# An open-source investment robo-advisor

*Linus Tong Chun*

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2025

# Abstract

The rise of financial technology (FinTech) has significantly transformed the investment landscape, with Robo-advisors emerging as one of its most significant innovations. These algorithm-driven systems offer automated, low-cost, and scalable investment solutions, making portfolio management more accessible to a broader range of investors. As financial firms continue to invest heavily in Robo-advisor infrastructure, there is growing interest in developing more cost-effective and adaptable alternatives.

This project explores the feasibility and challenges of developing an investment Robo-advisor using freely available datasets and open-source tools. The objective was to build a low-cost system capable of generating, evaluating, and adjusting investment portfolios to achieve consistent, positive returns. The Robo-advisor integrates historical performance analysis with predictive modeling to guide portfolio decisions and perform periodic adjustments over time.

## Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

## Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Linus Tong Chun*)

# Acknowledgements

I would like to thank my supervisor, Michael Rovatsos, who provided incredible support, feedback, and encouragement throughout this project.

I would also like to thank my family and friends for their physical, moral, and **mental** support throughout my years here at Edinburgh University.

# Table of Contents

# Chapter 1

# Introduction

The financial services industry has undergone significant change due to the rapid development of FinTech [15]. One of the key developments brought about by FinTech is the emergence of investment Robo-advisors. These are computer algorithms designed to assist investors in monitoring and automating the investment process. This includes investment recommendations, constructing investment portfolios, evaluating performance, ensuring diversification, and making asset adjustments as needed. The primary goal of Robo-advisors is to make investing more accessible, cost-efficient, and time-saving for a broad range of users.

While investors greatly benefit from the convenience and efficiency of Robo-advisors, finance companies invest heavily to operate these systems. According to one source [16], the annual operational costs of running one Robo-advisor system can be up to $2,000,000 USD. These costs cover aspects like technology and software licensing, research and analytics, cloud computing, hosting services, and other essential infrastructure. These costs compound when companies have to deploy multiple systems to encompass more users. Nonetheless, companies will continue to invest heavily in Robo-advisors because the alternative — **human advisors** — is far more expensive and considerably less scalable.

## 1.1 Motivation

With many financial firms heavily investing in Robo-advisor development, and their growing popularity among investors, I was motivated to explore the challenges and feasibility of building a more cost-effective alternative. My objective was to leverage freely available datasets and open-source tools to develop a Robo-advisor capable of generating investment portfolios with consistent, positive returns, all while minimizing operational costs. This study aims to examine each component of the Robo-advisor in detail, outlining the development process and highlighting the challenges encountered along the way. Afterwards, the system was tested within a single market to serve as a proof of concept. If successful, the core algorithm could then be adapted to perform effectively across other markets.

# Chapter 2

# Background

This chapter will begin by exploring the key differences between Robo-advisors and traditional advisors, along with possible reasons why Robo-advisors may become the future of financial advising. Next, the main investment strategy and evaluation metric used by my Robo-advisor will be introduced to establish the foundation for its development.

## 2.1 Robo-Advisor vs Traditional Advisors

In recent years, investors have become increasingly conflicted about whether to seek financial advice from a Robo-advisor or human advisor. While the finance industry has rapidly evolved through digitization and the adoption of innovative technologies, the acceptance of Robo-advisors by investors has yet to surpass traditional advisors [3]. The following are key considerations investors might take into account when choosing between Robo-advisor and traditional advisor:

**Investment Bias** - Robo-advisors help mitigate common investment biases such as loss aversion, trend chasing, and the disposition effect. By offering objective advice, they provide a more rational approach to investing that some investors might prefer[6].

**Conflict of interest** - Human financial advisors are often influenced by commission-based incentives, potentially steering clients toward higher-commission products, which is a conflict of interest [1]. In contrast, Robo-advisors operate with automated and transparent fee structures that aim to minimize this problem. They have no personal incentive to favor certain investments over others [2], and this transparency helps clients feel more secure in the objectivity of the advice they receive.

**Personalized Advice** - While Robo-advisors are capable of delivering more objective advice, they lack the empathetic understanding that human advisors possess [20]. Many clients value human advisors who understands their unique life circumstances and with whom they can build a long-term relationship [4]. This deeper connection often allows human advisors to provide more tailored and effective investment recommendations.

**Diverse Investment Strategies** - Robo-advisors typically rely on standardized algorithms that are adjusted based on rudimentary factors such as risk tolerance, capital, investment length, and diversification preferences [14]. While this works well for a broad range of investors, those seeking more active portfolio management or highly specialized investment strategies may find that traditional advisors offer greater flexibility and are better equipped to help them meet their financial goals.

**Availability** - Most platforms offering Robo-advisor services are accessible 24/7, allowing investors to manage their portfolios at any time and from virtually anywhere [17]. This contrasts with traditional advisors, who have limited availability and often require scheduled appointments for consultation. The instantaneous access Robo-advisors provide appeals especially to investors who value convenience and real-time control [2].

**Cost and Affordability** - Robo-advisor services are generally more affordable than traditional financial advisors [8]. By leveraging algorithms for portfolio creation and management, operational costs are significantly reduced. This also allows for efficient scaling, enabling Robo-advisors to serve thousands of clients with minimal additional resources [17]. Furthermore, since Robo-advisors are operate primarily online, costs associated with physical offices and in-person consultations can be minimized. These savings are passed on to clients, lowering the barrier to entry and making investing more accessible to a wider audience.

