UCLouvain

LOUVAIN
School of Management

Louvain School of Management

Dynamic Shrinkage Estimation for Portfolio Optimisation

A Supervised Learning and Reinforcement Learning Approach

Author: Loïc Remy
Supervisor: Nathan Lassance
Academic year 2025-2026
Dissertation for the master of Business engineering with a focus on Financial Engineering
Daytime schedule

AI Declaration

During the preparation of this master's thesis, the author used Claude as an AI coding assistant and ChatGPT and DeepL Write as support tools for rewriting and reformulation, for the following purposes:

1. Code assistance and writing support: Claude was used to assist with parts of the code developed for this thesis and to support the understanding of certain programming functions and modules. ChatGPT and DeepL Write were used to help rewrite, rephrase, and reformulate parts of the text in order to improve clarity, readability, and academic writing style.
2. Author's responsibility: After using these AI tools, I carefully reviewed, revised, and edited all generated or reformulated content. I take full responsibility for the final content presented in this thesis.

By signing this declaration, I affirm that the content of this master's thesis reflects my original work, supported by the responsible use of AI tools.

Louvain-la-Neuve, 15th April 2026

Abstract

This master thesis investigates whether machine learning methods can improve covariance matrix estimation for global minimum variance portfolios by learning time-varying shrinkage intensities. In high-dimensional portfolio problems, the sample covariance matrix is often unstable, which can lead to extreme portfolio weights and poor out-of-sample risk control. Classical shrinkage estimators address this issue by combining the sample covariance matrix with a structured target, but they generally rely on a shrinkage intensity that is static or derived under stable distributional assumptions. This thesis instead studies shrinkage intensity as a state-dependent parameter that may vary with market conditions.

The empirical framework compares supervised learning and reinforcement learning approaches. Supervised models are trained to approximate an ex-post oracle shrinkage intensity using features extracted from rolling estimation windows, including volatility, correlation, eigenvalue and covariance-stability indicators. Reinforcement learning models treat shrinkage selection as a sequential decision problem and learn policies directly from realised portfolio outcomes. Two policy-gradient architectures are implemented: a feedforward policy gradient agent and a recurrent policy gradient agent based on a gated recurrent unit.

The results show that the oracle shrinkage intensity varies substantially over time and is linked to observable market regimes, supporting the idea that covariance shrinkage should be modelled dynamically. The oracle is also partly predictable, with nonlinear supervised models, especially tree-based methods, capturing more informative patterns than regularised linear models. In portfolio terms, learning-based shrinkage policies improve upon the sample covariance benchmark in the baseline setting, with the strongest gains coming from nonlinear models and, in particular, the recurrent reinforcement learning specification. However, robustness checks show that these improvements are not unconditional: performance depends on the estimation window, the dataset and the modelling framework.

Overall, the thesis shows that supervised learning and reinforcement learning can learn economically meaningful dynamic shrinkage policies for minimum-variance portfolio construction. The main contribution is to bridge classical covariance shrinkage with data-driven decision rules, showing that adaptive regularisation can improve portfolio stability when market conditions change.

Acknowledgments

I would first like to express my sincere gratitude to my supervisor, Nathan Lassance, for his guidance, advice, and support throughout the entire process of writing this master's thesis. His availability, constructive feedback, and valuable insights helped me greatly in developing this work and in improving both its academic and empirical dimensions.

I am also very grateful to LIDAM/LFIN and its PhD students for warmly welcoming me during my internship. Their kindness, advice, and practical tips on both the writing process and the research itself were extremely valuable. Being surrounded by such a stimulating academic environment was a real source of motivation.

I would also like to thank my parents, who have supported me throughout my life and have always brought joy, encouragement, and balance into it. I am especially grateful for their patience during the many moments when I tried to explain what my thesis was about, and for the impressive effort they made to look like they fully understood covariance shrinkage and reinforcement learning. Their support, real or strategically pretended, meant a lot to me.

I would also like to thank my brother Robin and my sister Rose for their presence, encouragement, and sense of humour. Their support helped make this demanding period lighter and more enjoyable.

A special thank you also goes to my friends from Kot Cinéforum and Kap Signes, who brightened my university life far beyond lectures, exams, and deadlines. I am grateful for the moments shared, the laughs, the projects, and the memories that made these years much richer and more meaningful.

Finally, I would like to thank all the friends I have met throughout my university journey. These many friendships have made my studies not only an academic experience, but also a deeply human and joyful one.

To all of them, thank you.

Finally, I would like to acknowledge that this achievement was accomplished despite a difficult personal journey, including my deafness, which made traditional learning more challenging, a brain cavernoma requiring delicate surgery, and epilepsy. These challenges never prevented me from persevering and pursuing my goals with determination.

I would like to end with a quote from a Japanese proverb that has accompanied me throughout this journey and that I hope may inspire others:

"Fall down seven times, stand up eight."

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Context and motivation

Accurate estimation of the covariance matrix of asset returns is a central problem in portfolio management. Within the classical mean-variance framework introduced by Markowitz (1952), portfolio risk is determined by the covariance structure of asset returns. As a result, the quality of covariance estimation plays a critical role in determining the stability and performance of optimal portfolio allocations.

In empirical applications, the covariance matrix must be estimated from historical data. The most common estimator is the sample covariance matrix. However, when the number of assets is large relative to the available sample size, the sample covariance matrix becomes highly unstable. Estimation errors are amplified by the optimisation procedure, often resulting in extreme portfolio weights and poor out-of-sample performance. This phenomenon has been widely documented in the literature on portfolio optimisation and is often referred to as error maximisation (Best and Grauer, 1991; Chopra and Ziemba, 1993; Michaud, 1989).

To mitigate these problems, a large body of research has focused on regularising covariance estimates. One of the most widely used approaches is covariance shrinkage, which combines the sample covariance matrix with a structured target matrix in order to reduce estimation variance. Seminal contributions by Ledoit and Wolf (2003, 2004) demonstrate that shrinkage estimators produce well-conditioned covariance matrices that significantly improve portfolio stability. More broadly, shrinkage methods can be interpreted as exploiting the bias-variance trade-off in covariance estimation in order to reduce estimation risk.

Subsequent research has further extended this framework. In particular, nonlinear shrinkage estimators have been proposed to improve the estimation of the eigenvalues of large covariance matrices under high-dimensional asymptotics (Ledoit and Wolf, 2004, 2020). These approaches

rely on results from random matrix theory to derive asymptotically optimal transformations of the sample eigenvalues while retaining the sample eigenvectors.

While these methods substantially improve covariance estimation, most shrinkage estimators rely on a constant shrinkage intensity that is assumed to be stable over time. Financial markets, however, exhibit time-varying volatility, evolving correlation structures and structural breaks (Stărică and Granger, 2005). In such environments, a fixed shrinkage coefficient may fail to adapt to changing market conditions.

1.2 Research gap

Recognising the limitations of static shrinkage estimators, recent research has explored the possibility of allowing the shrinkage intensity to vary over time. Data-driven approaches have been proposed in which the shrinkage coefficient is modelled as a function of observable market characteristics. For example, machine learning methods have been used to learn covariance regularisation parameters directly from historical data with the objective of improving portfolio risk estimation.

Despite these advances, determining optimal shrinkage policies remains challenging. Most existing approaches rely on static estimation procedures or supervised learning models that treat the problem as a cross-sectional prediction task. However, portfolio construction is inherently a sequential decision problem in which decisions must be made repeatedly over time and evaluated based on their future portfolio performance.

Reinforcement learning provides a natural framework to address this issue. In reinforcement learning, an agent interacts with an environment and learns a policy that maximises expected cumulative reward through sequential decision making (Sutton and Barto, 1998). Recent studies have applied reinforcement learning techniques to portfolio allocation problems, demonstrating their ability to adapt to changing market conditions and learn dynamic investment strategies (Moody and Saffell, 2001).

1.3 Research question

Building on these developments, this master thesis investigates whether data-driven methods can be used to learn dynamic shrinkage policies for covariance estimation in portfolio construction.

More specifically, the master thesis examines whether machine learning algorithms can be used to determine state-dependent shrinkage intensities that improve the out-of-sample performance of minimum-variance (GMV) portfolios relative to standard statistical and static estimation techniques.

The central research question can therefore be formulated as follows:

Can supervised learning and reinforcement learning methods learn time-varying covariance shrinkage policies that improve out-of-sample performance for minimum-variance portfolios?

1.4 Contributions

This master thesis contributes to the literature on covariance estimation and portfolio construction in several ways.

First, it proposes a data-driven framework in which the shrinkage intensity is modelled as a function of market information rather than being treated as a fixed parameter. This allows the covariance estimator to adapt to changing market environments.

Second, the master thesis compares different learning approaches for estimating shrinkage policies. In addition to supervised learning models, reinforcement learning algorithms are implemented in order to explicitly account for the sequential nature of portfolio decisions.

Third, the reinforcement learning framework is implemented using two policy gradient architectures. A policy gradient agent (PGA) based on a multilayer perceptron learns shrinkage decisions directly from the current return window, while a recurrent policy gradient agent (RPGA) based on a gated recurrent unit captures temporal dependencies in the return sequence.

Finally, the economic value of the proposed approach is evaluated in the context of global minimum variance (GMV) portfolios. The performance of the learned shrinkage policies is compared with traditional covariance estimators using an extensive out-of-sample analysis.

1.5 Structure of the master thesis

The remainder of this master thesis is structured as follows.

Chapter 1 reviews the existing literature on covariance estimation, with a particular focus on shrinkage methods and recent machine learning approaches to portfolio construction. It also introduces the conceptual framework underlying dynamic shrinkage.

Chapter 2 presents the empirical methodology. It describes the data, outlines the portfolio construction framework, and details the procedures used to estimate covariance matrices and implement both supervised and reinforcement learning approaches to shrinkage.

Chapter 3 reports the empirical results. It analyses the properties of the oracle shrinkage intensity, evaluates its predictability, and compares the out-of-sample performance of the different covariance estimators and learning-based strategies.

Finally, Chapter 4 concludes by summarising the main findings and discussing potential limitations as well as directions for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Risk estimation in mean-variance portfolio

Mean-variance portfolio optimisation, introduced by Markowitz (1952), provides the theoretical foundation for modern portfolio management. In this framework, an investor selects portfolio weights by minimising portfolio risk for a given expected return. Denoting w as the vector of portfolio weights and Σ as the covariance matrix of asset returns, the variance of the portfolio is given by

$$\sigma_p^2 = w^\top \Sigma w.$$

The optimal portfolio weights therefore depend critically on the accurate estimation of the covariance matrix.

In empirical applications, the covariance matrix is typically estimated using the sample covariance matrix. Given a matrix of asset returns $R_{1:T} \in \mathbb{R}^{T \times N}$ with T observations and where $R_t \in \mathbb{R}^N$ denotes the return vector at date t, the estimator is

$$S = \frac{1}{T-1} \sum_{t=1}^{T} (R_t - \bar{R})(R_t - \bar{R})^\top \quad \text{where} \quad \bar{R} = \frac{1}{T} \sum_{t=1}^{T} R_t$$

While this estimator is unbiased under standard assumptions, its use in portfolio optimisation can lead to significant estimation problems. In particular, the optimisation procedure amplifies estimation errors in the covariance matrix, often resulting in extreme and unstable portfolio weights. Michaud (1989) argues that mean-variance optimisation based on noisy parameter estimates tends to produce portfolios that appear optimal in-sample but perform poorly out-of-sample, a phenomenon often referred to as error maximisation. Similarly, Chopra and Ziemba (1993) show that even small estimation errors in the inputs of the optimisation problem can lead to large distortions in portfolio allocations and substantial deterioration in portfolio performance.

These estimation issues become particularly severe in modern financial applications characterised by a large number of assets relative to the available time series observations. In such high-dimensional settings, the ratio between the number of assets N and the sample size T is not negligible and may even approach or exceed one. In the high-dimensional asymptotic framework, both N and T grow simultaneously while their ratio converges to a constant $c = N/T \in (0, \infty)$, often referred to as the concentration ratio. Under these conditions, the classical properties of the sample covariance matrix no longer hold, as the estimator becomes poorly conditioned and may even be singular when $N > T$ (Ledoit and Wolf, 2004; Fan et al., 2011). In this framework, the sample covariance matrix is no longer a consistent estimator of the true covariance matrix, which further exacerbates estimation error in portfolio optimisation. These findings highlight the fundamental limitations of the sample covariance matrix in large-dimensional settings and motivate the development of improved covariance estimators designed specifically for high-dimensional environments.

2.2 Classical shrinkage estimation of covariance matrices

To address these limitations of the sample covariance matrix in high-dimensional settings, a large literature has developed improved covariance estimators designed to reduce estimation error and improve matrix conditioning. Among these approaches, shrinkage estimators have become particularly influential. The central idea is to combine the sample covariance matrix with a structured target matrix, thereby reducing estimation variance while introducing a controlled bias.

Formally, shrinkage estimators take the form

$$\hat{\Sigma} = (1 - \lambda)S + \lambda F$$

where S denotes the sample covariance matrix, F is a structured target matrix, and $\lambda \in [0, 1]$ is the shrinkage intensity controlling the weight assigned to each component. Typical targets include the scaled identity matrix or constant correlation matrices, which impose additional structure and improve the conditioning of the covariance estimator (Ledoit & Wolf, 2004).

2.2.1 Linear shrinkage estimator

The pioneering contribution of Ledoit and Wolf (2004) provides a data-driven approach for estimating the optimal linear shrinkage intensity. Their method derives an estimator for λ that minimises the mean squared error between the estimated covariance matrix and the true covariance matrix under large-dimensional asymptotics. The resulting estimator takes the form

$$\hat{\Sigma}_{\text{LS}} = (1 - \hat{\lambda})S + \hat{\lambda}F.$$

Empirical studies show that linear shrinkage estimators substantially improve the conditioning of the covariance matrix and lead to more stable portfolio allocations compared with the sample covariance matrix.

Further details on the linear Ledoit-Wolf shrinkage estimator and its role as a benchmark in this master thesis are provided in appendix B.1.

2.2.2 Nonlinear shrinkage estimators

While linear shrinkage applies a uniform adjustment to the covariance matrix, nonlinear shrinkage methods allow for more flexible adjustments. These approaches typically modify the eigenvalues of the sample covariance matrix while preserving the corresponding eigenvectors.

The spectral decomposition of the sample covariance matrix is given by $S = U\Lambda U^\top$ where $U = [u_1, \ldots, u_p]$ is the orthogonal matrix whose columns are the sample eigenvetors of S, and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_p)$ is the diagonal matrix of sample eigenvalues. Nonlinear shrinkage estimators preserve the sample eigenvectors and replace the sample eigenvalues by shrunk eigenvalues, yielding

$$\hat{\Sigma}_{\text{NS}} = U\tilde{\Lambda}U^\top.$$

where $\tilde{\Lambda} = \text{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_p)$. Such estimators are often derived using tools from random matrix theory and have been shown to provide significant improvements in high-dimensional environments. In particular, Ledoit and Wolf (2020) propose an analytical nonlinear shrinkage estimator that performs well when the number of assets is large relative to the sample size.

2.2.3 Bias-variance trade-off interpretation

The effectiveness of shrinkage estimators can be understood through the bias-variance trade-off. Although the sample covariance matrix is unbiased, it exhibits high estimation variance in high-dimensional settings. Shrinkage estimators introduce a small bias by combining the sample covariance matrix with a structured target matrix, but this bias is compensated by a significant reduction in variance. As a result, shrinkage estimators often achieve lower mean squared estimation error and produce more stable portfolio allocations.

2.3 Dynamic shrinkage under non-stationarity

Classical shrinkage estimators are generally derived under the assumption of stationarity, where the covariance matrix is treated as a fixed population object and the shrinkage intensity is chosen to minimise expected quadratic loss under stable distributional assumptions. However, financial return series exhibit significant time variation in their second moments. Ignoring these dynamics may reduce the effectiveness of static shrinkage rules. This section therefore examines the implications of non-stationarity for covariance estimation and motivates the interpretation

of shrinkage intensity as a time-varying parameter.

2.3.1 Time variation in covariance structure

A large empirical literature documents that volatility and correlations in financial returns vary over time. The autoregressive conditional heteroskedasticity framework of R. F. Engle (1982) shows that conditional variances evolve dynamically, while Bollerslev (1986) extends this approach through the GARCH model. In the multivariate setting, Bollerslev (1990) proposes the constant conditional correlation (CCC) model, and R. Engle (2002) introduces the dynamic conditional correlation (DCC) framework, allowing correlations to evolve over time.

Beyond parametric volatility models, numerous studies document structural instability in covariance matrices. Market crises and macroeconomic regime shifts can induce substantial changes in cross-asset dependence. For example, Longin and Solnik (2001) show that international correlations tend to increase during market downturns, while Ang and Bekaert (2002) provide evidence that volatility dynamics depend on the prevailing market regime.

These findings suggest that the covariance matrix is more appropriately interpreted as a time-varying process rather than a fixed parameter. In this framework, estimation error is compounded by structural change, since past observations may provide only imperfect information about the current covariance structure.

2.3.2 Shrinkage intensity as a state-dependent parameter

In classical shrinkage estimation, the shrinkage intensity λ is treated as a constant parameter selected to minimise a statistical loss, such as expected quadratic loss. Yet this objective is not fully consistent with the economic goal of global minimum-variance portfolio construction. A covariance estimator may be close to the true covariance matrix in terms of matrix loss while still producing portfolio weights that lead to high realised variance out of sample. Furthermore, when covariance matrices vary over time, the associated bias-variance trade-off also becomes time-dependent. As a result, the optimal shrinkage coefficient should depend on the relative importance of the estimation variance of the sample covariance matrix and the bias induced by the shrinkage target in the prevailing market environment.

If these quantities vary across market conditions, as suggested by conditional heteroskedasticity models, the optimal shrinkage intensity should also adjust accordingly. From this perspective, shrinkage intensity can be interpreted as a function of the prevailing market state, reflecting the stability of recent covariance estimates, the reliability of the shrinkage target and the presence of structural changes in the dependent structure.

This interpretation provides a natural bridge between classical covariance regularisation and dynamic portfolio allocation, suggesting that shrinkage intensity should be treated as a time-varying parameter rather than a fixed constant.

2.4 Machine learning for portfolio construction

Advances in statistical learning and the increasing availability of financial data have led to growing interest in the application of machine learning methods to portfolio construction. Unlike traditional econometric approaches, machine learning algorithms can accommodate high-dimensional predictors, nonlinear relationships and complex interactions among variables. These characteristics are particularly relevant in financial markets, where asset returns and risk measures often depend on a large set of observable state variables and exhibit substantial nonlinearity.

Recent empirical research demonstrates that machine learning models can extract economically meaningful patterns from financial data and improve predictive performance relative to traditional linear models. For example, Gu et al. (2020) show that a wide range of machine learning methods outperforms classical factor models in predicting cross-sectional stock returns. More broadly, statistical learning provides a flexible tool for approximating complex mappings between observable market characteristics and latent quantities relevant for investment decisions.

Beyond return prediction, machine learning has increasingly been applied directly to portfolio construction. In this setting, learning algorithms estimate mappings from observable characteristics portfolio decisions or risk parameters that improve out-of-sample portfolio performance. A notable example is the supervised portfolio framework developed by Chevalier et al. (2022) in which machine learning models map firm characteristics directly into portfolio weights. This approach integrates prediction and portfolio optimisation within a unified framework, allowing the learning algorithm to directly target portfolio performance objectives.

More recently, machine learning methods have been used to estimate parameters involved in covariance matrix regularisation. Rather than deriving shrinkage intensities solely from asymptotic statistical arguments, data-driven approaches learn the shrinkage coefficient directly from historical data with the objective of improving portfolio risk estimation (De Nard and Kostovic, 2026; Mattera and Mattera, 2023). In this framework, shrinkage intensity can be interpreted as a function of observable market characteristics, allowing covariance estimation to adapt to changing market conditions.

2.5 Reinforcement learning for portfolio management

Reinforcement learning (RL) provides an alternative framework for financial decision-making by modelling portfolio management as a sequential decision problem under uncertainty. In contrast to supervised learning methods, which rely on predicting target variables from historical data, reinforcement learning focuses on learning optimal decision rules through interaction with a stochastic environment. In this framework, an agent observes the current state of the market, selects an action and receives a reward signal reflecting the economic performance of the

resulting portfolio decision. Over time, the agent updates its policy in order to maximise expected cumulative rewards (Sutton and Barto, 1998).

The reinforcement learning framework is particularly well suited for financial environments characterised by uncertainty, non-stationarity, and sequential dependence. Portfolio decisions are inherently dynamic, as allocation choices made today influence future portfolio performance and risk exposure. By explicitly accounting for these intertemporal effects, RL provides a flexible approach to modelling dynamic investment strategies. Early applications of reinforcement learning in finance include the work of Moody and Saffell (2001), who demonstrate how RL algorithms can be applied to adaptive trading strategies and dynamic asset allocation.

More recently, reinforcement learning methods have been applied to portfolio optimization problems involving large-dimensional covariance matrices. In this context, the shrinkage intensity used in covariance estimation can be treated as a decision variable selected by a learning agent. The agent updates its policy based on realised portfolio performance, learning how to adjust shrinkage intensity dynamically in response to evolving market conditions. For example, Mattera and Mattera (2023) propose a reinforcement learning framework in which the shrinkage parameter is chosen by the agent with the objective of minimising portfolio risk.

These developments illustrate how reinforcement learning can be used to adapt portfolio decisions to changing market environments. By learning policies directly from realised economic outcomes, RL approaches provide a flexible mechanism for adjusting risk estimation and portfolio allocation in dynamic financial settings. In the context of covariance estimation, this perspective interprets shrinkage intensity not as a fixed statistical parameter but as a control variable that can be adjusted sequentially as new market information becomes available.

2.6 Conceptual synthesis: prediction vs. sequential control

The literature reviewed above suggests two distinct approaches to dynamic covariance shrinkage. Classical shrinkage estimators treat the shrinkage intensity as a statistical parameter derived under asymptotic assumptions, whereas recent machine learning approaches model it as a data-driven object. In supervised learning, the shrinkage intensity is estimated as a conditional function of observable market characteristics, $\lambda_t = f(x_t)$. Reinforcement learning provides a different perspective by treating shrinkage intensity as an action within a sequential decision-making problem, where policies are updated according to realised portfolio performance.

These two paradigms correspond to different views of financial decision-making: prediction versus adaptive control. Supervised learning seeks to approximate an ex-post optimal shrinkage rule from labelled data, whereas reinforcement learning learns a sequential decision rule directly from realised portfolio outcomes. This thesis contributes to the literature by comparing these two approaches within a common empirical framework for dynamic covariance shrinkage.

CHAPTER 3

EMPIRICAL METHODOLOGY

3.1 Data description

The empirical analysis relies on daily equity returns of S&P500 constituent firms, together with a series of risk-free rates obtained from the Kenneth French Data Library. The risk-free rate series spans the period from January 1, 1990 to December 31, 2024 and is used exclusively for performance evaluation, more precisely for the computation of excess returns and Sharpe ratios.

A balanced panel of S&P 500 stocks is constructed. The initial universe includes all firms that appeared in the index at any time during the sample period (approximately 1070 firms). To avoid complications arising from entry and exit of firms, only companies that remain continuously listed throughout the entire period are kept. This filtering procedure ensures a consistent cross-sectional dimension and avoids survivorship-related inconsistencies in covariance estimation.

Finally, as a robustness check, three daily datasets were retrieved from the Kenneth French Data Library: the 100 portfolios formed on size and book-to-market, the 100 portfolios formed on size and operating profitability, and the 100 portfolios formed on size and investment. These three datasets span from 1990 to 2024 and were merged together. Finally, the portfolio columns with missing values were extracted from the final dataset.

- S&P500 data consists of 286 stocks (N) over 8817 trading days.
- The merged portfolio consists of 294 portfolios (N) over 8817 trading days.

3.2 Rolling estimation windows

To replicate a realistic investment process and avoid look-ahead bias, portfolio construction follows a rolling estimation framework combined with a forward evaluation period. At each rebalancing date t, model inputs are computed using a fixed-length backward window, while

portfolio performance is evaluated over a subsequent forward block. The structure of this procedure is illustrated in figure 3.1.

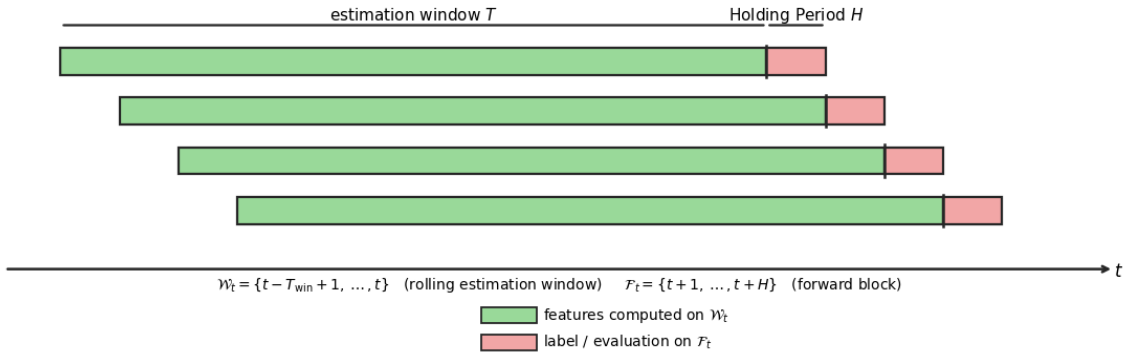


Figure 3.1: Rolling estimation and forward evaluation framework

3.2.1 *Estimation window*

At time t, all quantities required for portfolio construction are estimated using the previous L observations.

$$\mathcal{W}_t = \{t - L + 1, \ldots, t\}$$

which corresponds to the set of return observations $R_{t-L+1:t}$.

Within each window $\mathcal{W}_t$, we compute the sample covariance matrix, shrinkage target and the feature variables used to predict the shrinkage intensity λ_t.

