1.751	1.813
Skewness	0.462	-0.071	0.359	-0.040	-0.057	0.202	0.027	0.121	0.048
Excess Kurtosis	8.725	4.279	6.732	5.986	6.623	4.469	5.343	3.371	3.370
Annualized									
Returns (%)	-26.344	-30.287	-25.186	-21.810	-23.201	-23.206	-20.440	-23.997	-17.956
Standard dev. (%)	32.580	28.599	31.671	26.343	28.720	27.819	29.940	27.801	28.780
Sharpe Ratio	-1.059	-0.795	-0.828	-0.808	-0.834	-0.683	-0.863	-0.624	

Note: The table details the financial performance of the k = 10 portfolio before and after transaction costs for all model variants from 2006 to 2023. Transaction costs of 0.05% are applied for both buying and selling. The statistical significance of mean returns is indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. These values are calculated using Newey-West standard errors with a one-lag correction.

and selling stocks for the long portfolio, and selling and buying for the short portfolio.

Before accounting for transaction costs, both short and long portfolios' returns are positive. However, only the long portfolios exhibit significant returns, while the short portfolios' returns are not significantly different from zero at any reasonable level. Models using $Returns_{0.5}$ exhibit slightly lower standard deviation compared to those using *Returns*. Ensemble models do not consistently reduce standard deviation or maximum daily loss.

All models display positive excess kurtosis, indicating leptokurtic distributions with heavier tails and more outliers than a normal distribution. Most models have positive skewness, except for three core models (DM2, DM4, and DM5) which exhibit slight negative skewness. Positive skewness, although small, suggests a distribution of returns skewed to the right.

Annualized returns are notably high, ranging from 15% to 35%, with an annualized standard deviation between 26% and 33%. We also calculate the Sharpe Ratio, assuming a risk-free rate of 0%. This assumption does not affect our conclusions, as we use the Sharpe Ratio to compare model performance and, in subsequent sections, to compare performance across different horizons. Thus, assuming a 0% risk-free rate for all horizons does not impact the interpretation of our results. We observe that $Returns_{0.5}$ have higher Sharpe Ratio compared to the *Returns* models. This indicates that these models not only deliver higher returns but also achieve this without incurring much additional volatility.

After accounting for transaction costs, none of the models yield positive returns. Consequently, this strategy does not generate any profit. This outcome aligns with the findings of our benchmark studies, which also reported unprofitable results post-2010.

Futures

Next, we analyze whether adding futures to our dataset can enhance the performance of our models. Given that we have approximately 400-500 stocks for each study period and we are adding only around 20 futures, we do not expect significant changes. However, it is still worthwhile to investigate any potential effects.

Table 5.4 displays the accuracy and mean daily returns of models that include both stocks and futures. Since we have futures data for the $Returns_{0.5}$ models starting in 2015, our analysis begins in 2018, allowing for a two-year

Table 5.4: Futures and stocks performance comparison

Returns type		*Returns*			$Returns_{0.5}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	$DM1$	$DM2$	$DM3$	$DM4$	$DM5$	$DM6$	$DM7$	$DM8$	$DM9$
Stocks and futures									
Mean returns	-0.044	0.062	0.019	0.106	0.071	0.085	0.052	0.106	0.083
Accuracy	50.906	51.216	51.397	51.906	51.253	51.943	51.202	51.296	51.424
Accuracy stocks	50.954	51.712	51.448	51.976	51.333	51.979	51.254	51.437	51.470
Accuracy futures	50.116	46.331	50.579	51.394	50.526	51.626	50.391	49.736	50.733
Futures share in dataset	4.323	4.323	4.323	4.241	4.241	4.241	4.241	4.241	4.241
Futures share in portfolio	5.791	9.223	5.818	12.074	9.923	10.148	6.020	8.286	6.205
Stocks only									
Mean returns	0.089	0.010	0.061	0.108	0.099	0.108	0.143	0.052	0.101
Accuracy	51.162	50.876	51.155	51.562	51.212	51.869	51.626	50.700	51.168

Note: The table displays the accuracy and mean daily returns of the k = 10 portfolio for all model variants that include both stocks and futures from 2018 to 2023. For comparison, the performance of the original models, which use only stocks over the same period, are also provided. Aditionally, the table includes the proportion of futures in the dataset and in the k = 10 portfolio to assess whether the models favor the inclusion of futures in the portfolio.

training period and a one-year validation period. For comparison, we also include the performance of the original models for the same period between 2018 and 2023.

We observe that the accuracy of models incorporating both futures and stocks is comparable to those using stocks only. However, the mean returns are almost always higher for the stocks-only models. Interestingly, looking at the futures and stocks models, the accuracy is consistently higher for stocks than for futures, the $Returns_{0.5}$ models exhibit higher accuracy for futures compared to the *Returns* models. Additionally, in the $Returns_{0.5}$ models, futures constitute about 10% of the selected portfolio, despite representing only around 4% of the entire dataset. This suggests that the fractionally differentiated series used in the $Returns_{0.5}$ models might be capturing patterns that the *Returns* models are unable to detect. Finally, based on these findings, we prefer stocks-only models over those extended by futures.

5.2 Hourly horizon

For the hourly horizon, we use a similar architecture to the daily horizon but with only one input feature, as realized volatility cannot be calculated for hourly data (see Chapter 4). The input length is 160. Due to computational and memory constraints, we randomly select 50% of data points for training

and validation while still using all stocks (or futures). For testing, we retain 100% of the data.

Since we cannot use realized volatility as an input feature and we can start using both futures and stocks in our dataset from 2012 we have 54 models. The total training time was approximately 40 hours without hyperparameter tuning.

Evaluation

For the hourly horizon, we evaluate only two core models: HM1, which has a single input feature *Returns*, and HM4, which also has a single input feature $Returns_{0.3}$. Given that we have only two models, we can perform only one ensemble, HM7. The numbering of these models corresponds to their respective daily models.

Accuracy and Financial performance

Table 5.5 presents the accuracy and mean hourly returns for all the study periods individually as well as for the whole period. We can see that on average over the entire period from 2006 to 2023 *Returns* (HM1) outperforms both the $Returns_{0.3}$ (HM4) and the Ensemble model (HM7) in terms of overall accuracy and mean hourly returns.

There are interesting findings when comparing the hourly horizon with the daily horizon. For the hourly horizon, the accuracy is consistently statistically significant, and this significance also applies to mean returns for most periods. One explanation for this is the larger dataset available for the hourly horizon, which inherently enhances the statistical significance. Despite this, the accuracy for the hourly horizon is still higher than that for the daily horizon, indicating an actual improvement beyond just dataset size. This improvement might be due to the presence of more market inefficiencies in the short horizon, as the market does not have sufficient time to correctly price all the information.

