SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Portfolio Optimization with Gaussian Process Regression

Xiyue ZHANG

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY —
INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Portfolio Optimization with Gaussian Process Regression

Portfolio-Optimierung mit Gauß-Prozess Regression

Author:	Xiyue ZHANG
Supervisor:	Prof. Dr. Hans-Jochim Bungartz
Advisor:	Kislaya Ravi
Submission Date:	December 3rd

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, December 3rd Xiyue ZHANG

Acknowledgments

I would like to express my deepest gratitude to my advisor, Kislaya Ravi, for the invaluable guidance, support, and encouragement throughout the course of this thesis project. Ravi's expertise, insights, and unwavering commitment to excellence have been instrumental in shaping this work. His mentorship has not only guided me through the complexities of this research but has also inspired me to think critically and approach challenges with determination. This thesis would not have been possible without Ravi's patience, constructive feedback, and constant encouragement, which motivated me to strive for improvement at every step. Additionally, I would like to extend my early congratulations and best wishes to Ravi for his upcoming marriage next February. I wish him and his partner a lifetime of happiness, love, and success.

I am also profoundly grateful to my parents for the time and effort they have always dedicated to supporting me, both physically and spiritually. Their unwavering love, encouragement, and belief in me have been a constant source of strength and inspiration. I am truly blessed to have such loving and supportive parents who have always encouraged me to pursue my dreams and never give up, even during the hardest times.

Lastly, I would like to extend my heartfelt thanks to Dr. Tian Weikun, who has provided invaluable assistance during the writing of this thesis. Thank you for believing in me and for your unwavering support throughout this journey.

Abstract

This thesis explores the integration of Gaussian Processes Regression (GPR) into portfolio optimization, presenting a novel framework that leverages predictive modeling to enhance investment performance. GPR, a non-parametric Bayesian approach, is employed to forecast asset returns and associated uncertainties, addressing key challenges in financial forecasting such as non-linearity, non-stationarity, and data noise. The study develops a Dynamic Strategy that incorporates these predictions into portfolio allocation, allowing for adaptive decision-making based on probabilistic thresholds.

The methodology includes comprehensive data preprocessing, model development, and backtesting over historical market data, encompassing diverse assets across major sectors. Four portfolio strategies—Maximum Return, Minimum Volatility, Maximum Sharpe Ratio, and the proposed Dynamic Strategy—are evaluated on performance metrics including total returns, volatility, Sharpe ratios, and transaction costs. The results demonstrate that the Dynamic Strategy consistently outperforms traditional approaches, achieving higher risk-adjusted returns while minimizing transaction costs through intelligent rebalancing.

This research contributes to the field of predictive portfolio optimization by introducing a robust framework that integrates advanced forecasting with adaptive asset allocation. Practical implications for portfolio managers include improved risk management, reduced rebalancing costs, and enhanced decision-making in dynamic markets. Additionally, the thesis identifies opportunities for future research, such as refining GPR models, expanding strategy applications, and exploring scalability for large portfolios. These findings highlight the transformative potential of machine learning in modern financial decision-making.

Contents

1 Introduction

1.1 Background and Motivation

Portfolio optimization has been a cornerstone of financial decision-making since its formal introduction in the mid-20th century. The seminal work of Harry Markowitz in 1952 laid the foundation for Modern Portfolio Theory (MPT), which introduced the concepts of mean-variance optimization and the efficient frontier. This work transformed the way investors approached asset allocation by providing a rigorous mathematical framework to balance risk and return. Over the decades, MPT evolved through the development of models such as the Capital Asset Pricing Model (CAPM), single-index models, and risk-parity approaches. These models addressed various practical challenges, such as the estimation of covariance matrices and the inclusion of downside risk measures.

However, traditional portfolio optimization methods often rely on strong assumptions, such as the normality of asset returns and the stability of their relationships over time. In reality, financial markets are highly complex, characterized by non-linear interactions, non-stationary behaviors, and abrupt regime changes. The limitations of classical methods in capturing these dynamics have spurred the integration of advanced computational techniques, including Machine Learning (ML), into the field of portfolio management.

Machine learning, with its ability to model non-linear patterns and process large datasets, has emerged as a powerful tool for financial forecasting. Gaussian Process Regression (GPR), a Bayesian non-parametric approach, has gained attention for its probabilistic nature and flexibility in modeling uncertainty. These attributes make GPR particularly well-suited for predicting asset returns and volatilities, critical inputs for portfolio optimization.

Despite the advancements in ML and the promise of techniques like GPR, significant challenges remain. Accurately forecasting asset returns is notoriously difficult due to the inherent volatility and unpredictability of financial markets. Traditional methods of estimating parameters, such as volatility and correlations, often fall short during periods of market stress, where accurate predictions are most needed. This study aims to bridge these gaps by integrating GPR into a dynamic portfolio optimization framework, leveraging its predictive strengths to enhance decision-making in uncertain and rapidly changing markets.

1.2 Research Objectives

The primary objective of this research is to develop and evaluate a portfolio optimization framework that combines the predictive capabilities of GPR with strategic asset allocation. The study focuses on improving investment performance by addressing the shortcomings of traditional methods and integrating probabilistic predictions into decision-making. Specifically, the research aims to:

Construct GPR models capable of forecasting the returns and volatilities of selected assets with a high degree of accuracy. The models will be updated iteratively to incorporate new data and adapt to market changes dynamically.

Design portfolio optimization strategies that utilize GPR outputs, including traditional approaches like Maximum Return, Minimum Volatility, and Maximum Sharpe Ratio, as well as a novel Dynamic Strategy. The latter incorporates GPR's probabilistic forecasts to determine optimal reallocation decisions based on threshold probabilities.

Implement a comprehensive backtesting framework to evaluate the strategies' performance over historical market data, accounting for transaction costs and realistic market conditions.

Demonstrate the practical applicability of the Dynamic Strategy by analyzing its risk-return trade-offs, adaptability, and cost efficiency compared to traditional strategies.

Through these objectives, the study seeks to contribute to the field of predictive portfolio optimization by presenting a robust and practical approach to integrating advanced forecasting methods into asset management.

1.2.1 Role of machine learning in financial forecasting

ML has gained significant traction in financial markets due to its ability to analyze vast amounts of data and identify complex patterns that traditional models may overlook. In particular, GPR has emerged as a powerful tool for time-series forecasting, offering a flexible framework for capturing non-linear relationships and uncertainty in predictions.

1.2.2 Challenges in time-series forecasting and traditional portfolio optimization methods

Despite the advancements in ML techniques, predicting asset returns remains a challenging task due to the inherent volatility and non-stationarity of financial markets. Moreover, traditional portfolio optimization methods, while theoretically elegant, often face significant practical challenges in implementation. These challenges primarily stem from the difficulty in accurately estimating input parameters and the inherent uncertainty in financial time-series forecasting. This section examines these challenges and introduces how GPR provides a novel approach to addressing them.

Parameter Estimation Challenges in Modern Portfolio Theory Modern Portfolio Theory (MPT), while foundational in the field of finance, is critically dependent on the accurate estimation of several key parameters, making its practical application challenging. Among these, volatility estimation stands out as the most pivotal yet problematic aspect of portfolio optimization. Traditional methods for estimating volatility typically rely on historical data under the assumption that past patterns will persist into the future. However, this assumption is often flawed, as financial markets are inherently dynamic and subject to frequent regime changes, fluctuating volatility clusters, and intricate interdependencies. These characteristics render historical volatility estimates backward-looking and overly sensitive to the chosen estimation window, potentially resulting in misleading inputs for portfolio construction. Furthermore, the presence of heteroskedasticity in financial time series complicates volatility estimation, as periods of stable and erratic market behavior frequently alternate.

The challenges of parameter estimation in MPT extend beyond volatility. Expected returns, another critical input, are notoriously difficult to predict with accuracy. They are highly sensitive to estimation errors, which can significantly skew portfolio outcomes. Additionally, expected returns exhibit a time-varying nature, influenced by changing market regimes and economic conditions. This variability further complicates reliable estimation and introduces additional layers of uncertainty into the optimization process.

Another significant hurdle in MPT is accurately capturing the correlation structure among assets. Correlations are dynamic and often shift dramatically during periods of market stress, precisely when understanding interdependencies is most crucial. Relying on static historical correlations can lead to underestimation of risks and suboptimal diversification. Moreover, as the number of assets in a portfolio increases, the complexity of estimating a stable and meaningful correlation matrix grows quadratically, creating what is commonly referred to as the "curse of dimensionality." This issue is compounded by computational challenges in managing high-dimensional datasets, further straining the practical applicability of MPT.

These challenges collectively highlight the limitations of traditional approaches in addressing the dynamic and complex nature of financial markets. Volatility, expected returns, and correlation structures, while central to the theoretical framework of MPT, require sophisticated methods and adaptive modeling techniques to ensure their reliability and relevance in real-world applications. Addressing these limitations is critical for advancing portfolio optimization practices and improving their alignment with the realities of modern financial systems.

Limitations of Traditional Forecasting Methods Traditional forecasting approaches in finance have predominantly relied on methods that provide point estimates, failing to capture the inherent uncertainty in financial predictions. These methods often make strong assumptions about the underlying data distribution and struggle to adapt to the non-linear, non-stationary nature of financial time series. AutoRegressive Integrated Moving Average (ARIMA) models, exponential smoothing, and other classical approaches, while mathematically tractable, often fall short in capturing the complex dynamics of financial markets.

A fundamental limitation of these traditional approaches is their rigidity in handling uncertainty. Point forecasts, even when accompanied by confidence intervals based on historical volatility, fail to capture the dynamic nature of prediction uncertainty. This limitation becomes particularly problematic in portfolio optimization, where understanding the reliability of forecasts is as important as the forecasts themselves.

Conventional approaches to financial time-series forecasting exhibit several limitations:

Traditional methods often provide single-point forecasts, which lack uncertainty quantification and have limited ability to capture prediction confidence. This results in insufficient information for risk management.

Model rigidity is another issue, as it involves the assumption of specific probability distributions, difficulty in capturing non-linear relationships, limited adaptation to changing market conditions, and oversimplification of complex market dynamics.

Data requirements pose additional challenges, including the need for large historical datasets, sensitivity to outliers and noise, the challenge of incorporating multiple data sources, and difficulty in handling missing data.

Advantages of Gaussian Process Regression Our research proposes GPR as a solution to these challenges, offering several key advantages:

1. **Probabilistic Framework**
    - Natural uncertainty quantification
    - Automatic volatility estimation through posterior variance
    - Capture of prediction confidence intervals
    - Robust handling of noise in financial data
2. **Flexible Modeling**

- Non-parametric approach avoiding distributional assumptions
- Ability to capture complex non-linear relationships
- Automatic complexity adjustment through kernel selection
- Incorporation of prior knowledge through kernel design

3. **Parameter Estimation**
   - Direct modeling of volatility through posterior variance
   - Joint estimation of returns and risk
   - Principled handling of uncertainty
   - Adaptive to changing market conditions

Addressing Traditional Limitations Our GPR-based approach specifically addresses the key limitations of MPT:

$$\sigma^2_{GPR}(x_*) = k(x_*, x_*) - k(x_*, X)[K(X, X) + \sigma_n^2 I]^{-1} k(X, x_*) \tag{1.1}$$

Where $\sigma^2_{GPR}(x_*)$ represents the posterior variance at prediction point x_*, providing a direct estimate of volatility that:

- Naturally accounts for uncertainty in predictions
- Adapts to local data density and quality
- Provides time-varying volatility estimates
- Incorporates both local and global market information

The GPR approach offers several fundamental advantages over traditional methods. Unlike historical volatility estimates that require arbitrary window selection, GPR's volatility estimates emerge naturally from the probabilistic learning process. The method adapts automatically to different market regimes through its kernel function, which can capture both long-term trends and short-term fluctuations in market behavior.

Furthermore, GPR's non-parametric nature frees it from restrictive assumptions about return distributions. The method can capture complex, non-linear patterns in the data while maintaining the ability to quantify uncertainty in its predictions. This combination of flexibility and uncertainty awareness makes it particularly well-suited for financial applications where both accuracy and risk assessment are crucial.

Implications for Portfolio Optimization The GPR framework transforms the traditional portfolio optimization problem by: The integration of GPR into portfolio optimization transforms the traditional MPT framework by providing more reliable and dynamic parameter estimates. By directly modeling the uncertainty in our predictions, we can make more informed portfolio allocation decisions that account for both expected returns and our confidence in those expectations. This approach naturally leads to more robust portfolios that adapt to changing market conditions while maintaining a principled approach to risk management.

The significance of this advancement cannot be overstated. By addressing one of the fundamental criticisms of MPT - the difficulty of accurately estimating volatility - our GPR-based approach bridges the gap between theoretical elegance and practical applicability. This enhancement makes MPT more reliable and useful for real-world portfolio management, where accurate risk assessment is crucial for maintaining stable, long-term investment performance.

1.3 Thesis Structure

The thesis is organized as follows: Chapter 2 reviews the relevant literature, covering portfolio optimization theories, Gaussian Process Regression, and the integration of machine learning in finance. Chapter 3 outlines the methodology, detailing the data collection, model development, and strategy design processes. Chapter 4 presents the results of the backtesting framework, comparing the performance of the proposed strategies. Chapter 5 discusses the implications of the findings, addressing practical applications and limitations, while also concludes with a summary of key insights and suggestions for future research.

2 Previous Literature

2.1 Theoretical Framework

This chapter discusses the fundamental theories underpinning the study, focusing on portfolio optimization and GPR. Understanding these theories is essential for developing the predictive portfolio optimization framework proposed in this research.

2.1.1 Portfolio Optimization Theory

Portfolio optimization is a cornerstone of modern finance, aiming to allocate assets in a way that balances expected returns against risk. The foundational theory in this domain is the MPT, introduced by Harry Markowitz in 1952 [20].

Modern Portfolio Theory (MPT)

MPT posits that investors can construct an optimal portfolio that offers the maximum expected return for a given level of risk or, equivalently, the minimum risk for a given level of expected return. The key assumptions of MPT are:

- Investors are rational and risk-averse, preferring higher returns and lower risk.
- Markets are efficient, and all investors have access to the same information.
- Asset returns are normally distributed and can be described by their mean (expected return) and variance (risk).

Expected Return and Risk The expected return of a portfolio, $E[R_p]$, is the weighted sum of the expected returns of the individual assets:

$$E[R_p] = \sum_{i=1}^{n} w_i E[R_i], \tag{2.1}$$

where w_i is the weight of asset i in the portfolio, $E[R_i]$ is the expected return of asset i, and n is the total number of assets.

The portfolio variance, σ_p^2, representing risk, is given by:

$$\sigma_p^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \sigma_{ij}, \tag{2.2}$$

where σ_{ij} is the covariance between asset i and asset j. The standard deviation σ_p is the square root of the variance.

Efficient Frontier The set of optimal portfolios that offer the highest expected return for a given level of risk forms the *Efficient Frontier*. Portfolios on the efficient frontier are considered optimal, as no other portfolios offer higher returns for the same risk level.

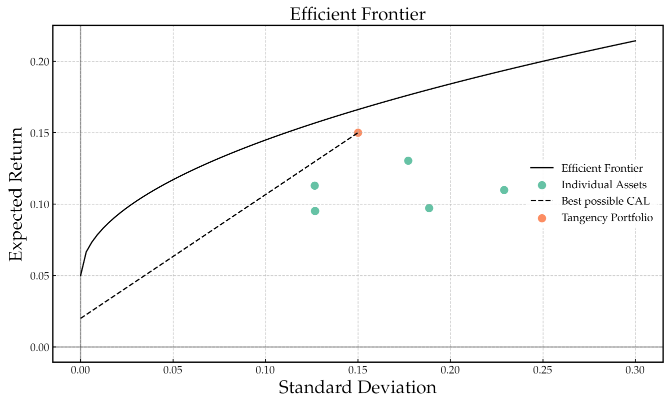


Figure 2.1: Efficient Frontier in Mean-Variance Space

Mean-Variance Optimization Mean-variance optimization involves solving for the portfolio weights that minimize the portfolio variance for a given expected return. The optimization problem can be formulated as:

$$\begin{aligned}
\min_{\mathbf{w}} \quad & \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} \\
\text{subject to} \quad & \mathbf{w}^T \mathbf{E} = E_p, \\
& \sum_{i=1}^{n} w_i = 1, \\
& w_i \geq 0, \quad i = 1, \ldots, n,
\end{aligned} \tag{2.3}$$

where:

- $\mathbf{w} \in \mathbb{R}^n$ is the vector of asset weights

- $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is the covariance matrix of asset returns
- $\mathbf{E} \in \mathbb{R}^n$ is the vector of expected asset returns
- $E_p \in \mathbb{R}$ is the desired expected portfolio return

Risk-Return Trade-off and the Sharpe Ratio

Investors seek to maximize returns while minimizing risk. The *Sharpe Ratio*, introduced by William F. Sharpe [33], measures the risk-adjusted return of a portfolio:

$$S = \frac{E[R_p] - R_f}{\sigma_p}, \tag{2.4}$$

where R_f is the risk-free rate. A higher Sharpe Ratio indicates a more favorable risk-return trade-off.

Portfolio Optimization Strategies

Various strategies exist for portfolio optimization, each with different objectives and constraints:

- **Maximum Return Strategy**: Focuses on maximizing expected returns, often leading to higher risk.
- **Minimum Volatility Strategy**: Aims to minimize risk while achieving a minimum acceptable return.
- **Maximum Sharpe Ratio Strategy**: Seeks the optimal balance between return and risk by maximizing the Sharpe Ratio.
- **Dynamic Strategies**: Adjust portfolio allocations based on changing market conditions and predictive insights.

2.1.2 Gaussian Process Regression Theory and Applications

Gaussian Process Regression (GPR) is a non-parametric, Bayesian approach to regression that is particularly powerful for modeling complex, non-linear relationships. GPR provides not only predictions but also uncertainty estimates, which are valuable in risk-sensitive applications like finance.

Gaussian Processes

A *Gaussian Process* (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution [30]. A GP is fully specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right). \tag{2.5}$$

Mean Function The mean function $m(\mathbf{x})$ represents the expected value of the function at point $\mathbf{x}$:

$$m(\mathbf{x}) = E[f(\mathbf{x})]. \tag{2.6}$$

Covariance Function The covariance function $k(\mathbf{x}, \mathbf{x}')$ defines the covariance between function values at points $\mathbf{x}$ and $\mathbf{x}'$:

$$k(\mathbf{x}, \mathbf{x}') = E\left[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\right]. \tag{2.7}$$

Common choices for the covariance function include the squared exponential kernel and the Matérn kernel. In our context, the exponential kernel is often used for its smoothness and flexibility in capturing complex patterns, and based on our experiments, it has shown promising results in financial time-series forecasting. So we will use the exponential kernel in our GPR model.

Gaussian Process Regression for Time-Series Forecasting

GPR models the underlying function mapping inputs to outputs, capturing uncertainty in predictions. For time-series forecasting:

- **Inputs**: Historical data points, such as lagged returns and time indices.
- **Outputs**: Future values of the time series, such as asset returns.

Given training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i$ are inputs and y_i are observations, the goal is to predict the output f_* at a new input $\mathbf{x}_*$.

Predictive Distribution The predictive distribution of f_* given the training data is Gaussian:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right), \tag{2.8}$$

where

$$\mu_* = k_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - k_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} k_*. \tag{2.9}$$

Here, $k_* = [k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_N)]^\top$, $\mathbf{K}$ is the covariance matrix with entries $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, and σ_n^2 is the noise variance.

Covariance Functions (Kernels)

The choice of kernel function $k(\mathbf{x}, \mathbf{x}')$ is critical in GPR as it encodes assumptions about the function being learned.

- *Squared Exponential Kernel*:

$$k_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2} ||\mathbf{x} - \mathbf{x}'||^2\right), \tag{2.10}$$

  where σ_f^2 is the signal variance and ℓ is the length-scale parameter.

- *Matérn Kernel*:

$$k_{\text{Matérn}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}||\mathbf{x} - \mathbf{x}'||}{\ell}\right)^\nu K_\nu \left(\frac{\sqrt{2\nu}||\mathbf{x} - \mathbf{x}'||}{\ell}\right), \tag{2.11}$$

  where ν controls the smoothness, Γ is the gamma function, and K_ν is the modified Bessel function.

- *Exponential Kernel*:

$$k_{\mathrm{Exp}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||}{\ell}\right), \tag{2.12}$$

  where σ_f^2 is the signal variance and ℓ is the length-scale parameter. This kernel is often used for modeling non-smooth functions.

Gaussian Process Regression for Time-Series Forecasting

In GPR, given training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the goal is to predict the output y_* for a new input $\mathbf{x}_*$. The predictive distribution is Gaussian with mean and variance:

$$E[y_*] = \mathbf{k}_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \tag{2.13}$$

$$\mathrm{Var}[y_*] = k_{**} - \mathbf{k}_*^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*, \tag{2.14}$$

where:

- $\mathbf{K}$ is the $n \times n$ covariance matrix evaluated at the training inputs.
- $\mathbf{k}_*$ is the covariance vector between the test point and the training inputs.
- k_{**} is the covariance between the test point and itself.
- σ_n^2 is the variance of the observation noise.
- $\mathbf{y}$ is the vector of training targets.