In the baseline specification, the estimation window length is set to $L = 252$ trading days, corresponding approximately to one year of daily observations.

3.2.2 *Forward holding period*

After estimating the covariance matrix and determining the shrinkage intensity at time t, portfolio weights are computed and held over a forward evaluation block:

$$\mathcal{F}_t = \{t + 1, \ldots, t + H\}$$

corresponding to the sequence of future returns $R_{t+1:t+H}$.

In the baseline implementation, $H = 21$, corresponding roughly to one trading month. Portfolio weights are computed at the rebalancing date t and then kept throughout the forward holding period $F_t = \{t + 1, \ldots, t + H\}$, so that no interim rebalancing takes place within the block. At the next rebalancing date, the covariance matrix is re-estimated using the updated rolling window and a new set of portfolio weights is formed. Returns are therefore realised strictly out of sample over each holding period.

3.2.3 *Ex-post oracle shrinkage intensity*

For each estimation window $\mathcal{W}_t$, an oracle shrinkage intensity λ_t^* is computed. The oracle corresponds to the value of $\lambda \in [0, 1]$ that minimises realised portfolio volatility over the forward block:

$$\lambda_t^* = \arg\min_{\lambda \in [0,1]} \sigma\big(w(\lambda; \mathcal{W}_t), \mathcal{F}_t\big)$$

where $\hat{\Sigma}_t(\lambda) = (1 - \lambda)S_t + \lambda F_t$ is the shrunk covariance shrinkage estimated using the returns in $\mathcal{W}_t$, and $w(\lambda, \mathcal{W}_t) = \frac{\hat{\Sigma}_t^{-1}(\lambda)\mathbf{1}}{\mathbf{1}^\top \hat{\Sigma}_t(\lambda)\mathbf{1}}$ denotes the corresponding GMV portfolio weights.

This procedure ensures strict temporal separation between estimation and evaluation and provides the target variable used in the supervised learning framework.

3.3 Portfolio construction framework

Portfolio allocation is based on the GMV portfolio. Given a covariance matrix estimate $\hat{\Sigma}_t$ at time t, the GMV portfolio weights are defined as

$$w_t = \frac{\hat{\Sigma}_t^{-1}\mathbf{1}}{\mathbf{1}^\top \hat{\Sigma}_t^{-1}\mathbf{1}},$$

where $\mathbf{1}$ denotes a N-dimensional vector of ones.

This portfolio minimises portfolio variance subject to full-investment constraint $\mathbf{1}^\top w_t = 1$. The GMV portfolio is widely used in the covariance estimation literature because its performance depends directly on the accuracy of the estimated covariance matrix.

At each rebalancing date, portfolio weights are computed using the covariance estimate obtained from the rolling estimation window described in Section 3.2. The resulting portfolio is then held over the forward evaluation period.

3.4 Covariance shrinkage framework

Covariance matrices are estimated using a shrinkage estimator that combines the sample covariance matrix with a structured target matrix. The estimator takes the form

$$\hat{\Sigma}_t(\lambda_t) = (1 - \lambda_t)S_t + \lambda_t F_t,$$

where S_t denotes the sample covariance matrix computed over the estimation window and F_t is the shrinkage target matrix, and $\lambda_t \in [0, 1]$ is the shrinkage intensity controlling the degree of regularisation.

In this study, the selected target matrix is the scaled identity matrix.

$$F_t = \bar{\sigma}_t^2 I$$

where I is the $N \times N$ identity matrix and $\bar{\sigma}_t^2$ denotes the average of the sample variances across assets. This specification corresponds to a covariance matrix with identical variances and zero correlations between assets, providing a well-conditioned benchmark toward which the sample covariance matrix is shrunk.

3.5 Construction of the supervised learning dataset

The supervised learning framework aims to estimate the mapping between observable market characteristics and the optimal shrinkage intensity. Formally, the objective is to learn a function

$$\lambda_t^* = f(x_t)$$

where x_t denotes a vector of features summarising the state of the market at time t and λ_t^* represents the oracle shrinkage intensity minimising the realised volatility on the forward holding period.

The formal definition of the learning problem are developed in appendix A.5.

3.5.1 Target variable

The target variable is defined as the ex-post optimal shrinkage intensity λ_t^* obtained from the rolling-window procedure described in section 3.2. For each estimation window $\mathcal{W}_t$, the oracle shrinkage coefficient is computed as explained in the section 3.2.3.

3.5.2 Feature variables

The feature vector x_t contains summary statistics of the return distribution computed over the estimation window $\mathcal{W}_t$. These variables capture different aspects of the covariance structure and the stability of the sample covariance matrix. Table 3.1 summarizes the set of features used in the supervised learning model.

Table 3.1: Feature variables used in the supervised learning model

Feature	Definition
mean_var	Cross-sectional average of asset return variances.
mean_corr	Average pairwise correlation between asset returns.
eig_concentration	Ratio of the largest eigenvalue of the covariance matrix to the total variance.
cond_proxy	$\log\left(\frac{\lambda_{\max}(S_t)}{\lambda_{\min}(S_t)}\right)$, a proxy for the log-condition number of the sample covariance matrix.
eig_dispersion	$\frac{\mathrm{sd}(\lambda_i(S_t))}{\mathrm{mean}(\lambda_i(S_t))}$ measuring the relative dispersion of the eigenvalues.
ew_vol	Realised volatility of the equally weighted portfolio over the estimation window.
cs_mean_vol	Cross-sectional mean of individual asset volatilities.
cs_disp_vol	Standard deviation of asset volatilities across assets.
mean_abs_corr	Average absolute pairwise correlation across assets.
corr_std	Standard deviation of pairwise correlations across assets.
corr_p90	90th percentile of the pairwise correlation distribution.
corr_p10	10th percentile of the pairwise correlation distribution.
effective_rank	Participation-ratio measure of the effective dimensionality of the covariance spectrum $\frac{(\sum_i \lambda_i)^2}{\sum_i \lambda_i^2}$.
cov_shift	Relative Frobenius-norm difference between covariance matrices estimated on the first and second halves of the estimation window.

3.5.3 Theoretical justification of the supervised learning objective

The supervised learning framework is designed to approximate the shrinkage intensity that minimises future portfolio risk. For each rebalancing date t, define the mapping $g_t(\lambda)$ as the realised volatility of the GMV portfolio constructed using the covariance estimator $\hat{\Sigma}_t(\lambda)$ and evaluated over the forward holding period.

The optimal shrinkage intensity is then defined as

$$\lambda_t^* = \arg\min_{\lambda \in [0,1]} g_t(\lambda),$$

which corresponds to the oracle choice that minimises out-of-sample volatility.

The key property underlying the supervised approach is that the function $g_t(\lambda)$ is Lipschitz continuous in λ under mild regularity conditions. In otherwords, Lipschitz continuity means that for any two admissible shrinkage intensities $\lambda, \lambda' \in [0, 1]$, there exists a finite constant L_t such that

$$g_t(\lambda) - g_t(\lambda') \leq L_t|\lambda - \lambda'|.$$

In particular, the positive definiteness and boundedness of the shrinkage estimator ensure that

small variations in the shrinkage intensity lead to controlled variations in the inverse covariance matrix and, consequently, in portfolio weights and realised volatility.

This regularity implies that the volatility regret of any estimated shrinkage rule $\hat{\lambda}_t$ satisfies

$$\Delta_t := g_t(\hat{\lambda}_t) - g_t(\lambda_t^*) \leq L_t |\hat{\lambda}_t - \lambda_t^*|,$$

where L_t is a finite Lipschitz constant. The proof of this regret bound is provided in appendix A.6.

Taking expectations and applying standard inequalities yields that the expected regret is bounded by a term proportional to the root mean squared error (RMSE) of the shrinkage prediction. As a result, minimising the prediction error of $\hat{\lambda}_t$ with respect to the oracle λ_t^* directly controls an upper bound on expected out-of-sample volatility.

This result provides a theoretical justification for the supervised learning formulation. Rather than optimising portfolio volatility directly, which is difficult in a predictive setting, the problem is recast as a regression task on the oracle shrinkage intensity. Under the Lipschitz property, accurate prediction of λ_t^* is sufficient to ensure good out-of-sample risk performance.

The full derivations and proofs are provided in Appendix A.

3.6 Model selection, model interpretation and hyperparameter optimisation

3.6.1 Training and testing samples

To ensure a strict separation between model estimation and performance evaluation, the supervised learning dataset is divided into a training period and an out-of-sample testing period.

All machine learning models are estimated using observations up to December 2014, while the period from January 2015 to December 2024 is reserved exclusively for out-of-sample evaluation. Formally, letting t denote the rebalancing dates generated by the rolling-window procedure, the sample is divided as follows:

- Training sample: $t \leq 2014$
- Testing sample: $2015 \leq t \leq 2024$

The training sample is used for model estimation and hyperparameter selection. The testing sample is not used during model training or model selection and is reserved exclusively for the evaluation of out-of-sample portfolio performance.

During the testing period, predicted shrinkage intensities are generated using the trained models and incorporated into the covariance shrinkage framework described in section 3.4 to construct portfolio weights. This procedure guarantees that all reported results correspond to genuine

out-of-sample performance.

3.6.2 Supervised learning models

This study considers several supervised learning models to predict the oracle shrinkage intensity based on the feature variables described in section 3.5. The selected models include both penalised linear regression methods and nonlinear tree-based methods, these are further explained in details in appendix B.3.2.

The first group consists of regularised linear models. Ridge regression introduces an ℓ_2 penalty to the least-square objective in order to reduce coefficient variance when predictors are highly correlated (Hoerl & Kennard, 1970). Lasso regression instead relies on an ℓ_1 penalty, which can shrink some coefficients exactly to zero and therefore performs variable selection (Tibshirani, 1996). Elastic Net combines both penalties, providing a compromise between the stability of Ridge regression and the sparsity properties of Lasso (Zou & Hastie, 2005).

The second group consists of tree-based ensemble methods. Random Forest constructs an ensemble of decision trees estimated on bootstrap samples of the training data and aggregates their predictions, allowing the model to capture nonlinear relationships and interaction effects (Breiman, 2001). Extreme gradient boosting (XGBoost) is a gradient boosting algorithm in which trees are sequentially added to improve the prediction errors of the previous ensemble (Chen & Guestrin, 2016).

Together, these models provide a range of predictive approaches, from linear specifications to highly flexible nonlinear methods, allowing the analysis to assess whether the relationship between market conditions and the optimal shrinkage intensity is primarily linear or nonlinear. This does not imply that neural networks are unsuitable in general, but rather that they were not retained in the supervised learning benchmark because their complexity is less well suited to the effective size and structure of the supervised dataset.

3.6.3 Hyperparameter optimisation (HPO)

Model hyperparameters are selected using the training sample only in order to avoid any contamination of the out-of-sample evaluation. The search for optimal hyperparameter configurations is performed using the Optuna optimisation framework (Akiba et al., 2019). The optimisation relies on the Tree-structured Parzen Estimator (TPE) sampler, a Bayesian optimisation algorithm that models the distribution of promising and non-promising hyperparameter configurations in order to efficiently guide the search toward regions of the parameter space with better expected performance (Watanabe, 2023).

For each candidate configuration, the model performance is evaluated using time-series cross-validation. This procedure preserves the temporal ordering of the data by iteratively training the

model on an expanding window of past observations and validating it on subsequent observations, thereby preventing look-ahead bias.

To improve computational efficiency, the optimisation procedure also relies on Optuna's pruning mechanism. Trials that exhibit poor immediate performance during cross-validation are terminated early, allowing computational resources to be focused on more promising hyperparameter configurations.

The objective of the optimisation procedure is to minimise the prediction error of the oracle shrinkage intensity on the validation folds. For each model, the hyperparameter configuration achieving the lowest average validation error is retained. The resulting model specification is then re-estimated on the full training sample and applied to the testing period to generate out-of-sample predictions of the shrinkage intensity.

For further details, these methodologies are referred in appendix B.4.

3.6.4 Time-series cross-validation

Hyperparameter selection relies on a time-series cross-validation procedure that preserves the temporal ordering of the data. Unlike standard cross-validation, observations cannot be randomly shuffled because doing so would introduce look-ahead bias.

The validation procedure follows an expanding-window scheme. For each fold, the model is trained on a set of past observations and validated on a subsequent block of observations. The training window is then expanded forward in time and the procedure is repeated, ensuring that the model is always trained using information available prior to the validation period. A schematic representation of this procedure is shown in Figure 3.2.

Because the oracle shrinkage intensity is computed using a forward evaluation window, observations near the validation boundary may contain overlapping information between training and validation samples. Following De Prado (2018), the cross-validation procedure is therefore purged, meaning that observations whose labels overlap with the validation window are removed from the training set.

In addition, an embargo period is imposed after each validation block. During this embargo interval, observations immediately following the validation sample are excluded from the training set in order to prevent any leakage of future information into the estimation procedure.

This purged and embargoed time-series cross-validation scheme ensures a strict temporal separation between training and validation samples and provides a reliable estimate of out-of-sample predictive performance.

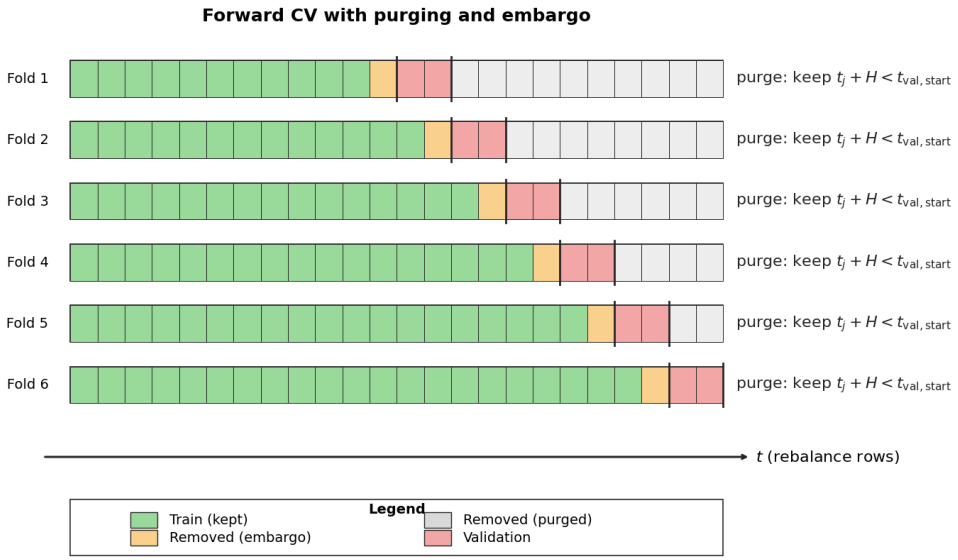


Figure 3.2: Illustration of the purged time-series cross-validation procedure used for hyperparameter selection.

3.6.5 Model interpretation with SHAP

To better understand the drivers of the predicted shrinkage intensity, SHAP (Shapley Additive Explanations) is used to analyse the feature importance of the supervised learning models. SHAP is a model interpretation framework based on cooperative game theory that attributes to each feature its marginal contribution to a prediction. The approach builds on the concept of Shapley values, which measure how much each variable contributes to the model output when all possible combinations of features are considered. In the context of machine learning models, SHAP therefore provides a consistent decomposition of a prediction into additive contributions from the input features. This property makes SHAP particularly suitable for interpreting complex models whose internal decision rules are not directly observable. The method was formalised by Lundberg and Lee (2017) and extended to tree-based models with an efficient algorithm known as TreeSHAP, which enables exact computation of Shapley values for ensemble methods such as Random Forests and gradient boosting models.

Beyond its theoretical foundation, SHAP has become a standard tool for interpreting supervised learning models because it provides both global and local explanations of model behaviour. Global feature importance can be assessed by aggregating SHAP values across observations, while the sign and magnitude of individual contributions indicate whether a feature increases or decreases the predicted outcome. Recent work highlights that Shapley-based explanations offer a consistent and theoretically grounded approach for interpreting predictive models and are therefore particularly well suited for empirical applications where transparency is important, for more theoretical background of Shapley Additive Explanations, refer to the appendix B.5.

3.7 Reinforcement learning formulation

The determination of the shrinkage intensity is formulated as a reinforcement learning (RL) problem in which an agent learns a policy that selects the shrinkage coefficient based on the current market state. The environment is modelled as a Markov Decision Process (MDP) characterized by states, actions and rewards. Although the problem is formulated in MDP terms, the transition probabilities of the environment are not assumed to be known in closed form. The empirical implementation is therefore model-free: the agent learns directly from realised transitions (s_t, a_t, u_t, s_{t+1}) generated by the return data. Even though the RL problem is introduced using the MDP framework, the financial environment is more appropriately viewed as only partially observable. The financial environment is therefore better interpreted as a Partially Observable Markov Decision Process (POMDP), in which the agent observes a finite window of past returns and constructs an approximate state representation for decision-making. In the PGA specification, this representation is based directly on the current return window, whereas in the RPGA it is enriched by a recurrent hidden state that summarises additional temporal information.

The observable input at time t is the $L \times N$ return matrix $R_{t-L+1:t}$ collected over the estimation window. In the PGA, this matrix constitutes the effective state representation used by the policy. In the RPGA, it serves as the observable input from which the recurrent network constructs a latent state representation.

The action consists of selecting a shrinkage intensity $\lambda_t \in [0, 1]$. Given this action, the corresponding covariance estimator is constructed and used to compute the portfolio allocation for the subsequent period.

The reward is defined using the realised portfolio risk obtained over the following forward window. In this setting, the learning objective is to select shrinkage intensities that lead to lower realised portfolio volatility. This perspective differs fundamentally from the supervised learning framework. In supervised learning, the objective is to predict the ex-post oracle shrinkage intensity computed from future realised volatility. In RL, by contrast, the agent does not attempt to predict this oracle coefficient directly. Instead, it learns a policy that selects shrinkage intensities so as to optimise realised portfolio performance through sequential feedback. RL therefore treats dynamic shrinkage as a control problem rather than a prediction problem.

The agent learns a policy

$$\pi_\theta(a_t \mid s_t),$$

parameterised by θ, which maps the observed state to a probability distribution over actions.

With $u_t = -g_t(\lambda_t)$, where $g_t(\lambda_t)$ denotes the realised portfolio volatility induced by the shrinkage

decision λ_t over the subsequent holding period, the policy objective is then

$$J(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} u_t\right]$$

where $\gamma \in (0, 1]$ denotes the discount factor. In the empirical implementation, $\gamma = 0.8$ for the PGA and $\gamma = 0.92$ for the RPGA. Maximising $J(\theta)$ is therefore equivalent to minimising discounted cumulative realised portfolio volatility.

This formulation allows the shrinkage intensity to adapt dynamically to the information contained in the recent return history, enabling the covariance estimator to respond to changing market conditions.

3.7.1 Policy gradient algorithm (PGA)

The first reinforcement learning specification is a policy gradient agent (PGA) in which the policy is parameterised by a multilayer perceptron (MLP). The network takes as input the state s_t, defined as the matrix of raw returns observed over the estimation window, and maps it to a stochastic policy over the shrinkage intensity.

Formally, the policy is written as

$$\pi_\theta(a_t \mid s_t),$$

where $a_t = \lambda_t \in [0, 1]$ denotes the selected shrinkage intensity and θ collects the parameters of the MLP. In the implementation, the network outputs the mean of a Gaussian policy, from which the action is sampled during training.

The parameters are estimated using a policy gradient method that seeks to maximise the expected cumulative reward $J(\theta)$.

Using the likelihood-ratio method of Williams (1992), the policy gradient is given by

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t\right],$$

where G_t is the discounted reward from time t onward.

The PGA therefore learns a direct mapping from the current return window to the shrinkage decision. Its feed-forward architecture allows for a flexible nonlinear policy, but it does not explicitly retain information beyond the current input window.

3.7.2 Recurrent Policy Gradient Algorithm (RPGA)

The second specification extends the previous approach by replacing the feed-forward MLP with a recurrent neural network based on a Gated Recurrent Unit (GRU). Consistent with the partially

observed nature of the environment discussed above, the RPGA uses a GRU to construct a latent state representation from the return sequence.

With $R_t \in \mathbb{R}^N$ denoting the vector of asset returns observed at rebalancing date t, the GRU processes the sequence of returns contained in the estimation window $(R_{t-L+1}, \ldots, R_t)$. To avoid confusion between the rebalancing date t and the position within the sequence, let $l = 1, \ldots, L$ index the observations inside the window. The recurrent update is then written as

$$h_l = \mathrm{GRU}(R_{t-L+l}, h_{l-1}), \quad l = 1, \ldots, L$$

where h_l summarises the relevant information extracted up to sequence position l. The final hidden representation h_L is then mapped to the mean of the stochastic policy over the shrinkage intensity.

As in the PGA case, the action corresponds to the shrinkage coefficient $\lambda_t \in [0, 1]$, and the policy is optimised using the same likelihood-ratio policy gradient method

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t\right].$$

The advantage of the RPGA is that the GRU can retain information from earlier observations within the return sequence, making the policy history-dependent. This is particularly relevant in financial markets, where volatility, correlations, and other latent conditions evolve over time. Compared with the PGA, the RPGA is therefore better suited to capturing persistent market dynamics in the determination of the shrinkage intensity.

A practical point concerns the choice of hyperparameters. Policy gradient methods are known to be sensitive to hyperparameter choices, as they rely on stochastic exploration and on reward signals that are noisy and delayed. This can generate high-variance gradient estimates and make training dynamics less stable. For this reason, hyperparameter selection is an important component of the empirical implementation. However, in order to preserve comparability across datasets and specifications, a single hyperparameter configuration is retained for each of the two RL architectures, PGA and RPGA, throughout the analysis.

Moreover, the detailed reinforcement learning architectures, including the MLP policy network, the GRU-based recurrent policy and the training logic are presented in Appendix B.6.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Oracle shrinkage intensity analysis

4.1.1 Distribution and time-series dynamics of oracle shrinkage intensity

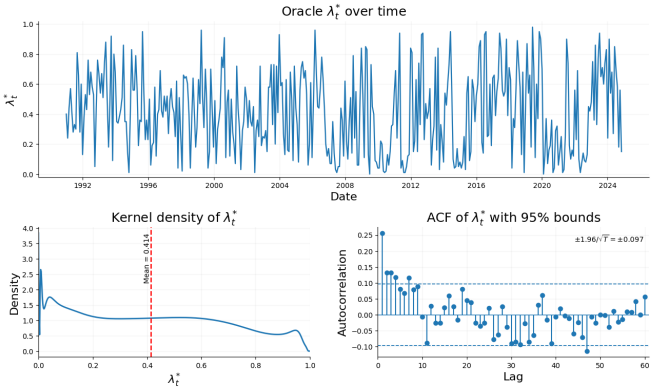


Figure 4.1: Oracle pre-analysis

Figure 4.1 presents a preliminary analysis of the ex-post oracle shrinkage coefficient that minimises realised GMV portfolio volatility over the holding period.

Several features emerge clearly. First, the time-series plot shows that the oracle shrinkage intensity varies substantially over time and spans almost the entire admissible interval $[0, 1]$. Rather than fluctuating around a stable level, the series repeatedly moves between values close to zero and values close to one. This provides strong evidence against the hypothesis that the optimal shrinkage intensity can be treated as a constant parameter and instead behaves as a state-dependent quantity.

Second, the kernel density estimate indicates that the distribution of λ_t^* is not concentrated around

a single value. Although the average shrinkage intensity is approximately 0.41, the distribution exhibits substantial mass near zero together with a broader interior mode corresponding to moderate shrinkage levels. The mean therefore masks significant heterogeneity across market conditions.

Finally, the autocorrelation function (ACF) reveals significant positive serial correlation in the oracle shrinkage series. Several lags remain statistically significant, indicating that the shrinkage intensity exhibits persistence over time. This temporal dependence suggests that past information about market conditions may contain predictive power for future shrinkage decisions, which provides a natural motivation for the data-driven learning approaches explored later in the master thesis.

4.1.2 Performance gain of time-varying shrinkage intensity

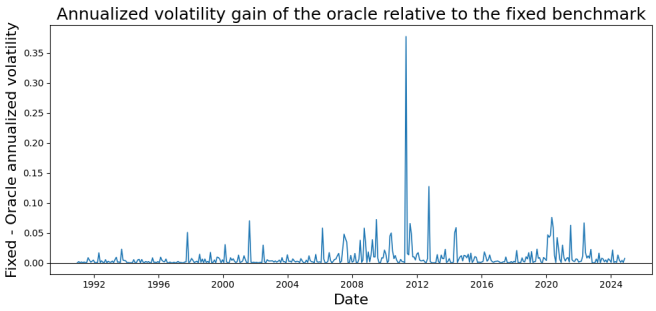


Figure 4.2: Objective gap between oracle and mean of the oracles

Figure 4.2 reports the annualised volatility gain of the oracle relative to the fixed benchmark, where the benchmark is defined by the mean oracle shrinkage intensity. Since the oracle coefficient is defined as the minimiser of realised forward-block volatility, the gap is non-negative by construction. The relevant information conveyed by the figure is therefore the magnitude of the gain, which measures the economic value of allowing the shrinkage intensity to vary over time.

The objective gap is positive during all of the sample, as expected given the construction of the oracle shrinkage intensity. The economically relevant result is therefore the time variation in the magnitude of the gap. During relatively stable periods, the gains remain modest, suggesting that a constant coefficient may approximate the oracle reasonably well. In contrast, pronounced spikes occur during periods of market stress, most notably around the global financial crisis, highlighting the importance of adaptive shrinkage in turbulent market environments.

4.1.3 Regime dependence of shrinkage intensity

To further investigate the determinants of the oracle shrinkage coefficient, its distribution is examined across volatility, correlation and drawdown regimes.