However, directly comparing mean returns between hourly and daily horizons is not feasible because hourly returns are naturally smaller due to the shorter time frame. Therefore, to provide a more meaningful comparison, we will evaluate annualized returns and other comparable metrics in the final section of this chapter.

Table 5.6 shows the financial performance. As expected, the HM1 model delivers the highest annualized returns, an impressive 131%. However, this

Table 5.5: Hourly accuracy and mean hourly returns

Returns type	*Returns*	$Returns_{0.3}$	*Ensemble*	*Returns*	$Returns_{0.3}$	*Ensemble*
# of Features	1	1	1	1	1	1
Model	$HM1$	$HM4$	$HM7$	$HM1$	$HM4$	$HM7$
	Accuracy			Mean returns		
2006	53.052***	52.296***	52.921***	0.064***	0.049***	0.065***
2007	53.244***	53.621***	53.507***	0.074***	0.069***	0.08***
2008	52.193***	51.445***	51.816***	-0.003	0.083***	0.06**
2009	53.509***	52.323***	53.058***	0.083***	-0.001	0.05**
2010	53.956***	52.12***	51.283***	0.106***	0.045***	0.063***
2011	53.014***	52.818***	52.506***	0.082***	0.068***	0.079***
2012	52.666***	52.38***	52.586***	0.07***	0.061***	0.077***
2013	53.16***	52.577***	52.856***	0.054***	0.059***	0.057***
2014	52.81***	52.639***	52.739***	0.052***	0.06***	0.057***
2015	52.062***	51.903***	51.645***	0.045***	0.042***	0.049***
2016	52.179***	51.382***	51.416***	0.012	0.007	-0.009
2017	53.481***	51.548***	52.295***	0.057***	0.012*	0.04***
2018	51.538***	51.909***	51.438***	0.018**	0.037***	0.028***
2019	51.713***	50.202	50.673***	0.01	0.011	0.03**
2020	52.456***	52.114***	51.99***	0.072***	0.078***	0.07***
2021	52.166***	51.216***	51.134***	0.059***	0.004	0.028**
2022	51.553***	51.199***	51.034***	0.037**	0.017	0.039***
2023	50.671***	50.598**	49.807	0.021*	0.002	0.009
Whole period	52.534***	51.912***	51.939***	0.051***	0.039***	0.049***

Note: The table shows the hourly accuracy and mean hourly returns (%) of the k = 10 portfolio for each study period and for all model variants based on the number and type of input features. Statistical significance is indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. p-values for accuracy are based on the binomial test against a 50% random chance, while p-values for mean returns are calculated using Newey-West standard errors with a one-lag correction. The year in the first column denotes the study period.

Table 5.6: Financial performance hourly horizon

Returns type	*Returns*	$Returns_{0.3}$	*Ensemble*
# of Features	1	1	1
Model	$HM1$	$HM4$	$HM7$
Before transaction costs			
Mean returns (%)	0.051***	0.039***	0.049***
Buy mean returns (%)	0.037***	0.030***	0.032***
Sell mean returns (%)	0.014***	0.010**	0.017***
Minimum (%)	-16.528	-11.793	-12.166
Quartile 1 (%)	-0.225	-0.207	-0.212
Median (%)	0.048	0.037	0.048
Quartile 3 (%)	0.332	0.287	0.313
Maximum (%)	9.974	9.248	12.873
Standard dev. (%)	0.763	0.660	0.703
Skewness	-1.192	-0.269	-0.622
Excess Kurtosis	31.199	24.365	27.071
Annualized			
Returns (%)	130.966	91.034	124.301
Standard dev. (%)	32.033	27.726	29.526
Sharpe Ratio	4.088	3.283	4.210
After transaction costs			
Mean returns (%)	-0.149	-0.161	-0.151
Buy mean returns (%)	-0.063	-0.070	-0.068
Sell mean returns (%)	-0.086	-0.090	-0.083
Minimum (%)	-16.728	-11.993	-12.366
Quartile 1 (%)	-0.425	-0.407	-0.412
Median (%)	-0.152	-0.163	-0.152
Quartile 3 (%)	0.132	0.087	0.113
Maximum (%)	9.774	9.048	12.673
Standard dev. (%)	0.763	0.660	0.703
Skewness	-1.192	-0.269	-0.622
Excess Kurtosis	31.199	24.365	27.071
Annualized			
Returns (%)	-93.016	-94.226	-93.218
Standard dev. (%)	32.033	27.726	29.526
Sharpe Ratio	-2.904	-3.398	-3.157

Note: The table details the financial performance of the k = 10 portfolio before and after transaction costs for all model variants from 2006 to 2023. Transaction costs of 0.05% are applied for both buying and selling. The statistical significance of mean returns is indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. These values are calculated using Newey-West standard errors with a one-lag correction.

Table 5.7: Futures and stocks performance comparison for hourly horizon

Returns type	*Returns*	$Returns_{0.3}$	*Ensemble*
# of Features	1	1	1
Model	$HM1$	$HM4$	$HM7$
Stocks and futures			
Mean returns	0.025	0.021	0.031
Accuracy	51.698	51.366	51.320
Accuracy stocks	51.769	51.423	51.338
Accuracy futures	50.179	50.977	51.014
Futures share in dataset	4.308	4.100	4.100
Futures share in portfolio	4.450	12.667	5.588
Stocks only			
Mean returns	0.037	0.023	0.032
Accuracy	51.995	51.350	51.287

Note: The table displays the accuracy and mean hourly returns of the k = 10 portfolio for all model variants that include both stocks and futures from 2015 to 2023. For comparison, the performance of the original models, which use only stocks over the same period, are also provided. Aditionally, the table includes the proportion of futures in the dataset and in the k = 10 portfolio to assess whether the models favor the inclusion of futures in the portfolio.

comes at the cost of the highest standard deviation, which is 32%. Despite this high volatility, HM1 still achieves the highest Sharpe ratio, approximately 4.1, indicating a strong risk-adjusted return. The ensemble model (HM7) also shows promising results with a Sharpe ratio of 4.2 the highest of all 3 models. Due to high transaction costs no model delivered positive returns.

Futures

Finally, Table 5.7 shows the impact of adding futures to our dataset from 2015 onwards (due to data availability). For comparability, we also include the performance of the original models over the same period. We observe that stocks-only models consistently exhibit higher mean returns and higher or very similar accuracy compared to models that use both stocks and futures. This trend is consistent with the findings observed for the daily horizon.