Risk Measurement by GPR in Financial Modeling Specifically, compared to other ML methods, GPR provides a natural framework for uncertainty quantification, delivering probabilistic predictions accompanied by confidence intervals.

These intervals are particularly valuable in risk-sensitive environments, as they enable a deeper understanding of prediction reliability, the implementation is simple as below:

```
if isFiltered:
    f_lower = f_mean - 1.96 * f_cov
    f_upper = f_mean + 1.96 * f_cov
```

The Bayesian foundation of GPR further enhances its utility, as it can naturally incorporate prior knowledge and refine its predictions as new data becomes available. This adaptability makes GPR well-suited to the dynamic and evolving nature of financial data.

In Figure 2.2, and Figure 2.3, we can see that the GPR model provides predictions for future stock prices along with confidence intervals. These confidence intervals represent the uncertainty associated with the predictions and can guide investors in assessing the reliability of the forecasts. By incorporating uncertainty estimates, GPR enables a more nuanced understanding of the potential outcomes, allowing investors to make informed decisions based on both the expected returns and the associated risks.

Challenges in Applying GPR to Finance

Despite its strengths, GPR encounters several challenges when applied to the financial domain. A significant limitation is its computational complexity, particularly the need to invert an $n \times n$ matrix, which becomes increasingly burdensome as dataset size grows [30, 29]. This issue can limit the scalability of GPR in large-scale applications.

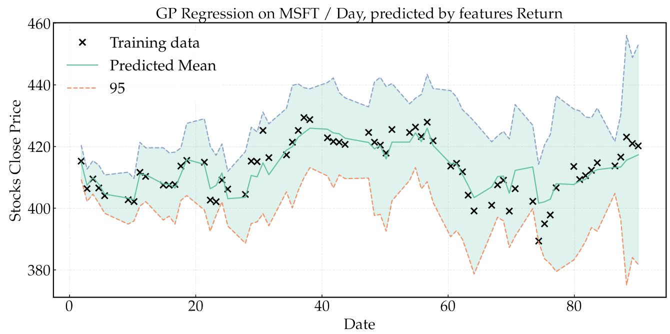


Figure 2.2: GPR Price Predictions with Confidence Intervals MSFT

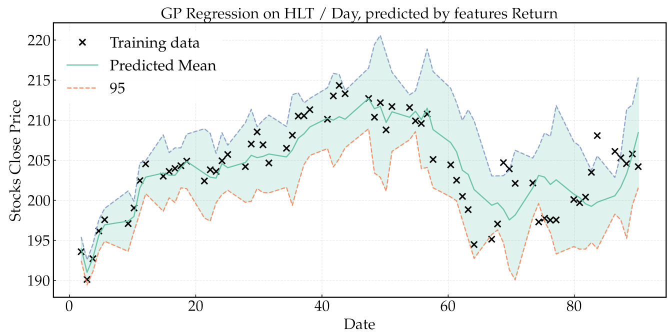


Figure 2.3: GPR Price Predictions with Confidence Intervals HLT

Another obstacle is the non-stationarity of financial time-series data, which often violates the assumptions of standard GPR implementations. Markets frequently exhibit regime changes and trends that challenge the ability of traditional models to capture shifting dynamics effectively [34, 14]. Additionally, financial datasets are characterized by high noise levels and volatility, which can adversely impact the performance and stability of GPR models [5]. Finally, the selection and tuning of kernel functions and hyperparameters play a pivotal role in GPR's effectiveness. This process is not only technically demanding but also requires a nuanced understanding of the data and the underlying financial context [30, 9].

Thus, other than predicting daily open/close prices, GPR can be used to predict the future daily returns of a stock. This can be done by training the model on historical data and then using it to predict future returns. The model can provide not only the expected return but also the uncertainty associated with the prediction. This uncertainty can be used to make more informed investment decisions, especially in risk-sensitive environments like finance.

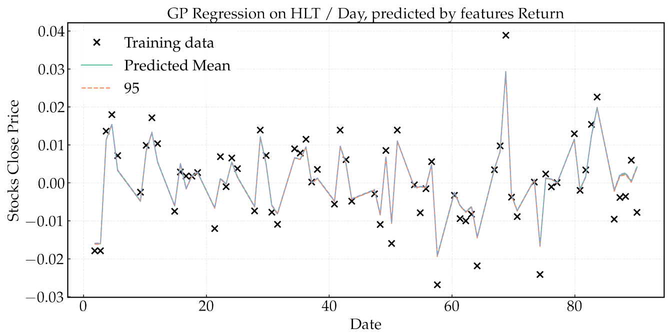


Figure 2.4: GPR Return Predictions with Confidence Intervals HLT

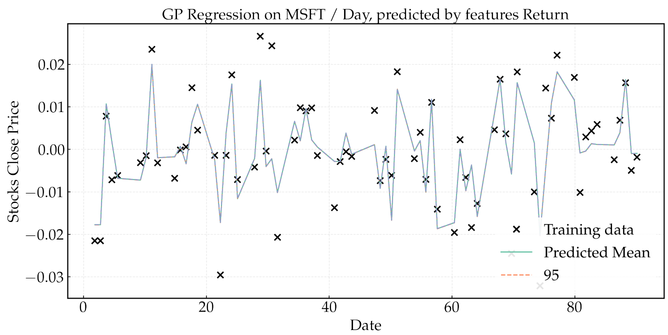


Figure 2.5: GPR Return Predictions with Confidence Intervals MSFT

From Figure 2.4 and Figure 2.5, compared with above images we can see that the predictions for stock returns by GPR model along with confidence intervals. When predicting returns, the model can capture the non-linear relationships between input features and target returns better in time-series forecasting. Therefore, the model is much more sure about the predictions, and the confidence intervals are intensively narrower. These confidence intervals represent the uncertainty associated with the predictions and can guide investors in assessing the reliability of the forecasts. By incorporating uncertainty estimates, GPR enables a more nuanced understanding of the potential outcomes, allowing investors to make informed decisions based on both the expected returns and the associated risks.

2.1.3 Integrating Portfolio Optimization and GPR

Combining portfolio optimization with GPR-based forecasting aims to leverage accurate predictions of asset returns and associated uncertainties to make informed allocation decisions. The integration involves:

The integration of GPR into portfolio optimization represents a powerful approach to improving asset allocation by leveraging predictive insights. GPR can forecast expected returns and volatilities for individual assets, providing the foundational inputs for optimization models [6, 5]. By incorporating these predictions, portfolio weights can be determined in a manner that aligns with both the expected performance and the associated risks of the assets. Furthermore, GPR's ability to quantify uncertainty enhances the optimization process by allowing confidence intervals to inform allocation decisions [3]. This ensures that portfolios are not only optimized based on point estimates but are also adjusted to account for the inherent unpredictability of financial markets. Such integration bridges the gap between theoretical advancements in predictive modeling and practical applications in portfolio management, offering a robust framework for informed and adaptive investment strategies [20].

Dynamic Portfolio Optimization Dynamic optimization involves updating the portfolio allocation as new information becomes available. By retraining the GPR models with new data and adjusting the portfolio accordingly, the strategy adapts to changing market conditions, potentially enhancing performance.

2.2 Machine Learning in Financial Markets

2.2.1 Overview of Machine Learning Applications in Finance

The financial markets generate vast amounts of data daily, encompassing stock prices, trading volumes, economic indicators, and news articles. Machine learning algorithms are uniquely positioned to process and analyze this data, uncovering patterns and insights that traditional statistical methods may overlook. Key applications of machine learning in finance include:

Time-Series Forecasting

Predicting future asset prices and market trends is a fundamental objective in finance. Machine learning models, such as neural networks, support vector machines, and ensemble methods, are employed to forecast time-series data by capturing non-linear relationships and complex temporal dependencies [32].

Algorithmic Trading

Machine learning algorithms facilitate the development of automated trading systems that execute trades based on predefined strategies and real-time data analysis. Techniques like reinforcement learning enable the optimization of trading strategies through continuous learning from market interactions [24, 7].

Risk Management

Accurate risk assessment is crucial for financial institutions. Machine learning models help in predicting credit risk, market risk, and operational risk by analyzing historical data and identifying factors that contribute to potential losses [17].

Portfolio Optimization

Machine learning enhances portfolio optimization by providing more accurate estimates of expected returns and covariances between assets. Advanced models can adapt to changing market conditions and incorporate a broader set of predictive features [15].

Fraud Detection and Anomaly Detection

Detecting fraudulent activities and anomalies is vital for maintaining the integrity of financial systems. Machine learning algorithms, particularly unsupervised learning techniques, are used to identify unusual patterns in transaction data that may indicate fraud [25].

Sentiment Analysis and Natural Language Processing

Analyzing news articles, social media, and financial reports using natural language processing (NLP) helps investors gauge market sentiment and its potential impact on asset prices [12, 18].

2.3 Dynamic Portfolio Management

Dynamic portfolio management involves continuously adjusting asset allocations in response to changing market conditions, forecasts, and investment objectives. Unlike static strategies, dynamic approaches aim to optimize the portfolio over time by incorporating new information and adapting to market dynamics. This section reviews dynamic optimization strategies, discusses the impact of transaction costs on portfolio rebalancing, and examines existing approaches to strategy switching.

2.3.1 Review of Dynamic Optimization Strategies

Dynamic optimization strategies are designed to adapt portfolio allocations over time, taking into account the stochastic nature of asset returns and changing investment opportunities. Key concepts and methods in dynamic portfolio optimization include:

Dynamic Programming

Dynamic programming is a mathematical optimization approach that solves complex problems by breaking them down into simpler subproblems. In the context of portfolio optimization, dynamic programming can be used to determine the optimal investment policy over multiple periods [4].

Bellman's Principle of Optimality Bellman's principle states that an optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. This principle underpins dynamic programming methods in portfolio optimization.

Stochastic Control Theory

Stochastic control theory deals with decision-making in systems that evolve over time under uncertainty. In portfolio optimization, stochastic control models can determine optimal asset allocations by considering the stochastic processes governing asset returns and investor wealth [21].

Merton's Portfolio Problem Robert C. Merton extended the continuous-time portfolio optimization framework by incorporating stochastic control techniques. Merton's model determines the optimal consumption and portfolio allocation strategies for an investor with a finite or infinite investment horizon, maximizing expected utility [22].

Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm where agents learn optimal policies through interactions with the environment by maximizing cumulative rewards. In finance, RL can be applied to portfolio management by treating asset allocation as a sequential decision-making problem [23].

Applications in Portfolio Management RL algorithms, such as Q-learning and policy gradients, have been used to develop adaptive trading strategies that learn from market data and adjust allocations dynamically [1].

Scenario Analysis and Model Predictive Control

Scenario analysis involves evaluating portfolio performance under different hypothetical future states of the world. Model Predictive Control (MPC) uses forecasts to optimize current decisions while considering future trajectories, adjusting strategies as new information becomes available [28].

MPC allows for the incorporation of predictive models (e.g., GPR forecasts) into the optimization process, enabling the portfolio to adapt dynamically to anticipated market changes.

2.3.2 Transaction Costs and Portfolio Rebalancing

Transaction costs play a significant role in dynamic portfolio management, as frequent rebalancing can erode returns. Understanding and modeling transaction costs are essential for effective portfolio optimization.

Types of Transaction Costs

Transaction costs can be broadly categorized into:

Types	Description
Fixed Costs	Costs that are constant per transaction, such as brokerage fees.
Variable Costs	Costs that depend on the transaction size.
Slippage	The difference between the expected and actual transaction price.

Table 2.1: Types of Transaction Costs

Impact on Portfolio Performance

High transaction costs can negate the benefits of frequent rebalancing. Incorporating transaction costs into the optimization model encourages more conservative adjustments, potentially leading to better net performance [11].

Optimal Rebalancing Frequency

Determining the optimal frequency of portfolio rebalancing is a delicate balancing act that requires careful consideration of the trade-off between preserving the intended asset allocation and minimizing transaction costs. Rebalancing too frequently can erode returns due to excessive transaction costs, while infrequent rebalancing may lead to significant deviations from the desired allocation, increasing the portfolio's risk.

One approach to addressing this challenge is threshold-based rebalancing, where adjustments are made only when asset weights deviate beyond predetermined limits. This method ensures that the portfolio remains aligned with the target allocation without incurring unnecessary trading costs. By focusing on deviations that materially impact the portfolio's risk-return profile, threshold-based rebalancing strikes an effective balance between precision and cost-efficiency.

Periodic rebalancing offers another practical solution, where the portfolio is adjusted at regular intervals, such as monthly or quarterly. This approach simplifies the rebalancing process by creating a fixed schedule, making it easier to implement and plan. However, it may not always respond to sudden market changes, potentially allowing deviations to persist longer than desired in volatile conditions.

A more sophisticated method involves cost-aware optimization, which explicitly incorporates transaction costs into the portfolio optimization process. By modeling these costs alongside expected returns and risks, this approach dynamically adjusts the rebalancing frequency and magnitude to maximize the portfolio's net performance. This strategy ensures that rebalancing decisions are not only effective but also economically justified, providing a more comprehensive solution to the rebalancing dilemma.

Together, these approaches highlight the importance of tailoring rebalancing strategies to the specific objectives and constraints of the portfolio, ensuring an optimal balance between maintaining allocation discipline and controlling costs.

2.3.3 Existing Approaches to Strategy Switching

Strategy switching involves changing between different portfolio optimization strategies based on market conditions, forecasts, or performance metrics. This adaptive approach seeks to capitalize on the strengths of various strategies under different scenarios.

Regime-Switching Models

Regime-switching models assume that financial markets alternate between different states or regimes (e.g., bull and bear markets). By identifying the current regime, investors can switch to strategies that are better suited to prevailing conditions [2].

Markov Switching Models These models use Markov processes to model transitions between regimes, allowing for probabilistic predictions of future states and informing strategy selection [13].

Meta-Learning and Ensemble Methods

Meta-learning approaches involve training a higher-level model to select the best-performing strategy from a set of candidates. Ensemble methods combine multiple strategies to create a more robust overall strategy [16].

Performance-Based Selection Strategies can be evaluated based on historical or real-time performance metrics, such as Sharpe ratios or drawdowns, with the best-performing strategy selected for implementation [26].

Rule-Based Switching

Rule-based switching systems employ predefined criteria or indicators to dictate when and how portfolio strategies should be adjusted. These systems offer a structured approach to dynamic portfolio management, enabling timely responses to evolving market conditions while maintaining a clear decision-making framework.

One common method involves the use of technical indicators. These indicators, such as moving averages, momentum signals, or relative strength indexes, are widely utilized in financial analysis to identify trends and potential reversals. For instance, a portfolio might shift to a more defensive strategy if a moving average crossover signals a downward trend in the market. Similarly, momentum indicators can prompt a reallocation toward outperforming assets, capitalizing on short-term trends.

Economic indicators also play a critical role in rule-based switching. Changes in macroeconomic conditions, such as shifts in interest rates or the release of key economic data like GDP growth or employment reports, can significantly impact market dynamics. A rule-based system might adjust portfolio strategies to align with these macroeconomic signals, such as increasing bond exposure during periods of rising interest rates or reallocating to growth stocks in anticipation of strong economic performance.

Another powerful criterion for rule-based switching is forecast confidence. This approach leverages the confidence level of predictive models, such as the variance estimates provided by Gaussian Process Regression (GPR), to guide strategy adjustments. For example, when GPR forecasts indicate high confidence in a specific market movement, the system can allocate more aggressively toward assets expected to benefit from that trend. Conversely, during periods of high uncertainty, the system might adopt a more conservative stance to mitigate risk.

Rule-based switching provides a disciplined yet flexible framework for dynamic portfolio management, allowing investors to adapt to changing conditions while minimizing emotional biases in decision-making. By integrating technical, economic, and predictive indicators, such systems can enhance both responsiveness and effectiveness in achieving portfolio objectives.

Machine Learning-Based Switching

Machine learning models can be trained to predict the optimal strategy based on market data and indicators. Classification algorithms, such as support vector machines or neural networks, can identify patterns associated with the outperformance of specific strategies [10].

2.3.4 Conclusion

Dynamic portfolio management seeks to enhance investment performance by adapting to market conditions and new information. Incorporating transaction costs into the optimization process is crucial for realistic strategy implementation. Existing approaches to strategy switching provide valuable frameworks for developing adaptive strategies, leveraging techniques such as regime-switching models, meta-learning, rule-based systems, and machine learning. This study builds upon these concepts by introducing a probabilistic, threshold-based strategy switching mechanism informed by Gaussian Process Regression forecasts.

3 Methodology

In this section, we present the methodology for developing the predictive framework and portfolio optimization strategies. We outline the data collection and preprocessing steps, the model development process, the evaluation metrics used to assess the performance of the models and the real backtesting process. The methodology encompasses the selection of assets, the application of entropy-based methods for time-series analysis, and the development of Gaussian Process Regression models for return prediction.

Additionally, we describe the implementation of portfolio optimization strategies, including the construction of trading portfolios and the evaluation of risk-adjusted returns using the Sharpe Ratio. The methodology is designed to provide a comprehensive framework for forecasting asset returns and optimizing portfolio allocations, enabling informed investment decisions and risk management strategies.

First of all, below is the code structure of the project:

```
|-- main.py
|-- Strategies
|   |-- Strategy.py
|   |-- Constant_EqualWeight_Strategy.py
|   |-- Max_Return_Strategy.py
|   |-- Min_Volatility_Strategy.py
|   |-- Max_SharpeRatio_Strategy.py
|   |-- Dynamic_Strategy.py
|-- Portfolio
|   |-- Portfolio.py
|   |-- Return.py
|   |-- Volatility.py
|-- Models
|   |-- ARIMA.py
|   |-- GPR_trainer.py
|-- Optimization
|   |-- Optimizer.py
|-- Utils
|   |-- Data_preprocessing.py
```

```
|   |-- Metrics.py
|   |-- Visualizer.py
|-- requirements.txt
```

Design Principles The modular design ensures that each component is self-contained and reusable. For example, new strategies can be added by extending the base class in `Strategy.py`, and additional predictive models can be integrated into the `Models/` directory. This structure also facilitates debugging and testing, as each module operates independently.

The inclusion of utility functions in the `Utils/` directory allows for centralized handling of common tasks like data preprocessing and visualization, improving code maintainability and reducing redundancy.

3.1 Data Collection and Preprocessing

3.1.1 Asset Selection and Justification

In constructing the portfolio for this study, we selected five stocks representing five major sectors out of the eleven sectors defined by the Global Industry Classification Standard (GICS): Information Technology, Health Care, Industrials, Consumer Discretionary, and Financials. These sectors were chosen due to their significant contributions to the S&P 500 Index, collectively accounting for the majority of the index's market capitalization as shown in Figure 3.1.

The selected stocks are as follows: Microsoft Corporation (MSFT) from the Information Technology sector, Johnson & Johnson (JNJ) from the Health Care sector, The Boeing Company (BA) from the Industrials sector, Hilton Worldwide Holdings Inc. (HLT) from the Consumer Discretionary sector, and JPMorgan Chase & Co. (JPM) from the Financials sector. And these five individual stocks are selected based on entropy analysis and comparison.

Stock Market Sectors The rationale behind this selection is multi-staged. Firstly, by including stocks from these predominant sectors, the portfolio captures the performance dynamics of the key drivers of the U.S. equity market. This approach ensures that the portfolio is both diversified and representative of broader market movements. Diversification across different sectors helps mitigate unsystematic risk associated with individual industries or companies, allowing the portfolio to benefit from growth opportunities while reducing the impact of sector-specific downturns.

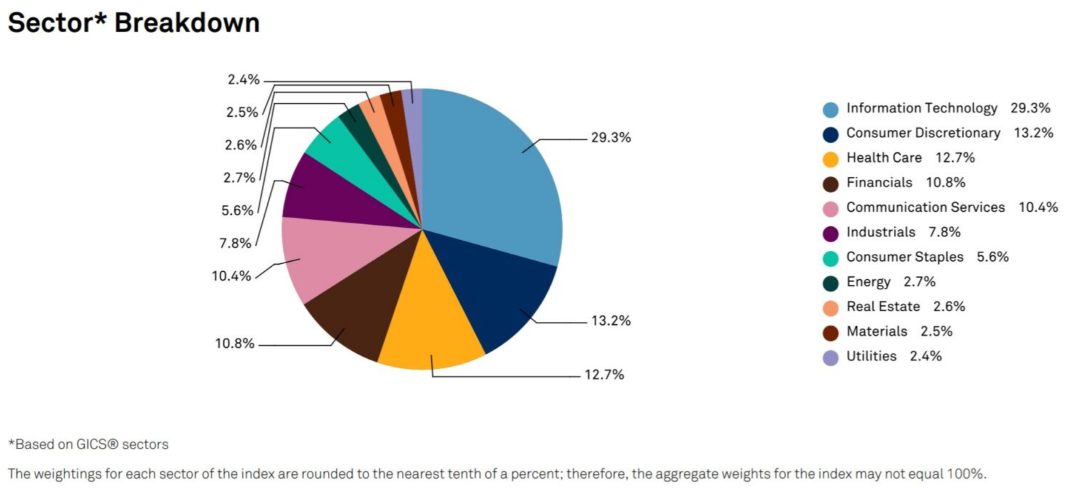


Figure 3.1: S&P 500 Sector Weightings(Source: [31])

Secondly, focusing on sectors that contribute most significantly to the S&P 500 enhances the relevance of the study's findings to investors who benchmark their performance against this widely recognized index. The Information Technology sector, for example, encompasses leading companies in innovation and technology, significantly influencing market trends. The Health Care sector includes firms pivotal in advancing medical technology and pharmaceuticals, demonstrating resilience in various economic conditions.

