Momentum in the Era of Machines

Master's Thesis
Cand.Merc FSM
Copenhagen Business School

112 pages
17 May 2021

Christine Mering (110613)
Maria Rosendal Moradjow-Namin (109867)

Supervisor: Lasse Heje Pedersen
Co-supervisor: Theis Ingerslev Jensen

Abstract

The rapidly evolving field of machine learning, fostered by the eruption of data availability and computing power, is auspicious for investment management. The ability of machine learning to uncover patterns in data holds particular promise for return prediction and thus, for the enhancement of trading strategies. One of the most pervasive trading strategies in the financial literature is the price momentum strategy. With its occasional but severe profit punctuations, the momentum strategy poses an interesting case for reformation through machine learning. Motivated by the unique potential of coupling "momentum and machines", we investigate the performance of three trading strategies based on machine learning models with momentum-related input variables. We find that such machine learning-based trading strategies outperform the conventional price momentum strategy and mitigate its profit punctuations considerably. Further, we document a large economic potential from an ensemble strategy that equally weighs the three machine learning-based trading strategies and the conventional price momentum strategy. The ensemble strategy outshines all individual strategies and approximately doubles the Sharpe ratio of the conventional momentum strategy. Thus, our findings offer supportive evidence for the benefits of applying machine learning to return prediction. However, the complex nature of machine learning models makes it difficult to draw meaningful interpretations of what drives their superior performance. Hence, while the machine learning models engender profitable trading strategies, their complexity induces limited interpretability. Consequently, the enigma of machine learning hinders the practical implementation of our findings.

Table of Contents

1 Introduction

The era of machine learning has emerged from the nexus of unprecedented computing power, widely available data, and advances in scientific methods. This era offers traders and investment managers novel opportunities to exploit previously undiscovered patterns in, and beyond, stock markets for superior investment decisions (Arnott, Harvey & Markowitz, 2019). This thesis participates in the ongoing discussion on the prospect of applying machine learning in investment management.

"Machine learning" is the study of computer algorithms that possess the ability to automatically learn through experience (Mitchell, 1997). As the public media frequently reminds us, machine learning can accomplish the once unthinkable - from recognizing images and speech to beating grandmasters at complex games of strategy (Israel, Kelly & Moskowitz, 2020). The tremendous success of machine learning has spurred interest in applying it across a variety of fields - and investment management is no exception. At the heart of investment management is portfolio construction, in which the most important task is return prediction. The ability of machine learning algorithms to learn through experience allows them to uncover subtle, contextual, and nonlinear relationships when predicting returns and thus, shows great promise for developing successful trading strategies (Gu, Kelly & Xiu, 2019; Israel et al., 2020). Against this background, a question arises: Which trading strategy emerges as the preferred choice for capitalizing on the opportunities presented by machine learning?
While research documenting the benefits of machine learning for trading strategies lies within its infancy, the findings of Gu et al. (2019) point to large economic gains for traders and investment managers when utilizing machine learning forecasts for *stock-selection strategies*.

One of the most pervasive stock-selection strategies documented in the financial literature is the price momentum strategy. The strategy exploits the continuing trend in stock prices over a horizon of up to 12 months, by predicting the cross-section of future stock returns based on the cross-section of past stock returns. The existence of price momentum in the US stock market was first documented by Jegadeesh and Titman (1993). They observe that stocks with higher (lower) past returns during the past 3-12 months, continue to exhibit higher (lower) returns over the following 3-12 months. Multiple researchers confirm the findings of Jegadeesh and Titman (1993), by demonstrating the profitability of a stock-selection strategy exploiting the momentum effect (e.g., Rouwenhorst, 1998; Griffin, Ji & Martin, 2003, Israel & Moskowitz, 2013). While the financial literature documents the efficacy of momentum, the strong performance of the strategy is punctuated with occasional crashes. These crashes are referred to as "momentum crashes" and occur in times of *market stress* - when a bear market is followed by a sudden and dramatic market upswing (Daniel & Moskowitz, 2016).

Figure 1.1: Cumulative returns of the momentum strategy

Figure 1.1 notes: The figure depicts the total cumulative return (including the risk-free rate) of the momentum strategy during the period January 1995 - December 2020. The momentum strategy is based on a 12-month formation period, skipping the most recent month, and a one-month holding period. The momentum strategy crashed in 2009, evident by a substantial drop in the cumulative returns. (Source: Own creation based on Kenneth French Data Library).

The momentum crash following the 2008-2009 financial crisis (Figure 1.1) engendered scholars to shift their attention towards enhancing the "conventional" momentum strategy. By incorporating various risk factors or combining the strategy with alternate trading strategies, scholars mitigate the crash risk inherent to the conventional strategy. Specifically, they utilize several variables that impact the performance of the conventional momentum strategy ("momentum variables"), such as the volatility of the strategy or the market state. These enhanced momentum strategies all exhibit superior performance relative to the conventional momentum strategy, in the literature (Blitz, Huij & Martens, 2011; Moskowitz, Ooi & Pedersen, 2012; Asness, Moskowitz & Pedersen, 2013; Barroso & Santa-Clara, 2015; Wang & Xu, 2015; Hanauer & Windmüller, 2020).

The crash risk inherent to the conventional momentum strategy makes it an unappealing strategy for any risk-averse trader or investment manager. At the same time, a large body of research on how to enhance the strategy exists. Uniting these two properties of the momentum strategy renders it an interesting case for amelioration through machine learning: Machine learning may be able to find a novel way to enhance the conventional strategy by uncovering contextual and nonlinear relationships across the momentum variables utilized in the literature that seeks to enhance the momentum strategy.

1.1 Problem Statement

With the unique potential of coupling momentum and machines, we aspire to address an untapped research gap in the literature. The research of this thesis manifests itself in two overarching forms. First, we investigate if stock-selection strategies based on machine learning showcase superior performance and less crash risk relative to the conventional momentum strategy. Second, we examine if machine learning-based stock-selection strategies may advantageously be combined with the conventional momentum strategy in order to enhance its performance.

Specifically, we employ the momentum variables that have been documented in the literature to enhance the performance of the conventional strategy. While a consensus remains to be found on which of the enhanced momentum strategies is superior, they all offer the potential for improving the conventional strategy in each their own way. As a result of the latter, this thesis deploys all of the momentum variables *collectively* as inputs for the machine learning models. By combining several momentum variables, the machine learning models can extract previously undiscovered patterns from each of these variables as well as the interactions between them. In contrast, the conventional momentum strategy relies exclusively on recent past returns when it sorts stocks into portfolios.

We aspire to construct machine learning-based stock-selection strategies that exhibit superior economic performance relative to the conventional momentum strategy while being exposed to less crash risk. As such, we participate in the ongoing debate on the potential of applying machine learning for investment management. Thus, we arrive at the following problem statement:

How do stock-selection strategies utilizing machine learning based on momentum variables perform relative to the conventional momentum strategy, and how does the impact of stressed economic markets compare for the two strategies?

To answer our problem statement, we dissect the research of this thesis into six areas: First, we outline the theoretical framework underpinning the conventional and enhanced momentum strategies as well as the machine learning models. We then replicate the conventional momentum strategy and examine its crash risk during periods of market stress. Subsequently, we investigate if stock-selection strategies based on machine learning exhibit superior performance relative to the conventional momentum strategy, across market states. More specifically, we implement three machine learning models, namely Random Forest, the Multilayer Perceptron, and a Recurrent Neural Network with a Long Short-Term Memory architecture. For each of these models, we utilize the various momentum variables as inputs. Next, we explore the performance of an ensemble strategy that combines the three machine learning-based stock-selection strategies with the conventional momentum strategy. Then, we investigate possible explanations for the performance of the machine learning-based stock-selection strategies (and the ensemble strategy, implicitly), by examining both

the input variables and renowned risk factors of the asset management literature. Finally, we discuss our findings and place them in an academic and practical context.

To guide our research and answer our overall problem statement, we have outlined the following research questions:

1. *What evidence underpins momentum strategies and the ability of machine learning models to predict stock returns?*
2. *How does a conventional momentum strategy perform over time and during periods of market stress?*
3. *How do stock-selection strategies based on machine learning perform over time and during periods of market stress, and how does the performance compare to the conventional momentum strategy?*
4. *What is the potential for enhancing the conventional momentum strategy when combining it with machine learning-based stock-selection strategies?*
5. *Which input variables and risk factors may explain the performance of the machine learning-based investment strategies?*
6. *What does the performance of the machine learning-based investment strategies relative to the conventional momentum strategy imply for academia and practitioners?*

1.2 Contribution of the Thesis

The research of this thesis carries relevance across multiple arenas in both academia and practice. Its *academic* contribution manifests itself in two ways: First, this thesis combines insights from multiple studies on enhanced momentum strategies. We do not seek to single out one superior enhanced strategy but to extract a *combined* effect by leveraging the insight of the studies. To the best of our knowledge, no previous studies apply this approach, substantiating the relevance and novelty of our research. Second, this thesis bridges the chasm between two previously separated bodies of research: momentum strategies and machine learning. Specifically, we deploy three machine learning models to predict future excess stock returns. The machine learning models utilize the momentum variables as inputs, enabling the models to extract previously undiscovered patterns *across* the variables. Ultimately, we contribute to the literature on momentum strategies, by using machine learning to enhance the performance of and mitigate the crash risk inherent to the conventional momentum strategy. With an annualized Sharpe ratio of 0.83 and a Sortino ratio of 1.16, the ensemble strategy exhibits superior performance and lower crash risk relative to the conventional momentum strategy. Thus, the findings of this thesis may also be of interest to *practitioners* who wish to deploy a machine learning-based stock-selection strategy. However, the research of this thesis offers limited insights into the underlying drivers of these strategies. In this regard, we note that potential traders or investment managers should be cautious and not blindly trust the findings of this thesis for two reasons: First, we rely on the method of backtesting, and thus, our results hold no promise for the future performance of a trading strategy. Second, the machine learning models we deploy are inscrutable and produce results that are associated with randomness, rendering them difficult to apply in practice.

1.3 Scope and Delimitations

The contribution of this thesis should be considered in light of its limitations. In bridging the chasm between momentum and machine learning, the research of this thesis is naturally nested in a wide area of both economic and machine learning theory. Thus, minding the scope of this thesis, we assume that the reader possesses a fundamental understanding of asset pricing models and statistics. However, we do not expect the reader to have any programming experience and do not include any code in our thesis, accordingly. Hence, when discussing the computational methods for constructing each machine learning model, we do not dwell on the subtle technical nuances of the models. Instead, we refer to our code in appendices B and point to relevant appendices underway with more in-depth technical details.

The research of this thesis is constrained along three dimensions, namely a geographic, temporal, and thematic dimension (Yin, 2018). We proceed by identifying the most important delimitations within these overarching dimensions.

Geographic scope

Several studies document that the profitability of the momentum strategy is not limited to the US market but also evident in other geographies (see section 2.1). However, given our problem statement, we do not consider it prudent to span our analysis across multiple markets. The rationale for this delimitation is twofold: First, the US stock market bears more relevance for our research as it eases comparability of our findings with studies on enhanced momentum strategies that mostly concentrate on this market. Second, the computational power in our possession constitutes a considerable limitation that makes it infeasible for us to include several markets. In fact, the entire US stock market is too large to include in this thesis: There are three major stock exchanges in the US stock market, namely the New York Stock Exchange ("NYSE"), the National Association of Securities Dealers Automated Quotations ("NASDAQ") and the American Stock Exchange ("AMEX"). The total amount of stocks listed on one of these exchanges at some point during our sample period is 25,432 stocks which result in an infeasible amount of data for this thesis. Consequently, we select the 500 largest stocks each month, measured by the beginning of month market capitalization[1]. We refer to this sample as our "investment universe".

Temporal scope

The research of the thesis is constrained temporally with a sample period from January 1929 to December 2020. Machine learning models thrive in data-rich environments from which the algorithms can learn the patterns in data (Israel et al., 2020). Hence, we seek to include a long sample period, dating back as far as possible. However, the data from the Center for Research in Security Prices (CRSP) prior to 1929 is not applicable for our thesis, as we require a minimum of 500 stocks to enter our universe each month. As a result, we exclude the first years (from 1926 to 1929) from our sample.

[1] The calculation of the market capitalization can be found in section 5.2.

Of particular importance to this thesis, the sample period includes the two periods with the most predominant momentum crashes: the Great Depression and the financial crisis. Thus, we are able to investigate whether a machine learning-based stock-selection strategy is subject to the same crash risk as the momentum strategy.

Thematic scope

Cross-sectional Momentum

The research of this thesis is centered around cross-sectional momentum which focuses on the *relative* performance of stocks in the cross-section rather than time-series momentum which focuses on a stock's own past returns. Time series momentum has gained much attention since it was introduced by Moskowitz, Ooi, and Pedersen (2012). However, time series momentum has been documented to exhibit superior performance than the conventional momentum strategy during crashes, rendering it less relevant for our research, as we focus on the crash risk of the momentum strategy. Throughout this thesis, any remarks on momentum refer to cross-sectional momentum.

This thesis focuses on long-short (zero-cost) momentum strategies as these are the most predominant momentum strategies in the literature (e.g., Jegadeesh & Titman, 1993; Rouwenhorst, 1998; Asness et al., 2013). Unless explicitly stated otherwise, we refer to a long-short momentum strategy when mentioning a momentum strategy.

Machine learning models

Several different machine learning algorithms exist with many variants of each algorithm. Deploying numerous algorithms would be infeasible within the scope of this thesis. Thus, we single out machine learning algorithms which the literature documents as most promising for stock predictions. Specifically, we select three different algorithms, namely Random Forest, Multilayer Perceptron, and Recurrent Neural Network with the Long Short-Term Memory architecture. The choice of machine learning algorithms relies mainly on the results of Gu et al. (2019), who obtain the best results using tree-based models and neural networks. Out of the several variations of tree-based models and neural networks, we limit ourselves to pick the three aforementioned models. We further elaborate on the rationale hereof in chapter 7.

Transaction costs and short sale

Consistent with most of the literature on momentum, the research of this thesis does not account for transaction costs but focuses on gross returns. In this regard, we note that gross returns are considered more suitable for investigating the relationship between risk and returns. Applying gross returns overstate the profitability of investment strategies if pursued in *practice* (Asness et al., 2013). Hence, the findings of this thesis bear no guarantee for profitability if applied as a real-time trading strategy - even if the methodology of backtesting and the randomness associated with the models are no constraint.

In line with extant literature, the research of this thesis assumes no margin calls during the holding period as well as no limitations on short sales (Daniel & Moskowitz, 2016).

Acknowledging that the contribution of this thesis is bound by its delimitations, we emphasize the possibility for further research to enrich our analysis.

1.4 Structure of the Thesis

With our problem statement and research questions as a point of departure, we structure this thesis into six overarching parts. To guide the reader, we have illustrated the structure of the thesis in Figure 1.2.

The **first part** sets the scene of the thesis. The aim of this part is to introduce the overall topic, clarify the contributions of the thesis, and specify its scope and delimitations. Most importantly, the first part articulates the problem statement and research questions that function as a guide for our research.

In **the second part** of the thesis, we strive to answer the first research question by placing the research in its academic and theoretical context. Specifically, this part provides an overview of the most profound studies on momentum strategies as well as studies documenting the ability of machine learning models to construct trading strategies. Furthermore, we outline the theoretical framework underpinning the research of this thesis: First, we offer an understanding of the key concepts of equity trading and the momentum strategy. Then, we provide a fundamental understanding of machine learning and unfold the architecture of machine learning algorithms.

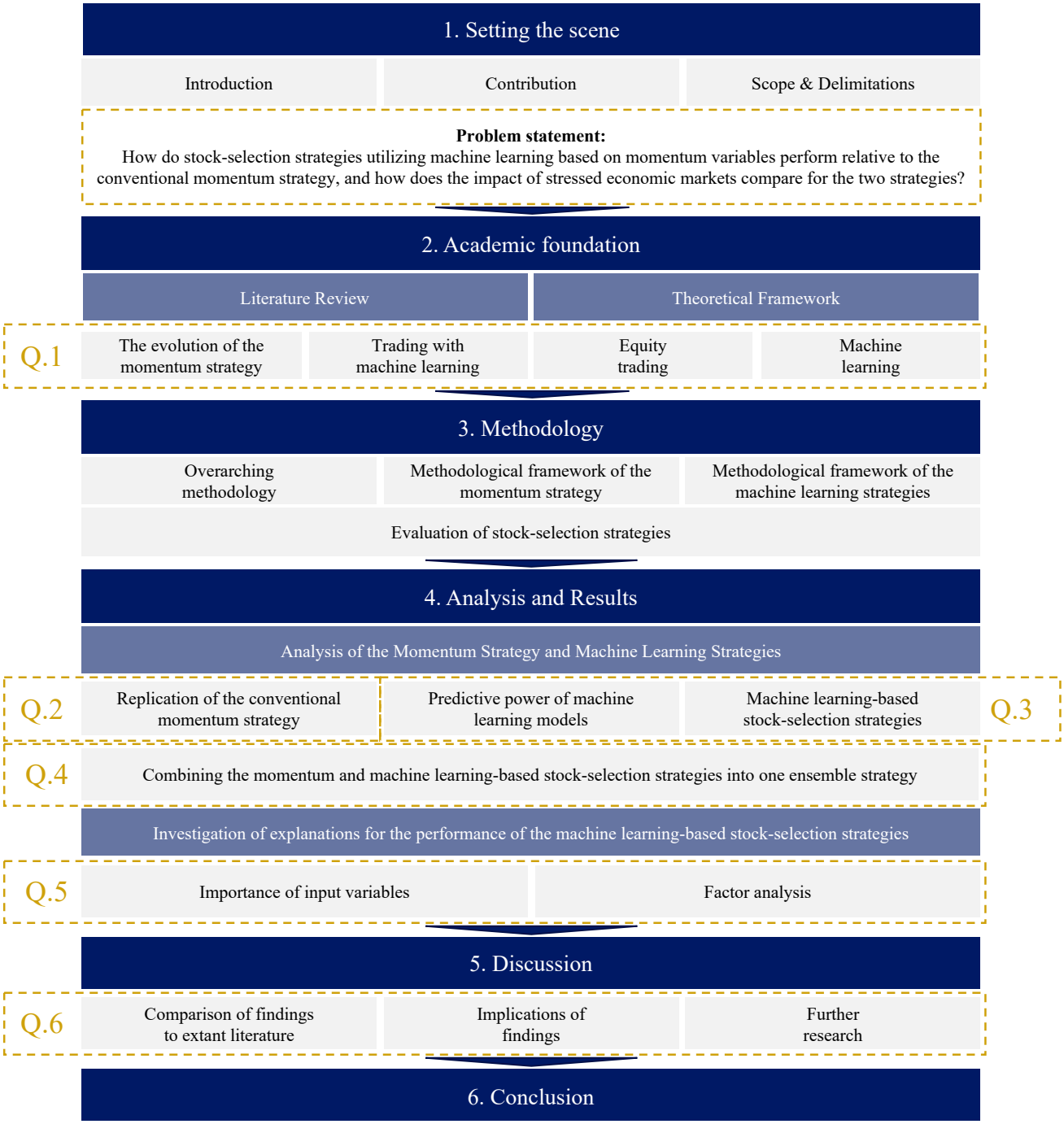
The third part sheds light on the methodology that our research relies on. We outline the overarching methodology of the thesis by introducing the method of backtesting and providing an overview of our data sample. Further, we zero in on the methodological approach for constructing the momentum variables, the conventional momentum strategy, and the stock-selection strategies based on machine learning. Finally, the third part presents a framework for evaluating the performance of the four stock-selection strategies.

On the basis of the first three parts, **the fourth part** of the thesis unfolds the analysis and results of our research, answering four of our research questions. More specifically, we initiate the fourth part by presenting the replicated conventional momentum strategy, and thus, we address the second research question. We answer the third research question by examining the predictability of the three machine learning models and construct stock-selection strategies that rely on the model predictions. In order to examine the ability of machine learning to enhance the momentum strategy and answer research question four, we form an ensemble strategy. Lastly, we aspire to answer research question five by examining two possible explanations for the performance of the machine learning-based stock-selection strategies, namely the input variables of the machine learning models and risk factors in the financial literature.

In **the fifth part**, we answer the final research question by discussing the findings of our research. In this regard, we shed light on the implications of our findings for both academia and practitioners and explore interesting topics for further research.

Finally, **the sixth part** of the thesis synthesizes the findings of our research and provides an overall conclusion to our problem statement.

Figure 1.2: Structure of the thesis



(Source: Own creation)

2 Literature review

This chapter presents a review of the most relevant studies for the research of this thesis. The chapter serves a two-fold purpose: The first section outlines the origins of the momentum literature and places the topic in its current academic context. The second section introduces the literature that combines machine learning and investment management, pointing out recent developments. Please note that this chapter presents an overview of the extant literature exclusively, while chapters 3 and 4 present the theoretical foundation underpinning the relevant studies of this thesis. Collectively, these three chapters serve to answer research question one.

2.1 The Evolution of the Momentum Strategy

The price momentum effect is one of the most widely studied phenomena in the financial literature and has arisen as an input to the debate on whether financial markets are efficient. The literature on momentum takes its point of departure with the seminal work of Jegadeesh and Titman (1993) who documented the existence of the price momentum effect; the tendency for past winners to outperform past losers. Focusing on NYSE and AMEX over the sample period spanning 1965 to 1989, they examine a variety of momentum strategies. In forming these strategies, they first observe the returns of stocks over a J-month period, referred to as the formation period. Then, they rank stocks into decile portfolios according to their performance in the formation period. They form a long-short portfolio, also referred to as the winner minus loser portfolio ("WML") by taking a long position in the stocks that yielded the highest returns and shorting the stocks that yielded the lowest returns. This portfolio is held for a K-month holding period. Jegadeesh and Titman (1993) demonstrated that following this strategy for J/K = [3, 6, 9, 12] months[2] generates a monthly return of as much as 1.49% when applying overlapping holding periods and rebalancing the portfolios every month to maintain equally weighted portfolios.

A range of different scholars adopt a similar approach as Jegadeesh and Titman (1993) and confirm the momentum effect across different geographical markets and asset classes (Rouwenhorst 1998, 1999; Chui, Wei & Titman, 2000; Jegadeesh and Titman 2001; Griffin, Ji & Martin, 2003; Okunev & White 2003; Sapp & Tiwari, 2004; Miffre & Rallis 2007; Chui, Titman & Wei, 2010; Asness et al., 2013; Menkhoff, Sarno, Schmeling, & Schrimpf, 2012; Jostova, Nikolova, Philipov & Stahel, 2013)

However, conventional price momentum strategies have also been criticized for their highly unstable performance across subperiods (e.g., Grundy & Martin, 2001; Blitz et al., 2011; Daniel & Moskowitz, 2016). Grundy and Martin (2001) document that price momentum strategies have substantial time-varying exposure to systematic risk factors. Specifically, momentum loads positively (negatively) on systematic factors in the holding period when these factors have positive (negative) returns during the formation period. Consequently, the strategy experiences losses when the sign of the factor returns over the holding period is opposite to the sign over the formation period.

[2] Today, most scholars deploy a formation period observing returns over the past 12 months, skipping the most recent month, and a holding period of one-month

Grundy & Martin (2001) argue that a momentum strategy tends to go long in stocks with betas greater than one and short in stocks with betas less than one following up markets ("positive beta bet" on the market) and vice versa for down markets ("negative beta bet" on the market). When a down market rebounds quickly, the negative beta bet leads the profit of the momentum strategy to punctuate.
In the light of Grundy and Martin's (2001) findings, Daniel and Moskowitz (2016) examine the "crashes" of the momentum strategy at market rebounds. For their US equity sample spanning 1927 to 2013, Daniel and Moskowitz (2016) find that the performance of a momentum strategy with a 12-month formation period (skipping the most recent month) experienced the worst performance following the Great Depression and the 2008-2009 financial crisis. In July and August 1932, the loser portfolio generated a return of 232% while the winner portfolio obtained a return of only 32%. Similarly, over a three-month period from March to May 2009, the loser portfolio rose by 163% while the winner portfolio gained only 8%. Based on their empirical observations, Daniel and Moskowitz (2016) conclude that the momentum crashes occur following market declines, when volatility is high, and the market dramatically rebounds. Moreover, they demonstrate that momentum crashes are mostly attributable to the performance of the loser portfolio.

The crash of the conventional momentum strategy following the 2008-2009 financial crisis engendered scholars to shift their attention towards enhancing the performance of the strategy (e.g., Blitz et al. 2011; Barroso & Santa-Clara 2015; Wang & Xu 2015; Daniel & Moskowitz, 2016). To mitigate the risk of momentum crashes, multiple scholars have developed volatility-scaled extensions of the conventional momentum strategy (e.g., Barroso & Santa-Clara, 2015; Daniel & Moskowitz, 2016). Where Barroso and Santa-Clara (2015) propose a constant volatility-scaled momentum strategy, Daniel and Moskowitz (2016) suggest a dynamic volatility-scaled momentum strategy. Both studies find that the volatility of the conventional momentum strategy is predictable and demonstrate that the volatility-scaled extensions approximately double the Sharpe ratios of the conventional momentum strategy.
Additional extensions of the conventional momentum strategy have emerged, including alpha momentum and idiosyncratic (residual) momentum. The difference between these extensions and the conventional momentum strategy revolves around the ranking of stocks during portfolio formation (Singh & Walia, 2020). While conventional momentum strategies rank stocks based on their total return, the extended momentum strategies focus on the stock-specific return components to minimize the dependency on factor realizations. Grundy and Martin (2001) as well as Hühn and Scholz (2018) rank stocks based on their alpha and show that alpha momentum exhibits less dynamic factor exposures than conventional momentum. Moreover, several scholars examine idiosyncratic momentum, where stocks are ranked based on their residual returns (e.g., Gutierrez & Prinsky, 2007; Blitz et al., 2011; Blitz et al., 2020).
More recently, Hanauer and Windmüller (2020) compare the constant volatility-scaled, the dynamic volatility-scaled, and the idiosyncratic momentum strategy. Based on a sample of 48 international markets that spans 1991 to 2017, they document that all three strategies exhibit higher t-statistics and Sharpe ratios than the conventional momentum strategy.

In summary, while the profitability of the momentum strategy has been documented extensively, it exhibits crash risk that makes it unappealing to any risk-averse investor. Several enhancements of the conventional momentum strategy exist that exhibit superior performance relative to the conventional strategy. However, a

consensus remains to be found on which of the extensions is superior. For the purpose of this thesis, the lack of consensus is emphasized, as it prompts us to leverage the insights of the strategies collectively.

2.2 Trading with Machine Learning

Resonating Israel et al. (2020), machine learning has the potential to be an important step forward for investment management. While *methodical* research examining the benefits of machine learning for stock-selection strategies is still in its early stages, research on utilizing machine learning for financial time series forecasting has been around for decades (Trippi & Turban, 1993; Kaastra & Boyd, 1996).

Throughout time, many different machine learning algorithms have been deployed within the field of finance. During the 2000s, the support vector machine ("SVM") became a popular algorithm for financial time series forecasting following the seminal work of Tay and Cao (2001A). They document the ability of SVMs to predict stock indices and bond values based on historical prices, obtaining a better mean squared error and mean absolute error than previous time series models. In the following years, several studies yield similar results, demonstrating the superior performance of the SVM relative to conventional statistical methods, when predicting the movements in stock indices (Tay & Cao, 2001B; Huang, Nakamori & Wang, 2002; Kim, 2003; Ince & Trafalis, 2008).

Recently, scholars document that neural networks and tree-based models substantially outperform the more traditional machine learning counterparts (e.g., Heaton, Polson, & Witte, 2016; Chong, Han, & Park, 2017; Krauss, Do, & Huck, 2017). Notably, Fischer & Krauss (2018) document that Recurrent Neural Networks with a Long Short-Term Memory architecture outperform less advanced machine learning models when predicting the out-of-sample directional movements for the stocks of the S&P 500 over a sample period spanning 1992 to 2015. Other studies support these findings, arguing that Recurrent Neural Networks with a Long Short-Term Memory architecture are superior due to their ability to learn long-term dependencies in time series (Bao, Yue, & Rao, 2017; Shen & Shafiq, 2020; Wang, Li, Zhang & Liu, 2020; Pang et al., 2020).

However, most previous research focuses on predicting stock index returns, while few scholars examine a stock-selection strategy that utilizes machine learning predictions of individual stock returns. Recently, Gu et al. (2019) lay forth pivotal and methodical research examining several different machine learning models for individual stock return predictions. Using a sample of US stocks from 1957 to 2016, they identify the best performing models for a long-short decile portfolio as neural networks, followed by tree-based models. Specifically, a stock-selection strategy that relies on stock returns predicted by a neural network earns an annualized out-of-sample Sharpe ratio of 2.45, more than doubling the performance of leading regression-based strategies from the literature. In a similar manner, other financial scholars are initiating the research of deploying machine learning for stock-selection strategies (e.g., Chen, Pelger & Zhu, 2020; Freyberger, 2020).

In summary, two key points can be deduced from this section: First, while methodical research into how machine learning can enhance stock-selection strategies is still in its infancy, the *potential* of machine learning

to enhance investment strategies is profound. Second, recent literature emphasizes the superior performance of tree-based models and neural networks. For the purpose of this thesis, the latter point is emphasized, as these findings constitute the rationale for our choice of machine learning models.

3 Equity Trading

This chapter provides the theoretical framework for replicating the momentum strategy and constructing the momentum variables used as inputs for the machine learning models. The chapter is composed of five sections. The two first sections survey the key concepts of technical analysis and efficient markets. The third section describes the asset pricing models utilized in this thesis. Subsequently, the fourth section zooms in on the momentum effect and the performance of the momentum strategy over time. Moreover, this section provides an overview of the four enhanced momentum strategies that constitute the foundation for the momentum variables. Finally, the fifth section provides a brief overview of the explanations for the momentum effect.

3.1 Technical Analysis

There are two major types of analyses for stock investing, namely fundamental analysis and technical analysis. Fundamental analysis focuses on the intrinsic value of a stock and compares it to the stock price (Pedersen, 2015). If the intrinsic value is higher than the price of the stock, the stock is undervalued and presents an attractive investment opportunity for an investor. To estimate the intrinsic value, analysts use factors that might affect the future profits of the stock, be it macroeconomic factors such as interest rates or microeconomic variables such as the earnings of a company. However, the research of this thesis relies on technical analysis and thus fundamental analysis is de-emphasized.

Technical analysis focuses on recurrent and predictable patterns in the movements of stock prices. Investment opportunities are identified through statistical trends in the history of the stock price, return, volatility, etc. (Pedersen, 2015). Technical analysis does not deny the value of fundamental information but seeks to exploit that prices diverge from their intrinsic value. Consequently, technical analysis relies on the *timing* of the stock price reaction to fundamental information.

The trends in stock prices arise due to initial stock price underreaction or delayed overreaction. If a stock price initially underreacts to fundamental information, investors can exploit a slow price adjustment towards a new equilibrium. In a similar manner, prices can be driven away from the fundamental value if investors overreact to information (see section 3.5 for elaboration on under- and overreaction) (Pedersen, 2015). As a result, technical analysis can exploit trends regardless of the (fundamental) reason for the stock price movement.

Both fundamental and technical analysts attempt to beat the market. However, the efficient market hypothesis contends that utilizing fundamental and/or technical analysis to beat the market is in vain.

3.2 The Efficient Market Hypothesis

The efficient market hypothesis was introduced by Fama (1970) and has since been a cornerstone in asset pricing theory. Fama (1970) outlined three market conditions consistent with efficiency: 1) No transaction costs, 2) all information is available to all market participants without charge, and 3) market participants all agree on the implications of the information for the stock price. If these conditions are fulfilled, the hypothesis dictates that prices reflect all relevant information at all times. Fama (1970) argued that competition in the market ensures that new information is quickly incorporated into stock prices. If all information is reflected in stock prices, and the flow of information is unpredictable and unimpeded, future movements in stock prices will be random. Therefore, stock prices are assumed to follow "a random walk" (Fama, 1970).

Fama (1970) introduced three forms of market efficiency, namely weak, semi-strong and strong. First, a market is efficient in the weak form if the stock prices reflect *market trading data*, such as past stock prices or trading volume. For this form of efficiency, investors are unable to use technical analysis as the basis for a profitable investment strategy, as the data would already have been exploited insofar the data conveyed reliable signals about future performance. Second, a market is efficient in the semi-strong form if the stock prices reflect *all publicly available information* regarding the prospects of a firm, e.g., annual reports, quality of management, and earnings forecasts. As a result, no publicly available data can be utilized to predict the future stock price, rendering technical and fundamental analysis futile. Lastly, a market is efficient in the strong form if the stock prices reflect *all information* relevant to the firm, be it publicly or privately available (Fama, 1970).

Of relevance to this thesis, the momentum strategy exploits patterns in stock returns and is one of the most infamous stock-selection strategies relying on technical analysis. Furthermore, the scholars that enhance the conventional momentum strategy equally rely on various technical variables. Consequently, generating abnormal returns through either the conventional momentum strategy or enhanced momentum strategies is a direct contradiction to the efficient market hypothesis, as the market would not even be efficient in the weak form.
The machine learning-based stock-selection strategies constructed in this thesis also rely on technical variables. Hence, if these strategies generate abnormal returns, they will equally be a violation of the weak form efficient market.

3.3 Asset Pricing Models

Multiple scholars within the field of momentum document that technical variables can be deployed to minimize the crash risk of the conventional momentum strategy. We rely on their findings and utilize these variables when creating the machine learning-based stock-selection strategies. The technical variables are constructed based on two asset pricing models. This section unfolds the theory of the asset-pricing models before section 3.4 elaborates on how the momentum variables are constructed in the literature. We initiate the section by describing the theory of the Capital Asset Pricing Model (CAPM). Subsequently, we outline the Fama/French

three-factor model. Finally, we briefly describe two additional Fama/French factors which are not used for variable construction but applied in the factor analysis in section 9.6.

The Capital Asset Pricing Model

In finance, the most acknowledged risk component is *systematic risk,* composed of the risk inherent in the overall market and the covariance of stocks with the market. The systematic risk of an individual asset denoted beta (β_i), is measured by its contribution to the variance of the overall market portfolio. Beta can be expressed as follows:

$$\beta_i = \frac{\sigma(R_i, R_m)}{\sigma^2(R_m)} \tag{1}$$

where R_i is the return of stock i and R_m is the market return

The beta of an asset can be derived from the most predominant asset pricing model in the literature of finance, namely the CAPM, introduced by Sharpe (1964) and Lintner (1965).

The central prediction of the CAPM is that the market portfolio is mean-variance efficient as argued by Markowitz (1952). The efficiency of the market implies that the expected return of an asset is independent of firm-specific risk factors. Consequently, the expected return of a security is a positive linear function of its beta, and beta suffices to describe the cross-section of expected returns in a univariate model. The model can be expressed as below where the stock (or portfolio) *i*'s expected return is a function of the expected market return in excess of the risk-free rate*:*

$$E(R_i) - R_f = \beta_i[E(R_m) - R_f] \tag{2}$$

where $E(R_i)$ denotes the expected return of stock i, R_f is the risk-free return, and $E(R_m)$ is the expected market return

In equation 2, beta can also be defined as the slope in the time-series regression model for stock *i*:

$$R_{i,t} - R_{f,t} = \alpha_i + \beta_i[R_{m,t} - R_{f,t}] + e_{i,t} \tag{3}$$

where α_i is the intercept of the regression model and $e_{i,t}$ is a zero-mean residual

As evident by equation 2, stocks with a high beta, i.e., stocks with high covariance with the market, are expected to generate higher returns than low beta stocks. Consequently, CAPM is an expression of a compensated risk premium; the higher the systematic risk of an asset, the higher the expected return must be. As CAPM assumes that all assets yield expected returns that are commensurate with their risk, all assets are "fairly priced" and hence, no alpha (intercept) exists in the CAPM. Any discrepancy in the market is assumed to be eliminated by competitive forces, leaving an alpha equal to 0.

CAPM relies on the efficiency of the market portfolio which is based on several non-viable assumptions. These include e.g., unrestricted risk-free borrowing and lending, as well as unrestricted short selling of risky assets

(see appendix A.1 for the full list of assumptions). As a result, subsequent research challenged the notion of CAPM which the following subsection unfolds.

The Fama/French Factor Models

While the CAPM has shaped the way financial academics and practitioners perceive the risk-return trade-off, the model has been heavily criticized for its inability to explain various market anomalies. That is, the market exposure does not capture all patterns in average returns, yielding a portfolio intercept (alpha) unequal to zero in equation 3.

Against this background, Fama and French (1992) rejected the CAPM based on their findings that the size and book-to-market ratio capture cross-sectional variation in average returns which the market factor cannot explain on its own. They found that realized average returns, ceteris paribus, have been higher for small firms relative to larger ones and for value firms relative to growth firms, historically. As a result, they introduced a three-factor model which better describes the expected return of a portfolio (Fama & French, 1993, 1996). The first factor reflects the excess return of the market portfolio ($R_m - R_f$), similarly to the CAPM. However, two additional components are included. The second factor incorporates the superior return on a (diversified) portfolio of small stocks relative to a portfolio of large stocks (small minus big, SMB). The third factor incorporates the superior return on a portfolio of high-book-to-market stocks relative to the return on a portfolio of low-book-to-market stock (high minus low, HML). The resulting three-factor model for portfolio *i* is given by the following:

$$E(R_i) - R_f = \beta_i[E(R_m) - R_f] + s_i E(SMB) + h_i E(HML) \quad (4)$$

where $E(R_M) - R_f$, E(SMB), and E(HML) are expected premiums. SMB is the difference between the average return on a portfolio of small stocks and a portfolio of big stocks, and HML is the difference between the average returns on a portfolio of high book-to-market stocks and a portfolio of low book-to-market stocks

The factor loadings, β_i, s_i, and h_i are the slopes in the time-series regression:

$$R_{i,t} - R_{f,t} = \alpha_i + \beta_i[R_m - R_f]_t + s_i SMB_t + h_i HML_t + e_{i,t} \quad (5)$$

However, scholars such as Novy-Marx (2013) and Titman, Wei, and Xie (2004) found evidence of incompleteness in the model as the three factors still missed some of the variations in average returns related to profitability and investment. Motivated by these studies, Fama and French (2015) added two additional factors to their model, creating a five-factor model. The fourth factor (robust minus weak, RMW) is the difference between the returns on stocks with robust and weak profitability, whereas the fifth factor (conservative minus aggressive, CMA) is the difference between the returns on stocks of low and high investment firms. The five-factor model can be expressed as:

$$E(R_i) - R_f = \beta_i[E(R_m) - R_f] + s_i E(SMB) + h_i E(HML) + r_i E(RMW) + c_i E(CMA) \quad (6)$$

where E(RMW) and E(CMA) are expected premiums. RMW is the difference between the average returns on a portfolio of the most profitable companies and a portfolio of the least profitable companies, and CMA is the difference between the average returns on a portfolio of firms that

invest conservatively and a portfolio of firms that invest aggressively.

While SMB, HML, RMW, and CMA may not themselves be obvious candidates for relevant risk factors, Fama and French (1993, 2015) argue that these factors are proxies for hard-to-measure fundamental determinants of risk. The argument is that the model captures the expected return effects of state variables without identifying them.

Relating these asset pricing models to our specific research, several momentum scholars have investigated how to enhance the performance of momentum strategies, accounting not only for market risk but also the SMB and HML factors (Hühn & Scholz, 2018, Blitz et al., 2011). The following section describes these enhanced momentum strategies.

3.4 Conventional and Enhanced Momentum Strategies

This section serves a threefold purpose. The first subsection defines the momentum effect which constitutes the building blocks for the conventional momentum strategy. The second subsection describes the performance of the conventional momentum strategy and unfolds the concept of momentum crashes. Building on the insights of these two subsections, the final subsection elaborates on four enhanced momentum strategies as well as the construction of the variables that the strategies rely on.