## 2.2 The Future of Robo-Advisors

While human advisors are still preferred by most investors, Robo-advisors are quickly gaining traction. A 2023 study shows that roughly 75% of investors still rely solely on human advisors, while the remaining 25% use either Robo-advisors exclusively or a combination of both [5]. However, further research highlights a notable shift: The number of investors incorporating algorithmic investment programs to manage their assets has increased more than tenfold since 2017, rising from 13 million to 147 million users as you can see in table 2.1. This trend suggests that Robo-advisors are becoming a more compelling option for investors and may become the new industry norm in the future.

Table 2.1: Growth in Users and Assets Managed by Robo-Advisors (2017-2023)

| | **2017** | **2018** | **2019** | **2020** | **2021** | **2022** | **2023** |
|---|---|---|---|---|---|---|---|
| Assets under management in million USD | 240,025 | 543,188 | 980,541 | 1,442,028 | 1,863,438 | 2,231,721 | 2,552,265 |
| Market Penetration rate in percent | 0.2 | 0.4 | 0.6 | 0.9 | 1.3 | 1.6 | 1.9 |
| **Users in thousand** | **13,105** | 26,101 | 45,774 | 70,509 | 97,398 | 123,539 | **147,018** |

Source: Adapted from *"Who uses robo-advisors? The Polish case"* [19].

Firstly, advancements in self-service technology (SST) have shifted many tasks from service employees to machines, offering users greater control and efficiency [10, 13]. Robo-advisors fit into this trend by transferring financial advisory tasks from human advisors to them. The current accessibility of Robo-advisors is already unmatched, and further advancement in SST will only continue to improve the ease of use of these programs. Combined with their affordability, Robo-advisors become a compelling, cost-effective alternative to traditional advisors, making investing more accessible to a broader audience.

Secondly, one of the main drawbacks of using machines for self service interactions is their inability to recognize and process emotion [9]. This limitation is especially critical for Robo-advisors, whom investors are entrusting their hard-earned capital with. Unlike human advisors, Robo-advisors lack empathy and cannot provide personalized financial advice that considers the unique circumstances of each individual. However, with rapid development in machine learning, AI, and natural language processing (NLP), this limitation may be partially mitigated. In the future, Robo-advisors could take investor comments as an additional input, allowing individuals to specify their preferred investment strategies or share relevant personal details. With advances in NLP, these Robo-advisors may be able to interpret investor input and give more tailored advice. While investors are still unlikely to form any meaningful relationships with a Robo-advisor the same way they might with a human advisor, this added capability is still a step towards improving the empathy and emotional issues of Robo-advisors, making them more appealing to some investors.

Lastly, since Robo-advisors are a relatively new concept, many older investors are hesitant to trust them. A 2020 study found that investors under the age of 35 are nearly twice as likely to use Robo-advisors compared to those aged 35 to 55, and five times more likely than those over 55 [2]. Hence, as new investors enter the market, they are expected to be more open to Robo-advisors than existing investors. Combined with the advancements in LLMs and SST, new investors are likely to be more inclined to use Robo-advisors, leading to an increase in their usage in the future.

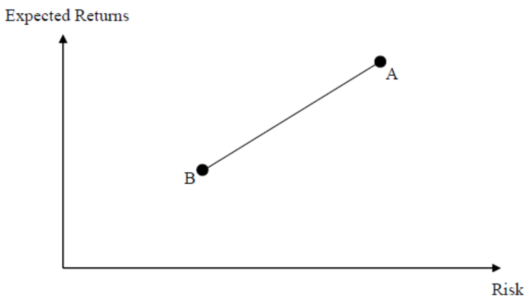
## 2.3 Investment Strategy and Evaluation Metrics

Investment strategies vary greatly, each with different focuses. Some involve greater risk for potentially greater profits, while others aim for stability and consistent returns. Some strategies emphasize diversification to reduce risk, while others concentrate investments to potentially maximize profits for greater risk. Each strategy offers unique advantages and drawbacks that vary depending on the goals of each investor.

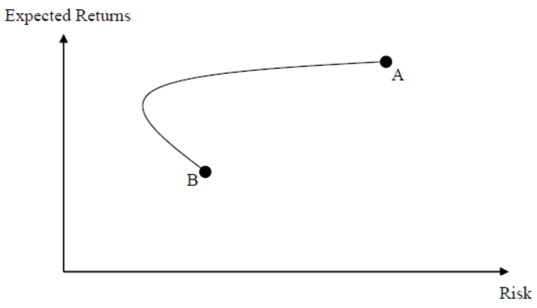
While many investment models exist — such as the Capital Asset Pricing Model (CAPM), the Black-Litterman model, and various Factor Investing models — most are derived from the **Markowitz Mean-Variance model**.