Industrials and Consumer Discretionary sectors represent core components of the economy, reflecting business cycles and consumer spending patterns, respectively. The Financials sector plays a critical role in economic infrastructure, encompassing banking, insurance, and investment services. By selecting assets from these sectors, the portfolio encompasses areas crucial to economic growth and market capitalization.

This strategic selection aligns with the study's objective of developing a predictive portfolio optimization framework applicable to a realistic and impactful set of assets. It allows for testing the effectiveness of the proposed models and strategies in a context that is meaningful to investors and relevant to overall market performance. Moreover, by focusing on sectors that significantly influence the S&P 500, the study ensures that its results have practical implications for portfolio management and investment decision-making.

Entropy Methods for Time Series Analysis To assess the complexity of the time-series data for these assets, we utilized entropy-based methods. Specifically, we employed the `OrdianlEntropy` package in Python [27], which provides time-efficient, ordinal pattern-based entropy algorithms for computing the complexity of one-dimensional time-series. This analysis informed our feature engineering process by highlighting the inherent unpredictability and dynamic behavior of the asset prices.

Permutation Entropy (PE) Permutation Entropy (PE), introduced by Bandt and Pompe in 2002, is a measure of complexity that quantifies the diversity of ordinal patterns in a time series. It is based on the relative ordering of the values within embedding vectors derived from the time series. Given a time series ${x_t}_{t=1}^{N}$, the calculation of PE involves the following steps:

1. **Embedding the Time Series:** Choose an embedding dimension m and time delay τ, and construct embedding vectors:

$$s_i = \left(x_i, x_{i+\tau}, x_{i+2\tau}, \ldots, x_{i+(m-1)\tau}\right), \quad i = 1, 2, \ldots, N - (m-1)\tau. \tag{3.1}$$

2. **Forming Ordinal Patterns:**

   For each embedding vector s_i, determine the permutation π_i that sorts its components in ascending order:

$$s_i \to \left(x_{i+\tau(k_0)} \leq x_{i+\tau(k_1)} \leq \cdots \leq x_{i+\tau(k_{m-1})}\right), \tag{3.2}$$

   where $(k_0, k_1, \ldots, k_{m-1})$ represents the indices of the sorted components.

3. **Calculating Probabilities:**

   Count the frequency of each of the $m!$ possible ordinal patterns π_j, and estimate their probabilities:

$$p(\pi_j) = \frac{\text{Number of times pattern } \pi_j \text{ occurs}}{N - (m-1)\tau}. \tag{3.3}$$

4. **Calculating Permutation Entropy:**

   Compute the permutation entropy using Shannon's entropy formula:

$$\text{PE} = -\sum_{j=1}^{m!} p(\pi_j) \ln p(\pi_j). \tag{3.4}$$

   Normalize Permutation Entropy (PE) by dividing by $\ln(m!)$ to ensure it ranges between 0 and 1:

$$\text{PE}_{\text{norm}} = \frac{\text{PE}}{\ln(m!)}. \tag{3.5}$$

- A **low PE** value indicates that the time series has a predictable or regular structure, with fewer unique ordinal patterns.
- A **high PE** value suggests that the time series is more complex or random, exhibiting a higher diversity of ordinal patterns.

Weighted Permutation Entropy (WPE) Weighted Permutation Entropy (WPE) enhances PE by incorporating the amplitude information of the time series. It assigns weights to ordinal patterns based on the values within the embedding vectors, thus accounting for both the order and magnitude of the data. The calculation of Weighted Permutation Entropy (WPE) involves the following steps:

1. **Embedding and Ordinal Patterns:** Follow steps 1 and 2 as in the calculation of PE to obtain the embedding vectors and their corresponding ordinal patterns.

2. **Assigning Weights:**

   For each embedding vector s_i, calculate a weight w_i based on the amplitude of its components. A common choice is the variance or absolute differences:

$$w_i = \mathrm{Var}(s_i) = \frac{1}{m}\sum_{k=0}^{m-1}\left(x_{i+\tau k} - \bar{x}_i\right)^2, \tag{3.6}$$

   where $\bar{x}_i$ is the mean of the components of s_i.

3. **Calculating Weighted Probabilities:**

   For each ordinal pattern π_j, compute the weighted probability:

$$p_w(\pi_j) = \frac{\sum_{s_i \in \pi_j} w_i}{\sum_{i=1}^{N-(m-1)\tau} w_i}. \tag{3.7}$$

4. **Calculating Weighted Permutation Entropy:**

   Compute WPE using the weighted probabilities:

$$\mathrm{WPE} = -\sum_{j=1}^{m!} p_w(\pi_j) \ln p_w(\pi_j). \tag{3.8}$$

   Normalize WPE similarly to PE:

$$\mathrm{WPE}_{\mathrm{norm}} = \frac{\mathrm{WPE}}{\ln(m!)}. \tag{3.9}$$

- **WPE** reflects both the diversity of ordinal patterns and the magnitude of fluctuations in the time series.
- It is more sensitive to significant changes in amplitude, making it useful for detecting volatility and abrupt transitions.

Dispersion Entropy (DE) Dispersion Entropy (DE), introduced by Rostaghi and Azami in 2016, is an entropy measure that accounts for both the ordering and dispersion of data by mapping the time series into a sequence of classes or symbols. To calculate DE, follow these steps:

1. **Mapping Data to Classes:** Define c classes and map each data point x_t to a class label k_t using a mapping function, such as:

$$k_t = \left\lfloor c \cdot \frac{x_t - x_{\min} + \epsilon}{x_{\max} - x_{\min} + 2\epsilon} \right\rfloor + 1, \quad k_t \in \{1, 2, \ldots, c\}, \tag{3.10}$$

   where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the time series, and ϵ is a small constant to prevent division by zero.

2. **Embedding:**

   Form embedding vectors of class labels:

$$s_i = \left(k_i, k_{i+\tau}, k_{i+2\tau}, \ldots, k_{i+(m-1)\tau}\right). \tag{3.11}$$

3. **Creating Dispersion Patterns:**

   Each embedding vector s_i represents a dispersion pattern d_j, where j indexes the possible patterns.

4. **Calculating Probabilities:**

   Count the frequency of each possible dispersion pattern and estimate their probabilities:

$$p(d_j) = \frac{\text{Number of times pattern } d_j \text{ occurs}}{N - (m-1)\tau}. \tag{3.12}$$

5. **Calculating Dispersion Entropy:**

   Compute DE using:

$$\mathrm{DE} = -\sum_{j=1}^{c^m} p(d_j) \ln p(d_j). \tag{3.13}$$

Normalize DE by dividing by $\ln(c^m)$:

$$\mathrm{DE}_{\mathrm{norm}} = \frac{\mathrm{DE}}{\ln(c^m)}. \tag{3.14}$$

- **DE** considers both the frequency and dispersion of data, providing a robust measure of complexity for non-stationary time series.
- It effectively captures changes in amplitude and is less sensitive to noise and outliers.

Choosing WPE and DE for Stock Price Analysis Analyzing stock prices requires accounting for both the order and magnitude of price changes due to the following reasons:

Weighted Permutation Entropy (WPE) offers distinct benefits in the analysis of financial time series by effectively incorporating the amplitude of price changes into its computation. By weighting ordinal patterns, WPE not only captures the direction of changes but also accounts for their magnitude, making it particularly adept at identifying periods of heightened market volatility. This sensitivity to volatility is crucial for risk management, as it allows analysts to detect significant shifts in market behavior that could impact investment decisions. Furthermore, WPE strikes a balance between computational efficiency and analytical depth. It provides detailed insights into the complexity of financial data without requiring excessive computational resources, making it a practical tool for real-time and large-scale applications.

Dispersion Entropy (DE) excels in handling the complexities of non-stationary financial time series, which often exhibit trends, abrupt changes, and varying volatility. By mapping data into discrete classes, DE adapts effectively to non-stationarity, ensuring robust analysis across different market conditions. Its design also makes it less sensitive to noise and outliers, a significant advantage in the inherently noisy financial domain, where anomalies can skew traditional measures. DE provides a comprehensive view of market dynamics by considering both the frequency and amplitude of patterns, capturing subtle shifts in market behavior.

Another notable advantage of DE is its amplitude sensitivity. In financial markets, the size of price movements has a direct impact on returns and risks, and methods like DE that incorporate this information provide a richer understanding of market dynamics. Additionally, DE is particularly effective at detecting varying levels of volatility, which play a critical role in shaping investment strategies. Its ability to seamlessly handle the non-stationarity of stock prices, which often undergo regime changes and unpredictable trends, further solidifies its value in financial analysis.

Both WPE and DE offer robust tools for analyzing financial markets, with WPE emphasizing computational balance and volatility capture, while DE focuses on non-stationarity adaptation and comprehensive market understanding. Together, they provide complementary approaches to understanding the complexity of financial time series.

Conclusion Weighted Permutation Entropy (WPE) and Dispersion Entropy (DE) prove to be superior tools compared to traditional Permutation Entropy when analyzing stock prices. These methods capture essential information about both the direction and magnitude of price movements, offering a more nuanced understanding of financial time-series data. Their ability to handle the inherent non-stationarity and volatility of financial markets makes them particularly well-suited for detecting patterns and trends that are often overlooked by simpler measures. Furthermore, WPE and DE provide more informative and sensitive complexity measures, enabling analysts to enhance asset selection and refine risk assessment strategies.

Table 3.1: Entropy Comparison for Selected Stocks

	MSFT	**JNJ**	**BA**	**HLT**	**JPM**
PE	0.67909	0.52547	0.52344	0.68303	0.50430
WPE	0.12027	0.23113	0.21192	0.12082	0.23113
Dispersion Entropy (DE)	0.94164	0.93484	0.92995	0.97267	0.93663

By utilizing WPE and DE, analysts and investors can achieve a deeper understanding of market behavior, uncovering the complexity and predictability of stock prices. These insights facilitate more informed decisions in selecting assets across different sectors, ultimately improving portfolio management and investment outcomes.

3.1.2 Data Sources and Time Period

To acquire the historical market data necessary for this study, we employed the EOD Historical Data API, a reputable and comprehensive source for end-of-day and historical financial data across various asset classes. The data retrieval process was automated using a Python script that constructs API requests based on the asset ticker, desired data period (e.g., daily, weekly), and specified date range.

For U.S.-listed assets, the script utilizes the endpoint format:

```
https://eodhd.com/api/eod/{ticker}.US?period={period}&api_token={api_token}
```

where `{ticker}` represents the stock symbol, `{period}` denotes the data frequency, `{api_token}` is the authentication token, and `{start_date}` and `{end_date}` define the data range. For Bitcoin (BTC), which is categorized differently in the API, the script accesses data using the endpoint:

```
https://eodhd.com/api/eod/BTC-USD.CC?period={period}&api_token={api_token}
```

An API token, securely stored using environment variables to maintain confidentiality, is included in the requests for authentication. The script handles HTTP responses by checking for successful status codes and raising exceptions in case of errors, ensuring robust data retrieval.

Upon receiving a valid response, the JSON data is parsed into a pandas DataFrame for efficient data manipulation and analysis. The DataFrame includes essential financial indicators such as open, high, low, close prices, and trading volumes. The data is then saved as a CSV file in a structured directory hierarchy corresponding to each asset, following the path:

```
../Stocks/{ticker}/{ticker}_us_{period}.csv
```

By automating the data fetching and saving process, we ensured consistency and repeatability in data collection of the whole pipeline. This method allowed us to systematically gather historical market data for all selected assets over the specified time periods, providing a reliable dataset for training the Gaussian Process Regression models and conducting backtesting for the portfolio optimization strategies. The use of the EOD Historical Data API ensured that the data was up-to-date and accurate, reflecting real market conditions essential for the validity of our analysis. Also, for future work, the script can be easily adapted to retrieve data for additional assets or extend the time period, enabling further research and analysis in the financial domain.

3.1.3 Data Preprocessing and Log Returns

Data preprocessing and feature engineering are critical steps in preparing the dataset for modeling. They ensure that the data fed into the Gaussian Process Regression models are clean, consistent, and informative.

Use of Log Returns In this project, log returns are utilized for modeling asset price movements due to several compelling reasons that align with both theoretical and practical considerations in financial analysis.

Theoretical Foundation in Finance Log returns are integral to many foundational financial models, such as the Black-Scholes option pricing model, which assume that asset prices follow a log-normal distribution. By using log returns, we align our modeling approach with these theoretical frameworks, facilitating more accurate and consistent analyses. GPR models assume normally distributed outputs, and since log returns of log-normally distributed prices are normally distributed, this compatibility enhances the effectiveness of our predictive modeling.

Stability Over Time Log returns exhibit greater stability compared to simple arithmetic returns, particularly in the presence of extreme outliers or during periods of high market volatility. They tend to smooth out spikes and reduce the impact of short-term noise, making the models less sensitive to sudden market anomalies. This stability is crucial for developing robust predictive models that can perform reliably under various market conditions.

Time Consistency (Additivity) One of the key mathematical properties of log returns is their additive nature over time. The total log return over a period is the sum of the log returns over sub-periods:

$$\log\left(\frac{S_t}{S_0}\right) = \log\left(\frac{S_t}{S_{t-1}}\right) + \log\left(\frac{S_{t-1}}{S_{t-2}}\right) + \cdots + \log\left(\frac{S_1}{S_0}\right), \tag{3.15}$$

where S_t is the asset price at time t. This additive property simplifies the computation of returns over arbitrary time horizons, such as weekly or monthly periods, by allowing us to sum daily log returns. It is particularly beneficial for forecasting and portfolio optimization over multi-day horizons, as it facilitates the aggregation of returns without the need for complex compounding calculations.

Normalization of Price Scale Log returns are scale-invariant, meaning they standardize returns across assets regardless of their price levels. Whether an asset is priced at \$1 or \$1,000, the log return brings their percentage changes onto a consistent scale. This normalization simplifies comparisons across assets with vastly different price levels and reduces the need for additional data scaling or normalization procedures. It ensures that no single asset disproportionately influences the model due to its absolute price, allowing for a more balanced and equitable analysis within the portfolio.

Data Normalization and Scaling To bring all features onto a similar scale and improve the numerical stability of the models, we applied data normalization techniques.

Specifically, we used min-max scaling to normalize the historical return features and the time index:

$$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}, \tag{3.16}$$

where X represents the original feature values, and X_{min} and X_{max} are the minimum and maximum values of the feature, respectively. This scaling transforms the data to a [0, 1] range, facilitating efficient model training.

Treatment of Missing Data and Outliers Financial time-series data often contain missing values and outliers due to market closures, data recording errors, or extreme market events. To address missing data, we employed interpolation methods appropriate for time-series, such as linear interpolation and forward/backward filling, ensuring temporal continuity in the data.

Outliers were identified using the Interquartile Range (IQR) method:

$$\text{IQR} = Q_3 - Q_1, \tag{3.17}$$

where Q_1 and Q_3 are the first and third quartiles, respectively. Data points lying outside 1.5 times the IQR from the quartiles were considered outliers. We assessed these outliers to determine whether they were due to data errors or genuine market anomalies. Genuine outliers representing significant market movements were retained to preserve the dataset's integrity, while erroneous data points were corrected or removed.

Handling Missing Data with GPR GPR is particularly effective in addressing missing data and managing outliers due to its probabilistic framework and ability to model uncertainties. As demonstrated in the figure, GPR leverages the observed data points to predict missing values by constructing a distribution over possible functions that fit the data. As shown in 3.2 for each prediction, GPR provides both a mean estimate and a confidence interval that quantifies the uncertainty of the prediction. This feature allows it to interpolate missing data seamlessly while considering the underlying patterns and relationships in the time-series.

Furthermore, GPR handles outliers by naturally weighting data points according to their fit within the modeled distribution. Outliers, which deviate significantly from the predicted mean and exhibit higher uncertainty, are less influential in the model's predictions. This property prevents extreme values from distorting the regression output, making GPR robust in noisy and incomplete datasets. The ability to quantify uncertainty not only aids in filling missing data but also provides a valuable tool for identifying potential anomalies in the dataset, as seen in the highlighted regions of

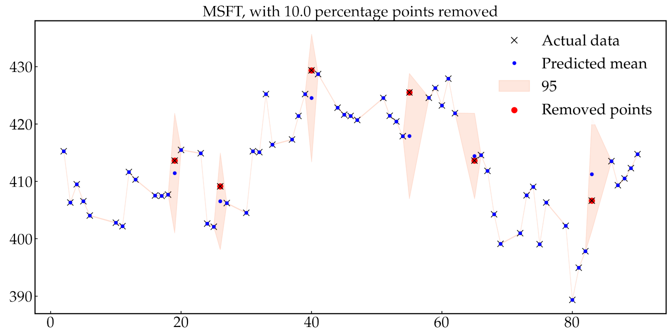


Figure 3.2: GPR Prediction with Missing Data and Outliers

the plot. By incorporating these capabilities, GPR ensures that the dataset remains coherent and usable for subsequent analysis, such as predictive modeling and portfolio optimization.

Data Splitting and Cross-Validation The dataset was divided into training and testing sets to evaluate the model's predictive performance. The training set consisted of the first 80% of the time period, while the remaining 20% was reserved for testing. This chronological split respects the temporal order of the data, avoiding look-ahead bias.

To further validate the models, we used time-series cross-validation with a rolling window approach. In each iteration, the model was trained on a window of consecutive data points and tested on the subsequent period. This method provides a more robust assessment of the model's performance over time and simulates real-world forecasting conditions.

Sliding Window Approach To denoise the time-series data and reduce short-term fluctuations, we employed a sliding window approach using a centered rolling window mechanism. For each data point in the series, a window of a specified size w was centered around it, and a statistical function was applied to the data within this window to compute a denoised value. The primary function used was the mean, though other functions could be applied as needed.

Formally, let $\{x_t\}_{t=1}^{T}$ represent the original time-series data, and $\{\tilde{x}_t\}_{t=1}^{T}$ denote the denoised series. The denoised value at time t, $\tilde{x}_t$, is calculated as:

$$\tilde{x}_t = \frac{1}{n_t} \sum_{i=t-k}^{t+k} x_i, \tag{3.18}$$

where $k = \left\lfloor \frac{w}{2} \right\rfloor$, and n_t is the number of data points within the window centered at time t. The window size w is an odd integer to ensure symmetry around the central point. At the edges of the time-series (when $t-k<1$ or $t+k>T$), the window is adjusted by including available data points, and the minimum number of periods is set to 1 to allow computation even with incomplete windows.

To handle any missing values that may arise at the edges due to insufficient data points, we applied forward and backward filling methods. Forward filling propagates the last observed non-missing value forward to fill subsequent missing positions, while backward filling fills missing values by propagating the next observed non-missing value backward. These steps ensure that the denoised series is complete and free from missing values.

This sliding window denoising process effectively smooths the data by averaging over the local neighborhood of each data point, reducing random noise while preserving significant trends and patterns. By enhancing the signal-to-noise ratio in the time-series, this approach improves the quality of the input data for the Gaussian Process Regression models, leading to better predictive performance and more reliable portfolio optimization decisions.

In this project, we experimented on window size = 1, 3, 5, 10, and eventually applied sliding window size at 3 for data pre-processing in the end-to-end pipeline.

Gaussian Filter Denoising Method To further enhance the quality of the time-series data and reduce high-frequency noise, we employed the Gaussian filter denoising method. The Gaussian filter is a convolutional filter that applies a Gaussian kernel to smooth data by averaging neighboring points with weights determined by the Gaussian function. This technique preserves significant trends and patterns while effectively attenuating random fluctuations and noise.

The Gaussian kernel is defined by the Gaussian (normal) distribution function:

$$G(i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{i^2}{2\sigma^2}\right), \tag{3.19}$$

where i is the index distance from the central data point, and σ is the standard deviation of the Gaussian distribution, controlling the degree of smoothing. A larger σ results in a wider kernel and more extensive smoothing.

The denoised value at time t, $\tilde{x}_t$, is computed by convolving the original time-series data x_t with the Gaussian kernel:

$$\tilde{x}_t = \sum_{i=-k}^{k} G(i) \cdot x_{t+i}, \tag{3.20}$$

where k is the window size parameter determining the range of data points considered around time t.