3.4.1 The Conventional Momentum Strategy

This thesis follows the definition of the conventional price momentum effect of Jegadeesh and Titman (1993). The momentum effect exists if stocks with returns above (below) the cross-sectional average over some past period, J, are expected to yield a return above (below) the cross-sectional average in the following period, K. This can be expressed mathematically as:

$$\mathrm{E}\left(\mathrm{R_{i,K}} - \overline{\mathrm{R}}_{\mathrm{K}} \middle| \mathrm{R_{i,J}} - \overline{\mathrm{R}}_{\mathrm{J}} > 0\right) > 0 \tag{7}$$

$$\mathrm{E}\left(\mathrm{R_{i,K}} - \overline{\mathrm{R}}_{\mathrm{K}} \middle| \mathrm{R_{i,J}} - \overline{\mathrm{R}}_{\mathrm{J}} < 0\right) < 0 \tag{8}$$

where $\mathrm{R_i}$ denotes the return of stock *i* and $\overline{\mathrm{R}}$ is the cross-sectional average return of the sample

Multiplying the return of stock *i* in excess of the cross-sectional average for the past and following period, will result in a value above 0 for both equations 7 and 8. Consequently, the equations can be combined as follows:

$$\mathrm{E}\left\{\left(\mathrm{R_{i,K}} - \overline{\mathrm{R}}_{\mathrm{K}}\right)\left(\mathrm{R_{i,J}} - \overline{\mathrm{R}}_{\mathrm{J}}\right)\right\} > 0 \tag{9}$$

Henceforth, when this thesis mentions the momentum effect, it refers to equation 9 being valid. When equation 9 is violated, stock returns reverse rather than follow a trend.

3.4.2 The Performance of the Conventional Momentum Strategy

The "zero-cost" momentum strategies documented by Jegadeesh and Titman (1993) constitute the foundation for the momentum strategies in the recent literature. In fact, the momentum effect became so predominant that Carhart (1997) proposed adding a momentum factor to the Fama/French three-factor model. Figure 3.1 visualizes the value of 1 dollar invested in the SMB and HML factor and the Fama/French momentum factor that ranks stocks based on the past 12 months' returns, skipping the most recent month, and holds this position for one month.

Figure 3.1: Cumulative returns of the SMB, HML, and momentum strategy from 1960-2020

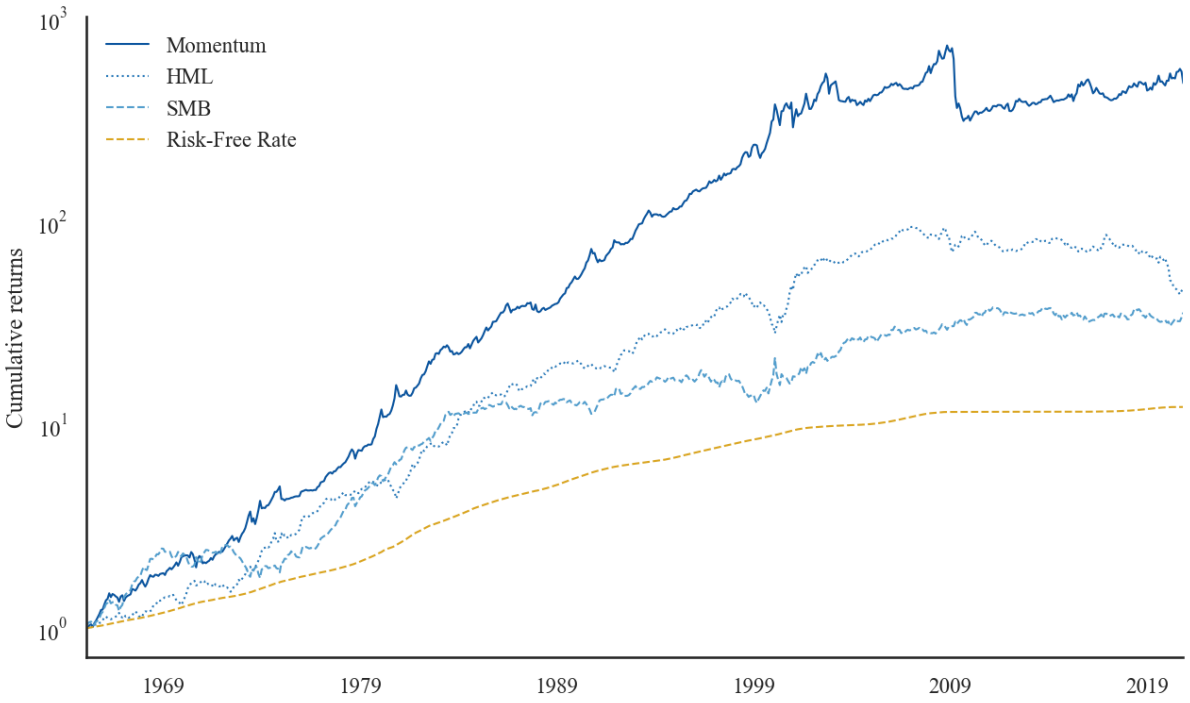


Figure 3.1 notes: The figure illustrates the total cumulative returns (including the risk-free rate) of the momentum, SMB, and HML portfolio on a log scale. The factor returns are based on the US equity market. The momentum portfolio is formed by ranking stocks on their prior 2-12 months returns. (Source: Own creation based on factor portfolios obtained from the Kenneth French Data Library).

Consistent with the findings of the existing literature, Figure 3.1 illustrates the superior performance of the momentum strategy and the presence of a strong momentum premium over the last century (Jegadeesh & Titman, 1993, 2001; Grinblatt, Titman & Wermers, 1995; Moskowitz & Grinnblatt, 1999; Israel & Moskowitz, 2013). The momentum portfolio substantially outperforms the SMB and HML portfolios. Despite the strong profitability of the momentum strategy, scholars have noted that the predictive power of returns seems to have diminished over time (Novy-Marx, 2012). As evident in Figure 3.1, a momentum strategy would have been less profitable if initiated at some point during the last two decades.

Moreover, Figure 3.1 illustrates the profit punctuations of the momentum strategy following the Great Depression in the 1930s and the 2008-2009 financial crisis (Daniel & Moskowitz, 2016). Recall that zero-cost

momentum strategies long past winners and short past losers. When the market outperforms Treasury bills, winner (loser) stocks tend to be stocks with betas greater (lower) than one. Therefore, in upstate markets, the momentum strategy places a positive beta bet on the market by longing high-beta stocks and shorting low-beta stocks. Conversely, in downstate markets, the momentum strategy places a negative beta bet on the market by longing low-beta stocks and shorting high-beta stocks (Grundy & Martin, 2001).

Figure 3.2: Market betas of winner and loser decile portfolios

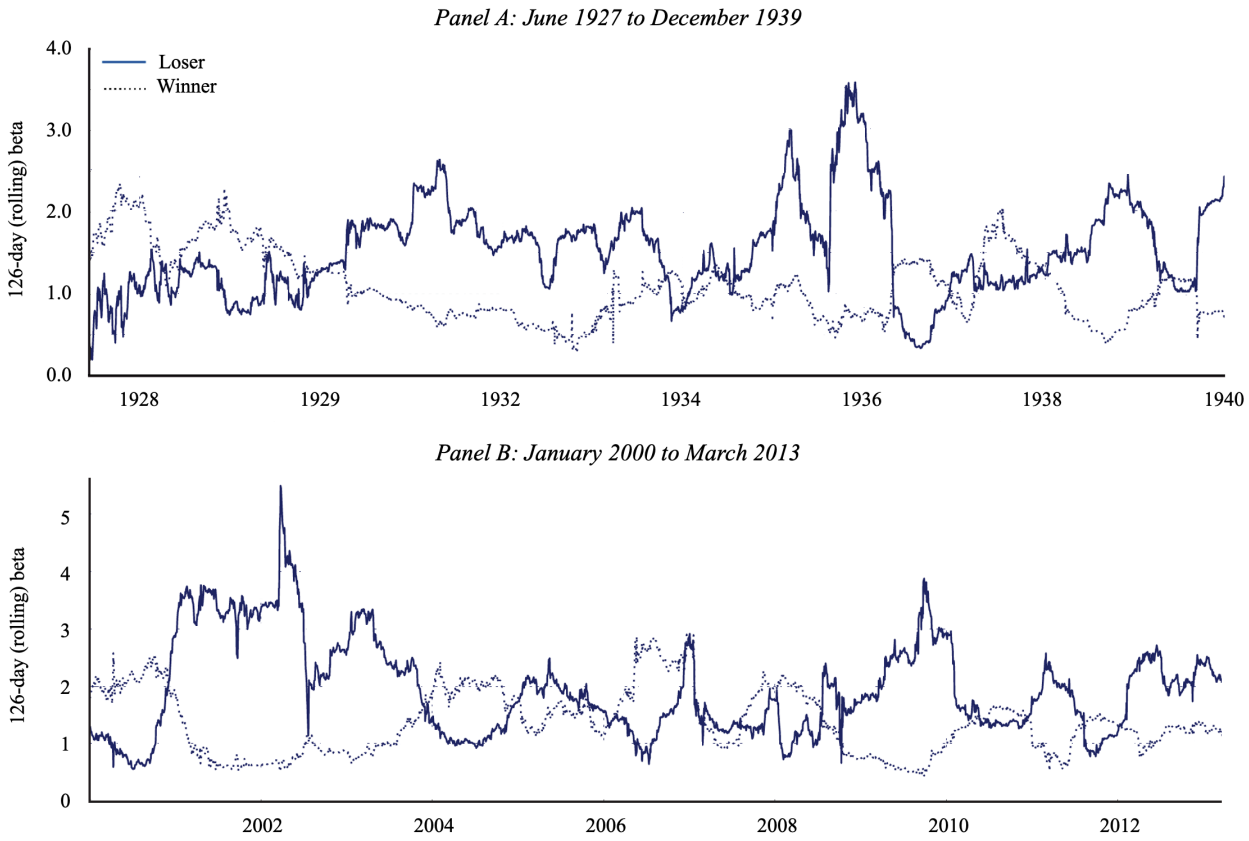


Figure 3.2 notes: The betas are estimated based on a 126-day rolling regressions of the excess returns of the momentum portfolio against the contemporaneous excess market return and ten (daily) lags of the market return, summing the betas (Source: Daniel & Moskowitz, 2016)

Figure 3.2 visualizes the time-varying market exposure (beta) of the winner and loser portfolio before, during, and after the momentum crashes of the Great Depression (panel A) and the 2008-2009 financial crisis (panel B), respectively. Notably, the beta of the loser portfolio increases dramatically during volatile periods. While the beta of the winner portfolio is rarely above 2, the beta of the loser portfolio reaches much higher levels of up to 5. Such a big difference in the betas, results in a large negative beta for the combined WML portfolio. A large negative beta of the WML portfolio coupled with a quick and dramatic market rebound lead the WML portfolio to experience huge losses. The crashes of the momentum strategy result in negative skewness in the return distribution that is highly unattractive from an investor perspective (Daniel & Moskowitz, 2016)[3].

[3] A return distribution is negatively skewed (left skewed) when the left tail of the distribution is longer or fatter than the right tail (see appendix A.2).

We note that Daniel & Moskowitz (2016) do not specify exact requirements for when a loss of the momentum strategy can be defined as a "momentum crash". Thus, we refer to "crashes" of the strategy only in periods that have been confirmed in the literature as "crash periods" (e.g., following the Great Depression, the dot-com bubble, and the 2008-2009 financial crisis).

The crash risk inherent to the momentum strategy has become a well-known phenomenon in the momentum literature. Several scholars have developed extensions of the momentum strategy attempting to enhance its performance which we describe in the following section.

3.4.3 Enhanced Momentum Strategies

The most predominant enhanced momentum strategies include constant and dynamic volatility-scaled momentum (Barroso & Santa-Clara, 2015; Daniel & Moskowitz, 2016) as well as idiosyncratic momentum (Blitz et al., 2011). Recently, an enhancement of the momentum strategy, referred to as alpha momentum, has also been introduced (Hühn & Scholz, 2018). The below section will describe the theoretical foundation underpinning these four strategies.

Constant and Dynamic Volatility-Scaled Momentum

The core idea of volatility scaling approaches is taking investment positions that scale inversely to the risk of the underlying asset (Asness, Frazzini, & Pedersen, 2012). In the literature of momentum, there are two prevalent volatility scaling methods. First, the constant volatility scaling approach of Barroso and Santa-Clara (2015), and second, the dynamic volatility scaling approach documented by Daniel and Moskowitz (2016).

Barroso and Santa-Clara (2015) estimate the risk of momentum in their study and find that it is highly predictable. Using the previous six months' realized volatility of daily returns as the variance forecast, they scale the long-short momentum portfolio, targeting a strategy with **constant volatility:**

$$R^{*}_{WML,t} = \frac{\sigma_{target}}{\hat{\sigma}_t} R_{WML,t} \tag{10}$$

where $R^{*}_{WML,t}$ is the monthly constant volatility scaled momentum return, $R_{WML,t}$ is the monthly returns of the conventional momentum strategy, and $\hat{\sigma}_t$ is the previous six months realized volatility used as the variance forecast, and

$$\hat{\sigma}^2_{WML,t} = 21 \sum_{j=0}^{125} R^2_{WML,d_{t-1}-j} \,/126 \tag{11}$$

where $R_{WML,d}$ is the daily return of the conventional momentum strategy

Daniel and Moskowitz (2016) extend the volatility scaling approach by additionally taking the forecasted momentum return and its variance into account, creating a **dynamic volatility-scaled** approach. In this approach, the first step is to forecast the returns of the momentum strategy using the following time-series regression, which depends on a market indicator and the realized volatility of the market:

$$\widetilde{R}_{WML,t} = \gamma_0 + \gamma_B I_{B,t-1} + \gamma_{\sigma_m^2} \widehat{\sigma}^2_{m,t-1} + \gamma_{int} I_{B,t-1} \widehat{\sigma}^2_{m,t-1} + \tilde{e}_t \quad (12)$$

where $I_{B,t-1}$ is the bear market indicator that equals one if the cumulative past two-year market return leading p to the formation date is negative (and zero otherwise), and $\widehat{\sigma}^2_{m,t-1}$ is the variance of the daily returns on the market, measured over the 126 days preceding the start of month t

The expected return is defined as the fitted values from the above regression with the dynamic momentum return given by:

$$R^*_{WML,t} = \frac{1}{2\lambda} * \frac{\tilde{\mu}_t}{\tilde{\sigma}^2_t} * R_{WML,t} \quad (13)$$

where $\tilde{\mu}_t = E(\widetilde{R}_{WML,t+1})$ and $\tilde{\sigma}^2_t = E\left(\widetilde{R}_{WML,t+1} - \mu_t\right)^2$ and
λ is a static scalar that scales the dynamic strategy to the full sample volatility of momentum return

Both volatility-scaled strategies almost eliminate the crash risk and approximately doubles the Sharpe ratio of the conventional momentum strategy (Barroso and Santa-Clara, 2015; Daniel and Moskowitz, 2016).

Idiosyncratic and Alpha Momentum

Two additional extensions of the conventional momentum strategy are idiosyncratic and alpha momentum, as described by Blitz et al. (2011, 2020)[4] and Hühn & Scholz (2018). In the spirit of Grundy and Martin (2001), both extensions of the conventional momentum strategy seek to account for its time-varying factor exposure. Accordingly, the two momentum adaptations rely on the Fama and French (1993) three-factor model (see equation 4). To minimize the effect of changes in factor realizations, Blitz et al. (2011, 2020) as well as Hühn and Scholz (2018), utilize only the part of the returns that is not dependent on the three factors.

Idiosyncratic momentum strategies focus on residual returns in the Fama/French three-factor regression model:

$$e_{i,t} = R_{i,t} - R_{f,t} - \alpha_i - \beta_i[R_m - R_f]_t - s_i SMB_t - h_i HML_t \quad (14)$$

The residual returns, $e_{i,t}$, are estimated over the past 36 months and are scaled using their volatilities so that

$$\text{Idiosyncratic Momentum}_{i,t} = \frac{\sum_{t-12}^{t-2} e_{i,t}}{\sqrt{\sum_{t-12}^{t-2} (e_{i,t} - \bar{e}_i)^2}} \quad (15)$$

Stocks are ranked based on the idiosyncratic momentum, defined as the 2-12 months volatility-scaled residuals (Blitz et al., 2020). Based on the idiosyncratic rank, they are placed into equal-weighted decile portfolios that are reformed monthly. A long position is taken in the highest decile portfolio whereas a short position is taken in the lowest decile portfolio.

[4] Other scholars also examine idiosyncratic momentum (Gutierrez & Prinsky, 2007; Chaves, 2016)

Alpha momentum relies on the same principle as idiosyncratic momentum but utilizes the alpha in the Fama/French three-factor model

$$\alpha_{i,t} = R_{i,t} - R_{f,t} - \beta_i[R_m - R_f]_t - s_i SMB_t - h_i HML_t - e_{i,t} \quad (16)$$

To find the alpha coefficient, Hühn & Scholz (2018) apply equation 4 on daily stock returns instead of the monthly returns used by Blitz et al. (2011, 2020). The regressions are estimated over a period of 12 months, excluding the most recent month. Similar to idiosyncratic momentum, Hühn & Scholz (2018) rank stocks based on past alphas and places them into decile portfolios with monthly rebalancing.

In summary, there are various ways to enhance the momentum strategy. The research of this thesis relies on the four enhanced momentum strategies presented in this subsection. Specifically, the four enhanced strategies constitute the foundation of the momentum variables that will be incorporated into the machine learning-based stock-selection strategies of this thesis. In this regard, we note that this thesis seeks to create stock-selection strategies that incorporate the momentum variables at the *individual* stock level. Hence, we do not scale the returns of the conventional momentum strategy, as done by Barroso and Santa-Clara (2015) and Daniel and Moskowitz (2016). Yet, their insights are implicitly deployed on an individual stock level. Section 5.3 elaborates on the implementation of the momentum variables.

3.5 Explanations of the Existence of Momentum

As the momentum effect is one of the most pervasive asset pricing anomalies documented in financial literature (Fama & French, 2012), many scholars have embarked on the journey of discovering *why* the momentum effect exists. This section provides an overview of the most predominant views on the existence of momentum. In this regard, we note that the research of this thesis does not examine possible explanations of the existence of momentum. Therefore, the following section outlines a brief overview of the most dominating views in this field.

Broadly speaking, explanations of the momentum effect can be divided into two perspectives, namely a rational and a behavioral perspective. The first perspective consists of rational thinkers that declare the momentum effect a manifestation of various risk premia (Conrad & Kaul, 1998; Berk, Green & Naik, 1999; Moskowitz & Grinblatt, 1999; Johnson, 2002; Bansal, Dittmar & Lundblad, 2005). However, the explanations provided by the rational perspective have been criticized and challenged by several studies providing contradicting evidence (e.g., Jegadeesh & Titman, 2002; Griffin, Ji & Martin, 2003). In fact, proponents of the behavioral perspective argue that the explanatory power of the rational perspective is due to overfitting bias and data mining (Singh & Walia, 2020). Thus, we do not unfold this perspective further.

The predominant perspective explaining the momentum effect is behavioral-based. This perspective interprets momentum as a market inefficiency, caused by irrational investor behavior. The behavioral perspective expands the seminal work by Tversky and Kahneman (1974) who found that people rely on heuristics when

they make judgments under uncertainty which can lead to biases. A variety of different behavioral biases exist that each can lead to stock price under- or overreactions.

Underreaction

Put simply, underreaction theories argue that stock prices underreact to news (Chan, Jegadeesh and Lakonishok, 1996; Barberis, Shleifer & Vishny, 1998; Hong & Stein 1999; Hong, Lim, & Stein, 2000). For instance, Barberis et al. (1998) argue that stock prices underreact to good news over horizons of 1-12 months. As a result, stock prices deviate from their fundamental value during this period. Because news is slowly incorporated into the stock prices, a predictable trend, and thus a momentum effect arises. However, once the stock price equals its fundamental value, there is no further predictability in the stock return (Barberis et al. 1998). Thus, if momentum is caused by an underreaction, the abnormal returns are followed by normal returns in the subsequent period (Jegadeesh & Titman, 2002).

Hong and Stein (1999) propose that the market consists of two types of agents, namely “news-watchers" and “momentum traders". The gradual diffusion of information amongst the news-watchers fosters an underreaction in the stock price. The underreaction creates a trend that momentum traders can exploit.

The underreactions in stock prices can be explained by various biases, such as conservatism (Barberis et al., 1998) and the disposition effect (Grinblatt & Han, 2005).

Overreaction

Several scholars argue that momentum profits can be explained by stock price overreactions (Barberis et al., 1998; Daniel, Hirshleifer & Subrahmanyam, 1998; Cooper, Gutierrez & Hameed, 2004). According to Barberis et al. (1998), overreaction evidence shows that over longer horizons of 3-5 years, stock prices overreact to consistent patterns of news pointing in the same direction. More specifically, stocks that have performed well for a long time tend to become overpriced, giving rise to a predictable trend and a momentum effect. Barberis et al. (1998) rely on what Tversky and Kahneman (1974) call representativeness heuristics in their explanation of stock price overreactions.

According to Daniel et al. (1998), biased self-attribution leads to stock price overreaction, fostering short-term momentum. Investors attribute good investment performance to their own skills and poor performance to bad luck. As a result, investors become overconfident about their ability to pick winner stocks, pushing the stock prices above the fundamental value of the stock. In this regard, Cooper et al. (2004) argue that the overconfidence stemming from self-attribution is magnified during upstate markets which leads to a stronger momentum effect.

The overreaction in prices is eventually corrected, as investors observe future news and realize their errors (Daniel et al., 1998). Thus, when momentum profits are caused by overreaction, the abnormal returns will be followed by negative returns, i.e., reversal (Jegadeesh & Titman, 2002).

Behavioral Explanations for the Momentum Crashes and the Enhanced Momentum Strategies

The above explanations of the momentum effect are centered around the conventional momentum strategy. In this regard, Daniel & Moskowitz note that momentum crashes are compatible with the behavioral explanation of Cooper et al. (2004) who argue that momentum profits depend on the state of the market. Following down-

state markets, risk aversion increases, as wealth has decreased, which leads to a smaller (delayed) overreaction. As a result, the momentum premium decreases. This is compatible with the momentum strategy performing particularly poorly during market rebounds.

Blitz et al. (2020) find that the relationship between conventional momentum and investor overreaction, as well as risk-based explanations, is much less profound for *idiosyncratic momentum*. Rather, their empirical findings are in line with the underreaction theories (Blitz et al., 2020). In addition, Hühn & Scholz (2018) suggest that while *conventional momentum* is predominantly driven by momentum trading that pushes prices up (overreaction), *alpha momentum* is more strongly related to underreaction to firm-specific news.

In summary, there are several explanations of the existence of the momentum effect, with the behavioral perspective proposing the most plausible explanations. Advocates of the behavioral perspective argue that the momentum effect can be explained by stock price under- and overreactions, caused by various behavioral biases. While consensus on specific behavioral biases and their implication on stock prices remains incomplete, the behavioral perspective reconciles by suggesting that active investment strategies can yield abnormal returns by exploiting market inefficiencies. Against this background, the behavioral perspective is contradicting the rational perspective and the notion of an efficient market.

4 Machine Learning

This chapter outlines the theoretical foundation for the machine learning models utilized in this thesis to create the stock-selection strategies. The first section introduces general concepts of machine learning. The two following sections provide a theoretical description of the machine learning models deployed in this thesis: Random Forest, the Multilayer Perceptron, and a Recurrent Neural Network with Long Short-Term Memory cells. The last section provides an overview of the key characteristics of the three models.

4.1 Introduction to Machine Learning

Before this section elaborates on the specific components of each machine learning model, we initiate with a brief introduction of general machine learning concepts. Machine learning lies at the intersection of computer science and statistics and revolves around building algorithms that automatically improve through experience (Mitchell, 1997). Compared with traditional statistical approaches, machine learning offers advantages to handle complex prediction problems (Mitchell, 1999). More specifically, machine learning algorithms can identify underlying patterns of complex data that humans and more simple models would struggle to uncover. These data patterns can be used to predict future events (Géron, 2019).
The acceleration in computing power over recent years, low cost of data storage, availability of big data, as well as the variety of open-source software has fostered a renaissance in the field of machine learning. Resonating Arnott et al (2019), the pace of transformation is striking. Machine learning as well as other statistical methods, that were impractical to use in the past, now hold considerable promise across various

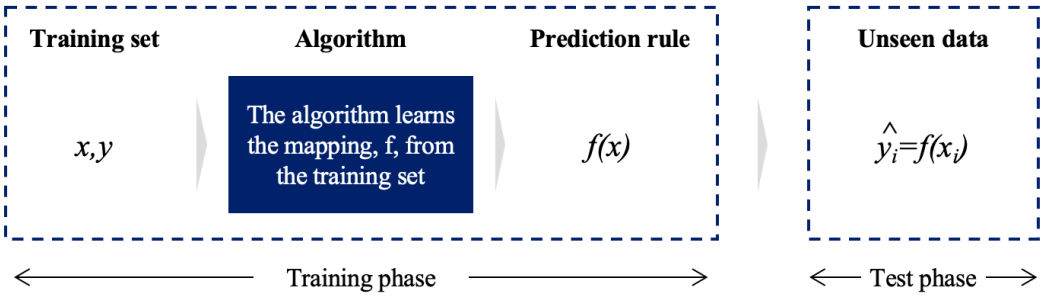
industries. Within the field of finance, Israel et al. (2020) argue that machine learning has the potential to become an important step forward for investment management, in which the most important task is return prediction.

4.1.1 Supervised Machine Learning

Supervised and unsupervised machine learning are two archetypes of machine learning. Unsupervised machine learning is a type of machine learning in which algorithms are used to extract knowledge of *input data* by identifying patterns in the data. The algorithm is solely fed input data, as there is no known output data, and thus the algorithm is said to "learn without a teacher" (Müller & Guido, 2016). However, the research of this thesis relies on supervised machine learning, and thus, unsupervised machine learning will not be further elaborated on.

Supervised machine learning is a type of machine learning where both input and output data are known, and thus, the output data are said to be "a teacher" from which the algorithm learns (Müller & Guido, 2016). Figure 4.1 depicts the process of a supervised machine learning algorithm in its simplest form. The algorithm is fed pairs of inputs and target outputs *(x,y)*, comprising the training set. The algorithm learns the mapping (the prediction rule, f(x)) between the inputs and outputs. Then, the algorithm relies on this prediction rule to make predictions of outputs for inputs it has never seen before ($\hat{y}_i = f(x_i)$).

Figure 4.1: The process of a supervised machine learning algorithm when making predictions



(Source: Own creation)

An example of a supervised machine learning problem is to predict future stock returns (the output variable, y) based on firm-specific news (the input variable, x). For a machine learning algorithm to be successful in predicting stock returns, it must have seen previous examples of how firm-specific news is related to stock returns. On the basis of these previous examples, the algorithm makes a prediction rule that it relies on when predicting future stock returns for a piece of firm-specific news.

Though supervised machine learning requires human effort to provide the input-output pairs of the training set, it enables the tedious task of finding the future output, for each given input, to be automated.

4.1.2 Classification and Regression Problems

There are two major categories of supervised machine learning problems, namely regression and classification problems. The primary difference between the two categories is the output variable, Y. In brief, classification problems have discrete categorical outputs, $y_i \in Y = \{0,1,2,\dots,C\}$, where C is the number of classes. The goal for classification problems is to predict a discrete class label (category), y_i, from a list of predefined class labels. The categorical output is based on the likelihood of the observation belonging to the respective category. An example of a (binary) classification problem is predicting whether a stock price will rise or fall (Müller & Guido, 2016).

In contrast, the task of a regression problem is to predict continuous and numerical outputs $y_i \in \mathbb{R}$ based on a given set of input variables, x_i (Müller & Guido, 2016). As this thesis applies machine learning models for predicting future excess stock returns, it revolves around a regression problem. A popular method to assess how well the predicted numerical outputs match the actual (target) output is by use of a *loss function.* This loss function is (often) given by the mean squared error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{17}$$

where $\hat{y}_i$ is the predicted output based on the input variables, y_i is the target output, and n is the number of predictions.

If the machine learning algorithm does well in capturing the relationship between the input and the output data, the predicted outputs will be similar to the target outputs, and thus MSE will be low (Géron, 2019).

4.1.3 Training and Optimizing Machine Learning Models

Training and Test Data

This section outlines how supervised machine learning models are trained and tested. To obtain a successful machine learning model, the complete dataset is split into two parts, namely a training set and a test set. First, the training set is fed to the algorithm for it to learn the relationship between the inputs and outputs. Second, the algorithm is fed unseen inputs of the test set for it to make predictions of the corresponding outputs. The test dataset is used to measure the performance of the model, by comparing the predicted outputs with the target outputs. The ability of the model to "generalize" refers to its ability to adapt to new and previously unseen test data. If the algorithm makes accurate predictions on the training set, and the distribution of the training and test data are similar, the model is expected to make accurate predictions on the test dataset as well (Müller & Guido, 2016).

Overfitting and Underfitting

If a machine learning model is complex enough, it will always be able to make accurate predictions on the training set. However, highly complex models fit the individual observations, noise, and particularities of the training data too closely, which results in poor performance on the test set. Such a model is said to overfit and

has high variance error as well as low generalizability. Overfitting poses a major issue when attempting to extract signals from noisy historical data. In contrast, models that are too simple cannot capture the patterns and variability in the training data, which results in poor performance on both the training and test dataset. Such a model is said to underfit and is associated with a high bias error (Müller & Guido, 2016).
The dilemma between choosing a complex model that may overfit and a simple model that may underfit is known as the bias-variance trade-off. The optimal model with the best ability to generalize to new data lies in a sweet spot between being too simple and too complex (Müller & Guido, 2016), as illustrated in Figure 4.2**:**

Figure 4.2: Bias Variance Trade-off (overfitting and underfitting)

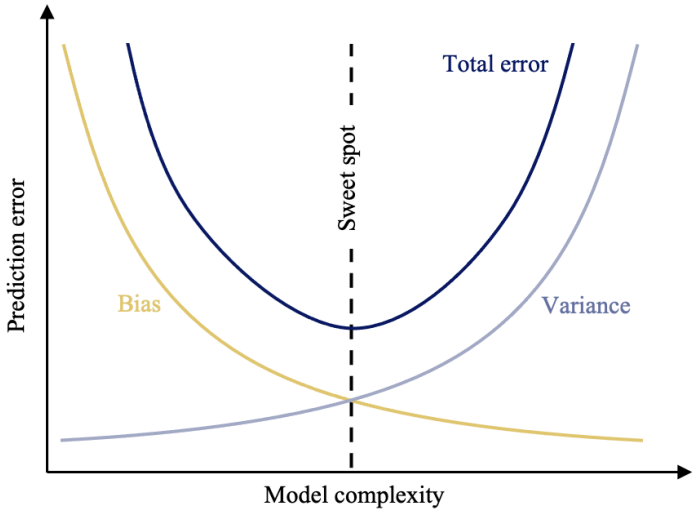


Figure 4.2 notes: The figure illustrates the sweet spot of the bias-variance trade-off which minimizes the total error (the loss). In practice, the prediction error cannot fully be eliminated. Thus, we do not model a prediction error equal to 0. (Source: Own creation based on Goodfellow, Bengio, & Courville, 2017).

The models we deploy, especially neural networks, are complex models that are often prone to overfitting. To avoid overfitting and approach the sweet spot, we deploy several regularization techniques which we describe in section 7.4.

Hyperparameter Optimization

Most machine learning algorithms have various "settings" that control the behavior of the algorithm. These settings are referred to as hyperparameters or the external characteristics of the model. The hyperparameters are not updated by the machine learning algorithm itself but set externally by the machine learning scientist (Goodfellow, Bengio, & Courville, 2017). To avoid that the model under- or overfits the data, the fit of the model can be adjusted through its hyperparameters. The hyperparameters are crucial for the performance of the model as they control model complexity. The search for optimal hyperparameters is an iterative process where models with different combinations of hyperparameters are trained and tested. The hyperparameters that minimize the loss function is chosen. In mathematical terms, hyperparameter optimization can be expressed as:

$$x^* = \underset{x \epsilon \chi}{\operatorname{argmin}} f(x) \tag{18}$$

where $f(x)$ is a loss function, such as MSE, to be minimized, x is a given set of hyperparameters that can take any value in the domain χ, and x^* is the set of optimized hyperparameters that minimize the objective function.

Note that different machine learning algorithms have distinct hyperparameters. We address the specific hyperparameters of the machine learning models utilized in this thesis in section 7.4.

Validation Set

There are various ways to optimize hyperparameters. One option is to train various versions of a model with different hyperparameters and then test the versions directly on the test set. The optimal model is the model that exhibits the best performance on the test set, given its hyperparameters. However, as the hyperparameters are specifically tuned on the test set, the model might overfit the test data, and as a result, the model is likely to exhibit inferior performance on unseen data (Géron, 2019).

An option that overcomes this problem is to hold-out a part of the training set for validation purposes. The held-out part of the training set is often referred to as the validation set. When utilizing a validation set, various versions of the model with different hyperparameters are trained on the reduced training set[5] and tested on the validation set – instead of on the test set, directly (Goodfellow et al., 2017). The optimal model is the one that performs best on the validation set given its hyperparameters. This model is then trained on the full training set (including the validation set) and subsequently tested on the true out-of-sample test set.

Thus, the key purpose of introducing a validation set is to simulate the out-of-sample performance of the model before presenting the test data to the model. This avoids overfitting the test data and consequently, it provides a more realistic estimate of the model's ability to generalize (Goodfellow et al., 2017).

Cross-Validation

Splitting the training set into a reduced training set and a validation set gives rise to another challenge; selecting the size of the held-out validation set. A validation set that is too small results in imprecise model evaluations. In contrast, a validation set that is too big leaves a highly reduced training set, which might hinder the model in learning patterns in the data. The best model for a highly reduced training set might not be the best model for the full training set. A solution to this dilemma is to use k-fold cross-validation, where the training set is split repeatedly into k folds, resulting in k small validation sets. A visual representation of an example with five-fold cross-validation is provided in Figure 4.3:

[5] The training set minus the validation set

Figure 4.3: Cross-validation for hyperparameter optimization

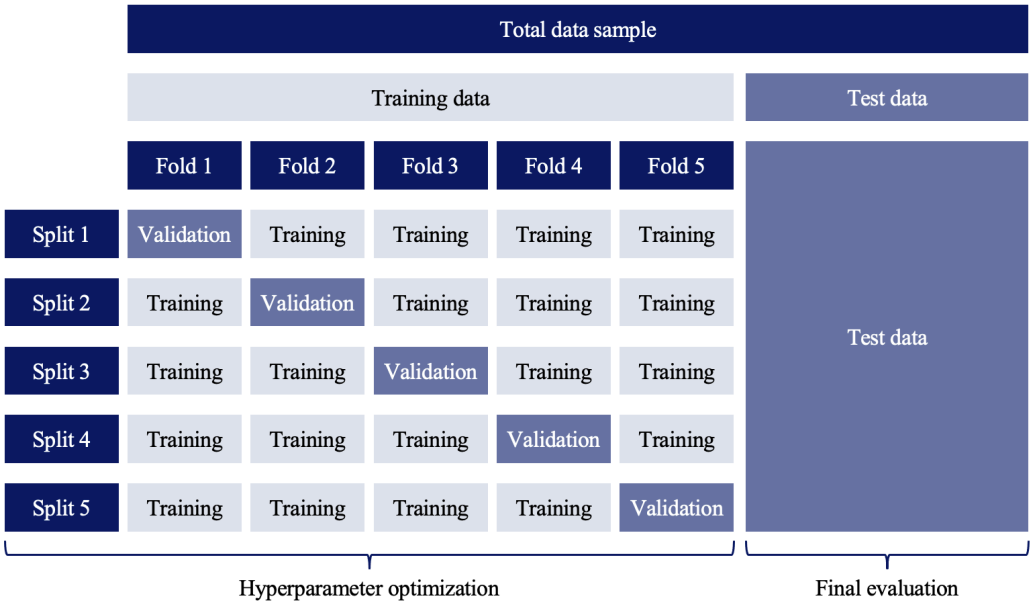


Figure 4.3 notes: The figure depicts a process of five-fold cross-validation. First, the overall data sample is split into a training and test data sample. The training sample is then split repeatedly into five folds, resulting in five small validation sets. The optimal model is defined as the model with the hyperparameters that minimize the loss function across all five validation sets. The optimal model is applied to the test data for final evaluation. (Source: Own creation).

One at a time, each of the five folds is left out of the training data and used as a validation set[6]. The optimal hyperparameters are found by minimizing the loss function across all five validations sets. Subsequently, the model with the optimal hyperparameters is applied to the true out-of-sample test set for final evaluation. K-fold cross-validation offers a more realistic estimate of the model's ability to generalize by averaging model performance for each k validation set (Müller & Guido, 2016). Unfortunately, there is also a drawback of using cross-validation. First, the method is time-consuming as the model must be trained k times across the training set. Second, it is not well suited for time-series data which this thesis relies on. Consequently, we deploy a time-series adaptation of k-fold cross-validation which we describe in section 7.3.

4.1.4 Gradient Descent

Having established how hyperparameters are optimized, we now turn to the optimization process of the *internal parameters* (or simply, parameters) of machine learning models. Parameters are internal configurations of the model that are not set manually by the machine learning scientist (in contrast to hyperparameters). In the field of machine learning, gradient descent is used across many different models to update the parameters of the models. Two of the three machine learning models deployed in this thesis also rely on gradient descent (the neural networks). Consequently, this section briefly describes the intuition behind the gradient descent algorithm while subsections 4.3 describe how the algorithm is applied for our models.

[6] This corresponds to repeating the method of creating a held-out validation set k times

Gradient descent is a first-order optimization algorithm for finding the minimum of a differentiable function. The algorithm is often deployed to minimize a loss function. The algorithm minimizes the loss by measuring the local gradient (derivative) of the loss function for a given set of parameters, θ, and taking a step (learning step) towards the descending gradient. The closer the gradient is to zero, the closer the loss function is to its minimum as illustrated in Figure 4.4 (Deisenroth, Faisal & Ong, 2020).

Figure 4.4: Gradient descent

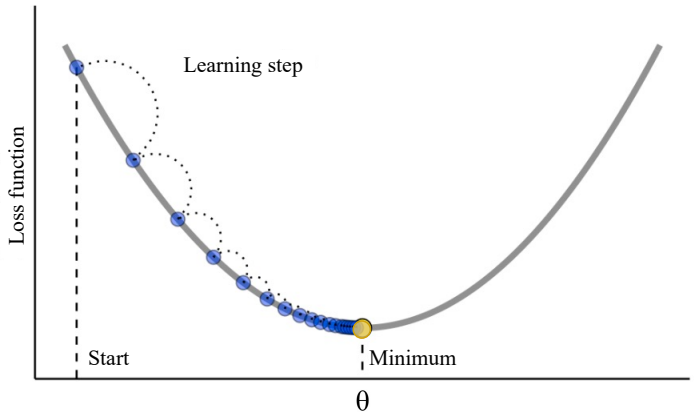


Figure 4.4 notes: The figure depicts the loss function with respect to the parameters θ. The gradient descent algorithm starts to the left and takes learning steps towards the descending gradient in order to minimize the loss function (Source: Own creation based on Géron, 2019).

The size of the learning step is an important parameter in gradient descent and is determined by the learning rate. With a learning rate that is too big, the algorithm might jump over the minimum of the loss function. In contrast, with a learning rate that is too small, the algorithm will require many training iterations to find the minimum of loss function which is very time-consuming (Deisenroth et al., 2020). Figure 4.5 depicts the possible result of a learning rate that is either too big (panel A) or too small (panel B).