### 2.3.1 Markowitz Mean-Variance Model

The Markowitz mean-variance model is a portfolio optimization method used to construct investment portfolios by weighing risk, as standard deviation, against expected return [7]. In this model, higher standard deviation implies greater risk but the potential for greater returns. The key feature of the Markowitz model is its emphasis on diversification, which theorizes that by diversifying investments, portfolio risk will decrease while expected returns increases. In figure 2.1a for instance, investment A has higher risk with greater returns potential, while investment B has lower risk with less potential for return. Intuitively, by holding both investments, the portfolio's risk-return will lie on the line between A and B. However, the Markowitz model predicts that by diversifying across multiple assets, expected return actually increases returns while risk decreases. As shown in figure 2.1b, the curved line is called the **Markowitz efficient frontier** and contains points where expected returns is increased while risk is decreased.



(a) Normal Model Intuition

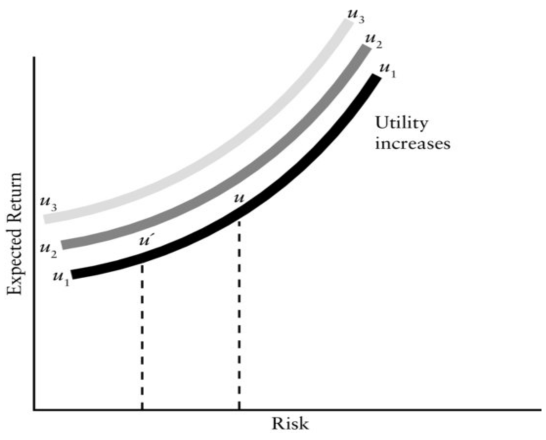


(b) Markowitz Model Prediction

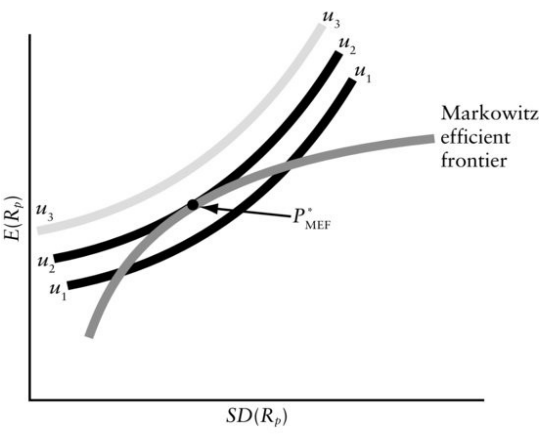
In accordance with the Markowitz Mean-Variance Model, diversification-alongside returns and volatility-will be a key factor in portfolio construction when developing my Robo-advisor. Experiments will also be conducted to evaluate the practical effectiveness of the model and determine whether its principles hold true in real-world scenarios.

### 2.3.2 Indifference Curves

All models share a common objective: Balancing risk and return according to investor preferences. To do so, it is helpful to first understand **indifference curves**. As shown in figure 2.2a, indifference curves like $u_1$, $u_2$, and $u_3$ represent combinations of expected return and risk that yield the same utility for an investor [7]. For instance, on curve $u_1$, $u'$ and $u$ lie on the same indifference curve, thus offering equal utility to the investor despite the variation in risk and return. A higher indifference curve, such as $u_2$ compared to $u_1$, implies greater utility at every point. Therefore, every point on the higher curve represents a more favorable return-to-risk ratio.



(a) Indifference Curve with Risk against Return Source: *"Mean-Variance Model for Portfolio Selection"* [7].



(b) Indifference Curves with Markowitz Model Source: *"Mean-Variance Model for Portfolio Selection"* [7].

Combining the indifference curves with the Markowitz Model, figure 2.2b is produced which illustrates optimal portfolio selection based on investor preferences. To maximize investor utility, the point where the Markowitz efficient frontier meets the highest possible indifference curve is selected at $P^*_{MEF}$. This intersection represents the portfolio with the best balance of risk and expected return [7].

In real-world applications, indifference curves are often used to help investors decide which investments suit their individual risk preferences. However, for the purposes of this study, a standard portfolio evaluation metric will be used to objectively assess portfolio performance.

### 2.3.3 Portfolio Evaluation Metric

Portfolio evaluation metrics are used to assess an investment's return performance relative to its volatility. One of the most commonly used metric is the Sharpe ratio:

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma}$$

where $R_p$ is expected return, $R_f$ is risk free return, $\sigma$ is the standard deviation/volatility. The risk free rate is the rate of return of an investment with no risk of financial lost. An example of a risk-free investment is U.S. treasuries which are backed by a guarantee from the U.S. government. The Sharpe ratio calculates the risk-adjusted return by taking the difference between the expected return of an investment and the risk-free rate, and dividing it by the volatility of the investment. A high Sharpe ratio indicates a better risk-adjusted return, meaning the investment has good returns for its level of risk [12].

Another common evaluation metric is the Sortino ratio. It is a variation of the Sharpe ratio but instead of considering total volatility, it focuses specifically on downside risk:

$$Sortino\ Ratio = \frac{R_p - R_f}{\sigma_d}$$

It calculates risk-adjusted returns by dividing excess return ($R_p - R_f$) with standard deviation of **negative returns** ($\sigma_d$)[12]. This method is helpful for investors who are risk averse and prioritize minimizing losses. However, this may lead to a larger preference for investments with lower potential returns.

# Chapter 3

# Objective and Setup

This chapter will outline the main objectives of the Robo-advisor, detailing its inputs and outputs. Afterwards, I will describe the necessary setup required to begin developing the Robo-advisor and the standards chosen to remain objective throughout the development process.