Notably, the fractionally differentiated series, $Returns_{0.3}$, selects around 12.7% of futures in its portfolio, significantly higher than the 4% that would be expected by random selection. This strengthens the argument that these series can capture patterns in futures data that standard logarithmic returns cannot. However, based on the results, we once again prefer models using stocks only over models using both stocks and futures.

Table 5.8: Weekly accuracy

Returns type		*Returns*			$Returns_{0.65}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	$WM1$	$WM2$	$WM3$	$WM4$	$WM5$	$WM6$	$WM7$	$WM8$	$WM9$
2007	50.577	51.349	51.827						
2008	50.872	47.406	50.581						
2009	50.769	50.887	51.154						
2010	49.423	47.839	50.769						
2011	54.135***	53.173**	55.096***						
2012	50.755	49.811	51.415						
2013	49.231	49.038	48.846	50.0	50.288	50.577	48.269	48.173	50.481
2014	54.135***	49.605	53.654***	50.481	48.365	51.442	52.212*	48.462	53.558**
2015	55.192***	50.355	55.481***	49.423	51.877	49.904	52.692**	50.673	51.731
2016	48.269	48.173	47.788	52.019	52.309*	53.846***	47.5	49.038	46.827
2017	51.509	51.038	52.453*	53.585**	50.094	53.302**	52.925**	53.962***	54.811***
2018	50.865	51.737	51.538	52.885**	51.25	51.442	51.538	53.173**	52.885**
2019	52.788**	47.343	50.962	48.362	46.408	47.981	50.0	47.308	49.038
2020	47.093	48.615	47.977	51.731	49.18	52.212*	50.673	49.423	50.385
2021	51.923	52.019	50.962	51.154	49.135	51.346	51.25	49.808	49.808
2022	50.769	50.865	50.0	55.096***	49.808	53.846***	51.058	50.481	52.788**
2023	49.375	49.236	49.792	51.042	48.125	49.062	50.833	47.708	48.75
2013-2023	51.029**	49.846	50.87**	51.441***	49.718	51.38***	50.817**	49.859	51.028**

Note: The table shows the weekly accuracy in percentage (%) of the k = 10 portfolio for each study period and for all model variants based on the number and type of input features. A binomial test was used to compare the accuracy against a 50% random chance, with statistical significance indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. The year in the first column denotes the study period.

5.3 Weekly horizon

For the weekly horizon, we follow the same procedure as with the daily horizon. In this case, we set the input length to 100. We decrease the batch size to 512, LSTM units to 20, and the fully connected layer to 4 units. Finally, we increase the training period to 3 years, while other hyperparameters remain the same.

We use $Returns_{0.65}$ as a fractionally differentiated series input feature. However, due to the data availability of futures, $Returns_{0.8}$ is used for comparing models using both stocks and futures with stocks-only models. In total, we trained 74 models which took around 6 hours on a GPU.

Evaluation

For this horizon, we have the same four core models and the corresponding five ensemble models as we do for the daily horizon. The naming convention is also the same, with the prefix W replacing D, leading to the models being labeled as WM1 through WM9. Models using $Returns_{0.8}$ instead of $Returns_{0.65}$ are denoted as WM4*-WM9*.

Table 5.9: Mean weekly returns

Returns type		*Returns*			$Returns_{0.65}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	$WM1$	$WM2$	$WM3$	$WM4$	$WM5$	$WM6$	$WM7$	$WM8$	$WM9$
2007	-0.052	-0.057	0.012						
2008	-0.102	-0.206	-0.321						
2009	-1.493	-0.195	-1.174						
2010	-0.232	-0.477	-0.172						
2011	0.559**	0.431*	0.647**						
2012	0.097	0.156	0.45**						
2013	-0.111	-0.124	-0.077	0.126	-0.196	-0.12	0.01	-0.506	-0.165
2014	0.615**	0.125	0.742**	0.108	-0.12	0.229	0.361*	0.115	0.444*
2015	0.563*	0.146	0.591**	-0.043	0.323*	0.096	0.181	0.346	0.384
2016	-1.257	-0.135	-1.205	0.759*	0.36*	0.53**	-0.679	0.437**	-0.648
2017	0.321	-0.165	-0.164	0.567*	0.223	0.614*	0.314	0.006	0.015
2018	0.148	0.117	0.084	0.503	0.316	0.189	0.272	0.546**	0.378
2019	0.117	-0.214	-0.026	-0.66	-0.338	-0.704	-0.322	-0.266	-0.252
2020	-0.717	-0.282	-0.688	0.323	-0.136	0.216	-0.306	-0.191	-0.736
2021	0.072	0.166	-0.026	0.14	-0.459	-0.013	0.108	-0.09	0.007
2022	0.283	0.176	0.267	0.917**	-0.075	0.743**	0.512	-0.267	-0.177
2023	0.529	0.172	0.544*	0.108	-0.127	-0.116	0.033	-0.09	0.032
2013-2023	0.048	-0.003	-0.0	0.26*	-0.02	0.154	0.045	0.004	-0.066

Note: The table shows the mean weekly returns in percentage (%) of the k = 10 portfolio for each study period and for all model variants based on the number and type of input features. Statistical significance is indicated as follows: *p $<$ 0.1; **p $<$ 0.05; ***p $<$ 0.01. These values are calculated using Newey-West standard errors with a one-lag correction. The year in the first column denotes the study period.

Accuracy and Financial performance

Table 5.8 displays the accuracy across all study periods based on data availability, as well as the overall accuracy from 2013 to 2023, the period for which data is available for all models. We observe the accuracy decreasing in comparison with previously discussed horizons and given the much smaller dataset for the weekly horizon, the accuracy for most years is not statistically better than a random guess. Notably, models using only one input feature tend to outperform other configurations.

The mean weekly returns, as illustrated in Table 5.9, further emphasize the disappointing outcomes. Among the models, only the WM4 model delivers statistically significant positive returns of 0.26% over the 2013-2023 period. Additionally, four out of the nine models report negative returns.