Figure 4.5: Size of learning steps in gradient descent

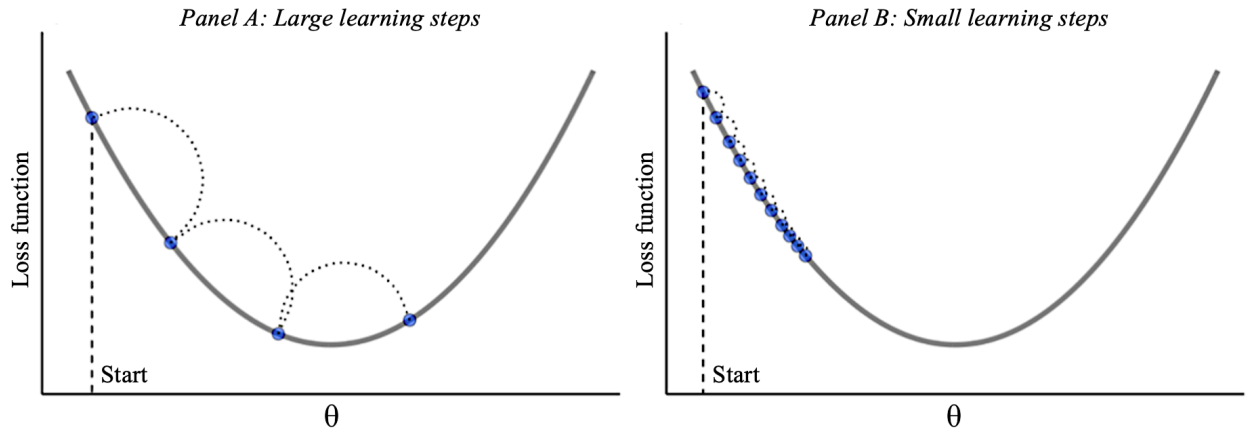


Figure 4.5 notes: The figure illustrates the learning steps of a gradient descent algorithm that minimizes the loss function. Panel A depicts learning steps that are too big and consequently, the gradient descent algorithm skips the minimum of the loss function. Panel

B depicts learning steps that are too small and consequently, the gradient descent algorithm is very slow in reaching the minimum of the loss function. (Source: Own creation based on Géron, 2019).

Gradient descent can be performed on all differentiable (non-linear) loss functions (Hastie, Tibshirani & Friedman, 2009). However, some loss functions are not convex and might have local minima or plateaus, implying that there is a risk of getting stuck in a local minimum rather than the global minimum, as illustrated in Figure 4.6.

Figure 4.6: Loss function with local and global minimum as well as plateau

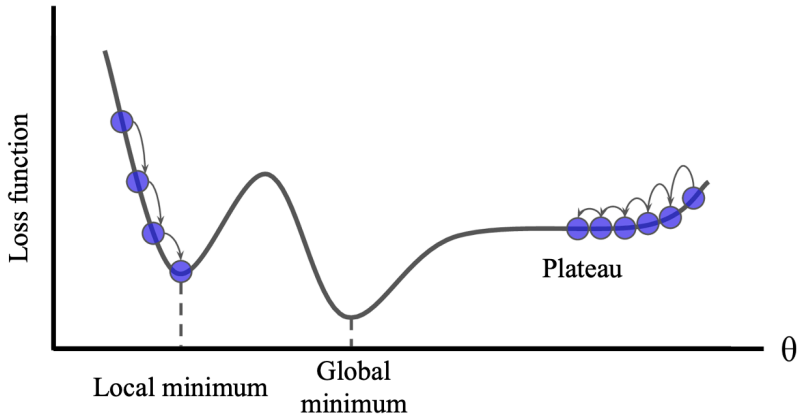


Figure 4.6 notes: The figure illustrates the learning steps of a gradient descent algorithm that minimizes a non-convex loss function. There may be ridges, plateaus, or other irregular terrains in the loss function, making convergence to the minimum difficult. For instance, if the algorithm initiates on the left, it will converge to a local minimum and never reach the global minimum. (Source: Own creation based on Géron, 2019).

Stochastic gradient descent addresses this issue by replacing the actual gradient (calculated from the entire dataset) with an *estimate* of the gradient (calculated from a randomly selected subset of the dataset). While this randomness might prevent the algorithm from identifying the exact global minimum it can help the algorithm escape local minima and get sufficiently close to the global minimum (Deisenroth et al., 2020).

Having established the fundamentals of machine learning and the process of constructing machine learning models, the following subsections turn to a more in-depth description of the models deployed in this thesis. We aim to provide a sufficiently in-depth description of the machine learning models, without dwelling on the specific programming details.

4.2 Tree-Based Models

This section is composed of three subsections. The first subsection unfolds the terminology and intuition behind tree-based machine learning models. The second subsection introduces regression trees. Finally, the third subsection presents the specific tree-based model used in this thesis, namely Random Forest.

4.2.1 Introduction to Tree-Based Models

The concept of tree-based models was introduced by Hunt, Marin, & Stone (1966). In general, a tree-based model is designed to find groups of observations that are similar to each other. The structure of the model is flow-based and resembles that of a tree, as is shown in Figure 4.7. The top of the tree is called the "root node". In the below example, the root node is divided into two new nodes (internal nodes). Those nodes are further split into either internal nodes or leaf nodes (leaves). Leaves are the end nodes that do not further split (Géron, 2019).

Figure 4.7: Structure of a tree-based model

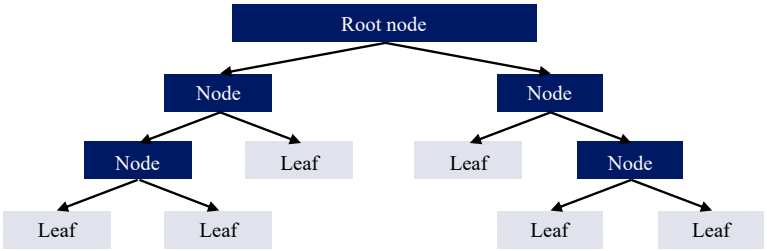


Figure 4.7 notes: The figure illustrates a simple architecture of a tree-based model with four nodes and six leaves. (Source: Own creation).

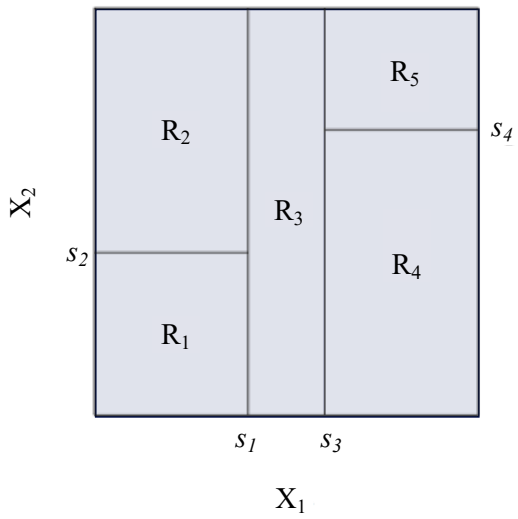
While various decision tree algorithms exist, we focus on the CART algorithm which is the most predominant decision tree algorithm (Breiman, Friedman, Stone, & Olshen, 1984). As this thesis revolves around predicting stock returns, *regression* trees are the focus of the following subsection.

Tree-based algorithms split the training dataset into a set of regions. The objective of the algorithms is to find the optimal splits of the training dataset that minimize the loss function. Specifically, the algorithm fits a simple model (e.g., a constant) in each region.

Assume a regression problem with a continuous output variable, Y, and inputs, X_1 and X_2, that each takes values in unit intervals. The splitting process starts by splitting the training set into two regions. Then, each of these two regions is split into subregions, and each of these subregions is split into sub-subregions, recursively, until some stopping rule is applied or the algorithm is unable to find a split that reduces the loss function further. To better understand this process, a visual representation of an example of the regression tree method is depicted in Figure 4.8[7]:

[7] For simplicity, we restrict attention to recursive binary splits in this explanation

Figure 4.8: The splitting process of the tree-based method



(Source: Own creation based on Hastie et al., 2009)

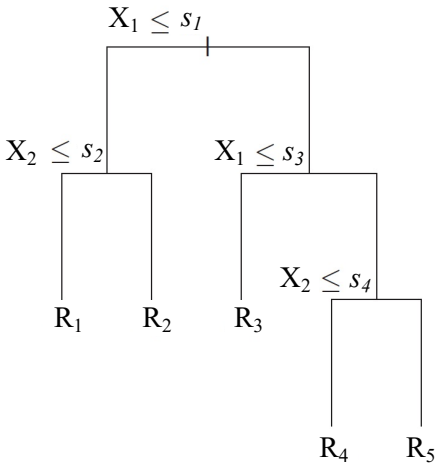
Let *s* denote the threshold that splits the inputs (the splitting point). First, X_1 is split at $X_1 = s_1$. Next, region "$X_1 \leq s_1$" is split at $X_2 = s_2$, and region "$X_1 > s_1$" is split at $X_1 = s_3$. Last, region "$X_1 > s_3$" is split at $X_2 = s_4$. The process results in a partition into five regions $(R_1, R_2, \dots, R_5)$. The model then fits a constant in each region.

The corresponding regression model predicting the output $\hat{y}$ with a constant c_m in region Rm can be formalized as:

$$\hat{y} = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\} \tag{19}$$

This model can also be represented as a binary tree, depicted in Figure 4.9 (Hastie et al., 2009). The complete training dataset is at the top of the tree. At each node, observations that satisfy the threshold are delegated to the left branch, while the remaining observations are delegated to the right branch. The leaves of the tree correspond to the five regions in Figure 4.8 (Hastie et al., 2009).

Figure 4.9: Binary regression tree



(Source: Own creation based on Hastie et al., 2009).

4.2.2 Mathematical Foundation for Regression Trees

This subsection explains the process of splitting data into regions for a single regression tree, such as the one in the above example, i.e., how the tree is "grown". Assume that the data consists of *p* input variables and one output variable for each of the *N* observations, i.e., (x_i, y_i) for $i = (1, 2, \dots, N)$, with $x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$. For simplicity, we apply *sum of squared errors,* $\sum(y_i - \hat{y}_i)^2$, as the loss function in this example.

Recall that the objective of the algorithms is to find the optimal splits of the training dataset that minimize the loss function. The algorithm needs to decide on the variables that are split (splitting variables), the points where the splitting variables should be split (splitting points), and the shape of the tree. To decide on these three aspects, assume that the training set is split into M regions, $R_1, R_2, \dots, R_M$ and the response is modeled as a constant c_m in each region:

$$y = \sum_{m=1}^{M} c_m I\{(X_1, X_2) \in R_m\} \tag{20}$$

Note that equation 20 generalizes the example in equation 19 but uses the *target* output to optimize c_m.

If the sum of squares $\sum(y_i - \hat{y}_i)^2$ is the minimization criterion, the best $\hat{c}_m$ simply becomes the average of the y_i in the respective region R_m:

$$\hat{c}_m = avg(y_i | x_i \in R_m) \tag{21}$$

The process of growing a complete tree that minimizes the loss function is computationally infeasible[8] if all the best splits need to be found at the same time. Consequently, the CART algorithm first searches for the optimal split at the top node level, and then finds the optimal split for the next level, given the top-level split. This splitting process is continued for all subsequent levels[9]. Starting with all the training data, consider a splitting variable, j, and split point, s. Define the pair of regions as:

$$R_1(j,s) = \{X|X_j \leq s\} \text{ and } R_2(j,s) = \{X|X_j > s\} \tag{22}$$

To find the splitting variable j and split point s that minimizes the loss function, the following must be solved:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i - c_2)^2\right] \tag{23}$$

For each combination of splitting variable j and split point s, the inner minimizations are solved by:

$$\hat{c}_1 = \text{avg}(y_i|x_i \in R_1(j,s)) \text{ and } \hat{c}_2 = \text{avg}(y_i|x_i \in R_2(j,s)) \tag{24}$$

Finding the best pair (j, s) is now feasible by scanning through the input variables. This best pair is used to split the training data into two regions. Then, the splitting process is repeated for each of those two regions and subsequently repeated for the subregions, recursively. This splitting process continues until the tree has grown to its maximum depth, or the algorithm is unable to find a split that reduces the loss function (Hastie et al., 2009).

The process described in this section is for one regression tree only. Individual trees are infamous for being associated with high variance error, i.e., overfitting the training data. The deeper the trees, the more specific behavior of the dataset they describe which results in predictions with high variance. Due to the hierarchical nature of the splitting process, the effect of an error in the first split is propagated through all the subsequent splits. As a result, individual trees tend to perform poorly on out-of-sample data. However, the limitations of individual trees can be mitigated by combining multiple trees into one ensemble model (Géron, 2019).

4.2.3 Random Forest

The ensemble method is a technique that combines the predictions from multiple algorithms to obtain better predictions than the ones provided by any of the individual models. A model consisting of multiple models is called an ensemble model. An ensemble model relies on the "wisdom of the crowd", in which multiple models

[8] Finding the optimal tree is known to be an NP-Complete problem

[9] The algorithm focuses on selecting the best split for one given level at a time. As the algorithm does not check whether the split at a given level is the split that minimizes the loss function *several levels down*, the solution is seldom optimal, but still reasonably good. (Géron, 2019)

protect each other from their individual weaknesses (Suthaharan, 2016). Ensemble models can consist of either multiple different algorithms or multiple of the same algorithms.

Random forest is an ensemble method that combines the predictions from a collection of trees, as illustrated in Figure 4.10. The idea of Random Forest is to build many trees that overfit individually in different ways but to reduce the amount of overfitting by averaging their predictions (Müller & Guido, 2016). Random Forest uses a variation of the bootstrap aggregation (bagging) technique to combine a collection of trees into one ensemble model (Breiman, 2001).

Figure 4.10: Random Forest prediction process

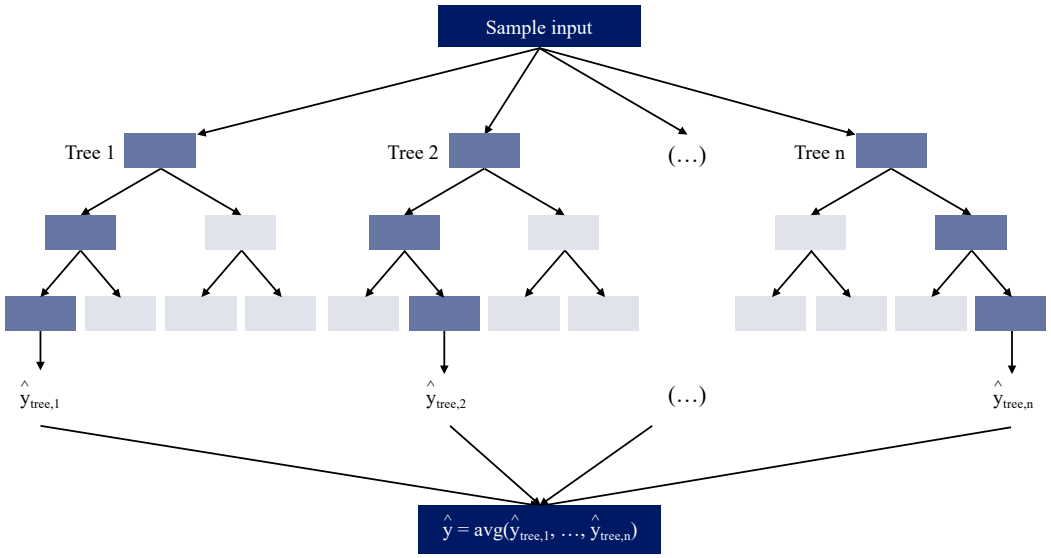


Figure 4.10 notes: The figure depicts the process of making predictions for a Random Forest model. The inputs are passed to the individual regression trees that each predict the corresponding target output. The final prediction made by the Random Forest model is found by averaging the predictions made by all the individual trees. (Source: Own creation).

The bagging technique fits the same regression tree to different subsamples (bootstrap samples) of the training data and averages the predictions of each sample. Note that each tree generated in bagging is identically distributed, and as a result, the expectations of an average of all trees are assumed to be equivalent to the expectation of any one of the trees individually. This implies that the bias of the bagged trees is equivalent to that of the individual trees. Thus, the only way to improve the prediction of the ensemble model is through variance reduction (Hastie et al., 2009).

Random Forests use a modification of the bagging technique that is specifically designed to reduce variance by de-correlating the trees across bootstrap samples (Gu et al., 2019). The de-correlation of trees is achieved by only choosing a random subset, m, of the input variables, p, to be split. Consequently, the early branches of each tree are split based on different input variables (splitting variables), which lowers the average correlation of predictions. Each individual predictor will not fit the training data as well (i.e., it has a higher bias error) as if it was trained on the original training set. However, the aggregation of predictions reduces both bias and variance. The result is that the ensemble model has a similar bias but a lower variance than a

single predictor (Hastie et al., 2009).

In mathematical terms, the Random Forest algorithm works as follows:
1. For b = 1 to B:
   a) Generate bootstrap samples Z* of size N from the training dataset $\{(x_i, y_{i,m})\}_{i=1}^{n}$
   b) Grow a Random Forest tree T_b on the bootstrapped data by recursively repeating the below three steps for each bottom node of the tree, until the leaf is reached
      i) Use a random subsample of m out of the p variables
      ii) Select the best variable/split-point among the m variables
      iii) Split a given node into two subnodes
2. The output of the resulting ensemble b^{th} tree is $\{T_b\}_1^B$

To make a prediction given a new input, x, the Random Forest model takes the average of the outputs of all the trees, so that:

$$\hat{y}_{rf}^{B} = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{25}$$

Having established the architecture of the Random Forest model, we proceed with the two additional models applied in this thesis. For the convenience of the reader, we note that an overview of the Random Forest model is presented in section 4.4.

4.3 Artificial Neural Networks

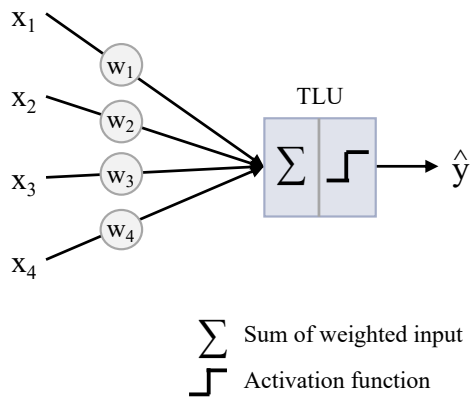
The neural networks leveraged in this thesis are a Multilayer Perceptron (MLP) and a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM). Before unfolding the theory behind these networks, it is necessary to understand the building blocks of neural networks. Accordingly, this section first elaborates on the basic characteristics of neural networks, after which an explanation of the specific architecture of MLP and RNN will be presented.

4.3.1 Introduction to Neural Networks and the Perceptron

In general, neural networks are often explained with an analogy to the human brain - a network composed of biological neurons that receive and pass signals to other neurons via the connections between them. This is what allows us, humans, to continuously learn from and adapt to our dynamic surroundings (Dawani, 2020). In a similar vein, a neural network consists of connected *artificial neurons* (often referred to as nodes) which are the building blocks of the network and enable it to "learn". A neural network typically has multiple nodes that are organized in rows. These rows are referred to as layers. Each node is a mathematical operation that takes an input, multiplies it by a specified weight, and passes an output (Géron, 2019). Note that the weight in a neural network is not a hyperparameter but a parameter that is updated internally by the network.

One of the first and simplest neural network architectures is the *Perceptron*, invented in 1958 by Frank Rosenblatt. It consists of only two layers of nodes, namely the input and the output layer. These layers are connected through one artificial node, which is referred to as a "threshold logic unit" (TLU). Figure 4.11 shows an example of the Perceptron taking four inputs and passing one output:

Figure 4.11: The perceptron



(Source: Own creation based on Géron, 2019)

The input layer takes the raw input and passes it inside the network. The Perceptron then applies a specific weight to each of the input connections before it is passed to the TLU. The TLU then transforms the weighted connections into a weighted sum, z, which can be articulated mathematically as follows:

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n \tag{26}$$

The sum of the weighted inputs is finally passed through an activation function that transforms the weighted inputs before passing it to the output node. The activation function in the preliminary Perceptron was a "heaviside step function" that computes a binary output based on a linear combination of the inputs. If the results exceed a certain threshold (typically 0), the model predicts a positive class. Otherwise, the prediction is a negative class (Géron, 2019). For a threshold of 0 it can be expressed as:

$$y_i = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} z_i \;\; x \geq 0 \\ 0 & \text{otherwise} \end{cases}. \tag{27}$$

Training of the Perceptron

For the Perceptron to learn and compute outputs that are accurate predictions, the TLU needs to be "trained". Training a TLU means finding the values for each weight of the input connections that minimize the prediction error (loss) of the network. The weights are initially picked randomly by the network but then adjusted towards the optimal values of the weights. This is done by feeding the Perceptron one training instance (i.e., one subsample of the training dataset) at a time and letting the network make predictions each time. Note that a greater weight of a certain input results in a greater impact on the respective prediction. Therefore, the

adjustment occurs by strengthening the weights of the inputs that would have contributed to the correct prediction, relative to the inputs that would have contributed to the wrong prediction. How much to adjust the weights in response to the prediction error depends on the learning rate. The learning rate controls how quickly the weights are adjusted. Mathematically it can be articulated as:

$$w_{i,o}^{new} = w_{i,o} + \eta(y_o - \hat{y}_o)x_i \quad (28)$$

where $w_{i,o}$ is the weight between input node *i* and output node *o*, x_i is the input variable of the current training subsample, $\hat{y}_o$ is the output of the output node *o* for the current training subsample, y_o is the target output of node *o* for the current training subsample, and η is the learning rate.

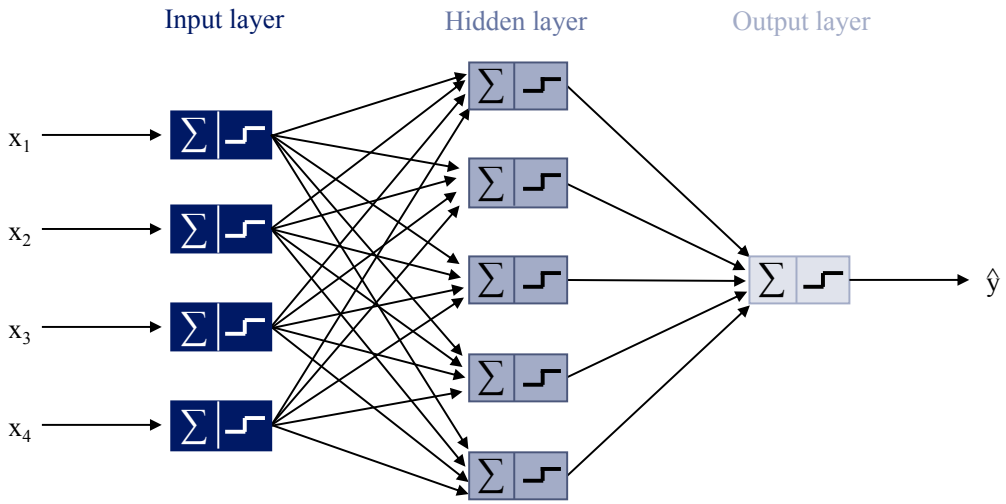
Rosenblatt (1958) proved that the Perceptron will converge to a solution if the training instances passed to the algorithm are *linearly* separable[10]. However, because the preliminary Perceptron is a linear and binary classifier, it is unable to solve nonlinear problems. Against this background, Perceptrons with multiple TLUs have been introduced, enabling a neural network to solve non-linear problems (Dawani, 2020).

4.3.2 The Multilayer Perceptron

A Perceptron with multiple TLUs is called a Multilayer Perceptron (MLP). It consists of multiple nodes combined in "layers" that are fully connected to one another. This includes the input layer, one or more hidden layers of nodes, and a final output layer of nodes which outputs a prediction of the target value. Every layer, except the output layer, includes a bias node that computes a constant which enhances the ability of the network to best fit the given data. The signal in the MLP is computed sequentially from inputs to outputs and flows in only one direction. This is also called a Feed-Forward Neural Network (Dawani, 2020).
Figure 4.12 visualizes an MLP with four input nodes, a hidden layer with five nodes, and one output node (for simplicity, we do not visualize the bias terms). The layers in this figure are fully connected (so-called 'dense layers'), meaning that the input nodes are connected to *all* nodes in the hidden layer which in turn are connected to all nodes in the output layer. More advanced MLPs have several hidden layers and are referred to as deep neural networks.

[10] This is referred to as the Perceptron convergence theorem, see appendix A.3 for visualization. Specifically, the Perceptron calculates the distance of the hyperplane (the decision boundary) from the points to be classified and adjusts itself to find the best position, so that it can perfectly linearly classify the two target classes

Figure 4.12: Multilayer Perceptron



(Source: Own creation based on Géron, 2019)

MLP is trained through an iterative process in order to minimize its loss function. This involves two stages, namely calculating the gradients of the error function with respect to the weights and subsequently using the gradients to compute the adjustments to be made to the weights, i.e., using gradient descent (Bishop, 2006). The two stages are elaborated upon below. Note that we, for simplicity, focus only on how weights are updated in the network during training, even though other internal parameters (e.g., biases) are also updated via the two stages.

Stage 1: Compute the Gradient Through Backpropagation

The first stage is performed using backpropagation, a method introduced by Rumelhart, Hinton, and Williams (1986). In training an MLP, the network is exposed to one data sample at a time. Initially, the model makes a *forward pass* in the network: The input nodes pass the data forward to the nodes in the first hidden layer that continues to pass the data forward to each consecutive layer until it reaches the last layer in which the final output is computed. The outputs of the input and hidden layers are transformed using an activation function (activation outputs) before entering as inputs to the subsequent layer. For a given node *j* and layer *l*, the input into the node is then given by

$$z_j = \sum_{k=1}^{n} w_{j,k} a_k \tag{29}$$

where k denotes the [1, ... , n] nodes in the *previous* layer, *l-1*, w_{jk} is the weight connection between node j in the current layer and k, and a_k denotes the activation output from the previous layer for a given node k.

The activation output of node j in the current layer can then simply be expressed as:

$$a_j = g(z_j) \tag{30}$$

where g(z) denotes some applied activation function

When the network reaches the output layer it computes the prediction and compares it to the target output. The loss is calculated via a loss function (MSE) which is given by

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{31}$$

where n denotes the number of the input-output pairs in a subsample (batch) of the training dataset

The gradient of the network loss function with respect to each weight is computed by *propagating back* through the network, one layer at a time. Specifically, the backpropagation algorithm measures how much of the output node's loss contribution came from *each* node in the last hidden layer, then how much of this loss contribution came from each node in the previous hidden layer, and so on. The process continues until the algorithm reaches the input layer. Mathematically, the backpropagation algorithm works by applying the chain rule. The gradient of the network loss function with respect to one specific weight in the last layer, L, can be expressed as follows:

$$\frac{\partial \mathcal{L}_i}{\partial w_{jk}^L} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\partial \mathcal{L}_i}{\partial a_j^L}\right)\left(\frac{\partial a_j^L}{\partial z_j^L}\right)\left(\frac{\partial z_j^L}{\partial w_{jk}^L}\right) \tag{32}$$

This thesis will not dwell any deeper into the math underlying this algorithm, however, we have shown an example in appendix A.4 which proves equation 32.

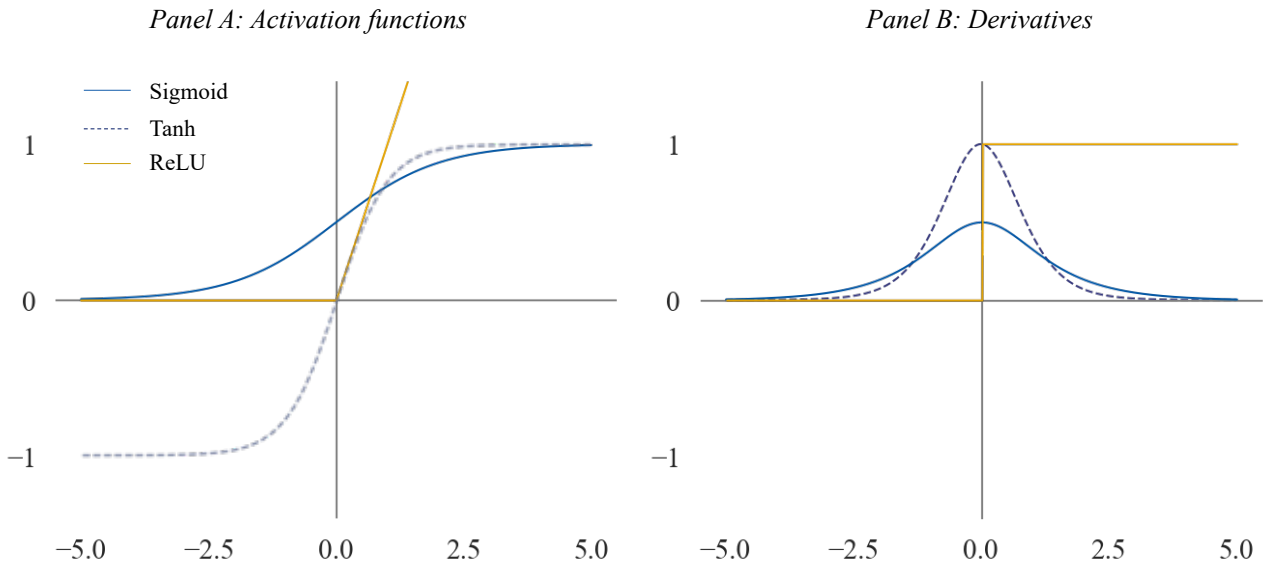
Stage 2: Adjusting the Weights

In order for the network to enhance its ability to predict an outcome, the network must adjust the weights in a way that minimizes the loss function in equation 31. After running one subsample (batch), the weights are adjusted by the amount that they contributed to the loss for that given batch using an optimization algorithm. Many optimization algorithms exist, however, the most predominant optimization algorithms in the literature rely on variations of gradient descent where the weights are adjusted one step at a time for each batch. This process of adjusting the weights is often repeated for multiple passes of the entire training dataset (epochs) until the network converges to a solution - ultimately computing the most accurate predictions.

Activation Functions

In order for the backpropagation algorithm to work properly, the activation function for the Perceptron (the heaviside step function) needs to be replaced with a differentiable activation function that enables gradient descent to make progress at every batch (recall that gradient descent works only for differentiable functions). Many potential options for a nonlinear activation function exist, including the sigmoid, hyperbolic tangent, rectified linear unit (ReLU), and softmax function. We apply the first three activation functions for our MLP and RNN model (see subsection 7.4.2 for rationale). Figure 4.13 shows the three activation functions (panel A) and their derivatives (panel B).

Figure 4.13: The three activation functions and their corresponding derivatives



(Source: Own creation)

The *sigmoid* activation function was the first activation function used in the MLP (Rumelhart et al., 1986). The sigmoid function squeezes the sum of the weighted inputs of a node into a range between (0, 1) which can be expressed as:

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}} \tag{33}$$

with the derivative

$$\frac{d}{dx}\mathrm{sig}(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x)\big(1 - f(x)\big) \tag{34}$$

As evident in Figure 4.13, the derivative of the sigmoid function becomes very small when the value of the sigmoid function is either very high or very low. This can cause vanishing gradients which lead to very small adjustments of the weights in the various layers within a neural network and ultimately poor learning for deep networks. Consequently, the sigmoid function is often used in the output layer only.

The hyperbolic tangent or 'tanh' function is similar to the sigmoid function but compresses the output values into the (-1, 1) range:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{35}$$

with the derivative

$$\frac{d}{dx}\tanh(x) = 1 - f(x)^2 \tag{36}$$

As evident by the tanh derivative in Figure 4.13, the function is centered around 0, leading to higher derivatives. Consequently, the updates of the weights are much larger which (often) makes it a superior activation function for hidden layers compared to the sigmoid function.

Finally, the rectified linear unit (ReLU) is one of the most widely used activation functions in recent literature as it encourages sparsity in the number of active nodes (Gu et al. 2019). This allows for faster derivative evaluation and consequently, for the network to converge towards a solution more quickly.
The ReLU function is given by:

$$\mathrm{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases} \tag{37}$$

with the derivative[11]:

$$\frac{d}{dx}\mathrm{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{38}$$

The ReLU can end up in what is referred to as the "dying ReLU problem". If x is below 0, the ReLU function has a gradient equal to 0. As a result, gradient descent will not work as it cannot alter the weights at this stage (i.e., the node is "dead"). However, ending up in such a stage can be mitigated by incorporating a lower learning rate into the model (Dawani, 2020).

4.3.3 Recurrent Neural Networks

Having established the fundamental architectures of neural networks, we now address the more advanced network, Recurrent Neural Network (RNN). Long Short-Term Memory (LSTM) cells are one of the several types of RNN architectures (How, Loo & Sahari, 2016). Therefore, this paragraph briefly introduces the idea behind RNN followed by its LSTM architecture.

The RNN is a neural network that is specialized for processing a sequence of values $x_1, \ldots, x_t$. MLPs store no information of the predictions made on one sample at a time step *t-1* when predicting the next sample in time step, *t*. In contrast, RNNs are known to have memory cells (hidden states) that enable them to remember and understand sequential data with e.g., a temporal relation. The hidden states have connections pointing back to themselves which enables them to have memory. Consider the following equation that defines how a simple RNN evolves over time:

$$h_t = f(h_{t-1}, x_t; \theta) \tag{39}$$

[11] Note that the derivative of ReLU(x)=0 is undefined

where, x_t is the input to the RNN at time t, h_{t-1} is the hidden state at time step t-1,
and θ is a parameter that (for simplicity) describes both the weights for h and x

Equation 39 is recurrent because the hidden state, *h*, at time step *t* refers back to *h* at time step *t−1*, with the former hidden state containing information about the whole past sequence, $x_1, \ldots, x_{t-1}$. Thus, we can also denote the hidden state at time step t as follow[12]:

$$h_t = g(x_t, x_{t-1}, \ldots, x_2, x_1; \theta) \tag{40}$$

Consequently, the RNN remembers its past, by allowing past computations to influence the present computations.

To better understand the RNN, it is helpful to visualize the network structure which can be done in two distinct ways. One way to visualize the RNN is through a diagram that contains one hidden state for all time steps. In this graph, the hidden state points back to itself, similar to a circuit that operates in real-time, with a current hidden state that influences the future hidden state (Figure 4.14, panel A). The second way to visualize the RNN is via an *unrolled* computational graph. Such a graph depicts the hidden state at each time step, with each hidden state representing the state at one point in time (similar to a "chain of events"). Unrolling refers to the operation that maps the circuit to a computational graph that has repeated pieces (Figure 4.14, panel B). The unrolled graph has a size equal to the length of the time steps, t, in the sequence **x**.

At a given time step t, the inputs are x_t and the hidden states are h_t. The function *f* maps the hidden state at t-1 to the hidden state at time step t. Note that we do not yet visualize the output of the network.

Figure 4.14: An RNN visualized as circuit diagram (panel A) unrolled through time (panel B)

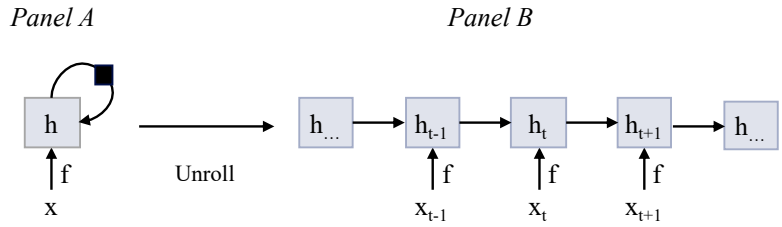


Figure 4.14 notes: The circuit diagram (panel A) is visualized with a black square indicating that an interaction takes place with a delay of a single time step. The same network (panel B) is unfolded to a computational graph where each hidden state is associated with one specific time step. The recurrent network processes information from the input **x** by incorporating it into the hidden state, h, which is passed forward through time. (Source: Own creation based on Géron, 2019).

The graph is unrolled by repeatedly applying the equation of h_t, t − 1 times. For example, if equation 39 is unrolled for t = 3 time steps the hidden state, h_3, is:

[12] Note that RNNs share parameters across time steps. Parameter sharing enables statistical strength across different positions in time and across different sequence lengths

$$
\begin{aligned}
h_3 &= f(h_2, x_3; \theta) \\
&= f(f(h_2, x_2; \theta), x_3; \theta) \\
&= f(f(f(x_1, \theta)x_2; \theta), x_3; \theta)
\end{aligned} \tag{41}
$$

In a similar manner as equation 40, we can also denote h_3 as

$$h_3 = g(x_1, x_2, x_3; \theta)$$

RNNs also add an output layer that reads information via a function I from the hidden state to make predictions:

$$\hat{y}_t = I(h_t, \theta) \tag{42}$$

where $\hat{y}_t$ is the output of the RNN at time step t

Figure 4.15 visualizes a computational graph of the entire RNN. At each time step t, the inputs are x_t, the hidden states are h_t, the outputs are $\hat{y}_t$:

Figure 4.15: An RNN visualized with outputs, as circuit diagram (panel A) unrolled through time (panel B)

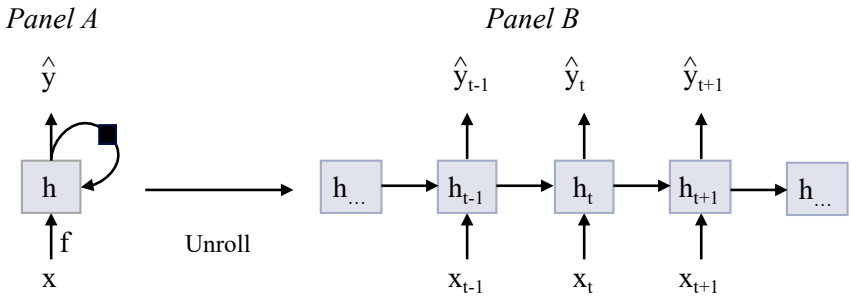
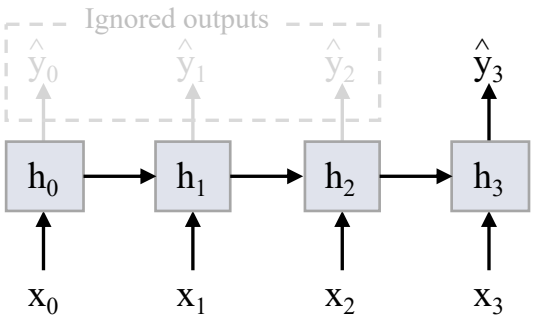


Figure 4.15 notes: The computational graph (panel B) maps the values of an input sequence, ***x***, to the corresponding sequence of output values $\hat{y}$. (Source: Own creation based on Géron, 2019).

Similar to MLPs, RNNs are trained via the backpropagation algorithm. However, for an RNN the algorithm is called "backpropagation through time", as it is applied to the unrolled RNN, that is, the network propagates backward through each time step when computing gradients.

An RNN can also take a sequence of inputs for a given number of time steps, ignore all outputs except for the last one, but still retain memory through the hidden state (Figure 4.16). This is typically referred to as a sequence-to-one network and is the type of network applied in this thesis:

Figure 4.16: A sequence-to-one Recurrent Neural Network



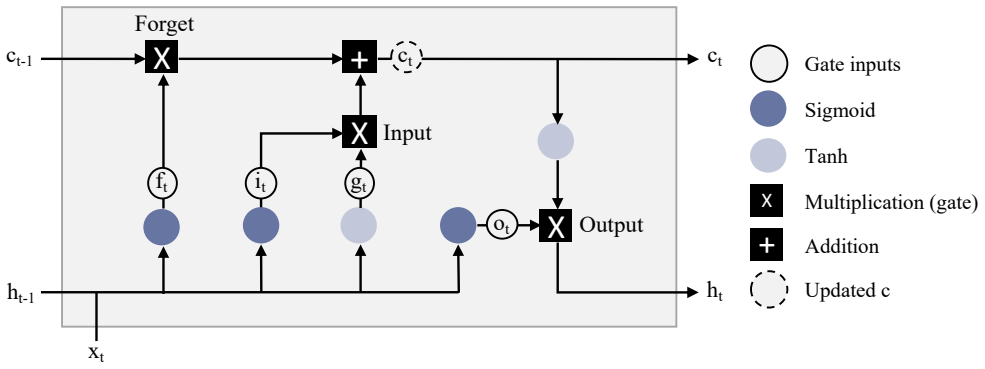
(Source: Own creation based on Géron, 2019)

4.3.4 Long Short-Term Memory

A specific type of RNN that applies a Long Short-Term Memory (LSTM) cell was first introduced by Hochreiter and Schmidhuber (1997). The key feature of the LSTM cell is its ability to learn both what to remember and what to forget. Such an ability enables the model to learn long-term dependencies which is challenging for a regular RNN that can (often) only retain short-term memory[13].