## 3.1 Main Objective

The main goal of the Robo-advisor is to construct investment portfolios based on historical data, then periodically assess and modify them as necessary to maximize financial returns. It is designed to take three main inputs from the investor as parameters: investment period, risk tolerance, and types of investments.

1. **Investment period** is the time length the investor hopes to invest. This parameter helps the Robo-advisor produce an investment portfolio that aligns the with investor's timeline. For example, long-term investments typically allow for greater risk tolerance, as there is more time for the market to recover from downturns, which allows for a more aggressive investment strategy.
2. **Risk tolerance** measures how comfortable an investor is with portfolio volatility. This input determines the aggressiveness of the generated portfolio.
3. **Types of investments** refers to the categories of assets the investor wants to hold, such as stocks, bonds, real estate, or ETFs. This input allows the Robo-advisor to diversify the investment portfolio according to the investor's preferences.

Additionally, the investor can specify other minor inputs such as the number of assets, the frequency of portfolio evaluation and restructuring, and specific investments to avoid. It is important to note that I did not include investment capital as an input to the Robo-advisor. Instead, I will use weights to specify the percentage of the total investment fund that should be allocated to each investment. For instance, an asset with a weight of 0.3 indicates that 30 percent of your investment funds should be allocated to it. This simplifies the process by not needing to calculate the specific dollar amounts for each investment.

After taking these inputs, the Robo-advisor will construct an investment portfolio based on the input parameters. It will also automatically re-evaluate the portfolio every week – 5 active trading days – by default, or if specified by the investor.

## 3.2 Investment Portfolio Selection

As previously discussed, investors can use indifference curves to select their ideal portfolios. Although the option to prioritize returns or risk has been implemented, for the purposes of this study, I will use the Sharpe ratio to approximate the optimal portfolio as objectively as possible. The Sharpe ratio is defined as follows:

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma}$$

where $R_p$ is the expected return of the portfolio, $R_f$ is the risk-free rate of interest, and $\sigma$ is the standard deviation/volatility of the portfolio. The higher the Sharpe ratio, the 'better' the portfolio is considered.

## 3.3 Programming Language

Python was selected as the programming language due to the extensive and easily accessible libraries. For instance, Pandas, a library designed for Python, proved extremely well-suited for data manipulation and analysis in my use case. Other libraries like SciPy and TensorFlow which were used for machine learning implementation also offer extensive documentation in Python, further supporting the choice.

## 3.4 Data Collection

For data collection, the primary requirement is obtaining the adjusted closing prices for assets, which reflect the prices of each asset at the end of each trading day. Adjusted closing prices account for splits, dividends, and new stock offerings. This simplifies our analysis by removing these variables to consider when generating a portfolio. It is important to note that second-to-second data is not needed, as the goal of the Robo-advisor is not day trading, but rather to create portfolios aimed at longer-term growth, such as over months or years.

Various data collection options were explored, each with its own set of limitations. For instance, the Alpaca Market API, although provided extensive asset data, restricted the number of free calls per week, making it unsuitable for back-testing the performance of the Robo-advisor across multiple periods. Another option considered was Quandl, a subsidiary of Nasdaq. However, the data had to be extensively processed before use, which increased computations.

Ultimately, I chose the Yahoo Finance, which offers suitable data and allows a sufficient number of free API calls to render limitations negligible. Additionally, the data integrates seamlessly with pandas data-frames, making it ideally suited for this use case.

## 3.5 Data Manipulation and Analysis

Once the adjusted closing prices of the assets were obtained, I used them to calculate the necessary inputs to begin constructing portfolios, including the **annual returns**, **variances**, **standard deviation**, **covariances**, and **correlations** of the assets.

1. **Annual returns** represent the percentage change in asset price over a year. To calculate this, I computed the average of all daily returns of an asset and multiplied it by 252, the typical number of active trading days in a year.
2. **Variance** measures the spread of the returns of an asset around its mean. However, because it is calculated in squared units of the original data, it does not meaningfully measure the volatility of an investment.
3. **Standard deviation** is the square root of variance that meaningfully reflects the volatility of an investment.
4. **Covariance** measures the relationship between the returns of two assets. A positive covariance between two assets suggests that they are correlated and their prices move together, while a negative covariance suggest they are inversely correlated. The covariances are stored in a covariance matrix for accessibility.
5. **Correlation** is a standardized form of covariance that Correlation measures the strength and direction of the relationship between two variables [11]. This metric ranges from -1 to 1. A score of 1 indicates that two assets are perfectly correlated, A score of 0 indicates that the assets are completely uncorrelated. A score towards -1 indicates that the assets are inversely correlated. The correlations are stored in a correlation matrix for accessibility.