Table 5.10 shows that annualized returns are positive only for the WM4 and WM6 models before accounting for transaction costs. However, with an annualized standard deviation of approximately 30%, we cannot consider this as a good performance. This poor performance can be attributed to the impact of short selling. When analyzing mean weekly returns exclusively from a buying

Table 5.10: Financial performance weekly horizon

Returns type		*Returns*			$Returns_{0.65}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	$WM1$	$WM2$	$WM3$	$WM4$	$WM5$	$WM6$	$WM7$	$WM8$	$WM9$
Before transaction costs									
Mean returns (%)	0.048	-0.003	-0.000	0.260*	-0.020	0.154	0.045	0.004	-0.066
Buy mean returns (%)	0.300***	0.223**	0.244**	0.471***	0.279***	0.420***	0.357***	0.237**	0.320***
Sell mean returns (%)	-0.252	-0.226	-0.244	-0.210	-0.298	-0.266	-0.313	-0.232	-0.385
Minimum (%)	-21.696	-11.210	-21.902	-40.025	-9.349	-40.025	-35.535	-10.969	-36.839
Quartile 1 (%)	-1.802	-1.014	-1.770	-1.442	-1.167	-1.266	-1.689	-1.251	-1.684
Median (%)	0.341	-0.013	0.356	0.232	-0.091	0.150	0.287	-0.034	0.190
Quartile 3 (%)	2.123	1.080	2.056	2.006	0.968	1.718	2.162	1.294	1.887
Maximum (%)	16.385	11.645	15.677	29.169	12.455	27.662	23.634	7.609	24.668
Standard dev. (%)	3.866	2.101	3.910	4.347	2.098	4.139	4.482	2.275	4.403
Skewness	-0.598	-0.113	-0.479	-0.828	0.398	-1.028	-1.373	-0.201	-1.498
Excess Kurtosis	0.597	1.152	0.837	17.694	0.656	21.165	11.104	-1.121	14.334
Annualized									
Returns (%)	-1.426	-1.294	-3.951	8.713	-2.126	3.354	-3.148	-1.114	-8.388
Standard dev. (%)	27.877	15.149	28.196	31.349	15.126	29.848	32.323	16.403	31.749
Sharpe Ratio	-0.051	-0.085	-0.140	0.278	-0.141	0.112	-0.097	-0.068	-0.264
After transaction costs									
Mean returns (%)	-0.152	-0.203	-0.200	0.060	-0.220	-0.046	-0.155	-0.196	-0.266
Buy mean returns (%)	0.200**	0.123	0.144*	0.371***	0.179**	0.320***	0.257**	0.137*	0.220**
Sell mean returns (%)	-0.352	-0.326	-0.344	-0.310	-0.398	-0.366	-0.413	-0.332	-0.485
Minimum (%)	-21.896	-11.410	-22.102	-40.225	-9.549	-40.225	-35.735	-11.169	-37.039
Quartile 1 (%)	-2.002	-1.214	-1.970	-1.642	-1.367	-1.466	-1.889	-1.451	-1.884
Median (%)	0.141	-0.213	0.156	0.032	-0.291	-0.050	0.087	-0.234	-0.010
Quartile 3 (%)	1.923	0.880	1.856	1.806	0.768	1.518	1.962	1.094	1.687
Maximum (%)	16.185	11.445	15.477	28.969	12.255	27.462	23.434	7.409	24.468
Standard dev. (%)	3.866	2.101	3.910	4.347	2.098	4.139	4.482	2.275	4.403
Skewness	-0.598	-0.113	-0.479	-0.828	0.398	-1.028	-1.373	-0.201	-1.498
Excess Kurtosis	0.597	1.152	0.837	17.694	0.656	21.165	11.104	-1.121	14.334
Annualized									
Returns (%)	-11.133	-11.008	-13.414	-1.977	-11.761	-6.817	-12.692	-10.846	-17.425
Standard dev. (%)	27.877	15.149	28.196	31.349	15.126	29.848	32.323	16.403	31.749
Sharpe Ratio	-0.399	-0.727	-0.476	-0.063	-0.778	-0.228	-0.393	-0.661	-0.549

Note: The table details the financial performance of the k = 10 portfolio before and after transaction costs for all model variants from 2013 to 2023. Transaction costs of 0.05% are applied for both buying and selling. The statistical significance of mean returns is indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. These values are calculated using Newey-West standard errors with a one-lag correction.

Table 5.11: Futures and stocks performance comparison for weekly horizon

Returns type		*Returns*			$Returns_{0.8}$			*Ensemble*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*	1	2	*Ensemble*
Model	$WM1$	$WM2$	$WM3$	$WM4*$	$WM5*$	$WM6*$	$WM7*$	$WM8*$	$WM9*$
Stocks and futures									
Mean returns	0.063	-0.206	-0.213	0.106	0.172	0.122	0.056	-0.096	-0.182
Accuracy	50.230	47.462	47.961	51.414	51.118	51.250	50.296	49.112	48.783
Accuracy stocks	50.137	48.335	48.631	51.000	50.729	51.300	49.816	49.487	49.162
Accuracy futures	52.381	37.603	38.021	53.579	53.973	50.904	54.348	44.186	42.614
Futures share in dataset	4.363	4.363	4.363	4.640	4.640	4.640	4.640	4.640	4.640
Futures share in portfolio	4.145	8.134	6.316	16.086	12.007	12.730	10.592	7.072	5.789
Stocks only									
Mean returns	0.288	0.171	0.254	0.358	-0.225	0.038	0.477	-0.027	0.220
Accuracy	50.724	50.955	50.263	50.296	50.066	49.507	51.480	50.329	50.033

Note: The table displays the accuracy and mean weekly returns of the k = 10 portfolio for all model variants that include both stocks and futures from 2021 to 2023. For comparison, the performance of the original models, which use only stocks over the same period, are also provided. Aditionally, the table includes the proportion of futures in the dataset and in the k = 10 portfolio to assess whether the models favor the inclusion of futures in the portfolio.

perspective, the returns are statistically positive. This could be because stocks, on average, tend to grow more frequently than they decline, making short selling less effective overall. Due to the lower turnover, the buy-only side of the portfolio results in positive and even significant mean weekly returns, even after accounting for transaction costs. Based on these results, we would prefer the WM4 model over other models.

These results are not very surprising, the results of Fischer & Krauss (2018) also show significantly poorer performance of weekly models over daily models. Another similarity of our study with that of Fischer & Krauss (2018) is the much better performance of the buy-only side. However, our results are still worse than that of Fischer & Krauss (2018). One possible explanation could be that since 2010, markets may have become more efficient in relation to machine learning methods as the authors argue and in the original study the authors analyzed a longer period prior to 2010, which is not covered in our study.

Futures

Table 5.11 compares the performance of models that use both futures and stocks against models that use only stocks. Because the analysis is based on a limited timeframe of just three years, from 2021 to 2023, the results should be interpreted with caution. Nonetheless, similar trends to those observed in previously described time horizons can be identified. Notably, models using

fractionally differentiated series show a higher tendency to include futures in their portfolios compared to other models. This strengthens our finding that these series can detect patterns that traditional logarithmic returns cannot.