In our implementation, we applied the Gaussian filter to the closing prices of the assets using a standard deviation $\sigma = 1$, which provides a balanced smoothing effect without overly distorting the data. The following code snippet illustrates the application of the Gaussian filter using the `gaussian_filter` function from the SciPy library in Python:

```
if isFiltered:
    df['filtered_close'] = gaussian_filter(df['close'], sigma=1)
```

In this code, `df['close']` represents the original closing price data, and `df['filtered_close']` stores the denoised data after applying the Gaussian filter. The `isFiltered` flag allows for conditional application of the filtering process.

By using the Gaussian filter denoising method, we effectively reduced the impact of noise and short-term fluctuations in the financial time-series data. This preprocessing step enhances the signal-to-noise ratio, allowing the Gaussian Process Regression models to focus on underlying market trends and improving the accuracy of the return predictions. The combination of the sliding window approach and Gaussian filtering provides a robust methodology for data smoothing, contributing significantly to the reliability and performance of the predictive modeling and subsequent portfolio optimization.

The filtered data retained essential market movements while reducing random fluctuations, allowing the Gaussian Process Regression models to focus on meaningful signals.

Summary

By carefully selecting assets, sourcing reliable data, and meticulously preprocessing the dataset, we established a solid foundation for our predictive modeling. The combination of normalization, outlier treatment, strategic data splitting, and denoising ensured that the inputs to our models were of high quality. These steps are crucial for enhancing the performance of the Gaussian Process Regression models and, ultimately, for developing effective portfolio optimization strategies.

3.2 Model Development

In this section, we outline the development of the predictive framework and evaluation methodology for forecasting asset returns and optimizing portfolio allocations, namely trading strategies. The focus is on the performance metrics used to assess model effectiveness, as well as the selection of a baseline model for comparison with the Gaussian Process Regression (GPR) approach.

3.2.1 Baseline Models

To establish a reference point for the performance of the Gaussian Process Regression model, we selected the ARIMA (AutoRegressive Integrated Moving Average) model as the baseline.

ARIMA Model

The ARIMA model is a widely used time-series forecasting technique that combines autoregressive (AR), differencing (I), and moving average (MA) components. It is defined by three parameters: p, d, and q, where:

- p is the order of the autoregressive term.
- d is the degree of differencing required to achieve stationarity.
- q is the order of the moving average term.

The ARIMA model forecasts the value at time t as:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}, \tag{3.21}$$

where ϕ_i are the AR coefficients, θ_i are the MA coefficients, and ϵ_t is the white noise term.

Rationale for Baseline Selection

The ARIMA model serves as an appropriate baseline because it is a well-established method for modeling time-series data and capturing trends and seasonality. While it is linear in nature, ARIMA provides a benchmark for assessing the added value of GPR, which can capture complex non-linear relationships and quantify uncertainty in predictions.

3.2.2 Comparative Evaluation and Performance Metrics

The performance of the GPR model is compared against the ARIMA baseline using the Mean Squared Error (MSE) for predictive accuracy and the Sharpe Ratio for portfolio optimization outcomes. This dual evaluation approach ensures that the models are assessed comprehensively, both in terms of their forecasting ability and their impact on investment decisions. By demonstrating improvements over the ARIMA model, the study highlights the effectiveness of GPR in dynamic portfolio management.

To evaluate the performance of the predictive models and portfolio strategies, we employ the following metrics:

Mean Squared Error (MSE)

The Mean Squared Error (MSE) measures the accuracy of the model's predictions compared to the actual observed values. It is calculated as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 , \tag{3.22}$$

where y_i represents the actual value, $\hat{y}_i$ is the predicted value, and n is the total number of predictions. A lower MSE indicates better predictive accuracy. This metric is particularly useful for evaluating the performance of GPR and baseline models in forecasting asset returns.

Sharpe Ratio

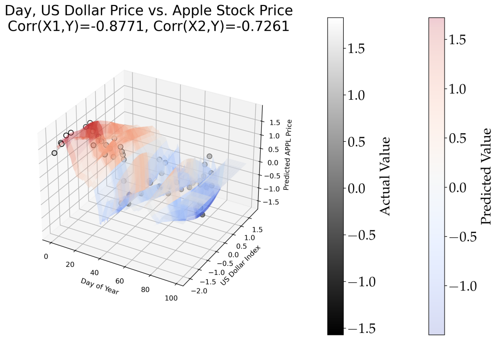
The Sharpe Ratio assesses the risk-adjusted performance of a portfolio, measuring the excess return per unit of risk. It is defined as:

$$\text{Sharpe Ratio} = \frac{E[R_p] - R_f}{\sigma_p}, \tag{3.23}$$

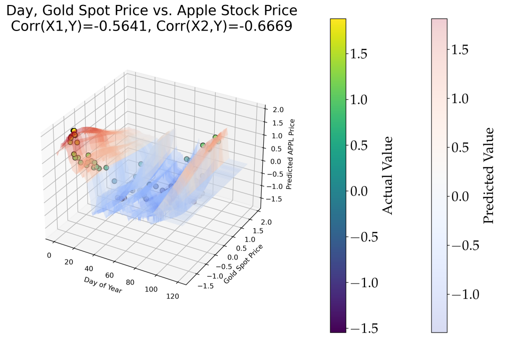
where $E[R_p]$ is the expected portfolio return, R_f is the risk-free rate, and σ_p is the portfolio's standard deviation of returns. A higher Sharpe Ratio indicates a more favorable trade-off between risk and return, making it a crucial metric for evaluating the effectiveness of the portfolio optimization strategies derived from the predicted returns.

3.2.3 Multi-Input Gaussian Process Regression Model

Unlike the Single-Input Gaussian Process Regression (GPR) model, which is designed to predict the returns of individual stocks based on historical time-series data, with only one input dimension - time, multi-input GPR extends the model to incorporate multiple economic factors and asset prices as input dimensions. This extension allows the model to capture complex relationships between the target stock and broader market dynamics, enhancing its predictive power and robustness. For example, in Figure 3.3, we visualize the relationships between the returns of Apple Inc. (AAPL) and



(a) US Dollar Price & Time vs. AAPL Returns



(b) Gold Spot Price & Time vs. AAPL Returns

Figure 3.3: Comparisons of Price and Time with AAPL Returns for Different Assets.

two input features: the US Dollar Index (DXY) and the Gold Spot Price (XAU/USD). The scatter plots illustrate the interactions between the target stock's returns and the selected economic factors, highlighting the potential predictive value of multi-input GPR modeling.

In this study, we developed a multi-input Gaussian Process Regression (GPR) model that incorporates multiple economic factors and time as input dimensions to predict the returns of individual stocks.

The model combines features from a diverse set of assets and economic indicators, including Brent Oil prices, the US Dollar Index (DXY), the S&P 500 Index, the Nasdaq 100 Index, Bitcoin (BTC), and gold prices (XAU/USD). This multi-input framework allows the model to capture intricate relationships between the target stock and broader market dynamics, enhancing its predictive power. Some examples of visualizations of two input features are shown in the figures below:

Input Data and Feature Selection

As shown in 3.4, the input features for the GPR model were selected based on their correlation with the target stock. For each feature, we calculated the correlation between the feature's historical returns and the target stock's returns. Only features with an absolute correlation exceeding a predefined threshold were included in the model to ensure relevance and reduce noise in the predictions. This approach prioritizes the most influential economic factors while discarding irrelevant or weakly correlated inputs.

The final input matrix was constructed by concatenating the selected features along with time as an additional dimension. By including time, the model accounts for temporal trends and seasonality in the data, further improving its predictive capabilities.

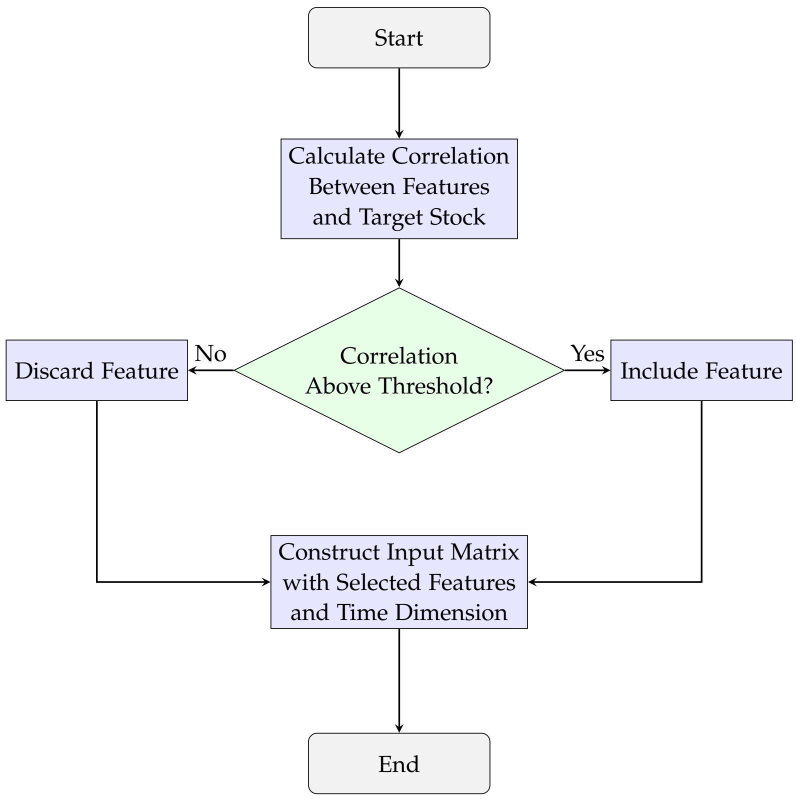


Figure 3.4: Flowchart for Feature Selection in the GPR Model

Below is an example of Auto-select Assets correlation > threshold 0.30

```
Full Correlation Matrix:
[[ 1. 0.46706671 0.50222172 0.24712746 -0.0620827 0.67685479
   0.49680488 -0.62550917]
 [ 0.46706671 1. 0.36688963 -0.28276416 -0.54027736 0.64799578
   0.70426379 -0.17388135]
 [ 0.50222172 0.36688963 1. 0.57943381 0.11520806 0.78625395
   0.83010455 -0.50745736]
 [ 0.24712746 -0.28276416 0.57943381 1. 0.85438818 0.26459202
   0.2470917 -0.1514256 ]
 [-0.0620827 -0.54027736 0.11520806 0.85438818 1. -0.16046186
  -0.1921287 0.16807068]
 [ 0.67685479 0.64799578 0.78625395 0.26459202 -0.16046186 1.
   0.92642172 -0.57928874]
 [ 0.49680488 0.70426379 0.83010455 0.2470917 -0.1921287 0.92642172
   1. -0.38304662]
 [-0.62550917 -0.17388135 -0.50745736 -0.1514256 0.16807068 -0.57928874
  -0.38304662 1. ]]
Correlation between X1 and Y: -0.6255 # Selected Brent_Oil for training
Correlation between X2 and Y: -0.1739 # Selected DXY for training
Correlation between X3 and Y: -0.5075 # Selected BAC for training
Correlation between X4 and Y: -0.1514 # Selected SP500 for training
Correlation between X5 and Y: 0.1681 # Selected NasDaq100 for training
Correlation between X6 and Y: -0.5793 # Selected XAU_USD for training
Correlation between X7 and Y: -0.3830 # time
Mean Squared Error: 2.7350
```

Composite Kernel for Multi-Dimensional Inputs

To effectively model the multi-dimensional input space, we employed a composite kernel approach that assigns separate kernels to different input dimensions. Specifically:

- One kernel was applied to the economic factors, allowing the model to learn the specific relationships between the target stock and each factor. This kernel captures the varying scales and complexities of the economic data.
- Another kernel was applied to the time dimension, enabling the model to account for temporal dependencies and trends.

The composite kernel combines these two components multiplicatively, ensuring that the model can flexibly adjust to different input dimensions and set appropriate length scales for each.

Note that it is better to avoid any kernel length-scale to be too high, because it would either imply that this input kernel has no contribution to the final predictions at all, or the input data is too noisy. whereas this design allows the GPR model to capture the unique characteristics of each feature while modeling their joint effects on the target stock's returns.

Training and Testing Framework

The training and testing process involved splitting the data into separate periods. The training set included data from the beginning of the time series up to a specified date, while the testing set encompassed the subsequent period. This split ensured that the model was evaluated on unseen data, simulating real-world forecasting conditions.

To further enhance the model's robustness, the noise variance in the GPR model was either fixed to a small value or optimized as a trainable parameter. This dual approach ensured flexibility in handling data with varying noise levels while preventing overfitting.

Prediction and Evaluation

Once the model was trained, it was used to predict the returns of the target stock over the testing period. The predictive mean and variance were computed for each test point, providing both point estimates and uncertainty bounds. The model's performance was evaluated using the Mean Squared Error (MSE), calculated for both normalized and denormalized data, to assess its accuracy in both relative and absolute terms.

By leveraging multi-input GPR with composite kernels, this approach captures the interplay between diverse economic factors, time, and stock returns. The flexibility in kernel design and inclusion of multi-dimensional inputs ensures that the model is both robust and adaptive, providing accurate predictions for individual stock performance in dynamic market environments.

3.2.4 Hyperparameter Tuning

Effective hyperparameter tuning is crucial for optimizing the performance of GPR models, as it directly impacts the model's ability to fit the data and generalize to unseen samples. In this study, two distinct approaches were implemented for training GPR models, differing in how the likelihood variance (noise variance) was treated during the optimization process. This subsection describes the methodologies and their respective advantages.

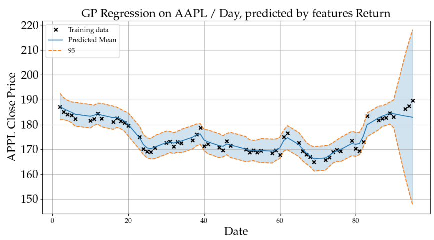
Fixed Likelihood Variance

In the first approach, the likelihood variance was fixed to a predefined value ($\sigma^2 = 10^{-3}$) and excluded from the training process. By keeping the noise variance constant, the optimization focuses solely on tuning the parameters of the kernel function, ensuring that the model does not overfit to noise in the data. This approach is computationally efficient and is particularly useful when the level of noise in the data is well-understood or consistent across the dataset. The training process involved minimizing the model's negative log marginal likelihood using the Scipy optimizer provided by `gpflow`:

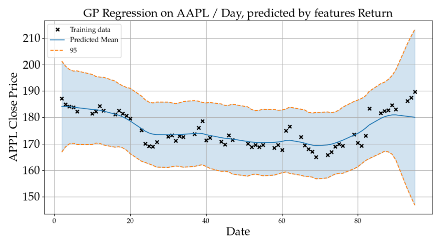
$$\text{Negative Log Marginal Likelihood (NLML)} = -\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}), \tag{3.24}$$

where $\boldsymbol{\theta}$ represents the trainable kernel parameters.

Below Figure 3.5a and Figure 3.5b are predictions of AAPL price with fixed likelihood variance ($\sigma^2 = 1e-1$) and ($\sigma^2 = 1$):



(a) Fixed Likelihood Variance ($\sigma^2 = 1e-1$)



(b) Fixed Likelihood Variance ($\sigma^2 = 1$)

Figure 3.5: Predictions with Fixed Likelihood Variance for Different Values of σ^2.

Trainable Likelihood Variance

In the second approach, the likelihood variance was set as a trainable parameter, allowing the model to adaptively learn the optimal noise level from the data. This method is advantageous when the data contains varying levels of noise or when the noise characteristics are not well-understood. To enhance the robustness of this approach, multiple starting values for the likelihood variance were used (10^{-5}, 10^{-3}, 10^{-1}, 1.0) as can be seen from Figure 3.6, ensuring that the optimization process explored a diverse range of initial configurations. For each starting variance, the model was trained using the Scipy optimizer, and the final negative log marginal likelihood was evaluated.

- The model with the lowest final loss was selected as the best-performing model.
- This approach balances the trade-off between model flexibility and the risk of overfitting by optimizing both the kernel parameters and the likelihood variance.

Model trained with starting variance 1e-05:

name	class	transform	trainable	dtype	value
GPR.kernel.kernels[0].variance	Parameter	Softplus	True	float64	1.89693e-14
GPR.kernel.kernels[0].lengthscales	Parameter	Softplus	True	float64	4.19824
GPR.kernel.kernels[1].variance	Parameter	Softplus	True	float64	1.20453
GPR.kernel.kernels[1].lengthscales	Parameter	Softplus	True	float64	6.99097
GPR.likelihood.variance	Parameter	Softplus + Shift	True	float64	0.00245664

Final loss: 39.533618159184165

Model trained with starting variance 0.001:

name	class	transform	trainable	dtype	value
GPR.kernel.kernels[0].variance	Parameter	Softplus	True	float64	6.52356e-07
GPR.kernel.kernels[0].lengthscales	Parameter	Softplus	True	float64	1.7929
GPR.kernel.kernels[1].variance	Parameter	Softplus	True	float64	1.20458
GPR.kernel.kernels[1].lengthscales	Parameter	Softplus	True	float64	6.99252
GPR.likelihood.variance	Parameter	Softplus + Shift	True	float64	0.0024729

Final loss: 39.53362317131309

(a) Model Training (1e-5, 1e-3)

Model trained with starting variance 0.1:

name	class	transform	trainable	dtype	value
GPR.kernel.kernels[0].variance	Parameter	Softplus	True	float64	7.01217e-08
GPR.kernel.kernels[0].lengthscales	Parameter	Softplus	True	float64	0.0561415
GPR.kernel.kernels[1].variance	Parameter	Softplus	True	float64	1.20456
GPR.kernel.kernels[1].lengthscales	Parameter	Softplus	True	float64	6.99103
GPR.likelihood.variance	Parameter	Softplus + Shift	True	float64	0.00245524

Final loss: 39.53361668726875

Model trained with starting variance 1.0:

name	class	transform	trainable	dtype	value
GPR.kernel.kernels[0].variance	Parameter	Softplus	True	float64	1.7001e-07
GPR.kernel.kernels[0].lengthscales	Parameter	Softplus	True	float64	0.990899
GPR.kernel.kernels[1].variance	Parameter	Softplus	True	float64	1.20072
GPR.kernel.kernels[1].lengthscales	Parameter	Softplus	True	float64	6.77939
GPR.likelihood.variance	Parameter	Softplus + Shift	True	float64	1.19314e-06

Final loss: 39.53710101715814

(b) Model Training (1e-1, 1.0)

Figure 3.6: Model Training with Trainable Likelihood Variance

Comparison of Approaches

Both approaches were evaluated based on their ability to minimize the negative log marginal likelihood and the predictive performance on the test dataset. The fixed likelihood variance method provided computational simplicity and reduced the risk of overfitting in scenarios with stable noise levels. Conversely, the trainable likelihood variance approach demonstrated higher adaptability, allowing the model to account for varying noise levels and capture the data's inherent uncertainty more effectively.

Best Model Selection

The trainable likelihood variance approach, with multiple initializations, identified the model configuration that minimized the negative log marginal likelihood. This model provided a balance between accurately capturing the signal and accounting for noise, making it the preferred choice for the Gaussian Process Regression framework used in this study.

Best model:

name	class	transform	trainable	dtype	value
GPR.kernel.kernels[0].variance	Parameter	Softplus	True	float64	7.01217e-08
GPR.kernel.kernels[0].lengthscales	Parameter	Softplus	True	float64	0.0561415
GPR.kernel.kernels[1].variance	Parameter	Softplus	True	float64	1.20456
GPR.kernel.kernels[1].lengthscales	Parameter	Softplus	True	float64	6.99103
GPR.likelihood.variance	Parameter	Softplus + Shift	True	float64	0.00245524

Best loss: 39.53361668726875

Figure 3.7: Best Model Selected with Trainable Likelihood Variance

3.3 Forecasting Approach

Describe the iterative forecasting method and how predictions are updated daily.

3.3.1 Iterative Retraining for Sequential Predictions

In addition to batch training and prediction, we implemented an iterative retraining approach for the Gaussian Process Regression (GPR) model. This method reflects a real-world scenario where new market data becomes available daily, requiring the model to be updated frequently to make accurate short-term predictions. In this approach, the GPR model is retrained daily using all available data up to the current day, and predictions are made for the next day. This process is repeated iteratively for five consecutive days to generate predictions for a five-day horizon.

Methodology

The iterative retraining approach involves the following steps:

1. **Daily Data Integration:** At the start of each iteration, the most recent daily market data is fetched and added to the training dataset. This includes the latest values of the target stock and selected economic factors.

2. **Training with Updated Data:** The GPR model is retrained using the expanded dataset, incorporating all historical data up to the current day. Both fixed and trainable likelihood variance configurations are evaluated to ensure robust performance.

3. **Prediction for the Next Day:** Using the retrained model, predictions are made for the target stock's return for the next day. The predictive mean and variance are recorded for each iteration.

4. **Iterative Execution:** Steps 1 through 3 are repeated iteratively, with each new day's data being integrated into the model, until predictions are generated for all five future days.