4.1.3.1 Volatility regimes

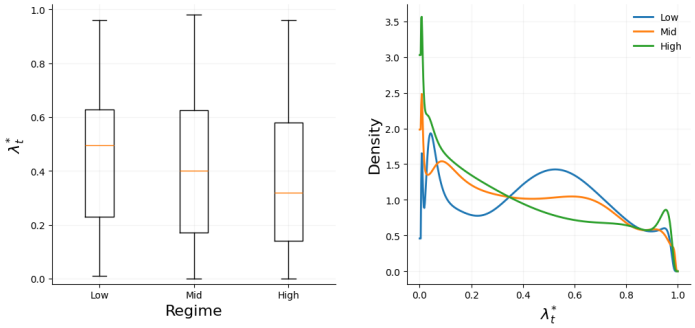


Figure 4.3: Volatility regime of Oracle shrinkage intensity

Figure 4.3 shows the distribution of λ_t^* across volatility regimes. The median shrinkage coefficient is higher during low-volatility periods and lower during high-volatility periods. Although this may appear counterintuitive, it can be explained by the structure of the shrinkage target, which in this study is a scaled identity matrix imposing equal variances and zero correlations.

When volatility is low, the empirical covariance matrix tends to be noisier and less informative, making stronger shrinkage toward the identity target more beneficial. In contrast, during high-volatility periods the empirical covariance structure reflects stronger common risk factors, reducing the need for aggressive regularisation.

4.1.3.2 Correlation regimes

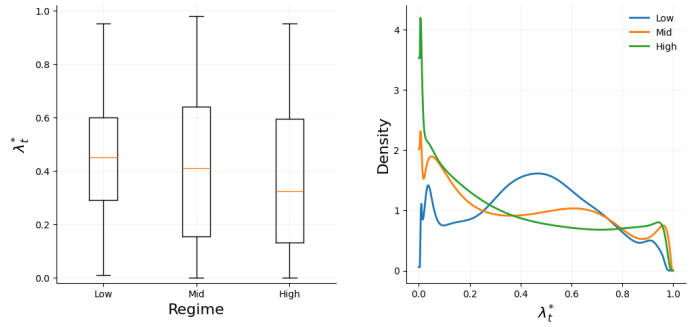


Figure 4.4: Correlation regime of Oracle shrinkage intensity

Figure 4.4 reports the distribution of oracle shrinkage intensity across regimes defined by average cross-sectional correlation. Higher correlation regimes are associated with lower shrinkage intensities, while lower correlation regimes correspond to higher shrinkage levels. Since the identity target implicitly assumes weak correlations, strong shrinkage toward this target becomes less appropriate when market correlations are elevated.

4.1.3.3 *Drawdown regimes*

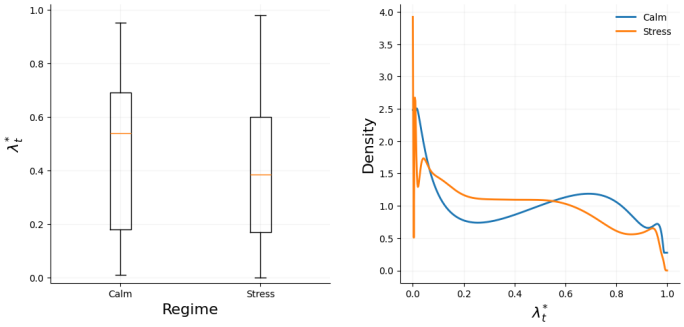


Figure 4.5: Drawdown regime of Oracle shrinkage intensity

Figure 4.5 examines the behaviour of the oracle shrinkage coefficient across market stress regimes defined by equity market drawdowns. Periods of market stress tend to be associated with lower shrinkage intensities, whereas tranquil periods correspond to higher shrinkage levels. During drawdowns, asset returns tend to exhibit stronger co-movement, as correlations typically rise in bear markets and under large negative returns (Longin and Solnik, 2001; Ang and Bekaert, 2002). This increase in cross-sectional dependence may, in turn, make the empirical covariance matrix relatively more informative for risk estimation in stresses environments.

Overall, these regime analyses indicate that the oracle shrinkage coefficient behaves as a state-dependent parameter, increasing when the covariance structure weakens and decreasing when market-wide risk factors dominate.

4.2 Predictability of the oracle shrinkage intensity

4.2.1 *Time-series properties of the oracle shrinkage intensity*

The oracle shrinkage intensity exhibits both temporal persistence and systematic relationships with observable market characteristics, suggesting that it may be predictable using information available at time t.

First, the autocorrelation function (ACF) reported in Figure 4.1 shows significant positive serial correlation across several lags. This indicates that the shrinkage coefficient evolves gradually over time rather than behaving as a purely random sequence, implying that past values may contain information about future shrinkage decisions.

Second, Figure 4.6 reports the correlations between the oracle shrinkage intensity λ_t^* and the set of features constructed from the rolling estimation window. Several features exhibit non-negligible correlations with the oracle coefficient. In particular, spectral properties of the covariance matrix, such as the condition number proxy, eigenvalue dispersion and eigenvalue concentration, display the strongest negative correlations with λ_t^*. Correlation measures, including mean correlation

and mean absolute correlation, also show systematic negative relationships with the oracle shrinkage intensity. These patterns are consistent with the structure of the scaled identity matrix, which assumes weak cross-sectional correlations.

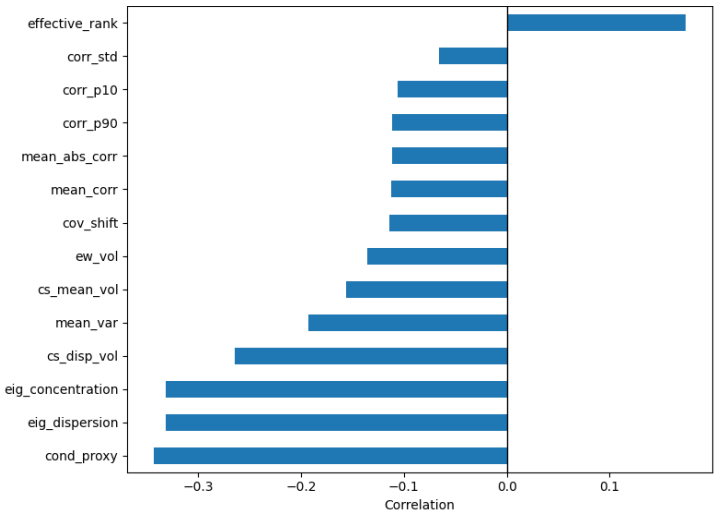


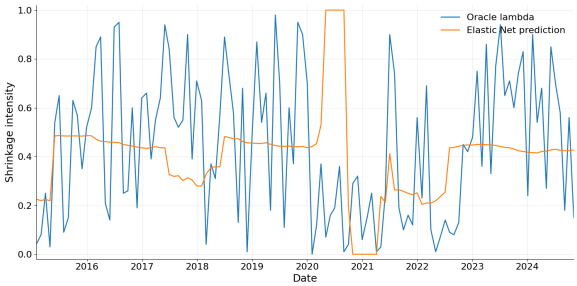
Figure 4.6: Feature-target correlations

Overall, the results indicate that the oracle shrinkage coefficient behaves as a state-dependent parameter that depends on observable characteristics of the return distribution. This provides motivation for the learning approaches implemented in the following sections, which aim to approximate the mapping between market information and shrinkage decisions.

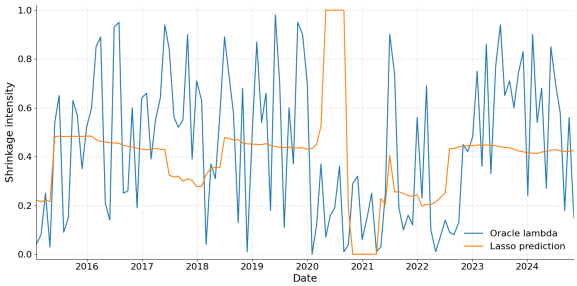
4.2.2 Predictive performance of machine learning models

Figure 4.7 reports the out-of-sample evolution of the oracle shrinkage intensity together with the predictions generated by the different machine learning approaches. A first striking feature of the oracle coefficient is its pronounced time variation and high-frequency volatility. The oracle series fluctuates sharply over short horizons, frequently moving between very low and very high shrinkage regimes. This behaviour reflects the strong sensitivity of the optimal shrinkage parameter to changes in the underlying covariance structure of asset returns. In practice, however, such rapid fluctuations are inherently difficult to predict using observable state variables.

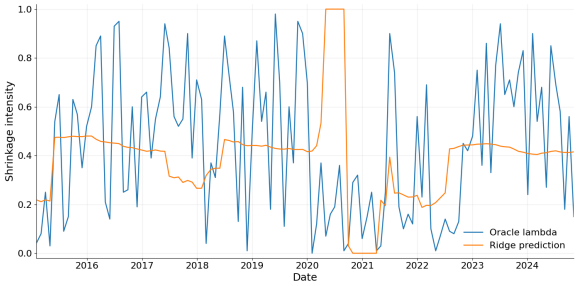
Across all supervised learning models, a common pattern emerges: predicted shrinkage intensities are substantially smoother than the oracle benchmark. Linear models, including Lasso, Ridge and Elastic Net, exhibit particularly strong smoothing behaviour. Their predictions remain confined to a relatively narrow band around intermediate shrinkage levels, indicating that these methods primarily capture the unconditional mean of the oracle coefficient rather than its short-term fluctuations. This outcome is consistent with the regularisation structure of these estimators, which shrinks the influence of individual predictors and therefore favors stable, low-variance estimates.



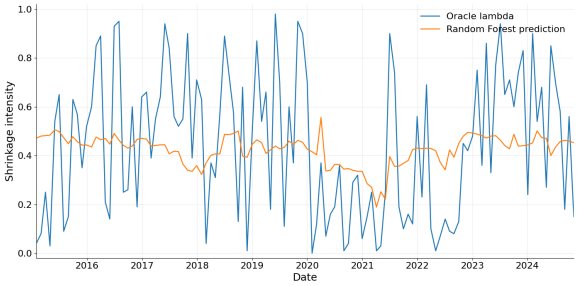
(a) Elastic Net



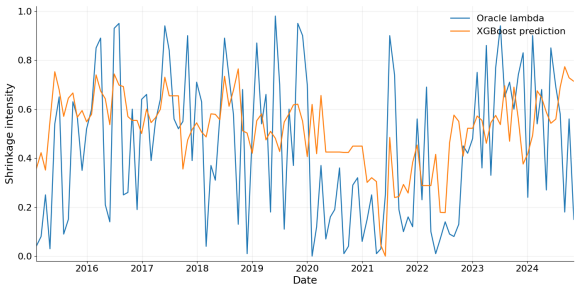
(b) Lasso



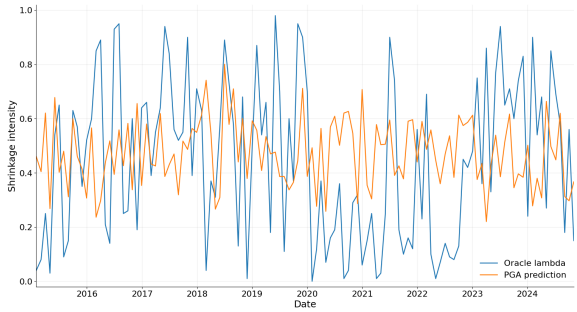
(c) Ridge



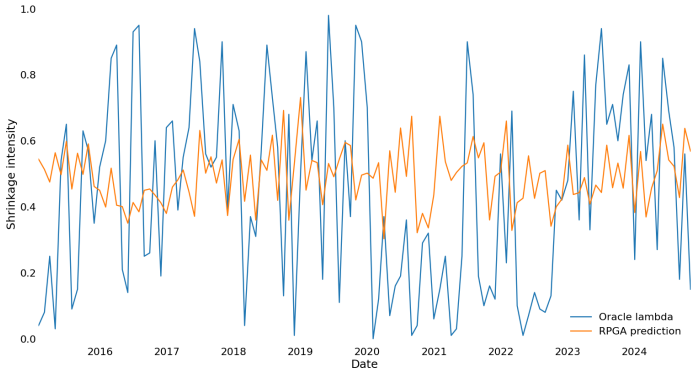
(d) Random Forest



(e) XGBoost



(f) PGA



(g) RPGA

Figure 4.7: Out-of-sample predictions of the oracle shrinkage intensity for each model

Tree-based ensemble methods display somewhat greater flexibility. In particular, the Random Forest estimator produces modest variation around the average shrinkage level, reflecting its

ability to capture nonlinear interactions among predictors. Nevertheless, the predicted trajectory remains clearly smoother than the oracle series. Among the supervised models, XGBoost exhibits the strongest responsiveness to changes in the oracle coefficient. The gradient boosting framework allows the model to approximate more complex nonlinear relationships, which results in predictions that occasionally track shifts in the oracle intensity more closely. Despite this improvement, the predicted series still fails to reproduce the extreme swings observed in the oracle benchmark, suggesting that a fraction of the oracle variability is driven by components that are either weakly predictable or dominated by estimation noise.

The reinforcement learning-based approaches display a different but equally informative pattern. Both the Policy Gradient Algorithm (PGA) and the Recurrent Policy Gradient Algorithm (RPGA) generate shrinkage policies that evolve gradually over time and remain concentrated within a moderate range. Rather than attempting to match the oracle series point by point, these methods learn policies that prioritise stability of the shrinkage parameter. The recurrent specification embedded in the RPGA introduces additional flexibility by allowing the policy to depend on past states. However, the resulting predictions still exhibit limited volatility relative to the oracle coefficient. This behaviour reflects the fundamental nature of the reinforcement learning objective: the algorithms optimise a long-run portfolio criterion rather than directly minimising prediction error with respect to the oracle shrinkage intensity.

Taken together, these results reveal a clear empirical regularity. Although the oracle shrinkage coefficient exhibits substantial short-term variability, the machine learning models generate smoother estimates that primarily capture the lower-frequency component of its dynamics. This pattern suggests that a large share of the high-frequency fluctuations in the oracle coefficient is difficult to predict using the available information set and likely reflects sampling variability in covariance estimation. Importantly, the oracle shrinkage intensity is computed using forward returns and therefore relies on information that is not observable at the time the prediction is made. As such, it represents an ex-post benchmark rather than a feasible prediction target, making it unrealistic to expect predictive models to match it closely on a point-by-point basis. Instead, the models learn shrinkage policies that remain dynamic but evolve more gradually over time. This smoother adjustment reflects the fact that part of the oracle's variability is driven by noise induced by forward-looking information, which cannot be anticipated using observable predictors.

Additionally to the figure 4.7, the Pearson correlations are reported in table 4.1, this helps to evaluate the extent to which each model reproduces the out-of-sample time variation of the oracle shrinkage intensity. The highest correlations are obtained by the tree-based supervised models, with XGBoost reaching 0.3697 and Random Forest 0.3387. This indicates that these models capture the dynamics of the oracle shrinkage intensity more effectively than the other approaches. The linear models display much weaker positive correlations, around 0.12, suggesting that they only partially reproduce the oracle trajectory.

By contrast, the reinforcement learning models exhibit correlations close to zero, with −0.0520 for PGA and 0.0250 for RPGA. This does not necessarily imply poor portfolio performance, since these models are not trained to mimic the oracle shrinkage intensity directly. Instead, they optimize a portfolio-based reward function. Therefore, the reinforcement learning agents may learn smoother and more stable shrinkage policies that differ from the ex-post oracle path while still leading to competitive out-of-sample portfolio results. Overall, these correlations show that approximating the oracle lambda and maximizing portfolio performance are related but distinct objectives.

Table 4.1: Pearson correlation per model

Model	**Pearson correlation**
XGBoost	0.3697
Random Forest	0.3387
Ridge	0.1273
Lasso	0.1238
Elastic Net	0.1236
PGA	-0.0520
RPGA	0.0250

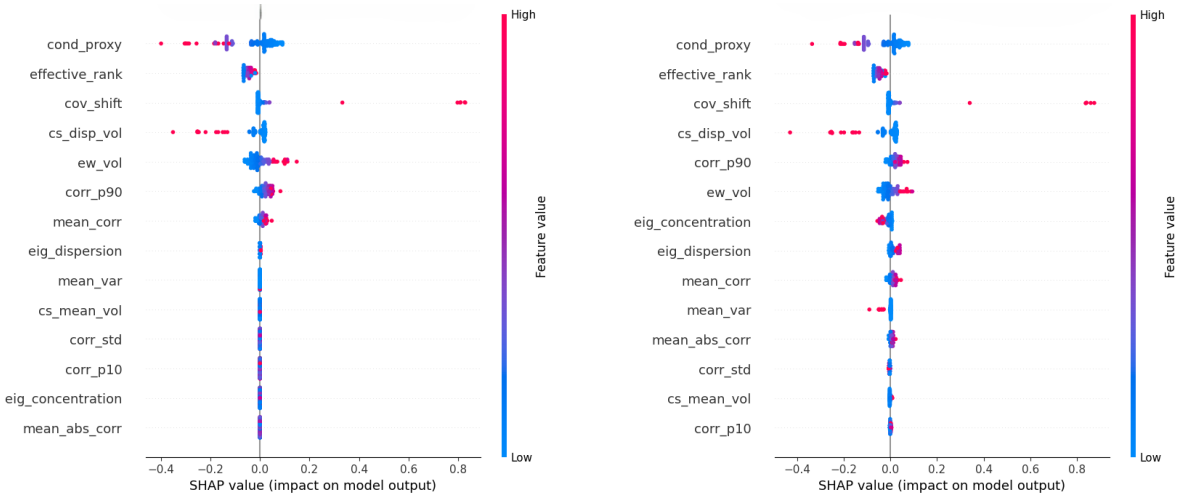
Finally, the model-specific optimisation histories and hyperparameter diagnostics are reported in appendix C.

4.2.3 SHAP analysis of the supervised learning models

Figure 4.8 presents the SHAP summary plots for the five supervised learning models considered in this thesis: Lasso, Ridge, Elastic Net, Random Forest and XGBoost. In each figure, the horizontal axis represents the SHAP values, which measures the impact of a feature on the predicted shrinkage intensity. Positive values increase the predicted shrinkage coefficient, whereas negative values decrease it. The color scale represents the feature value, with red corresponding to high values and blue to low values.

A first important result is that the models consistently rely on a similar set of economically meaningful predictors. Across specifications, the most influential variables include the conditioning proxy of the covariance matrix (`cond_proxy`), cross-sectional volatility dispersion (`cs_disp_vol`), average variance (`mean_var`), covariance instability (`cov_shift`) and several measures of correlation structure such as `corr_p90`, `mean_corr` and `corr_std`. The prominence of these variables indicates that the models primarily react to changes in the conditioning and stability of the covariance matrix as well as to shifts in the overall correlation environment. This is economically intuitive: shrinkage becomes more valuable when the covariance matrix is poorly conditioned, when correlations increase, or when the dispersion of asset risk becomes more pronounced.

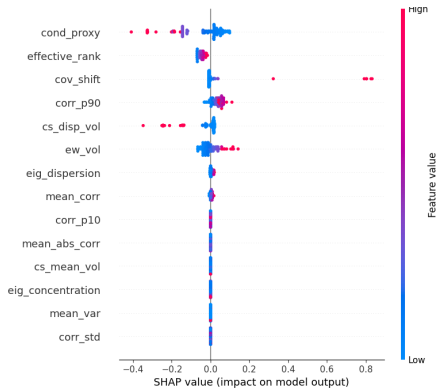
At the same time, the SHAP summaries reveal meaningful differences between model classes. The penalised linear models, Lasso, Ridge, and Elastic Net, display a relatively concentrated importance structure. In these models, predictive importance tends to be concentrated on a limited subset of features, while the remaining variables contribute only marginally to the predictions. This reflects the regularisation properties of these estimators, which shrink the influence of less informative predictors and, in the case of Lasso and Elastic Net, may also induce partial sparsity in the linear representation of the predictive signal. In particular, the three models assign most of the importance to the conditioning proxy of the covariance matrix `cond_proxy`, covariance instability `cov_shift`, cross-sectional volatility dispersion `cs_disp_vol`, and correlation-related statistics. These variables therefore capture the main linear component of predictability in the shrinkage coefficient.



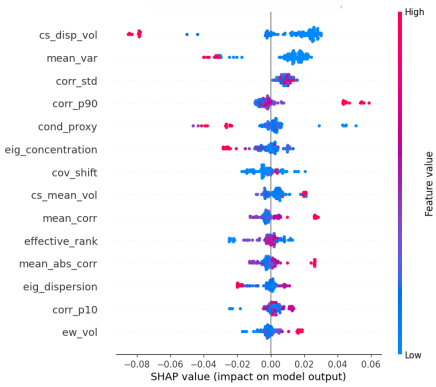
(a) Lasso (b) Ridge

Figure 4.8: SHAP summary plots for the five supervised learning models

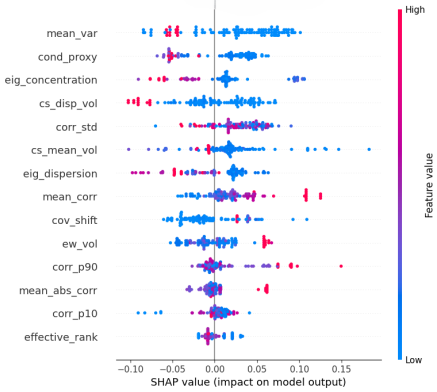
In contrast, the tree-based models, Random Forest and XGBoost, exhibit a richer importance structure and assign non-negligible contributions to a broader set of features. The Random Forest highlights the role of volatility dispersion (`cs_disp_vol`), average variance (`mean_var`), and correlation dispersion (`corr_std`), while XGBoost places strong emphasis on `mean_var`, `cond_proxy`, and eigenvalue concentration measures (`eig_concentration`). These models also reveal stronger nonlinear patterns, with some variables affecting predictions only when they reach extreme values. For instance, large values of covariance instability `cov_shift` generate substantial positive SHAP contributions in several observations, suggesting that abrupt changes in the covariance structure trigger stronger predicted shrinkage adjustments.



(c) Elastic Net



(d) Random Forest



(e) XGBoost

Figure 4.8: SHAP summary plots for the five supervised learning models (continued)

However, compared to the penalised linear models, the magnitude of SHAP values in the ensemble tree models is generally smaller. This indicates that while tree-based methods distribute importance across a wider set of predictors, each individual feature tends to have a more moderate marginal impact on the predicted shrinkage intensity. In contrast, the penalised models concentrate larger SHAP contributions on a few key variables, reflecting a more dominant linear effect of selected predictors.

Overall, the SHAP analysis confirms that the supervised learning models rely on economically interpretable signals rather than opaque statistical patterns. Periods characterized by unstable covariance structures, higher correlations, or more heterogeneous volatility across assets tend to be associated with higher predicted shrinkage intensities. Moreover, the comparison across models indicates that while penalised linear estimators capture the main linear component of predictability, tree-based models exploit additional nonlinearities and interaction effects. This evidence supports the view that the time variation in the oracle shrinkage intensity reflects systematic changes in the covariance estimation environment rather than purely random fluctuations.

4.3 Out-of-sample portfolio performance

Table 4.2: Portfolio Performance Metrics - Estimation window of 252 trading days

Method	Annualised return	Annualised volatility	Annualised Sharpe	Average turnover
EW	0.1140	0.1925	0.5697	0.0563
GMV (Sample Cov)	0.0535	0.2777	0.2744	10.2657
GMV (Ledoit-Wolf)	0.1289	0.1590	0.7352	1.8926
GMV (LW Analytical NL)	0.1836	0.2453	0.7402	0.1300
GMV (Lasso)	0.0409	0.1500	0.2301	1.5510
GMV (Ridge)	0.0401	0.1501	0.2248	1.5710
GMV (Elastic Net)	0.0407	0.1501	0.2289	1.5450
GMV (XGBoost)	0.0408	0.1386	0.2368	0.9310
GMV (RF)	0.0586	0.1315	0.3707	0.6847
GMV (PGA)	0.0619	0.1335	0.3909	0.9051
GMV (RPGA)	0.0818	0.1324	0.5325	0.7714

Table 4.2 reveals several interesting patterns. First, the global minimum variance (GMV) portfolio constructed using the sample covariance matrix exhibits the highest volatility among the covariance-based strategies, reaching 27.77% annually. This result illustrates the well-documented instability of the sample covariance estimator in high-dimensional settings, where estimation error propagates directly into the portfolio weights and leads to poor risk control out of sample. The combination of high volatility and extremely elevated turnover further reflects the sensitivity of the sample-based GMV portfolio to small fluctuations in estimated covariances.

Introducing shrinkage improves volatility control. The linear Ledoit and Wolf (2004) estimator reduces annualised volatility to approximately 15.9%, confirming the effectiveness of shrinkage techniques in stabilising covariance estimates and mitigating estimation error. A substantially higher volatility level of 24.53% is observed for the analytical nonlinear shrinkage estimator of Ledoit and Wolf (2020). This result indicates weaker performance in the present portfolio construction exercise relative to the other covariance estimators under consideration. Importantly, these shrinkage approaches also reduce portfolio turnover relative to the sample-based estimator, indicating more stable portfolio weights over time.

When the shrinkage intensity is predicted using linear machine learning models, the resulting portfolios achieve volatility levels close to those obtained with the linear Ledoit and Wolf (2004) estimator. In particular, Elastic Net, Lasso and Ridge models produce annualised volatilities around 15%, suggesting that these approaches generate reasonably stable covariance estimates despite their limited ability to fully capture time variation in the optimal shrinkage coefficient. In addition to volatility improvements, these supervised learning methods also exhibit a clear tendency to reduce turnover, reflecting smoother adjustments in portfolio allocations compared to purely sample-driven strategies.

More flexible models provide further improvements. The tree-based ensemble methods, Random Forest and XGBoost, reduce portfolio volatility to 13.15% and 13.86%, respectively. These results suggest that nonlinear machine learning models are better able to extract predictive information from the state variables that is relevant for determining the shrinkage intensity, thereby improving the conditioning of the covariance matrix and the resulting portfolio allocation. At the same time, these models contribute to lower turnover, as their more stable and structured predictions lead to less frequent and less extreme rebalancing.