By using annualized returns and standard deviation, we can create a standard graph that plots volatility, against returns to asses the annual performance of each asset. Below is graph 3.1 showing the performance of S&P500 company stocks from 2024 to 2025. [1]

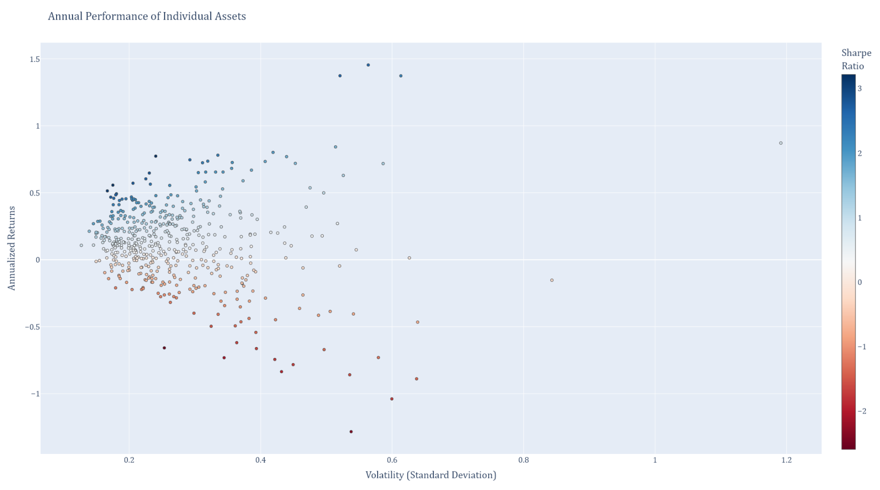


Figure 3.1: Performance of S&P500 assets between 2024-01-01 and 2025-01-01

[1]The interactive plot for figure 3.1 can be viewed at this link

The Sharpe ratio of each asset is calculated and each asset on the graph is colored-coded based on this metric. Investors can use this information to decide which assets they might want to invest in by evaluating their risk-to-return performance in the past, as indicated by the Sharpe ratio.

## 3.6 Standardization

Multiple strategies of portfolio generation were developed. To accurately measure the performance of each strategy, certain standards must be established. I will now explain the key standards chosen and the rationale behind them.

### 3.6.1 Weight Optimization

Before introducing the different models used for portfolio construction, it is necessary to explain the weight optimization function that is used in all models. As previously mentioned, instead of direct investment capital, weights are used for capital allocation.

Weight optimization involves finding the optimal weight allocation for each asset in a portfolio to maximize the Sharpe ratio. At its core, weight optimization is a gradient descent problem. Given an arbitrary number of assets in a portfolio, we must determine the weights of each asset that would produce the highest Sharpe ratio. To achieve this, I employed a commonly used gradient descent algorithm called Sequential Least Squares Programming (SLSQP). The algorithm is implemented through the minimize function in the SciPy library [18].

SLSQP begins with a random set of asset weights that sum up to one. Its objective function is to maximize the portfolio Sharpe ratio by iteratively adjusting the weights. At each iteration, it measures how small changes in each weight affect the overall Sharpe ratio, which is used estimate the gradient of the objective function. Guided by this gradient, SLSQP adjusts the asset weights and steps towards the maximum Sharpe ratio. The algorithm stops when the gradient is near zero and the final asset weights are returned.

The following graph 3.2 is a demonstration of SLSQP. Six randomly chosen assets from the S&P 500 asset pool were inputted into the algorithm. As you can see, it iteratively finds the optimal weights to maximize Sharpe ratio. Initial decreases in the Sharpe ratio can be attributed to the randomness of the starting weights.

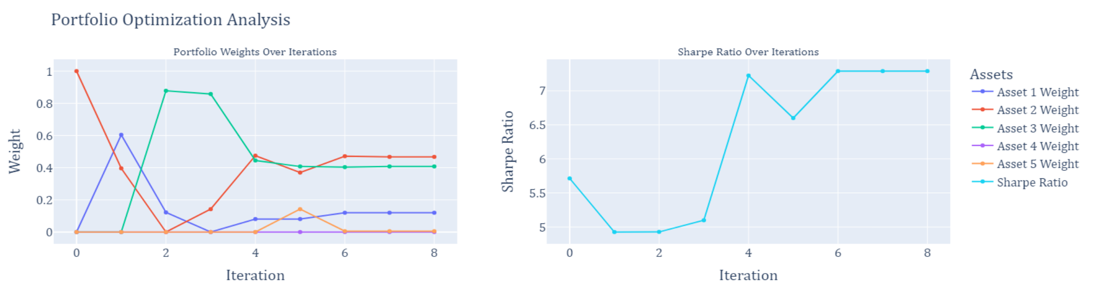


Figure 3.2: Demonstration of SLSQP

Throughout the development process, the SLSQP algorithm, which is implemented via Scipy's minimize function, is consistently used to find the optimal portfolio weights that maximize the Sharpe ratio. Since the minimize function is inherently designed to minimize objective functions, the algorithm instead minimizes the negative Sharpe ratio — effectively maximizing Sharpe ratio.

### 3.6.2 Portfolio Size

During weight optimization, I observed that SLSQP often assigns weights of zero to some assets within portfolios. For example, in figure 3.2 assets 4 and 5 have weights near zero. This suggests that these assets contribute little or nothing to the portfolio's Sharpe ratio, and excluding them might be better. Consequently, this lead me to investigate the optimal number of assets to include in a portfolio.

I generated numerous random portfolios, each consisting of 50 assets randomly selected from a pool with over 2000 assets, including stocks, bonds, and ETFs. I then ran the SLSQP algorithm and analyzed the number of assets in each portfolio that had a weight greater than 0.01% of the total weight. This is to examine which assets were actively and significantly contributing to the portfolio.