While the RNN has only one type of hidden state, the state of an LSTM cell can be split into two parts: the short-term state, h(t), and the long-term state, c(t). The LSTM cell regulates what to remember and what to forget through three gates, namely, a forget gate, an input gate, and an output gate. We first describe how the gates work followed by an explanation of how the two states are propagated forward in the LSTM cell through the three gates.

Figure 4.17: LSTM cell architecture



(Source: Own creation based on Géron, 2019)

Figure 4.17 illustrates the architecture of one LSTM cell. To explain the architecture of the LSTM cell, we follow Figure 4.17, closely: In order for the three gates to decide what to forget and what to remember, the

[13] This is due to the vanishing or exploding gradient problem (see subsection 7.4.2)

LSTM cell first computes four gate inputs, $f_t, i_t, g_t,$ and o_t. These gate inputs are transformed through the tanh or sigmoid activation function and fed to the three gates, namely the forget gate, the input gate, and the output gate.

The main gate input is g_t. The g_t is a tanh transformation of the current inputs, x_t, and the previous short-term state, h_{t-1}[14]. The additional three gate inputs are referred to as *gate controllers* (f_t, i_t and o_t). These three gate controllers are sigmoid transformations of x_t and h_{t-1}, and thus become values in the interval [0:1]. The gate controllers determine how much information is passed through the gates. As each gate operates through a multiplication operation, a gate will "close" if the respective gate controller equals 0. In contrast, a gate will "open" if the respective gate controller is above 0.

The forget gate (enabled by f_t) controls which parts of the long-term state should be dropped. The input gate (enabled by i_t) controls which parts of g_t should be added to the long-term state. Finally, the output gate (enabled by o_t) controls which parts of the long-term state should be passed on as the new short-term state (h_t).

Having established how the gates work, we turn to how the long-term state, c, is passed forward in the LSTM cell. First, the long-term state of the previous LSTM cell, c_{t-1}, encounters the forget gate. Here, elements of information are excluded. Following this gate, the long term-state encounters an additional operation that lets parts of new information enter the long-term state. At this point, the long-term state has both forgotten old and obtained new information, and becomes the *new* long-term state, c_t (marked as "updated c" in Figure 4.17). Subsequently, c_t is duplicated and passed forward along two paths: One path directly propagates c_t forward to the next LSTM cell. The other path transforms c_t via a tanh activation function and passes it through the output gate, ultimately resulting in the new short-term state, h_t. The h_t is fed forward to the next LSTM cell.

The above explanation is for one LSTM cell only. A snapshot of an unrolled RNN with LSTM cells is visualized in Figure 4.18 where an LSTM cell is represented at each time step. Note that for the last LSTM cell, the h_t is equivalent to the output $\hat{y}_t$ (Géron, 2019).

Figure 4.18: Snapshot of an RNN with LSTM cells, unrolled through time

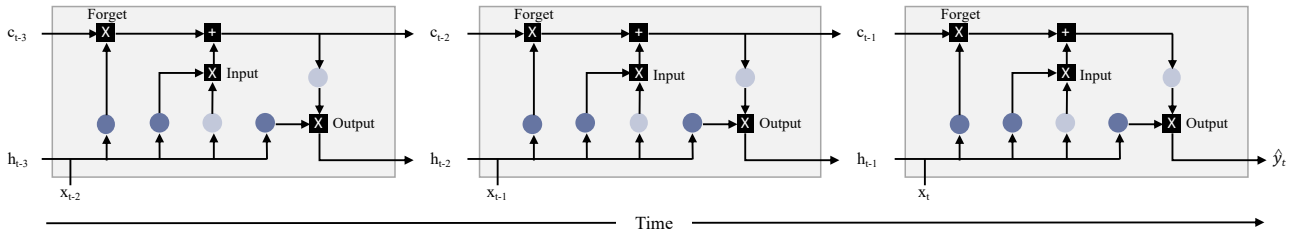


Figure 4.18 notes: The figure depicts an RNN with LSTM cells unrolled through time. Only the last three time steps in a sequence **x** are visualized. In the last layer, the hidden state h_t becomes $\hat{y}_t$. (Source: Own creation).

[14] Note that in a basic RNN cell, there is nothing else than this input, and h_t passes directly on to the next cell, without passing through any gates (Géron, 2019)

The computation of the LSTM cell's long-term state, short-term state, and the output at the last timestep is summarized below:

$$f_t = \sigma(w_{xf} * x_t + w_{hf} * h_{t-1}) \quad (43)$$

$$i_t = \sigma(w_{xi} * x_t + w_{hi} * h_{t-1}) \quad (44)$$

$$g_t = \tanh(w_{xg} * x_t + w_{hg} * h_{t-1}) \quad (45)$$

$$o_t = \sigma(w_{xo} * x_t + w_{ho} * h_{t-1}) \quad (46)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \quad (47)$$

$$\hat{y}_t = h_t = o_t \tanh \otimes (c_t) \quad (48)$$

where w_{xf}, w_{xi}, w_{xg}, w_{xo} are the weights of each of the four gate inputs for their connection o the input vector x(t), and w_{hf}, w_{hi}, w_{hg}, w_{ho} are the weights of each of the four gate inputs for their connection to the previous short-term state h_{t-1}

Collectively, the two states of the LSTM cell enable it to recognize essential inputs, preserve them for as long as necessary, and extract them when required. This is why RNNs with LSTM cells have been highly successful at capturing long-term patterns in time series - including stock prediction (Fischer & Krauss, 2018).

As a final note on neural networks, it can be extremely challenging to draw meaningful interpretations of the underlying mechanisms from these networks due to their high complexity (Ghorbani et al., 2019). Consequently, they are often referred to as "black boxes". This thesis aspires to unfold the black box by assessing the importance of each input variable of the machine learning models and regressing their predictions against profound risk factors in the literature in sections 9.5 and 9.6, respectively (research questions 5).

4.4 Summary of Machine Learning Models

For the convenience of the reader, we provide an overview of the theory outlined in the previous machine learning sections (sections 4.2-4.3). Table 4.1 presents the key characteristics of the three machine learning models deployed in this thesis.

Table 4.1: Summary of the three machine learning models

Algorithm	Overall description	Architecture	Training process	Predictions
Random Forest	A tree-based model that relies on ensemble learning via a collection of decision trees	Each tree consists of nodes that are repeatedly split into new nodes or leaves. Trees are constructed through a Random Forest bagging approach that draws random bootstrap samples from the training set and selects a random subset of variables for each split in each tree.	Relies on tree construction. For each tree, Random Forest finds the best splitting variables (among the random subset of variables) and the best splitting points for these variables.	Averaging the predictions of each individual tree
Multilayer Perceptron	A neural network composed of layers of nodes with data flowing *forward* only	Nodes are organized into input, hidden, and output layers. Weights connect the nodes with outputs passed to nodes in the subsequent layers via an activation function. For recurrent neural networks, an additional temporal dimension is added, allowing it to have memory (state) of previous events.	Initially, the network assigns random internal parameters (e.g. weights) to the network. Parameters are updated using the backpropagation method and gradient descent in order to minimize the loss. The parameters are adjusted for every subsample (batch).	Extracted from the final output layer that relies on the inputs from previous layers and the optimal internal parameters
Recurrent Neural Network with LSTM cells	A recurrent neural network that incorporates LSTM cells allowing it to forget and remember short- and long-term dependencies			

(Source: Own creation)

5 Overarching Methodology

This chapter sheds light on the overarching methodology on which our research relies. First, we address the validity and reliability of our data sample and the actions taken to ensure the quality of our research. Second, we introduce the method of backtesting, our investment universe, and the initial preprocessing of the data. Finally, we explain the eleven momentum variables that constitute the inputs for our selected machine learning models.

5.1 Validity and Reliability

This section reviews the quality of our data sample by assessing its validity and reliability. **Validity** is a measure of the extent to which the collected data can be used for answering the problem statement (Carmines & Zeller, 1979). We refine the validity of our research through two primary undertakings. First, as previously established, this thesis investigates the performance of machine learning-based stock-selection strategies relative to a conventional momentum strategy. Our research focuses, inter alia, on the momentum crashes of the momentum strategy that occur during stressed markets - a phenomenon that has been documented by several scholars (e.g., Grundy & Martin, 2001; Daniel & Moskowitz, 2016). However, when we replicate the

conventional momentum strategy, we limit ourselves to a smaller data sample than the literature which consists of 500 stocks each month (see section 1.3). Thus, our data sample can only be regarded as valid for answering our problem statement, insofar momentum crashes are *also* evident in our data sample. Therefore, we initiate our results (subsection 9.1) by examining if momentum crashes are also inherent in our data sample.

Second, we include periods of stressed markets in both the training and test set for our research to be valid. Doing so permits the machine learning models to *learn the patterns* that occur during stressed markets (using the training set). Moreover, it also allows us to *assess the performance* of the machine learning-based stock-selection strategies during stressed markets (using the test set). There have been two major momentum crashes over time, namely following the Great Depression and the 2008-2009 financial crisis. Thus, we have included the Great Depression in the training sample and the 2008-2009 financial crisis in the test sample.
Collectively, the two undertakings foster a data sample that can be used to answer our problem statement.

Reliability is a measure of how dependable the data sample is and whether the results might be affected by coincidences (Carmines & Zeller, 1979). We encourage reliability in our research through four areas:
First, we exclusively rely on credible data sources in our research. The qualitative data consist of peer-reviewed articles from renowned journals primarily, such as The Journal of Finance and Journal of Financial Economics. The quantitative data have been collected from the Center for Research in Security Prices ("CRSP"), the Kenneth French Data Library[15], and the website of AQR Capital Management[16]. These are all data sources that academic scholars rely on in their peer-reviewed research. By solely relying on these credible data sources, we attempt to ensure reliability of our data.

Second, we are cautious when interpreting our results. As we explain in the following section, this thesis follows the method of backtesting. The backtesting methodology is associated with certain data mining biases, some of which are unavoidable (Pedersen, 2015). For example, the machine learning-based stock-selection strategies rely on variables that previous scholars have identified as relevant for the enhancement of the conventional momentum strategy. However, at the beginning of our backtest period, these variables were not yet discovered by scholars, and hence, we would not be able to deploy them at that point on time. Furthermore, it would not have been possible to deploy machine learning 45 years back in time. We are attentive towards these unavoidable biases by acknowledging that our results function only as an estimate of whether the proposed stock-selection strategies would have been profitable, had they been deployed in real-time. In a similar manner, we note that our results (based on the past) are not guaranteed to hold in the future.
Besides the unavoidable bias, we have actively attempted to circumvent biases in our data analysis. For instance, we test the performance of our stock-selection strategies on an out-of-sample test set to obtain a reliable estimate of their performance (recall subsection 4.1.3). Further, we include dead stocks[17] in our data sample to avoid survivorship bias, and exclusively include stocks at the time they are a part of our investment universe to avoid look-ahead bias.

[15] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
[16] https://www.aqr.com/Insights/Datasets/Betting-Against-Beta-Equity-Factors-Daily
[17] Stocks that were delisted during the holding period

A third way in which we encourage reliability in our research is by ensuring replicability of our results. The results of our machine learning-based stock-selection strategies rely on the predictions made by the machine learning models. These predictions vary slightly every time the models are run, leading to a certain randomness in our results which ultimately hinders replicability. In a modest attempt to counter the randomness and improve the reliability of our results, we run the machine learning models five times and average their predictions.

Finally, we deploy a simple baseline model for comparison with the more complex models deployed in this thesis. Specifically, we apply a simple OLS linear regression model. The rationale for doing so is to ensure that the simpler counterpart does not exhibit superior or similar performance relative to the three machine learning models, rendering them overly tortuous (complexity bias). Hence, the performance of the baseline model functions as floor value which the more complex models should outperform.

5.2 Backtesting, Data Description, and Fundamental Preprocessing

As this thesis examines the performance of a conventional momentum strategy and three machine learning-based stock-selection strategies, we apply the method of backtesting. More specifically, we assess the viability of these stock-selection strategies by simulating their performance over time (Schmidt, 2011). Where the specific trading rules of the respective stock-selection strategies are specified in chapter 6 and 7, this section introduces the data and investment universe used for the backtesting of the stock-selection strategies.

This thesis examines the US equity market and covers a period from January 1929 to December 2020. The sample includes common equity stocks traded on NYSE, AMEX, or NASDAQ, extracted from CRSP. We include common stocks with a CRSP share code (SHRCD) equal to 10 or 11.

We rely on monthly returns and apply the monthly delisting returns when a stock has been delisted during the holding period. The choice of utilizing monthly returns rather than daily returns is consistent with scholars using machine learning for asset pricing (Gu et al., 2019; Gu, Kelly & Xiu, 2021). In fact, Israel et al. (2020) argue that using monthly returns is a sensible starting point when applying machine learning to stock market analysis. In addition, it would be infeasible to use daily returns as inputs to our machine learning models given the limited computational power in our possession. We do, however, *evaluate* the performance of the stock-selection strategies using daily return data. This allows us to evaluate the stock-selection strategies on a both daily and monthly basis, enabling comparability with the literature that relies on both daily returns (e.g., Daniel & Moskowitz, 2016) and monthly returns (e.g., Gu et al., 2019). Moreover, using daily returns resemble a real-life investment strategy.

We define our investment universe as the 500 largest stocks each month, measured by market capitalization. At the beginning of each holding period, (t-1), we rank the stocks based on their market capitalization so that:

$$X_{i,t} \in \mho_t | \, rank(MC_{i,t-1}) \geq 500 \tag{49}$$

where $X_{i,t}$ denotes stock i in the holding period t, and $\mho_t$ denotes the investment universe at month t

When no price is available, we apply the bid-ask average instead. The market capitalization used to rank stocks at time t is then given by:

$$MC_{i,t-1} = abs(PRC_{i,t-1})(SHROUT_{i,t-1}) \tag{50}$$

where PRC is the price or bid-ask average and $SHROUT$ is the number of shares outstanding

For a stock to be included in the portfolio, we require valid returns from months 't to t-24' as this is a necessity when calculating our variables for our machine learning-based stock-selection strategies. Note that this implies that the first full calendar year for which we investigate portfolio returns is 1931, as our data sample begins in 1929.

We end up with an investment universe consisting of 3,250 stocks, spanning a period of 1,080 months. Table 5.1 summarizes the descriptive statistics for the stocks in the final sample:

Table 5.1: Descriptive statistics of data sample

Number of stocks in investment universe	3,250
Number of stocks per month	500
First observation date	01 Jan 1931
Last observation date	31 Dec 2020
Duration of sample period in months	1,080
Return frequency, investment strategy construction	Monthly
Return frequency, investment strategy evaluation	Daily

(Source: Own creation)

5.3 Momentum Variables

We construct eleven momentum variables based on the momentum literature as input variables for our machine learning models. While the scholars described in subsection 3.4.3 estimate their momentum variables using different time horizons and different return frequencies, we apply monthly returns and a one-year time horizon when creating the variables. We describe these variables and the rationale for utilizing them below. The variables are divided into market and stock-specific variables.

The Market Variables

We include three market variables. Let *t* denote the month in which the machine learning models predict a return of $stock_i$. Then, the variables can be expressed as:

1) The return of the market at month t-1
2) The cumulative market return from month t-2 to t-12
3) The standard deviation of the market computed on a rolling basis over month t-2 to t-12

The rationale for including the market variables is inspired by the findings of Barroso and Santa-Clara (2015) as well as Daniel and Moskowitz (2016): As mentioned in subsection 3.4.2, Daniel and Moskowitz (2016) find that momentum crashes occur following bear markets that suddenly rebound. To incorporate these findings, we deploy the cumulative t-2 to t-12 months returns and the one-month market return at time t-1. We further include the standard deviation of the market as Daniel and Moskowitz (2016) as well as Barroso and Santa-Clara (2015) document that momentum crashes occur when the market *volatility* is high. Accordingly, this enables the machine learning models to extract additional information on stressed market situations.

The Stock-Specific Variables

We include eight stock-specific variables:

4) The return of $stock_i$ at month t-1
5) The cumulative return of $stock_i$ from month t-2 to t-12
6) The standard deviation of $stock_i$ computed on a rolling basis from month t-2 to t-12
7) The beta of $stock_i$ computed based on a rolling regression over month t-2 to t-12 using the CAPM model
8) The alpha of $stock_i$ at month t-1, computed based on a rolling regression over month t-1 to t-12 using the Fama/French three-factor model
9) The cumulative alpha of $stock_i$ from month t-2 to t-12
10) The idiosyncratic returns of $stock_i$ at month t-1, computed based on a rolling regression over month t-1 to t-12 using the Fama/French three-factor model
11) The cumulative idiosyncratic returns of $stock_i$ from month t-2 to t-12

The two first stock-specific variables (four and five) are included as they are closely related to the conventional momentum strategy: Variable four incorporates possible short-term reversal in stock returns and represents the skipping period for a conventional momentum strategy. The fifth variable constitutes the key variable for the

conventional momentum strategy that ranks stocks according to their past t-2 to t-12 months cumulative returns.

We have previously established that the conventional momentum strategy has a highly time-varying beta. Further, we have established that momentum crashes occur during periods characterized by high volatility. Consequently, we include the market beta and the standard deviation for each stock (variables six and seven) as input variables[18].
As the final four variables (eight to eleven), we include the monthly as well as the cumulative idiosyncratic and alpha momentum (using equations 14 and 16). These variables incorporate the findings of Hühn & Scholz (2018) and Blitz et al. (2011, 2020) on alpha and idiosyncratic momentum, respectively[19].
These eleven momentum variables incorporate the findings of the enhanced momentum strategies. Hence, they comprise the foundation for which the machine learning models may uncover contextual and nonlinear relationships to enhance the conventional momentum strategy.

Some of the momentum variables are based on a rolling basis while others are observations in a single month. The rationale for incorporating both is two-fold: First, the machine learning models deployed in this thesis do not all comprehend the temporal dimension (recall that only RNNs have a memory of past inputs). Thus, we need variables that in themselves contain a time dimension (e.g., cumulative returns) and variables at a specific month that can be placed at each time step (e.g., monthly returns). Second, we choose to deploy the two "types" of variables to all models to ensure consistency. This enables us to compare the performance of our models as no additional inputs are fed to one model. Note that this means we "artificially" assign timesteps to Random Forest and the MLP model. We explain how we apply such an approach in section 7.2.

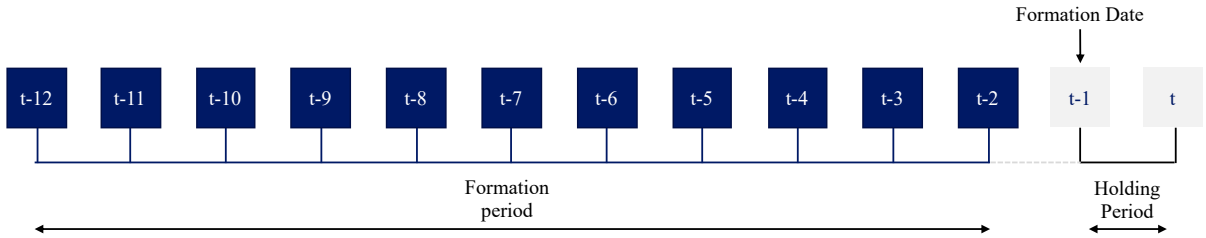
[18] Note that Daniel & Moskowitz (2016) as well as Barroso and Santa-Clara (2015) calculate the standard deviation of the aggregate momentum portfolio whereas we calculate the standard deviation on an individual stock level.
[19] We do not apply equation 15 as Blitz et al. (2020) notes that their results do not hinge on scaling idiosyncratic momentum

6 Methodological Framework of the Momentum Strategy

This chapter presents the methodological approach for replicating the conventional price momentum strategy, popularized by Jegadeesh and Titman (1993). As such, the chapter explains the trading rules underpinning the backtest of the conventional momentum strategy.

Recall that the first step in forming a momentum strategy is ranking stocks based on their past returns (in the formation period). Subsequently, stocks are held in a portfolio during the holding period. We follow the most broadly used momentum definition in recent literature (e.g., Asness et al., 2013; Daniel & Moskowitz, 2016; Blitz et al. 2020), and rank stocks based on their cumulative returns[20] from 12 months to one month before the formation date (i.e., from t−2 to t−12 months). We include a one-month skipping period (t−1) to disentangle the intermediate-term momentum effect from the short-term reversal effect as documented by Jegadeesh (1990) and Lehmann (1990). Figure 6.1 visualizes the formation and holding period.

Figure 6.1: Formation period and holding period of a conventional momentum strategy



(Source: Own creation)

Based on the stock ranking, we form decile portfolios and assign equal weights to the stocks in each decile which is common in the literature (see Jegadeesh & Titman 1993, Chordia & Shivakumar, 2002; Griffin et al., 2003; Blitz et al. 2011, 2020). Alternative weighting schemes incorporating market capitalization (value-weighted portfolios) are also applied in the literature (see Moskowitz & Grinblatt, 1999; Korajczyk & Sadka, 2004; Asness et al., 2013; Daniel & Moskowitz, 2016). The value-weighted approach is advantageous when the stock universe includes small and illiquid stocks, as they receive less weight in the momentum portfolios. However, our stock universe cannot be regarded as illiquid as the sample consists of the top 500 stocks based on their market capitalization. The top (bottom) portfolio comprises the 10% of stocks with the highest (lowest) cumulative returns. The literature has primarily focused on a "zero-cost" strategy involving a long position in the top decile (the winner portfolio) and a short position in the bottom decile (the loser portfolio). We deploy such a long-short momentum portfolio (recall that we refer to this portfolio as WML).

Consistent with the literature, we rebalance portfolios monthly (e.g., Daniel & Moskowitz, 2016; Blitz et al., 2011, 2020). While varying the rebalance frequency to optimize portfolio trading costs may improve the

[20] The cumulative return for each stock is calculated as the compounded returns, i.e., $\prod_{t=1}^{n}(1 + R_t)$, where t is the given month

implementation of the strategy (Garleanu & Pedersen, 2012), we note that this is beyond the scope of our thesis. For the convenience of the reader, we visualize the fundamental step-by-step approach of our replicated conventional momentum strategy in Figure 6.2.

Figure 6.2: Step-by-step approach of the conventional momentum strategy

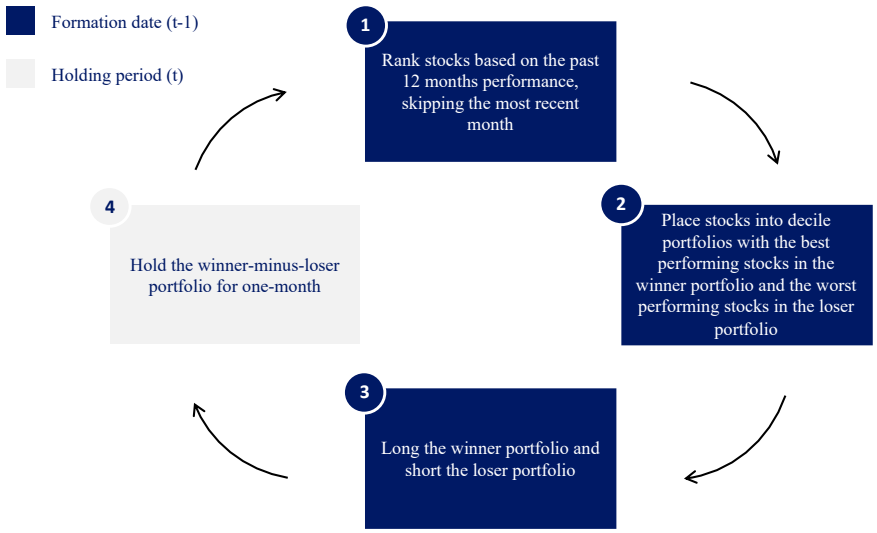


Figure 6.2 notes: The figure illustrates the four steps of our replicated conventional momentum strategy. Note that steps one, two, and three are performed on the formation date, while step four takes place during the holding period. The process (steps one to four) is continued in a cyclic form. (Source: Own creation).

7 Methodological Framework of the Machine Learning Strategies

This chapter provides an overview of the methodological considerations related to the machine learning-based stock-selection strategies. First, we provide a brief introduction to how we construct the stock-selection strategies based on the predictions made by the machine learning models. Subsequently, we outline how the data are preprocessed before we pass the data to the machine learning models. The third section explains the methodology used for optimizing the hyperparameters of the machine learning models. Finally, the last three sections offer an in-depth description of how the three machine learning models are constructed as well as their respective hyperparameters.

7.1 Introduction to the Machine Learning-Based Stock-Selection Strategies

To construct the machine learning-based stock-selection strategies, we build supervised machine learning models that predict the excess stock returns of the following month, based on the eleven momentum variables.

We sort stocks into decile portfolios according to their respective predicted excess returns[21]. Then, the machine learning-based stock-selection strategies follow the approach for the conventional momentum strategy by longing the winner portfolio and shorting the loser portfolio. The position is held for one month.

For the convenience of the reader, a visual representation of the process of constructing a machine learning model and utilizing it to create a stock-selection strategy is provided in Figure 7.1.

Figure 7.1: Step-by-step approach of constructing a machine learning model and applying it for a stock-selection strategy

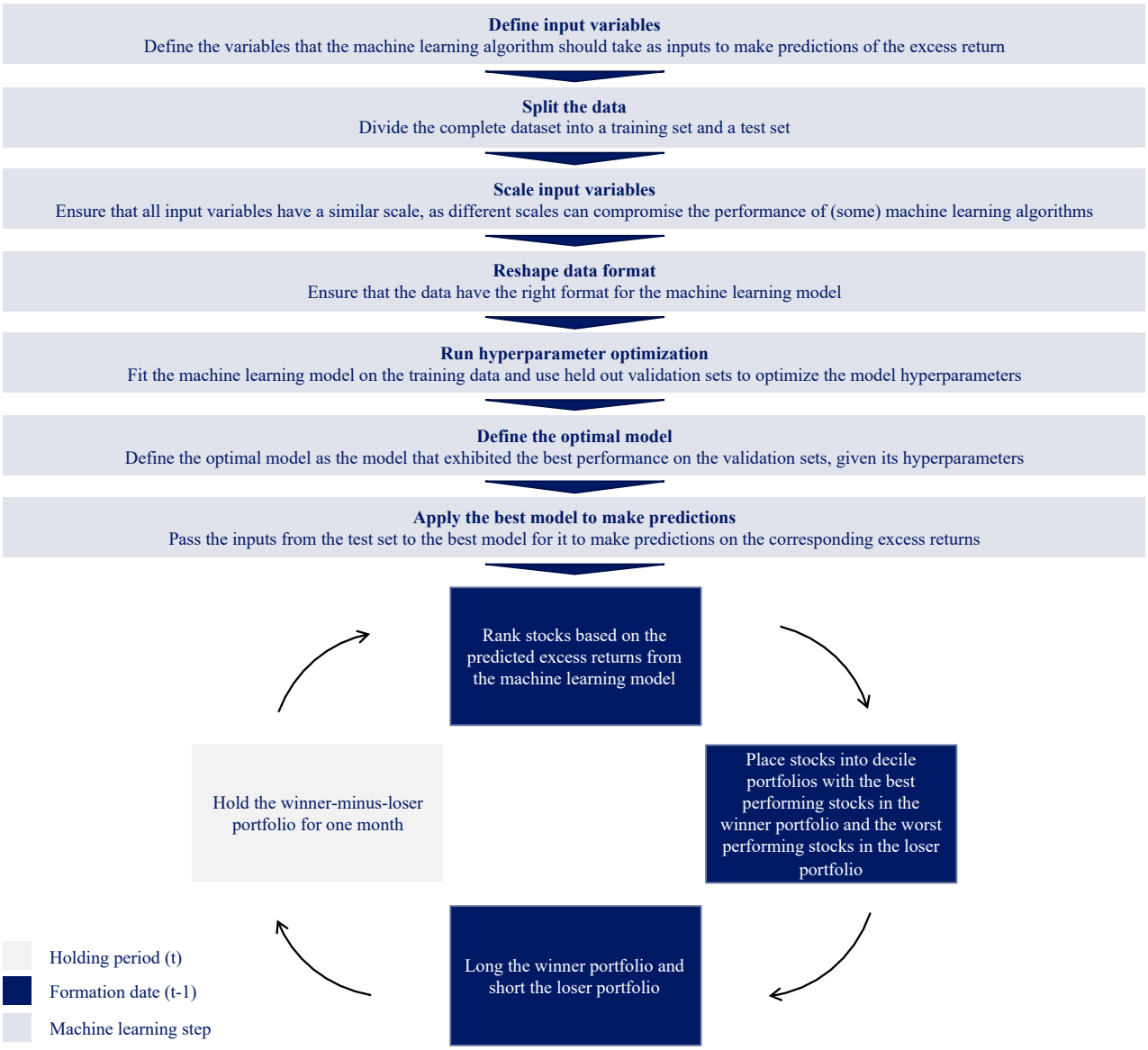


Figure 7.1 notes: The figure depicts the process for constructing a machine learning model and utilizing it to create a stock-selection strategy. We visualize three types of steps. The light blue steps are related to the machine learning modeling process. The dark blue

[21] Note that we rank stocks based on the excess stock returns in the cross-section at formation date. Thus, predicting returns or excess returns should not influence our result. In practice, however, withdrawing the risk-free rate creates less noise for the machine learning models, resulting in better predictions.

steps outline the process of forming portfolios based on the machine learning predictions. The grey step represents the holding period. (Source: Own creation).

The remaining sections in this chapter unfold the specific methodology deployed in this thesis when performing each of the machine learning-related steps (marked in light blue in Figure 7.1). We will not dwell on the details of the steps for creating the stock-selection strategies, as these resemble the process of forming a conventional momentum strategy. Hence, the trading rules for the backtest of the machine-learning-based stock-selection strategies are similar to that of the conventional momentum strategy, except that the stocks are ranked based on their *predicted* excess returns rather than their cumulative returns in the formation period.

We proceed by specifying the regression problem that we aspire to solve. We define our input variables as the eleven momentum variables. The input variables, *z*, for stock, *i,* of a given month, *t-1*, are used to predict the excess return for the following month, *t*. As the output variable is a continuous numerical value, the problem at hand is a regression problem. In the most general form, we describe the excess stock return as an additive prediction error model:

$$r_{i,t} = \hat{r}_{i,t} + e_{i,t} \tag{51}$$

and

$$\hat{r}_{i,t} = \mathrm{g}^{*}(\mathrm{z_{i,t-1}}) \tag{52}$$

where stocks are indexed as $i = 1, \ldots, N_t$, and months are indexed as $t = 1, \ldots, T$

Our objective is to find the predicted excess stock return, $\hat{r}_{i,t}$ that minimizes the loss function (MSE) for the target output $r_{i,t}$. Assume that the predicted excess return is a flexible function, $g^{*}(\cdot)$, of the eleven momentum variables, $z_{i,t}$. The prediction of a given excess stock return is based on the input variables *z* for stock *i* at time *t*. The input variables for each stock *i* do not contain information about *individual* stocks other than the i[th].

Statistical Properties of Data

Before we unfold how we have preprocessed the data for our machine learning models, we briefly explain the statistical properties of our data that are relevant for the models. First, we note that the machine learning models deployed in this thesis do not assume any formal distributions of the data and are not restricted by the various assumptions underlying more traditional statistical models. However, stationarity in time-series data is generally recommended in the field of machine learning. For machine learning models without a temporal dimension, stationary time series can be easier to analyze, as the mean and standard deviation of data are not time-dependent (Lazzeri, 2020). To ensure that the statistical properties of our data remain constant over time, we perform an Augmented Dickey-Fuller test (Dickey & Fuller, 1979) in appendix A.5 which illustrates that the variables are stationary.

Neural networks and tree-based models, similar to those deployed in this thesis, are documented in the financial literature to successfully handle correlated input variables (e.g., Gu et al., 2019). However, the input variables should not be *perfectly* correlated as this would involve redundancy in our machine learning models. Appendix

A.6 demonstrates that this is not the case for the eleven momentum variables that exhibit a correlation no higher than 0.64.

7.2 Preprocessing of Data for Machine Learning Models

Training and Test Set

Having defined the input and output variables for the machine learning algorithms, the first step of data preprocessing is to divide the dataset into a training and test sample. We use the training sample to build and train our machine learning models. The out-of-sample test set is used to assess the predictive performance of the machine learning models. Deciding which proportion of the data should be in each of these samples is somewhat arbitrary in the literature (Müller & Guido, 2016). The split is dependent on the size of the overall sample, as a small sample requires a higher fraction of training data for the algorithms to learn patterns in data. Gu et al. (2019) divide 60 years of data equally into a training set (including a validation set) and a test set, maintaining the temporal ordering of the data. We follow their approach and split the datasets equally, maintaining the temporal dimension. As our data sample is longer than the sample of Gu et al. (2019), spanning 90 years, the length of the training sample should constitute a solid foundation for the algorithms to learn patterns in data.

Scaling input variables

In general, a machine learning algorithm creates a mapping from the input variables to an output variable. The input variables can vary in terms of different scales and distributions, as is the case for our variables (see appendix A.7). While Random Forest, by construction, is capable of handling variables with different scales, it poses a challenge for neural networks[22]. Hence, we scale our eleven momentum variables. Two different scaling methods are applied, as a result of utilizing both market and stock-specific variables:

For each of the three market variables, m, we scale values X_m such that the distribution of a given variable has a mean of zero and a standard deviation of one. Specifically, when scaling a value for variable m, we subtract the mean of variable m in the *training* dataset and divide by the standard deviation of variable m from the *training* dataset. We apply the scaler on both values in the training and test dataset. The scaled observation Z for a variable m is given by:

$$Z_m = \frac{\mathrm{X_m} - \mu_{\mathrm{train}}}{\sigma_{train}} \tag{53}$$

where X_m is an observation for the variable m, μ_{train} is the mean of the training dataset and σ_{train} is the standard deviation of the training sample

[22] As the networks learn how to combine the inputs through a series of linear combinations (and nonlinear activation functions), the weights associated with each input will also have different scales. In practice, this leads to more emphasis on certain weight gradients and an unstable learning process (Bishop, 1995) (we elaborate on this in subsection 7.4.2)

We follow the approach of Gu et al. (2019); Kelly, Pruitt, and Yinan, 2019; as well as Freyberger, Neuhierl, and Weber (2020) when scaling the eight stock-specific variables. The purpose of deploying the three machine learning algorithms to predict excess returns is to sort stocks into portfolios for a given month. When we form portfolios, we are not interested in the variable for a given stock in isolation, but rather the value in the cross-section. Consequently, we cross-sectionally rank values of the stock-specific variables, month-by-month, and map the ranks into an interval of [-1,1]. Specifically, we apply the following function for each month, t:

$$Z_{m,t} = \frac{2}{N_t - 1} * rank\left(X_{m,t}\right) - 1 \tag{54}$$

where $rank\left(min_{i=1\dots N_t}\left(X_{m,t,i}\right)\right) = 0, \quad rank\left(max_{i=1\dots N_t}\left(X_{m,t,i}\right)\right) = N_t - 1,$

N_t denotes the total number of stocks at month t, and i denotes a single stock

Reshaping the Data Format

The three machine learning algorithms deployed in this thesis require different shapes of the input variables. Random Forest and Multilayer Perceptrons (MLPs) are only capable of handling inputs formatted as fixed-length vectors, *N*, with *M* input variables. In contrast, Recurrent Neural Networks (RNNs) are structured to specifically accommodate sequences of inputs, and thus the network requires a matrix format that adds the timesteps, *T*, to the N_i vector, where *i* is one row of observations. We proceed by explaining how we reshape our input data to accommodate the required format for the three models and refer to a graphical representation of the input shape of the data in Figure 7.2.

Let *M* denote our momentum variables, then $N_{i,t}$ is a vector of all *M* variables for one stock i at month *t*. For each month, we have 500 firms in our universe so $N_{i,t}$ is for $i = 1, \dots, 500$. We further add time steps to this vector. Currently, there is no rule of thumb to select the number of time steps (Bao, Yue, & Rao, 2017). We include a sequence of 12 time steps, *T*, corresponding to one year. We are unable to increase the time steps further, as this exceeds the computational resources available for this thesis. Note that we incorporate the time dimension into our models in two distinct ways: For Random Forest and the MLP that require a vector format, we add T to the vectors, so the length of the vectors $N^* = $ NxT. For the RNN we add T as a new dimension, creating NxT-matrices. In both cases, this results in $11x12 = 132$ inputs, presented in two different shapes. For the RNN, T enables the model to "memorize" observations one year back in time[23]. While the MLP and Random Forest do not comprehend the temporal aspect of the data, they still incorporate this aspect in their predictions through the vectors N^*.

Figure 7.2 visualizes the inputs to the models at month *t* with $i = 1, \dots, 500$ vectors N_i^* for MLP and Random Forest, and $N_i xT$ matrices for the RNN:

[23] We indirectly enable it to memorize further back in time as seven of our variables are estimated over a 12-month period, skipping the most recent month

Figure 7.2: Input data shape (at each month)

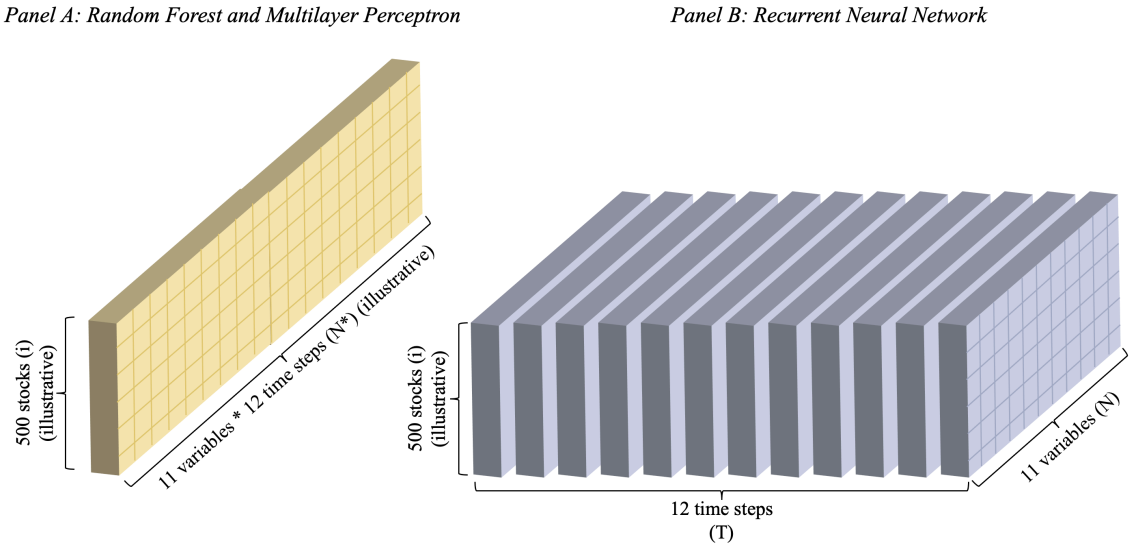


Figure 7.2 notes: The figure depicts how the input data are shaped to fit the required input shape of the three machine learning algorithms. Panel A shows the N^* vector required for Random Forest. Panel B shows the NxT matrix required for RNN. (Source: Own creation).

For each month, t, the machine learning models process 66,000 data points ($500\ x\ 11\ x\ 12$). This results in 71,280,000 data points for our entire data sample spanning 90 years.

7.3 Hyperparameter Optimization

Having outlined the preprocessing of the data for the machine learning models, we proceed by explaining how we optimize the performance of our models. Finding the optimal hyperparameters for a machine learning model is crucial for optimizing model performance. Recall that the optimal hyperparameters for a given model are the hyperparameters that minimize the loss function. We deploy MSE as the loss function. To optimize the hyperparameters, we follow the most common approach in the machine learning literature and decompose the training data into a hold-out validation set and a reduced training set (Goodfellow et al., 2017). The validation set is used for tuning the hyperparameters.