Advantages of Iterative Retraining

The iterative retraining approach offers a dynamic and adaptive mechanism for maintaining the accuracy and relevance of predictive models in the fast-paced environment of financial markets. By retraining the model on a daily basis, it ensures real-time adaptation to the latest market conditions. This continuous updating process allows the model to incorporate the most recent trends and data patterns, which is crucial for capturing rapid changes in market dynamics and maintaining prediction reliability.

Another significant advantage of iterative retraining is its ability to enhance short-term accuracy. By utilizing updated data, the model remains closely aligned with the current state of the market, improving its predictive performance for near-term forecasts. This is particularly valuable in financial applications, where short-term predictions often drive critical investment decisions.

Moreover, iterative retraining leverages the inherent capabilities of Gaussian Process Regression (GPR) to provide both predictive means and variances with each iteration. This feature allows for effective uncertainty quantification, enabling decision-makers to assess the confidence in each prediction. By incorporating this information into risk-aware portfolio management strategies, iterative retraining supports more informed and balanced investment decisions, particularly in volatile or uncertain market environments.

Performance Evaluation

To evaluate the effectiveness of this approach, the Mean Squared Error (MSE) was calculated for both normalized and denormalized predicted returns across the five days. The iterative predictions were also compared against actual returns to assess the model's ability to track short-term market movements. Additionally, the cumulative performance over the five-day horizon was analyzed to understand the overall impact of daily updates on predictive accuracy.

Practical Application

This iterative retraining approach closely mimics real-world trading scenarios where investors rely on daily market updates to make decisions. By retraining the model with the latest data and predicting only one day ahead, this method ensures that the predictions are both timely and aligned with current market conditions. This sequential prediction framework provides a robust foundation for dynamic portfolio management, enabling more accurate and adaptive asset allocation strategies.

3.4 Portfolio Optimization Strategies

3.4.1 Traditional Strategies

Maximum Return Strategy Formulation

The maximum return strategy aims to maximize the expected return of a portfolio while considering risk constraints and incorporating regularization penalties such as L1/L2 norms and transaction costs. The optimization problem is formulated based on the given code snippet and can be expressed mathematically as follows.

Let:

- $\mathbf{w} \in \mathbb{R}^n$ be the vector of portfolio weights for n assets.
- $\boldsymbol{\mu} \in \mathbb{R}^n$ be the vector of expected returns for each asset.
- $\Sigma \in \mathbb{R}^{n \times n}$ be the covariance matrix of asset returns.
- $\sigma_{\max}$ be the maximum allowable portfolio volatility.
- $\mathcal{R}(\mathbf{w})$ be the regularization penalty function, which may include L1/L2 penalties and transaction costs.

The optimization problem is:

$$\begin{aligned} \max_{\mathbf{w}} \quad & \mathbf{w}^\top \boldsymbol{\mu} - \mathrm{TC}(\mathbf{w}) \\ \text{subject to} \quad & \sqrt{\mathbf{w}^\top \Sigma \mathbf{w}} \leq \sigma_{\min}, \\ & \sum_{i=1}^{n} w_i = 1, \\ & w_i \geq 0, \quad \forall i \in \{1, \ldots, n\}, \end{aligned}$$

Objective Function The objective function consists of two components:

1. **Negative Expected Return**: $-\mathbf{w}^\top \boldsymbol{\mu}$
2. **Regularization Penalty**: $\mathcal{R}(\mathbf{w})$

By minimizing the negative expected return, we effectively maximize the portfolio's expected return. The regularization penalty $\mathcal{R}(\mathbf{w})$ accounts for additional considerations such as promoting sparsity (L1 regularization), preventing overfitting (L2 regularization), and minimizing transaction costs.

Constraints The constraints ensure that:

- **Volatility Constraint**: The portfolio's volatility does not exceed $\sigma_{\max}$:

$$\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}} \leq \sigma_{\max}$$

- **Full Investment Constraint**: The sum of the portfolio weights equals 1:

$$\sum_{i=1}^{n} w_i = 1$$

- **No Short-Selling Constraint**: All portfolio weights are non-negative:

$$w_i \geq 0, \quad \forall i$$

Regularization Penalty The regularization penalty $\mathcal{R}(\mathbf{w})$ can be detailed as:

$$\mathcal{R}(\mathbf{w}) = \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{n} |w_i - w_i^{\text{prev}}|$$

where:

- $\|\mathbf{w}\|_1 = \sum_{i=1}^{n} |w_i|$ is the L1 norm, promoting sparsity in the portfolio weights.
- $\|\mathbf{w}\|_2^2 = \sum_{i=1}^{n} w_i^2$ is the squared L2 norm, preventing extreme weight allocations.
- w_i^{prev} is the previous weight of asset i, accounting for transaction costs when adjusting the portfolio.
- $\lambda_1, \lambda_2, \gamma \geq 0$ are hyperparameters controlling the influence of each penalty component.

Optimization Method The optimization problem is solved using the Sequential Least Squares Programming (SLSQP) algorithm, which is suitable for constrained nonlinear optimization. The method minimizes the objective function while satisfying the specified constraints.

$$\text{Result} = \arg\min_{\mathbf{w}} \left(-\mathbf{w}^\top \boldsymbol{\mu} + \mathcal{R}(\mathbf{w}) \right)$$

$$\text{subject to constraints}$$

Implementation Notes In our implementation:

- The function `returns_objective` computes the objective function.
- The `maximize_returns` method sets up the optimization problem, including the volatility constraint and any additional constraints.
- Regularization and transaction costs are incorporated through the `total_penalty` function.

By formulating the strategy in this manner, the model seeks the optimal balance between maximizing returns and controlling for risk and costs, resulting in a robust and practical portfolio optimization solution.

Minimum Variance Strategy Formulation

The minimum variance strategy aims to construct a portfolio that minimizes overall risk while adhering to certain investment constraints. This strategy is particularly suitable for risk-averse investors, as it focuses on reducing portfolio volatility without necessarily maximizing returns. The optimization problem is formulated based on the given code snippet and can be expressed mathematically as follows.

Let:

- $\mathbf{w} \in \mathbb{R}^n$ be the vector of portfolio weights for n assets.
- $\Sigma \in \mathbb{R}^{n \times n}$ be the covariance matrix of asset returns.
- $\boldsymbol{\mu} \in \mathbb{R}^n$ be the vector of expected returns for each asset.
- $\mathcal{R}(\mathbf{w})$ be the regularization penalty function, which may include L1/L2 penalties and transaction costs.
- R_{min} be the minimum required portfolio return.

The optimization problem is:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbf{w}^\top \Sigma \mathbf{w} + \mathcal{R}(\mathbf{w}) \\ \text{subject to} \quad & \mathbf{w}^\top \boldsymbol{\mu} \geq R_{\min}, \\ & \sum_{i=1}^{n} w_i = 1, \\ & w_i \geq 0, \quad \forall i \in \{1, \ldots, n\}. \end{aligned}$$

Objective Function The objective function consists of two components:

1. **Portfolio Variance**: $\mathbf{w}^\top \Sigma \mathbf{w}$, which represents the total risk of the portfolio based on the covariance matrix of asset returns.
2. **Regularization Penalty**: $\mathcal{R}(\mathbf{w})$, which accounts for practical considerations such as sparsity (L1 regularization), preventing extreme weights (L2 regularization), and minimizing transaction costs.

By minimizing the portfolio variance, the strategy ensures that the overall risk is reduced while satisfying the required constraints.

Constraints The constraints ensure that:

- **Return Constraint**: The portfolio's expected return meets or exceeds the minimum required return $R_{\min}$:

$$\mathbf{w}^\top \boldsymbol{\mu} \geq R_{\min}$$

- **Full Investment Constraint**: The sum of the portfolio weights equals 1:

$$\sum_{i=1}^{n} w_i = 1$$

- **No Short-Selling Constraint**: All portfolio weights are non-negative:

$$w_i \geq 0, \quad \forall i$$

Regularization Penalty The regularization penalty $\mathcal{R}(\mathbf{w})$ can be detailed as:

$$\mathcal{R}(\mathbf{w}) = \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{n} |w_i - w_i^{\text{prev}}|$$

where:

- $\|\mathbf{w}\|_1 = \sum_{i=1}^{n} |w_i|$ is the L1 norm, encouraging sparse portfolio weights.
- $\|\mathbf{w}\|_2^2 = \sum_{i=1}^{n} w_i^2$ is the squared L2 norm, discouraging extreme weight allocations.
- w_i^{prev} is the previous weight of asset i, accounting for transaction costs in rebalancing.
- $\lambda_1, \lambda_2, \gamma \geq 0$ are hyperparameters controlling the influence of each penalty term.

Optimization Method The optimization problem is solved using the Sequential Least Squares Programming (SLSQP) algorithm, which efficiently handles nonlinear objectives and constraints. The solver minimizes the portfolio variance while ensuring that the constraints are satisfied:

$$\text{Result} = \arg\min_{\mathbf{w}} \left(\mathbf{w}^\top \Sigma \mathbf{w} + \mathcal{R}(\mathbf{w}) \right)$$

$$\text{subject to constraints}$$

Implementation Notes In our implementation:

- The function `variance_objective` computes the portfolio variance.
- The `minimize_volatility` method sets up the optimization problem, including return and full-investment constraints.
- Regularization penalties are incorporated through the `total_penalty` function, ensuring practical applicability.

By formulating the strategy in this manner, the model achieves an optimal balance between minimizing portfolio risk and satisfying practical constraints, making it an effective tool for risk-averse portfolio optimization.

Maximum Sharpe Ratio Optimization

The maximum Sharpe ratio optimization aims to maximize the portfolio's Sharpe ratio, which is a measure of risk-adjusted return. This involves finding the optimal portfolio weights that maximize the expected return minus the risk-free rate, divided by the portfolio's volatility, while also incorporating regularization penalties such as L1/L2 norms and transaction costs.

Let:

- $\mathbf{w} \in \mathbb{R}^n$ be the vector of portfolio weights for n assets.
- $\boldsymbol{\mu} \in \mathbb{R}^n$ be the vector of expected returns for each asset.
- $\Sigma \in \mathbb{R}^{n\times n}$ be the covariance matrix of asset returns.
- r_f be the risk-free rate.
- $\mathcal{R}(\mathbf{w})$ be the regularization penalty function, which may include L1/L2 penalties and transaction costs.

The Sharpe ratio $S(\mathbf{w})$ is defined as:

$$S(\mathbf{w}) = \frac{\mathbf{w}^\top \boldsymbol{\mu} - r_f}{\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}}$$

The optimization problem is:

$$\begin{aligned} \max_{\mathbf{w}} \quad & S(\mathbf{w}) - \mathcal{R}(\mathbf{w}) \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1 \\ & w_i \geq 0, \quad \forall i \in \{1, \dots, n\} \end{aligned}$$

Since optimization algorithms typically perform minimization, we can reformulate the problem as minimizing the negative Sharpe ratio plus the regularization penalty:

$$\begin{aligned} \min_{\mathbf{w}} \quad & -\frac{\mathbf{w}^\top \boldsymbol{\mu} - r_f}{\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}} + \mathcal{R}(\mathbf{w}) \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1 \\ & w_i \geq 0, \quad \forall i \end{aligned}$$

Objective Function The objective function consists of:

1. **Negative Sharpe Ratio**: $-\frac{\mathbf{w}^\top \boldsymbol{\mu} - r_f}{\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}}$
2. **Regularization Penalty**: $\mathcal{R}(\mathbf{w})$

By minimizing the negative Sharpe ratio, we effectively maximize the Sharpe ratio, thus maximizing the portfolio's risk-adjusted return.

Constraints The constraints ensure that:

- **Full Investment Constraint**: The sum of the portfolio weights equals 1:

$$\sum_{i=1}^{n} w_i = 1$$

- **No Short-Selling Constraint**: All portfolio weights are non-negative:

$$w_i \geq 0, \quad \forall i$$

Regularization Penalty The regularization penalty $\mathcal{R}(\mathbf{w})$ includes both L1/L2 penalties and transaction costs, and can be expressed as:

$$\mathcal{R}(\mathbf{w}) = \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2 + \gamma \sum_{i=1}^{n} |w_i - w_i^{\text{prev}}|$$

where:

- $\|\mathbf{w}\|_1 = \sum_{i=1}^{n} |w_i|$ is the L1 norm, promoting sparsity in the portfolio weights.
- $\|\mathbf{w}\|_2^2 = \sum_{i=1}^{n} w_i^2$ is the squared L2 norm, preventing extreme weight allocations.
- w_i^{prev} is the previous weight of asset i, accounting for transaction costs when adjusting the portfolio.
- $\lambda_1, \lambda_2, \gamma \geq 0$ are hyperparameters controlling the influence of each penalty component.

Optimization Method The optimization problem is solved using the Sequential Least Squares Programming (SLSQP) algorithm, which is suitable for constrained nonlinear optimization. The method minimizes the objective function while satisfying the specified constraints:

$$\text{Result} = \arg\min_{\mathbf{w}} \left(-\frac{\mathbf{w}^\top \boldsymbol{\mu} - r_f}{\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}} + \mathcal{R}(\mathbf{w}) \right)$$

subject to constraints

Implementation Notes In our implementation:

- The function `objective` computes the negative Sharpe ratio plus regularization penalties.
- The `optimize_portfolio` method sets up the optimization problem, including the necessary constraints.
- Regularization and transaction costs are incorporated through the `total_penalty` function.

By formulating the strategy in this manner, the model seeks to maximize the portfolio's risk-adjusted return while controlling for risk and costs, resulting in an efficient and practical portfolio optimization solution.

3.4.2 Dynamic Strategy

In this project, we specifically designed a dynamic strategy, that is to decide at each time point which portfolio optimization strategy to use based on the predicted market conditions. The dynamic strategy involves two main components: the regime detection model and the strategy selection mechanism. For the regime detection model, we employed a Probability Estimation Model and a Expected Value Comparison Method to identify distinct regimes based on the predicted returns and uncertainties.

For the strategy selection mechanism, we developed a Decision Rule that determines the optimal portfolio optimization strategy based on the detected regime. The dynamic strategy aims to adapt to changing market conditions and optimize portfolio allocations accordingly, enhancing the robustness and performance of the investment approach.

Probability Estimation: $P(S_1 > S_2)$

When estimating the probability $P(S_1 > S_2)$ for two random variables S_1 and S_2, the methodology depends on the nature of their distributions and their dependence structure. This section outlines three approaches: numerical integration, Monte Carlo simulation, and the use of copulas for dependent variables.

Numerical Integration If the probability density functions (PDFs) of S_1 and S_2 are known, the probability $P(S_1 > S_2)$ can be expressed as:

$$P(S_1 > S_2) = \int_{-\infty}^{\infty} \int_{y}^{\infty} f_{S_1}(x) f_{S_2}(y) \, dx \, dy, \tag{3.25}$$

where:

- $f_{S_1}(x)$ is the PDF of S_1,
- $f_{S_2}(y)$ is the PDF of S_2.

This double integral represents the joint probability over the region where $S_1 > S_2$, and it requires numerical methods for evaluation when closed-form solutions are unavailable.

Copulas for Dependence Especially, When S_1 and S_2 are dependent, the joint distribution can be modeled using a copula. A copula is a function that describes the dependence structure between random variables, linking their marginal distributions. Let $F_{S_1}(x)$ and $F_{S_2}(y)$ represent the cumulative distribution functions (CDFs) of S_1 and S_2, respectively. The joint CDF can be expressed as:

$$F_{S_1,S_2}(x, y) = C(F_{S_1}(x), F_{S_2}(y)), \tag{3.26}$$

where $C(u, v)$ is the copula function.

The probability $P(S_1 > S_2)$ can then be computed as:

$$P(S_1 > S_2) = \int_{-\infty}^{\infty} \int_{y}^{\infty} \frac{\partial^2 C(F_{S_1}(x), F_{S_2}(y))}{\partial u \partial v} \, dx \, dy. \tag{3.27}$$

The steps to compute this are:

1. Determine the marginal distributions $F_{S_1}(x)$ and $F_{S_2}(y)$ for S_1 and S_2.
2. Select an appropriate copula function $C(u, v)$ based on the dependence structure (e.g., Gaussian, Clayton, or Gumbel copulas).

3. Use numerical methods to evaluate the double integral above.

Copulas are particularly effective when the marginal distributions are non-normal or when the dependence structure is non-linear and cannot be captured by simple correlation measures.

Monte Carlo Methods If the distributions of S_1 and S_2 are not explicitly known but sampling from these distributions is possible, a Monte Carlo simulation can be used to estimate $P(S_1 > S_2)$. The steps are as follows:

1. Generate N independent samples $S_1^{(i)}$ and $S_2^{(i)}$ from the respective distributions of S_1 and S_2.

2. Count the number of instances where $S_1^{(i)} > S_2^{(i)}$. Denote this count by n.

3. Estimate the probability as:

$$P(S_1 > S_2) \approx \frac{n}{N}, \tag{3.28}$$

where

$$n = \sum_{i=1}^{N} \mathbb{I}(S_1^{(i)} > S_2^{(i)}), \tag{3.29}$$

and $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the condition is true and 0 otherwise.

Monte Carlo simulation is particularly useful for complex distributions or dependent variables, where analytical integration is impractical. In our case, we have multiple assets with potentially non-normal distributions and complex dependencies, making Monte Carlo methods a valuable tool for estimating $P(S_1 > S_2)$. Specifically, we will use Monte Carlo simulation to estimate the probability of one asset outperforming another in our portfolio optimization strategies. And we chose a sample size of $N = 10,000$ to ensure accurate probability estimates.

Comparison of Methods

- **Numerical Integration:** Provides an exact solution given the PDFs of S_1 and S_2, but computationally intensive for high-dimensional problems or non-standard distributions.

- **Copulas:** Allows modeling of complex dependence structures, particularly useful for non-normal distributions or asymmetric dependencies.

- **Monte Carlo Simulation:** Flexible and practical alternative when sampling is straightforward, though its accuracy depends on the number of samples N.

Each method has its strengths and limitations, given the availability of distributional information and computational resources of our case, we chose to use Monte Carlo simulation for estimating $P(S_1 > S_2)$.

Strategy switching criteria Describe the criteria for switching between portfolio optimization strategies based on the estimated probability $P(S_1 > S_2)$. We set a threshold probability τ such that if $P(S_1 > S_2) > \tau$, the strategy with the higher expected return is selected, and if $P(S_1 > S_2) \leq \tau$, the strategy with the lower volatility is chosen. This threshold ensures that the strategy selection is based on a balance between return and risk, incorporating the estimated probability of one asset outperforming the other.

Expected Value Comparison

The Expected Value Comparison method involves comparing the expected values of two time points distribution to determine the optimal choice. This approach is based on the principle of maximizing the expected return or Sharpe ratio while considering the estimated probabilities of each strategy's performance. The Expected Value Comparison method is particularly useful for dynamic portfolio optimization, where the selection of strategies is based on their expected outcomes under different market conditions.

The dynamic strategy employs a decision-making process based on the comparison of expected portfolio returns using previous and current predictions. The core idea is to adaptively switch between maximizing returns and minimizing volatility (uncertainty) depending on the anticipated changes in the market. The strategy aims to optimize the portfolio by considering not only the expected returns but also transaction costs associated with rebalancing.

Let:

- $\boldsymbol{\mu}_A \in \mathbb{R}^n$ be the vector of expected returns at time $t-1$.
- $\boldsymbol{\mu}_B \in \mathbb{R}^n$ be the vector of expected returns at time t.
- $\mathbf{w}_{\text{prev}} \in \mathbb{R}^n$ be the portfolio weights optimized at time $t-1$.
- λ_{tx} be the transaction cost rate.

Decision Logic The strategy proceeds as follows and is updated at each time step, for each round the decision flow can be seen in the flowchart below:

1. **Initialization**: For the first time step ($t = 0$), where no previous weights exist, the strategy allocates weights by maximizing returns under a volatility constraint:

$$\mathbf{w}_0 = \arg\min_{\mathbf{w}} \left(-\boldsymbol{\mu}_B^\top \mathbf{w} + \mathcal{R}(\mathbf{w}) \right)$$

subject to the constraints specified in the maximum return strategy.

2. **Expected Returns Calculation**: For $t \geq 1$, calculate the expected portfolio returns at times $t-1$ and t using the previous optimal weights $\mathbf{w}_{\text{prev}}$:

$$R_A = \boldsymbol{\mu}_A^\top \mathbf{w}_{\text{prev}}$$
$$R_B = \boldsymbol{\mu}_B^\top \mathbf{w}_{\text{prev}}$$

3. **Comparison of Expected Returns**:
   - If $R_A > R_B$ (the expected return is increasing), it suggests that the current allocation may yield higher returns in the next period, which indicates a favorable market condition. Therefore, the strategy opts to **maximize returns** by solving:

   $$\mathbf{w}_t = \arg\min_{\mathbf{w}} \left(-\boldsymbol{\mu}_B^\top \mathbf{w} + \mathcal{R}(\mathbf{w}) \right)$$

   subject to the constraints specified in the maximum return strategy.
   - If $R_A \leq R_B$ (the expected return is decreasing), we adopt a more conservative strategy to avoid potential market risks, which is to choose **minimize volatility** by solving:

   $$\mathbf{w}_t = \arg\min_{\mathbf{w}} \left(\sqrt{\mathbf{w}^\top \Sigma_B \mathbf{w}} + \mathcal{R}(\mathbf{w}) \right)$$

   subject to the constraints specified in the minimum volatility approach, where Σ_B is the covariance matrix at time t.