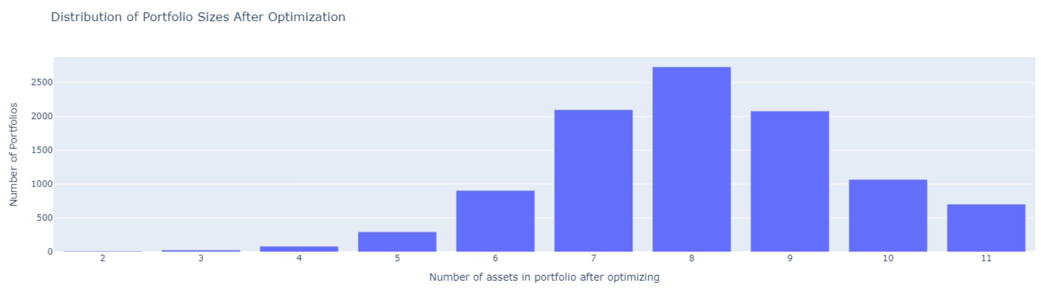


Figure 3.3: Normal Distribution of Portfolio Size after SLSQP

The results in figure 3.3 show a significant reduction in the original portfolio sizes of 50 assets. After applying the SLSQP algorithm, the portfolio sizes were found to be distributed between 2 to 11, with the most common size at 8 assets per portfolio. Based on these findings, the following strategies used to construct portfolios have been standardized to 8 assets to maintain objectivity and maximize Sharpe ratio.

# Chapter 4

# Portfolio Generation Strategies

This chapter will detail the various portfolio generation strategies developed, explaining the thought process behind each one. It will also highlight the challenges encountered during the implementation of each strategy and explain how subsequent strategies addressed and overcame these issues.

## 4.1 Brute Force Strategy

The first portfolio generation model I devised utilized a **brute force strategy**. An investor provides a list of potential assets. The model then generates all possible portfolios from this list. Afterwards, the SLSQP algorithm is applied to each portfolio to determine the maximum Sharpe ratio that specific asset combination can achieve. The portfolio with the highest Sharpe ratio is then recommended to the investor. The following pseudo-code 1 is a simplified outline of the algorithm implemented.

**Algorithm 1** Portfolio Generation: Brute Force Method

```
procedure BRUTE FORCE STRATEGY(Potential Assets List)
    $Best\ Portfolio \leftarrow$ None
    $Best\ Sharpe \leftarrow -\infty$
    $All\ Possible\ Portfolios \leftarrow$ Combinations($Potential\ Assets\ List$)
    for each $portfolio$ in $All\ Possible\ Portfolios$ do
        $result \leftarrow$ scipy.minimize(NegativeSharpe, portfolio, method='SLSQP')
        if $result.sharpe > Best\ Sharpe$ then
            $Best\ Sharpe \leftarrow result.sharpe$
            $Best\ Portfolio \leftarrow portfolio$
        end if
    end for
    return $Best\ Portfolio$
end procedure
```

This method is guaranteed to obtain the best possible portfolio based on the Sharpe ratio because it evaluates every possible combination of assets. However, a significant limitation of the algorithm is that the asset list provided by the investor must remain small. This is because the number of possible combinations of assets grows factorially, leading to a significant increase in computation time. Specifically, the algorithm has a time complexity of $O(n!/(k! * (n-k)!))$ where $n$ is the number of total assets and $k$ is the combination size. Consequently, this factorial growth in complexity becomes a problem when attempting to scale up the number of assets to consider.

## 4.2 Monte Carlo Strategy

The subsequent model I developed uses a **Monte Carlo strategy**. Instead of evaluating every asset combination, this model would generate portfolios with pseudo-random asset combinations. The randomness is monitored by a hashmap that tracks the frequency an asset is included in portfolios. Assets that are less frequently included are then more likely to be selected for subsequent portfolios, ensuring all assets are explored diversely. This method addresses the problem associated with the brute force method, which requires evaluating every possible asset combination. Using the Monte Carlo strategy, the model evaluates portfolios sufficiently to achieve a reasonably optimal portfolio without excessive amounts of computation.

**Algorithm 2** Portfolio Generation: Monte Carlo Method

```
procedure MONTE CARLO STRATEGY($Potential\ Assets\ List$, $N$)
    $Dominant\ Portfolios \leftarrow \{\}$
    for $i = 1$ to $N$ do
        $Current\ Portfolio \leftarrow$ Generate Random Portfolio($Potential\ Assets\ List, 8$)
        $result \leftarrow$ scipy.minimize(NegativeSharpe, Random Portfolio, method='SLSQP')
        $Current\ Sharpe \leftarrow result.Sharpe$
        $isDominant \leftarrow$ True
        for all $Portfolio$ in $Dominant\ Portfolios$ do
            if $Current\ Sharpe \leq portfolio.Sharpe$ then
                $isDominant \leftarrow$ False
                break
            end if
        end for
        if $isDominant$ then
            Dominant Portfolios $\leftarrow Dominant\ Portfolios \cup \{Current\ Portfolio\}$
        end if
    end for
    return D$ominant\ Portfolios$
end procedure
```

The Monte Carlo model first iteratively generates portfolios composed of eight randomly selected assets. For each portfolio, the SLSQP algorithm is applied to optimize asset weights and determine the maximum Sharpe ratio that particular asset combination can achieve. The resulting Sharpe ratio is compared to the Sharpe ratios of other portfolios in a list called 'dominant portfolios' — a list containing portfolios that have outperformed all others in previous iterations. If a portfolio's Sharpe ratio exceeds all Sharpe ratios within 'dominant portfolios', it is added to to the list. The final portfolio in 'dominant portfolios' is considered the best and will be returned as the output.