The reinforcement learning approaches deliver the lowest volatility levels among all predictive methods. The Policy Gradient Algorithm (PGA) achieves an annualised volatility of 13.35%, while the Recurrent Policy Gradient Algorithm (RPGA) further improves upon this result with a volatility of 13.24%. The superior performance of RPGA likely reflects the ability of the recurrent architecture to incorporate temporal dependencies in the state representations, allowing the shrinkage policy to adapt more effectively to evolving market conditions. Crucially, both RL-based approaches also display comparatively lower turnover than most competing methods, highlighting their ability to produce smoother, more persistent allocation decisions.

It is also worth noting that the Sharpe ratio remains lower than of the Ledoit-Wolf benchmark. This is not inconsistent with the design of the proposed methods, since the supervised learning and reinforcement learning approaches are not trained to maximise risk-adjusted return, but rather to improve shrinkage decisions from a risk-minimisation perspective. At the same time, this result suggests that future research could extend the framework by incorporating an explicit penalty related to expected return, or by targeting a risk-adjusted objective directly.

Overall, the results highlight the importance of dynamically determining the shrinkage intensity when constructing minimum variance portfolios. While traditional shrinkage estimators already provide substantial improvements over the sample covariance matrix, machine learning approaches, particularly reinforcement learning methods, further enhance out-of-sample risk control. These gains are not limited to volatility reduction but are also accompanied by a consistent decrease in portfolio turnover, which is essential from an implementation perspective due to its direct impact on transaction costs.

4.4 Economic interpretation

The empirical results have several important economic implications for portfolio construction in high-dimensional environments. In this setting, covariance estimation is not an end in itself but an intermediate input into portfolio construction. The economic value of a shrinkage rule therefore lies in its ability to improve realised portfolio risk, weight stability and implementability out of sample. In the minimum variance framework, improvements in realised volatility directly translate into more efficient risk management. Even relatively small reductions in portfolio volatility can generate economically meaningful gains for investors, particularly when portfolios

are implemented at large scale or under strict risk constraints. Consequently, methods that improve the stability of covariance estimation can provide substantial practical value.

A central economic challenge in minimum variance portfolio allocation is the presence of estimation error in covariance matrices. When the number of assets is large relative to the available sample size, small errors in estimated covariances can lead to large distortions in portfolio weights. This phenomenon is well documented in the portfolio optimisation literature and explains why naive estimators often perform poorly out of sample. Shrinkage techniques address this problem by trading a small amount of bias for a reduction in estimation variance, thereby producing more stable covariance estimates and more reliable portfolio allocations.

Allowing the shrinkage intensity to vary over time further enhances this mechanism. Market conditions such as volatility, correlations, and cross-sectional dispersion evolve through time, which implies that the degree of shrinkage required to control estimation error may also vary. Data-driven approaches that adapt the shrinkage parameter dynamically can therefore better balance the bias-variance trade-off embedded in covariance estimation. From an economic perspective, this flexibility allows the portfolio construction process to respond more effectively to changing market environments.

The reinforcement learning framework provides an additional advantage by learning shrinkage policies that are directly aligned with the investor's objective. Rather than attempting to replicate the oracle shrinkage coefficient, the policy gradient algorithms determine the shrinkage intensity that minimises realised portfolio risk. This objective-driven learning mechanism allows the algorithm to internalise the relationship between shrinkage, covariance stability, and portfolio volatility. As a result, the learned policies may generate economically superior allocations even when the oracle shrinkage coefficient itself is difficult to predict.

Finally, the improved stability of the resulting portfolios has important implications for real-world implementation. More stable covariance estimates tend to produce smoother portfolio weights, which reduces unnecessary trading and improves the implementability of the strategy in the presence of transaction costs. In this sense, dynamically learned shrinkage policies contribute not only to improved risk control but also to more practical and robust portfolio management.

Overall, the evidence points to the economic value of learning the shrinkage intensity in a data-driven manner. By allowing the degree of shrinkage to adjust dynamically to evolving market conditions, machine learning approaches improve the reliability of portfolio risk estimation and lead to more effective out-of-sample portfolio construction in high-dimensional settings. More importantly, the reinforcement learning framework introduced in this thesis departs from the traditional paradigm of predicting an ex-post oracle coefficient and instead learns shrinkage policies that directly optimise the portfolio objective. The results therefore provide empirical evidence that policy-based learning can constitute a meaningful improvement over conventional shrinkage methods when the objective is not only to regularise covariance estimates, but ultimately to

deliver lower realised risk and smoother portfolio implementation.

4.5 Robustness checks

In order to ensure that the main findings are not driven by specific modelling choices, data characteristics, or evaluation procedures, a series of robustness checks are conducted. First, using the S&P 500 dataset, the different strategies of supervised learning and reinforcement learning are assessed using an estimation window of 126 trading days (approximately 6 months) and an alternative window of 504 trading days (approximately 2 years). Secondly, the merged dataset detailed in Section 3.1 is used to compute the different portfolio allocations.

Table 4.3: Portfolio Performance Metrics on the S&P 500 – Estimation window of 126 trading days

Method	Annualised return	Annualised volatility	Annualised Sharpe	Average turnover
EW	0.1140	0.1925	0.5697	0.0563
GMV (Sample Cov)	0.0836	0.1694	0.4604	4.4437
GMV (Ledoit-Wolf)	0.0847	0.1521	0.5004	2.2112
GMV (LW Analytical NL)	0.1129	0.1931	0.5629	0.0671
GMV (Lasso)	0.0457	0.1608	0.2569	1.0932
GMV (Ridge)	0.0463	0.1609	0.2604	1.1339
GMV (Elastic Net)	0.0413	0.1571	0.2323	1.3618
GMV (XGBoost)	0.0456	0.1469	0.2643	1.1410
GMV (RF)	0.0455	0.1448	0.2656	1.0777
GMV (PGA)	0.0318	0.1609	0.1741	0.7805
GMV (RPGA)	0.0533	0.1448	0.3162	1.1167

Table 4.4: Portfolio Performance Metrics on the S&P 500 – Estimation window of 504 trading days

Method	Annualised return	Annualised volatility	Annualised Sharpe	Average turnover
EW	0.1140	0.1925	0.5697	0.0563
GMV (Sample Cov)	0.1374	0.2005	0.6529	3.5885
GMV (Ledoit-Wolf)	0.0988	0.1731	0.5310	0.9032
GMV (LW Analytical NL)	0.1128	0.1878	0.5733	0.0662
GMV (Lasso)	0.0977	0.1631	0.5464	1.2638
GMV (Ridge)	0.1011	0.1642	0.5628	1.2860
GMV (Elastic Net)	0.0977	0.1631	0.5465	1.2638
GMV (XGBoost)	0.0601	0.1328	0.3792	0.4469
GMV (RF)	0.0536	0.1301	0.3371	0.4556
GMV (PGA)	0.0554	0.1399	0.3347	0.6034
GMV (RPGA)	0.0708	0.1387	0.4410	0.5338

The results for the S&P 500 dataset confirm the relative performance of the different strategies to the length of the estimation window. For the shorter estimation window of 126 trading days, the linear Ledoit-Wolf benchmark achieves an annualised volatility of 15.21%, which already represents an important improvement over the sample covariance portfolio (16.94%) and remains close to the best-performing data-driven methods. Among the supervised learning models, the linear models produce mixed results: Lasso, Ridge and Elastic Net yield annualised volatilities of 16.08%, 16.09% and 15.71%, respectively, all above the linear Ledoit-Wolf benchmark. By contrast, the nonlinear supervised models perform better, with XGBoost reaching 14.69% and Random Forest 14.48%. The reinforcement learning strategies, however, do not systematically outperform the benchmark in this short-sample setting. PGA records an annualised volatility of 16.09% which is above the linear Ledoit-Wolf level, while RPGA achieves 14.48%, matching the performance of Random Forest but not providing a clear additional gain.

For the longer estimation window of 504 trading days, the results display a clearly different pattern. The linear Ledoit-Wolf benchmark achieves an annualised volatility of 17.31%, which is now clearly dominated by all learning-based approaches. Linear supervised models already improve upon this benchmark, with Lasso and Elastic Net reaching 16.31% and Ridge 16.42%. The improvement is even more pronounced for nonlinear supervised models, as XGBoost and Random Forest reduce volatility to 13.28% and 13.01%. Reinforcement learning strategies exhibit a similar performance with PGA achieving an annualised volatility of 13.99% and RPGA 13.87%.

Compared to the 126-day window, where volatility differences were limited and only nonlinear methods provided moderate gains, the 504-day window allows both linear and nonlinear learning-based approaches to consistently outperform the Ledoit-Wolf benchmark. This shift can be attributed to the reduction in estimation error when the sample size increases. With 504 observations, the covariance matrix is estimated more precisely, enabling both linear models to improve shrinkage calibration and nonlinear models, particularly ensemble methods and reinforcement learning, to exploit more complex structures in the data. In contrast, when the estimation window is limited to 126 days, the higher level of noise reduces the effectiveness of flexible models, thereby preserving the competitiveness of the Ledoit-Wolf estimator.

Then robustness analysis on the merged dataset of 294 portfolios confirms the stability of the results across different data environments. The linear Ledoit-Wolf estimator achieves an annualised volatility of 13.1%, significantly improving on the sample covariance matrix (17.04%). Linear supervised models provide only marginal gains, with Lasso and Elastic Net reducing volatility to 12.71%, while Ridge remains close to the benchmark (13.08%).

Table 4.5: Portfolio Performance Metrics on the merged dataset of 294 portfolios – Estimation window of 252 trading days

Method	Annualised return	Annualised volatility	Annualised Sharpe	Average turnover
EW	0.1124	0.2100	0.5323	0.0244
GMV (Sample Cov)	0.0755	0.1704	0.4128	11.6033
GMV (Ledoit-Wolf)	0.1025	0.1310	0.6816	6.8259
GMV (LW Analytical NL)	0.1095	0.2146	0.5134	0.0891
GMV (Lasso)	0.0963	0.1271	0.6541	1.9990
GMV (Ridge)	0.0921	0.1308	0.6100	3.7182
GMV (Elastic Net)	0.0963	0.1271	0.6541	1.9995
GMV (XGBoost)	0.0945	0.1272	0.6409	1.8339
GMV (RF)	0.0976	0.1261	0.6409	2.0899
GMV (PGA)	0.0867	0.1320	0.5683	2.2107
GMV (RPGA)	0.0925	0.1319	0.6093	1.1543

Nonlinear supervised methods deliver the strongest performance, with XGBoost reaching 12.72% and Random Forest achieving the lowest volatility at 12.61%. In contrast, reinforcement learning strategies do not outperform the benchmark, with PGA and RPGA yielding volatilities of 13.2% and 13.19% respectively.

Overall, the robustness checks confirm that the main findings are stable across estimation windows and datasets. While the linear Ledoit-Wolf estimator remains highly competitive in short-sample settings, its performance deteriorates relative to learning-based approaches when more data is available. In particular, nonlinear supervised models consistently deliver the lowest volatility for longer estimation windows, while reinforcement learning strategies exhibit competitive but less systematic improvements. On the merged portfolio dataset, performance differences become more moderate, reflecting the lower level of idiosyncratic noise.

As noted in the methodological section, the RL results should be interpreted conditional on a fixed hyperparamter configuration for each architecture.

These results indicate that the benefits of data-driven shrinkage methods are robust but depend on the sample size, the underlying data structure, and, in the case of reinforcement learning, the specification of model hyperparameters.

CHAPTER 5

CONCLUSION

This master thesis investigated whether covariance shrinkage for GMV portfolio construction should be treated as a static statistical correction or as a dynamic decision that adapts to changing market conditions. More specifically, it examined whether supervised learning and reinforcement learning can learn time-varying shrinkage policies that improve the out-of-sample performance of minimum-variance portfolios. This question arises from a clear gap in the literature: although classical shrinkage methods reduce estimation error, they generally rely on a fixed shrinkage intensity, whereas portfolio construction takes place in an environment characterised by changing volatility, evolving correlations and structural instability.

5.1 Synthesis of the findings

The results show, first, that the optimal shrinkage intensity is not constant through time. The oracle analysis indicates that the ex-post shrinkage coefficient varies considerably, exhibits persistence and behaves differently across volatility, correlation and drawdown regimes. This supports the interpretation of shrinkage intensity as a state-dependent parameter that changes with the reliability of the sample covariance matrix and with the broader covariance environment.

Second, the results show that this time variation is partly predictable. The oracle shrinkage coefficient is systematically related to observable features of the estimation window, especially variables linked to covariance instability, matrix conditioning, eigenvalue dispersion and correlation structure. The SHAP analysis confirms that the supervised learning models rely on economically interpretable drivers rather than arbitrary statistical patterns.

Third, the portfolio results show that learned shrinkage policies can improve out-of-sample minimum-variance portfolio construction. In the baseline setting, all learning-based approaches improve on the sample covariance benchmark, with the strongest gains coming from nonlinear

models, especially the recurrent reinforcement learning specification. However, the robustness checks show that these improvements are not unconditional. When the estimation window is short, the linear Ledoit-Wolf benchmark remains highly competitive, while longer estimation windows make learning-based methods more effective. On the merged portfolio dataset, nonlinear supervised models appear more robust than reinforcement learning.

Overall, the master thesis answers the research question positively, but not unconditionally: supervised learning and reinforcement learning can learn time-varying shrinkage policies that improve out-of-sample performance, but the strength and consistency of the gains depend on the empirical setting.

5.2 Contribution and significance

This master thesis contributes to the literature in three main ways. First, it shows empirically that covariance shrinkage should be modelled dynamically rather than statically. This shifts the role of shrinkage intensity from a fixed tuning parameter to a state-dependent object linked to market conditions.

Second, it provides a unified comparison between two approaches to dynamic shrinkage: supervised learning, which treats shrinkage as a prediction problem, and reinforcement learning, which treats it as a sequential control problem. This comparison is central to the originality of the thesis because these two perspectives are often discussed separately rather than within a common empirical framework.

Third, the thesis demonstrates the economic relevance of this framework in the context of GMV portfolios, where improved covariance estimation translates directly into more stable risk control and smoother portfolio implementation. More broadly, the results suggest that data-driven shrinkage policies can improve the stability of covariance estimation and contribute to more robust minimum-variance portfolio construction when market conditions change.

5.3 Limitations and recommendations for future research

Despite these contributions, the thesis has several limitations. First, the results depend on the structure of the dataset. Performance differs between the S&P 500 stock panel and the merged portfolio dataset, suggesting that the relative strength of each method depends on idiosyncratic noise, dimensionality and the informational structure of the data.

Second, the results depend on empirical design choices, especially the estimation window and rolling-window construction. Model rankings change across shorter and longer windows, which means that the conclusions are partly shaped by the amount of information available for covariance estimation at each decision date.

Third, the results also depend on methodological choices such as time-series cross-validation,

purging and embargo procedures, and hyperparameter optimisation. This is especially relevant for reinforcement learning, since RL performance is sensitive to hyperparameter specification, even though one fixed set of hyperparameters is maintained for comparability across settings.

Finally, the analysis focuses on GMV portfolios and on a scaled identity shrinkage target. This is appropriate for isolating covariance-estimation quality, but it leaves open whether the same conclusions would hold for richer shrinkage targets, constrained portfolio problems or objectives that incorporate expected returns.

Future research could therefore test dynamic learning-based shrinkage with alternative targets, such as factor-based or constant-correlation estimators. Another extension would be to study more systematically the sensitivity of the results to cross-validation design, hyperparameter tuning and rolling-window specifications. A further avenue would be to combine supervised learning and reinforcement learning in hybrid frameworks, for example by using supervised learning to provide stable initial shrinkage estimates and reinforcement learning to refine them through direct portfolio feedback. Finally, future work could extend the analysis beyond GMV portfolios to settings with portfolio constraints, transaction cost penalties and broader investor objectives.

The final takeaway is that the value of covariance shrinkage lies not only in regularising noisy covariance matrices, but also in its potential to adapt to changing market conditions. This perspective shifts shrinkage from a static estimation device to a dynamic component of portfolio risk management.

BIBLIOGRAPHY

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631.

Ang, A., & Bekaert, G. (2002). International asset allocation with regime shifts. *The Review of Financial Studies*, *15*(4), 1137–1187.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Best, M. J., & Grauer, R. R. (1991). On the sensitivity of mean-variance-efficient portfolios to changes in asset means: Some analytical and computational results. *The Review of Financial Studies*, *4*(2), 315–342.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Bollerslev, T. (1990). Modelling the coherence in short-run nominal exchange rates: A multivariate generalized arch model. *The Review of Economics and Statistics*, 498–505.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Wadsworth & Brooks.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Chevalier, G., Coqueret, G., & Raffinot, T. (2022). Supervised portfolios. *Quantitative Finance*, *22*(12), 2275–2295.

Cho, K., Van Merriënboer, B., Gulçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734.

Chopra, V. K., & Ziemba, W. T. (1993). The effect of errors in means, variances, and covariances on optimal portfolio choice. *Journal of Portfolio Management*, *19*(2), 6–11.

De Nard, G., & Kostovic, D. (2026). Learning the shrinkage intensity: A data-driven approach for risk-optimized portfolios. *Journal of Financial Econometrics*, *24*(2), nbag002.

De Prado, M. L. (2018). *Advances in financial machine learning*. John Wiley & Sons.

Engle, R. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, *20*(3), 339–350.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the Econometric Society*, 987–1007.

Fan, J., Liao, Y., & Mincheva, M. (2011). High dimensional covariance matrix estimation in approximate factor models. *Annals of Statistics*, *39*(6), 3320.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189–1232.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT Press.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), 2223–2273.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

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

Konda, V., & Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in Neural Information Processing Systems*, *12*.

Ledoit, O., & Wolf, M. (2003). Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, *10*(5), 603–621.

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, *88*(2), 365–411.

Ledoit, O., & Wolf, M. (2020). Analytical nonlinear shrinkage of large-dimensional covariance matrices. *The Annals of Statistics*, *48*(5), 3043–3065.

Longin, F., & Solnik, B. (2001). Extreme correlation of international equity markets. *The Journal of Finance*, *56*(2), 649–676.

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, *7*(1), 77–91.

Mattera, G., & Mattera, R. (2023). Shrinkage estimation with reinforcement learning of large variance matrices for portfolio selection. *Intelligent Systems with Applications*, *17*, 200181.

Michaud, R. O. (1989). The markowitz optimization enigma: Is 'optimized' optimal? *Financial Analysts Journal*, *45*(1), 31–42.

Minsky, M., & Papert, S. (1969). *An introduction to computational geometry* (Vol. 479). HIT.

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

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, *12*(4), 875–889.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814.

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

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shapley, L. S. (1953). A value for n-person games. In H. W. Kuhn & A. W. Tucker (Eds.), *Contributions to the theory of games ii* (pp. 307–317, Vol. 28). Princeton University Press.

Silverman, B. W. (2018). *Density estimation for statistics and data analysis*. Routledge.

Stărică, C., & Granger, C. (2005). Nonstationarities in stock returns. *Review of Economics and Statistics*, *87*(3), 503–522.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1). MIT Press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, *12*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267–288.

Wand, M. P., Marron, J. S., & Ruppert, D. (1991). Transformations in density estimation. *Journal of the American Statistical Association*, *86*(414), 343–353.

Watanabe, S. (2023). Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* [Doctoral dissertation, King's College, University of Cambridge].

Werbos, P. J. (2002). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3), 229–256.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *67*(2), 301–320.

Appendix

APPENDIX A

THEORETICAL PROOFS

A.1 Framework and assumptions

Standing Assumptions

1. Asset returns $\{R_t\}_{t=1}^{T}$, $R_t \in \mathbb{R}^N$, have finite second moments.
2. For every rebalancing date t, the shrinkage estimator $\hat{\Sigma}_t(\lambda)$ is positive definite for all $\lambda \in [0, 1]$.
3. There exist constants $0 < m_t \leq M_t < \infty$ such that

$$m_t I \preceq \hat{\Sigma}_t(\lambda) \preceq M_t I \quad \forall \lambda \in [0, 1].$$

where $\preceq$ is the Loewner order. Formally if $A \preceq B$, $B - A$ is positive semi definite.

We observe daily returns $\{R_t\}_{t=1}^{T}$ and rebalancing every H trading days using a rolling estimation window of length L.

Let $\mathcal{T} = \{t_1, \ldots, t_n\}$ denote the rebalancing dates. For each $t \in \mathcal{T}$:

- $R_{t-L+1:t}$ are the returns in the estimation window $\mathcal{W}_t$.
- $R_{t+1:t+H}$ are the returns in the forward holding window $\mathcal{F}_t$.

The sample covariance matrix is given by the following formula where $\bar{R}_t$ is the average return vector over the estimation window $R_{t-L+1:t}$.

$$S_t := \frac{1}{L-1} \sum_{l=t-L+1}^{t} (R_l - \bar{R}_t)(R_l - \bar{R}_t)^\top, \tag{A.1}$$

and the scaled identity target is

$$F_t := \frac{\operatorname{tr}(S_t)}{N} I.$$

The linear shrinkage estimator is defined as

$$\hat{\Sigma}_t(\lambda) := (1-\lambda)S_t + \lambda F_t, \quad \lambda \in [0,1]. \tag{A.2}$$

A.2 GMV portfolio and realised volatility

For any positive definite covariance matrix Σ, the Global Minimum Variance portfolio solves

$$\min_{w \in \mathbb{R}^N} w^\top \Sigma w \quad \text{s.t.} \quad \mathbf{1}^\top w = 1, \tag{A.3}$$

where the unique solution is

$$w^{\mathrm{GMV}}(\Sigma) = \frac{\Sigma^{-1}\mathbf{1}}{\mathbf{1}^\top \Sigma^{-1}\mathbf{1}}. \tag{A.4}$$

Given weights w, realised portfolio returns over the holding window are

$$R^p_{t+k}(w) = w^\top R_{t+k}, \quad k = 1, \ldots, H. \tag{A.5}$$

The realised volatility criterion is

$$\hat{\sigma}_t(w) = \sqrt{\frac{1}{H-1}\sum_{k=1}^{H} (R^p_{t+k}(w) - \bar{R}^p_t(w))^2}. \tag{A.6}$$

For fixed t, define the mapping

$$g_t(\lambda) := \hat{\sigma}_t(w^{\mathrm{GMV}}(\hat{\Sigma}_t(\lambda))). \tag{A.7}$$

Thus $g_t(\lambda)$ measures the realised volatility over $R_{t+1:t+H}$ of the GMV portfolio computed from the shrunk covariance estimated on $R_{t-L+1:t}$.

A.3 Oracle shrinkage intensity

Let $\Lambda \subset [0,1]$ be a finite grid of candidate shrinkage intensities.

Definition of Oracle Optimality.

$$\lambda_t^* := \arg\min_{\lambda \in [0,1]} g_t(\lambda).$$

This follows directly from finite minimisation over a finite grid.

Let δ denote the grid mesh:

$$\delta := \max_i |\lambda_{i+1} - \lambda_i|. \tag{A.8}$$

By Lemma 1, g_t is Lipschitz with constant L_t. Therefore, the discretisation error satisfies.

$$0 \leq g_t(\lambda_t^*) - \min_{\lambda \in [0,1]} g_t(\lambda) \leq L_t \delta. \tag{A.9}$$

Consequently, as $\delta \to 0$, the grid oracle converges to the continuous oracle. With the assumption that $\mathbb{E}[L_t] < \infty$, the discretisation error also vanishes in expectation.

A.4 Lipschitz regularity of g_t

Lemma 1 (Lipschitz Continuity of g_t).
Under the standing assumptions, for each fixed t there exists $L_t < \infty$ such that

$$|g_t(\lambda) - g_t(\lambda')| \leq L_t |\lambda - \lambda'| \quad \forall \lambda, \lambda' \in [0,1].$$

Remark. For each fixed t, the mapping g_t is a random function induced by the randomness of the return samples. The Lipschitz constant L_t is therefore a random variable. All Lipschitz statements are understood to hold almost surely.

Sketch of Proof.

1. $\hat{\Sigma}_t(\lambda)$ is affine in λ:

$$\hat{\Sigma}_t(\lambda) = (1-\lambda) S_t + \lambda F_t \tag{A.10}$$

$$= S_t - \lambda S_t + \lambda F_t \tag{A.11}$$

$$= S_t + \lambda (F_t - S_t) \tag{A.12}$$

$$\hat{\Sigma}_t(\lambda) - \hat{\Sigma}_t(\lambda') = S_t + \lambda(F_t - S_t) - S_t - \lambda'(F_t - S_t) \tag{A.13}$$
$$= \lambda(F_t - S_t) - \lambda'(F_t - S_t) \tag{A.14}$$
$$= (F_t - S_t)(\lambda - \lambda') \tag{A.15}$$

$$\|\hat{\Sigma}_t(\lambda) - \hat{\Sigma}_t(\lambda')\|_F = \|(\lambda - \lambda')(F_t - S_t)\|_F \tag{A.16}$$
$$= |\lambda - \lambda'|\|F_t - S_t\|_F \tag{A.17}$$

2. Because matrix inversion is not globally Lipschitz as if eigenvalues approach zero we have that $\|\Sigma^{-1}\| \to \infty$. Therefore, inversion becomes unstable near singularity. That is why a eigenvalue lower bound is needed :

$$\hat{\Sigma}_t(\lambda) \succeq m_t I \quad \forall \lambda \in [0, 1]$$

This guarantees that :

$$\|\hat{\Sigma}_t(\lambda)^{-1}\|_F \leq \frac{1}{m_t}$$

and using

$$A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1},$$

Applying this with

$$A = \hat{\Sigma}_t(\lambda), \quad B = \hat{\Sigma}_t(\lambda')$$

gives :

$$\hat{\Sigma}_t(\lambda)^{-1} - \hat{\Sigma}_t(\lambda')^{-1} = \hat{\Sigma}_t(\lambda)^{-1}(\hat{\Sigma}_t(\lambda') - \hat{\Sigma}_t(\lambda))\hat{\Sigma}_t(\lambda')^{-1} \tag{A.18}$$

Then, by using submultiplicativity of the Frobenius norm :

$$\|ABC\|_F \leq \|A\|_2\|B\|_F\|C\|_2 \tag{A.19}$$

it becomes,

$$\|\hat{\Sigma}_t(\lambda)^{-1}(\hat{\Sigma}_t(\lambda') - \hat{\Sigma}_t(\lambda))\hat{\Sigma}_t(\lambda')^{-1}\|_F = \|\hat{\Sigma}_t(\lambda)^{-1}\|_2\|\|\hat{\Sigma}_t(\lambda') - \hat{\Sigma}_t(\lambda)\|_F\|\hat{\Sigma}_t(\lambda')^{-1}\|_2 \tag{A.20}$$

$$\|[\hat{\Sigma}_t(\lambda)^{-1} - \hat{\Sigma}_t(\lambda')^{-1}\|_F = \|\hat{\Sigma}_t(\lambda)^{-1}\|_2\|\|\hat{\Sigma}_t(\lambda') - \hat{\Sigma}_t(\lambda)\|_F\|\hat{\Sigma}_t(\lambda')^{-1}\|_2 \tag{A.21}$$

Using the assumption $\hat{\Sigma}_t(\lambda) \succeq m_t I \quad \forall \lambda \in [0, 1]$.