Additionally, the accuracy of stocks-only models is higher for *Returns* models, while for $Returns_{0.8}$, the accuracy is higher for models incorporating both stocks and futures. Based on these results, we would likely prefer models extended with futures. However, due to the limited three-year analysis period, we cannot definitively conclude that futures provide a significant benefit.

5.4 Monthly horizon

Finally, we train monthly horizon. We decrease LSTM units to 16. The fully connected layer to 4 units and the batch size to 512. Other hyperparamters are the same as for the weekly horizon. For the monthly horizon, we do not have a fractionally differentiated series since it would significantly reduce our training sample as we already explained in Chapter 4. The input length is 60 for realized volatility and logarithmic returns. In total, we trained 32 models which took around 1 hour.

Evaluation

For the monthly horizon, we once again evaluate only two core models: MM1, which has a single input feature, *Returns*, and MM2, which includes RV as an additional input feature. This allows us to perform only one ensemble, MM3. Again, the numbering of these models corresponds to their respective daily models.

Accuracy and Financial performance

Table 5.12 exhibits both monthly accuracy as well as mean monthly returns. Notably, only the ensemble model (MM3) achieved an accuracy significantly higher than the 50% random threshold throughout the entire period. This high average accuracy is primarily due to a few exceptional years, such as 2014, which had an accuracy exceeding 58%, and 2015 and 2017, both with accuracies over 57%. These outlier years contributed to the overall high average accuracy from 2011 to 2023. While this performance is comparable to previously analyzed horizons, caution is warranted because the high accuracy holds

Table 5.12: Monthly accuracy and mean monthly returns

Returns type		*Returns*			*Returns*	
# of Features	1	2	*Ensemble*	1	2	*Ensemble*
Model	$MM1$	$MM2$	$MM3$	$MM1$	$MM2$	$MM3$
	Accuracy			Mean returns		
2011	47.917	50.22	48.333	-1.773	0.512	-1.605
2012	49.167	53.333	48.333	-0.053	0.576	-0.591
2013	49.167	42.917	49.167	-1.131	-1.203	-1.104
2014	49.583	57.083**	58.75***	0.028	-0.699	0.28
2015	53.333	52.083	57.5**	0.791	1.434	1.932
2016	45.833	47.917	46.25	-1.592	-2.174	-3.07
2017	54.167	52.917	57.5**	2.139**	1.866**	2.67**
2018	47.083	55.833**	52.083	-0.794	0.468	0.353
2019	50.417	41.25	45.833	0.427	-3.571	-2.072
2020	55.0*	44.828	51.667	0.458	-2.751	-1.08
2021	52.917	49.583	52.083	1.365	-0.08	1.379
2022	44.167	53.333	51.667	-4.516	-0.995	-1.371
2023	53.636	49.545	50.455	-0.435	0.388	-1.203
Whole period	50.161	50.081	51.516**	-0.391	-0.485	-0.417

Note: The table shows the monthly accuracy and mean monthly returns (%) of the k = 10 portfolio for each study period and for all model variants based on the number and type of input features. Statistical significance is indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. p-values for accuracy are based on the binomial test against a 50% random chance, while p-values for mean returns are calculated using Newey-West standard errors with a one-lag correction. The year in the first column denotes the study period.

true only for this one model, and the small dataset might mean the model's success is due to chance.

While the other models also exceeded the 50% accuracy mark, their performance was not statistically significant at any reasonable level. This lack of significance can be attributed to both their lower accuracy and the smaller dataset used for these monthly horizons compared to previously discussed horizons.

The mean monthly returns are also relatively weak, with only the year 2017 showing positive and significant returns. Interestingly, all models delivered positive and significant results in that year. However, when looking at the average returns over the entire period, none of the models generated positive returns.

Table 5.13 provides a more detailed breakdown of the financial performance of our models. Similar to the weekly horizon results, we observe that the buy-only side of the portfolio produced positive returns. For MM2, these returns were even statistically significant. Another notable observation is that the standard deviation of the portfolio with monthly turnover is much lower, around 21% annualized, compared to other horizons. This is expected as fewer buy and sell orders are executed on a monthly basis, reducing overall volatility.

Once again, these results are not surprising. As previously explained, markets have become more efficient in regard to machine learning models after 2010 (Fischer & Krauss 2018). Additionally, on a monthly horizon, we expect fewer inefficiencies for the models to uncover, as these inefficiencies are likely to be priced in due to the longer reaction period of market participants.

Futures

Table 5.14 compares the performance of models using both stocks and futures as input data to models using stocks only. Due to data availability, we can only analyze the period between 2021 and 2023. Therefore, these results should be interpreted with caution due to the short test span. We observe that incorporating futures into the dataset significantly reduces accuracy. None of the models achieved an accuracy higher than 50% with the combined dataset. This contrasts with the stocks-only models, which all achieved accuracies above 50% over the same period. Specifically, the accuracy of predicting futures movements alone for the core models is around 44-46%. The ensemble model, however, shows a slightly better performance with an accuracy of approxi-

Table 5.13: Financial performance monthly horizon

Returns type	*Returns*		
# of Features	1	2	*Ensemble*
Model	$MM1$	$MM2$	$MM3$
Before transaction costs			
Mean returns (%)	-0.391	-0.485	-0.417
Buy mean returns (%)	0.295	0.882**	0.25
Sell mean returns (%)	-0.687	-1.367	-0.667
Minimum (%)	-22.173	-21.564	-22.124
Quartile 1 (%)	-3.553	-3.502	-3.478
Median (%)	0.372	-0.078	0.370
Quartile 3 (%)	3.034	3.060	2.875
Maximum (%)	12.641	13.121	13.767
Standard dev. (%)	6.120	5.676	6.078
Skewness	-0.805	-0.660	-0.747
Excess Kurtosis	-1.377	-1.534	-1.408
Annualized			
Returns (%)	-6.762	-7.490	-7.012
Standard dev. (%)	21.202	19.664	21.055
Sharpe Ratio	-0.319	-0.381	-0.333
After transaction costs			
Mean returns (%)	-0.591	-0.685	-0.617
Buy mean returns (%)	0.195	0.782**	0.150
Sell mean returns (%)	-0.787	-1.467	-0.767
Minimum (%)	-22.373	-21.764	-22.324
Quartile 1 (%)	-3.753	-3.702	-3.678
Median (%)	0.172	-0.278	0.170
Quartile 3 (%)	2.834	2.860	2.675
Maximum (%)	12.441	12.921	13.567
Standard dev. (%)	6.120	5.676	6.078
Skewness	-0.805	-0.660	-0.747
Excess Kurtosis	-1.377	-1.534	-1.408
Annualized			
Returns (%)	-8.979	-9.690	-9.223
Standard dev. (%)	21.202	19.664	21.055
Sharpe Ratio	-0.423	-0.493	-0.438

Note: The table details the financial performance of the k = 10 portfolio before and after transaction costs for all model variants from 2011 to 2023. Transaction costs of 0.05% are applied for both buying and selling. The statistical significance of mean returns is indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. These values are calculated using Newey-West standard errors with a one-lag correction.