For demonstration, the S&P 500 assets and their adjusted closing data from 2024-2025 that was previously used in figure 3.1 will be used. Selecting an appropriate number of portfolios to generate is crucial because finding a balance between portfolio quality and computation efficiency is key for the Monte Carlo strategy to be effective. To determine the optimal number of portfolios, I generated 50,000 portfolios and plotted a graph 4.1 shown below to observe the frequency of dominant portfolios found.

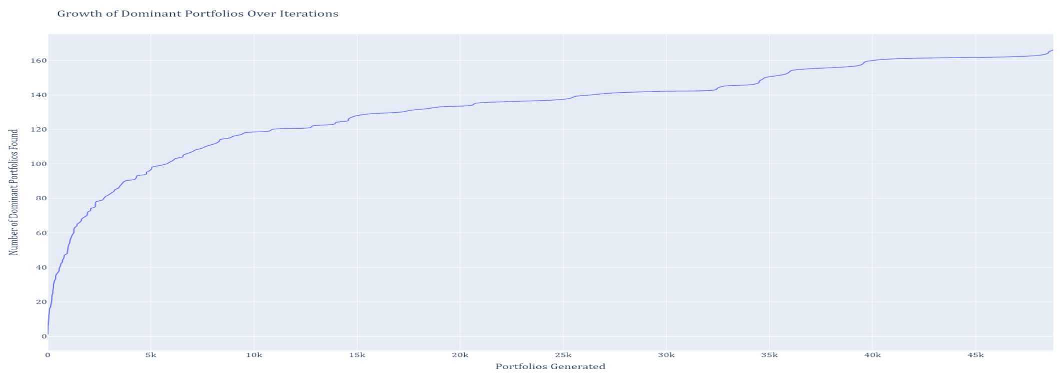


Figure 4.1: Frequency of Dominant Portfolios

As shown, there are significant diminishing returns in the frequency of dominant portfolios relative to the total number of portfolios generated. Half of the dominant portfolios were found in the first 5000 generated portfolios compared to the remaining 45000. However, it is important to note that while fewer dominant portfolios were found later, some were still identified. This implies that there are likely better portfolios to be discovered with continued exploration. This issue will be addressed in subsequent models. For the current model, generating between 10,000 and 15,000 portfolios has proven to be a good balance between quality and efficiency.

The figure 4.2 consists of 10,000 randomly generated portfolios, with volatility plotted against returns. Each point on the graph is color-coded based on its Sharpe ratio. Portfolios towards the top left of the graph are considered better according to historical data as they exhibited high returns and low volatility.

On the same graph, but this time plotting only the dominant portfolios from the simulation alongside individual assets, the Markowitz efficient frontier begins to form, as shown in figure 4.3. Notably, many portfolios in the graph exhibit lower volatility than even the least volatile individual asset, while also achieving higher returns. This experiment supports the Markowitz mean-variance model which theorizes that by diversifying investments, portfolio risk can decrease while financial returns increases.

When deciding which portfolio to invest in based on this graph, investors should consider portfolios near the top of the Markowitz curve, where returns are maximized for a given level of risk. The specific portfolio ultimately depends on the individual preferences of the investor. However, the objective recommendation by the algorithm is the portfolio with the highest Sharpe ratio.

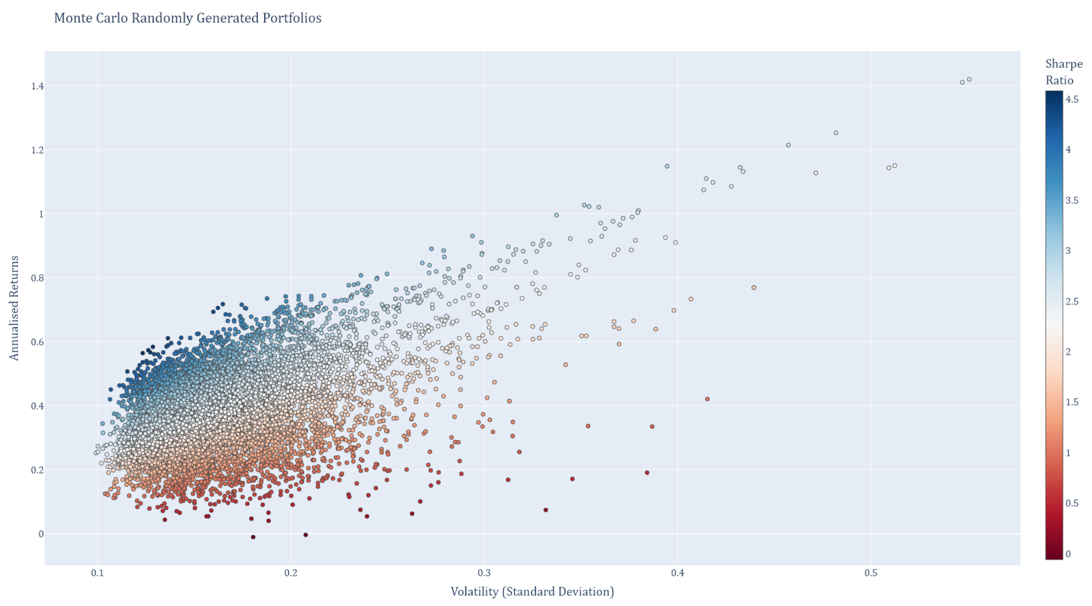


Figure 4.2: Performance of 10,000 randomly generated portfolios using Monte Carlo[1]