As mentioned in subsection 4.1.3, a common approach for finding the optimal hyperparameters is k-fold cross-validation, where the model is validated on multiple validation sets. However, the standard k-fold cross-validation method is not applicable for time series data, as it does not maintain the temporal ordering of the training and validation data (Arnott et al., 2019). As our sample consists of time series data, deploying k-fold cross-validation would imply that investors have perfect future hindsight of stock returns, i.e., look-ahead bias. Consistent with the literature applying machine learning on stock data, we cross-validate our models by repeatedly increasing the size of the training sample and rolling the validation set forward so that it consists of the most recent data (Chen et al., 2020; Gu et al., 2021). We split the training set to obtain *ten* validation sets and show the rolling process in Figure 7.3:

Figure 7.3: Training, validation, and out-of-sample test data

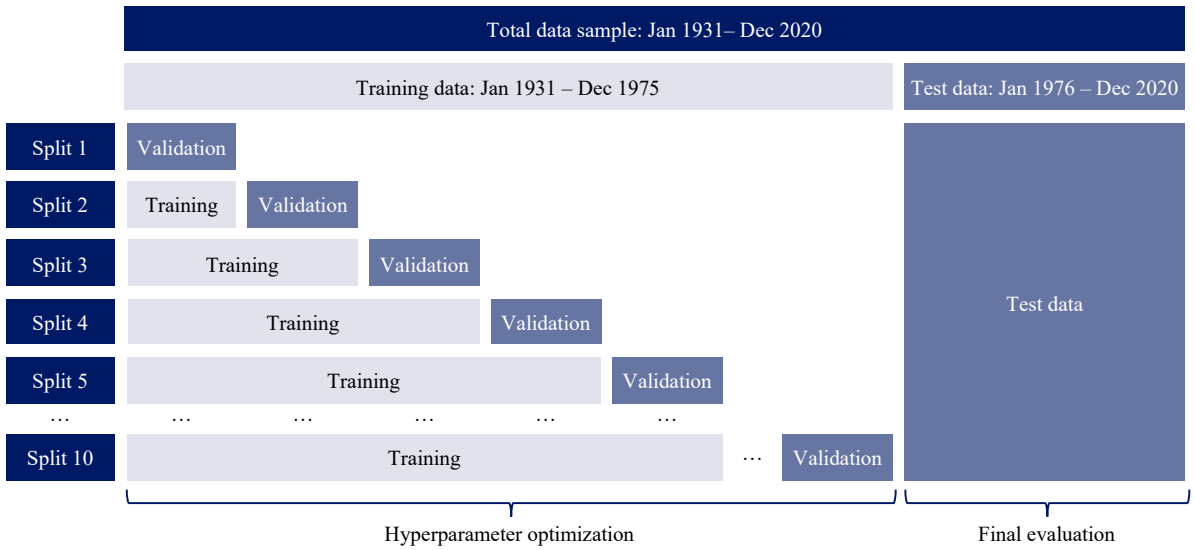


Figure 7.3 notes: The total dataset is split into a training and test dataset. The training dataset is further divided into a reduced training set and a validation set. To validate the models, the size of the reduced training set is repeatedly increased (10 times), and the validation set is rolled forward to include the most recent data. This method is a variation of k-fold cross-validation that remains the temporal ordering of the time series data. (Source: Own creation).

There are several different methods for obtaining the optimal hyperparameters that minimize the loss function for a given model. Two common methods are grid search, which systematically evaluates specified values of the hyperparameters "in a grid", and random search, which randomly tests different values of hyperparameters from a search range. We visualize an example of the two methods in Figure 7.4:

Figure 7.4: Grid search and random search

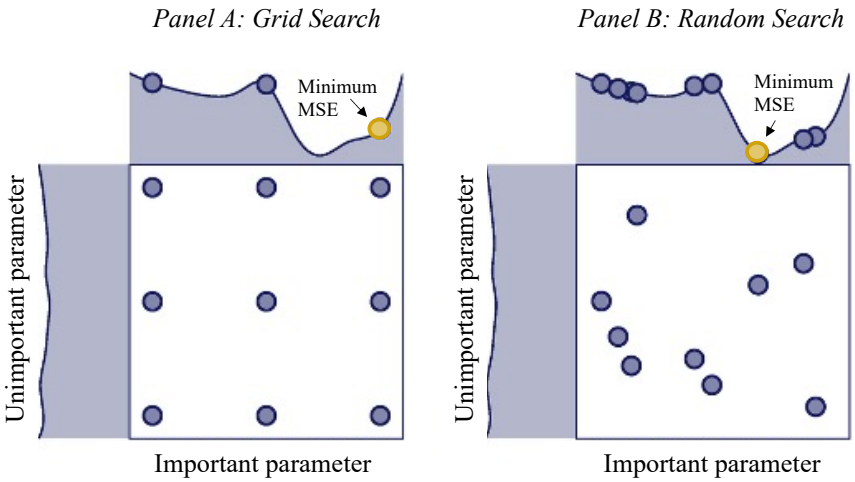


Figure 7.4 notes: Panel A shows the grid search method and panel B shows the random search method. The y-axis depicts a hyperparameter that has little impact on the model's loss function (MSE). The x-axis depicts a hyperparameter that has a substantial effect on the MSE. (Source: Own creation based on Géron, 2019).

Consider an optimization process for two hyperparameters. For a grid search with a 3x3 grid, only three different values for each hyperparameter are tuned (corresponding to three rows and three columns in panel A), even though the model is evaluated nine times. In contrast, when the model is evaluated nine times using random search, nine different values for *each* hyperparameter are evaluated (corresponding to nine distinct rows and nine distinct columns in panel B). Because grid search evaluates much fewer values for each parameter, there is a risk that it never encounters the value that minimizes the MSE. Thus, deploying random search (in general) results in more optimal hyperparameters (Géron, 2019).
However, both methods are completely uninformed when deciding which values for the hyperparameters to try next. As a result, they spend time and computational resources evaluating unimportant hyperparameters. The two neural networks deployed in this thesis are highly complex models with many hyperparameters to optimize and as a result, they are renowned for being tricky to train. As a result, we must test a wide range of hyperparameter values to ensure that optimal value lies within the search range, rendering grid search and random search computationally infeasible for our research. Therefore, we use Bayesian optimization, an efficient hyperparameter optimization approach, that leverages past information when deciding on which values of the hyperparameters to try next.

Bayesian Optimization

We proceed by explaining how Bayesian optimization works and how we deploy it to find the optimal hyperparameters for the machine learning models in this thesis. Bayesian optimization is used to find the extrema of a loss function, without having a closed-form expression of the function, but where *observations* of the function can be obtained. When we optimize the hyperparameters, we do not have a closed-form expression of the loss function that can lead us to the best hyperparameters. However, we can try different combinations of hyperparameters and evaluate the model performance in terms of the MSE. In this regard, we note that the Bayesian optimization is a maximization problem rather than a minimization problem. Therefore, we transform the minimization of the loss function into a maximization of an objective function by $f(x) = -g(x)$ (Brochu, Cora and De Freitas, 2010)**.**

Bayesian optimization relies on the Bayes' theorem. Put simply, Bayes' theorem suggests that the *posterior* probability of a model, M, given the evidence (or observations), E, is proportional to the likelihood of the E given the M, multiplied with the *prior* probability of M (Brochu et al., 2010):

$$P(M|E) \propto P(E|M)P(M) \tag{55}$$

The Bayesian optimization method builds a probabilistic model of the objective function[24] that follows Bayes theorem. While the actual objective function is unknown, we assume some *prior* belief about the function, such as its smoothness. The prior belief makes some possible objective functions more plausible than others. The actual objective function is tested on samples, x_i, which generates observations of the objective function, $f(x_i)$. The observations are accumulated $\mathcal{D}_{1:t} = \{x_{1:t}, f(x_{1:t})\}$, and the likelihood of these accumulated

[24] The probabilistic model of the objective function is often referred to as the surrogate function

observations, given the prior belief, is found $P(\mathcal{D}_{1:t}|f)$. For example, if the prior belief is that the objective function is smooth, noisy observations are less likely than observations similar to the mean. By multiplying the likelihood of the accumulated observations with the prior belief, we get the proportional posterior distribution:

$$P(f|\mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t}|f)P(f) \tag{56}$$

The posterior $P(f|\mathcal{D}_{1:t})$ distribution contains the *updated* belief of the objective function, based on the *prior* belief and the likelihood of the accumulated observations given the *prior* belief.

Furthermore, Bayesian optimization uses an acquisition function to efficiently locate the next sample of hyperparameters $x_{t+1} \in \mathcal{A}$. The acquisition function describes the utility for all values of the hyperparameters, according to a probabilistic model over the objective function. The best next sample is the sample with the highest utility which is found by maximizing the acquisition function. The decision of where to locate the best next sample is a utility trade-off between exploitation and exploration. Exploitation focuses on locations where the value of the objective function is expected to be high. Exploration focuses on undiscovered locations associated with high *uncertainty* (Brochu et al., 2010).
The following figure provides an example of a Bayesian optimization for a 1-dimensional problem:

Figure 7.5: Bayes optimization for a 1-dimensional problem

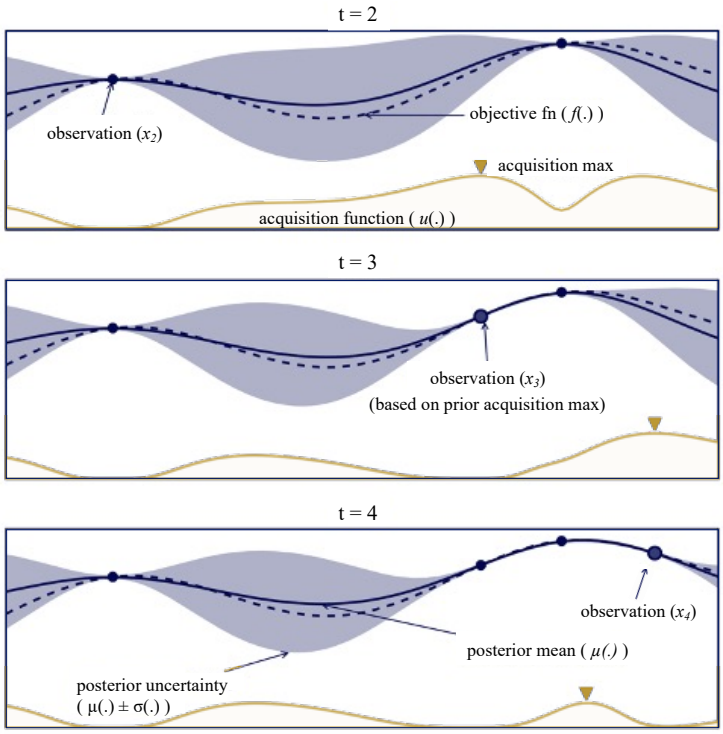


Figure 7.5 notes: The dashed line represents the actual unknown objective function. The probabilistic model of the objective function for this example is a Gaussian process, with the dark blue line as the mean function and the blue area as the uncertainty, $\mu(\cdot) \pm \sigma(\cdot)$. The yellow shaded area is the acquisition function. (Source: Brochu et al., 2010)

As shown in Figure 7.5, for each iteration, the acquisition function is maximized to find the location for the best next sample of the objective function (yellow arrow). The acquisition is high when the Gaussian process predicts a high observed objective function (exploitation) and when the prediction uncertainty is high (exploration). The objective function is sampled $f(x_t)$ at the best next location, and the probabilistic model is updated accordingly. This process is iteratively repeated until the objective function is maximized, leading to the optimal hyperparameters (Brochu et al., 2010).

In general terms, the Bayesian optimization algorithm can be articulated as (Brochu et al., 2010):
For each $t = 1, 2, \dots$, iteration

1) Find a sample, x_t, by maximizing the acquisition function (utility), based on the prior distribution (according to exploration and exploitation):

$$x_t = argmax_x u(x|\mathcal{D}_{1:t-1})$$

2) Sample the actual objective function:

$$y = f(x_t) + e_t$$

3) Accumulate the observations $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (x_t, y_t)\}$, and combine the prior distribution P(f) with the likelihood function $P(\mathcal{D}_{1:t}|f)$ to get the posterior distribution. The posterior distribution contains the updated beliefs about the unknown objective function:

$$P(f|\mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t}|f)P(f)$$

This iteration continues until the algorithm converges to the global extrema or until some stopping rule is applied.

There are two decisions to make when using Bayesian Optimization. The first decision is regarding which approximation of the objective function to use for the probabilistic model. We utilize the Gaussian process as an approximation of the objective function due to its flexibility and power, in line with extant literature (Brochu et al., 2010; Snoek, Larochelle & Adams, 2012). The second decision concerns which acquisition function to use. We incorporate three different acquisition functions that are often deployed in the literature, namely the probability of improvement, expected improvement, and the lower confidence bound (Kushner, 1964; Mockus, Tiesis & Zilinskas, 1978; Srinivas, Krause, Kakade & Seeger, 2010). We will not dwell on the specifics of the Gaussian process and the three acquisition functions, however, we provide an explanation of these in appendix A.8.

In summary, we use Bayesian optimization to optimize model hyperparameters as it is efficient from a computational resources point-of-view, enabling a wider search range compared to the grid or a random search.

7.4 Model Construction and Hyperparameter Specification

Having established how our data are preprocessed and how we optimize the hyperparameters of the machine learning models, we proceed by describing the *specific* hyperparameters deployed in each of our three models. The first subsection describes the hyperparameters of Random Forest. The subsequent subsection outlines the hyperparameters of MLP and RNN and depicts how the networks are constructed. As limited methodical literature exists on applying machine learning for stock return prediction, we rely primarily on the findings of Gu et al. (2019) when setting the search reach for our hyperparameters. The final subsection provides a table, summarizing the search range for our hyperparameters along with their optimal values according to Bayesian Optimization. Note that we do not include the linear regression (baseline) model in this subsection as it has no hyperparameters to optimize, by construction.

7.4.1 Construction of the Random Forest Model

The first machine learning model deployed in this thesis is Random Forest. The Random Forest model has been documented to be highly successful in the literature despite its simple nature (Gu et al., 2019). Random Forest requires less adaptation to a specific problem compared to neural networks, as it has fewer hyperparameters to tune. The simpler architecture of the model makes it an attractive alternative for practitioners deploying machine learning for return prediction insofar the model is capable of obtaining an equivalent performance to the neural networks. We optimize the hyperparameters that have been identified as most important in the literature (Moritz & Zimmermann, 2016).

First, we define the number of trees in the forest. Recall that the prediction of Random Forest relies on the average predictions of individual regression trees. Averaging across trees will result in a more robust model with reduced overfitting. However, there are diminishing returns of adding more trees, and the computational resources required to run the model increase with the number of trees added to the forest (Müller & Guido, 2016). Consistent with Gu et al. (2019) we set the number of trees to [300], which provides enough trees to create a robust model that is feasible to implement, given our computational resources.

Second, we specify the maximum depth of the trees in the forest. The depth of a tree depends on the number of times the nodes are split into subnodes. The more splits, the deeper the tree, and thus, the more specific information is captured in the tree. Therefore, limiting the maximum depth of the tree can reduce the overfitting of the model. We set the maximum depth of trees to the range of [1:6] to avoid the default value with no depth limitation (Gu et al., 2019).

Finally, we define the number of input variables (momentum variables) to consider in the search for the best split. This hyperparameter controls how similar the trees in the forest are. A high number of momentum variables leads to more similar trees as the trees will consider more of the same variables when finding the best split. A low number of momentum variables leads to more different trees and the trees might need to grow

deeper to fit the data. Similar to Gu et al. (2019) we define the hyperparameter ranging from three to the total number of input variables, i.e., as [3:132].

7.4.2 Construction of the MLP and RNN

As argued by Gu et al. (2019), there are many choices to make when structuring and selecting hyperparameters for a neural network. At the same time, there is little theoretical guidance on the best approach. Consequently, our rationale for the search range of the hyperparameters is somehow arbitrary and relies on a combination of literature suggestions and a "trial and error" approach. This subsection first describes the key hyperparameters for neural networks, namely the number of layers, nodes, and batch size. Second, it describes the regularization hyperparameters utilized to mitigate overfitting.

Layers

Recall that a neural network consists of an input layer, one or more hidden layers, and an output layer, the latter of which produces the prediction for the given inputs. In recent literature, there is no consensus on the number of layers to include in a neural network. Some scholars suggest that deeper networks (with as many as 152 hidden layers) can achieve the same results as shallow networks with substantially fewer parameters (Hinton, Osindero, & Teh, 2006; Eldan & Shamir, 2016; Rolnick & Tegmark, 2018). Eldan and Shamir (2016) document that depth, when increased by just one layer, can be exponentially more valuable than an increase in the width of neural networks (adding more nodes). In contrast, Gu et al. (2019) demonstrate that a shallower network is superior to deeper networks in their study. Training a very deep neural network is also highly challenging: It entails many parameters, because the loss function is highly non-convex, and because back-propagation entails recursive calculation of the gradients which values can explode or vanish (we explain this further below, see batch normalization). Gu et al. (2019) find that when predicting stock returns, their best performing neural network has three layers. Consequently, we choose to incorporate architectures for both MLP and RNN with up to three hidden layers. We start by building a simple model with just one hidden layer and subsequently increase the number of hidden layers to test if it improves model performance.

Nodes

A common practice in the literature is to select a number of nodes that form a funnel by reducing the number of nodes at each subsequent layer. The rationale for this approach is that many low-level input variables can coalesce into far fewer but high-level input variables (Géron, 2019). In our example, we have 132 input nodes. In line with common practice, we define the search range for the number of nodes in the first hidden layer so that it does not exceed the number of input nodes. Specifically, we define the range [10:132] for the first hidden layer. For each subsequent layer, we bisect the number of nodes to form a funnel, which resembles the geometric pyramid rule[25], as proposed by Masters (1993). All layers are fully connected so each node receives an input from all nodes in the previous layer.

[25] The nodes in a hidden layer are given by $\sqrt{NxM}$ nodes where n is the nodes of the previous layers, and m is the nodes of the subsequent layer

Batch Size

During training, neural networks do not utilize the total number of samples at the same time, but rather use smaller fractions of the total sample, called batches. The batch size defines the number of samples to work through before the internal parameters (e.g., the weights) of the model are updated (Goodfellow et al., 2017). For our specific problem, it is important that the batch size is 500, because we rank the stock variables cross-sectionally, month-by-month. Placing the firms of a given month into different batches would make it difficult for the network to understand potential patterns in the dataset. Consequently, we do not create a search range for optimizing the batch size hyperparameter but specify it as [500]. Specifically for the RNN, we reset the long- and short-term states in the LSTM cells after each batch (i.e., a stateless LSTM). For our problem, it is crucial that the state of the LSTM cells is independent from the previous batches, as the same firms are not part of the investment universe each month.

Activation Function

Referring to subsection 4.3.2, an activation function transforms the output of a given layer to an "activation output" before the output is passed as input to the next layers. The choice of activation function results in different activation outputs and thus, affects network training. There are many choices of nonlinear activation functions. The LSTM cell deploys a sigmoid and tanh activation function which we rely on. For the MLP, we use the same activation function at all nodes, namely the ReLU. We deploy this function, as it efficiently makes the network converge towards a solution, and is widely deployed in recent literature (Glorot, Bordes & Bengio, 2011; Feng, He & Polson, 2018; Gu et al., 2019).

Batch Normalization

We apply batch normalization to the MLP[26]. Recall that a neural network is trained through backpropagation, where gradients are computed with respect to each parameter, and the parameters are updated through gradient descent. If the network ends up creating vanishingly small gradients, then the update to the parameter at each subsequent training step is going to be vanishingly small as well. As a result, the newly updated parameter will barely move from its prior value. Moreover, the update to the parameter will have a vanishing effect when carried through the network, ultimately not helping to reduce the loss. The same principle goes for exploding gradients, however, instead of the parameter barely changing, the update at each training step will become too large, and the parameter will continue to move further and further away from its optimal value (Géron, 2019). Batch normalization was proposed by Ioffe and Szegedy (2015) to counter the vanishing/exploding gradient problem and is a simple technique for controlling the variability of outputs in different layers of a network. Note that batch normalization is applied to a network layer on a per batch basis. Batch normalization consists of two overarching steps. First, when applying batch normalization to a layer, the algorithm cross-sectionally de-means and standardizes the variance of the activation outputs from the previous layer. The algorithm does so by computing the mean and standard deviation of all activation outputs over the current batch. Let $\mathbf{X}$ be the activation output from the previous layer, l. To normalize $\mathbf{X}$, we replace it with $\mathbf{X}^*$:

[26] We do not deploy batch normalization to RNNs as batch normalization does not consider the recurrent part of the network

$$\mu_B = \frac{1}{m_B}\sum_{l=1}^{m_B} \boldsymbol{X_l} \tag{57}$$

$$\sigma_B^2 = \frac{1}{m_B}\sum_{l=1}^{m_B} (\boldsymbol{X_l} - \mu_B)^2 \tag{58}$$

$$\boldsymbol{X_l^*} = \frac{\boldsymbol{X_l} - \mu_B}{\sqrt{\sigma_B^2 + e}} \tag{59}$$

where μ_B is the empirical mean, evaluated over the whole batch B, σ_B is the empirical standard deviation, evaluated over the whole batch, m_B is the number of instances in the batch and e is a smoothing term (a small number to avoid division by zero, typically 10^{-3})

Second, the algorithm multiplies the normalized output X* by some arbitrary scaling parameter, γ, and adds another arbitrary offset parameter, β, to the product. Denote z_l the final output of the batch normalization algorithm, then:

$$\boldsymbol{z_l} = \gamma \boldsymbol{X_l^*} + \beta \tag{60}$$

Applying the two parameters scales and shifts the activation with a new standard deviation and mean. During model training, the two parameters γ (scale) and β (offset) are optimized together with other internal parameters (such as the weights) through backpropagation. The values are optimized so the normalized activation outputs best correct weight imbalances within the network (Géron, 2019).

Note that the scaling process mentioned in section 7.2 scales the variables before they are passed to the network. In contrast, batch normalization scales the activation outputs for the individual layers *within* a model. Consistent with most literature, we apply both techniques to optimize model training (Gu et al., 2019).

Regularization

The high degree of nonlinearity and nonconvexity in neural networks, together with their rich parameterization, makes neural networks both highly computationally intensive and prone to overfitting. Consistent with the literature, we deploy several regularization techniques to enhance the performance of the network (Chen et al, 2020; Gu et al., 2019). Specifically, we leverage three popular regularization techniques: early stopping, dropout, and the learning rate. We elaborate on each of the regularization parameters below.

Epochs and Early Stopping

The number of epochs defines the number of complete passes through the entire training set (i.e., all the batches). As commonly done within the field of machine learning (Goodfellow et al., 2017), we implement early stopping to find the number of training epochs. Too many (few) epochs can lead to overfitting (underfitting) of the training dataset. By utilizing the early stopping method, we simply set the epochs to a large number [300] and stop the training of the model, when the model performance stops improving on the validation set (Goodfellow et al., 2017). More specifically, we use MSE as the error measure and stop the

algorithm if there has not been an 0.0001 absolute decrease in the MSE after 10 consecutive epochs. Apart from mitigating overfitting, early stopping also makes sense from a limited computational resources point-of-view.

Dropouts

In recent literature, one of the most popular regularization techniques for neural networks is dropouts, proposed by Srivastava et al. (2014). Networks utilizing dropouts are less sensitive to slight changes in the inputs and are more robust networks with a better ability to generalize. During *training*, nodes in a given layer (excluding the output nodes) have a probability of being kept (p) or of being temporarily dropped out ($1-p$). One node may be ignored during one training batch, while possibly being active during the next.

Figure 7.6: Dropout of neural networks

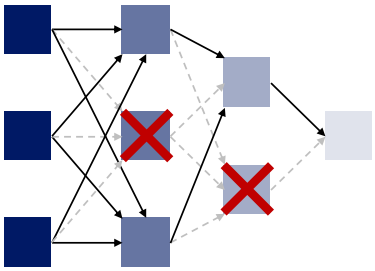


Figure 7.6 notes: The figure depicts an example of a neural network that deploys dropouts. The network has two hidden layers, and one neuron is (temporarily) dropped out in each hidden layer. (Source: Own creation based on Géron, 2019).

The hyperparameter $(1-p)$ is referred to as the dropout rate, and we set this to a range of [0:0.7] for our networks in line with common practice (Géron, 2019). Note that because dropouts are only applied during training, more nodes are present in the network at the same time during testing. As a result, the strength of the activation outputs of the nodes during testing is higher than the activation outputs of the nodes during training. For instance, if the dropout rate is 50% during training, then a node during testing will be connected to twice as many nodes as it was (on average) during training. To counter this effect, a technique referred to as inverted dropout is applied. Specifically, during training of a neural network, nodes are first randomly dropped (with probability 1-p), and then the values of the activation outputs of the kept nodes are divided by the keep rate, p. As a result, no changes are required to the network during testing (Géron, 2019).

A more traditional approach that resembles the effect of dropouts is deploying L2 regularization to the loss function which penalizes large weights. We restrain ourselves from deploying both approaches simultaneously, as the two do not work well collectively (Phaisangittisagulm, 2016; Brownlee, 2017). Moreover, Chen, et al. (2020) argue that dropouts are preferable and generally result in better performances.

Optimization Algorithm and Learning Rate

It is common to train neural networks using the backpropagation algorithm with variations of stochastic gradient descent (SGD) for optimizing the weights in the network (Goodfellow et al., 2017). A popular optimization algorithm for neural networks in recent literature is the adaptive moment estimation algorithm (Adam), which is an efficient adaptation of the SGD, introduced by Kingma and Ba (2014). Consistent with

the literature, we apply Adam in this thesis (Chen et al., 2020; Gu et al., 2019). As subsection 4.1.4 elaborated on the gradient descent optimization algorithm, we do not dwell on the technical details of Adam. Instead, we refer to appendix A.9 which provides a more in-depth explanation of the algorithm.

While Adam is generally regarded as being fairly robust to the choice of hyperparameters, Goodfellow et al. (2017) point out that altering the learning rate from its default value can be beneficial, as the learning rate can impact model performance substantially. If the learning rate is set too high, the Adam algorithm risks shooting past the minimum of the loss function. In contrast, if the learning rate is set too low, the algorithm might take too long to minimize the loss. The learning rate is typically set to a value < 0.1 (Goodfellow et al., 2017). Through trial-and-error, we observed that our models did not perform well with high learning rates. Consequently, we define our search range as $[10^{-6}: 10^{-2}]$ for the MLP and $[10^{-5}: 10^{-2}]$ for the RNN.

Neural Network Construction

We summarize this subsection by visualizing the process of constructing our neural networks (Figure 7.7). For simplicity, we show only one of our neural networks, the RNN. However, the process is identical for MLP, except that we add batch normalization, and the layers do not contain LSTM cells.

Figure 7.7: Construction of the RNN

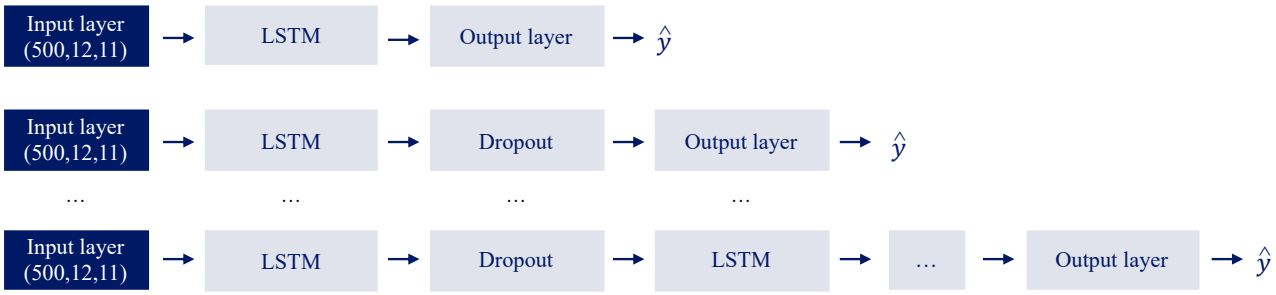


Figure 7.7 notes: The figure illustrates the process of constructing the RNN. We initiate with one hidden layer with LSTM cells. Then, we increase the complexity of the network by adding a layer with dropouts. We proceed by increasing the complexity of the network by adding additional layers, repeatedly, until we reach three hidden layers with LSTM cells that all have layers with dropouts in between. (Source: Own creation).

The process for constructing the network is as follows:

1) We initiate by creating an RNN in its simplest form with only one hidden layer with LSTM cells. We set the batch size to 500 to account for the number of firms in our universe at a given month. Our *training* sample consists of 45 years of data, and therefore 540 batches (45 years x 12 months) are fed to the network during training. We use Bayesian optimization to optimize the hyperparameters using the hold-out validation set and compare the predictive performance of the model on our training and validation data.
2) Having trained the model with the simplest architecture, we proceed by increasing model complexity. We add an additional layer with dropouts (and batch normalization for MLP) to avoid overfitting and to enhance model performance. Again, we optimize the hyperparameters using the hold-out validation set and compare the predictive performance of the model on our training and validation data.

3) Steps 1 and 2 continue. We proceed until we have three hidden layers of LSTM cells that all have layers with dropouts in between. We choose the model with the best performance on the *validation set.*

7.4.3 Defining the Optimal Models

The above subsections explained the methodology for optimizing model hyperparameters as well as the specific hyperparameters of each of the three machine learning models that we deploy in this thesis. We summarize this subsection with an overview of the search range for the hyperparameters as well as the optimal hyperparameters that are identified through Bayesian optimization. Note that the models with the optimal hyperparameters are the ones applied on the out-of-sample dataset for evaluation.

Table 7.1: Search ranges and optimal hyperparameters

Random Forest	Search range			Optimal value Random Forest
	Lower		Upper	
Number of trees				300
Maximum depth of trees	1	-	6	1
Number of variables for best split	3	-	132	3

MLP and LSTM	Search range			Optimal value MLP	Optimal value LSTM
	Lower		Upper		
Hidden layers	1	-	3	3	3
Neurons in first layer	10	-	132	10	132
Batch size				500	500
Activation function				ReLU	Sigmoid, tanh
Epochs				300	300
Dropout rate	0	-	0.7	0.39	0.70
Optimization algorithm				Adam	Adam
Learning rate	10^{-6} (10^{-5})	-	10^{-2}	$7*10^{-4}$	$1*10^{-5}$
Early stopping				√	√
Batch normalization				√	

Table 7.1 notes: The table presents the search range for each of the hyperparameters, and the optimal values identified through Bayesian optimization for each model. For the RNN, the lower search range value is reported in brackets. For early stopping and batch normalization, the tick mark denotes implementation during training. (Source: Own creation).

7.5 Evaluation of the Machine Learning Models

Having constructed the machine learning models with the optimal hyperparameters, this section turns to the method used for evaluation of the models when applied on the out-of-sample test set. In the first subsection, we present two statistical measures used to evaluate the predictive power of the *individual* machine learning models. In the second subsection, we unfold a method for *pairwise comparison* of the predictive power of the machine learning models. Lastly, the third subsection outlines the method for examining the importance of the input variables for the machine learning models.

7.5.1 Predictive R^2 and Spearman Correlation

Predictive R^2

We deploy the predictive R^2 coefficient to assess the predictions of the excess returns made by our machine learning models relative to the target returns. The R^2 aggregates the prediction error across excess stock returns and over time into one single figure that expresses the predictive power for each of the three machine learning models. We calculate the out-of-sample R^2 following the methodology of Gu et al. (2019):

$$R^2_{OOS} = 1 - \frac{\sum_{(i,t)}\left(R_{i,t} - \widehat{R}_{i,t}\right)^2}{\sum_{(i,t)}\left(R_{i,t}\right)^2} \tag{61}$$

where $R_{i,t}$ denotes the target excess return and $\widehat{R}_{i,t}$ is the predicted excess return for a single stock *i* at time *t*

Note that the denominator of the out-of-sample R^2 metric is the sum of squared excess returns without demeaning. Resonating Gu et al. (2019), assuming a mean of zero is more sensible when predicting excess stock returns, as the historical mean is too noisy.

We briefly note that the machine learning models are optimized by minimizing the MSE. However, we use the similar metric R^2 instead of the MSE measure to *evaluate* the out-of-sample performance of the machine learning models to ensure that our results are comparable with the literature (Gu et al. 2019).

Spearman Rank Correlation

The Spearman rank correlation is deployed as an additional coefficient to examine the *rank* correlation between the predicted excess returns of the machine learning models and the target excess returns. R^2 is used as a measure of the ability of the machine learning models to predict excess returns but it relies on the predictions of individual stocks. However, the ability to approximate an individual excess stock returns does not necessarily imply that the model can successfully predict the *cross-section* of excess stock returns. As understanding the cross-section of excess stock returns constitutes the foundation for utilizing predictions for stock-selection strategies, we also deploy the Spearman rank correlation. Consequently, the rationale for utilizing the Spearman rank correlation is to bridge the chasm between machine learning predictions and stock-selection strategies.

We calculate the Spearman correlation by converting the predicted and target excess returns into ranks on a month-by-month basis. Each stock *i* receives two ranks, R_i and $\hat{R}_i$ for its target and predicted excess returns, respectively. For the total sample period T, the spearman rank correlation between the two ranks is calculated as:

$$\rho_s = \frac{1}{T}\sum_{t=1}^{T} 1 - \frac{6\sum d_{i,t}^2}{B(B^2-1)} \tag{62}$$

where B denotes the batch size (in our case 500 stocks), t denotes one month and
where $d_i = \mathcal{R}_i - \hat{\mathcal{R}}_i$, for $i = (1,..,500)$

If the model is successful in predicting excess returns in the cross-section, the Spearman correlation between the predicted and target excess returns will be positive.

7.5.2 Diebold-Mariano Test

We deploy the Diebold and Mariano (1995) test to make pairwise comparisons of the machine learning models' performance. The null hypothesis of the statistical test is that there is "no difference in the accuracy of two competing forecasts"[27] (Diebold & Mariano, 1995). A condition of the Diebold-Mariano test is that the forecast errors have weak dependence. Resonating Gu et al. (2019), it is unlikely that the condition of weak error dependence underlying the Diebold Mariano test is fulfilled when predicting excess stock returns, due to a potentially strong dependence in the cross-section. Consequently, we follow Gu et al. (2019) and adjust the test to a cross-sectional setting by comparing the *cross-sectional average* of the prediction errors rather than stock level prediction errors. We calculate a single time series of error differences, d_{12}, as

$$d_{12,t} = \frac{1}{B_t}\sum_{i=1}^{B}\left(\left(\hat{e}_{i,t}^{(1)}\right)^2 - \left(\hat{e}_{i,t}^{(2)}\right)^2\right) \tag{63}$$

where $\hat{e}_{i,t}^{(1)}$ and $\hat{e}_{i,t}^{(2)}$ are the prediction errors for stock i at time t for model 1 and 2, respectively, and B is the batch size (for our case 500 stocks)

The modified Diebold-Mariano test statistic is given by

$$DM_{12} = \frac{\bar{d}_{12}}{\hat{\sigma}_{d_{12}}} \tag{64}$$

where $\bar{d}_{12}$ denotes the mean and $\hat{\sigma}_{d_{12}}$ denotes the Newey-West standard error of $d_{12,t}$ with a maximum lag of 1 month

Diebold-Mariano statistics are normally distributed, *N(0;1),* under the null hypothesis that the predictive power of the models does not differ. As a result, the test statistic maps to p-values in a similar fashion as t-statistics in a regression model.

[27] Note that we utilize MSE as the accuracy metric to compare the predictions of the machine learning models

7.5.3 Importance of Input Variables

Having established the methodology for examining the predictive performance of the machine learning models, we proceed by laying out the methodology for analyzing what *explanation* underpins their performance. Machine learning models are renowned for making powerful predictions but also for being difficult to interpret. In an attempt to improve interpretability, we investigate which input variables are the most important for the predictions of each model. There are several different methods for calculating the importance of the input variables. We deploy the commonly utilized method of SHapley Additive exPlanations (SHAP), introduced by Lundberg and Lee in 2016. The SHAP method unifies six previous methods[28] and shows improved computational performance and better consistency with human intuition than previous approaches (Lundberg & Lee, 2016). Explaining these six complex methods and how the SHAP method unifies them is beyond the scope of this thesis, and thus, we focus on the *intuition* behind the method in this subsection.

In simple terms, the SHAP method explains which input variables a model relies on when making predictions. The SHAP method assigns a "SHAP value" to each input variable that expresses the importance of the variable for the predictions of the model. More specifically, the process for assigning the SHAP values to each input variable is as follows: As depicted in Figure 7.8, the model starts with some base value, $E[f(z)]$, which is the expected predicted value without any known input variables. Then, one input variable, x_1, is incorporated into the model which influences the expected predicted value and shifts it to $E[f(z)|z_1 = x_1]$. The influence of incorporating this input variable x_1 is denoted, ϕ_1. Then, input variable x_2 is incorporated into the model, and the expected predicted value is shifted by ϕ_2 to $E[f(z)|z_{1,2} = x_{1,2}]$. The process of incorporating individual input variables x_i continues, until all the known variables M are included (for our case, the eleven momentum variables). For the example in Figure 7.8, the total number of input variables is four. Once all the input variables have been incorporated into the model, the expected predicted value becomes the final predicted value, $f(x)$.

Figure 7.8: SHAP values of input variables (illustrative)

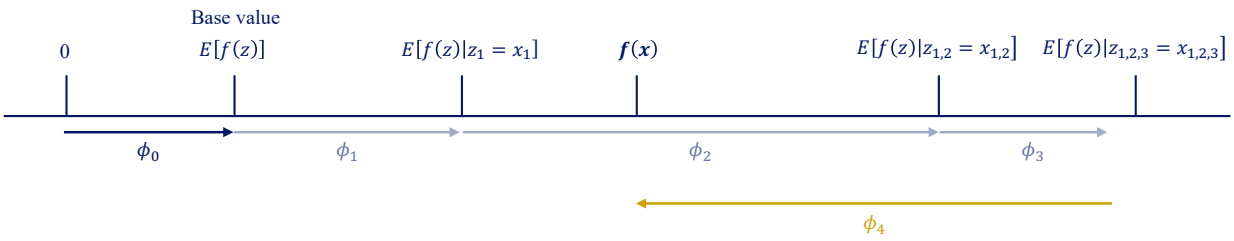


Figure 7.8 notes: The figure illustrates a simplified example of the input variables' SHAP values for one prediction made by a machine learning model. In this example, variables one, two, and three increase the prediction, while variable four decreases the prediction. Variables two and four have the highest importance for the prediction (they have the highest ϕ). (Source: Own creation).

In general, the mapping from the base value to the prediction of the model, $f(x)$, can be expressed as:

[28] The SHAP method combines six previous methods into one unified approach, namely LIME, DeepLIFT, Quantitative Input Influence, Layer-Wise Relevance Propagation, Shapley Regression Value and Shapley Sampling Values

$$E[f(z)] + \sum_{i=1}^{M} \phi_i = f(x) \tag{65}$$

The SHAP value of each input variable x_i is calculated based on the ϕ_i in the mapping between the base value and the final predicted value. However, the order in which the input variables are incorporated into the model affects the size of ϕ_i, when the model is non-linear, and the input variables are not independent (due to interaction effects between the input variables). Consequently, the SHAP value is calculated by averaging the ϕ_i value across all the possible orderings of the input variables.

Note that when we deploy the SHAP method, we compute the SHAP values of the input variables five times for each model (recall that we counter randomness in the results by running each of the three machine learning models five times). We base the analysis on (five) random subsamples of 5000 predictions, as it is computationally infeasible to calculate the importance of the input variables on the entire out-of-sample test set. Finally, we construct the *relative* importance of each input variable by normalizing their values to the range of [0:1] in our result, similar to Gu et al. (2019).

In summary, we deploy the R^2, Spearman rank correlation, Diebold-Mariano test and SHAP value to evaluate the predictive power and underlying drivers of the machine learning *models*. However, these metrics do not contain information on the economic contribution of the machine learning-based *strategies* and thus, we also include metrics that capture this aspect.

8 Evaluation of Stock-Selection Strategies

This chapter provides an overview of how we evaluate the *economic* performance of our stock-selection strategies. In the first section, we describe four performance metrics utilized for evaluating the performance of the stock-selection strategies. In the second section, we outline how we perform an OLS regression of the three machine learning-based stock-selection strategies against common risk factors in the literature.

8.1 Performance Metrics

Building on the insights from the asset pricing models (section 3.3), we use four metrics to evaluate the performance of the stock-selection strategies, namely alpha, Sharpe Ratio, Sortino Ratio and Drawdown. This enables us to compare the performance of the machine learning-based stock-selection strategies with the performance of the conventional momentum strategy. We utilize alpha and the Sharpe ratio, as these are common performance metrics in the literature, enabling us to compare our findings to that of similar studies. However, as the research of this thesis focuses on periods of market stress, we also investigate performance measures that specifically shed light on downside risk, namely the Sortino ratio and drawdown of the returns.