This implies

$$\lambda_{\min}(\hat{\Sigma}_t(\lambda)) \geq m_t.$$

For any symmetric positive definite matrix X,

$$\|X\|_2 = \lambda_{\max}(X) \tag{A.22}$$

$$\|X^{-1}\|_2 = \frac{1}{\lambda_{\min}(X)} \tag{A.23}$$

Hence,

$$\|\hat{\Sigma}_t(\lambda)^{-1}\|_2 = \frac{1}{\lambda_{\min}\big(\hat{\Sigma}_t(\lambda)\big)} \leq \frac{1}{m_t} \quad \text{and} \quad \|\hat{\Sigma}_t(\lambda')^{-1}\|_2 \leq \frac{1}{m_t}.$$

Substituting into the equation A.21

$$\|\hat{\Sigma}_t(\lambda)^{-1} - \hat{\Sigma}_t(\lambda')^{-1}\|_F \leq \frac{1}{m_t}|\|\hat{\Sigma}_t(\lambda') - \hat{\Sigma}_t(\lambda)\|_F \frac{1}{m_t} \tag{A.24}$$

$$\leq \frac{1}{m_t^2}|\|\hat{\Sigma}_t(\lambda') - \hat{\Sigma}_t(\lambda)\|_F \tag{A.25}$$

$$\leq \frac{1}{m_t^2}|\|F_t - S_t\|_F|\lambda - \lambda'| \quad \lambda, \lambda' \in [0,1] \tag{A.26}$$

Thus the map $\lambda \mapsto \hat{\Sigma}_t(\lambda)^{-1}$ is Lipschitz constant

$$L_t = \frac{\|F_t - S_t\|_F}{m_t^2} \tag{A.27}$$

3. The GMV weight map is a smooth function of $\hat{\Sigma}_t(\lambda)^{-1}$ with denominator bounded away from zero due to positive definiteness.

4. The sample volatility as a function of w is Lipschitz on bounded sets.

By composition of Lipschitz mappings, g_t is Lipschitz.

A.5 The learning problem

With the vector $x_t = \phi(R_{t-L+1:t}) \in \mathbb{R}^p$ of p features. The supervised dataset is

$$\mathcal{D} := \{(x_t, \lambda_t^*)\}_{t \in \mathcal{T}}.$$

The supervised dataset $\mathcal{T}$ is split into two sets :

- The training set : $\mathcal{T}_{\text{train}} \subseteq \mathcal{T}$
- The test set : $\mathcal{T}_{\text{test}} \subseteq \mathcal{T}$

Where $\mathcal{T}_{\text{train}} \cap \mathcal{T}_{\text{test}} = \emptyset$ and $\mathcal{T}_{\text{train}} \cup \mathcal{T}_{\text{test}} \subseteq \mathcal{T}$

A regression model $\hat{f}$ (e.g., XGBoost) is trained via

$$\hat{f} := \arg\min_{f \in \mathcal{F}} \frac{1}{|\mathcal{T}_{\text{train}}|} \sum_{t \in \mathcal{T}_{\text{train}}} (\lambda_t^* - f(x_t))^2 \tag{A.28}$$

Predictions are clipped to $[0, 1]$:

$$\hat{\lambda}_t = \Pi_{[0,1]}(\hat{f}(x_t)). \tag{A.29}$$

A.6 Regret bound

Define volatility regret:

$$\Delta_t := g_t(\hat{\lambda}_t) - g_t(\lambda_t^*) \tag{A.30}$$

Proposition 2 (Regret Bound).
Under Lipschitz regularity,

$$\Delta_t \leq L_t |\hat{\lambda}_t - \lambda_t^*|.$$

This implies that small changes in the shrinkage intensity will not lead to large changes in volatility.

Proof.

By Lipschitz continuity,

$$|g_t(\hat{\lambda}_t) - g_t(\lambda_t^*)| \leq L_t |\hat{\lambda}_t - \lambda_t^*| \tag{A.31}$$

Since λ_t^* minimises g_t on Λ, $\Delta_t \geq 0$, hence the result.

A.7 Expected regret

Taking expectations,

$$\mathbb{E}[\Delta_t] \leq \mathbb{E}[L_t |\hat{\lambda}_t - \lambda_t^*|] \tag{A.32}$$

Additional Assumption (Square-integrable Lipschitz constant)
For the expected regret bound, assume $\mathbb{E}[L_t^2] < \infty$

Applying Cauchy–Schwarz,

$$\mathbb{E}[L_t |\hat{\lambda}_t - \lambda_t^*|] \leq \sqrt{\mathbb{E}[L_t^2]} \sqrt{\mathbb{E}[(\hat{\lambda}_t - \lambda_t^*)^2]}. \tag{A.33}$$

The second factor is precisely the RMSE.

Theorem (Learning Implies Regret Control).
Under Lipschitz regularity and $\mathbb{E}[L_t^2] < \infty$, minimising the RMSE of $\hat{\lambda}_t$ minimises an upper bound on expected volatility regret.

A.8 Interpretation

The oracle λ_t^* directly minimises realised future volatility within the grid.

The learned predictor $\hat{\lambda}_t$ does not directly minimise volatility; instead, it minimises RMSE relative to the oracle, which in turn controls expected volatility regret.

Thus, the proposed procedure constitutes a theoretically justified data-driven approximation to the dynamic volatility-optimal shrinkage rule.

APPENDIX B

ADDITIONAL THEORETICAL DETAILS

B.1 Linear Ledoit–Wolf shrinkage estimator

The linear Ledoit and Wolf (2004) estimator chooses the shrinkage intensity $\hat{\lambda}$ by minimising the expected Frobenius loss between the shrinkage estimator and the true covariance matrix Σ:
μ

$$\hat{\lambda} = \arg\min_{\lambda \in [0,1]} \mathbb{E}[\|(1-\lambda)S + \lambda F - \Sigma\|_F^2$$

The population shrinkage intensity can be written as

$$\hat{\lambda} = \frac{b^2}{d^2}$$

where $b^2 = \mathbb{E}[\|S - \Sigma\|_F^2]$ measures the estimation error of the sample covariance matrix and $d^2 = \mathbb{E}[\|S - F\|_F^2]$ measures the distance between the sample covariance matrix and the target.

Since Σ is unknown, b^2 and d^2 are estimated from the data. The distance to the target is estimated by $d^2 = \|S - F\|_F^2$.

The estimation error of the sample covariance matrix is estimated by $\hat{b}^2 = \frac{1}{T^2} \sum_{t=1}^{T} \|x_t x_t^\top - S\|_F^2$, where $x_t = R_t - \bar{R}$ is the demeaned return vector.

The feasible Ledoit-Wolf shrinkage intensity is therefore

$$\hat{\lambda} = \min\left(\frac{\hat{b}^2}{\hat{d}^2}, 1\right).$$

In the empirical implementation, the Ledoit–Wolf shrinkage intensity is re-estimated at each rebalancing date using only the returns contained in the current rolling estimation window. In the baseline specification, this window contains 252 daily observations. Therefore, the estimator

is fully backward-looking and does not use the forward holding period.

This distinction is important because the Ledoit–Wolf benchmark is an implementable covariance estimator, whereas the oracle shrinkage intensity used in the supervised learning framework is computed ex post by minimising realised volatility over the future holding window. Hence, the Ledoit–Wolf estimator serves as a feasible statistical benchmark, while the oracle shrinkage intensity serves only as a theoretical target for learning.

B.2 Kernel density computation

The distributions of the oracle shrinkage intensity $\lambda_t^* \in [0, 1]$ reported in section 4.1 of chapter 3 are estimated by using a kernel density estimator adapted to bounded support. A direct Gaussian kernel estimator would suffer from boundary bias, therefore, a transformation-based approach is used.

With $x_1, \ldots, x_n$ being the observations. Since these already lie in $[0, 1]$, the rescaling to the unit interval is trivial and leaves them unchanged, that is,

$$u_i = \frac{x_i - a}{b - a}, \quad a = 0, \quad b = 1$$

and clipped to $[\epsilon, 1 - \epsilon]$ with $\epsilon = \frac{0.5}{n}$ where n is the number of observations. A logit transformation is then applied,

$$z_i = \log\left(\frac{u_i}{1 - u_i}\right)$$

which maps the data to $\mathbb{R}$.

For any evaluation point $x \in [a, b]$, define similarly

$$u = \frac{x - a}{b - a}, \quad z = \log\left(\frac{u}{1 - u}\right)$$

Kernel density estimation is then performed in the transformed space using a Gaussian kernel,

$$\hat{f}_Z(z) = \frac{1}{nh} \sum_{i=1}^{n} \phi\left(\frac{z - z_i}{h}\right),$$

where $h = 0.9\hat{\sigma}n^{-\frac{1}{5}}$ is selected according to Silverman's rule of thumb (Silverman, 2018), with $\hat{\sigma} = \min\left(s, \frac{IQR}{1.349}\right)$.

The density is then mapped back to the original domain using the change-of-variables formula,

$$\hat{f}_X(x) = \hat{f}_Z(z) \frac{1}{u(1 - u)} \frac{1}{b - a}, \quad u = \frac{x - a}{b - a}$$

which ensures that the estimator respects the bounded support. This transformation-based

procedure, following Wand et al. (1991), provides a bias-corrected and flexible estimate of the distribution of λ_t^*.

B.3 Machine learning specifications

B.3.1 Families of machine learning methods

Machine learning methods can be broadly classified into three main families: supervised learning, unsupervised learning and reinforcement learning, each characterised by a distinct learning objective and information structure (Hastie et al., 2009).

B.3.1.1 Supervised learning

Supervised learning considers a dataset of labelled observations $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ denotes the feature vector and $y_i \in \mathcal{Y}$ the associated target variable, with $\mathcal{Y} \subseteq \mathbb{R}$ being the space of all possible target values. The objective is to estimate a function $f \in \mathcal{F}$, where $\mathcal{F}$ denotes the hypothesis space, i.e. the class of admissible prediction functions $f : \mathbb{R}^p \to \mathcal{Y}$, that minimises the expected prediction error,

$$\min_{f \in \mathcal{F}} \mathbb{E}[\ell(Y, f(X))],$$

where $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_+$ is a loss function measuring the discrepancy between the true outcome Y and the prediction $f(X)$.

Since the joint distribution of (X, Y) is unknown, this objective is typically approximated by empirical risk minimisation,

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)).$$

In this study, the loss function is chosen as the squared error loss,

$$\ell(y, f(x)) = (y - f(x))^2,$$

so that the performance criterion is based on the root mean squared error (RMSE), defined as

$$RMSE(f) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2}.$$

Depending on the nature of $\mathcal{Y}$, supervised learning problems are divided into regression ($\mathcal{Y} \subseteq \mathbb{R}$) and classification ($\mathcal{Y}$ discrete). A key characteristic of supervised learning is that it explicitly exploits the joint distribution of (X, Y), allowing direct optimisation of predictive performance. However, its effectiveness depends critically on the availability and quality of labeled data.

From a statistical perspective, supervised learning involves a bias-variance trade-off: simple models (e.g., linear regression) exhibit high bias but low variance, whereas flexible models (e.g.,

tree-based ensembles) reduce bias at the cost of higher variance. Regularisation and model averaging are therefore central tools in this framework.

B.3.1.2 Unsupervised learning

Unsupervised learning operates on unlabeled data $\{x_i\}_{i=1}^n$ and aims to uncover structural properties of the marginal distribution P_X where P_X is the distribution of the feature vector X. Formally, the objective is not prediction of a target variable, but rather the extraction of latent representations or patterns.

Typical problems include clustering, where the goal is to partition the data into groups $\{C_k\}_{k=1}^K$ that maximise within-group similarity, and dimensionality reduction, where one seeks a lower-dimensional representation $z_i = g(x_i)$, with $g : \mathbb{R}^p \to \mathbb{R}^q$ and $q < p$, that preserves relevant information. For example, principal component analysis (PCA) solves

$$\max_{v \in \mathbb{R}^p} Var(v^\top X) \quad s.t. \quad \|v\|_2 = 1,$$

leading to orthogonal directions of maximal variance.

Unlike supervised learning, unsupervised methods do not rely on an explicit loss function tied to a target variable, which makes evaluation inherently more difficult. However, they are particularly useful for feature engineering and data preprocessing, specifically in high-dimensional settings such as financial applications.

B.3.1.3 Reinforcement learning

Reinforcement learning (RL) addresses sequential decision problems in which an agent interacts with an environment over time (Sutton and Barto, 1998). At each time t, the agent observes a state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$ and receives a reward r_t. The objective is to learn a policy $\pi(a|s)$ which specifies a probability distribution over actions given the current state, that maximises the expected cumulative reward,

$$\max_{\pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

where $\gamma \in (0, 1)$ is a discount factor and the expectation is taken with respect to the distribution over trajectories induced by the policy π.

A central concept in RL is the value function,

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right]$$

which measures the expected return starting from state s under policy π. Learning typically

relies on dynamic programming or stochastic approximation methods, such as Q-learning or policy gradient algorithms.

Compared to supervised learning, RL differs in two fundamental aspects:

1. Feedback is delayed and indirect, as rewards depend on sequences of actions.
2. The data-generating process is endogenous, since actions influence future states.

These characteristics make RL particularly suited for dynamic portfolio allocation problems, although it is not the primary focus of this research.

B.3.1.4 Comparative characteristics

These three learning paradigms differ fundamentally: supervised learning is prediction-driven, unsupervised learning is structure extraction, and reinforcement learning is decision-driven.

In this master thesis, the empirical analysis focuses on supervised learning and reinforcement learning. Supervised models aim to approximate the oracle shrinkage intensity by minimising a squared loss (evaluated using RMSE), while reinforcement learning directly maximises expected cumulative reward, corresponding to portfolio performance. Unsupervised learning is not implemented explicitly but provides a conceptual framework for potential extensions involving feature extraction and dimensionality reduction. In this context, the target variable corresponds to the oracle shrinkage intensity λ_t^*, while the predictors are features derived from the covariance structure.

B.3.2 Supervised learning models

This section presents the supervised learning models used to approximate the oracle shrinkage intensity. The selected models include penalised linear estimators and non-parametric tree-based methods, allowing for varying degrees of flexibility in capturing the relationship between predictors and the target variable.

B.3.2.1 Linear models with regularisation

Considering the linear regression model

$$y = X\beta + \epsilon, \quad y \in \mathbb{R}^n$$

where $X \in \mathbb{R}^{n \times p}$, $y \in \mathbb{R}^n$ and $\epsilon \in \mathbb{R}^n$ is the error vector. In high-dimensional settings or when predictors are strongly correlated, ordinary least squares estimation becomes unstable, as the matrix $X^\top X$ may be ill-conditioned. Regularisation addresses this issue by introducing a penalty term that controls model complexity (Hastie et al., 2009).

Ridge

Ridge regression (Hoerl and Kennard, 1970) solves

$$\hat{\beta}^{\text{Ridge}} = \arg\min_{\beta} \|y - X\beta\| + \alpha\|\beta\|_2^2$$

where $\alpha > 0$ is a tuning parameter and $\|\beta\|_2^2 = \sum_{j=1}^{p} \beta_j^2$. The estimator admits the closed-form solution

$$\hat{\beta}^{\text{Ridge}} = (X^\top X + \alpha I_p) X^\top y.$$

The ℓ_2-penalty shrinks coefficients continuously toward zero, reducing estimator variance at the cost of introducing bias. Ridge regression is particularly effective in the presence of multicollinearity, as it regularises the Gram matrix $X^\top X$, improving numerical stability and out-of-sample performance.

Lasso

The Least Absolute Shrinkage and Selection Operator (Lasso) (Tibshirani, 1996) replaces the ℓ_2-penalty with an ℓ_1-penalty,

$$\hat{\beta}^{\text{Lasso}} = \arg\min_{\beta} \|y - X\beta\|_2^2 + \alpha\|\beta\|_1,$$

where $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$.

Unlike Ridge regression, the Lasso produces sparse solutions due to the non-differentiability of the ℓ_1-norm at zero. This induces variable selection and yields more interpretable models. However, in the presence of highly correlated predictors, the Lasso tends to select a single variable from a group, which may lead to instability in variable selection.

Elastic Net

Elastic net (Zou and Hastie, 2005) combines ℓ_1 and ℓ_2 penalties,

$$\hat{\beta}^{\text{EN}} = \arg\min_{\beta} \|y - X\beta\|_2^2 + \alpha(\delta\|\beta\|_1 + (1-\delta)\|\beta\|_2^2),$$

where $\alpha > 0$ controls the overall regularisation strength and $\delta \in [0, 1]$ determines the relative contribution of the ℓ_1 and ℓ_2 penalties.

This formulation keeps the sparsity-inducing property of the Lasso while mitigating its limitations in the presence of correlated predictors. Particularly, Elastic Net encourages grouped selection, assigning similar coefficients to correlated variables, which is especially relevant in financial applications where predictors often exhibit strong dependence structures.

B.3.2.2 Regression trees

Regression trees provide a non-parametric approach to approximating the conditional expectation function (Breiman et al., 1984)

$$f(x) = \mathbb{E}\left[Y|X = x\right].$$

The estimator takes the form

$$f(x) = \sum_{m=1}^{M} c_m \mathbf{1}\{x \in R_m\},$$

where $\{R_m\}_{m=1}^{M}$ is a partition of the feature space $\mathbb{R}^p$ and c_m is the average response within the region R_m,

$$c_m = \frac{1}{|R_m|} \sum_{x_i \in R_m} y_i$$

The partition is constructed via recursive binary splitting. At each step, the algorithm selects a feature j and a threshold s that minimise the residual sum of squares

$$\min_{j,s} \left[\sum_{x_i \in R_1(j,s)} (y_i - \bar{y}_{R_1})^2 + \sum_{x_i \in R_2(j,s)} (y_i - \bar{y}_{R_2})^2 \right]$$

where

$$R_1(j,s) = \{x : x_j \leq s\}, \qquad R_2(j,s) = \{x : x_j > s\}, \quad \text{and} \quad \bar{y}_R = \frac{1}{|R|} \sum_{x_i \in R} y_i.$$

Regression trees are able to capture non-linearities and high-order interactions without requiring explicit specification. However, they typically exhibit high variance and are sensitive to perturbations in the data, motivating the use of ensemble methods.

B.3.2.3 Ensemble tree-based methods

Random Forest

Random Forest (Breiman, 2001) addresses the high variance of individual trees by averaging a large number of decorrelated trees. Each tree is trained on a bootstrap sample of the data, and at each split, only a random subset of predictors is considered.

In this master thesis, Random Forest is applied to dated feature-target pairs (x_t, λ_t^*), where x_t is constructed from the rolling estimation window. The model is therefore used as a nonlinear supervised learner on a tabular dataset. Although bootstrap resampling does not preserve serial dependence across dated observations, temporal consistency is ensured by the rolling-window design and the purged and embargoed chronological cross-validation procedure.

The Random Forest estimator is given by

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} f_b(x),$$

where f_b denotes the b-th tree. This procedure reduces variance through averaging while maintaining low bias.

Gradient boosting and XGBoost

Gradient boosting constructs an additive model by sequentially fitting base learners to the negative gradient of a loss function (Friedman, 2001). At iteration k, the pseudo-residuals are defined as

$$r_i^{(k)} = - \left. \frac{\partial \ell(y_i, \hat{y}_i)}{\partial \hat{y}_i} \right|_{\hat{y}_i = f^{(k-1)}(x_i)}$$

and a new tree f_k is fitted to these residuals. The model is updated as

$$f^{(k)}(x) = f^{(k-1)}(x) + \eta f_k(x)$$

where $\eta \in (0, 1]$ is a learning rate.

XGBoost (Chen and Guestrin, 2016) extends this framework by introducing explicit regularisation. The objective function is

$$\mathcal{L} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k),$$

where

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i),$$

and the regularisation term is defined as

$$\Omega(f) = \gamma T + \frac{1}{2} \delta \sum_{j=1}^{T} w_j^2,$$

with T the number of leaves and w_j the weights associated with leaf j. This penalisation controls model complexity and improves generalisation performance.

B.4 Tree-structured Parzen Estimator (TPE) and Optuna

B.4.1 Tree-structured Parzen Estimator

The hyperparameters are optimised using Bayesian optimisation, as implemented via Optuna's Tree-structured Parzen estimator (TPE) sampler.

Let $y_i = \mathcal{L}(\theta_i)$ denote the validation loss associated with a hyperparameter configuration $\theta_i \in \Theta$, where $\Theta \subset \mathbb{R}^d$ is the hyperparameter search space and d is the number of hyperparameters. Classical Bayesian optimisation models the conditional density $p(y|\theta)$ and selects new configurations by maximising an acquisition function.

Instead of directly modelling $p(y|\theta)$, the TPE approach models $p(\theta|y)$ by partitioning the observed configurations into two groups based on a quantile threshold γ:

$$\mathcal{D}_{good} = \{\theta_i : y_i \leq \gamma\}, \qquad \mathcal{D}_{bad} = \{\theta_i : y_i > \gamma\}.$$

Two density models are then estimated:

$$l(\theta) = p(\theta|y \leq \gamma), \quad g(\theta) = p(\theta|y > \gamma)$$

The next hyperparameter configuration is selected by maximising the ratio of these densities,

$$\theta^{(t+1)} = \arg\max_{\theta} \frac{l(\theta)}{g(\theta)}, \qquad \theta \in \Theta \subset \mathbb{R}^d$$

The hyperparameter optimisation is initialised with a fixed random seed to ensure reproducibility of the sampling process.

A key feature of the implementation is the use of a multivariate sampling strategy. Unlike the standard TPE formulation, which models each hyperparameter independently via a factorised density

$$p(\theta|y) = \prod_{j=1}^{d} p(\theta_j|y).$$

The multivariate extension considers the joint distribution of the hyperparameter vector $\theta \in \Theta$. This enables the algorithm to capture dependencies between hyperparameters, which is particularly important in high-dimensional or structured models where interactions between parameters significantly affect performance.

In addition, a grouping mechanism is employed to improve sampling efficiency. Specifically, subsets of related hyperparameters are sampled jointly, thereby preserving structural relationships within the parameter space. This approach enhances the ability of the optimisation algorithm to explore regions where coordinated parameter changes yield improved performance.

The optimisation proceeds sequentially. At each iteration $m = 1, \ldots, M$, a candidate configuration $\theta^{(m)}$ is sampled based on the current probabilistic model of promising regions of the search space. The corresponding objective function is evaluated as $y^{(m)} = \mathcal{L}(\theta^{(m)})$ defined as the cross-validated prediction error of the shrinkage intensity. Then λ_t^* is evaluated, yielding $y^{(m)} = \mathcal{L}(\theta^{(m)})$.

The pair $(\theta^{(m)}, y^{(m)})$ is incorporated into the dataset of observed trials, and the TPE model is updated accordingly. Subsequent candidates are generated by maximising an acquisition function derived from the ratio of densities associated with "good" and "bad" observations.

The iterative procedure results in an adaptive exploration of the hyperparameter space, where regions associated with low prediction error are sampled more intensively over time.

B.4.2 Early stopping via pruning

Given the important computational cost associated with evaluating each hyperparameter configuration, particularly due to the use of time-series cross-validation, an early stopping mechanism is introduced to improve efficiency. This mechanism is based on a median-based pruning rule.

$\mathcal{L}_k^{(m)}$ denotes the objective value of trial m at step k, where each step corresponds to a fold of the cross-validation. The pruning criterion compares the current performance of a trial to the historical performance of previously completed trials at the same stage. Specifically, trial m is terminated at step k if

$$\mathcal{L}_k^{(m)} > \text{median}(\mathcal{L}_k^{(1)}, \ldots, \mathcal{L}_k^{(M_k)})$$

where M_k is the number of completed trials for which an intermediate value at step k is available.

To ensure stability of the pruning decision, the mechanism is only activated after an initial set of trials has been completed, thereby providing a reliable estimate of the median benchmark. Furthermore, pruning is applied only after a minimum number of evaluation steps have been observed within a given trial, preventing premature termination based on insufficient information.

Conceptually, this procedure can be interpreted as a dynamic filtering mechanism that allocates computational resources preferentially toward promising regions of the hyperparameter space. Trials that exhibit relatively poor intermediate performance are discarded early, while more competitive configurations are allowed to proceed to full evaluation.

Importantly, pruning does not modify the definition of the objective function itself, rather it alters the allocation of computational effort across candidate configuration. As such, it preserves the statistical validity of the optimisation procedure while substantially reducing the overall computational burden.