Table 5.14: Futures and stocks performance comparison for monthly horizon

Returns type	*Returns*		
# of Features	1	2	*Ensemble*
Model	$MM1$	$MM2$	$MM3$
Stocks and futures			
Mean returns	-1.405	0.092	-0.766
Accuracy	48.143	48.143	49.286
Accuracy stocks	48.826	48.952	48.944
Accuracy futures	44.231	45.714	50.758
Futures share in dataset	4.989	4.989	4.989
Futures share in portfolio	14.857	25.000	18.857
Stocks only			
Mean returns	-1.217	-0.247	-0.375
Accuracy	50.143	50.857	51.429

Note: The table displays the accuracy and mean monthly returns of the k = 10 portfolio for all model variants that include both stocks and futures from 2021 to 2023. For comparison, the performance of the original models, which use only stocks over the same period, are also provided. Aditionally, the table includes the proportion of futures in the dataset and in the k = 10 portfolio to assess whether the models favor the inclusion of futures in the portfolio.

mately 50.8% for the futures. Despite the poor accuracy in predicting futures movements, all models showed a preference for selecting futures over stocks in the portfolio. This is interesting because, for previous horizons, we observed this tendency only for the fractionally differentiated series. However, for the monthly data analyzed here, fractionally differentiated series were not available due to data constraints. Given the very short test period and the small data sample, we should be cautious in drawing strong conclusions from these results.

5.5 Comparing horizons

To conclude this chapter, we evaluate the performance of our models across various prediction horizons. Our analysis indicates that incorporating futures into our dataset does not result in significant performance improvements. Therefore, we focus our evaluation on models using stock data exclusively. For each horizon, we select the model with the highest accuracy. Specifically, we choose DM6 for the daily horizon, HM1 for the hourly horizon, WM4 for the weekly horizon, and MM3 for the monthly horizon. This selection indicates that there is no single model specification that consistently performs best across all horizons.

Additionally, we consider ensemble models that combine all core models. For the different prediction horizons, the selected ensemble models are DM9 for

the daily horizon, HM7 for the hourly horizon, WM9 for the weekly horizon, and MM3 for the monthly horizon. These ensemble models are chosen because they aggregate the predictions of all core models, potentially leading to enhanced robustness and improved generalization.

Table 5.15 presents the annual accuracies for all horizons. On average, the hourly horizon model predicts future price movements with the highest accuracy, followed by the monthly, daily, and weekly horizons. Additionally, we observe that the accuracy for all horizons is significantly higher than 50% over the entire period from 2013 to 2023. This pattern holds true for both the models selected by the highest accuracy and the ensemble models.

The high accuracy observed for the monthly horizon is quite surprising. But as we explained earlier this appears to be driven by a few exceptionally high-performing years. For instance, the accuracy for 2014 was as high as 58.75%, and both 2015 and 2017 had accuracies of 57.5%. Given the relatively small testing sample due to the monthly nature of the data, it is possible that the model's high performance in these years may be due to chance, leading to an overall high accuracy.

When comparing the annualized returns before transaction costs and the annualized standard deviation in Table 5.16, we observe that the highest returns are achieved with an hourly horizon, followed by daily, weekly, and monthly horizons. Notably, the monthly horizon fails to produce positive returns. The standard deviation is approximately 30% for the hourly, daily, and weekly horizons, whereas it decreases to 22% for the monthly horizon. For the weekly horizon, much of the high standard deviation can be attributed to the year 2020, when the standard deviation was 76%, excluding this year, the standard deviation would also be lower. Consequently, the risk-return ratio is most favorable for the hourly horizon. However, as discussed in previous sections, no model yielded positive returns after accounting for transaction costs. Therefore, a trading strategy based solely on a simple k portfolio is impractical when transaction costs are 0.05%. This is unfortunate but as we already explained it is in check with our benchmark.

Table 5.15: Accuracy

	Highest accuracy models				Ensemble models			
Horizon	*Hourly*	*Daily*	*Weekly*	*Monthly*	*Hourly*	*Daily*	*Weekly*	*Monthly*
Returns type	*Returns*	$Returns_{0.5}$	$Returns_{0.65}$	*Returns*	*Ensemble*	*Ensemble*	*Ensemble*	*Returns*
# of Features	1	*Ensemble*	1	*Ensemble*	1	*Ensemble*	*Ensemble*	*Ensemble*
Model	$HM1$	$DM6$	$WM4$	$MM3$	$HM7$	$DM9$	$WM9$	$MM3$
2006	53.052***	52.829***			52.921***	54.522***		
2007	53.244***	52.41***			53.507***	51.673***		
2008	52.193***	52.569***			51.816***	53.498***		
2009	53.509***	51.349**			53.058***	51.19**		
2010	53.956***	51.746***			51.283***	50.238		
2011	53.014***	51.925***		48.333	52.506***	51.726***		48.333
2012	52.666***	52.28***		48.333	52.586***	51.44**		48.333
2013	53.16***	52.004***	50.0	49.167	52.856***	51.885***	50.481	49.167
2014	52.81***	50.774	50.481	58.75***	52.739***	51.27**	53.558**	58.75***
2015	52.062***	53.016***	49.423	57.5**	51.645***	52.738***	51.731	57.5**
2016	52.179***	50.317	52.019	46.25	51.416***	50.952*	46.827	46.25
2017	53.481***	51.076*	53.585**	57.5**	52.295***	50.837	54.811***	57.5**
2018	51.538***	51.454**	52.885**	52.083	51.438***	49.442	52.885**	52.083
2019	51.713***	51.925***	48.362	45.833	50.673***	51.369**	49.038	45.833
2020	52.456***	52.391***	51.731	51.667	51.99***	52.609***	50.385	51.667
2021	52.166***	51.885***	51.154	52.083	51.134***	51.31**	49.808	52.083
2022	51.553***	50.916*	55.096***	51.667	51.034***	49.88	52.788**	51.667
2023	50.671***	52.721***	51.042	50.455	49.807	52.522***	48.75	50.455
2013-2023	52.177***	51.671***	51.441***	52.099***	51.564***	51.337***	51.028**	52.099***

Note: The table shows the accuracy in percentage (%) of the k = 10 portfolio for each study period and for selected models based on either the highest accuracy or the ensemble for each horizon. A binomial test was used to compare the accuracy against a 50% random chance, with statistical significance indicated as follows: *p < 0.1; **p < 0.05; ***p < 0.01. The year in the first column denotes the study period.