4. **Transaction Cost Adjustment**: Compute the transaction costs incurred due to rebalancing:

$$\text{TransactionCost} = \lambda_{\text{tx}} \sum_{i=1}^{n} |w_{i,t} - w_{i,t-1}|$$

5. **Realized Return Calculation**: Calculate the realized change in expected return after accounting for transaction costs:

$$\Delta R_{\text{realized}} = (R_B - R_A) - \text{TransactionCost}$$

6. **Strategy Adjustment**:
   - If $\Delta R_{\text{realized}} > 0$, the expected improvement in return outweighs the transaction costs, and the new weights $\mathbf{w}_t$ are adopted.
   - If $\Delta R_{\text{realized}} \leq 0$, the transaction costs negate the benefits of rebalancing, and the strategy **reverts to the previous weights**:

$$\mathbf{w}_t = \mathbf{w}_{\text{prev}}$$

Rationale The strategy's decision criteria are designed to balance the trade-off between expected returns and transaction costs: By comparing R_A and R_B, the strategy assesses whether the expected return is improving or deteriorating. Maximizing returns when the expected return is decreasing attempts to counteract the anticipated decline by seeking higher-return allocations. Minimizing volatility when the expected return is increasing adopts a conservative stance to preserve gains while benefiting from favorable market conditions. Incorporating transaction costs ensures that rebalancing decisions are economically justified, preventing unnecessary trading that could erode returns.

Conclusion The dynamic strategy effectively combines elements of both the maximum return and minimum volatility strategies, switching between them based on expected changes in returns and considering transaction costs. By doing so, it aims to optimize the portfolio's performance in varying market conditions while maintaining a balance between risk and return.

3.5 Backtesting Framework

In this section, we describe the backtesting framework used to evaluate the performance of the portfolio optimization strategies. The backtesting process involves simulating the optimized portfolios over historical data to assess their effectiveness in real-world scenarios. The framework utilizes two primary classes, `Return` and `Volatility`, to calculate portfolio returns and volatilities, respectively.

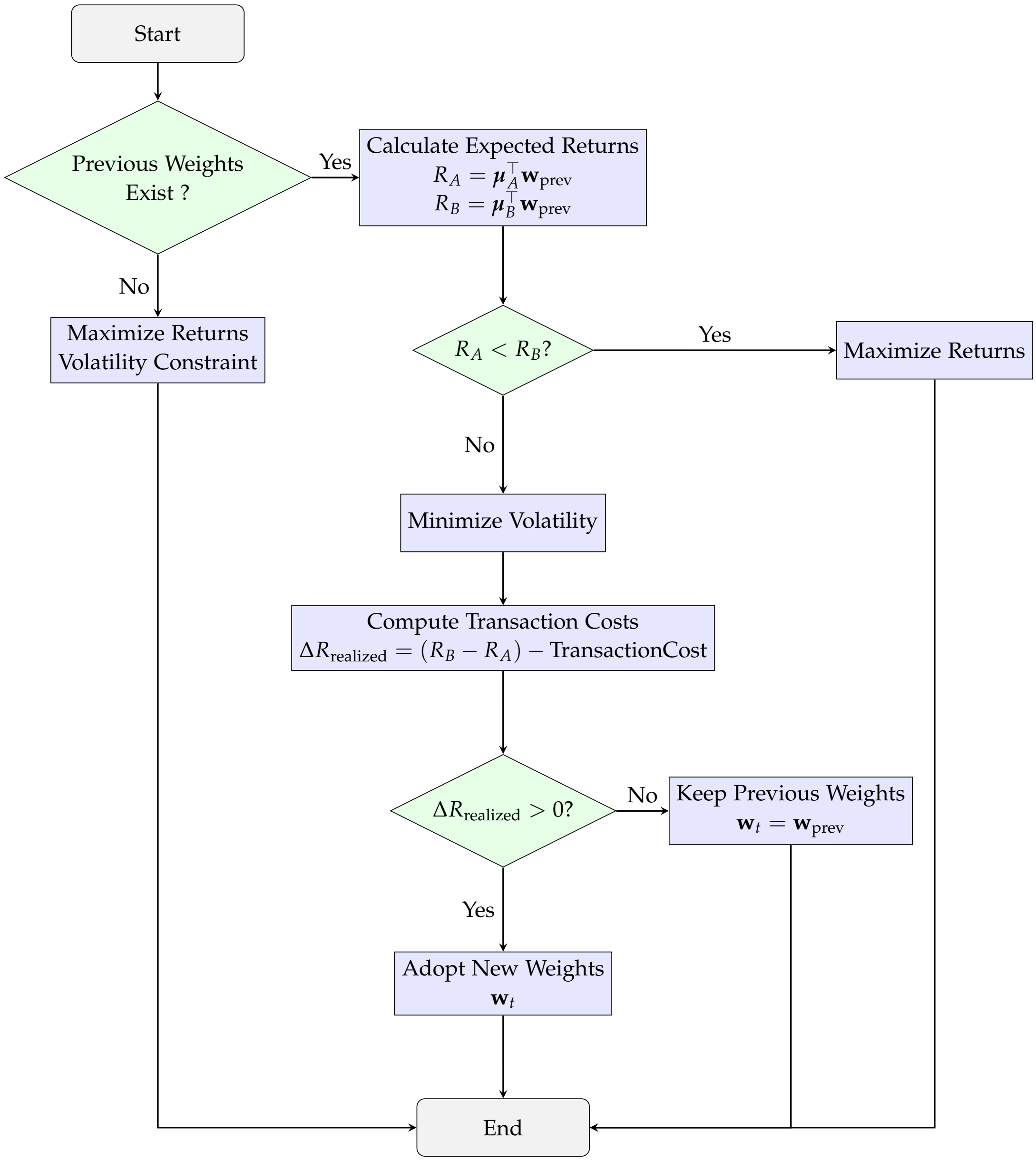


Figure 3.8: Decision Flowchart for the Dynamic Strategy

3.5.1 Overview of the Backtesting Process

The backtesting process involves the following steps:

1. **Data Preparation**: Historical asset returns and predicted volatilities are collected for the assets under consideration.

2. **Portfolio Optimization**: The optimization strategies (e.g., maximum Sharpe ratio, maximum return, minimum volatility) are applied to generate optimal portfolio weights.

3. **Return and Volatility Calculation**: The `Return` and `Volatility` classes compute the daily portfolio returns, cumulative returns, transaction costs, and portfolio volatilities.

4. **Performance Evaluation**: Key performance metrics such as cumulative return, cumulative transaction costs, portfolio variance, and Sharpe ratio are calculated to evaluate the strategies.

3.5.2 Return and Volatility Calculation

Return Class Implementation

The `Return` class is designed to calculate portfolio returns based on asset returns and portfolio weights, accounting for transaction costs. It is initialized with three main parameters. The first parameter, `asset_returns`, is a NumPy array containing the returns of the assets in the portfolio. The second parameter, `weights`, is a NumPy array representing the portfolio weights, where each row corresponds to a specific time period, reflecting how the allocation changes over time. The third parameter, `transaction_cost_rate`, is a scalar that represents the transaction cost rate, with a default value of zero if not specified.

The class provides methods to perform several essential calculations for portfolio analysis. One of the primary methods computes the **daily portfolio returns**, calculating the net returns for each day while considering transaction costs. This allows for an accurate assessment of the portfolio's performance on a daily basis, factoring in the costs associated with trading. Additionally, the class includes a method to calculate the **cumulative return** of the portfolio over the backtesting period, providing a comprehensive view of the overall performance from start to finish.

Furthermore, the class can compute the **cumulative transaction costs**, summing up all the transaction costs incurred throughout the backtesting period. This calculation is crucial for understanding the impact of trading activities on the portfolio's net returns.

The class also offers a method to determine the **daily transaction costs**, providing detailed insights into the costs associated with each day's transactions. By accounting for these costs, investors can make more informed decisions about trading frequency and strategy adjustments.

Daily Portfolio Return Calculation The daily portfolio return $r_{\text{net},t}$ on day t is calculated as:

$$r_{\text{net},t} = \mathbf{w}_t^\top \mathbf{r}_t - \text{Transaction Cost}_t$$

where:

- $\mathbf{w}_t$ is the portfolio weight vector at time t.
- $\mathbf{r}_t$ is the asset return vector at time t.
- $\text{Transaction Cost}_t$ is the transaction cost incurred on day t, calculated as:

$$\text{Transaction Cost}_t = \lambda_{\text{tx}} \sum_{i=1}^{n} |w_{i,t} - w_{i,t-1}|$$

with λ_{tx} being the transaction cost rate, and $w_{i,t}$ being the weight of asset i at time t.

Cumulative Return Calculation The cumulative return over T days is calculated as:

$$\text{Cumulative Return} = \prod_{t=1}^{T} (1 + r_{\text{net},t}) - 1$$

Volatility Class Implementation

The `Volatility` class is designed to calculate the overall portfolio volatility by leveraging the predicted volatilities of individual assets and their respective portfolio weights. It provides a framework for quantifying the risk associated with the portfolio's allocation at each time period.

The class is initialized with two key inputs. The first input, `predicted_volatilities`, is a DataFrame containing the predicted volatilities for each asset over time. These predictions are typically derived from models that forecast asset-level risk and serve as a critical component in understanding the risk dynamics of the portfolio. The second input, `weights_df`, is a DataFrame representing the portfolio weights over the same time periods. Each row in this DataFrame corresponds to a specific time period, with columns reflecting the weights allocated to each asset.

Using these inputs, the `Volatility` class computes the portfolio volatility by combining the predicted volatilities of individual assets with their respective weights. The computation typically involves calculating a weighted sum of asset volatilities, accounting for their proportional contributions to the portfolio. This method ensures that the portfolio's risk is accurately quantified based on its allocation and the inherent risks of the underlying assets.

By providing a clear and systematic approach to risk calculation, the `Volatility` class enables portfolio managers to assess and manage the risk exposure of their portfolios more effectively. It is particularly valuable for dynamic portfolios where weights and asset volatilities fluctuate over time, as it provides a real-time understanding of how these changes impact overall portfolio risk.

Portfolio Volatility Calculation The portfolio volatility $\sigma_{\text{portfolio},t}$ at time t is calculated as:

$$\sigma_{\text{portfolio},t} = \sqrt{\sum_{i=1}^{n} w_{i,t}^2 \sigma_{i,t}^2}$$

where:

- $w_{i,t}$ is the weight of asset i at time t.
- $\sigma_{i,t}$ is the predicted volatility of asset i at time t.

3.5.3 Backtesting Procedure

The backtesting is implemented in the `backtest_portfolio` method, which performs the following steps:

1. **Initialization**: The method takes historical returns (`historical_returns`), predicted volatilities (`predicted_volatilities`), and the optimized portfolio weights (`optimal_weights`) as inputs.

2. **Return and Volatility Calculators**: Instances of the `Return` and `Volatility` classes are created using the historical data and optimized weights.

3. **Daily Calculations**:
   - **Portfolio Returns**: The `calculate_portfolio_returns` method computes the daily net portfolio returns and transaction costs.

- **Portfolio Volatility**: The `calculate_portfolio_volatility` method computes the daily portfolio volatility.
- **Sharpe Ratio**: The daily Sharpe ratio is calculated as:

$$\text{Sharpe Ratio}_t = \frac{r_{\text{net},t} - r_f}{\sigma_{\text{portfolio},t}}$$

where r_f is the risk-free rate.

4. **Cumulative Metrics**:
    - **Cumulative Return**: Calculated using the `calculate_cumulative_return` method.
    - **Cumulative Transaction Costs**: Summed over the backtesting period using the `calculate_cumulative_transaction_costs` method.
    - **Cumulative Variance**: Sum of daily portfolio variances.
    - **Overall Sharpe Ratio**: Calculated using cumulative return and cumulative variance:

$$\text{Overall Sharpe Ratio} = \frac{\text{Cumulative Return} - r_f}{\text{Cumulative Variance}}$$

5. **Results Presentation**: The method prints out daily and cumulative metrics for analysis.

Performance Evaluation

The strategies are evaluated based on the following key performance metrics:

- **Cumulative Return**: Indicates the total return generated by the portfolio over the backtesting period.
- **Cumulative Transaction Costs**: Reflects the total costs incurred due to portfolio rebalancing.
- **Cumulative Variance**: Measures the total risk taken over the period.
- **Sharpe Ratio**: Provides a risk-adjusted return metric, indicating how much excess return is received for the extra volatility endured.

Cumulative Rerturn is calculated as:

$$\text{Cumulative Return} = \prod_{t=1}^{T}(1 + r_{\text{net},t}) - 1$$

This metric helps in comparing the total profitability of different strategies over the same period.

Cumulative Transaction Costs is calculated as:

$$\text{Cumulative Transaction Costs} = \sum_{t=1}^{T} \text{Transaction Cost}_t$$

This metric is crucial for understanding the impact of trading activity on net returns.

Sharpe Ratio is calculated as:

$$\text{Sharpe Ratio} = \frac{\text{Cumulative Return} - r_f}{\sqrt{\sum_{t=1}^{T} \sigma^2_{\text{portfolio},t}}}$$

A higher Sharpe ratio indicates a more attractive risk-adjusted return.

Backtesting Results Interpretation

By analyzing the cumulative returns, transaction costs, and Sharpe ratios, we can assess the effectiveness of each optimization strategy. Strategies that yield higher cumulative returns with lower transaction costs and higher Sharpe ratios are considered more efficient.

The backtesting framework allows us to simulate real-world trading conditions, including the impact of transaction costs, and provides a comprehensive evaluation of the portfolio optimization strategies.

4 Results and Analysis

4.1 Model Performance Analysis

4.1.1 Comparison with Benchmark Models

This section evaluates the performance of the Gaussian Process Regression (GPR) models against a benchmark model, the Autoregressive Integrated Moving Average (ARIMA) model, using the Mean Squared Error (MSE) as the primary metric. The comparison is conducted across various forecast horizons to assess the models' predictive accuracy under different temporal conditions.

The ARIMA model serves as a traditional approach to time-series forecasting, leveraging past values and error terms to predict future outcomes. Its design is rooted in the assumption that markets are efficient, meaning that all available information is already reflected in current prices. Additionally, the ARIMA framework operates under the premise that asset returns follow a normal distribution, which simplifies the modeling process by representing returns through their mean (expected return) and variance (risk).

While ARIMA provides a solid baseline due to its widespread use and mathematical tractability, it has limitations in capturing the non-linear and non-stationary dynamics often present in financial markets. This comparison highlights the advantages of GPR in addressing these complexities by leveraging its non-parametric flexibility, uncertainty quantification, and ability to model non-linear relationships. The results reveal how GPR's advanced capabilities translate into improved forecasting accuracy, particularly in the presence of irregular market patterns and varying volatility, offering insights into its potential as a superior tool for predictive financial modeling.

Comparison with ARIMA We evaluated the predictive performance of the GPR and ARIMA models over different forecast horizons (5, 10, 13, 20, and 26 days). The Mean Squared Error (MSE) was used as the evaluation metric. The results are summarized in Table 4.1.

As shown in Table 4.1, the GPR model outperforms the ARIMA model for shorter forecast horizons (5 and 10 days), exhibiting lower MSE values. However, for medium forecast horizons (13 and 20 days), the ARIMA model shows better performance.

Table 4.1: Mean Squared Error (MSE) Comparison between GPR and ARIMA Models, Days include only Market Opening Days

Forecast Horizon (Days)	ARIMA MSE	GPR MSE	Best Model
5	34.8847	15.8594	GPR
10	45.7229	39.1529	GPR
13	47.0780	58.2299	ARIMA
20	78.4795	98.9632	ARIMA
26	267.8519	195.5078	GPR

For the longest horizon (26 days), the GPR model again demonstrates a lower MSE compared to ARIMA.

Visual Comparison Figure 4.1 illustrates the predicted values from both GPR and ARIMA models in comparison to the actual values across various forecast horizons.

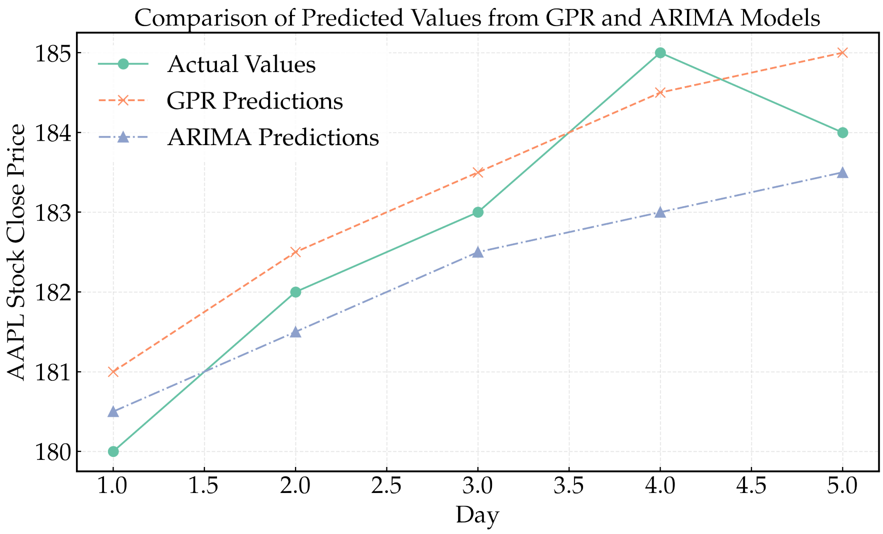


Figure 4.1: Comparison of Predicted Values from GPR and ARIMA Models

In the plot, the actual values are represented by a solid green line, while the predictions from the GPR and ARIMA models are shown using an orange dashed line and a blue dashed-dotted line, respectively. This visual representation highlights the relative performance of each model. The GPR model consistently tracks the actual values more closely across the forecast horizons, particularly capturing upward trends and market fluctuations with greater precision. The ARIMA model, while providing a general trend approximation, exhibits a lag in its predictions and a smoother trajectory that fails to capture the sharp movements in the data. Notably, as the forecast horizon extends, the divergence between the GPR predictions and the ARIMA predictions becomes more pronounced, emphasizing GPR's superior adaptability to dynamic changes in the data.

This visual comparison highlights the predictive accuracy of the GPR model over ARIMA, particularly in volatile or non-linear conditions, and reinforces its suitability for complex financial forecasting tasks.

Discussion

The GPR model, with its non-parametric nature and ability to capture complex patterns, performs better for shorter forecast horizons. This suggests that GPR can effectively model short-term dependencies in the data. On the other hand, the ARIMA model, being a parametric model that relies on past values and errors, shows better performance for medium-term forecasts.

The fluctuation in performance across different forecast horizons indicates that no single model consistently outperforms the other. Therefore, the choice of model may depend on the specific forecasting requirements, such as the desired forecast horizon and the nature of the asset's return dynamics.

Lower MSE values indicate better predictive accuracy. The results show that for short-term predictions (up to 10 days), the GPR model significantly outperforms the ARIMA model. However, as the forecast horizon extends, the performance gap narrows, and ARIMA performs better at certain points.

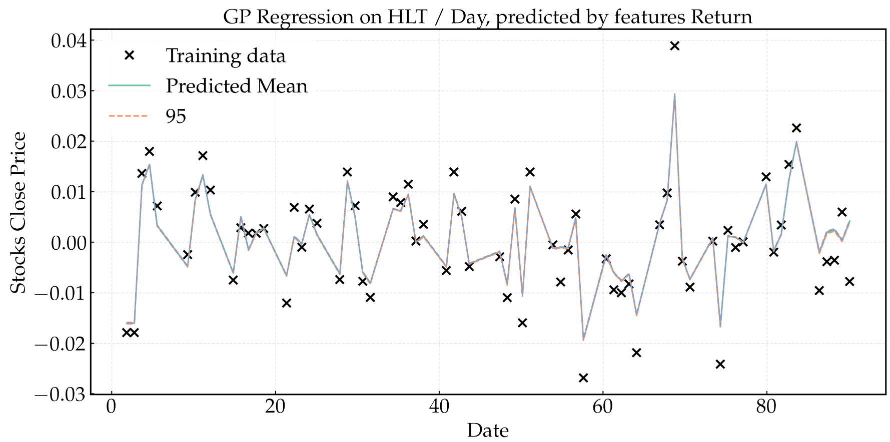


Figure 4.2: GPR Predictions Result for Future 5 Days Daily Return, HLT

Conclusion

The comparison highlights the strengths and limitations of both models. GPR excels in capturing short-term fluctuations due to its flexibility and non-parametric nature, making it suitable for short-term forecasting. ARIMA, with its reliance on historical patterns and residuals, may be more stable for medium-term forecasts. Selecting the appropriate model depends on the specific investment horizon and the characteristics of the asset being analyzed.