B.5 SHAP-based model interpretation

To interpret the predictions of the supervised learning models, this study employs SHAP (Shapley Additive Explanations), a framework grounded in cooperative game theory that provides a consistent decomposition of model predictions into feature-level contributions (Lundberg and Lee, 2017). For a prediction function $f(x)$, SHAP expresses the output as an additive decomposition

$$f(x) = \phi_0 + \sum_{j=1}^{p} \phi_j(x),$$

where $\phi_0 = \mathbb{E}[f(X)]$ denotes the baseline prediction and ϕ_j the contribution of feature j. These contributions correspond to Shapley values (Shapley, 1953), defined as

$$\phi_j(x) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!(p - |S| - 1)!}{p!} \left[f_{S\cup\{j\}}(x_{S\cup\{j\}}) - f_S(x_S) \right]$$

which quantify the marginal contribution of each feature averaged over all possible subsets of features.

As exact computation is infeasible in high-dimensional settings, SHAP relies on model-specific algorithms. For tree-based ensemble models, such as Random Forest and XGBoost, SHAP values are computed using the TreeSHAP algorithm, which exploits the structure of decision trees to evaluate conditional expectations efficiently and exactly (Lundberg et al., 2018). For regularised linear models, such as Ridge, Lasso, and Elastic Net, SHAP values are computed using linear explainers that leverage the additive structure of the model (Lundberg and Lee, 2017).

SHAP values are computed at the observation level, yielding a matrix $\{\phi_j(x_i)\}_{i=1,\ldots,n;j=1,\ldots,p}$ that provides a local decomposition of each prediction. Each value $\phi_j(x_i)$ measures the contribution of feature j to the predicted shrinkage intensity for observation i, relative to the baseline.

In this study, model interpretation is conducted using beeswarm plots, which summarise the distribution of SHAP values across observations for each feature. For a given feature j, the beeswarm plot displays all values $\{\phi_j(x_i)\}_{i=1}^{n}$, with each point representing an individual observation. The horizontal position reflects the magnitude and sign of the contribution, while the colour encodes the corresponding feature value.

Features are ordered by their global importance, measured as the average absolute contribution,

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j(x_i)|,$$

ensuring that the most influential predictors appear at the top of the plot.

Beeswarm plots provide a compact yet highly informative representation of model behaviour. They simultaneously capture:

- the magnitude of feature effects (spread along the horizontal axis),
- the direction of influence (positive or negative contributions),
- and the heterogeneity of effects across observations.

This allows the identification of non-linear relationships, interaction effects, and regime-dependent behaviour that are not visible through aggregate measures alone. In the context of shrinkage intensity estimation, these plots reveal how different market characteristics influence the model's predictions across varying conditions, providing an economically meaningful interpretation of the learned relationships.

B.6 Reinforcement learning models

Throughout this appendix, the index t denotes the rebalancing date, while $l = 1, \ldots, L$ denotes the position of an observation within the estimation window processed by the recurrent architecture. Accordingly, $R_t \in \mathbb{R}^N$ denotes the cross-sectional return vector observed at date t, whereas $X_t = R_{t-L+1:t} \in \mathbb{R}^{L \times N}$ denotes the sequence of returns used as input at decision date t.

B.6.1 Reinforcement learning: general framework

Reinforcement learning (RL) studies sequential decision-making problems in which an agent interacts with an environment over time. The standard formalisation is a Markov Decision Process (MDP) defined by the tuple

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, u, \gamma),$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s' \mid s, a)$ is the transition probability, $u(s, a)$ is the reward function, and $\gamma \in (0, 1]$ is a discount factor (Sutton and Barto, 1998).

At each time t, the agent observes a state s_t, selects an action $a_t \sim \pi_\theta(.|s_t)$ and receives reward u_t. The objective is to maximise the expected discounted cumulative reward:

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^{t-1} u_t\right]$$

where $\mathbb{E}_{\pi_\theta}$ is the expectation under the trajectory distribution induced by policy π_θ.

A policy $\pi_\theta(a|s)$ maps state to probability distributions over actions and learning consists of finding an optimal policy:

$$\pi^* = \arg\max_{\pi} J(\pi).$$

B.6.2 Classes of reinforcement learning algorithms

RL algorithms can be broadly divided into three main classes depending on how the optimal policy is learned: value-based methods, policy-based methods and actor-critic methods.

B.6.2.1 Value-based methods

Value-based methods aim to learn a function that evaluates the quality of states or state-action pairs under a given policy. The central objects are the state-value function which evaluates states under a given policy and the action-value function which evaluates state-action pairs.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} u_{t+k} | s_t = s\right],$$

and the action-value function:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} u_{t+k} | s_t = s, a_t = a\right].$$

The optimal action-value function satisfies the Bellman optimality equation:

$$Q^{*}(s,a) = \mathbb{E}\left[u_t + \gamma \max_{a' \in \mathcal{A}} Q^{*}(s_{t+1}, a') | s_t = s, a_t = a\right].$$

where the maximisation is taken over all admissible actions $a' \in \mathcal{A}$ available in the next state.

Algorithms in this class estimate Q^{*} and derive a policy implicitly via:

$$\pi^{*}(s) = \arg\max_{a \in \mathcal{A}} Q^{*}(s,a).$$

One example is Q-learning (Watkins, 1989), which iteratively updates:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[u_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right].$$

Modern implementations use deep neural networks to approximate Q, leading to deep Q-networks (DQN) (Mnih et al., 2015).

While effective, value-based methods are primarily suited to discrete action spaces, limiting their applicabilty to problems involving continuous decisions.

B.6.2.2 Policy-based methods

Policy-based methods directly parameterise the policy $\pi_{\theta}(a|s)$, and optimise the expected discounted reward $J(\theta) = \mathbb{E}_{\pi_{\theta}}\left[\sum_{t=1}^{T} \gamma^{t-1} u_t\right]$. Using the log-derivative trick, the policy gradient is

given by (Williams, 1992):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t \right]$$

where

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} u_k$$

is the discounted cumulative reward.

In practice, this corresponds to minimising the loss

$$\mathcal{L}(\theta) = -\mathbb{E} \left[\log \pi_\theta(a_t|s_t) G_t \right] .$$

Policy-based methods offer several advantages:

- they naturally handle continuous action spaces,
- they allow for stochastic policies, facilitating exploration and
- they avoid the need to solve a maximisation problem over actions.

However, they typically suffer from high variance in gradient estimates.

In this study, the proposed algorithms (PGA and RPGA) belong to this class, as they directly learn a stochastic policy for the shrinkage intensity, which is a continuous decision variable.

B.6.2.3 Actor-Critic methods

Actor-critic methods combine the strengths of value-based and policy-based approaches. They consist of:

- an actor, which parameterises the policy $\pi_\theta(a|s)$,
- a critic, which estimates a value function $V^\pi(s)$ or $Q^\pi(s, a)$.

The policy gradient is modified using the advantage function:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s),$$

leading to:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, A^\pi(s_t, a_t) \right]$$

The critic reduces variance by providing a baseline estimate of expected returns (Konda and Tsitsiklis, 1999).

Popular algorithms in this class include Advantage Actor-Critic (A2C) and Proximal Policy Optimisation (PPO) (Schulman et al., 2017). These methods are particularly effective in high-

dimensional and continuous settings but require careful joint training of both components.

B.6.3 Function approximators: MLP and GRU

B.6.3.1 Multilayer Perceptron (MLP)

The perceptron

The perceptron is the fundamental building block of artificial neural networks and can be interpreted as a parametric function mapping an input vector to a scalar output through a linear transformation followed by a nonlinear activation (Rosenblatt, 1958).

Given an input vector $x \in \mathbb{R}^d$, the perceptron computes:

$$z = w^\top x + b$$

where $w \in \mathbb{R}^d$ is a vector of weights and $b \in \mathbb{R}$ is a bias term. The output is then obtained by applying an activation function ϕ:

$$y = \phi(z)$$

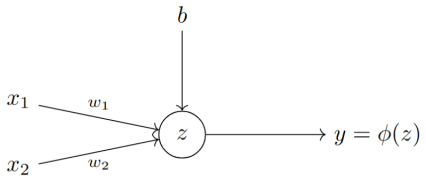


Figure B.1: Visualisation of a perceptron

In its original formulation, the perceptron uses a binary step function:

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

This formulation corresponds to a linear classifier, as the decision boundary is given by the hyperplane:

$$w^\top x + b = 0$$

While the perceptron is capable of separating linearly separable data, it is limited in its expressive power and cannot represent nonlinear decision boundaries (Minsky and Papert, 1969).

Activation functions

To overcome these limitations, modern neural networks employ nonlinear activation functions, enabling the approximation of complex mappings (Goodfellow et al., 2016). Common choices include

1. Sigmoid function:
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$
which maps inputs to the interval $(0, 1)$. It is often used when outputs must be interpreted as probabilities.

2. Hyperbolic tangent:
$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$
which maps inputs to $(-1, 1)$ and is zero-centred.

3. Rectified Linear Unit (ReLU):
$$\phi(z) = \max(0, z)$$
which offers several advantages such as computational simplicity, alleviation of the vanishing gradient problem and sparse activation patterns. As a result, it has become the standard choice in deep architectures (Nair and Hinton, 2010).

From perceptron to multilayer perceptron

A single perceptron can only represent linear mappings. To capture nonlinear relationships, multiple perceptrons are stacked into layers, forming a multilayer perceptron (MLP).

A MLP is a feedforward neural network (neural network where information flows into a single direction) composed of an input layer, one or more hidden layers and an output layer. Each layer applies an affine transformation followed by a nonlinear activation. Formally, for L layers:

$$f_\theta(x) = h^{(L)}$$

where:

$$\begin{aligned} h^{(1)} &= \phi\big(W^{(1)}x + b^{(1)}\big), \\ h^{(2)} &= \phi\big(W^{(2)}h^{(1)} + b^{(2)}\big), \\ &\vdots \\ h^{(L)} &= \phi\big(W^{(L)}h^{(L-1)} + b^{(L)}\big). \end{aligned}$$

Here, $W^{(l)}$ are weight matrices, $b^{(l)}$ are bias vectors, ϕ is a hidden-layer activation function and ψ is the output activation.

The parameter set is denoted: $\theta = \{W^{(l)}, b^{(l)}\}_{l=1}^{L}$.

B.6.3.2 Universal approximation property

A key theoretical justification for MLPs is the universal approximation theorem (Hornik et al., 1989), which states that a feedforward neural network with at least one hidden layer and a

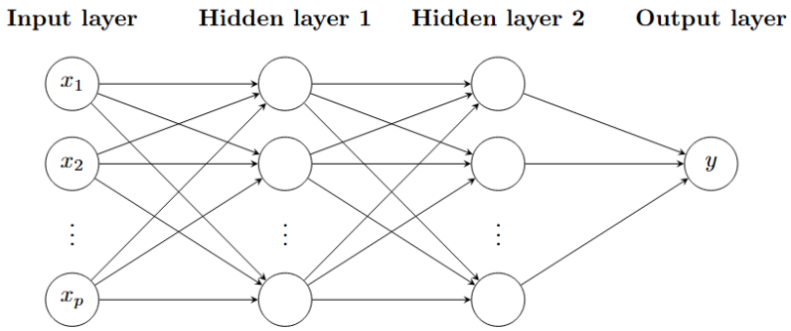


Figure B.2: Visualisation of a MLP

nonlinear, non-polynomial activation function can approximate any continuous function on a compact domain $\mathbb{R}^d$ to arbitrary precision.

Formally for any continuous function f and any $\epsilon > 0$, there exists an MLP f_θ such that:

$$\sup_{x \in K} |f(x) - f_\theta(x)| < \epsilon$$

which means that the largest difference between the neural network and the true function (anywhere in the region K) is smaller than ϵ.

B.6.3.3 Training via gradient-based optimisation

The parameters θ are learned by minimising a loss function $\mathcal{L}(\theta)$, typically using gradient-based optimisation methods.

Gradients are computed via the backpropagation algorithm, which applies the chain rule to efficiently evaluate $\nabla_\theta \mathcal{L}(\theta)$ (Rumelhart et al., 1986). In practice, stochastic optimisation methods such as Adam (Kingma and Ba, 2014) are used to update parameters iteratively.

B.6.3.4 Interpretation in the context of financial data

In the present framework, the MLP serves as a nonlinear function approximator for the policy mapping:

$$\mu_\theta : \mathbb{R}^{L \times N} \to (0, 1)$$

which associates a rolling window of asset returns with a shrinkage intensity.

The output is constrained to the interval $(0, 1)$ through the use of a sigmoid activation function at the final layer. This ensures that the predicted shrinkage intensity remains admissible.

Importantly, the MLP treats the input as a static vector, meaning that temporal dependencies are only implicitly encoded through the input window and not explicitly modelled. This limitation motivates the use of recurrent architectures, which are introduced in the following section.

B.6.4 Gated Recurrent Unit (GRU)

B.6.4.1 Recurrent neural network and sequential modeling

While multilayer perceptrons provide flexible nonlinear mappings, they are inherently limited to static input-output relationships. In contrast, many applications, particularly in finance, require modelling sequential data with temporal dependencies.

Recurrent Neural Networks (RNNs) address this limitation by introducing a hidden state that evolves along the input sequence. Given a sequence $(x_l)_{l=1}^{L}$, an RNN defines

$$h_l = f_\theta(h_{l-1}, x_l)$$
$$y_l = g_\theta(h_l)$$

where $h_l \in \mathbb{R}^d$ is the hidden state at sequence position l, summarising past information.

However, standard RNNs suffer from vanishing and exploding gradient problems, which limit their ability to capture long-range dependencies (Bengio et al., 1994). To overcome these issues, gated architectures such as the Gated Recurrent Unit (GRU) have been introduced (Cho et al., 2014).

B.6.4.2 GRU architecture

The GRU augments the standard RNN with gating mechanisms that regulate the flow of information. At each sequence position l, given input $x_l \in \mathbb{R}^N$ and previous hidden state $h_{l-1} \in \mathbb{R}^d$, the GRU computes:

1. Update gate:
$$z_l = \sigma(W_z x_l + U_z h_{l-1} + b_z)$$

2. Reset gate:
$$r_l = \sigma(W_r x_l + U_r h_{l-1} + b_r)$$

3. Candidate hidden state:
$$\tilde{h}_l = tanh(W_h x_l + U_h(r_l \odot h_{l-1}) + b_h)$$

4. Hidden state update:
$$h_l = (1 - z_l) \odot h_{l-1} + z_l \odot \tilde{h}_l$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function, $tanh(x)$ is the hyperbolic tangent, $\odot$ denotes element-wise multiplication and $W_z, U_z, b_z, W_r, U_r, b_r, W_h, U_h, b_h$ are learnable parameters.

B.6.4.3 Interpretation of the gating mechanisms

The GRU can be interpreted as a dynamic filtering mechanism:

- The update gate z_l controls the trade-off between keeping past information and incorporating new information. When $z_l \approx 1$, the model prioritises the new candidate state; whereas when $z_l \approx 0$, it keeps the previous state.
- The reset gate r_l determines how much past information contributes to the candidate state. A small value of r_l effectively forgets past information.
- The hidden state update:

  $$h_l = (1 - z_l)h_{l-1} + z_l\tilde{h}_l$$

  can be viewed as a convex combination of past memory and new information.

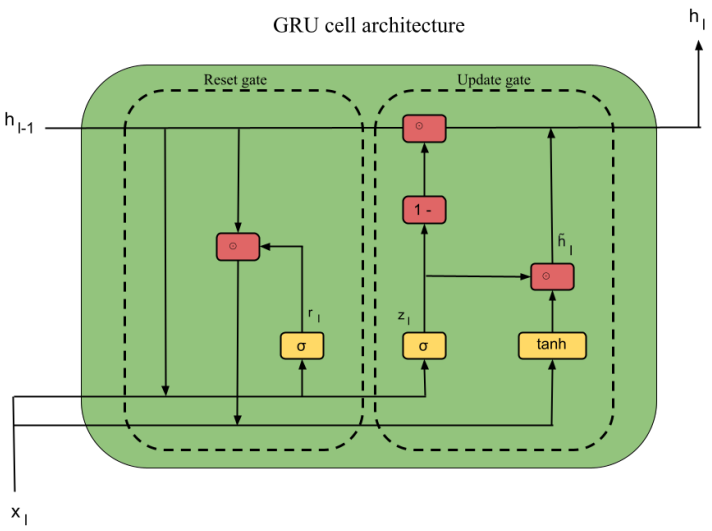


Figure B.3: Visualisation of a GRU cell

This gating structure enables the GRU to capture both short-term fluctuations and long-term dependencies, while mitigating gradient instability.

B.6.4.4 Stacked GRU layers

In practice, multiple GRU layers can be stacked to increase representational power. For a two-layer GRU:

$$h_l^{(1)} = GRU_1(x_l, h_{l-1}^{(1)})$$
$$h_l^{(2)} = GRU_2(h_l^{(1)}, h_{l-1}^{(2)})$$

where each layer has its own set of parameters. Lower layers capture short-term dynamics, while

higher layers encode more abstract temporal patterns.

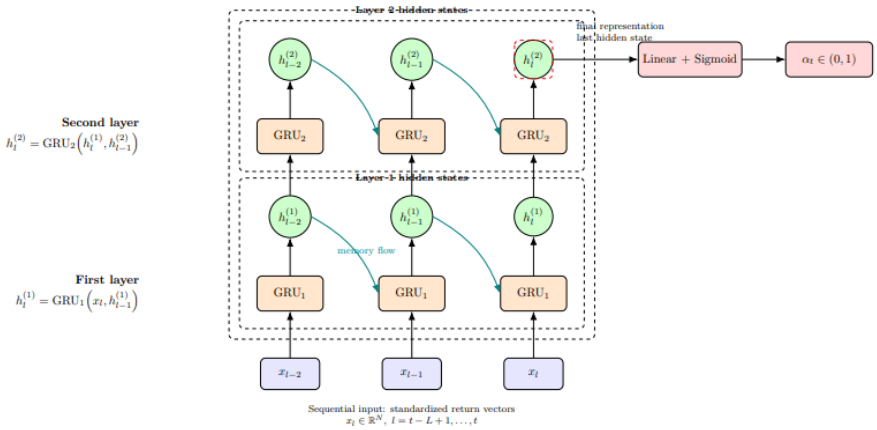


Figure B.4: Visualisation of the GRU logic

Stacking allows the model to learn hierarchical temporal representations, where lower layers capture local dynamics and higher layers encode more abstract temporal patterns.

B.6.4.5 Training and backpropagation through time

The parameters of the GRU are learnt by minimising a loss function via gradient-based optimisation. Gradients are computed using backpropagation through time (BPTT) (Werbos, 2002), which extends standard backpropagation to sequential settings.

In practice, truncated BPTT is often used to limit computational cost and improve numerical stability.

B.6.4.6 Interpretation in financial time series

In the context of financial data, the GRU provides a natural framework for modelling volatility clustering, serial correlation and regime changes.

The hidden state h_l can be interpreted as a latent representation of market conditions, summarising past return dynamics. Formally, for the input sequence associated with rebalancing date t, the GRU defines a mapping from the return window

$$X_t = R_{t-L+1:t}$$

to a sequence of hidden states $(h_l)_{l=1}^{L}$ with the final hidden state h_L providing a finite-dimensional summary of the information contained in the window. Equivalently, one may write

$$h_L = f_\theta(X_t)$$

where f_θ denotes the nonlinear transformation implemented by the GRU.

This latent representation is then used by the policy network to generate the shrinkage intensity, allowing the model to adapt dynamically to evolving market conditions.

B.6.5 Comparison between MLP and GRU architectures

B.6.5.1 Static versus dynamic mappings

A fundamental distinction between multilayer perceptrons (MLPs) and Gated Recurrent Units (GRUs) lies in the nature of the mappings they implement.

A MLP defines a static nonlinear mapping

$$y = f_\theta(x)$$

where the output depends exclusively on the current input x. In time series applications, this input typically consists of a fixed window of past observations, for instance

$$x_t = \mathrm{vec}(R_{t-L+1:t}).$$

Temporal information is therefore incorporated only through a finite-dimensional representation of the past. Importantly, the mapping itself is memoryless: once the input vector is specified, the model does not keep any additional internal state.

In contrast, a GRU defines a dynamic mapping with memory.

$$h_l = f_\theta(h_{l-1}, x_l), \qquad y_l = g_\theta(h_l)$$

where h_l is the hidden state at sequence position l. The hidden state evolves recursively over time and provides a low-dimensional sufficient representation of past information under the model parameterisation. As a result, the output at sequence position l depends not only on the current input x_l, but also on the entire history encoded in h_{l-1}.

This recursive structure allows GRUs to represent sequential dependencies in a fundamentally different way compared to MLPs, which rely solely on fixed input windows.

B.6.5.2 Modelling temporal dependencies

In a MLP-based architecture, temporal dependencies are incorporated implicitly through the input construction. The model receives a stacked vector of past observations, but does not explicitly account for the sequential structure of the data. In particular, there is no parameter sharing across time and the ordering of observations is not exploited beyond their position in the input vector.

By contrast, GRUs are especially designed to model sequential data. This recursive formulation

$$h_l = (1 - z_l)h_{l-1} + z_l\tilde{h}_l$$

introduces a state-dependent update mechanism, where the update gate z_l and the candidate hidden state $\tilde{h}_l$ control how new information is incorporated. This structure enables the network to selectively keep or discard past information.

From a probabilistic perspective, the hidden state h_l can be interpreted as a latent Markovian representation of the underlying process, summarising all relevant past information for predicting future outcomes. This makes GRUs particularly well-suited for financial time series, which exhibit persistence, regime shifts and time-varying dependence structures.

B.6.5.3 Dimensionality and memory

A key practical difference between MLPs and GRUs concerns the dimensionality of the input and the associated parameterisation.

In MLPs, the input vector grows with the length of the observation window:

$$\dim(x_t) = L \times N$$

where L is the window size and N is the number of assets. As L increases, this leads to a rapid growth in dimensionality, exacerbating the curse of dimensionality and increasing the risk of overfitting. Moreover, the number of model parameters typically scales with the input size, making estimation more difficult in high-dimensional settings.

In contrast, GRUs process inputs sequentially:

$$x_l \in \mathbb{R}^N$$

and maintain a hidden state $h_l \in \mathbb{R}^F$ of fixed dimension F, independent of the sequence length. The number of parameters is therefore independent of the time horizon and information from the past is compressed into the hidden state through the gating mechanism.

B.6.5.4 Implications for reinforcement learning models

The architectural differences between MLPs and GRUs directly motivate the distinction between the two reinforcement learning approaches considered in this thesis.

The policy gradient algorithm (PGA) relies on a MLP and defines a static policy, where decisions depend solely on a fixed representation of recent observations. As such, the model does not explicitly account for temporal dependencies beyond those encoded in the input vector.

In contrast, the Recurrent Policy Gradient Algorithm (RPGA) employs a GRU and defines a dynamic policy with memory. The hidden state evolves recursively along the input sequence and provides a latent representation of past information, allowing decisions at date t to depend on the sequential structure of the observations contained in the input window. This distinction is particularly relevant in financial applications, where market dynamics exhibit persistence, regime shifts and time-varying dependence structures. In this context, recurrent architectures offer a more flexible framework for capturing temporal dependencies and adapting to changing environments.

A detailed formulation of the PGA and RPGA models, including their optimisation procedures is provided in the following sections.

B.6.6 Policy Gradient Algorithm (PGA)

B.6.6.1 Stochastic policy parameterisation

In the considered policy gradient framework, the agent does not directly output a deterministic action. Instead, it parameterises a stochastic policy that defines a probability distribution over admissible actions conditional on the observed state (Sutton et al., 1999). The policy is parameterised by a neural network μ_θ which defines the conditional mean of a Gaussian policy π_θ.

The policy is modelled as a Gaussian distribution:

$$\pi_\theta(a_t|s_t) = \mathcal{N}(a_t|\mu_\theta(s_t), \sigma_\varepsilon^2),$$

where θ denotes the parameters of the policy network $\mu_\theta(s_t) \in (0, 1)$ is the conditional mean and $\sigma_\varepsilon > 0$ is a scalar parameter controlling the dispersion of the distribution. In the present implementation, the mean $\mu_\theta(s_t)$ is obtained as the output of a multilayer perceptron, while σ_ε is gradually decreased during training.

The quantity $\mu_\theta(s_t)$ plays a fundamental role in the policy parameterisation. It can be interpreted as the conditional expectation of the action under the policy π_θ, that is, $\mu_\theta \approx \mathbb{E}_{\pi_\theta}[a_t|s_t]$. Rather than directly solving an optimisation problem over actions, the policy gradient approach adjusts the parameters θ so as to increase the likelihood of actions that yield higher realised rewards.

To ensure exploration, the action is sampled from the policy distribution as:

$$a_t = \mu_\theta(s_t) + \sigma_\varepsilon \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 1).$$

This stochastic sampling mechanism allows the agent to explore actions in the neighbourhood of its current estimate. In particular, a larger value of σ_ε induces greater variability in the sampled actions and promotes exploration, while a smaller value results in more concentrated actions

around the mean, corresponding to exploitation.

Given that the admissible action space is bounded, the sampled action is projected onto the interval $[0, 1]$ with $\lambda_t = \text{clip}(a_t, 0, 1)$.

This ensures that the shrinkage intensity remains well-defined.

B.6.6.2 Policy gradient estimation

The stochastic nature of the policy is central for gradient-based learning. The parameters θ are updated using the likelihood ratio method (Williams, 1992), which relies on the gradient of the log-policy:

$$\nabla_\theta \log \pi_\theta(a_t|s_t)$$

For a Gaussian policy, this gradient takes the form:

$$\nabla_\theta \log \pi_\theta(a_t|s_t) = \frac{a_t - \mu_\theta(s_t)}{\sigma_\varepsilon^2} \nabla_\theta \mu_\theta(s_t)$$

This expression holds since the variance is treated as a constant and does not depend on θ, moreover it provides a clear interpretation of the learning mechanism. The term $(a_t - \mu_\theta(s_t))$ measures the deviation between the sampled action and the current mean prediction. When combined with the discounted cumulative reward G_t, it determines the direction in which the parameters are updated.