Table 5.16: Annualized returns before transaction costs (annualized stadard devation)

	Highest accuracy models				Ensemble models			
Horizon	*Hourly*	*Daily*	*Weekly*	*Monthly*	*Hourly*	*Daily*	*Weekly*	*Monthly*
Returns type	*Returns*	$Returns_{0.5}$	$Returns_{0.65}$	*Returns*	*Ensemble*	*Ensemble*	*Ensemble*	*Returns*
# of Features	1	*Ensemble*	1	*Ensemble*	*Ensemble*	*Ensemble*	*Ensemble*	*Ensemble*
Model	$HM1$	$DM6$	$WM4$	$MM3$	$HM7$	$DM9$	$WM9$	$MM3$
2006	201 (21)	30 (18)			209 (19)	85 (17)		
2007	257 (22)	32 (20)			294 (22)	22 (22)		
2008	-24 (64)	68 (42)			145 (57)	153 (53)		
2009	278 (52)	-14 (50)			113 (49)	-5 (48)		
2010	518 (30)	29 (16)			198 (19)	30 (17)		
2011	315 (24)	42 (21)		-18 (10)	293 (21)	70 (20)		-18 (10)
2012	234 (20)	12 (19)		-9 (24)	278 (20)	14 (19)		-9 (24)
2013	152 (18)	30 (16)	6 (15)	-13 (13)	169 (18)	29 (17)	-9 (12)	-13 (13)
2014	143 (21)	32 (14)	5 (14)	1 (23)	166 (19)	26 (15)	25 (14)	1 (23)
2015	112 (28)	120 (28)	-3 (14)	22 (25)	129 (28)	117 (22)	20 (20)	22 (25)
2016	15 (37)	-21 (31)	44 (26)	-34 (28)	-18 (32)	27 (34)	-34 (36)	-34 (28)
2017	165 (17)	20 (18)	32 (19)	36 (16)	101 (16)	10 (19)	-1 (17)	36 (16)
2018	35 (19)	8 (21)	26 (23)	4 (12)	60 (19)	-20 (22)	20 (18)	4 (12)
2019	16 (21)	25 (28)	-32 (27)	-23 (12)	63 (25)	67 (29)	-16 (27)	-23 (12)
2020	214 (50)	61 (39)	-13 (76)	-14 (22)	209 (46)	83 (38)	-50 (74)	-14 (22)
2021	171 (26)	34 (31)	2 (32)	15 (21)	58 (27)	0 (32)	-5 (32)	15 (21)
2022	83 (29)	-10 (29)	56 (25)	-19 (28)	92 (29)	-18 (31)	-12 (26)	-19 (28)
2023	35 (26)	47 (28)	3 (22)	-16 (29)	13 (21)	71 (26)	-1 (22)	-16 (29)
2013-2023	92 (29)	27 (27)	9 (31)	-6 (22)	82 (28)	29 (27)	-8 (32)	-6 (22)

Note: The table shows the annualized returns along with annualized standard deviation (in parentheses), both expressed as percentages (%), for the k = 10 portfolio across each study period. The models selected are based on either the highest accuracy or the ensemble for each horizon. The first column of the table indicates the corresponding study period.

Chapter 6

Conclusion

In this thesis, we investigate the predictability of financial returns across four distinct time horizons: hourly, daily, weekly, and monthly. Specifically, we analyze whether deep learning models utilizing LSTM units can accurately predict the future movements of stocks and futures. While most existing literature on financial return prediction focuses on the daily horizon, there are studies examining both intraday and longer time horizons. However, these studies often employ different methodologies and datasets, making direct comparisons challenging. Our research addresses this gap by using a consistent dataset and methodology, enabling direct comparisons of model performance across various time horizons.

Additionally, we enhance the existing literature by incorporating fractionally differentiated series, as described by Prado (2018). Traditional logarithmic or simple returns often remove excessive memory from the series. Fractionally differentiated series mitigate this issue by using fractional differences instead of first differences, retaining more memory while ensuring stationarity. For all time horizons except the monthly data, we calculated such series that ensure stationarity with a differentiation factor d less than 1, preserving some memory in the process.

Our third contribution is examining whether the inclusion of futures data improves the performance of our models when combined with stock data. Although we did not observe a performance improvement from this inclusion, we discovered other interesting findings.

Following the methodology of Krauss *et al.* (2017), we construct k portfolios by purchasing k stocks (or futures) with the highest probability of outperforming the median returns of the next horizon and short-selling k stocks (or futures)

with the lowest probability.

Our analysis of these portfolios indicates that the prediction accuracy consistently exceeds the 50% threshold, which would be expected if the models were making random predictions. This observation holds particularly true for hourly and daily horizons across most years. For weekly and monthly horizons, the significance of the results also suggests a meaningful predictive capability, although it may vary in degree.

Two possible explanations account for the lower accuracy performance on longer horizons. First, the smaller test sample for these horizons decreases the significance. Second, over longer horizons, market inefficiencies are likely to be fewer, as they tend to be priced in due to the longer reaction period of market participants.

Examining the returns before accounting for transaction costs, the highest annual returns are achieved on the hourly horizon, followed by the daily horizon. However, the weekly and monthly horizons perform significantly worse.

Further inspection shows that the poor performance of weekly and monthly horizons is largely due to the underperformance of short selling. This underperformance can be attributed to the fact that our models are trained to predict whether the next horizon's returns will be above or below the median returns, which provides a balanced target variable. However, given the general long-term growth trend of stocks, correctly predicting the next horizon movement might still lead to short-selling stocks that ultimately grow. This effect is less pronounced on shorter horizons, where stock price movements are more volatile and less influenced by long-term trends.

The accuracy of models using fractionally differentiated series is slightly better than those using standard logarithmic returns for the daily and weekly horizons. However, for the hourly horizon, models using standard logarithmic returns perform better. Therefore, we are cautious about drawing strong conclusions regarding the superiority of fractionally differentiated series over standard logarithmic returns in improving model performance.