Alpha

Alpha is a measure of the systematic risk-adjusted excess return (Jensen, 1967). Recall that the CAPM dictates that alpha is equal to zero for any stock or portfolio. Therefore, if an investment strategy yields a positive alpha, it earns higher returns than simple compensation for systematic risk and thus, the strategy defies the CAPM (Pedersen, 2015). We estimate alpha as the intercept of the CAPM regression model over the sample period:

$$R_p - R_f = \alpha_p + \beta_p\left[R_m - R_f\right] + \mathrm{e_p} \tag{66}$$

where R_p denotes the portfolio return, R_m is the market return, R_f is the risk-free rate, β_p is the beta of the portfolio, and e_p is the zero-mean residual

Sharpe Ratio

The Sharpe ratio was proposed by Sharpe (1966) and also relies on the mean-variance theory. In contrast to alpha that adjusts excess returns for systematic risk only, the Sharpe ratio adjusts the excess return of a portfolio for its total risk. Hence, the Sharpe ratio measures the reward per unit of risk (Pedersen, 2015):

$$\text{Sharpe ratio} = \frac{\mathrm{R_p} + -\mathrm{R_f}}{\sigma_\mathrm{p}} \tag{67}$$

where σp denotes the standard deviation of the portfolio

Using the total risk of the portfolio implies that the Sharpe ratio is only valid for returns that are (approximately) normally distributed. In other words, the Sharpe ratio is a useful performance measure when the risk of a portfolio can be adequately measured by its standard deviation. In contrast, it may result in misleading conclusions when return distributions are skewed (Bernardo & Ledoit, 2000). Daniel and Moskowitz (2016) document that the conventional momentum strategy is tempered by negatively skewed returns. The infrequent but substantial negative returns entail large losses during market rebounds. Consequently, we combine the Sharpe ratio with complementary performance measures concentrating on downside risk.

Sortino Ratio

The Sortino ratio was introduced by Sortino (1994) and is a measure of the excess returns of a portfolio, adjusted for the *downside* risk, rather than the total standard deviation of portfolio returns. As a result, the Sortino ratio provides a more accurate result than the Sharpe ratio in periods of highly skewed returns (Sortino & Stachell, 2001). We calculate the Sortino ratio as:

$$\text{Sortino ratio} = \frac{\mathrm{R_p} + -\mathrm{R_f}}{\sigma_{\mathrm{p,d}}} \tag{68}$$

where $\sigma_{\mathrm{p,d}}$ denotes the downside volatility of the portfolio

Drawdown and Maximum Drawdown

We extend our analysis of the downside risk by incorporating the drawdown and maximum drawdown metrics. These metrics are deployed to investigate the periods with the most substantial losses across our sample period.

The drawdown is the cumulative loss from a historical peak (where a loss started) to the current value (Pedersen, 2015). We calculate the percentage drawdown since the peak (HWM) as:

$$\mathrm{DD_t} = \frac{\mathrm{HWM_t} - \mathrm{P_t}}{\mathrm{HWM_t}} \tag{69}$$

where P_t is the cumulative return at time t
If the current value < historical peak, the drawdown > 0.
If the current value >= the historical peak, the drawdown = 0.

The maximum drawdown is an expression of the largest drawdown, i.e., loss, observed from a peak to a trough until a new peak is reached (Pedersen 2015):

$$\mathrm{MDD_t} = \max_{t \leq T} \mathrm{DD_t} \tag{70}$$

8.2 Regression against Risk Factors

This thesis further performs OLS regressions of the machine learning-based stock-selection strategies against seven risk factors. This section consists of three subsections: First, we briefly explain how the regression analyses are performed. Then, we explain the risk factors included in the regression analyses. Finally, we introduce the information ratio which we apply as a performance metric based on the regression analyses.

8.2.1 OLS Regressions

We perform OLS regressions (spanning tests) for each of the three machine learning-based stock selection strategies. We denote the daily excess returns of the seven risk factors as the independent variable and the daily excess returns of the machine learning-based stock-selection strategies as the dependent variable. As we conduct the OLS regressions on time series data, we implement the variance-covariance estimator proposed by Newey and West (1987) to ensure robustness against heteroskedasticity and autocorrelation in the error terms of the regression models. By performing such regression analyses, we are able to examine the factor loadings of the three strategies. Note that we also report the R^2 for the regression models, which is different from the R^2 utilized to examine the predictive power of the machine learning models (subsection 7.5.1). Rather, the R^2s for the regression models express to which degree the variance of the excess returns of the machine learning-based stock-selection strategies can be explained by the variance of the excess returns of the risk factors.

8.2.2 Risk Factors

Having established how we perform the OLS regressions, we briefly explain the risk factors that we use as independent variables in the regressions. While the existence of a variety of different factors has been documented during recent years, not all these factors have been replicable post-publication (Harvey, Liu, and

Zhu, 2016; Arnott, Harvey, Kalesnik and Linnainmaa, 2019), and many are intertwined with high correlations (Jensen, Kelly and Pedersen, 2021). We select some of the most profound risk factors in the literature, including the Fama/French five factors (Fama & French, 2015) as well as the betting against beta factor (Frazzini & Pedersen, 2014). These six risk factors are all based on the US stock market which closest resemble our investment universe. This section describes how we obtain these factors, while the theory underpinning them can be found in section 3.3.
Note that we also include the excess returns from our replicated momentum strategy in the OLS regressions. The rationale for including our momentum "factor" is to examine whether the momentum variables utilized as inputs for the machine learning models infer a loading on the factor.

The Fama/French five factors
The five factors are computed via six value-weighted portfolios formed according to size (market equity) and book-to-market ratio, six value-weighted portfolios formed according to size and operating profitability, and finally, six value-weight portfolios based on size and investment. Thus, the factors rely on 18 portfolios which are presented in Table 8.1:

Table 8.1: 18 Fama/French research portfolios

Size and Book-to-Market	Size and operating profitability	Size and investment
Small Value	Small Robust	Small Conservative
Small Neutral	Small Neutral	Small Neutral
Small Growth	Small Weak	Small Aggressive
Big Value	Big Robust	Big Conservative
Big Neutral	Big Neutral	Big Neutral
Big Growth	Big Weak	Big Aggressive

(Source: Own creation based on the Kenneth French Data Library)

The five factors are calculated as follows:

1) The market factor, $(R_m - R_f)$, is formed as the value-weighted return of all the stocks minus the one-month Treasury bill rate.

2) The SMB factor is calculated as the difference between the average return on the nine small stock portfolios and the average return on the nine big stock portfolios. This can be expressed as:

$$\text{SMB}_{\left(\frac{\text{B}}{\text{M}}\right)} = \frac{1}{3}(\text{Small Value} + \text{Small Neutral} + \text{Small Growth}) - \frac{1}{3}(\text{Big Value} + \text{Big Neutral} + \text{Big Growth}) \quad (71)$$

$$\text{SMB}_{(\text{OP})} = \frac{1}{3}(\text{Small Robust} + \text{Small Neutral} + \text{Small Weak}) - \frac{1}{3}(\text{Big Robust} + \text{Big Neutral} + \text{Big Weak}) \quad (72)$$

$$\text{SMB}_{(\text{INV})} = \frac{1}{3}(\text{Small Conservative} + \text{Small Neutral} + \text{Small Aggressive}) - \frac{1}{3}(\text{Big Conservative} + \text{Big Neutral} + \text{Big Aggressive}) \quad (73)$$

$$\text{SMB} = \frac{1}{3}\text{SMB}_{(\frac{\text{B}}{\text{M}})} + \frac{1}{3}\text{SMB}_{(\text{OP})} + \frac{1}{3}\text{SMB}_{(\text{INV})} \quad (74)$$

3) The HML factor is the difference between the average return on the two value portfolios and the average return on the two growth portfolios:

$$\text{HML} = \frac{1}{2}(\text{Small Value} + \text{Big Value}) - \frac{1}{2}(\text{Small Growth} + \text{Big Growth}) \quad (75)$$

4) The RMW factor is the difference between the average return on the two robust operating profitability portfolios and the average return on the two weak operating profitability portfolios:

$$\text{RMW} = \frac{1}{2}(\text{Small Robust} + \text{Big Robust}) - \frac{1}{2}(\text{Small Weak} + \text{Big Weak}) \quad (76)$$

5) Finally, CMA is calculated as the difference between the average return on the two conservative investment portfolios and the average return on the two aggressive investment portfolios:

$$\text{CMA} = \frac{1}{2}(\text{Small Conservative} + \text{Big Conservative}) - \frac{1}{2}(\text{Small Aggressive} + \text{Big Aggresive}) \quad (77)$$

Betting Against Beta

The Betting Against Beta (BAB) factor is obtained from the website of AQR Capital Management. The factor is constructed by taking a long position in portfolios containing low beta securities, and short-selling portfolios containing high-beta securities[29]. The securities are cross-sectionally ranked on the basis of their estimated betas and assigned a weight so that the lower-beta securities have larger weights in the low-beta portfolio, while higher-beta securities have larger weights in the high-beta portfolio.

Denote z as the $nx1$ vector of beta ranks $z_i = rank(ß_{it})$ at portfolio formation, and let $\bar{z} = 1_n' z/n$ be the average rank where n is the number of securities, and where 1_n is a nx1 vector of ones. The portfolio weight of the two high and low beta portfolios can then be described as:

[29] Note that Frazzini & Pedersen (2014) use ex-ante betas that rely on estimated volatilities for the stock and the market as well as their correlation

$$w_H = k(z - \bar{z})^+ \tag{78}$$

and

$$w_L = k(z - \bar{z})^- \tag{79}$$

where k is a normalizing constant $k = 2/1_n'|z - \bar{z}|$ and
x^+ and x^- indicate the positive and negative elements of a vector, x.

The final BAB-factor can be defined as:

$$R_{t+1}^{BAB} = \frac{1}{\beta_t^L}\left(R_{t+1}^L - R_f\right) - \frac{1}{\beta_t^H}\left(R_{t+1}^H - R_f\right) \tag{80}$$

where $R_{t+1}^L = R_{t+1}'w_L$, $R_{t+1}^H = R_{t+1}'w_H$, $\beta_t^L = \beta_t'w_L$, and $\beta_t^H = \beta_t'w_H$

Note that to construct the BAB factor, the long and short portfolios are rescaled to have a beta of one at portfolio formation (Frazzini & Pedersen, 2014).

8.2.3 Information Ratio

Lastly, we introduce the information ratio which we utilize in relation to the regression analyses. The information ratio expresses the risk-adjusted abnormal return relative to a given benchmark. For our case, the benchmark for the excess returns of the machine learning-based stock-selection strategies is the excess returns of the seven risk factors. The information ratio is given by (Pedersen, 2015):

$$IR = \frac{\alpha}{\sigma(e)} \tag{81}$$

where $\sigma(e)$ is the volatility of the error and both $\sigma(e)$ and α are obtained from regressing the excess returns of the machine learning-based stock-selection strategies against the excess returns of the seven risk factors

We apply the information ratio in this thesis to test if the risk factors can explain the performance of the machine learning-based stock selection strategies (in which case the information ratio should be close to zero).

9 Analysis of the Momentum Strategy and Machine Learning-Based Stock-Selection Strategies

On the basis of the methodology described in chapters 5 to 8, this chapter presents the analysis and results of this thesis. We aspire to answer four of our research questions (two to five) throughout the different sections in this chapter. Note that this chapter presents the results and offers an interpretation of our findings, while an in-depth discussion hereof is conducted in the next chapter.

We present the analysis and results of the thesis in a six-fold structure. The first section outlines the performance of our replicated conventional momentum strategy. The second section investigates the predictive power of the machine learning models, individually and pairwise. Subsequently, the third section presents the machine learning-based stock-selection strategies and examines their performance relative to the momentum

strategy, over time and during periods associated with substantial crash risk. Section four introduces an ensemble strategy that combines the three machine learning-based stock-selection strategies with the momentum strategy. The fifth section dissects the performance of the machine learning-based stock-selection strategies by analyzing the importance of the input variables when the machine learning models make predictions. The sixth section adopts a different perspective for investigating the performance of the machine learning-based stock-selection strategies, by examining their loadings on seven risk factors. We finalize the chapter by synthesizing our findings.

Henceforth, we refer to the conventional momentum strategy as the "MOM" strategy (and its winner-minus-loser portfolio as "WML"). For simplicity, we refer to the machine learning-based stock-selection strategies as "MLS" strategies. Further, we refer to the individual MLS strategies, Random Forest, Multilayer Perceptron, and Recurrent Neural Network with Long Short-Term Memory architecture as the "RAF", "MLP" and "RNN" strategy, respectively. The linear regression model applied as a baseline model is denoted "LR".

9.1 Replicating the Conventional Momentum Strategy

As previously established, a variety of scholars have documented a momentum effect in the US market. We initiate this section by replicating the conventional 2-12 months MOM strategy to ensure that the renowned momentum premium, as well as the crash risk inherent to the strategy, are also evident in *our* sample, which differs from those in the literature. Thus, this section seeks to answer research question two, related to the performance of the MOM strategy over time and during periods of market stress. First, we explore the performance of the MOM strategy for the total sample period from January 1931 to December 2020. Subsequently, we zoom in on sub-periods and examine the crashes of the MOM strategy. We note that while we provide an overview of various descriptive statistics of the MOM strategy, we only comment on the most relevant statistics for the research of this thesis.

Table 9.1: Performance of the conventional momentum strategy for the total sample period

	Momentum portfolios		
	Winner	Loser	WML
$\overline{r - r_f}$	16.51	5.75	7.72
$t(\overline{r - r_f})$	(7.49)	(3.39)	(4.59)
σ	23.21	24.42	19.88
α	9.18	-1.10	10.39
$t(\alpha)$	(6.97)	(-0.83)	(4.81)
β	1.17	1.22	-0.06
Sharpe ratio	0.78	0.35	0.47
Sortino ratio	1.09	0.52	0.64
Maximum drawdown	60.42	85.64	75.14

Table 9.1 notes: This table provides descriptive statistics of the excess returns of the winner, loser, and WML momentum portfolios.

For each portfolio, we report the following metrics: The annualized return in excess of the risk-free rate, volatility, and alpha, in percent; the market beta; the annualized Sharpe ratio and Sortino ratio; and the maximum drawdown in percent. The t-statistics are reported in parenthesis and are for the null hypothesis of the metrics being equal to zero. The sample period spans January 1931 to December 2020. Portfolios are equal-weighted and reformed monthly. (Source: Own creation).

Table 9.1 reports a significant excess return of 7.72% for the WML portfolio, as a result of a significantly higher excess return of the winner portfolio relative to the loser portfolio (t-stat = 6.97). The beta of the WML is negative (-0.06), providing the portfolio a significant CAPM annualized alpha of 10.39%, and thus, confirming a significant momentum premium relative to the market.

Figure 9.1: Cumulative returns of the WML portfolio for the total sample period and momentum crashes

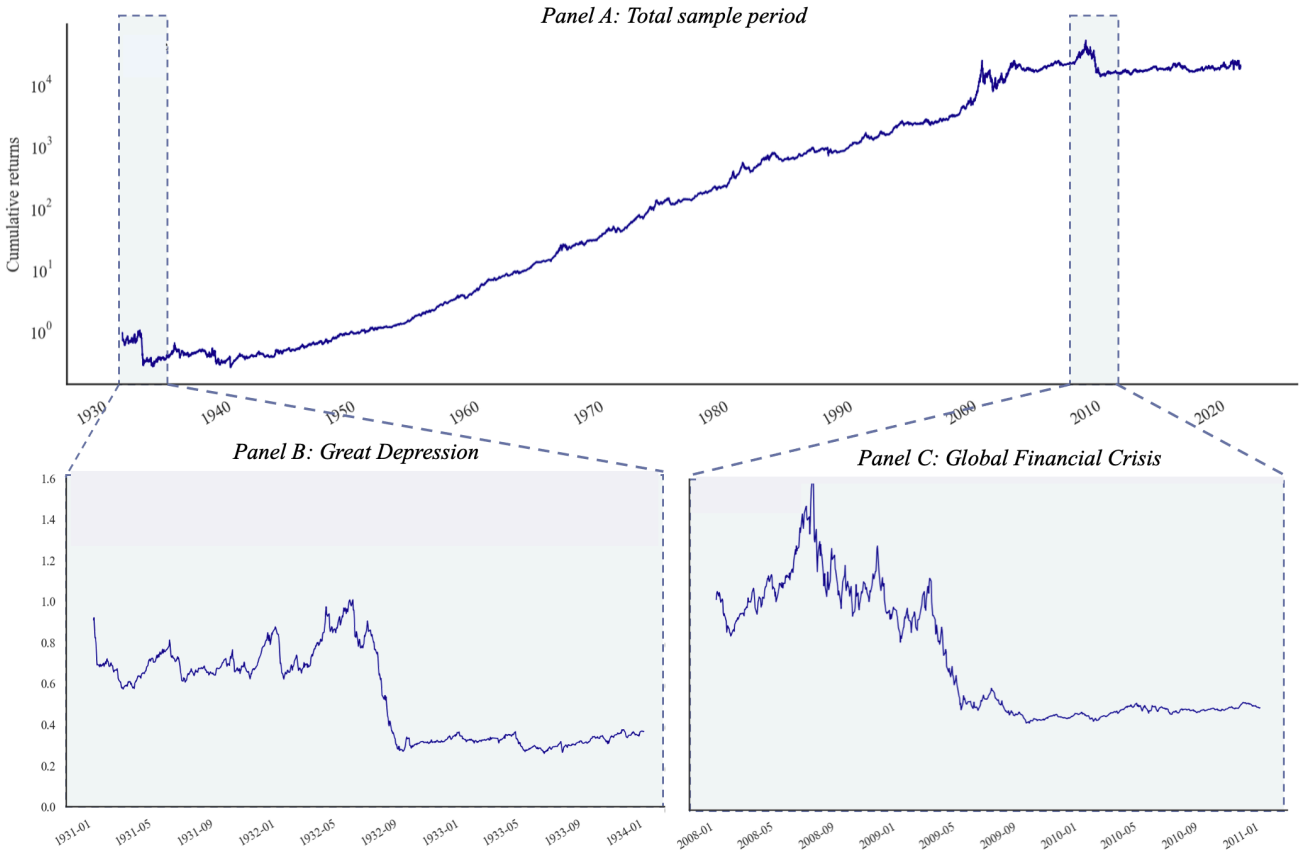


Figure 9.1 notes: The figure depicts the cumulative total return (including the risk-free rate) of the WML portfolio. Panel A shows the cumulative return for the total sample period, January 1931 to December 2020 on a log scale. Panel B and C provide the reindexed portfolio value for two sub-periods, January 1931 to January 1934, and January 2008 to January 2011. (Source: Own creation).

Panel A in Figure 9.1 presents the cumulative returns of the MOM strategy in the total sample period from January 1931 until December 2020. Three observations are made from this panel. First, the cumulative returns of the MOM strategy have shown a strong positive trend with the strategy obtaining a cumulative return of 19,133 for the total period. Specifically, the strategy has been highly profitable during the period from approximately 1940 until 2000. Second, the profitability of the MOM strategy appears less persistent for the more recent time-period spanning from 2000 to 2020, which has also been documented in recent literature

(Novy-Marx, 2012). Third, we confirm the findings of the existing literature on momentum crashes. Two major drawdowns in returns during 1932 and 2009 are evident in our sample (see Figure 9.1, panel B and C, respectively). The latter observation is emphasized, as this constitutes the foundation for answering the problem statement of this thesis. Consequently, we zero in on the momentum crashes below.

Table 9.2: The worst performing five months of the WML portfolio during the total sample period

Rank	Month	WML return	Bear market indicator	Market return
1	Aug 1932	-45.16	-67.77	36.49
2	Jul 1932	-39.76	-74.91	33.63
3	Jan 2001	-32.04	10.68	3.92
4	Apr 2009	-30.57	-40.55	5.20
5	Sep 1939	-29.66	-21.46	16.97

Skewness	-1.65

Table 9.2 notes: The table reports the worst five months in terms of one-month (non-excess) returns observed for the WML portfolio. Further, the table provides a bear market indicator, dictating that a bear market is present when the two-year cumulative market returns, leading up to the portfolio formation date, is negative. The table also presents the contemporaneous market return. The returns are reported in percent and are ranked according to the size of the portfolio loss for a given month. The sample skewness is computed as the realized Fisher-Pearson skewness of the daily log returns. The total sample period spans January 1931 to December 2020. (Source: Own creation).

Table 9.2 presents the five worst one-month returns for the WML portfolio of our sample which coincide with the worst-performing months documented by Daniel & Moskowitz (2016). The worst months follow bear markets that suddenly rebound, with the exception of January 2001 during the dot-com bubble. The two worst performing months occurred during the Great Depression, with one-month returns of -45.16% (August 1932) and -39.76% (July 1932), respectively. The WML further experienced a large negative one-month return of -30.57% during the global financial crisis (April 2009). Thereby, we confirm the findings of Daniel and Moskowitz (2016), i.e., that the MOM strategy crashes.

Table 9.2 further outlines the skewness of the daily returns[30]. As a result of the large negative returns earned by the MOM strategy during the crash periods, the strategy exhibits a negative daily skewness of -1.65 (Table 9.2). Moreover, Figure 9.2 shows the drawdowns of the WML portfolio during the total sample period in both tabular and graphical format:

[30] In addition, the return distribution of the MOM strategy can be found in appendix A.10.

Figure 9.2: Drawdowns of the WML portfolio for the total sample period

Drawdown	Peak date	Valley date	Recovery date	Duration (days)
75.14	31 May 1932	11 Sep 1939	14 Apr 1950	4664
74.34	14 Jul 2008	22 Sep 2009		
68.83	09 Mar 2000	19 Jan 2001	25 Sep 2007	1969

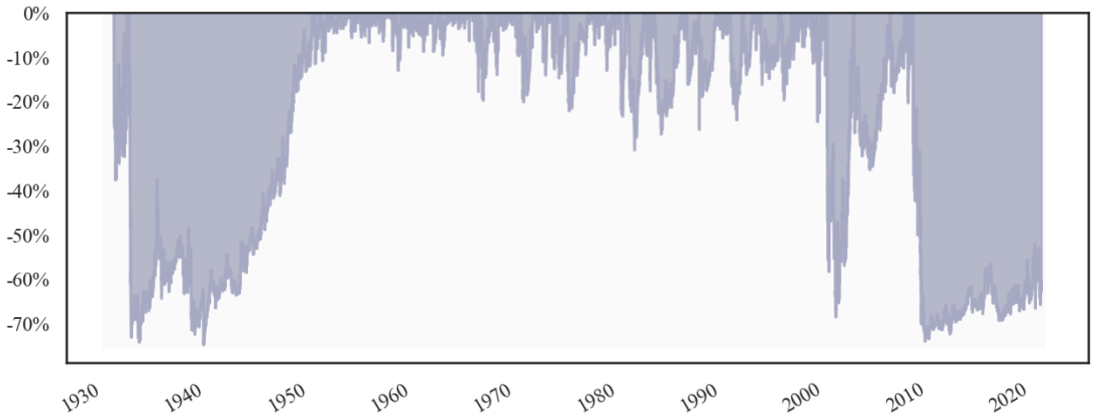


Figure 9.2 notes: The figure reports the worst three drawdowns in tabular format and all drawdowns graphically for the WML portfolio. The drawdowns are computed based on daily returns. The figure reports the drawdown of the period in percent, the peak date before the loss occurs, the valley date (the date where the most negative value occurs), the date of recovery, and the total duration of the drawdown period (in days). If the drawdown has not yet recovered, the recovery date and duration are left blank. The figure further plots the drawdowns of the WML portfolio in percent. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

We observe that the WML portfolio has experienced the worst drawdown periods following the Great Depression, the 2008-2009 financial crisis, and the dot-com bubble. The MOM strategy crashed substantially following May 1932, leading to a drawdown of 75.14%. The MOM strategy did not recover until April 1950, i.e., after 19 years. In addition, the MOM strategy has still not recovered since its drawdown of 74.34% in September 2009. Thus, we document that the crash risk inherent to the MOM strategy results in substantial losses that take many years to recover from.

In summary, this subsection confirms that two findings documented in the literature are present in our sample. First, we observe a strong momentum premium over the last century in our sample as a result of returns to the winner portfolio that significantly differs from the returns to the loser portfolio. This indicates that the momentum effect is indeed present in our data sample. Second, we confirm the crashes of the MOM strategy following market declines when market volatility is high, and the market rebounds contemporaneously. Having established that our sample is a valid representation of the findings in the literature, we proceed with the analysis of our machine learning models.

9.2 Machine Learning Models as Predictors of Excess Stock Returns

Before we initiate the analysis on the MLS *strategies*, we commence by evaluating the predictive power of the machine learning *models*. We conduct this analysis through three evaluation metrics. The first metric is the predictive R^2 which assesses the ability of the models to predict monthly excess stock returns. The second metric is the Diebold Mariano-test which compares the monthly prediction performance of the models. The last metric is the Spearman rank correlation coefficient which expresses the ability of the models to predict the monthly excess stock returns in the *cross-section*.

Table 9.3 presents the monthly predictive performance (R^2) of the baseline model and the three machine learning models for the out-of-sample period. Note that there is considerable noise in monthly returns which enables only a slightly positive R^2 (we elaborate on this in section 10.1).

Table 9.3: Predictive R^2 of the machine learning models for the out-of-sample period

	OLS (baseline)	RAF	MLP	RNN
R^2	-6.67	0.66	0.75	0.59

Table 9.3 notes: For each of the four models, the table reports the predictive out-of-sample R^2 for the predicted excess returns versus the target excess returns. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

As the only model, the baseline model generates a *negative* R^2 of -6.67%, indicating that the predictions of the model are more inaccurate than the average value of the excess returns for the out-of-sample period. The inferior performance of the baseline model relative to the other machine learning models is not surprising, as this model has no hyperparameters to optimize, and thus lacks regularization. Consequently, the baseline model is highly susceptible to in-sample overfitting which is also evident in our case (for the training set R^2 is 9.9%, see appendix A.11).

In contrast, the R^2 of the three more complex machine learning models are all positive with values of 0.66%, 0.75%, and 0.59% for RAF, MLP, and RNN, respectively. The positive R^2s point to the value of incorporating complexity and nonlinear relationships into the predictions, which are embedded in tree and neural network models but are missed by a simple OLS regression. In this regard, we also note that the three machine learning models do not overfit the in-sample data to the same extent as the baseline model (appendix A.11). The limited overfitting of these models is not surprising, as we have deployed extensive regularization techniques to counter overfitting.

While Table 9.3 offers an overview of the individual predictive performance of the four models, we further conduct a Diebold-Mariano test for a *pairwise comparison* of the predictive performance of the models in Table 9.4:

Table 9.4: Diebold-Mariano test of the machine learning models for the out-of-sample period

	RAF	MLP	RNN
OLS (baseline)	**(4.59)**	**(4.66)**	**(4.63)**
RAF		(0.72)	(-0.23)
MLP			(-0.56)

Table 9.4 notes: The table reports the Diebold-Mariano test statistics that compare the out-of-sample stock-level prediction performance of the three machine learning models and the baseline model. A positive number indicates that the column model outperforms the row model. The test statistics are marked in bold if the difference in the prediction accuracy between two models is statistically significant at a 5% level. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

Table 9.4 outlines the statistical significance of the models' predictive performance when compared pairwise. Bold values indicate significance at the 5% level, and a positive test statistic implies that the column model outperforms the row model. The Diebold-Mariano test statistics support our findings based on the predictive R^2. All three machine learning models have significant and positive test statistics when compared to the baseline model. Hence, the three machine learning models exhibit superior predictive performance relative to the baseline model. There is no statistical evidence to support a difference in the predictive performance between the three machine learning models. The RNN model is marginally inferior with test statistics of -0.23 and -0.56 compared to RAF and MLP model, respectively. This may seem surprising as the two latter models carry less information than the RNN model which embodies the temporal dimension and thus, incorporates a memory of variables in past months. While the difference is too small to draw any conclusions, this could serve as an indicator of complexity bias in our thesis (i.e., the tendency to prefer complicated models over simple models). We examine if this is the case when analyzing and comparing the performance of the strategies in the following sections.

At this point, the baseline model has already served its purpose. Having established its relative inferior performance to the more complex models as well as its negative R^2, we document highly unstable out-of-sample predictions of the model. The negative R^2 implies that the model is unfit for a stock-selection strategy. Thus, we follow the approach of Gu et al. (2019) and exclude the OLS regression from our further analysis[31], when we examine the ability of the models to rank stocks in the cross-section and deploy their predictions for stock-selection strategies.

Table 9.5 reports the Spearman rank correlation and the corresponding t-statistics of the three machine learning models:

[31] Note that Gu et al. 2019 obtain a positive R^2 for an OLS regression with the use of Huber loss, and thus this model is included in their further research. We refer to the exclusion of their simple OLS regression model which is similar to the one we deploy

Table 9.5: Spearman rank correlation coefficient of the machine learning models for the out-of-sample period

	RAF	MLP	RNN
$\bar{\rho}_S$	2.52	2.26	2.72
$t(\bar{\rho}_S)$	**(3.84)**	**(5.07)**	**(4.08)**

Table 9.5 notes: The table reports the out-of-sample Spearman rank correlation coefficients in percent between the monthly target excess returns and the monthly excess returns predicted by each of the three machine learning models, respectively. The t-statistics are reported in parenthesis and are for the null hypothesis that the Spearman rank correlation coefficient is equal to zero. The t-statistics are marked in bold if statistically significant at a 5% level. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

The machine learning models report a positive and significant Spearman rank correlation coefficient of 2.52, 2.26, and 2.72 for the RAF, MLP, and RNN model, respectively. Hence, the ranks of predicted excess returns are positively correlated with the ranks of target excess returns, implying that all models are able to predict excess stock returns in the cross-section. The ability of the models to understand the cross-section of excess returns constitutes the foundation for utilizing the predictions for stock-selection strategies that rely on stock rankings.

The RNN and RAF have the highest Spearman rank correlations but smaller t-statistics than the MLP, as a result of higher volatility in the Spearman rank correlation coefficients relative to the MLP. The volatility implies that the RAF and RNN are superior in ranking stocks in some months relative to the MLP, but less so in other months. Thus, we expect the RAF and RNN strategies to be more volatile than the MLP strategy. Furthermore, appendix A.12 presents the Spearman rank correlation over time for each of the three machine learning models, visualizing the less volatile Spearman rank correlation of the MLP strategy.

9.3 Performance of the Momentum and Machine Learning-Based Strategies

Up to this point, our assessment of the predictive performance of the machine learning models has been entirely statistical, relying on predictive R^2, Diebold-Mariano tests, and Spearman rank correlation. We proceed by examining how predictability translates into economic gains in the form of a stock-selection strategy. Thus, this section aspires to answer research question three. Specifically, this section investigates the performance of the three constructed MLS strategies and relates it to the performance of the MOM strategy. First, we examine the performance of the strategies during the total out-of-sample period[32]. Second, we focus on sub-periods in which the MOM strategy crashes, to examine if the MLS strategies exhibit superior performance during these periods.

[32] Note that the sample period which we examine differs from the one presented in section 9.1, as we focus on the *out-of-sample* period spanning January 1976 to December 2020.

9.3.1 Total Out-of-Sample Period

We commence by presenting the cumulative returns of the MOM strategy and the three MLS strategies in Figure 9.3 for the total out-of-sample period:

Figure 9.3: Cumulative returns of the four stock-selection strategies for the out-of-sample period

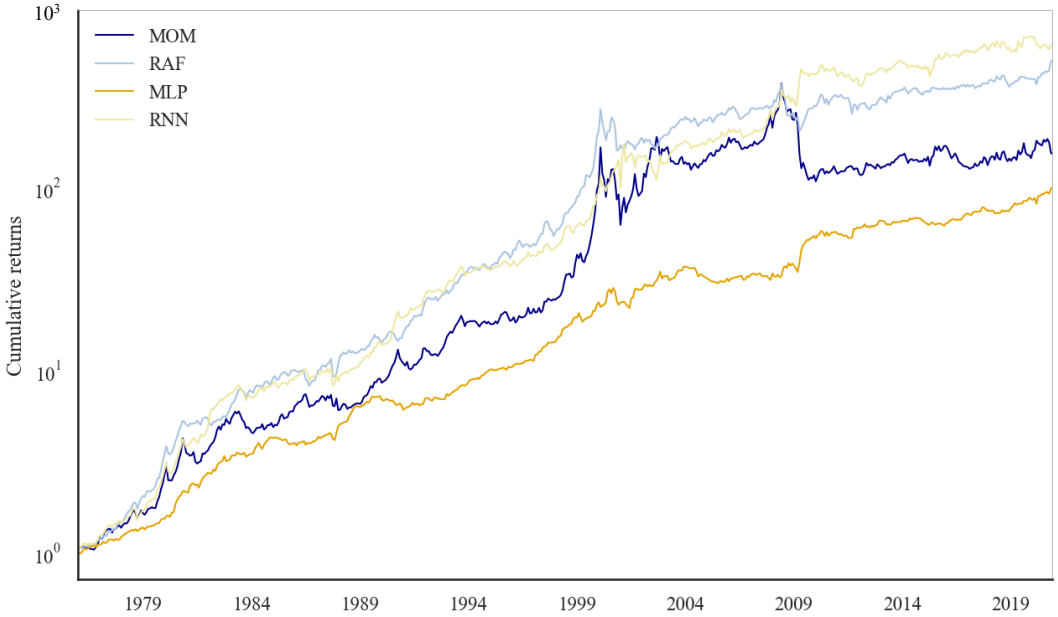


Figure 9.3 notes: The figure depicts the total cumulative return (including the risk-free rate) of the conventional MOM strategy and the three MLS strategies, on a log scale. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

Figure 9.3 depicts a positive trend in the cumulative returns of all four strategies over the out-of-sample period. Of particular interest to this thesis, all three MLS strategies seem to exhibit less crash risk than the MOM strategy which encounters two major crashes in the out-of-sample period, namely following the dot-com bubble and during the global financial crises. Similar to the MOM strategy, the cumulative returns of the RAF strategy decrease in these two periods, though to a smaller extent. In contrast, the RNN and MLP strategies exhibit more stable returns in the two periods. We dissect the performance of the three strategies when zeroing in on the periods where the MOM strategy crashes in the following subsection (9.3.2).

Table 9.6 presents the correlation between the MOM strategy and the three MLS strategies for the whole out-of-sample period:

Table 9.6: Pearson correlation coefficients between the MLS strategies and the MOM strategy

	RAF	MLP	RNN
$\rho_{MOM,daily}$	**0.38**	**0.05**	**0.39**
$\rho_{MOM,monthly}$	**0.42**	-0.03	**0.46**

Table 9.6 notes: The table reports the Pearson correlation between each of the MLS strategies and the replicated MOM strategy based on daily and monthly returns, respectively. Bold numbers mark statistical significance of the correlation at the 5%-level. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

We observe from Table 9.6 that the *daily* returns of all three MLS strategies are significantly and positively correlated with the MOM strategy with coefficients of 0.38, 0.05, 0.39 for the RAF, MLP, and RNN strategy, respectively. However, the interdependence between the daily returns of the MLP and MOM strategy is less prominent. In fact, the *monthly* returns of the MLP strategy do not appear to be correlated with the monthly returns of the MOM strategy, as revealed by the insignificant correlation coefficient (-0.03). In contrast, the interdependencies between the *monthly returns* of the MOM strategy and the two other MLS strategies (RAF and RNN) are similar to the interdependencies when using *daily returns*.

Performance Metrics of the Machine Learning-Based Strategies Versus the Momentum Strategy

Table 9.7 presents the descriptive statistics of the MOM strategy along with the three MLS strategies for the out-of-sample period. We observe statistically significant alphas for both the MOM strategy and the MLS strategies when adjusting for market risk. The MOM strategy yields a higher alpha than the MLS strategies, as a result of its negative market loading. However, the alpha of the MOM strategy is less significant than the alpha of the MLS strategies due to higher volatility in the returns of the strategy. From Table 9.7 we further observe that the volatilities of the MLS strategies' excess returns all lie in the range from 11% to 17%. Thus, they are associated with substantially lower volatility relative to the volatility of the MOM strategy (22.59%).

Table 9.7: Performance of the MOM strategy versus the three MLS strategies for the out-of-sample period

	MOM	RAF	MLP	RNN
$\overline{r - r_f}$	7.14	9.96	6.11	10.49
$t(\overline{r - r_f})$	(2.81)	(4.40)	(3.98)	(4.65)
σ	22.59	16.60	11.02	16.34
α	10.09	6.56	5.40	9.91
$t(\alpha)$	(2.86)	(3.05)	(3.29)	(3.99)
β	-0.02	0.53	0.15	0.22
Sharpe ratio	0.42	0.66	0.59	0.69
Sortino ratio	0.57	0.94	0.90	0.97
Maximum drawdown	74.34	49.96	24.80	45.42
Skewness	-1.37	-0.33	0.49	-1.30

Table 9.7 notes: This table provides descriptive statistics of the excess returns of the MOM strategy and MLS strategies. For each portfolio, we report the following metrics: The annualized return in excess of the risk-free rate, volatility, and alpha, in percent; the market beta; the annualized Sharpe ratio and Sortino ratio; and the maximum drawdown in percent. The sample skewness is computed as the realized Fisher-Pearson skewness of the daily log returns. The t-statistics are reported in parenthesis and are for the null hypothesis of the metrics being equal to zero. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

An important observation for answering research question three is the superior Sharpe ratios of the three MLS strategies relative to the MOM strategy. More specifically, the RAF, MLP, and RNN strategies generate Sharpe ratios of 0.66, 0.59, and 0.69, respectively. To support the notion that the MLS strategies are superior to the MOM strategy, we point to the relative performance of the four strategies in Table 9.8. The table presents the alphas of the MLS strategies over the MOM strategy (rather than the market). All three MLS strategies demonstrate significant and positive alphas of 8.58%, 6.49%, and 9.04% for the RAF, MLP, and RNN strategies, respectively. The positive alpha over the MOM strategy of the MLP strategy might appear puzzling since the MOM strategy yields a higher excess return than the MLP strategy. However, the explanation is nested in the low correlation between the two strategies, causing a low loading of the MLP strategy on the MOM strategy (i.e., a low beta coefficient). Moreover, the alpha of the MLP strategy is highly significant due to the low volatility that the returns of the strategy exhibit.

Table 9.8: Alpha over the MOM strategy for the out-of-sample period

	RAF	MLP	RNN
α_{MOM}	8.58	6.49	9.04
$t(\alpha_{MOM})$	**(3.60)**	**(3.84)**	**(3.86)**
β_{MOM}	0.28	0.03	0.28

Table 9.8 notes: The table reports the alpha and beta for the three MLS strategies over the MOM strategy. The t-statistics are reported in parenthesis and are for the null hypothesis that alpha is equal to zero. The t-statistics are marked in bold if statistically significant at a 5% level. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

Downside risk of the MLS strategies versus the MOM strategy

Furthermore, the downside risk of the strategies is relevant in light of the research of this thesis. Figure 9.4 depicts the drawdowns of the four strategies during the out-of-sample period. We observe that the MOM strategy crashes more severely and over longer time periods relative to the MLS strategies. In fact, where the MOM strategy experienced a maximum drawdown of as much as 74.34% during the out-of-sample period, the maximum drawdowns of the MLS strategies are all less than 50% (see also Table 9.7). In addition, all the MLS strategies report less negative skewness compared to the MOM strategy. As evident by the drawdowns and the skewness of the strategies, the downside risk of the MLS strategies are less prevalent relative to the MOM strategy. This is also evident by the Sortino ratios of the four strategies. Pointing back to Table 9.7, the Sortino ratios of the three MLS strategies all lie in the range of 0.9-0.97 which is substantially higher than the ratio of 0.57 generated by the MOM strategy.