In particular, if an action a_t yields a high reward and lies above the mean, the gradient update increases $\mu_\theta(s_t)$. Conversely, if a poorly performing action is sampled, the update shifts the mean away from that action. Thus, the policy gradient update progressively moves the mean toward regions of the action space associated with higher rewards.

B.6.6.3 State representation and policy network

In the implemented PGA, the state is constructed directly from the matrix of raw returns observed over the estimation window. $R_{t-L+1:t} \in \mathbb{R}^{L \times N}$ is denoted as the matrix of returns for N assets over the last L observations. Before entering the network (MLP), the return matrix is standardised asset by asset using the training sample means and standard deviations. Denoted by $\bar{R}_i$ and s_i the training-sample mean and standard deviation of asset i, the standardised return on each estimation window is

$$\tilde{R}_i = \frac{R_i - \bar{R}_i}{s_i}$$

Then, the standardised window is flattened into a vector, $x_t = \text{vec}(\tilde{R}_{t-L+1:t}) \in \mathbb{R}^{L \times N}$ which serves as the input of the policy network. Hence, although the underlying data are sequential, the PGA treats the return window as a fixed-dimensional state vector. This is precisely the static mapping discussed in section B.6.5, and it implies that temporal dependencies are only captured

implicitly through the window itself rather than through an explicit memory mechanism. This implies that the Markov property is imposed on the finite observation window rather than on the full return history

The policy network is a MLP with three hidden layers. In the implementation, its architecture is $LN \to 256 \to 128 \to 32 \to 1$.

With $h^{(0)} = x_t$, the hidden layers satisfy

$$h^{(1)} = \phi(W^{(1)}h^{(0)} + b^{(1)}),$$
$$h^{(2)} = \phi(W^{(2)}h^{(1)} + b^{(2)}),$$
$$h^{(3)} = \phi(W^{(3)}h^{(2)} + b^{(3)}),$$

where $\phi(z) = \max(0, z)$ is the ReLU activation. A layer normalisation step is applied after the first linear transformation in order to stabilise the scale of intermediate activations. The output layer is given by

$$\mu_\theta(s_t) = \sigma(W^{(4)}h^{(3)} + b^{(4)})$$

where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function. The sigmoid ensures that the conditional mean of the policy lies in the interval $(0, 1)$ which is coherent with the admissible range of the shrinkage intensity.

All the linear layers are initialised using Xavier uniform initialisation (Glorot and Bengio, 2010), which improves optimisation by controlling the scale of forward activations and backward gradients at the start of training.

B.6.6.4 Reward function

The reward is defined as the negative realised volatility over the subsequent holding window of length H. If $R_{t+1:t+H}$ is the matrix of future returns, then

$$u_t = -\sqrt{\frac{1}{H-1}\sum_{k=1}^{H}(w_t^\top R_{t+k} - \bar{R}_t)^2}$$

where $\bar{R}_t = \frac{1}{H}\sum_{k=1}^{H} w_t^\top R_{t+k}$. Volatility is used instead of the Sharpe ratio to avoid instability due to division by small returns and to ensure a well-behaved learning signal.

This reward definition implies that the agent is not trained to reproduce the ex-post oracle shrinkage intensity directly. Instead it is trained to select shrinkage intensities that yield low future realised portfolio volatility. This distinction is conceptually important. The reinforcement learning problem is objective-driven rather than target-driven.

The parameters θ are estimated using the REINFORCE algorithm of Williams (1992). As previously, the objective function is

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} \gamma^{t-1} u_t \right]$$

where τ is a trajectory generated by the policy.

For each episode, the discounted cumulative reward from step t onward is computed recursively as

$$G_t = u_t + \gamma G_{t+1}$$

with the terminal condition $G_T = u_T$. Equivalently,

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} u_k$$

The policy gradient identity yields

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right]$$

In practice, the implementation uses the Monte Carlo estimator

$$\widehat{\nabla_\theta J(\theta)} = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \tilde{G}_t$$

with $\tilde{G}_t$ being the normalised discounted cumulative reward. Specifically,

$$\tilde{G}_t = \frac{G_t - \bar{G}}{s_G + \epsilon}$$

where $\bar{G}$ and s_G are the sample mean and standard deviation of discounted cumulative rewards within the episode. This normalisation corresponds to a baseline subtraction and reduces the variance of the policy gradient estimator.

The corresponding empirical loss is

$$\mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) \tilde{G}_t$$

B.6.6.5 Optimisation and training procedure

The policy parameters are optimised using the Adam algorithm (Kingma and Ba, 2014) and in the implementation, the learning rate η is decreased linearly over episodes:

$$\eta_e = \eta_{init}(1 - \rho_e) + \eta_{min}\rho_e$$

where $\rho_e \in [0, 1]$ is the normalised training progress. This gradual decay allows relatively large steps early in training and smaller, more stable updates near convergence.

At the same time, the exploration standard deviation is reduced according to an exponential schedule,

$$\sigma_{\varepsilon,e+1} = \max\{\sigma_{\varepsilon,min}, \delta\sigma_{\varepsilon,e}\}$$

where $\delta \in (0, 1)$ is the decay factor and $\sigma_{\varepsilon,1} = \sigma_{\varepsilon,\text{init}}$. This schedule governs the exploration-exploitation trade-off. Initially, a larger standard deviation promotes exploration over the action space. Then, as training progresses, the standard deviation shrinks and the policy becomes more concentrated around the learned mean.

To improve numerical stability, gradient clipping is applied after backpropagation: $\|\nabla_\theta \mathcal{L}\| \leq 1$. This prevents excessively large updates and contributes to stable training dynamics. The learning curves reported in appendix C are consistent with this design and show the evolution of episode reward and average shrinkage intensity over training. These stabilisation mechanisms are particularly important in policy gradient methods because stochastic exploration and noisy Monte Carlo reward signals can generate high-variance gradient estimates.

B.6.6.6 Deterministic prediction and economic interpretation

After training, the learned policy is used deterministically in the out-of-sample period. Instead of sampling from the Gaussian policy, the predicted shrinkage coefficient is set equal to the network output $\hat{\lambda}_t = \mu_\theta(s_t)$. Since the network already applies a sigmoid output layer, the prediction naturally lies in $(0, 1)$, and a final clipping step ensures numerical admissibility.

From an economic perspective, the PGA learns a nonlinear state-dependent shrinkage rule:

$$\hat{\lambda}_t = f_\theta(vec(\tilde{R}_{t-L+1:t}))$$

where the parameters of f_θ are chosen not to minimise a prediction loss relative to the oracle coefficient but to optimise a long-run portfolio objective based on realised risk. The learned policy therefore reflects an adaptive regularisation mechanism. When recent patterns suggest that the sample covariance matrix is unreliable, the network can shift the distribution of actions toward higher shrinkage whereas when the recent covariance structure appears more informative, it can favour lower shrinkage levels.

At the same time, because PGA is based on a MLP, the policy is conditionally independent of past observations given the current state representation: all relevant information must be extracted from the current standardised return window. This limitation motivates the recurrent extension developed in the RPGA specification, where the policy can depend on an evolving hidden state and thus exploit information that extends beyond the current input window.

B.6.7 Recurrent Policy Gradient Agent (RPGA)

The Recurrent Policy Gradient Agent (RPGA) extends the PGA framework by explicitly modelling the temporal structure of the estimation window. While the PGA treats the state as a fixed-dimensional vector obtained by flattening past returns, the RPGA preserves the sequential nature of the data and augments the policy with a latent memory. This allows the agent to condition its decisions not only on the current window of observations but also on information extracted from preceding windows.

B.6.7.1 Sequential state representation

In contrast to the feedforward model, the RPGA represents each state as a sequence of standardised returns $X_t = \tilde{R}_{t-L+1:t} \in \mathbb{R}^{L\times N}$. Each observation in the sequence corresponds to a cross-section of asset returns at a given date, and the ordering of observations is preserved. The standardisation procedure is identical to the one described previously, but the resulting data is no longer reshaped into a vector. Instead, it is directly fed into the recurrent network as time-ordered sequence of length L.

B.6.7.2 Recurrent policy parameterisation

The policy is parameterised by a two-layer Gated Recurrent Unit (GRU) network. Let $X_{t,l} \in \mathbb{R}^N$ be the l-th row of X_t, that is, the return vector at position l within the window associated with decision date t.

The sequential processing of the estimation window is then defined by

$$h_{t,l}^{(1)} = \mathrm{GRU}_1(X_{t,l}, h_{t,l-1}^{(1)})$$

and

$$h_{t,l}^{(2)} = \mathrm{GRU}_2(h_{t,l}^{(1)}, h_{t,l-1}^{(2)}),$$

for $l = 1, \ldots, L$, where $h_{t,l}^{(1)}$ and $h_{t,l}^{(2)}$ denote the hidden states of the first and second GRU layers at position l within the window. The initial hidden states $h_{t,0}^{(1)}$ and $h_{t,0}^{(2)}$ are given by the recurrent states carried over from the previous decision step.

The shrinkage decision at rebalancing date t is based on the final hidden representation of the

second layer. More precisely, the conditional mean of the policy is defined as

$$\mu_\theta(X_t) = \sigma(W_0 h_{t,L}^{(2)} + b_0),$$

where $\sigma(\cdot)$ denotes the sigmoid activation function and $h_{t,L}^{(2)}$ is the final hidden state obtained after processing the entire input sequence of length L. This specification ensures that the predicted shrinkage intensity remains in the admissible interval $(0, 1)$.

The action is then sampled according to

$$\lambda_t = \mu_\theta(X_t) + \sigma_\varepsilon \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, 1),$$

and is clipped to $[0, 1]$ whenever necessary for numerical admissibility.

In the implementation, the GRU consists of two stacked layers with hidden dimensions $(32, 16)$, followed by a linear output layer and a sigmoid activation. The weight matrices are initialised using Xavier initialisation (Glorot and Bengio, 2010), which promotes stable signal propagation across layers during training.

B.6.7.3 History-dependent policy

A key distinction with respect to the PGA lies in the introduction of a recurrent hidden state. The policy therefore takes the form

$$\pi_\theta(\lambda_t | s_t), \quad \text{where} \quad s_t = (X_t, h_{t,0}^{(1)}, h_{t,0}^{(2)})$$

where the state X_t denotes the current observation window and $h_{t,0}^{(1)}$ and $h_{t,0}^{(2)}$ denote the initial hidden states of the first and second GRU layers at rebalancing date t, inherited from the previous step of the episode.

At the beginning of each episode, hidden states are initialised to zero and subsequently propagated across the sequence of training windows. As a result, the action at time t depends on both the current estimation window and the latent representation of earlier market conditions. This construction provides a natural approximation of a partially observed decision process, in which the agent builds an internal representation of the underlying market dynamics.

To ensure tractable optimisation, hidden states are detached after each step, preventing gradients from propagating indefinitely through the entire sequence of windows. This corresponds to a truncated backpropagation scheme, commonly adopted in recurrent policy gradient methods.

B.6.7.4 Exploration and optimisation scheme

While the underlying policy gradient objective remains unchanged, the RPGA differs from the PGA in its treatment of exploration and optimisation.

First, the learning rate is kept constant throughout training in contrast to the decay schedule used previously. Second, the exploration standard deviation follows a linear decay: $\sigma_{\varepsilon,e} = \sigma_{\varepsilon,init}(1 - \rho_e) + \sigma_{\varepsilon,min}\rho_e$ where $\rho_e \in [0, 1]$ is the relative progress of the training procedure. This schedule ensures a smoother transition from exploration to exploitation compared with the exponential decay used in the PGA.

These choices reflect the greater expressive capacity of the recurrent architecture which typically requires less aggressive learning rate adaptation but benefits from a smoother exploration schedule.

B.6.7.5 Deterministic inference with memory

At the prediction stage, the predicted shrinkage intensity is given by $\lambda_t = \mu_\theta(s_t)$ where the state s_t includes both the current observation window and the initial hidden states carried over from the previous rebalancing date. Although stochastic sampling is removed, the recurrent structure remains active during inference, allowing the agent to exploit temporal dependencies learned during training.

B.6.7.6 Interpretation

The RPGA can be interpreted as a dynamic extension of the PGA in which the shrinkage decision is conditioned on both contemporaneous observations and a latent memory of past states. This additional structure is particularly relevant in financial applications, where covariance dynamics often exhibit persistence and regime dependence.

By leveraging recurrent representations, the RPGA is able to capture temporal patterns that are not directly observable within a single estimation window. This corresponds to a dynamic policy $\pi_\theta(\lambda_t|s_t)$ where the state evolves according to a latent state transition induced by the GRU.

In contrast to the PGA, where the policy is conditionally independent of past observations given the input window, the RPGA explicitly incorporates temporal dependencies through the hidden state, leading to a richer state representation.

APPENDIX C

ADDITIONAL EMPIRICAL RESULTS

This appendix provides additional empirical results that complement the main analysis and offer deeper insights into the behaviour and implementation of the different models considered in this master thesis.

The analysis is organised model by model. For each model, we present a consistent set of additional elements that help better understand its empirical performance and underlying mechanisms. Finally, all the additional empirical insights are drawn from the S&P 500 data.

C.1 Lasso

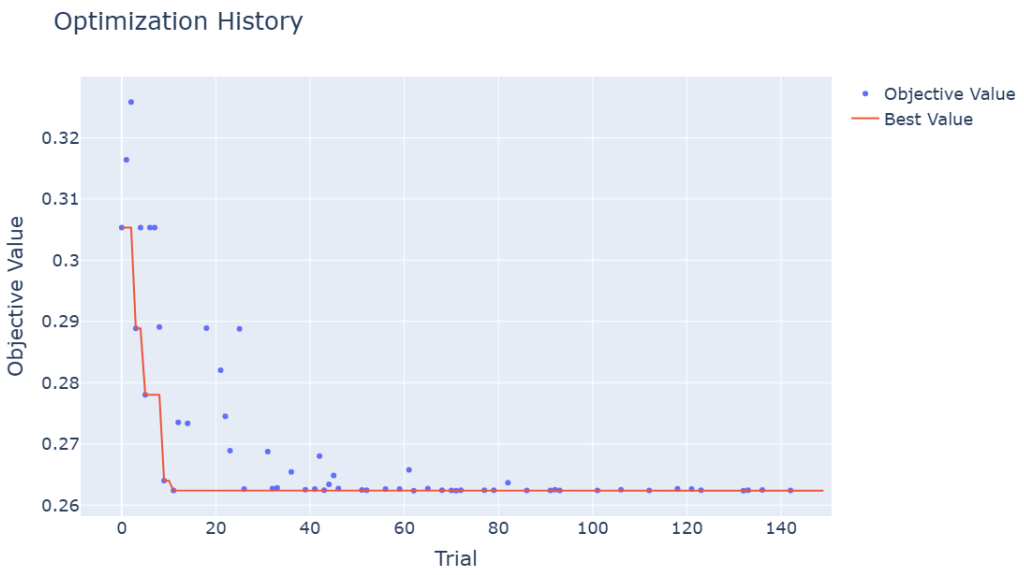


Figure C.1: Lasso: Optimisation history

The optimisation of the Lasso model converges rapidly, with most of the improvement achieved within the first iterations. After approximately 10 – 15 trials, the objective function stabilises and no further improvement is observed, indicating that the optimal region of the hyperparameter space has been efficiently identified. The subsequent trials mainly confirm the robustness of this solution with objective values remaining tightly clustered around the optimum.

Table C.1: Best Lasso Hyperparameter

Parameter	value
Lasso_alpha	0.001114640363052249

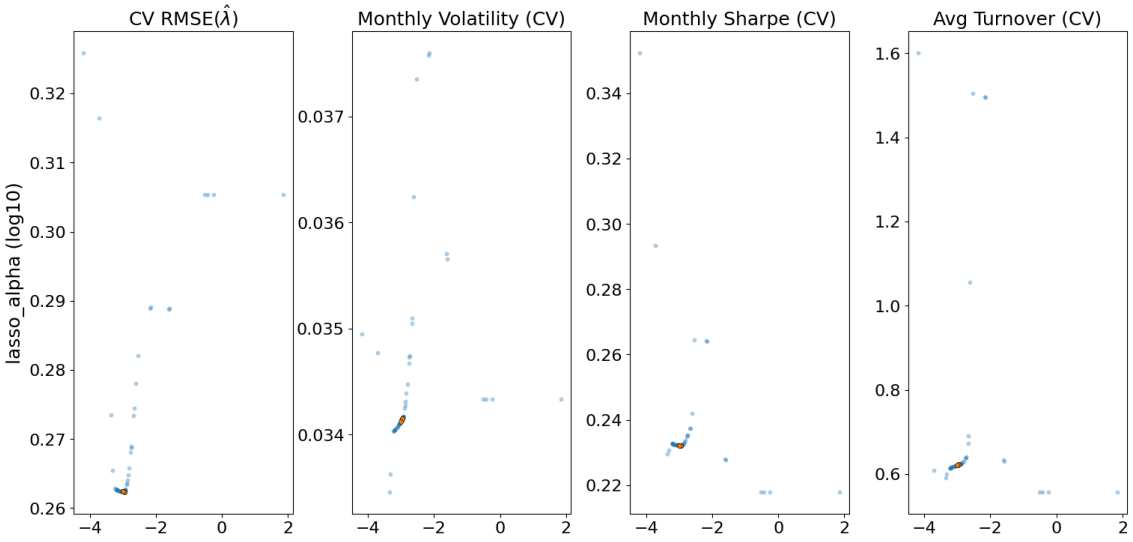


Figure C.2: Lasso: Hyperparameters vs metrics

The hyperparameter sensitivity analysis plots performance as a function of the Lasso regularisation parameter, where the x-axis represents $\log_{10}(\alpha)$ and the y-axis corresponds to the metric of interest in each panel (CV RMSE, monthly volatility, monthly Sharpe ratio and turnover). The parameter α controls the strength of the ℓ_1 penalty, with higher values inducing stronger sparsity in the estimated coefficients. The CV RMSE exhibits a clear minimum around $\log_{10}(\alpha) \in [-3, -2.5]$, decreasing from approximately 0.31 – 0.32 for very low values of α to about 0.262 at the optimum, indicating an important reduction in estimation error.

This region is consistent across economic metrics: portfolio volatility declines from roughly 3.7% to 3.4% while turnover decreases remarkably from above 1.5 to around 0.6 – 0.7, reflecting improved portfolio stability. The Sharpe ratio remains relatively stable around 0.22 – 0.24, suggesting that gains are primarily driven by risk reduction rather than higher returns.

The selected value $\alpha = 0.0011146$ (i.e., $\log_{10}(\alpha) \approx -2.95$) lies within this optimal region, con-

firming that moderate regularisation achieves an effective balance between sparsity, estimation accuracy and portfolio stability.

C.2 Ridge

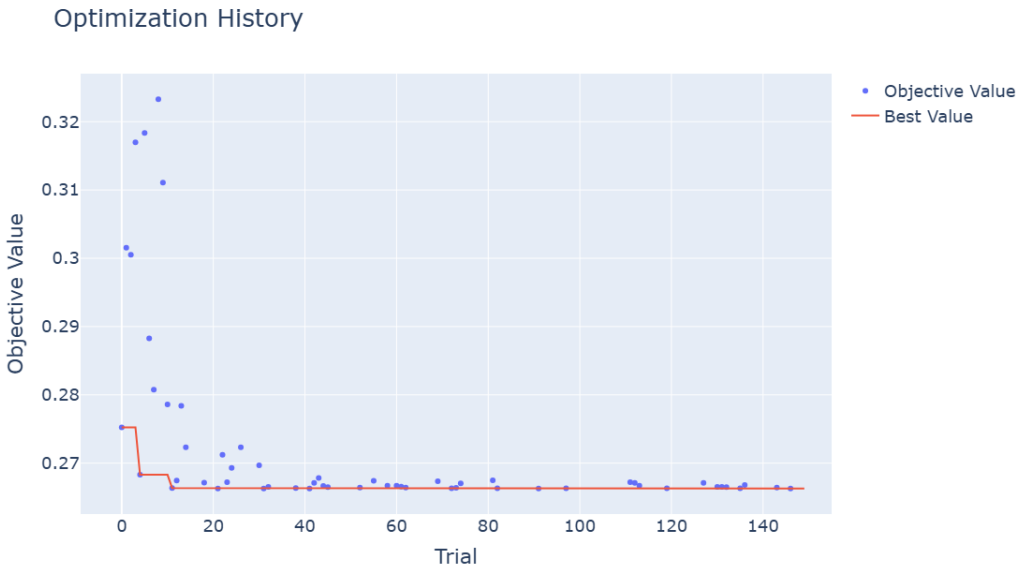


Figure C.3: Ridge: Optimisation history

The optimisation history exhibits a rapid decrease in the objective value during the initial iterations, indicating an efficient exploration of the hyperparameter space. Convergence is reached after 10 – 15 trials, beyond which the best objective value stabilises and no further improvement is observed. The subsequent trials yield values tightly clustered around the optimum, suggesting that the solution is robust and that the optimisation procedure has successfully identified a well-defined optimal region for the Ridge regularisation parameter.

Table C.2: Best Ridge Hyperparameter

Parameter	value
Ridge_alpha	4.853423020255109

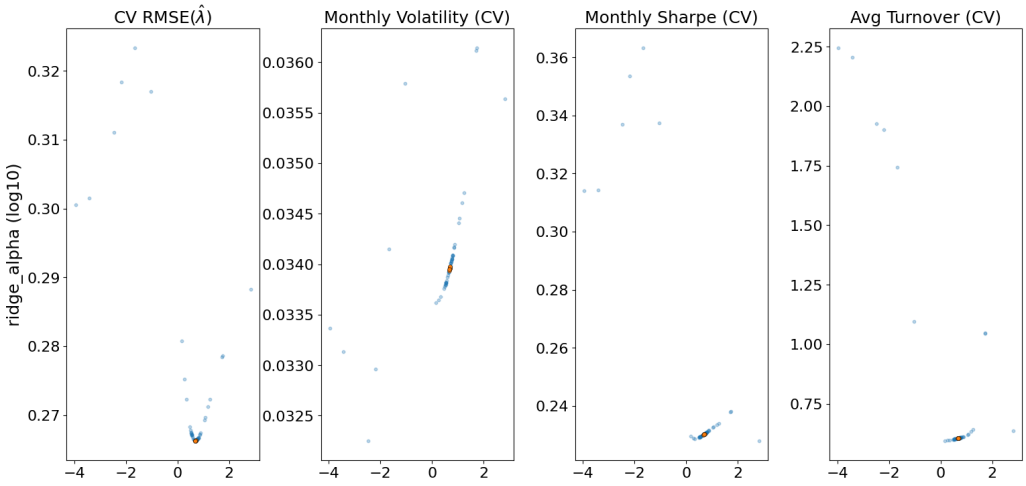


Figure C.4: Ridge: Hyperparameters vs metrics

The hyperparameter sensitivity analysis displays performance as a function of the Ridge regularisation parameter. The parameter α controls the strength of the ℓ_2 penalty, with higher values inducing stronger shrinkage of the coefficients toward zero. The CV RMSE exhibits a clear minimum around $\log_{10}(\alpha) \in [0.5, 1]$, decreasing from approximately 0.32 for low values of α to about 0.265 at the optimum, indicating a substantial reduction in estimation error. This region is consistent across economic metrics: portfolio volatility declines from roughly 3.6% to 3.4% while turnover drops significantly from above 2 to around $0.6 - 0.7$, reflecting improved stability. The Sharpe ratio remains relatively stable around $0.22 - 0.24$, suggesting that performance gains primarily originate from risk reduction rather than return enhancement.

The selected value $\alpha = 4.85$ (i.e., $\log_{10}(\alpha) \approx 0.69$) lies within this optimal region, confirming that strong regularisation materially improves both statistical accuracy and portfolio stability.

C.3 Elastic Net

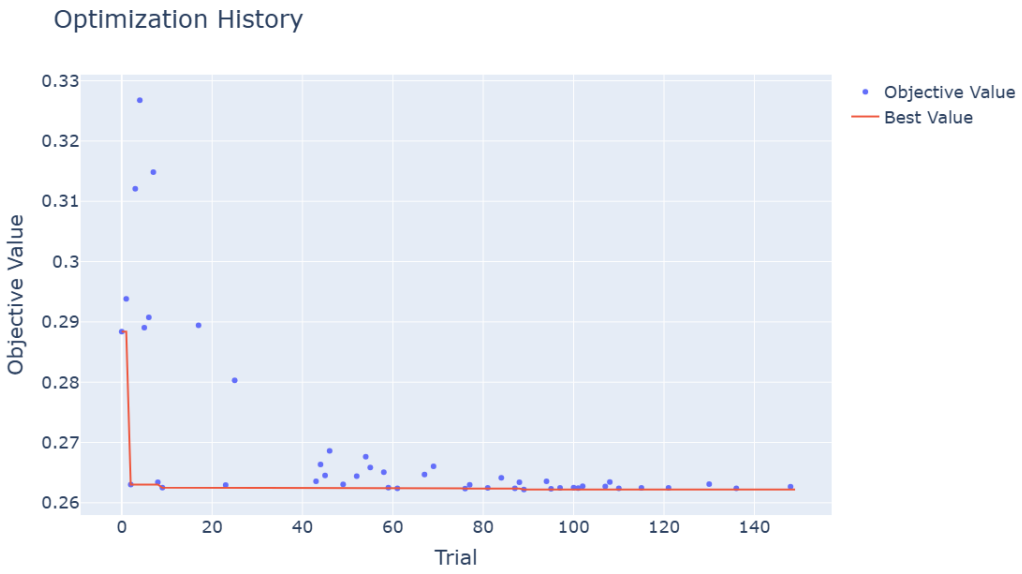


Figure C.5: Elastic Net: Hyperparameters vs metrics

The optimisation history shows a rapid decrease in the objective value during the initial iterations, with the objective of falling from approximately 0.29 to 0.263 within the first 10 trials. Convergence is reached shortly thereafter, as indicated by the stabilisation of the best objective value. Subsequent trials yield values tightly clustered around the optimum ($\sim 0.262 - 0.264$), suggesting that the hyperparameter space has been efficiently explored. This pattern indicates a well-defined and robust optimum for the Elastic Net model, with limited gains from additional exploration beyond the initial phase.