A more interesting finding is the trend of models using fractionally differentiated series to select futures for inclusion in the k portfolio at a higher rate than would be expected if the selection were uniform. This contrasts with the almost uniform selection observed in most models using logarithmic returns. This preference for futures in models utilizing fractionally differentiated series is consistent across the hourly, daily, and weekly horizons. For the monthly horizon, we do not calculate fractionally differentiated series due to

data availability. These findings suggest that fractionally differentiated series might capture certain patterns that standard logarithmic returns are unable to detect.

In conclusion, our findings provide practical implications for the ability of deep learning models to predict financial returns. The most crucial takeaway is that LSTM networks are particularly effective for short-term financial return predictions at hourly and daily horizons. However, their performance decreases for longer horizons, such as weekly and monthly possibly due to fewer market inefficiencies to exploit. While we could not find a profitable strategy under 0.05% transaction costs, we demonstrated how the same model architecture behaves across different horizons and markets, enhancing the robustness and generalizability of the findings. Future research could expand the analysis to a broader futures market using fractionally differentiated series to test our findings. Additionally, testing different transaction costs and removing the short-selling component of the portfolio could potentially deliver profitable returns even for longer horizons.

Bibliography

Akyildirim, E., A. Goncu, & A. Sensoy (2021): "Prediction of cryptocurrency returns using machine learning." *Annals of Operations Research* **297(1)**: pp. 3–36.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, & D. Amodei (2020): "Language Models are Few-Shot Learners." ArXiv:2005.14165 [cs].

Cerchiello, P., G. Nicola, S. Ronnqvist, & P. Sarlin (2018): "Deep learning bank distress from news and numerical financial data." ArXiv:1706.09627 [cs, stat].

Dixon, M. F., I. Halperin, & P. Bilokon (2020): *Machine Learning in Finance: From Theory to Practice*. Cham: Springer International Publishing.

Fama, E. F. (1970): "Efficient Capital Markets: A Review of Theory and Empirical Work." *The Journal of Finance* **25(2)**: pp. 383–417. Publisher: [American Finance Association, Wiley].

Fischer, T. & C. Krauss (2018): "Deep learning with long short-term memory networks for financial market predictions." *European Journal of Operational Research* **270(2)**: pp. 654–669.

Ghosh, P., A. Neufeld, & J. K. Sahoo (2022): "Forecasting directional movements of stock prices for intraday trading using LSTM and random forests." *Finance Research Letters* **46**: p. 102280.

Goodfellow, I., Y. Bengio, & A. Courville (2016): *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gu, S., B. Kelly, & D. Xiu (2020): "Empirical Asset Pricing via Machine Learning." *The Review of Financial Studies* **33(5)**: pp. 2223–2273.

Hambly, B., R. Xu, & H. Yang (2023): "Recent Advances in Reinforcement Learning in Finance." ArXiv:2112.04553 [cs, q-fin].

Hsu, M.-W., S. Lessmann, M.-C. Sung, T. Ma, & J. E. Johnson (2016): "Bridging the divide in financial market forecasting: machine learners vs. financial economists." *Expert systems with Applications* **61**: pp. 215–234.

Ioffe, S. & C. Szegedy (2015): "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In "International conference on machine learning," pp. 448–456. pmlr.

Israel, R., B. T. Kelly, & T. J. Moskowitz (2020): "Can machines' learn'finance?" *Journal of Investment Management* .

Jiang, W. (2021): "Applications of deep learning in stock market prediction: Recent progress." *Expert Systems with Applications* **184**: p. 115537.

Jurgovsky, J., M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, & O. Caelen (2018): "Sequence classification for credit-card fraud detection." *Expert Systems with Applications* **100**: pp. 234–245.

Krauss, C., X. A. Do, & N. Huck (2017): "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500." *European Journal of Operational Research* **259(2)**: pp. 689–702.

Krizhevsky, A., I. Sutskever, & G. E. Hinton (2012): "ImageNet Classification with Deep Convolutional Neural Networks." In "Advances in Neural Information Processing Systems," volume 25. Curran Associates, Inc.

Lago, J., G. Marcjasz, B. De Schutter, & R. Weron (2021): "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark." *Applied Energy* **293**: p. 116983.

Li, J., H. Bu, & J. Wu (2017): "Sentiment-aware stock market prediction: A deep learning method." In "2017 International Conference on Service Systems and Service Management," pp. 1–6. ISSN: 2161-1904.

Li, Z., J. Han, & Y. Song (2020): "On the forecasting of high-frequency financial time series based on ARIMA model improved by deep learning." *Journal of Forecasting* **39(7)**: pp. 1081–1097. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/for.2677.

Luo, C., D. Wu, & D. Wu (2017): "A deep learning approach for credit scoring using credit default swaps." *Engineering Applications of Artificial Intelligence* **65**: pp. 465–470.

Mitchell, T. M. (1997): *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, & D. Hassabis (2015): "Human-level control through deep reinforcement learning." *Nature* **518(7540)**: pp. 529–533. Number: 7540 Publisher: Nature Publishing Group.

Orimoloye, L. O., M.-C. Sung, T. Ma, & J. E. Johnson (2020): "Comparing the effectiveness of deep feedforward neural networks and shallow architectures for predicting stock price indices." *Expert Systems with Applications* **139**: p. 112828.

Ozbayoglu, A. M., M. U. Gudelek, & O. B. Sezer (2020): "Deep learning for financial applications: A survey." *Applied soft computing* **93**: p. 106384.

Prado, M. L. d. (2018): *Advances in Financial Machine Learning*. John Wiley & Sons. Google-Books-ID: oU9KDwAAQBAJ.

Rosenblatt, F. (1958): "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* **65(6)**: p. 386.

Rumelhart, D. E., G. E. Hinton, & R. J. Williams (1986): "Learning representations by back-propagating errors." *nature* **323(6088)**: pp. 533–536.

Sezer, O. B., M. U. Gudelek, & A. M. Ozbayoglu (2020): "Financial time series forecasting with deep learning : A systematic literature review: 2005–2019." *Applied Soft Computing* **90**: p. 106181.

Staněk, F. (2023): "A Note on the M6 Forecasting Competition: Rank Optimization." *SSRN Electronic Journal* .

Thakkar, A. & K. Chaudhari (2021): "A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions." *Expert Systems with Applications* **177**: p. 114800.

Zhu, X., H. Wang, L. Xu, & H. Li (2008): "Predicting stock index increments by neural networks: The role of trading volume under different horizons." *Expert Systems with Applications* **34(4)**: pp. 3043–3054.