Figure 9.4: Drawdowns of the four strategies for the out-of-sample period

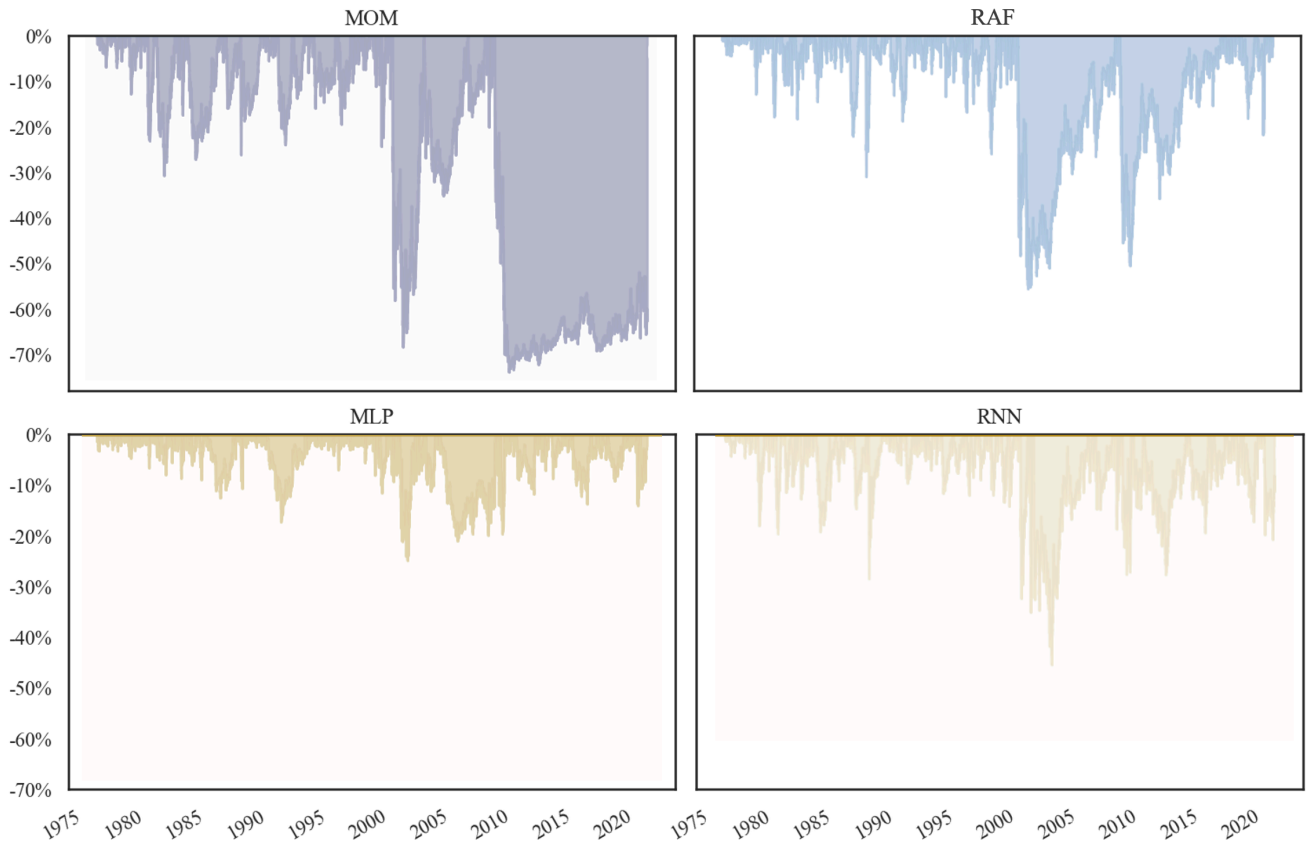


Figure 9.4 notes: The figure illustrates the drawdowns of the MOM strategy and the three MLS strategies. The drawdowns are calculated based on daily returns. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

Relative Performance of the Machine Learning-Based Strategies

Having established that the MLS strategies are superior to the MOM strategy for the out-of-sample period, we briefly compare the performance of the three MLS strategies. We observe notable differences among the three strategies in Table 9.7. In line with the findings from the Spearman correlation, we find that the MLP strategy is associated with less volatility in returns than the RAF and RNN strategies. The MLP strategy generates a lower annual excess return than the two other MLS strategies but also exhibits less downside risk. More specifically, the maximum drawdown of the MLP strategy is only (approximately) half the size of the maximum drawdown for the RAF and RNN strategy. Further, the returns of the MLP strategy are positively skewed, in contrast to the negative skewness associated with the returns of the RAF and RNN strategy. Thus, we expect the MLP to perform particularly well, relative to the two other MLS strategies, during the periods associated with momentum crashes.

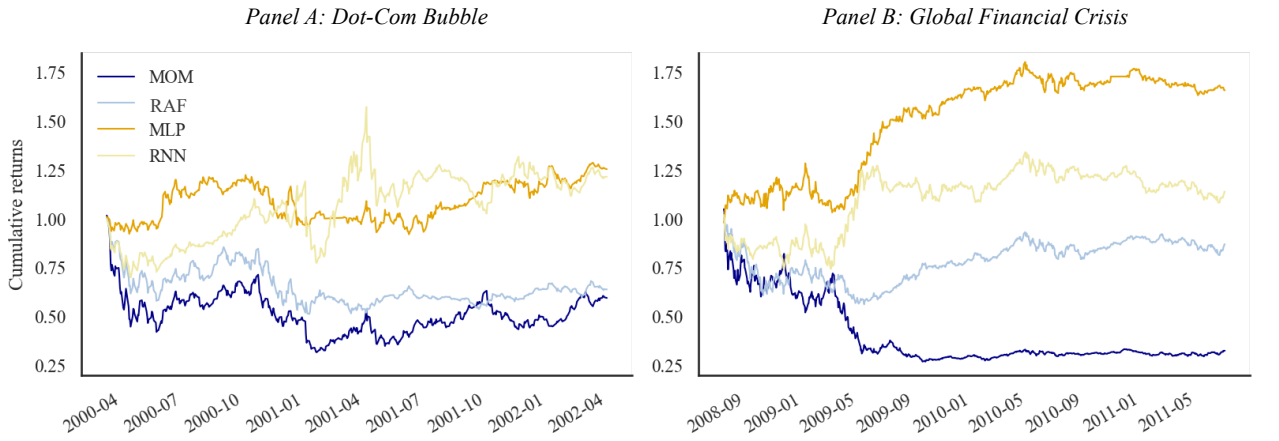
In summary, we document a superior performance of the MLS strategies relative to the MOM strategy for our investment universe during the out-of-sample. Most importantly, we demonstrate a higher Sharpe Ratio and a significantly positive alpha over the MOM strategy. However, in aspiration to answer our problem statement, we further test the robustness of our results across two sub-periods. Specifically, we zero in on the dot-com bubble and the global financial crisis in which the MOM strategy exhibits considerable crash risk.

9.3.2 Sub-periods

Recall that crash periods are only loosely defined in the literature (section 3.4.2). Thus, to divide the total out-of-sample period into sub-periods characterized by the largest crashes of the MOM strategy, we define specific criteria for the beginning and end date of a crash period: We initiate each of our two sub-periods on the *peak date* before the crash of the MOM strategy. We end each of our two sub-periods on the first date following the peak date, where the two-year cumulative MOM return becomes positive. Based on these criteria, the following sub-periods are defined: the sub-period related to the dot-com bubble spans 9 March 2000 to 1 April 2002, and the sub-period related to the global financial crisis spans 14 July 2008 until 1 July 2011. For simplicity, we refer to these sub-periods as the dot-com bubble and the global financial crisis. Also, note that we only present and comment on the most important performance metrics for the four strategies during each of the two sub-periods.

We initiate the analysis by depicting the cumulative returns of the four strategies during each of the two sub-periods:

Figure 9.5: Cumulative returns and correlations of the four strategies for the two sub-periods



	RAF	MLP	RNN
$\rho_{MOM,daily}$	0.42	0.43	0.52

	RAF	MLP	RNN
$\rho_{MOM,daily}$	0.11	-0.14	-0.26

Figure 9.5 notes: The figures depict the total cumulative returns (including the risk-free rate) of the conventional MOM strategy and the three MLS strategies. Panel A and B show the sub-periods 9 March 2000 until 1 April 2002, and 14 July 2008 until 1 July 2011, respectively. The figure further reports the Pearson correlation between each of the three strategies and the MOM strategy, calculated on a daily basis. (Source: Own creation).

In general, two conspicuous observations can be made from Figure 9.5. First, the cumulative returns of the MLP and RNN strategies substantially surpass the MOM strategy during both sub-periods. Second, the superior performance of these strategies relative to the MOM strategy is most prominent during the global financial crisis (Figure 9.5, panel B), where the RAF strategy also outperforms the MOM strategy.

Zooming in on the dot-com bubble (Figure 9.5, panel A), all MLS strategies are positively correlated with the MOM strategy. However, the MLS strategies are associated with less negative returns from March to June 2000. Notably, the MLP strategy exhibits a relatively stable performance compared to the other strategies. While the RAF and RNN strategies do experience decreases in their cumulative returns, their performance remain superior to the MOM strategy.

Examining the global financial crisis (Figure 9.5, panel B), all three MLS strategies outperform the MOM strategy. The respective correlations between the MLS strategies and the MOM strategy are lower than for the dot-com bubble. In fact, the MLP and RNN strategy are negatively correlated with the MOM strategy. The negative correlation is especially prominent in the period from April until July 2009 where the returns of the MLP and RNN strategy dramatically increase, as opposed to the MOM strategy that crashes. Notably, the MLP strategy demonstrates the strongest performance out of all three MLS strategies with a considerable upswing in the cumulative returns when the MOM strategy crashes.

Table 9.9: Performance of the four strategies for the two sub-periods

	MOM	RAF	MLP	RNN
Dot-com bubble				
$\overline{r - r_f}$	-25.73	-23.10	6.96	5.32
$t(\overline{r - r_f})$	-(0.38)	(-0.67)	(0.56)	(0.49)
σ	54.62	39.48	25.44	40.18
α_{MOM}	n.a.	-13.07	13.70	20.90
$t(\alpha_{MOM})$	n.a.	(-0.56)	(0.80)	(0.79)
Sharpe ratio	-0.27	-0.47	0.39	0.33
Sortino ratio	-0.35	-0.64	0.59	0.44
Maximum drawdown	68.83	48.95	24.80	35.08
Global financial crisis				
$\overline{r - r_f}$	-31.59	-4.81	18.22	4.17
$t(\overline{r - r_f})$	(-1.46)	(-0.09)	(1.73)	(0.50)
σ	38.81	26.80	18.36	25.12
α_{MOM}	n.a.	1.16	17.72	1.77
$t(\alpha_{MOM})$	n.a.	(0.08)	(1.54)	(0.12)
Sharpe ratio	-0.84	-0.05	1.00	0.29
Sortino ratio	-1.09	-0.07	1.56	0.42
Maximum drawdown	74.34	44.76	19.57	27.58

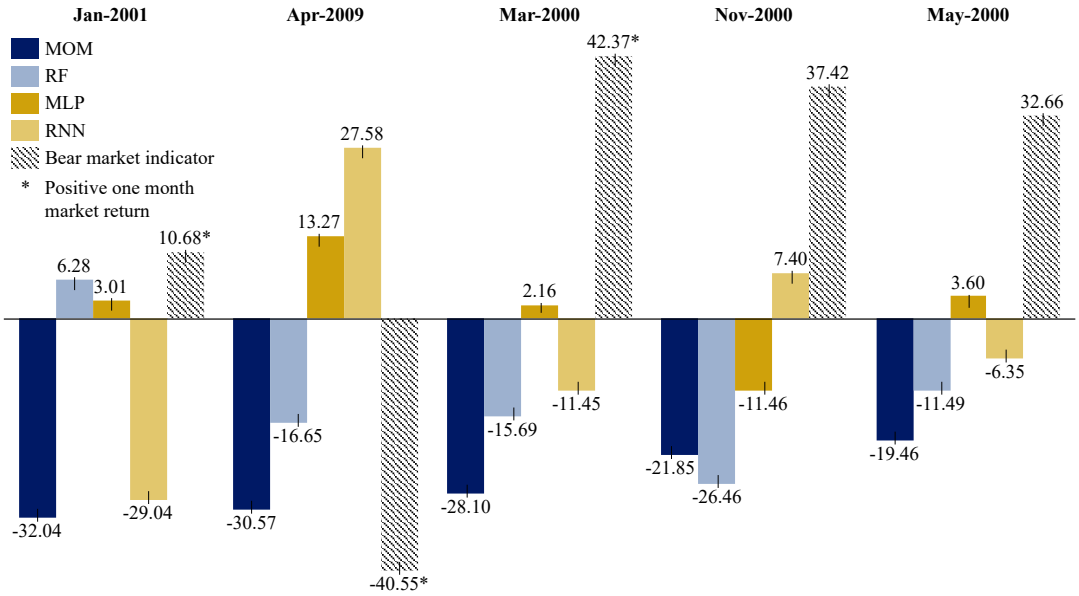
Table 9.9 notes: The table reports the worst five months in terms of one-month (non-excess) returns observed for the WML portfolio during the total sample period spanning January 1931 to December 2020. Further, the table provides a bear market indicator, dictating that a bear market is present when the two-year cumulative market returns, leading up to the portfolio formation date, is negative. The table also presents the contemporaneous market return. The returns are reported in percent and are ranked according to the size of the portfolio loss for a given month. The sample skewness is computed as the realized Fisher-Pearson skewness of the daily log returns. (Source: Own creation).

Table 9.9 presents the descriptive statistics of the MOM strategy along with the three MLS strategies for the two sub-periods. In general, we document a superior performance of all three MLS strategies relative to the MOM strategy, with a positive Sharpe and Sortino ratio for the MLP and RNN strategies during both crash periods. Supporting the findings from Figure 9.5, the RAF strategy exhibits inferior performance relative to the two other MLS strategies. Note that the RAF strategy outperforms the MOM strategy despite its more negative Sharpe and Sortino ratio during the dot-com bubble, in terms of less negative excess returns and lower realized volatility.

Due to the high volatility of returns and the limited number of observations in our two sub-periods, we do not find statistical evidence for the difference in the relative performance when examining the alpha of the MLS strategies over the MOM strategy.

To obtain a more in-depth understanding of the performance of the strategies relative to the MOM strategy during these two sub-periods, we illustrate the five worst one-month returns of the MOM strategy and the respective one-month returns of the three MLS strategies in Figure 9.6. The five months are ranked according to the size of the negative returns of the MOM strategy. Recall that the MOM strategy crashes following bear markets that suddenly rebound (Daniel & Moskowitz, 2016). Thus, we further depict a bear market indicator (based on the two years cumulative market returns leading up to the formation date) as well as an asterisk denoting market rebounds (based on the sign convention of the contemporaneous one-month market return). This further enables us to explore how the market state impacts the performance of the MLS strategies.

Figure 9.6: The five worst one-month returns of the MOM strategy, and the respective performance of the MLS strategies for the out-of-sample period[33]



[33] We note that four out of the five worst one-month returns for the MOM strategy occur during the dot-com bubble, whereas only one of the worst returns occurs during the global financial crisis. This might seem puzzling, as the crash of the momentum strategy during the global financial crisis was much more pronounced than the crash during the dot-com bubble. The explanation is nested in the length of the crash periods, as the dot-com bubble crash was more abrupt relative to the crash during the financial crisis (recall Figure 9.4)

Figure 9.6 notes: The figure illustrates the worst five one-month returns observed for the MOM strategy. The one-month returns are ranked according to the size of the negative return and are in percent. The figure further depicts the corresponding returns of the MLS strategies at the same months. In addition, the figure shows the cumulative two-year market returns which function as a bear market indicator, following Daniel and Moskowitz (2016). Lastly, the bear market indicator is marked with an asterisk if the contemporaneous market return is positive, indicating a market rebound. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

A glance at Figure 9.6 reveals that the three MLS strategies generally outperform the MOM strategy for the five months in terms of one-month returns. Not including January 2001, the RAF strategy yields returns with a similar sign convention as the MOM strategy but with smaller fluctuations in the returns. The MLP and RNN strategies exhibit superior one-month returns relative to the MOM strategy for all months. While no clear-cut pattern can be derived from the five months, the MLP strategy appears to exhibit low volatility in the one-month returns relative to both the MOM strategy and the other MLS strategies, supporting our findings from Figure 9.5. In contrast, the RNN strategy exhibits a more volatile performance with a highly negative return in January 2001 and a highly positive return in April 2009. The more volatile returns in these five months of the RNN strategy are also in line with our initial findings in Figure 9.5.

Resonating Daniel and Moskowitz (2016), the worst momentum crashes occur following bear markets that suddenly rebound. These observations do not hold true for the worst one-month returns for our out-of-sample period: Only the worst one-month return during the global financial crisis follows a bear market that suddenly rebounds. The MLP and RNN strategy perform particularly well in this single month. Following the logic of Daniel & Moskowitz (2016) a possible explanation for the superior performance of the MLP and RNN strategy is the ability of the strategies to time market rebounds, following bear markets. In a modest attempt to test this notion, appendix A.13 expands the worst one-month return analysis to the worst 15 months for the MOM strategy, documenting that the RNN strategy generates positive returns in *all* months following a bear market that suddenly rebounds. Thus, one possible explanation of the superior performance of the RNN strategy is its ability to time market that follow bear markets. In contrast, such a pattern is not as clear-cut for the MLP strategy.

To ensure that the MLS strategies do not crash during different periods than the MOM strategy, we further present the rolling six-month volatility of all four strategies in Figure 9.7. In addition, we provide a *full overview* of the one-month returns for each of the four strategies for the total out-of-sample period in appendix A.14. Figure 9.7 as well as appendix A.14 support our findings, as we observe peaking volatility in returns for all three MLS strategies concurrent with the MOM strategy, namely during the two crash periods. Nevertheless, comparing the performance of the MLS strategies to the MOM strategy, the three strategies are all associated with less volatility during these crash periods. Especially the MLP and RNN strategy capture more upside volatility during the two crash periods, as evident by their superior Sortino ratio in Table 9.9. The MLP and RNN strategy obtain Sharpe ratios of 0.59 and 0.44, respectively, relative to the -0.35 of the MOM strategy during the dot-com bubble. Similarly, during the global financial crisis, the MLP and RNN strategy obtain Sortino ratios of 1.56 and 0.42, respectively, relative to -1.09 of the MOM strategy. Hence, we document that the MLP and RNN strategy, in particular, are associated with less downside risk.

Figure 9.7: Rolling six-month volatility of the four strategies for the out-of-sample period

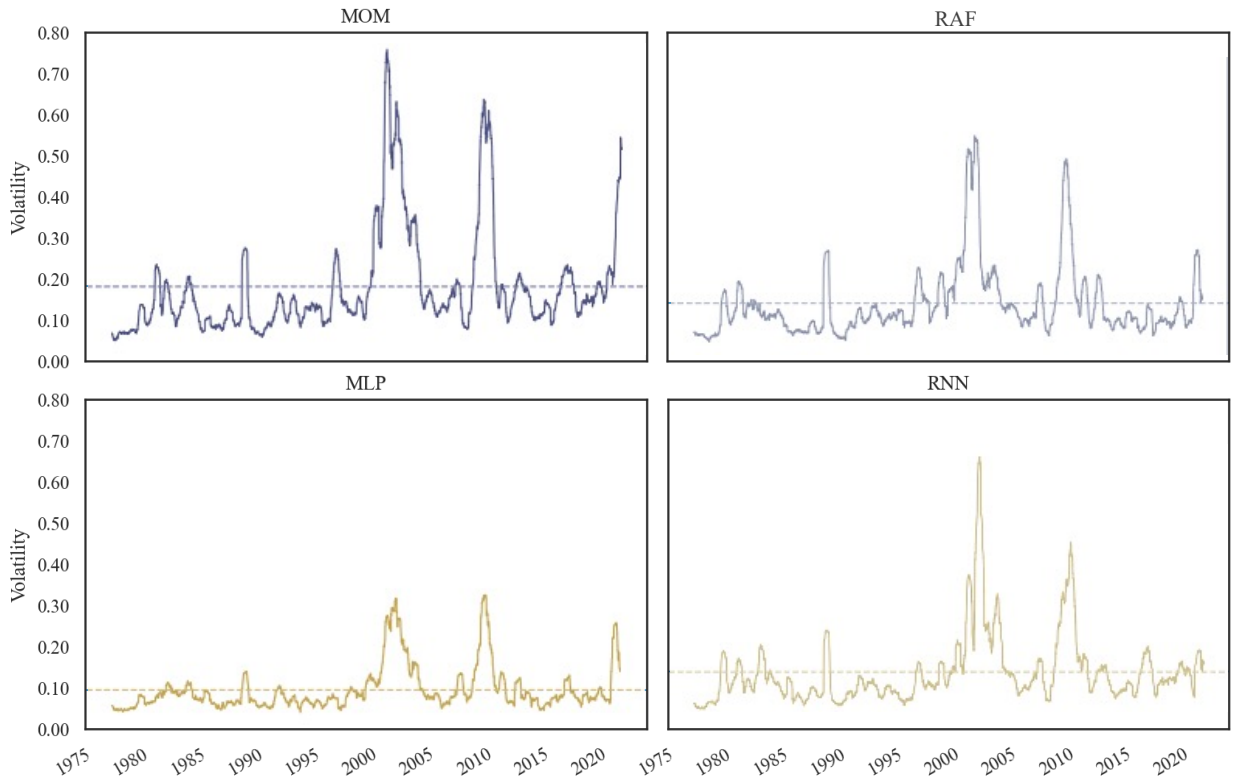


Figure 9.7 notes: The figure depicts the six-month rolling volatility of returns for each of the four strategies. The dashed lines indicate the mean of the volatility, calculated based on the six-month rolling volatility. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

In summary, the overall findings of this subsection support the observations from the total out-of-sample period with all MLS strategies outperforming the MOM strategy. We document a superior performance of the MLP and RNN strategy relative to the MOM strategy in terms of both Sharpe and Sortino ratios for both sub-periods. The performance of the MLP strategy is more stable relative to the other strategies during both sub-periods. Notably, the RNN strategy appears able to time market upswings that follow bear markets. The superiority of the RAF strategy is less conspicuous when compared to the MLP and RNN strategy, evident by its negative Sharpe and Sortino ratios in both crash periods.

9.4 Ensemble Strategy

Having established the superior performance of the MLS strategies relative to the MOM strategy based on both the total out-of-sample period and during crash periods, this section discusses the possible implications of combining the three MLS strategies with the MOM strategy. Thus, we provide an answer to the fourth research question in this section.

Recall that an ensemble model relies on the "wisdom of the crowd" in which multiple models protect each other from their individual weaknesses (Suthaharan, 2016). As long as the individual models are diverse and

independent, the prediction errors will decrease as a result of applying the ensemble approach. Hence, we coalescence all four strategies into one ensemble strategy despite the divergence in their individual performance. Specifically, we aspire to construct an *enhanced* MOM strategy that exhibits less crash risk when combined with the MLS strategies. We construct a simple ensemble strategy that equally weighs the returns of each of the four strategies:

$$R_{ensemble} = ¼\, R_{MOM} + ¼\, R_{RAF} + ¼\, R_{MLP} + ¼\, R_{RNN} \tag{82}$$

Having documented that the MLS strategies are not perfectly correlated with the *MOM strategy*, we expect the strategies to perform particularly well in an ensemble strategy, insofar they are not perfectly correlated with *each other*. Table 9.10 provides an overview of the Pearson correlation between the four strategies.

Table 9.10: Pearson correlation coefficients between the four strategies for the out-of-sample period

$\bar{\rho}_{daily}$	RAF	MLP	RNN
MOM	0.38	0.05	0.39
RAF		0.35	0.37
MLP			0.12

Table 9.10 notes: The table reports the pairwise Pearson correlation between each of the four stock-selection strategies, calculated based on daily returns. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

We document a low correlation between all MLS strategies, with the RNN and MLP strategy exhibiting a particularly low correlation of 0.12. This implies that the strategies can advantageously be combined into an ensemble model in order to reduce their individual prediction errors. Figure 9.8 illustrates the cumulative return of the ensemble model for the out-of-sample period.

Figure 9.8: Cumulative returns and drawdowns of the ensemble strategy for the out-of-sample period

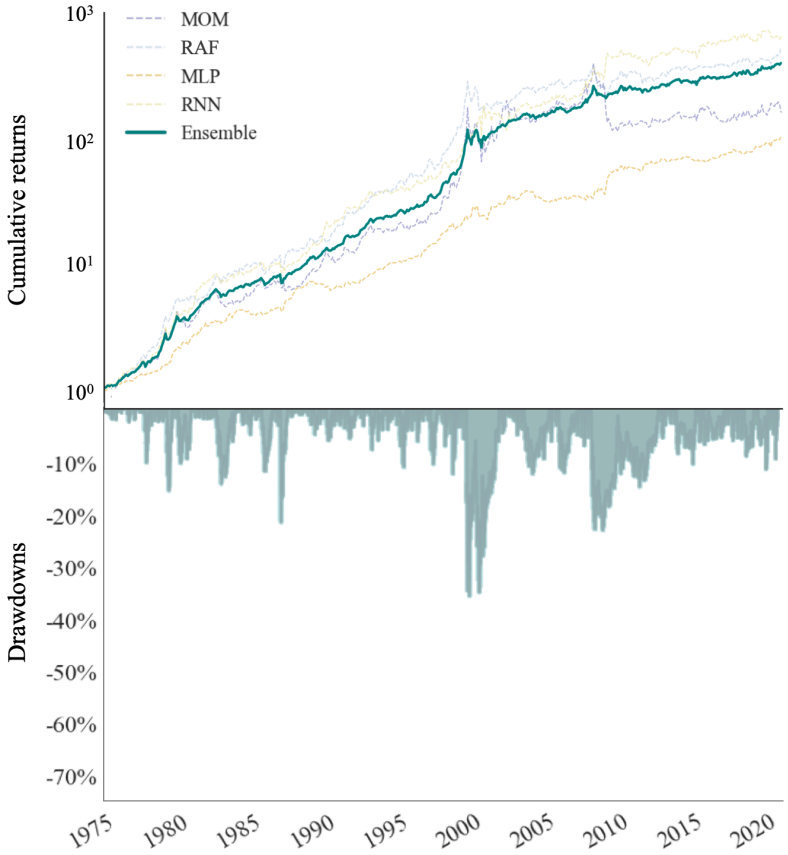


Figure 9.8 notes: The figure depicts the total cumulative returns (including the risk-free rate) of the ensemble model that equally weighs the MOM strategy and the three MLS strategies. The cumulative returns of the four individual strategies are shaded. The cumulative returns are shown on a log scale. Furthermore, the figure illustrates the corresponding drawdowns of the ensemble strategy in percent. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

As depicted in Figure 9.8, the ensemble strategy demonstrates a more stable performance relative to the strategies individually (with the exception of the MLP strategy) over the out-of-sample period. The ensemble strategy exhibits little downside risk compared to the MOM strategy during the global financial crisis, where the most substantial momentum crash occurred. This is a result of the low (or even negative) correlation between the three MLS strategies and the MOM strategy during this sub-period, as documented in subsection 9.3.2. In contrast, the ensemble strategy exhibits the worst drawdown during the dot-com bubble, due to a higher correlation between the MLS strategies and the MOM strategy in this period. Pointing to appendix A.15 which depicts the rolling six-month volatility of the ensemble strategy, we confirm that the dot-com-bubble is also the most volatile period for the strategy. Moreover, the maximum drawdown of the strategy occurs during this period. However, as evident in Table 9.11, the maximum drawdown of the ensemble strategy is much smaller than the maximum drawdown for all the individual strategies, except the MLP strategy. Having established that the MLP strategy is substantially less volatile than the other three strategies, it is not surprising that the maximum drawdown of the ensemble strategy is larger than that of the MLP strategy.

Table 9.11: Performance of the ensemble strategy relative to the four individual strategies for the out-of-sample period

	MOM	RAF	MLP	RNN	Ensemble
$\overline{r - r_f}$	7.14	9.96	6.11	10.49	9.29
$t(\overline{r - r_f})$	(2.81)	(4.40)	(3.98)	(4.65)	(5.54)
σ	22.59	16.60	11.02	16.34	11.57
α	10.09	6.56	5.40	9.91	7.97
$t(\alpha)$	(2.86)	(3.05)	(3.29)	(3.99)	(4.70)
β	-0.02	0.53	0.15	0.22	0.22
Sharpe ratio	0.42	0.66	0.59	0.69	0.83
Sortino ratio	0.57	0.94	0.90	0.97	1.16
Maximum drawdown	74.34	49.96	24.80	45.42	36.48
Skewness	-1.37	-0.33	0.49	-1.30	-0.86

Table 9.11 notes: This table provides descriptive statistics of the excess returns of the ensemble models. We report the following metrics: The annualized return in excess of the risk-free rate, volatility, and alpha, in percent; the market beta; the annualized Sharpe ratio and Sortino ratio; and the maximum drawdown in percent. The sample skewness is computed as the realized Fisher-Pearson skewness of the daily log returns. The t-statistics are reported in parenthesis and are for the null hypothesis of the metrics being equal to zero. The out-of-sample period spans January 1976 to December 2020. (Source: Own creation).

Finally, the ensemble strategy outperforms all four individual strategies with an annualized Sortino and Sharpe ratio of 1.16 and 0.83, respectively, confirming the considerable potential of combining several strategies into one.

This thesis does not conduct an in-depth analysis of the ensemble strategy during the two sub-periods that we defined in subsection 9.3.2. However, we note that the superior performance of the ensemble relative to the MOM strategy remains robust when zooming in on the dot-com bubble and the global financial crisis (see appendix A.16). This is an important observation in the attempt to construct an enhanced momentum strategy that exhibits less crash risk.

Recapitulating the findings of this section, the possible coalescence of the MLS strategies with the MOM strategy offers an opportunity to enhance the profitability of the four individual strategies. Particularly relevant for this thesis, the ensemble strategy offers a novel way to enhance the performance of the MOM strategy and counter its crash risk. The superior performance of the ensemble strategy relative to the four individual strategies emerges as the MLS strategies are not perfectly correlated with neither the MOM strategy nor each other.

Based on the above sections, we have documented several analyses that point to the superiority of the MLS strategies. Recall that the MOM strategy does not incorporate any information besides the 2-12 months past returns. Hence, the superior performance of the MLS strategies and the ensemble strategy is likely nested in

the additional information that the machine learning models rely on. Against this background, we zero in on research question five in which we seek to uncover explanations for the superior performance of the MLS strategies.

9.5 Importance of Input Variables

As machine learning models suffer from opacity, the underlying drivers of the MLS strategy's performance are highly challenging to extract and interpret. We attempt to peek inside the black box of the machine learning models by analyzing the importance of each input variable (SHAP value) when the three machine learning models predict excess returns. We aspire to uncover if one or multiple of the input variables are associated with high importance for *all* models, composing a common denominator for the superior performance of the MLS strategies. In this way, we might be able to link the superior performance of the MLS strategies to patterns in the information passed to the machine learning models.

We commence by providing an overview of the importance assigned to each input variable for the three machine learning models in Figure 9.9. To obtain the relative importance of the input variables, we normalize the variables into a [0:1] interval. Recall that the inputs of the machine learning models consist of our eleven momentum variables, of which three are market variables and eight are stock-specific variables. From Figure 9.9, we observe that all the machine learning models accentuate market variables when making predictions, relative to the stock-specific variables.

Figure 9.9: Relative importance of all input variables for the three machine learning models for the out-of-sample period

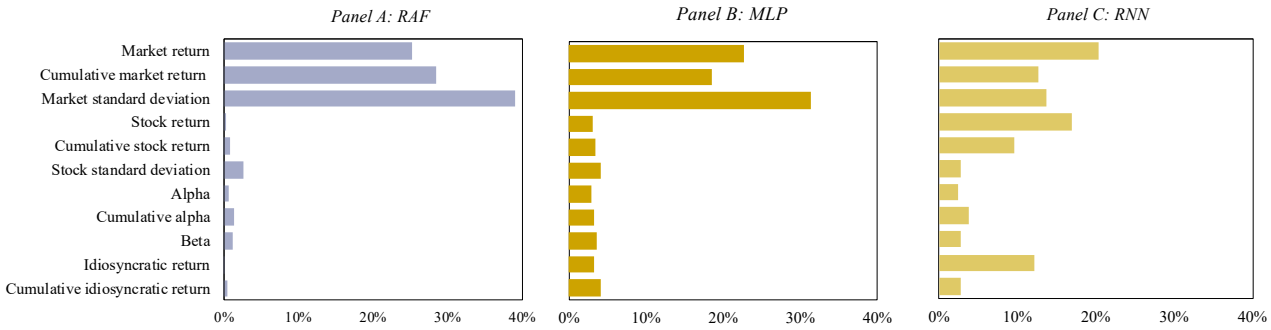


Figure 9.9 notes: The variable importance within each model is normalized to sum to one, allowing for interpretation of the relative importance for a given model. The variable importance is based on an average of five subsamples of 5000 observations for the out-of-sample period. (Source: Own creation).

The emphasis on the market variables across the three models is likely a result of how we design our machine learning problem: Recall that for a given month, the market variables are identical for all 500 stocks. Thus, it seems that the machine learning models apply the market variables to determine an overall return *level* (e.g., an average) for all stocks in one given month. While this is an important aspect for the machine learning models in order to minimize MSE (and thus, obtain a higher R^2), it provides no information about the cross-

section of the excess stock returns for a given month. The latter point is essential in the context of a stock-selection strategy, as the excess returns in the cross-section constitute the foundation for sorting stocks into portfolios. In predicting excess stock returns, the cross-sectional aspect is derived from the *stock-specific* variables. Hence, we follow the approach of Gu et al. (2019) and separate the market variables from the stock-specific variables in order to pinpoint the importance of the latter. Note that we renormalize the stock-specific variables to a [0:1] interval to investigate the relative importance of these variables, exclusively.

Figure 9.10: Relative importance of the stock-specific input variables for the three machine learning models for the out-of-sample period

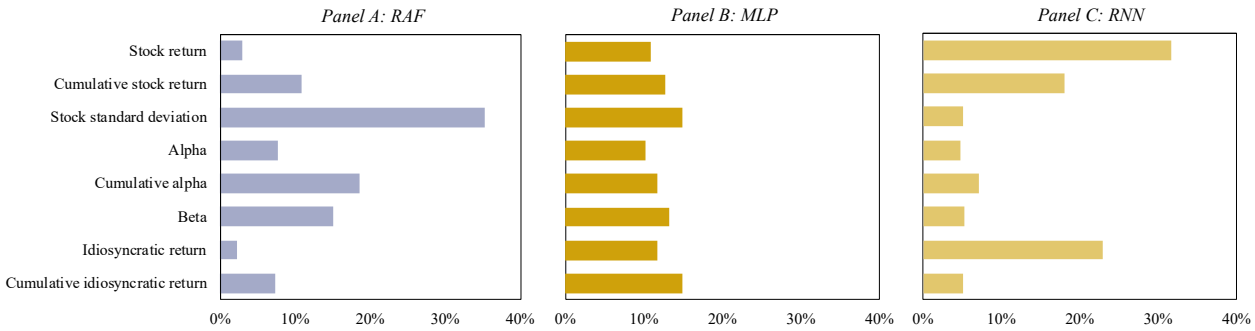


Figure 9.10 notes: The variable importance within each model are normalized to sum to one, allowing for interpretation of the relative importance for a given model. The variable importance is based on an average of five subsamples of 5000 observations for the out-of-sample period. (Source: Own creation).

Figure 9.10 depicts the relative importance of the stock-specific variables for each of the three machine learning models. All models draw predictive information from a broad set of input variables rather than relying on one or a few inputs. In general, the models are not in agreement regarding the most influential stock-level variables, indicating that no single input variable underlies the superior performance of the machine learning models. In fact, the RNN places great emphasis on exactly those input variables that are ascribed the least importance by the RAF model. More specifically, the RAF model relies mostly on the standard deviation of the stocks, the cumulative alpha, and beta when making predictions. In contrast, the RNN relies mostly on stock returns, cumulative stock returns, and idiosyncratic returns when making predictions. Finally, the MLP model appears to place similar emphasis on all stock-specific variables.

Broadly speaking, we observe an inconsistent pattern in the importance associated with the stock-specific variables for the machine learning models. The differences in the importance ascribed to the input variables are not surprising considering the low correlation observed among the models. However, *why* they diverge in the assigned importance remains unanswered. Thus, the importance of the input variables for the machine learning models offers limited clarity on what drives the superior performance of the MLS strategies. Rather, our findings imply that the explanation for the superior performance of the three strategies is elsewhere to be found. In this regard, we highlight the possibility of potential interaction effects *between* the input variables which is captured by the SHAP value. In fact, while the market variables in themselves do not contribute to the cross-sectional ranking of the stocks, interdependencies among the market variables and the stock-specific

variables serve as a potential explanation for the superior performance. For instance, interaction effects between the stock-specific variables and the market variables might explain the RNN strategy timing market rebounds. However, a detailed examination of the interaction effects amongst the input variables is beyond the scope of this thesis.

In summary, all our models ascribe high predictive power to the market variables. However, more importantly, they disagree on the importance of the stock-specific variables when predicting the cross-section of excess stock returns. Consequently, we are unable to identify a common denominator driving the superior performance of the MLS strategies. We discuss the implication of these findings in section 10.1.

9.6 Factor Analysis

In this section, we conduct a factor analysis, as often done in the financial literature, in another attempt to unfold the superior performance of the MLS strategies (e.g., Blitz et al., 2001, 2020; Daniel & Moskowitz, 2016). While both the input variable importance analysis and the factor analysis attempt to dissect the performance of the MLS strategies, a clear distinction should be made between the two analyses. The former analysis investigated which of the input variables are important for the machine learning models when making *predictions*. In contrast, this section explores whether additional risk factors besides the market are able to *explain* the profitability of the MLS strategies. Thus, we do not examine the predictive ability of the risk factors.

We conduct spanning tests in which we regress the daily excess returns of each of the three MLS strategies against the daily excess returns of the five Fama/French factors (Fama & French, 2015), the betting against beta factor (Frazzini & Pedersen, 2014) and our replicated momentum factor. Table 9.12 presents the results of the three OLS regressions over the out-of-sample period.

Table 9.12: Spanning tests of the three MLS strategies for the out-of-sample period

Panel A: RAF				*Panel B: MLP*				*Panel C: RNN*			
	Coefficient	Standard error	t-stat		Coefficient	Standard error	t-stat		Coefficient	Standard error	t-stat
α	**11.82**	0.00	(6.78)	α	**6.53**	0.00	(3.81)	α	**5.51**	0.00	(2.29)
Mkt-RF	**0.33**	0.01	(23.38)	Mkt-RF	**0.10**	0.02	(6.76)	Mkt-RF	**0.23**	0.02	(13.76)
SMB	**0.16**	0.02	(7.13)	SMB	**0.05**	0.02	(2.03)	SMB	**0.08**	0.03	(2.53)
HML	**0.08**	0.03	(2.52)	HML	**0.15**	0.03	(4.53)	HML	**0.45**	0.05	(9.08)
RMW	**-0.44**	0.03	(-13.06)	RMW	-0.06	0.04	(-1.69)	RMW	-0.01	0.06	(-0.14)
CMA	**-0.56**	0.05	(-12.31)	CMA	-0.08	0.05	(-1.66)	CMA	0.11	0.08	(1.38)
BAB	**-0.36**	0.03	(-12.24)	BAB	**-0.14**	0.03	(-5.19)	BAB	**-0.12**	0.04	(-3.53)
MOM	**0.31**	0.01	(25.60)	MOM	**0.06**	0.01	(4.86)	MOM	**0.38**	0.02	(16.77)

R2	Information ratio
58.53	1.09

Durbin-Watson
1.9

R2	Information ratio
8.69	0.62

Durbin-Watson
1.91

R2	Information ratio
27.27	0.40

Durbin-Watson
1.8

Table 9.12 notes: The table displays OLS regression outputs for the three MLS strategies when applying the Newey-West error estimations, for the out-of-sample period. The alpha coefficients of the regressions are annualized and are in percent. The R2 is also reported in percent. The three panels present the results of regressing the daily excess returns of each of the MLS strategies against the daily excess returns of the five Fama/French factors, betting against beta and our replicated momentum factor. Bold values are statistically significant at a 5% level. Note that the Durbin-Watson test statistic has a value near 2.0, implying that the Newey-West estimation was able to reduce the autocorrelation in the error terms of the models. Finally, the table reports the annualized information ratios for each of the three MLS strategies. (Source: Own creation).