Table C.3: Best Elastic Net Hyperparameters

Parameter	value
en_alpha	0.0010588665658071827
en_l1_ratio	0.9703912349655303
en_max_iter	4079
en_tol	5.475393837052734e-05

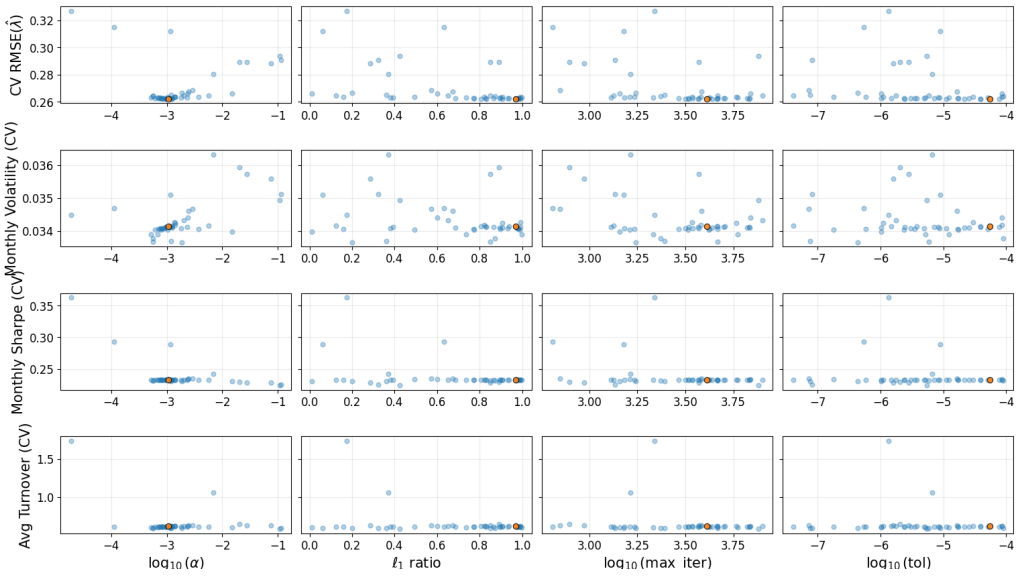


Figure C.6: Elastic Net: hyperparameters vs metrics

The hyperparameter sensitivity analysis evaluates performance across multiple Elastic Net parameters. The parameter α controls the overall strength of regularisation, while the ℓ_1-ratio governs the balance between ℓ_1 (sparsity) and ℓ_2 (shrinkage) penalties. In contrast, $\log_{10}(\texttt{max\_iter})$ and $\log_{10}(\texttt{tol})$, which control solver convergence, exhibit no clear impact on performance, indicating that results are largely insensitive to numerical optimisation settings.

The CV RMSE is minimised around $\log_{10}(\alpha) \approx -3$, decreasing from approximately 0.32 to 0.263, indicating substantial gains in estimation accuracy. This region is associated with improved economic outcomes, with volatility stabilising around 3.4%, turnover reduced to $\sim 0.6{-}0.7$ and Sharpe ratios, remaining stable near 0.23. The optimal configurations consistently favor high ℓ_1-ratios ($\approx 0.9-1$), indicating that a predominantly Lasso-like structure is preferred, while the influence of max iterations and tolerance appears limited within the explored range.

The selected hyperparameters ($\alpha \approx 0.00106$, ℓ_1-ratio ≈ 0.97) lie within this optimal region, confirming that strong regularisation combined with a predominantly sparse structure yields robust estimation and stable portfolio performance.

C.4 Random Forest

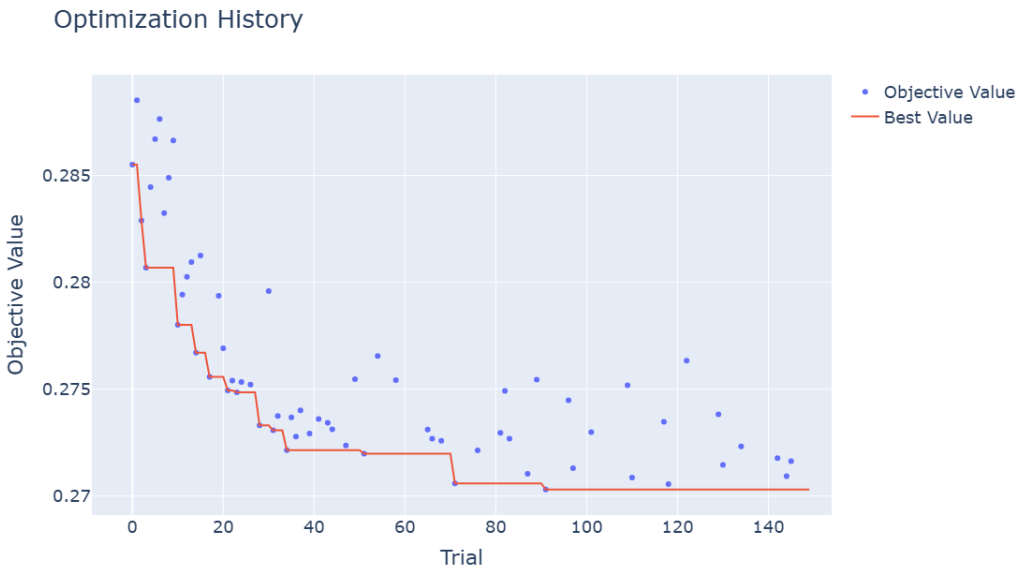


Figure C.7: Random Forest: Optimisation history

The optimisation history shows a more gradual decrease in the objective value compared to linear models, with the objective declining from approximatively 0.285 to 0.270 over the course of the trials. While early iterations yield noticeable improvements, convergence is reached later, around 70 – 90 trials, after which the best value stabilises. The dispersion of objective value remains relatively high throughout, reflecting a more complex and irregular hyperparameter space. This pattern suggests that Random Forest requires more extensive exploration to identify an optimal configuration, with incremental gains achieved over a large number of trials.

Table C.4: Best Random Forest hyperparameters

Parameter	value
n_estimators	1749
max_depth	11
min_samples_split	9
min_samples_leaf	1
max_features	0.3

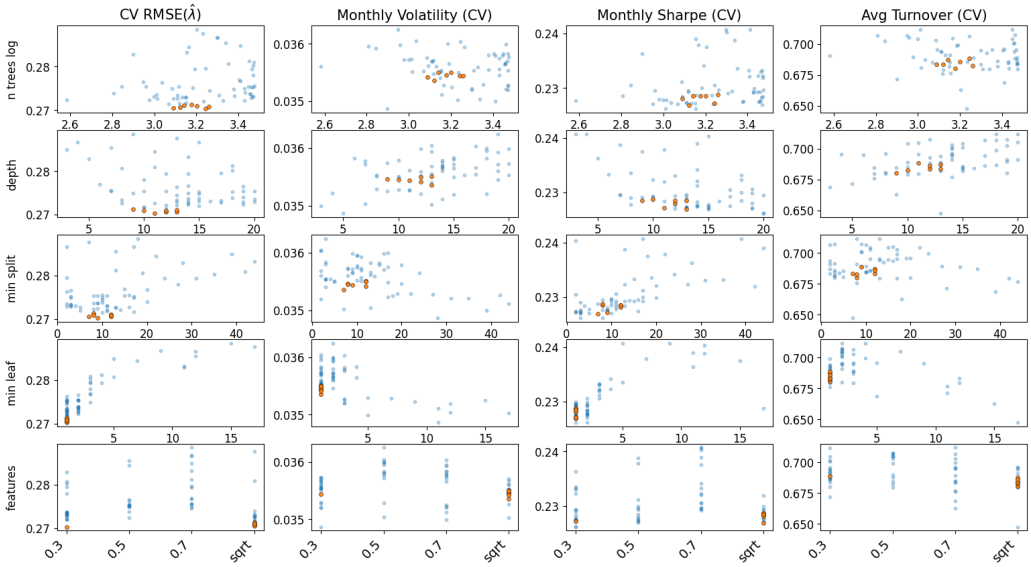


Figure C.8: Random Forest: Hyperparameters vs metrics

The hyperparamter sensitivity analysis evaluates performance across Random Forest hyperparameters such as n trees log (the logarithm of the number of trees), min_leaf (minimum samples per leaf), depth (maximum tree depth), features (fraction of features considered at each split) and min_split (minimum samples required to split a node). The CV RMSE decreases to approximately 0.285 to 0.270, with the best configurations concentrated around n_estimators $\approx 1000 - 1900$ ($\log_{10}(3) - \log_{10}(3.4)$) and depth $\approx 10 - 12$, indicating that sufficiently deep trees and a large ensemble improve predictive accuracy.

This region is consistent across economic metrics: portfolio volatility stabilises around 3.5%, turnover remains moderate at $\sim 0.68 - 0.7\%$ and Sharpe ratios are relatively stable around $0.23 - 0.24$. Smaller values of min_split and min_samples_leaf are generally preferred, suggesting that more flexible trees enhance performance, while features has a limited impact within the tested range. The selected configuration (n_estimators = 1749, depth = 11, min_split = 9, min_leaf = 1, features = 0.3) lies within this optimal region, confirming that a large ensemble of moderately deep and flexible trees yields robust estimation and stable portfolio performance.

C.5 XGBoost

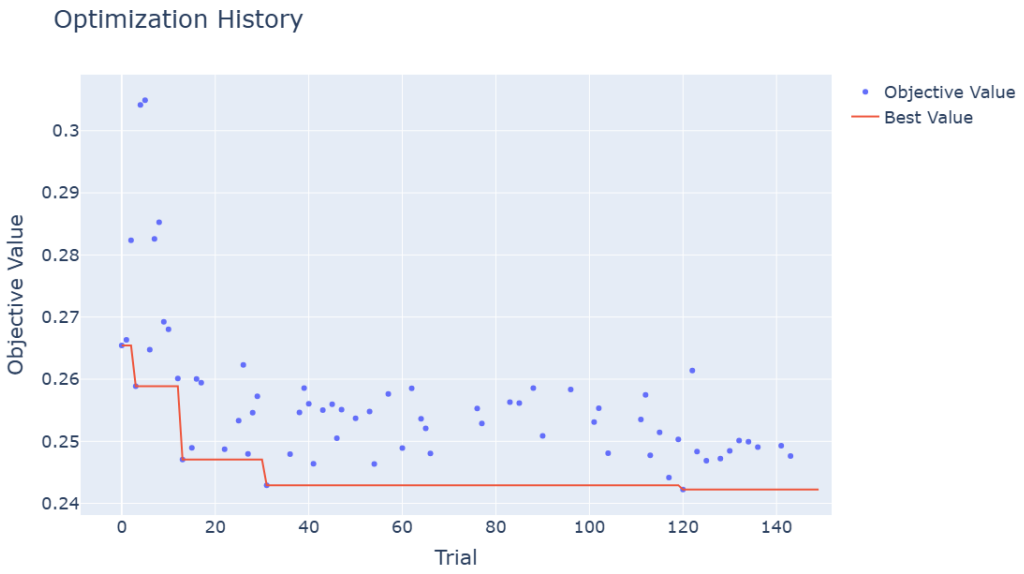


Figure C.9: XGBoost: Optimisation history

The optimisation history shows a progressive decrease in the objective value, from approximately 0.305 to 0.242, indicating substantial improvement in model performance. While the initial iterations yield rapid gains, further improvements are achieved more gradually, with convergence reached around $100 - 120$ trials. The continued decline in the best objective value over a large number of trials reflects a complex and high-dimensional hyperparameter space, requiring extensive exploration. The dispersion of objective values remains relatively high throughout, suggesting sensitivity to hyperparameter choices. Overall, the results indicate that XGBoost benefits from deeper and more sustained optimisation, with incremental improvements leading to a well-defined and robust optimum.

Table C.5: Best XGBoost hyperparameters

Parameter	value
eta	0.22718979486243887
max_depth	5
min_child_weight	3.1024869502478873
subsample	0.6362547502936644
colsample_bytree	0.5618755265954234
gamma	1.3539908401930289e-05
lambda	0.008212347291226548
alpha	0.0016601929287760723

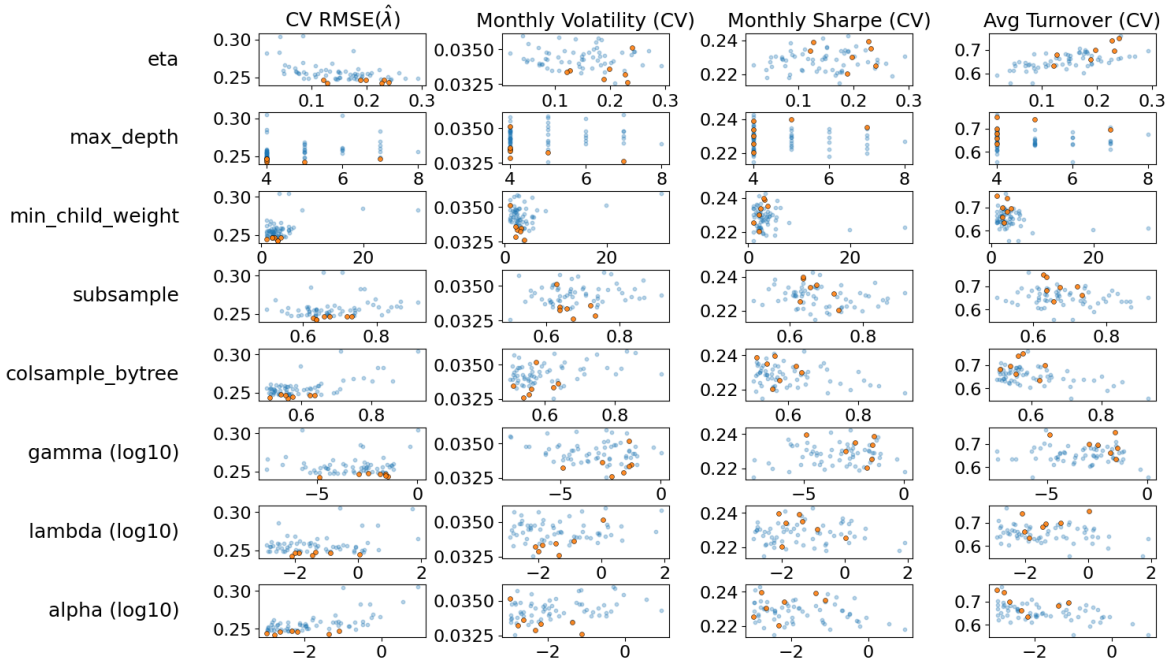


Figure C.10: XGBoost: Hyperparameters vs metrics

The hyperparameter sensitivity analysis evaluates performance across XGBoost parameters such as the learning rate η, tree depth (max_depth), minimum child weight, subsampling ratios (subsample, colsample_bytree), and regularization parameters (γ, λ, and α). The CV RMSE decreases from approximately 0.30 to 0.242, indicating substantial improvements in estimation accuracy. The best configurations are concentrated around moderate learning rates ($\eta \approx 0.2$) and shallow trees (max_depth $\approx$ 4–5), suggesting that controlled model complexity enhances generalization.

This region is consistent across economic metrics: portfolio volatility stabilizes around 3.3–3.4%, while turnover remains moderate at $\sim$ 0.65–0.70, and Sharpe ratios are relatively stable near 0.23–0.24. Subsampling parameters (subsample $\approx$ 0.6–0.7, colsample_bytree $\approx$ 0.55–0.65) further improve robustness by reducing overfitting, while regularization parameters (γ, λ, α) exhibit a limited but stabilizing effect within moderate ranges.

The selected configuration ($\eta \approx 0.23$, max_depth = 5, min_child_weight $\approx$ 3.1, subsample $\approx$ 0.64, colsample_bytree $\approx$ 0.56) lies within this optimal region, confirming that controlled tree complexity combined with subsampling and moderate regularization yields robust estimation and stable portfolio performance.

C.6 PGA

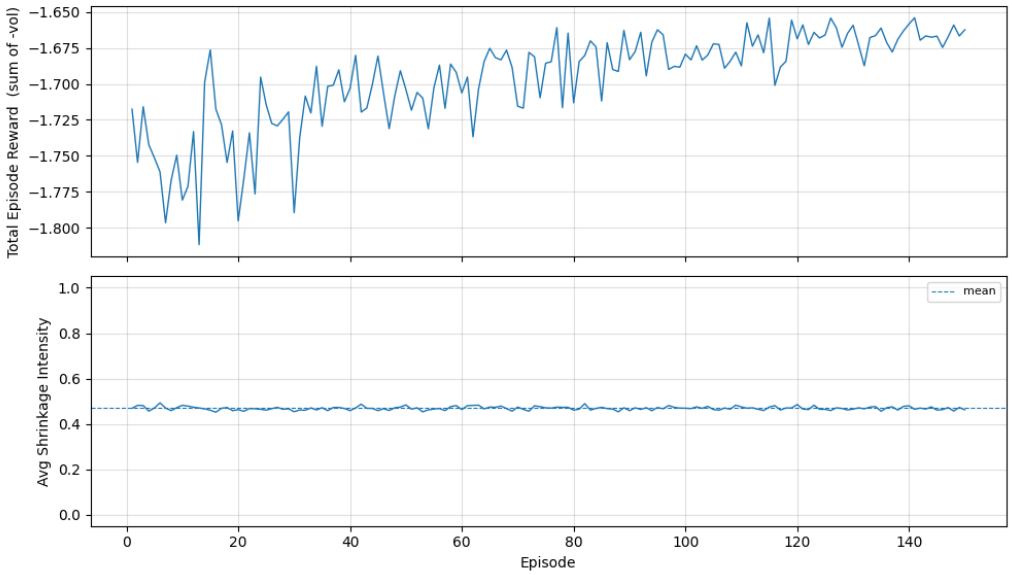


Figure C.11: PGA training

The training dynamics of the PGA model show a gradual improvement in performance over episodes, with the total reward increasing from approximately -1.8 to -1.66, indicating effective learning of the shrinkage policy. While early episodes exhibit substantial variability, the reward stabilises progressively, suggesting convergence toward consistent strategy.

The average shrinkage intensity remains stable throughout training, fluctuating narrowly around 0.46 – 0.48, with limited dispersion. This indicates that the agent converges to a well-defined and robust shrinkage level, with no evidence of excessive exploration or instability in the policy. Overall, the results suggest stable convergence and consistent policy learning in the PGA framework.

For reproducibility of the results displayed in the master thesis, the value of the hyperparameters used for the PGA and MLP strategy are indicated in the table below C.6.

Table C.6: Hyperparameters used in the PGA specification

Category	**Hyperparameter**	**Value**
Policy distribution	Initial exploration noise $\sigma_{\varepsilon,\text{init}}$	0.20
Policy distribution	Minimum exploration noise $\sigma_{\varepsilon,\text{min}}$	0.03
Optimizer (Adam)	Initial learning rate lr_{init}	10^{-3}
Optimizer (Adam)	Minimum learning rate lr_{min}	10^{-4}
Optimizer (Adam)	Adam β_1	0.90
Optimizer (Adam)	Adam β_2	0.99
Training	Maximum episodes	150

Category	Hyperparameter	Value
Training	Discount factor γ	0.80
Training	Decay factor	0.995
Training	Gradient clipping max norm	1.0
Training	Device	CPU
Network architecture	Hidden layer 1 width	256
Network architecture	Hidden layer 2 width	128
Network architecture	Hidden layer 3 width	32
Network architecture	Output layer width	1
Network architecture	Hidden activations	ReLU
Network architecture	Output activation	Sigmoid
Network architecture	Normalization	LayerNorm after first hidden layer
Network architecture	Weight initialization	Xavier uniform
Network architecture	Bias initialization	Zeros

C.7 RPGA

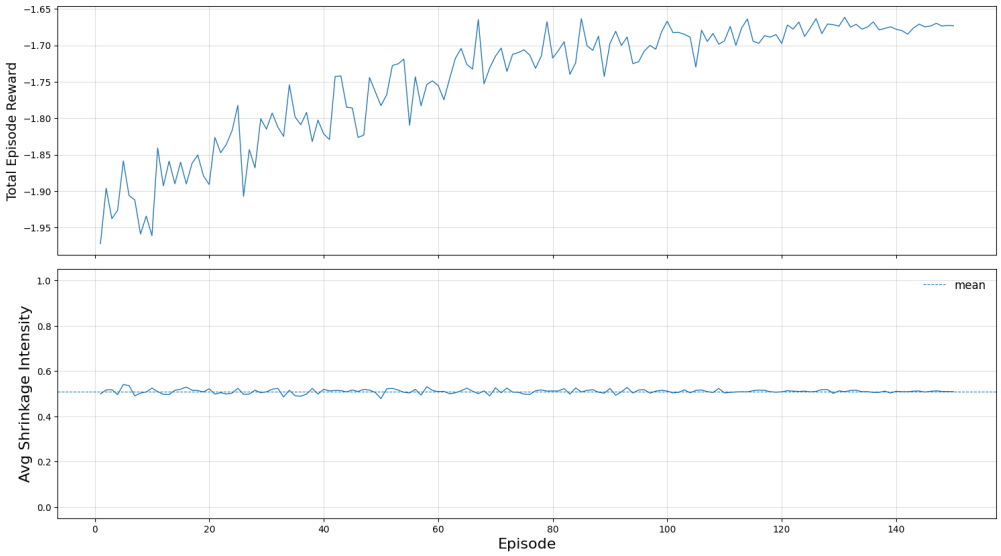


Figure C.12: RPGA training

The training dynamics of the RPGA model exhibit a clear steady improvement in performance, with the total reward increasing from approximately -1.95 to -1.66, indicating effective learning of the dynamic shrinkage policy. Compared to the PGA, the convergence is more gradual but smoother, with reduced volatility in later episodes, suggesting enhance stability due to the recurrent structure.

The average shrinkage intensity remains highly stable throughout training, fluctuating narrowly around 0.5 – 0.52, with minimal dispersion. This indicates that the agent converges to a

consistent and robust shrinkage level, while incorporating temporal dependencies. Overall, the results highlight stable convergence and improved learning dynamics in the RPGA framework.

Finally, for reproducibility of the results displayed in the master thesis, the value of the hyperparameters used for the RPGA and GRU strategy are indicated in the table below C.7.

Table C.7: Hyperparameters used in the RPGA specification

Category	**Hyperparameter**	**Value**
GRU architecture	Hidden layer 1 width	32
GRU architecture	Hidden layer 2 width	16
GRU architecture	Number of GRU layers	2
GRU architecture	GRU activation	Tanh (default GRU cell)
GRU architecture	Output layer width	1
GRU architecture	Output activation	Sigmoid
GRU architecture	Weight initialization	Xavier uniform
GRU architecture	Bias initialization	Zeros
Policy distribution	Initial exploration noise σ_{init}	0.30
Policy distribution	Minimum exploration noise σ_{min}	0.03
Policy distribution	Sigma decay schedule	Linear
Optimizer (Adam)	Learning rate	3×10^{-4}
Optimizer (Adam)	Adam β_1	0.88
Optimizer (Adam)	Adam β_2	0.97
Training	Maximum episodes	150
Training	Discount factor γ	0.92
Training	Gradient clipping max norm	1.0
Training	Random seed	42
Training	Device	CPU

Abstract :

This master thesis investigates whether machine learning methods can improve covariance matrix estimation for global minimum variance portfolios by learning time-varying shrinkage intensities. In high-dimensional portfolio problems, the sample covariance matrix is often unstable, which can lead to extreme portfolio weights and poor out-of-sample risk control. Classical shrinkage estimators address this issue by combining the sample covariance matrix with a structured target, but they generally rely on a shrinkage intensity that is static or derived under stable distributional assumptions. This thesis instead studies shrinkage intensity as a state-dependent parameter that may vary with market conditions.

The empirical framework compares supervised learning and reinforcement learning approaches. Supervised models are trained to approximate an ex-post oracle shrinkage intensity using features extracted from rolling estimation windows, including volatility, correlation, eigenvalue and covariance-stability indicators. Reinforcement learning models treat shrinkage selection as a sequential decision problem and learn policies directly from realised portfolio outcomes. Two policy-gradient architectures are implemented: a feedforward policy gradient agent and a recurrent policy gradient agent based on a gated recurrent unit.

The results show that the oracle shrinkage intensity varies substantially over time and is linked to observable market regimes, supporting the idea that covariance shrinkage should be modelled dynamically. The oracle is also partly predictable, with nonlinear supervised models, especially tree-based methods, capturing more informative patterns than regularised linear models. In portfolio terms, learning-based shrinkage policies improve upon the sample covariance benchmark in the baseline setting, with the strongest gains coming from nonlinear models and, in particular, the recurrent reinforcement learning specification. However, robustness checks show that these improvements are not unconditional: performance depends on the estimation window, the dataset and the modelling framework.

Overall, the thesis shows that supervised learning and reinforcement learning can learn economically meaningful dynamic shrinkage policies for minimum-variance portfolio construction. The main contribution is to bridge classical covariance shrinkage with data-driven decision rules, showing that adaptive regularisation can improve portfolio stability when market conditions change.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management
Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve
Boulevard Emile Devreux 6, 6000 Charleroi, Belgique
Chaussée de Binche 151, 7000 Mons, Belgique
www.uclouvain.be/lsm