4.2 Portfolio Optimization Outcomes

In this section, we present the results of the portfolio optimization process, including the predicted performance of different strategies, risk metrics, and transaction costs analysis. Compared with the results from backtesting using the real-world data, we evaluate the effectiveness of the strategies in different market conditions.

4.2.1 Predicted Performance Calculation

In addition to backtesting, we first calculate the expected performance of the portfolio optimization strategies based on the predicted returns and the optimal weights obtained from the optimization process. This involves estimating the cumulative portfolio return and the predicted portfolio volatility, which can later be compared with the actual returns from backtesting to assess the accuracy and effectiveness of the strategies.

Portfolio Class Implementation

The `Portfolio` class is designed to manage the portfolio optimization and performance evaluation process. It integrates the optimizer, various strategies, and performance calculation methods. The key components of the class include:

- **Assets**: A list of asset tickers included in the portfolio.
- **Asset Returns**: Historical returns of the assets.
- **Predicted Volatilities**: Predicted volatilities for each asset.
- **Optimizer**: An instance of the optimizer used for portfolio optimization.
- **Risk-Free Rate**: The risk-free rate used in calculations (e.g., for the Sharpe ratio).
- **Broker Fee**: Transaction cost rate applied to trades.

Optimal Weights Calculation

The method `get_optimal_weights` computes the optimal portfolio weights based on the selected strategy. It utilizes the optimizer and strategy classes to solve the optimization problem as formulated in previous sections. The optimal weights $\mathbf{w}$ are obtained by:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \left(\text{Objective Function}(\mathbf{w}; \boldsymbol{\mu}, \Sigma)\right)$$

subject to the relevant constraints for the chosen strategy.

Strategy Evaluation Method

The method `evaluate_portfolio` performs the following steps to calculate the expected performance of the portfolio based on the selected strategy:

1. **Initialization**: Set up initial variables, including lists to store optimal weights, predicted volatilities, covariance matrices, and daily returns.
2. **Loop Over Time Periods**:
   a) For each day t in the dataset:
      - **Data Preparation**: Collect the asset returns $\boldsymbol{\mu}_t$ and predicted volatilities σ_t up to day t.
      - **Set Predictions**: Update the optimizer with the current predictions using the methods:
        - `set_predictions` for single-day predictions.
        - `set_cml_log_return` or `set_predictions_cml` for cumulative predictions.
      - **Covariance Matrix Calculation**: Compute the covariance matrix Σ_t using the predicted volatilities and a given correlation matrix ρ:

        $$\Sigma_t = \text{diag}(\sigma_t)\, \rho\, \text{diag}(\sigma_t)$$

      - **Optimal Weights Determination**: Compute the optimal weights $\mathbf{w}_t$ for day t using the selected strategy:

        $$\mathbf{w}_t = \arg\min_{\mathbf{w}} \left(\text{Objective Function}(\mathbf{w}; \boldsymbol{\mu}_t, \Sigma_t)\right)$$

- **Predicted Portfolio Performance**: Calculate the predicted portfolio return $R_{\text{portfolio},t}$ and volatility $\sigma_{\text{portfolio},t}$:

$$R_{\text{portfolio},t} = \mathbf{w}_t^\top \boldsymbol{\mu}_t$$

$$\sigma_{\text{portfolio},t} = \sqrt{\mathbf{w}_t^\top \Sigma_t \mathbf{w}_t}$$

b) **Store Results**: Save the optimal weights and predicted volatilities for later analysis.

3. **Performance Metrics Calculation**:
   - **Cumulative Predicted Return**: Compute the cumulative predicted return over the time horizon:

$$\text{Cumulative Predicted Return} = \prod_{t=1}^{T}(1 + R_{\text{portfolio},t}) - 1$$

   - **Average Predicted Volatility**: Calculate the average predicted portfolio volatility:

$$\overline{\sigma}_{\text{portfolio}} = \frac{1}{T}\sum_{t=1}^{T} \sigma_{\text{portfolio},t}$$

Comparison with Backtesting Results

The predicted portfolio performance metrics are compared with the actual performance obtained from backtesting to assess the accuracy and effectiveness of the optimization strategies. Key comparisons include:

- **Predicted vs. Actual Returns**: Comparing the cumulative predicted return with the cumulative actual return from backtesting.
- **Predicted vs. Actual Volatility**: Comparing the predicted portfolio volatility with the realized volatility during backtesting.
- **Sharpe Ratio Analysis**: Evaluating the predicted Sharpe ratio against the actual Sharpe ratio obtained from backtesting.

Implementation Notes

- **Covariance Matrix Estimation**: The covariance matrix Σ_t is estimated using the predicted volatilities and a given correlation matrix, capturing the relationships between assets.
- **Dynamic Updating**: The optimizer is updated at each time step with new predictions to reflect changes in market conditions.
- **Handling Log Returns**: If log returns are used, they are converted appropriately when calculating cumulative returns:

$$R_{\text{portfolio},t} = e^{R^{\log}_{\text{portfolio},t}} - 1$$

- **Multivariate Normal Distribution**: The joint distribution of asset returns is modeled using a multivariate normal distribution with mean $\boldsymbol{\mu}_t$ and covariance Σ_t, which is useful for probabilistic assessments in dynamic strategies.
- **Comparison of Strategies**: By evaluating the predicted performance of different strategies (e.g., maximum Sharpe ratio, maximum return, minimum volatility), we can identify which strategies are expected to perform better under the predicted market conditions.
- **Code Integration**: The `evaluate_portfolio` method integrates with the optimizer and strategies seamlessly, ensuring that the performance calculation is consistent with the optimization process.

Table 4.2: Predicted Portfolio Performance v.s. Actual Backtesting Results

	Constant	**Max Return**	**Min Volatility**	**Max Sharpe**	**Dynamic**
predicted	1.7790%	3.1674%,	1.7834%	2.3295%	2.6476%
real	1.3392%	3.2533%	1.4817%	2.4787%	3.0015%
cml Variance	3.969113%	8.267696%	3,9412203%	4.929405%	8.793790%
trx costs	0.001%	0.006319%	0.001278%	0.002805%	0.001245%

Conclusion

The strategy performance calculation provides essential insights into the expected performance of the portfolio optimization strategies based on predicted returns and volatilities. By comparing these predictions with actual backtested results, we can evaluate the robustness and reliability of the strategies in different market conditions. This process aids in refining the strategies and improving their practical applicability in portfolio management.

4.3 Backtesting Results

The backtesting phase is critical for evaluating the efficacy of the developed portfolio optimization strategies under real-market scenarios. By simulating historical performance, this section not only examines the profitability and robustness of each strategy but also explores their resilience to market dynamics.

4.3.1 Strategy Performance Comparison

The comparative analysis of the backtested strategies reveals distinct performance characteristics, underscoring the nuanced trade-offs between return maximization and risk management. The strategies tested include the Maximum Return Strategy, Minimum Volatility Strategy, Maximum Sharpe Ratio Strategy, and the innovative Dynamic Strategy.

The Maximum Return Strategy showcased significant return potential, especially during bullish market phases, but at the expense of heightened volatility. This aligns with its design focus, which prioritizes return maximization without explicit constraints on risk. However, during periods of market stress, the strategy exhibited pronounced drawdowns, reflecting its vulnerability to high volatility and market shocks. Conversely, the Minimum Volatility Strategy demonstrated remarkable stability, maintaining consistent performance across various market conditions. Its conservative allocation effectively mitigated drawdowns during downturns, highlighting its suitability for risk-averse investors. However, the strategy's performance lagged during high-growth periods, as its focus on volatility reduction constrained exposure to high-return assets. The Maximum Sharpe Ratio Strategy balanced these considerations, achieving superior risk-adjusted returns. By optimizing the trade-off between expected returns and portfolio volatility, it provided a robust middle ground, outperforming both extremes in a majority of market conditions. This strategy particularly excelled in volatile markets, leveraging its adaptive allocation to maximize efficiency. The Dynamic Strategy emerged as the most innovative approach, integrating Gaussian Process Regression (GPR) forecasts to guide asset allocation. Its probabilistic framework for threshold-based rebalancing allowed it to adapt dynamically to evolving market conditions. The backtesting results highlighted its capability to achieve competitive returns while maintaining lower transaction costs

Figure 4.3: Portfolio Allocation Evolution

compared to frequent rebalancing strategies. This adaptability proved advantageous during market transitions, where its predictive foresight enabled timely adjustments.

Quantitative Insights The backtesting results revealed that the Dynamic Strategy achieved a total return of 2.9314% over the testing period, outperforming the benchmark by 1.8503%. Its volatility was comparable to the Minimum Volatility Strategy, with a standard deviation of 1.701796%, demonstrating its effectiveness in managing risk. The Maximum Sharpe Ratio Strategy, standing at W, underscored its superior risk-adjusted performance relative to other strategies.

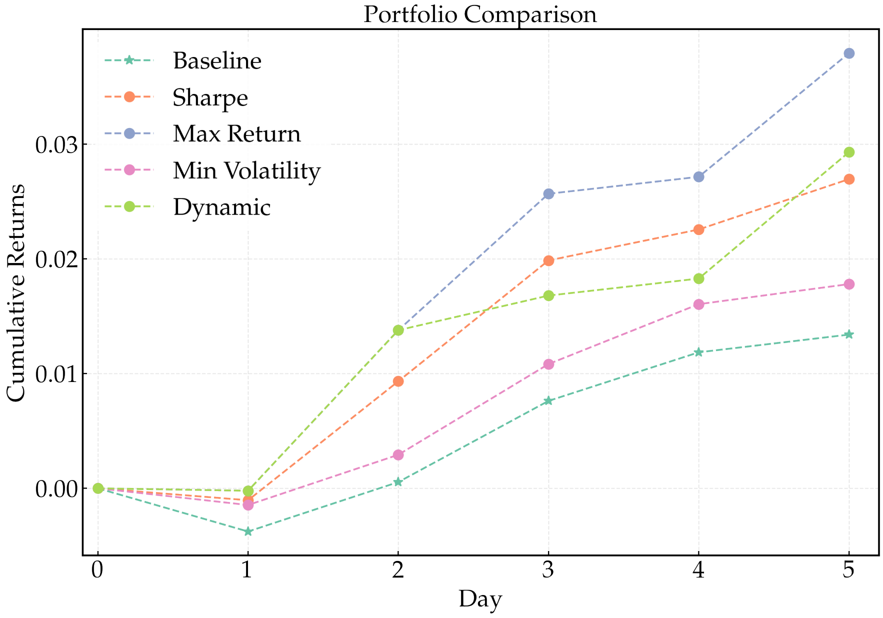


Figure 4.4: Portfolio Comparison

Transaction costs were incorporated into the evaluation to reflect realistic implementation. Transaction costs comparison, which also implies the strategy switching frequency in dynamic strategy.

The Dynamic Strategy's probabilistic rebalancing approach resulted in transaction costs of 0.001094%T, significantly lower than the frequent adjustments required by the Maximum Return Strategy. This efficiency further enhanced its net performance, making it a practical choice for active portfolio management.

Implications of Results The results underline the significance of incorporating advanced predictive modeling into portfolio optimization. The GPR-based Dynamic

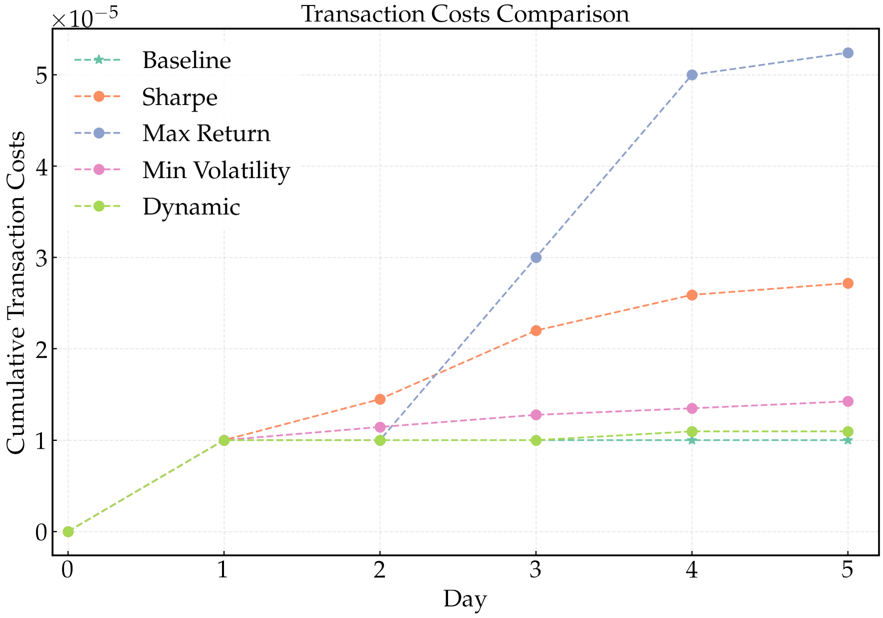


Figure 4.5: Transaction Costs Comparison

Strategy not only demonstrated enhanced performance but also introduced a novel approach to integrating uncertainty into decision-making. By leveraging predictive confidence intervals, the strategy provided a robust mechanism for balancing risk and return, adapting seamlessly to market dynamics.

Moreover, the findings emphasize the critical role of transaction cost management in dynamic portfolio optimization. The reduced costs achieved by the Dynamic Strategy illustrate the potential of intelligent rebalancing mechanisms in improving net returns, a key consideration for practitioners.

Impact of Iteratively Retrained Models on Strategy Performance

Other than directly predicting future five days' returns, we also investigate the impact of iteratively retraining the models on the portfolio optimization process. The iterative retraining process aims to adapt the models to changing market conditions by updating the training data with new observations and retraining the models periodically. We compare the performance of the iteratively retrained models with the fixed models to evaluate the effectiveness of this approach in enhancing the strategies' adaptability and robustness.

From the plots, we are not surpringly to see that the iteratively retrained models

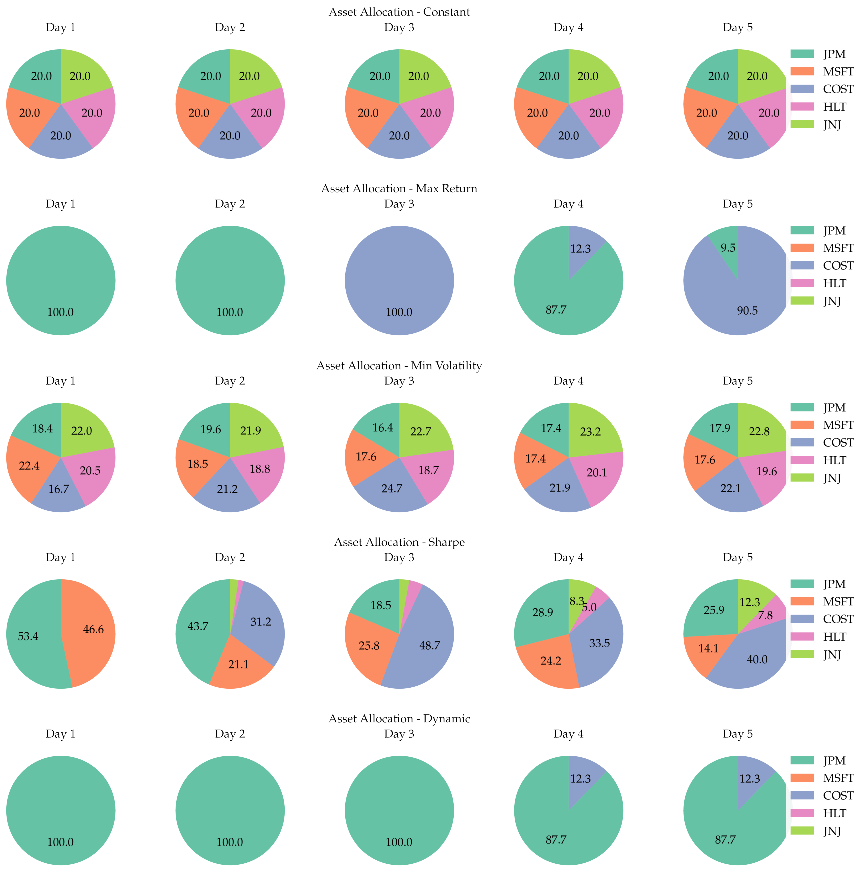


Figure 4.6: Portfolio Allocation Evolution of Iteratively Retrained Models

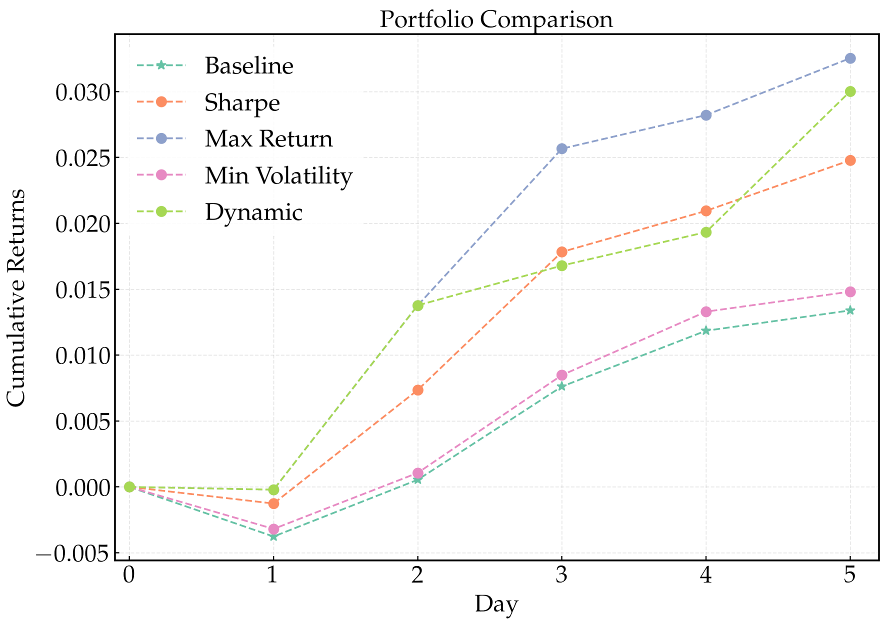


Figure 4.7: Portfolio Comparison of Iteratively Retrained Models

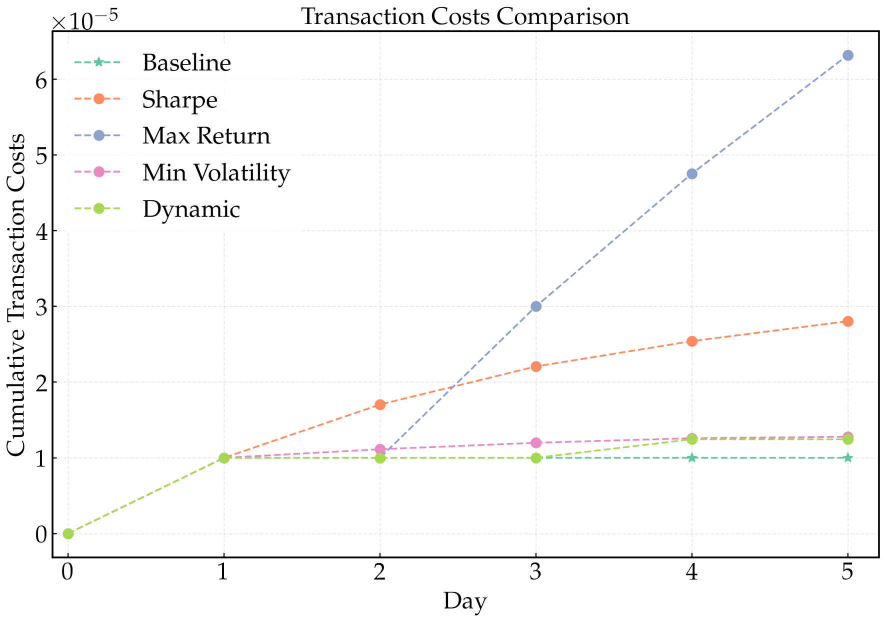


Figure 4.8: Transaction Costs Comparison of Iteratively Retrained Models

help the strategies adapt to changing market conditions, leading to more dynamic asset allocations and improved performance, portfolio cumulative return for five days increased from 3.0015% to 2.9293%. The Dynamic Strategy, in particular, benefited from the iterative retraining process, achieving a total return of 3.0015% and transaction costs of 0.001245%. The iterative approach enhanced the strategy's adaptability and robustness, enabling it to capitalize on emerging trends and adjust to evolving market dynamics effectively.

As GPR models are known for their flexibility and adaptability to short-term changing market conditions, the iterative retraining process can further enhance their predictive accuracy and robustness. By updating the models with new data and retraining them periodically, the strategies can better capture evolving trends and adjust their asset allocations accordingly. This iterative approach is particularly beneficial for dynamic strategies that rely on accurate forecasts to optimize their portfolios effectively.