Table 9.12 reports the results of the spanning tests. We first observe an R^2 of 58.53%, 8.69%, and 27.27%, for the RAF, MLP, and RNN strategy, respectively. Hence, the variations in the seven risk factors explain as much as 58% of the variations in the RAF strategy, but only 8.69% and 27.27% of the variations in the MLP and RNN strategies.

The spanning tests show highly significant annualized alphas of 11.82%, 6.53%, and 5.51% for the RAF, MLP, and RNN strategy, respectively, indicating that the excess returns of the strategies are not fully captured by the market factor and the additional six risk factors. In fact, the alphas of the RAF and MLP strategy *increase* when we adjust for the additional factors besides the market. The increase in alpha is caused by the two strategies' negative loadings on three of the additional factors (RMW, CMA, and BAB). In contrast, the alpha of the RNN strategy decreases when adjusting for additional factors besides the market, implying that the additional factors have driven part of the alpha for the RNN strategy. The high alpha of the RAF strategy is also reflected in its superior information ratio of 1.09 relative to the MLP and RNN strategy with information ratios of 0.62 and 0.4, respectively. For comparison, the RAF strategy obtained a Sharpe ratio of only 0.66 before adjusting for the risk factors.

Examining the beta coefficients of the MLS strategies, we document that *all* our strategies are positively and significantly loaded on the market and momentum factor while negatively loaded on the BAB factor. This is not surprising, as the input variables passed to the machine learning models contain market variables and momentum-related variables. However, the loading on the MOM factor for the MLP strategy is less profound compared to the other MLS strategies (though still significant). This observation is consistent with the low correlation we document between MLP and MOM.
Moreover, we observe that the MLS strategies load significantly and positively on the SMB and HML factors. In contrast, the strategies load negatively or insignificantly on the RMW and CMA factors. Notably, the RAF strategy has a highly significant negative loading on the two latter factors (-0.44 and -0.56, respectively).
The SMB, HML, RMW, and CMA factors are not closely linked to the momentum variables utilized for our machine learning models. Thus, we cannot explain why the MLS strategies agree on the sign convention of these factor loadings.

While an in-depth analysis of alpha and the loadings of the MLS strategies on the seven factors are beyond the scope of this thesis, we point to appendix A.17 for an overview of the 12-month rolling alphas and factor loadings. Non-surprisingly the 12-month rolling alpha of the MLP strategy is lower but less volatile relative to the other MLS strategies.

Summarizing this section, the expanded factor analysis offers supportive evidence for the profitability of the MLS strategies, however, it does not provide any further clarity on their superior performance. Thus, we remain limited in our understanding of the success of the MLS strategies.

9.7 Concluding Remarks on the Analysis of Momentum and Machines

We finalize this chapter by synthesizing the results of our research. First, when replicating the conventional MOM strategy, we document a momentum premium as a result of the winner portfolio significantly outperforming the loser portfolio during the total sample period. In addition, we confirm that the MOM strategy crashes following a bear market that rebounds abruptly. These findings on the conventional MOM strategy are in line with the momentum literature. Hence, we argue that our data sample constitutes a valid representation of the findings in the literature.

Against this background, we examine the predictive ability of three machine learning models that utilize momentum variables as inputs. We document a positive predictive R^2 for all the machine learning models with the MLP model exhibiting the best R^2 score of 0.75%. The predictive R^2s of all machine learning models are significantly higher than the R^2 of the baseline model. The inferior predictive performance of the baseline model is a result of it being highly susceptible to in-sample overfitting. Hence, the positive and improved predictive R^2s for the three machine learning models point to the value of incorporating complex predictor interactions, which are embedded in tree and neural networks but missed by a simple OLS regression (baseline) model. Moreover, we document that the predicted excess returns of the machine learning models exhibit significant positive Spearman rank correlations with the target excess return. The positive and significant Spearman rank correlations imply that the models understand the cross-section of excess stock returns which constitutes the foundation for utilizing the predictions for stock-selection strategies. Thus, we form three stock-selection strategies based on the predictions made by the machine learning models. We document a superior performance of the MLS strategies relative to the MOM strategy for the total out-of-sample period spanning January 1976 to December 2020 with a higher Sharpe Ratio, Sortino Ratio, and significant positive alpha over the MOM strategy.
We further zero in on two sub-periods in which the MOM strategy crashes to examine the performance of the three MLS strategies. We demonstrate that the superior performance of all the MLS strategies relative to the MOM strategy remains robust across both sub-periods. The MLP and RNN strategy emerge as the best performing strategies during the sub-periods. Notably, we find observations that point to the possibility of the RNN strategy timing market rebounds.

With no strategies being perfectly correlated, our findings accentuate the opportunity to counter the crash risk inherent to the MOM strategy by constructing an ensemble strategy. We form an ensemble strategy that equally weighs the excess returns of the MOM strategy and the three MLS strategies. We demonstrate that this strategy outperforms *all* four individual strategies with a Sharpe ratio of 0.83 and a Sortino ratio of 1.16. The superior performance of the ensemble strategy relative to the conventional MOM strategy remains robust across the two sub-periods.

In a modest attempt to understand what drives the superior performance of the three MLS strategies (and the ensemble strategy, implicitly), we conduct two distinct analyses. First, we analyze the importance of the input variables for our machine learning models when making predictions. We observe an inconsistent pattern in the importance ascribed to the stock-specific variables by the machine learning models. Hence, the analysis of the stock-specific variables offers limited interpretability on what drives the superior performance of the MLS strategies. Second, we examine if the profitability of the MLS strategies is explained by seven risk factors. We find that the performance of the MLS strategies cannot be fully captured by the risk factors, evident by their positive and significant alphas. In fact, the performance of the RAF strategy when adjusting for the seven risk factors, is substantially higher than its performance, when adjusting for the exposure to the market only. Hence, the expanded factor analysis underlines the profitability of the MLS strategies, however, it does not provide any further explanation of their superior performance. Our findings suggest that the complex nature of machine learning models fosters high accuracy of predictions but comes at a price of limited interpretability.

10 Discussion

This chapter discusses the results of our research. We initiate this chapter by discussing the comparability of the magnitude and direction of our findings to the extant literature. Subsequently, we explore the implications of our findings for academia as well as for practitioners. As the scope of this thesis is limited, several areas remain unexplored. Thus, the final section outlines the most relevant areas for further research.

10.1 Comparison of Findings to Extant Literature

The Conventional Momentum Strategy

We commence by exploring how the results of our replicated conventional momentum strategy compare to the findings of similar studies in the literature. We document that our replicated momentum strategy is subject to negative returns following a bear market that quickly rebounds i.e., momentum crashes. The presence of momentum crashes for the replicated momentum strategy is highly comparable to the findings of Daniel & Moskowitz (2016) (see also Cooper et al., 2004; Stivers & Sun, 2010). In fact, the five months with the worst returns of the replicated momentum strategy are identical to those identified by Daniel and Moskowitz (2016). However, the *magnitude* of our findings on momentum crashes differs, as the momentum crashes observed for our sample are smaller. For instance, the worst one-month return for our replicated momentum strategy is -45.16%, substantially smaller than the worst one-month return of -74.36%, documented by Daniel and Moskowitz (2016). The divergence in the magnitude of our findings is likely a result of our choice of data sample. Recall that our investment universe exclusively consists of the largest 500 stocks trading on NYSE, AMEX, and NASDAQ for a given month. Minding our data sample, it is not surprising that our findings vary from the literature that most commonly utilizes a data sample consisting of *all* stocks listed on the US stock market - including microcaps (Daniel & Moskowitz, 2016).

Machine Learning for Stock-Selection Strategies

The three machine learning models obtain predictive R^2s in the range between 0.59% to 0.75% when predicting excess returns. While these R^2s are unimpressive from a statistical standpoint solely, the R^2s are in line with the standards in the financial literature. The explanation for the low standards of R^2 within this field is nested in the low predictability of (excess) stock returns. Two fundamental properties of stock markets render stock prediction a difficult science. First, drastic and unexpected fluctuations are inevitable in stock prices due to the impact of unanticipated news that per definition is *random*. The noise in stock returns is consistent with the efficient market hypothesis, dictating that stock returns cannot be predicted. Second, even with stock market inefficiencies, the predictable signal of (excess) stock returns is small, as traders constantly seek to exploit such inefficiencies. Hence, the science of return predictions revolves around a very low signal-to-noise ratio which makes it difficult to document statistical findings with the same significance as in other disciplines that have astronomical datasets with much stronger predictive signals (Israel et al., 2020).

Thus, our findings of predictive R^2 must be seen in the light of the comparable literature within stock return prediction. In this regard, we note once again that little *methodical* research exists on the application of machine learning models for stock prediction - especially in the context of stock-selection strategies. Thus, no studies in the current literature are fully comparable with our findings. Our data sample differs from the data samples in the literature, and the momentum variables that we apply as inputs for the machine learning models have not formerly been applied in the literature. With this in mind, our results on R^2 are at most comparable with the findings of Gu et al. (2019) who (amongst other algorithms) deploy Random Forest and neural networks when predicting excess stock returns for their out-of-sample period (1987-2016). When limiting their universe to the top 1000 stocks in the US stock market, their best performing neural network obtains an R^2 of up to 0.7%, while Random Forest obtains 0.63%. These results are generated on the basis of a shorter time period relative to our data sample (30 versus 45 years, respectively) and with many more input variables relative to our study (900+ versus 108, respectively). However, the R^2s they obtain for the two models are remarkably similar to those documented in this thesis.

Having established the R^2 of our machine learning models to be broadly consistent with the literature, we point to the economic gains that an MLS strategy should also be associated with. Once again, the literature offers limited comparable results of the profitability of MLS strategies. Freyberger et al. (2020) apply adaptive group LASSO to approximate a nonlinear function for expected returns and demonstrate its superiority relative to a linear model. For their out-of-sample period (1991-2014), their long-short portfolio yields a Sharpe ratio of 1.33[34]. Gu et al. (2019) also document economic gains from their machine learning predictions[35]. The long-short portfolio based on their best model (a neural network) yields a Sharpe ratio of 1.69.

Our smaller stock universe and different sample period lead to limited comparability with the magnitude of their findings, as we document the highest Sharpe ratio of 0.67 for the RNN strategy, substantially lower than the Sharpe ratios demonstrated by Freyberger et al. (2020) and Gu et al. (2019). Still, the direction of our findings supports the documentation made in the literature on the economic gains of MLS strategies.

[34] For the equally weighted long-short portfolio based on a stock universe that excludes stocks below the 10th percentile on NYSE market capitalization as this is the most comparable portfolio to this thesis

[35] For the equally weighted long-short portfolio formed on the basis of a stock universe that excludes stocks below the 20th percentile on NYSE market capitalization as this is the most comparable portfolio to this thesis

In this regard, we note that several scholars document that economic gains are more prominent for a trading strategy that relies on an ensemble model rather than individual models (e.g., Krauss et al., 2017; Bao et al, 2017; Gu et al., 2019). We also confirm the potential of combining models into one ensemble, as our ensemble strategy outperforms all our individual stock-selection strategies in terms of both Sharpe and Sortino ratios.

Predictive Power of Input Variables

The superior performance of the MLS strategies is likely nested in the additional information that the strategies rely on. While the momentum strategy does not incorporate any information besides the 2-12 months cumulative returns, we provide additional stock-specific and market variables to the machine learning models. This enables them to leverage further insights when making predictions which comprise the foundation of the MLS strategies.

As machine learning models suffer from opacity, the drivers underlying the performance of the MLS strategies are highly challenging to extract and interpret. In a modest attempt to understand these drivers, we analyze the importance of each input variable when the three machine learning models make predictions. Our analysis shows that the three models ascribe some predictive power to *all* the stock-specific variables, however, they vary in terms of how much predictive power is ascribed to *each* of these variables.

Relating the results of our analysis to the literature, the findings of Gu et al. (2019) offer a potential explanation for why our machine learning models assign predictive power to *all* stock-specific variables. They document that the variables related to recent price trends are the most important for all their models (five out of the seven most influential stock-specific variables). In a similar manner, Fisher and Krauss (2018) document the high predictive power of past returns. As our stock-specific variables are related to price trends, one explanation for the machine learning models emphasizing all variables might simply be that all variables are important for the predictions made by the models. However, this does not explain why the models assign *different* importance to each of these variables.

A potential explanation for the differences is nested in the *interdependencies* amongst the variables: Note that Random Forest and neural networks are able to capture potentially complex interactions among the variables (Hastie et al., 2009; Moritz & Zimmermann, 2016; Israel et al., 2020). However, the three machine learning models vary in terms of architecture, and thus one model might be able to capture interdependencies amongst variables that the others cannot. For instance, recall that the RNN is able to memorize long and short-term dependencies, as opposed to the RAF and MLP models. As the interaction effects between the variables are incorporated into the SHAP value of a variable, this could explain why the values differ across models. While investigating interdependencies among the input variables are of substantial relevance, such an analysis lies beyond the scope of our research.

Regardless of whether one of the above justifications - or one not addressed in this section - can explain the difference in the predictive power ascribed to the stock-specific variables, we remain limited in our ability to interpret what drives the superior performance of the MLS strategies.

10.2 Implications of Findings

In bridging the chasm between two previously separated bodies of research; momentum strategies and machine learning, the implications of our findings carry relevance for a broad area of research. The following section discusses the most predominant implications of this thesis for academia and practitioners and thus, answers our final research question (six).

Academic Implications of Our Research

We first point out the implications of our research in relation to the literature that enhances the conventional momentum strategy (Barroso & Santa-Clara, 2015; Daniel & Moskowitz, 2016; Blitz et al., 2011, 2020; Hühn & Scholz, 2018). For our out-of-sample period, the ensemble strategy, which equally weights the predictions of the three MLS strategies and the replicated momentum strategy, approximately doubles the Sharpe ratio and Sortino ratio of the conventional momentum strategy. Moreover, this strategy exhibits much less crash risk with a maximum drawdown of 36.48%. The superior performance of the ensemble strategy relative to the conventional momentum strategy remains robust across the two sub-periods, namely the dot-com-bubble and the global financial crisis. Thus, placing the contribution of this thesis in its academic context, we document novel research that utilizes machine learning to enhance the performance of and mitigate the crash risk inherent to the conventional momentum strategy.

Our research further points to the benefits of combining insights from multiple studies. The three machine learning models deployed in our thesis rely on several findings that have each been documented individually in the enhanced momentum literature. Capitalizing on these combined insights, our machine learning models exhibit predictive abilities that can be successfully deployed for stock-selection strategies. Thus, our research points to the ability of machine learning models to extract formerly undiscovered patterns *across* the findings of the enhanced momentum literature.

In a broader academic context, our research contributes to the current debate on the potential of machine learning for investment management - more specifically for stock return prediction. We find evidence that supports the argument put forth by Arnott et al. (2019), namely that machine learning holds considerable promise for the development of successful trading strategies.

By documenting a superior predictive performance of our three machine learning models relative to a simple OLS regression, we further confirm the findings of Gu et al. (2019), who demonstrate that the superiority of tree-based models and neural networks rely on their ability to comprehend *complex and nonlinear* interactions between the input variables and excess stock returns.

Recapitulating the key academic implications of this thesis, we present a novel way to enhance the conventional momentum strategy and find evidence that helps justify the increasing role of machine learning in investment management.

Practical Implications of Our Research

The aforementioned complexity that lies at the heart of our machine learning models leads us to the *practical* implications of our research. Machine learning is frequently referred to as a black box; inputs are passed to the model and a prediction comes out, but the processes in between are opaque. Resonating Ghorbani, Abid & Zou (2019), it can be extremely challenging to draw meaningful interpretations of the underlying mechanisms of machine learning models. This challenge is amplified when the complexity of the models increases. The field of machine learning universally agrees on interpretability being the cornerstone for user trust in machine learning models (Ribeiro, Sing & Guestrin, 2016). The inability to correctly interpret the predictions of a model hinders insights into how the model may be improved as well as an understanding of the process being modeled. In contrast, the reasoning process of more simple predictive models is more understandable to humans. Hence, in some applications, simple models (e.g., linear models) are preferred for their ease of interpretation, even if they may be less accurate than complex ones (Lundberg & Lee, 2016).

The ability to understand the inner workings of an applied model is no different for investment managers and traders who seek to exploit inefficiencies in stock markets. While any trader or investment manager prefers a successful predictive model (inter alia), they can be averse to deploying historically reliable models that they *cannot* decipher (Israel et al., 2020). For instance, for an investment manager, the interpretability of a model is essential when clients or other major stakeholders demand an explanation of the predictive models. Thus, while investment managers prefer a model with more predictability to less, their fiduciary duty of comprehending and communicating the risks in their clients' portfolios also induces them to prefer more interpretable models. Israel et al. (2020) refer to this as a second "risk-return tradeoff" that any investment manager aspires to solve in the new era of machine learning and finance.

Against this background, we reach the crescendo of our discussion and place the research of this thesis in a practical setting: While we document the great potential of our machine learning models for creating profitable trading strategies, the difficulty of interpreting their inner workings hinder the practical implementation of our findings. Our unsuccessful attempt to unfold the underlying drivers of our models renders them inscrutable and confirms the notion of machine learning models being black boxes, as documented in the literature. Hence, our research directly taps into the newly identified "risk-return trade-off" that lies at the intersection of machine learning and finance. In this regard, we note that the trade-off between predictability and interpretability is not very different from the mean-variance trade-off already faced by investment managers: In the end, selecting a point on the "predictability/interpretability frontier" is ultimately a decision of the investment manager.

10.3 Further Research

Our research points to new ventures that may be of interest for further research. Many areas are relevant for further research, such as testing the robustness of our findings across geographies and asset classes, incorporating transaction costs, deploying other machine learning models, or including additional input variables (e.g., fundamental variables). However, this section highlights the three areas that we consider to be the most relevant.

First, the performance of the enhanced momentum strategies documented in the literature could be tested on our specific sample. As previously established, our research relies on the insights from studies that develop enhanced momentum strategies (Blitz et al. 2011, 2020; Barroso & Santa-Clara, 2015; Daniel & Moskowitz, 2016; Hühn & Scholz, 2018) for the construction of the momentum variables that are passed to our machine learning models. However, we have not tested the performance of, for instance, idiosyncratic and alpha momentum strategies on our data sample. We document both a strong momentum premium as well as the crashes of the momentum strategy in our sample, consistent with the momentum literature. Therefore, we assume that the findings on how to enhance momentum strategies (that rely on the entire US stock market) hold true for our smaller sample as well. However, we consider testing the performance of the enhanced momentum strategies on our sample a relevant topic for further research.

Second, we point to the notion put forth by Israel et al. (2020), namely that machine learning models need not be a black box. Thus, further research should address how to improve the interpretability of the predictions made by the machine learning models. Improving the interpretability should be considered at both a specific level for our research and on a more general level, emphasized in the literature:
For our specific research, one option to improve the interpretability of the machine learning models is to explore interaction effects between the input variables (Chen et al., 2020). As previously established, the superior performance of machine learning models is embedded in their ability to comprehend complex and nonlinear *interactions* between the variables. While we do not investigate such interaction effects in this thesis, exploring the interdependencies between the variables can offer insights on what drives the superior performance of the MLS strategies and thus, the ability to better interpret the predictions of the machine learning models.
The interpretability of machine learning models can also be improved on a more general level. For instance, researchers are progressing in the field of "explainable artificial intelligence" that focuses on improving the opacity of machine learning models to enable users to draw more meaningful and intuitive conclusions from the predictions made by the models (Zhang et al., 2018; Horel & Giesecke, 2019). Alternatively, machine learning methods can be incorporated into smaller sub-components of regular modeling approaches that are more interpretable (Israel et al., 2020).
The two examples (on a specific and general level) offer the potential to improve the interpretability of machine learning models and thus constitute tangible starting points for further research.

Finally, the regression problem which comprises the foundation of our research could be designed as a classification problem instead. The rationale for designing our machine learning problem as a regression problem is to ensure comparability of our results, by following the approach of the most profound literature that combines machine learning and finance (Gu et al., 2019). Reiterating section 4.1.2, machine learning models that solve regression problems aim to minimize the error between the predicted output and the observed output (MSE). For our specific regression problem, two components minimize MSE: Predicting the correct return *level* for all stocks and predicting the correct *cross-section* of stock returns for a given month. However, stock-selection strategies rely on the cross-section of stock returns, exclusively (i.e., not on the return level). We document that the machine learning models deployed in our thesis are able to predict the cross-section of

stock returns (at least to some degree), evident by positive and significant Spearman rank correlations. Recall that when we construct an MLS strategy, we predict excess stock returns for a given month, rank stocks based on these predictions, and subsequently place them into decile portfolios. An alternative method for creating an MLS strategy is to design the problem as a classification problem, where the machine learning models directly predict which decile portfolio a stock belongs to. Yet, such a classification problem would lead to lower interpretability, as predictions can solely be observed on a decile portfolio level, rather than on an individual stock level. Still, we consider the performance of a stock-selection strategy, based on predictions that solely concentrate on the cross-section of stock returns, as an interesting area for further research.

11 Conclusion

The benefits of applying machine learning in a variety of fields have spurred the interest in machine learning for investment management. The data-driven approach of machine learning offers traders and investment managers unprecedented opportunities to exploit contextual and nonlinear relationships in stock markets.

While this thesis finds that the field of machine learning for return prediction is still in its infancy, we document that the ability of machine learning to identify patterns in complex data constitutes a promising foundation for investment management. With the conventional momentum strategy calling for improvement, and a large body of research documenting various methods to improve the strategy, this thesis establishes that the momentum strategy poses an interesting case for reformation through machine learning. Against this background, we examine the potential of combining three machine learning models with the momentum strategy and, as such, bridge the chasm between two previously separated bodies of research. Specifically, we deploy Random Forest, the Multilayer Perceptron, and a Recurrent Neural Network with Long-Short-Term Memory cells.

For the out-of-sample period spanning January 1976 to December 2020, we backtest a conventional momentum strategy and three machine learning-based stock-selection strategies that rely on eleven momentum variables. First and foremost, we document a strong momentum premium during the last century and confirm the findings of Daniel and Moskowitz (2016), namely that the momentum strategy crashes following a bear market that rebounds abruptly. Thus, our data sample constitutes a valid representation of the findings documented by the momentum literature.

In light of these findings, we construct stock-selection strategies based on the predictions made by the three machine learning models. The stock-selection strategies exhibit superior performance relative to the momentum strategy for the out-of-sample period with a higher Sharpe Ratio, Sortino Ratio, and significant positive alpha over the momentum strategy. Our results remain robust when zeroing in on two sub-periods in which the momentum strategy crashes, namely the dot-com bubble and the global financial crisis. Our ensemble strategy that equally weighs the four stock-selection strategies outshines all individual strategies with a Sharpe ratio of 0.83 and a Sortino ratio of 1.16. Thus, we contribute to academia by documenting a novel method for enhancing the performance of the conventional momentum strategy through machine learning.

We investigate possible explanations for the superior performance of the machine learning-based stock-selection strategies (and the ensemble strategy, implicitly). When analyzing the importance of the input variables for the machine learning models we are unable to identify a common denominator that drives the superior performance of the strategies. An expanded factor analysis does not provide any further clarification. In fact, all machine learning-based stock-selection strategies exhibit significant and positive alphas after adjusting for seven risk factors.

Our research suggests that the complexity of machine learning models fosters high accuracy of predictions, and thus profitable stock-selection strategies, but comes at a price of limited interpretability. We find evidence that supports the notion of machine learning models being black boxes, as often referred to in the literature. The difficulty of interpreting the underlying mechanisms of our machine learning models hinders the practical implementation of our findings. If traders or investment managers are unable to understand the inner workings of machine learning models, they will likely be hesitant towards pursuing a machine learning-based stock-selection strategy, even if it exhibits superior performance. Thus, our research relates directly to the newly identified "predictability-interpretability trade-off" that lies at the junction of machine learning and investment management (Israel et al., 2020).

Concluding, while machine learning offers unprecedented opportunities for traders and investment managers, it carries difficulties of its own, that must be resolved in order to realize its full potential: The contribution of machine learning to investment management is not revolutionary - rather evolutionary.

12 Bibliography

Arnott, R., Harvey, C. R., & Markowitz, H. (2019). A backtesting protocol in the era of machine learning. *The Journal of Financial Data Science*, *1*(1), 64-74.

Arnott, R., Harvey, C. R., Kalesnik, V., & Linnainmaa, J. (2019). Alice's adventures in factorland: Three blunders that plague factor investing. *The Journal of Portfolio Management, 45*(4), 18-36.

Asness, C. S., Frazzini, A., & Pedersen, L. H. (2012). Leverage aversion and risk parity. *Financial Analysts Journal*, *68*(1), 47-59.

Asness, C. S., Moskowitz, T. J., & Pedersen, L. H. (2013). Value and momentum everywhere. *The Journal of Finance*, *68*(3), 929-985.

Bansal, R., Dittmar, R. F., & Lundblad, C. T. (2005). Consumption, dividends, and the cross section of equity returns. *The Journal of Finance, 60*(4), 1639-1672.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS One, *12*(7).

Barberis, N., Shleifer, A., & Vishny, R. (1998). A model of investor sentiment. *Journal of Financial Economics*, *49*(3), 307-343.

Barroso, P., & Santa-Clara, P. (2015). Momentum has its moments. *Journal of Financial Economics, 116*, 111-120.

Berk, J. B., Green, R. C., & Naik, V. (1999). Optimal investment, growth options, and security returns. *The Journal of Finance*, 54(5), 1553-1607.

Bernardo, A. E., & Ledoit, O. (2000). Gain, loss, and asset pricing. *Journal of Political Economy*, *108*(1), 144-172.

Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blitz, D., Hanauer, M. X., & Vidojevic, M. (2020). The idiosyncratic momentum anomaly. *International Review of Economics & Finance*, *69*, 932-957.

Blitz, D., Huij, J., & Martens, M. (2011). Residual momentum. *Journal of Empirical Finance*, *18*(3), 506-521

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint, arXiv:1012.2599.*

Brownlee, J. (2017). *Long short-term memory networks with python: Develop sequence prediction models with deep learning*. Machine Learning Mastery.

Cao, L., & Tay, F. E. (2001). Financial forecasting using support vector machines. *Neural Computing &*

Applications, *10*(2), 184-192.

Carhart, M. M., (1997). On persistence in mutual fund performance. *Journal of Finance, 52*(1), 57–82.

Carmines, E. & Zeller, R. (1979). *Reliability and validity assessment*. Sage publications.

Chan, L. K., Jegadeesh, N., & Lakonishok, J. (1996). Momentum strategies. *The Journal of Finance*, *51*(5), 1681-1713.

Chaves, D. B. (2016). Idiosyncratic momentum: US and international evidence. *The Journal of Investing*, 25(2), 64-76.

Chen, L., Pelger, M., & Zhu, J. (2020). Deep learning in asset pricing. *arXiv preprint, arXiv:1904.00745.*

Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, *83*, 187-205.

Chordia, T., & Shivakumar, L. (2002). Momentum, Business Cycle, and Time-Varying Expected Returns. *The Journal of Finance, 57*(2), 985-1019.

Chui, A. C., Titman, S., & Wei, K. J. (2010). Individualism and momentum around the world. *The Journal of Finance*, *65*(1), 361-392.

Chui, A. C., Wei, K. J., & Titman, S. (2000). Momentum, Legal Systems and Ownership Structure: An Analysis of Asian Stock Market. *Available at SSRN.*

Conrad, J., & Kaul, G. (1998). An anatomy of trading strategies. *The Review of Financial Studies,* 11(3), 489-519.

Cooper, M. J., Gutierrez Jr, R. C., & Hameed, A. (2004). Market states and momentum. *The Journal of Finance, 59*(3), 1345-1365.

Daniel, K. D., & Moskowitz, T. J. (2016). Momentum crashes. *Journal of Financial Economics, 122(1),* 221-247.

Daniel, K.D., Hirshleifer, D., Subrahmanyam, A., (1998). Investor psychology and security market under- and overreactions. *The Journal of Finance. 53*(6), 1839–1885.

Dawani. (2020). *Hands-On Mathematics for Deep Learning: Build a Solid Mathematical Foundation for Training Efficient Deep Neural Networks*, Packt Publishing.

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, *74*(366a), 427-431.

Diebold, F., and Mariano, R. 1995. Comparing predictive accuracy. *Journal of Business & Economic Statistics, 13*, 134-144.

Dolvin, S., & Foltice, B. (2017). Where has the trend gone? An update on momentum returns in the US Stock Market. *The Journal of Wealth Management*, *20*(2), 29-40.

Eldan, R., & Shamir, O. (2016, June). The power of depth for feedforward neural networks. *Conference on*

Learning Theory, 907-940. PMLR.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 383-417.

Fama, E. F., & French, K. R. (1992). The cross-section of expected stock returns. *The Journal of Finance*, *47*(2), 427-465.

Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, *33*(1), 3-56.

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics, 116*(1)*,* 1-22.

Fama, E., & French, K. (1996). Multifactor explanations of asset pricing anomalies. *The Journal of Finance*, *51*(1), 55-84.

Feng, G., He, J., & Polson, N. G. (2018). Deep learning for predicting asset returns. *arXiv preprint, arXiv:1804.09314*.

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

Freyberger, J., Neuhierl, A., & Weber, M. (2020). Dissecting characteristics nonparametrically. *The Review of Financial Studies*, *33*(5), 2326-2377.

Gârleanu, N., & Pedersen, L. H. (2013). Dynamic trading with predictable returns and transaction costs. *The Journal of Finance*, *68*(6), 2309-2340.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Ghorbani, A., Abid, A., & Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence, 33*(1), 3681-3688.

Glorot, X., Bordes, A., & Bengio, Y. (2011, June). Deep sparse rectifier neural networks. In Proceedings of the fourteenth international conference on artificial intelligence and statistics. *JMLR Workshop and Conference Proceedings*, 315-323.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*. MIT Press.

Griffin, J. M., Ji, X., & Martin, J. S. (2003). Momentum investing and business cycle risk: Evidence from pole to pole. *The Journal of Finance*, *58*(6), 2515-2547.

Grinblatt, M., & Han, B. (2005). Prospect theory, mental accounting, and momentum. *Journal of Financial Economics*, *78*(2), 311-339.

Grinblatt, M., Titman, S., & Wermers, R. (1995). Momentum investment strategies, portfolio performance, and herding: A study of mutual fund behavior. *The American Economic Review*, 1088-1105.

Grundy, B., & Martin, S. (2001). Understanding the Nature of the Risks and the Source of the Rewards to Momentum Investing. *The Review of Financial Studies, 14(1)*, 29-78.

Gu, S., Kelly, B., & Xiu, D. (2019), Empirical Asset Pricing via Machine Learning. *NBER Working Paper No. w25398.*

Gu, S., Kelly, B., & Xiu, D. (2021). Autoencoder asset pricing models. *Journal of Econometrics, 222(1),* 429-450.

Gutierrez Jr, R. C., & Prinsky, C. A. (2007). Momentum, reversal, and the trading behaviors of institutions. *Journal of Financial Markets*, *10*(1), 48-75.

Hanauer, M. X., & Windmüller, S. (2020). Enhanced momentum strategies. *Available at SSRN.*

Harvey, C. R., Liu, Y., & Zhu, H. (2016). … and the cross-section of expected returns. *The Review of Financial Studies*, *29*(1), 5-68.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media.

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: Deep portfolios. *Applied Stochastic Models in Business and Industry*, *33*(1), 3-12.

Hinton, G., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527-1554.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780.

Hong, H., & Stein, J. C. (1999). A unified theory of underreaction, momentum trading, and overreaction in asset markets. *The Journal of Finance, 54*(6), 2143-2184.

Hong, H., Lim, T., & Stein, J. C. (2000). Bad news travels slowly: Size, analyst coverage, and the profitability of momentum strategies. *The Journal of Finance*, *55*(1), 265-295.

Horel, E., & Giesecke, K. (2019). Towards explainable AI: Significance tests for neural networks. *arXiv preprint, arXiv:1902.06021*.

How, D., Loo, C. & Sahari, K. (2016). Behavior recognition for humanoid robots using long short-term memory. *International Journal of Advanced Robotic Systems*, *13*(6).

Huang, W., Nakamori, Y., & Wang, S (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, *32*(10), 2513-2522.

Hühn, H. L., & Scholz, H. (2018). Alpha momentum and price momentum. *International Journal of Financial Studies*, *6*(2), 49.

Ince, H., & Trafalis, T. (2008). Short term forecasting with support vector machines and application to stock price prediction. *International Journal of General Systems*, *37*(6), 677-687.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning,* 448-456. PMLR.

Israel, R., & Moskowitz, T. J. (2013). The role of shorting, firm size, and time on market anomalies. *Journal of Financial Economics, 108*(2), 275-301.

Israel, R., Kelly, B. T., and Moskowitz, T. J. (2020). Can Machines 'Learn' Finance? *Journal of Investment*

Management, 18(2).

Jegadeesh, N., & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance, 48(1)*, 65-91.

Jegadeesh, N., & Titman, S. (2001). Profitability of momentum strategies: An evaluation of alternative explanations. *The Journal of Finance*, *56*(2), 699-720.

Jegadeesh, N., & Titman, S. (2002). Cross-sectional and time-series determinants of momentum returns. *The Review of Financial Studies, 15*(1), 143-157.

Jensen, M. C. (1968). The performance of mutual funds in the period 1945-1964. *The Journal of Finance, 23*(2), 389-416.

Jensen, T. I., Kelly, B. T., & Pedersen, L. H. (2021). Is There A Replication Crisis In Finance?. *National Bureau of Economic Research.*

Johnson, T. C. (2002). Rational momentum effects. *The Journal of Finance, 57*(2), 585-608.

Jostova, G., Nikolova, S., Philipov, A., & Stahel, C. W. (2013). Momentum in corporate bond returns. *The Review of Financial Studies*, *26*(7), 1649-1693.

Kaastra, I., & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, *10*(3), 215-236.

Kelly, B. T., Pruitt, S., & Su, Y. (2019). Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics*, *134*(3), 501-524.

Kim, K. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1-2), 307-319.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization, *arXiv preprint, arXiv:1412.6980.*

Korajczyk, R. A., & Sadka, R. (2004). Are momentum profits robust to trading costs?. *The Journal of Finance*, *59*(3), 1039-1082.

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

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, *86*(1), 97-106.

Lazzeri, F. (2020). *Machine Learning for Time Series Forecasting with Python*. John Wiley & Sons.

Lewellen, J. (2002). Momentum and autocorrelation in stock returns. *The Review of Financial Studies*, 15(2), 533-564.

Lintner, J. (1965). The Valuation of Risk Assets and the Selection of Risky Investments in Stock Portfolios and Capital Budgets. *Review of Economics and Statistics, 47*, 13-37.

Lundberg, S., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *arXiv preprint, arXiv:1705.07874.*

Markowitz, H. (1952). Portfolio Selection. *Journal of Finance, 7*(1), 77-91.

Masters, T., 1993*, Practical Neural Network Recipes in C++.* Academic Press.

Menkhoff, L., Sarno, L., Schmeling, M., & Schrimpf, A. (2012). Currency momentum strategies. *Journal of Financial Economics*, *106*(3), 660-684.

Miffre, J., & Rallis, G. (2007). Momentum strategies in commodity futures markets. *Journal of Banking & Finance*, *31*(6), 1863-1886.

Mitchell, T. M. (1999). Machine learning and data mining. *Communications of the ACM*, *42*(11), 30-36.

Mitchell, T. M.(1997). *Machine Learning*, McGraw-Hill International Editions (McGraw-Hill).

Mockus, J., Tiesis, V., & Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards Global Optimization, 2,* 117-129.

Moritz, B., & Zimmermann, T. (2016). Tree-based conditional portfolio sorts: The relation between past and future stock returns. *Available at SSRN 2740751*.

Moskowitz, T. J., & Grinblatt, M. (1999). Do industries explain momentum?. *The Journal of Finance, 54(*4), 1249-1290.

Moskowitz, T. J., Ooi, Y. H., & Pedersen, L. H. (2012). Time series momentum. *Journal of Financial Economics*, *104*(2), 228-250.

Myers, J. L., Well, A., & Lorch, R. F. (2010). *Research design and statistical analysis.* Routledge.

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: A guide for data scientists*. O'Reilly Media.

Newey, W. K., & West, K. D. (1987). Hypothesis testing with efficient method of moments estimation. *International Economic Review*, 777-787.

Novy-Marx, R. (2012). Is momentum really momentum?. *Journal of Financial Economics, 103*(3), 429-453.

Novy-Marx, R. (2013). The other side of value: The gross profitability premium. *Journal of Financial Economics, 108*(1), 1-28.

Okunev, J., & White, D. (2003). Do momentum-based strategies still work in foreign currency markets?. *Journal of Financial and Quantitative Analysis*, 425-447.

Pang, X., Zhou, Y., Wang, P., Lin, W., & Chang, V. (2020). An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, *76*(3), 2098-2118.

Pedersen, L. H. (2015). *Efficiently inefficient*. Princeton University Press.

Phaisangittisagul, E. (2016, January). An analysis of the regularization between L2 and dropout in single hidden layer neural network. *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS),* 174-179. IEEE.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-agnostic interpretability of machine learning. *arXiv preprint, arXiv:1606.05386*.

Rolnick, D., & Tegmark, M. (2017). The power of deeper networks for expressing natural functions. *arXiv*

preprint, arXiv:1705.05502.

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

Rouwenhorst, K. (1998). International Momentum Strategies. *The Journal of Finance, 53(1),* 267-284.

Rouwenhorst, K. (1999). Local return factors and turnover in emerging stock markets. *The Journal of Finance*, *54*(4), 1439-1464.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation*. California Univ. San Diego La Jolla Inst. for Cognitive Science.

Sapp, T., & Tiwari, A. (2004). Does stock return momentum explain the "smart money" effect?. *The Journal of Finance*, *59*(6), 2605-2622.

Schmidt, A. B. (2011). *Financial Markets and Trading : An Introduction to Market Microstructure and Trading Strategies (1st ed.).* John Wiley & Sons.

Sharpe, W. F. (1964). Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk. *The Journal of Finance, 19*(3), 425-442.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, *39*(1), 119-138.

Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of Big Data*, *7*(1), 1-33.

Shen, S., Jiang, H., & Zhang, T. (2012). Stock market forecasting using machine learning algorithms. *Department of Electrical Engineering, Stanford University.*

Singh, S., & Walia, N. (2020). Momentum investing: A systematic literature review and bibliometric analysis. *Management Review Quarterly*, 1-27.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *arXiv preprint, arXiv:1206.2944.*

Sortino, F & Satchell, S. (2001). *Managing downside risk in financial markets*. Butterworth-Heinemann.

Sortino, F, & Price, L. (1994). Performance measurement in a downside risk framework. *The Journal of Investing, 3*(3), 59-64.

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint, arXiv:0912.3995*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

Stivers, C., & Sun, L. (2010). Cross-sectional return dispersion and time variation in value and momentum premiums. *Journal of Financial and Quantitative Analysis,* 987-1014.

Suthaharan, S. (2016). Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, *36*, 1-12. Springer.

Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega, 29*(4), 309-317.

Titman, S., Wei, K. J., & Xie, F. (2004). Capital investments and stock returns. *Journal of Financial and Quantitative Analysis, 39*(4), 677-700.

Trippi, R. R., & Turban, E. (1992). *Neural networks in finance and investing: Using artificial intelligence to improve real world performance*. McGraw-Hill, Inc.

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. S*cience*, *185*(4157), 1124-1131.

Wang, K. Q., & Xu, J. (2015). Market volatility and momentum. *Journal of Empirical Finance*, *30*, 79-91.

Wang, W., Li, W., Zhang, N., & Liu, K. (2020). Portfolio formation with preselection using deep learning from long-term financial data. *Expert Systems with Applications*, *143*, 113042.

Yin, R. K. (2018). *Case study research and applications: Design and methods*. Sage Publications.

Zhang, Q., Wu, Y. N., & Zhu, S. C. (2018). Interpretable convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 8827-8836. Chicago.