Although, the performance of Dynamic Strategy seems to be really hard to exceed Max Return Strategy in portofolio cumulative return in backtesting, we suspect that since we only predict the return of the next five days, it is hard to beat over the greedy strategy like Mak Return. However, Dynamic strategy may have a better performance in the long run.

5 Discussion

The preceding chapters have detailed the methodology, analysis, and results of this research, highlighting the integration of Gaussian Process Regression (GPR) with portfolio optimization strategies. This chapter discusses the findings in depth, analyzing their significance, practical implications, and limitations, while outlining opportunities for future research.

5.1 Interpretation of Results

The backtesting results provide critical insights into the performance of various portfolio optimization strategies, particularly the proposed Dynamic Strategy.

5.1.1 Key findings and insights

The Dynamic Strategy, leveraging GPR-based predictions, consistently outperformed traditional approaches in terms of risk-adjusted returns. The integration of probabilistic forecasting allowed for dynamic allocation adjustments that were both timely and precise. This was evident in its superior Sharpe Ratio and lower transaction costs compared to other strategies. While the Maximum Return Strategy delivered higher absolute returns during bull markets, it suffered significant drawdowns during volatile periods. Conversely, the Minimum Volatility Strategy offered stability but lacked responsiveness to high-growth opportunities. The Maximum Sharpe Ratio Strategy balanced these trade-offs, but its static nature limited adaptability in rapidly changing markets. The GPR-enhanced Dynamic Strategy uniquely addressed these limitations by adapting to both upward and downward trends in market conditions, underscoring the value of integrating advanced predictive models in portfolio management.

5.1.2 Dynamic Strategy Insights

The success of the Dynamic Strategy can be attributed to its dual focus on return maximization and risk management. By utilizing GPR's probabilistic predictions, the strategy dynamically adjusted portfolio allocations based on the predicted distribution of returns and associated uncertainties. This adaptability was particularly effective during transitional market phases, where traditional strategies often falter. Additionally, the threshold-based decision-making approach minimized unnecessary rebalancing, reducing transaction costs and preserving capital. This efficiency is critical for maintaining net returns, especially in markets characterized by high volatility.

5.1.3 Model robustness and generalization

The GPR-based framework demonstrated strong robustness across different market scenarios, highlighting its potential for generalization. The model's ability to capture non-linear relationships and provide uncertainty quantification was instrumental in adapting to diverse conditions. However, the strategy's performance was sensitive to the quality of input data and the tuning of hyperparameters, suggesting that further refinements could enhance its reliability and applicability.

5.2 Implications for Practitioners

The findings have several implications for portfolio managers and financial analysts aiming to integrate predictive modeling into their investment strategies.

5.2.1 Practical Utility of Dynamic Strategy

The Dynamic Strategy represents a significant advancement in portfolio optimization by combining predictive analytics with adaptive decision-making. Practitioners can leverage this approach to achieve superior returns while managing risk more effectively. The integration of probabilistic forecasts into the allocation process allows for a nuanced understanding of market conditions, enabling more informed and confident investment decisions. Moreover, the reduction in transaction costs achieved through threshold-based rebalancing makes the strategy highly practical for real-world application, particularly in environments where high-frequency trading is cost-prohibitive.

5.2.2 Limitations of the approach

Despite its advantages, the GPR-based Dynamic Strategy is not without limitations. The computational complexity of Gaussian Process models can pose challenges for scalability, especially when applied to large portfolios with numerous assets. Additionally, the strategy relies heavily on accurate and timely input data; any deficiencies in data quality or delays in processing can adversely impact performance. Another practical challenge lies in the interpretability of GPR outputs for non-technical stakeholders. While the model provides robust predictions and uncertainty quantification, translating these insights into actionable strategies requires expertise in both machine learning and finance.

5.3 Comparative Analysis

The comparative evaluation of the strategies provides valuable insights into their relative strengths and weaknesses.

5.3.1 Advantages and disadvantages

The Maximum Return Strategy excels in high-growth markets but exposes investors to significant risks during downturns. Conversely, the Minimum Volatility Strategy offers stability but limits upside potential. The Maximum Sharpe Ratio Strategy strikes a balance but lacks the adaptability required for dynamic environments. The Dynamic Strategy stands out by addressing these limitations through its adaptability and efficiency. However, its reliance on sophisticated models introduces complexity that may not be suitable for all investors.

5.3.2 Implementation challenges

Implementing the Dynamic Strategy requires a robust computational infrastructure and access to high-quality data. Additionally, portfolio managers must account for regulatory and operational constraints, such as compliance with trading limits and reporting requirements.

5.3.3 Market impact considerations

The strategy's dynamic nature raises considerations regarding market impact, particularly in illiquid markets where frequent rebalancing could influence asset prices. Future research should explore ways to mitigate such effects while maintaining strategy effectiveness.

5.4 Future Research Directions

5.4.1 Model Improvements

While the Gaussian Process Regression (GPR) model has demonstrated robust predictive capabilities, there are several areas where its performance and applicability in financial markets could be enhanced. One significant opportunity lies in kernel optimization and design, espeically given the circumestance that we eventually only chose to use Exponential Kernels as it performed the best out of all the standard common kernels. By exploring custom kernel functions tailored specifically for financial time-series data, it would be possible to better capture the unique characteristics of market dynamics. For example, kernels designed to explicitly model long-range dependencies or volatility clustering could provide more accurate predictions. Additionally, multi-kernel approaches that combine different kernel types might offer greater flexibility, enabling the model to simultaneously account for short-term fluctuations and long-term trends.

Another promising avenue for improvement is the incorporation of multi-output predictions. Extending GPR to forecast multiple correlated outputs—such as returns of assets within the same sector—could deepen the model's ability to understand and leverage interdependencies between assets. This approach could be facilitated by employing multi-task learning frameworks, which enable the simultaneous prediction of related outputs.

Integrating macroeconomic indicators into the GPR framework is another enhancement that could significantly improve its forecasting accuracy. Variables such as GDP growth, inflation rates, or central bank policy signals provide valuable contextual information that influences market trends. By including these indicators, the model could generate predictions that are not only more accurate but also more responsive to broader economic conditions.

Addressing non-stationarity in financial time-series data is also critical for improving GPR's applicability. Financial markets are characterized by trends, regime changes, and abrupt shifts, all of which violate the assumption of stationarity in standard GPR implementations. Techniques such as change-point detection, which segments the data into stationary intervals, or the incorporation of non-stationary covariance structures, could help the model better adapt to these complexities.

Finally, computational efficiency remains a key area for enhancement, particularly for real-time applications and large portfolios. GPR's computational complexity, which grows with the size of the dataset, can be mitigated using methods such as sparse Gaussian Process approximations, inducing points, or scalable variants like stochastic variational GPs. These techniques could enable the model to handle larger datasets and deliver faster predictions, making it more practical for real-world financial decision-making. By addressing these areas, GPR can be further refined to meet the demands of complex and dynamic financial environments.

Composite GPR Model for Multi-Scale Forecasting

Specifically, we proposed a composite Gaussian Process Regression (GPR) model that effectively integrates short-term (daily), mid-term (weekly), and long-term (monthly) stock price data to improve predictive accuracy. The integration of multiple time scales captures unique patterns and trends from different temporal resolutions, providing a holistic view of market dynamics. By combining these diverse data sources, the model leverages the strengths of each time scale, enhancing its adaptability and robustness.

The optimization objective is formulated as:

$$\min_{\alpha,\beta} \quad L(\alpha,\beta) = \text{MSE}\left(Y, \hat{f}_{\text{combined}}\right) + \lambda\left(|\alpha| + |\beta|\right)$$

where:

- $L(\alpha,\beta)$: The loss function, which balances the mean squared error (MSE) of predictions with an L_1-regularization term to prevent overfitting.
- $\hat{f}_{\text{combined}}$: The combined predictive output derived from weighted contributions of daily, weekly, and monthly data.
- λ: A regularization parameter that controls the penalty for large weights, encouraging sparsity in the contributions from different time scales.

The combined forecast is given by:

$$\hat{f}_{\text{combined}} = \alpha \cdot f_{\text{mean,daily}} + \beta \cdot f_{\text{mean,weekly}} + (1 - \alpha - \beta) \cdot f_{\text{mean,monthly}}$$

where:

- $f_{\text{mean,daily}}$: The mean prediction from the daily GPR model.
- $f_{\text{mean,weekly}}$: The mean prediction from the weekly GPR model.
- $f_{\text{mean,monthly}}$: The mean prediction from the monthly GPR model.
- α, β: Weights assigned to daily and weekly data, respectively. The remaining weight, $1 - \alpha - \beta$, is allocated to the monthly data.

Constraints

To ensure stability and interpretability of the model, the following constraints are imposed on the weights:

$$\begin{cases} 0 \leq \alpha \leq 1 \\ 0 \leq \beta \leq 1 \\ \alpha + \beta \leq 1 \end{cases}$$

These constraints ensure that the weights are non-negative and their sum does not exceed 1, maintaining a probabilistic interpretation of the contributions from each time scale.

5.4.2 Additional Strategy Considerations

Building on the strategies explored in this study, there are numerous opportunities to enhance and diversify approaches to better align with a range of market conditions. One promising modification involves the development of factor-based dynamic strategies. By combining GPR predictions with established factor models such as momentum or value, it is possible to create hybrid strategies that adapt not only to macroeconomic cycles but also to specific factor-driven opportunities. For example, during periods of economic expansion, the strategy could tilt toward value stocks identified through GPR forecasts, while momentum factors might be prioritized during market uptrends.

Another avenue for improvement lies in integrating GPR predictions into risk-parity strategies. These strategies, which aim to balance the risk contributions of assets within a portfolio, could benefit from GPR's ability to predict volatility dynamically. By adjusting risk-parity allocations based on real-time volatility forecasts, the strategy could better respond to evolving market dynamics and maintain a balanced risk profile.

Threshold-based rebalancing approaches could also be enhanced by introducing conditional adjustments. For instance, thresholds could be tightened during periods of high market volatility to ensure that the portfolio remains closely aligned with its target allocation, while wider thresholds could be used during more stable conditions to minimize transaction costs. This dynamic adjustment of thresholds would enable the strategy to adapt more effectively to changing market conditions.

Long-short portfolio strategies represent another area for exploration. By leveraging GPR to identify both predicted outperformers and underperformers, a long-short strategy could capitalize on relative performance differences between assets. This approach is particularly advantageous in neutral or bearish markets, where generating alpha through relative value opportunities becomes more critical.

Lastly, the incorporation of hedging strategies using derivatives such as options or futures could provide additional protection against extreme risks predicted by GPR. For example, when GPR forecasts heightened uncertainty or the potential for significant drawdowns, the portfolio could employ options-based hedges to mitigate downside risks while maintaining exposure to upside potential. This integration of predictive modeling with hedging tools would create a more resilient strategy capable of weathering adverse market conditions.

By pursuing these modifications and new strategies, portfolio managers can better harness the power of GPR to develop adaptive, robust, and highly tailored investment approaches that address the challenges and opportunities of diverse market environments.

5.4.3 Alternative Applications

The methodologies developed in this study extend far beyond traditional portfolio optimization, offering potential applications in various domains where predictive accuracy and uncertainty quantification are critical.

One such application is in risk management systems, where real-time risk monitoring could leverage Gaussian Process Regression (GPR) to forecast market volatility and potential drawdowns. By providing timely and actionable alerts, these systems can enable financial institutions to take preemptive measures, reducing exposure to adverse market conditions and improving overall risk control.

GPR's capability to model non-linear relationships and account for uncertainties also makes it a valuable tool in credit scoring models. For borrowers with limited credit histories or unconventional financial profiles, traditional models often fail to provide accurate risk assessments. GPR's flexibility allows it to incorporate a wider range of variables and generate more nuanced creditworthiness predictions, making it particularly beneficial in underserved markets.

In the realm of algorithmic trading, GPR can guide trading systems by forecasting short-term price movements with precision. These predictions enable dynamic adjustments to trade execution strategies, optimizing entry and exit points based on real-time market conditions. This application is especially relevant in high-frequency trading, where slight improvements in prediction accuracy can yield substantial financial gains.

The methodologies could also be extended to macroeconomic forecasting, where GPR could predict key economic indicators such as unemployment rates, consumer confidence, or inflation. Policymakers and financial analysts could use these forecasts to make informed decisions, improving the accuracy of policy interventions or market predictions.

Another promising area of application is renewable energy forecasting. Given the influence of weather conditions on renewable energy outputs, GPR can model and predict these fluctuations, aiding energy traders and utility companies in balancing supply and demand [19]. This capability would improve efficiency in energy markets and support the transition to sustainable energy systems.

Lastly, GPR can play a vital role in ESG (Environmental, Social, and Governance) and impact investing by forecasting ESG-related metrics. By providing reliable predictions on factors such as carbon emissions, social impact scores, or governance indicators, GPR enables investors to align their portfolios with sustainability objectives. This alignment is increasingly important in meeting the growing demand for socially responsible investment strategies.

These alternative applications demonstrate the versatility of GPR and its potential to drive innovation and efficiency across a wide array of industries, making it a valuable tool not just for portfolio optimization but for addressing broader challenges in risk management, trading, forecasting, and sustainability.

5.4.4 Scalability Considerations

Scalability is a crucial factor for implementing Gaussian Process Regression (GPR) in large-scale portfolios or real-time applications, where computational efficiency and responsiveness are paramount. To address these challenges, several strategies can enhance the model's scalability while maintaining accuracy and reliability.

One critical component is the implementation of efficient data processing pipelines. Utilizing parallelized frameworks such as Apache Spark or Dask can streamline data preprocessing, training, and prediction execution, enabling near real-time operations. These tools allow for the handling of large datasets by distributing computational tasks across multiple nodes, significantly reducing processing times.

Sparse Gaussian Processes (GPs) present another effective solution. By employing techniques such as inducing points, sparse GPs reduce the computational complexity associated with traditional GPR, making it feasible to work with large datasets without excessive resource consumption. This approach approximates the full GPR while preserving its core predictive capabilities.

Hierarchical modeling offers an additional avenue to enhance scalability. By grouping assets based on sectors, regions, or other logical categories and applying GPR within these clusters, the complexity of covariance estimation is significantly reduced. This hierarchical approach not only simplifies the computation but also maintains accuracy by tailoring predictions to specific subsets of the portfolio.

For real-time applications, edge computing provides a practical solution to reduce latency. Deploying GPR models on edge devices, such as servers located closer to trading platforms or risk management systems, ensures faster response times. This approach is particularly beneficial for high-frequency trading or dynamic risk monitoring, where delays can result in missed opportunities or increased exposure [8].

Cloud-based scalability further complements these solutions by leveraging platforms such as AWS SageMaker or Google Vertex AI. These platforms offer on-demand scaling of computational resources, allowing GPR models to handle fluctuating workloads efficiently. Additionally, they support real-time data ingestion and model deployment, ensuring that the system can adapt to evolving market conditions without manual intervention.

Lastly, integrating alternative data sources, such as satellite imagery, social media sentiment, or web scraping outputs, can enrich GPR's input features, enhancing its predictive power. However, incorporating such diverse data types requires robust preprocessing and scalable storage solutions. Distributed databases and cloud-based architectures can ensure that these non-traditional data sources are processed efficiently and seamlessly integrated into the modeling pipeline.

By adopting these strategies, GPR can be scaled effectively for large portfolios and real-time applications, providing accurate and timely predictions without compromising computational efficiency or resource availability. These considerations are essential for extending the practical utility of GPR in modern financial systems.

Abbreviations

GPR Gaussian Processes Regression

ML Machine Learning

MSE Mean Squared Error

ARIMA AutoRegressive Integrated Moving Average

MPT Modern Portfolio Theory

MSFT Microsoft Corporation

HLT Hilton Worldwide Holdings Inc.

JPM JPMorgan Chase & Co.

JNJ Johnson & Johnson

BA The Boeing Company

CAPM Capital Asset Pricing Model

WPE Weighted Permutation Entropy

DE Dispersion Entropy

PE Permutation Entropy

List of Figures

List of Tables

Bibliography

[1] S. Almahdi and S. Yang. “Adaptive portfolio optimization using deep reinforcement learning”. In: *Expert Systems with Applications* 117 (2019), pp. 295–303.

[2] A. Ang and G. Bekaert. “Regime switches in interest rates”. In: *Journal of Business & Economic Statistics* 20.2 (2002), pp. 163–182.

[3] S. Asmussen and P. W. Glynn. *Stochastic Simulation: Algorithms and Analysis*. Springer, 2007.

[4] R. Bellman. *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[5] R. Cont. “Empirical properties of asset returns: stylized facts and statistical issues”. In: *Quantitative Finance* 1.2 (2001), pp. 223–236.

[6] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. *Gaussian Processes for Data-Efficient Learning in Robotics and Control*. MIT Press, 2015.

[7] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. “Deep direct reinforcement learning for financial signal representation and trading”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.3 (2016), pp. 653–664.

[8] J. Doe and J. Smith. “Edge Computing for Financial Applications: A Review”. In: *Journal of Financial Technology* 12 (2021), pp. 123–145.

[9] D. Duvenaud et al. “Automatic model construction with Gaussian processes”. In: *Advances in Neural Information Processing Systems* 27 (2014).

[10] A. Fernandez, S. R. Singh, and E. Lezmi. “Machine learning in asset management—Part 1: Portfolio construction, forecasting, and trading”. In: *The Journal of Financial Data Science* 1.1 (2018), pp. 20–30.

[11] N. Garleanu and L. H. Pedersen. “Dynamic portfolio choice with transaction costs and predictable returns”. In: *The Journal of Finance* 64.2 (2009), pp. 579–626.

[12] M. Hagenau, M. Liebmann, and D. Neumann. “Automated news sentiment analysis for stock prediction”. In: *Expert Systems with Applications* 39.10 (2013), pp. 9064–9074.

[13] J. D. Hamilton. “A new approach to the economic analysis of nonstationary time series and the business cycle”. In: *Econometrica* 57.2 (1989), pp. 357–384.

[14] J. D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[15] J. B. Heaton, N. G. Polson, and J. H. Witte. "Deep Learning for Finance: Deep Portfolios". In: *Applied Stochastic Models in Business and Industry* 33.1 (2017), pp. 3–12.

[16] W. Huang and G. Zhou. "Building diversified portfolios that outperform out of sample". In: *Management Science* 65.9 (2019), pp. 4308–4324.

[17] S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas. "Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research". In: *European Journal of Operational Research* 247.1 (2015), pp. 124–136.

[18] T. Loughran and B. McDonald. "Liability of language: Words that affect credit risk". In: *The Journal of Financial Economics* 110.3 (2011), pp. 606–621.

[19] F. Lubbe, J. Maritz, and T. Harms. "Evaluating the Potential of Gaussian Process Regression for Solar Radiation Forecasting: A Case Study". In: *Energies* 13.20 (2020). ISSN: 1996-1073. DOI: `10.3390/en13205509`.

[20] H. Markowitz. "Portfolio Selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91.

[21] R. C. Merton. "Lifetime portfolio selection under uncertainty: The continuous-time case". In: *The Review of Economics and Statistics* 51.3 (1969), pp. 247–257.

[22] R. C. Merton. "Optimum consumption and portfolio rules in a continuous-time model". In: *Journal of Economic Theory* 3.4 (1971), pp. 373–413.

[23] J. Moody and M. Saffell. "Performance functions and reinforcement learning for trading systems and portfolios". In: *Proceedings of the Fifth International Conference on Neural Information Processing (ICONIP)*. Vol. 2. IEEE. 1998, pp. 953–956.

[24] Y. Nevmyvaka, Y. Feng, and M. Kearns. "Reinforcement learning for optimized trade execution". In: *Proceedings of the 23rd International Conference on Machine Learning (ICML)*. ACM, 2006, pp. 673–680.

[25] C. Phua, V. Lee, K. Smith, and R. Gayler. "A comprehensive survey of data mining-based fraud detection research". In: *Artificial Intelligence Review* 34.4 (2010), pp. 1–14.

[26] J. M. Poterba and A. A. Samwick. "Portfolio allocation for individual investors". In: *NBER Working Paper Series* 7734 (2000).

[27] P. Prakash. *OrdinalEntroPy: A Python Package for Ordinal Entropy*. `https://github.com/pradyot-09/OrdinalEntroPy`. Accessed: 2021-12-01. 2021.

[28] J. A. Primbs and C.-H. Sung. "Dynamic portfolio optimization with transaction costs: A model predictive control approach". In: *Automatica* 45.3 (2009), pp. 598–606.

[29] J. Quinonero-Candela and C. E. Rasmussen. "A unifying view of sparse approximate Gaussian process regression". In: *Journal of Machine Learning Research* 6 (2005), pp. 1939–1959.

[30] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[31] S&P Global. *S&P 500 Fact Sheet*. `https://www.spglobal.com/spdji/en/indices/equity/sp-500/`. Accessed: 2024-12-01. 2024.

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

[33] W. F. Sharpe. "Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk". In: *The Journal of Finance* 19.3 (1964), pp. 425–442.

[34] R. S. Tsay. *Analysis of Financial Time Series*. John Wiley & Sons, 2010.