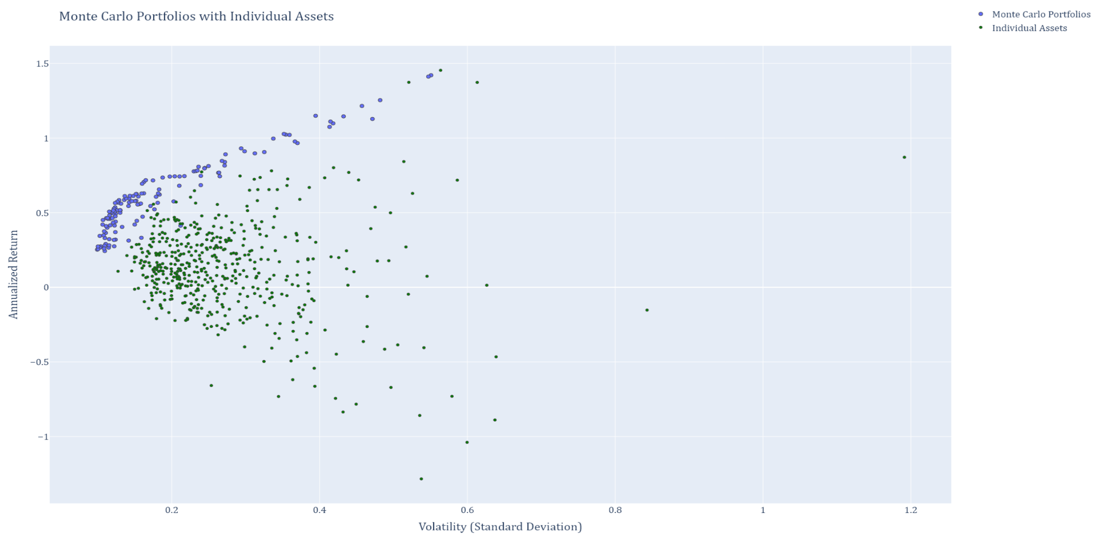


Figure 4.3: Dominant Portfolios plus Individual Asset data[2]

The issue the Monte Carlo strategy ultimately faces is the random walk problem. As shown previously in figure 4.1, despite diminishing returns in the frequency of dominant portfolios, new dominant ones would still appear, albeit less frequently. Since portfolios are generated and evaluated at random, it is possible that the optimal portfolio is never

[1]Figure 4.2 for 10,000 randomly generated portfolios can be viewed at this link
[2]Figure 4.3 for the Monte Carlo dominant portfolios can be viewed at this link

found, especially when the number of generated portfolios is finite. Additionally, the randomness makes the process stochastic rather than deterministic, leading to varying results each time the Monte Carlo algorithm is run, even on the same dataset. This problem becomes more pronounced when the asset pool is increased or when the number of assets per portfolio increases. Both factors expand the number of possible asset combinations, making it less likely that the optimal portfolio or near optimal portfolio will be found when evaluating a finite number of portfolios.

## 4.3 Convergence Strategy

To resolve the random walk problem presented in the Monte Carlo strategy, a different approach was needed that could identify the optimal portfolio without relying on random sampling. The strategy I developed begins with a randomly generated portfolio and incrementally improve it by substituting assets with better options from the asset pool, ultimately converging toward the optimal portfolio. This approach is called the **convergence strategy**:

**Algorithm 3** Portfolio Generation: Convergence Method

```
 1: procedure CONVERGE TO OPTIMAL PORTFOLIO(AssetPool)
 2:     Generate a random base portfolio with a fixed number of assets
 3:     Evaluate the base portfolio using SLSQP to maximize Sharpe ratio
 4:     Store the base portfolio in a list of improvement portfolios
 5:     for each iteration do
 6:         Identify the least valuable asset in the current portfolio
 7:         Generate a ranked list of candidate assets from the asset pool
 8:         for each candidate asset in the ranked list do
 9:             Create a new portfolio by substituting the least valuable asset
10:             Evaluate the new portfolio using SLSQP
11:             if the new portfolio has a higher Sharpe ratio then
12:                 Accept the new portfolio as the new base
13:                 Store the new portfolio in the improvement list
14:                 break inner loop to restart substitution process
15:             end if
16:         end for
17:         if no improvement is found after testing all candidates then
18:             Terminate the loop
19:         end if
20:     end for
21:     return Final portfolio, list of improved portfolios
22: end procedure
```

The convergence strategy begins by generating a completely random base portfolio, which is optimized using the SLSQP algorithm to determine the asset weights that yield the highest achievable Sharpe ratio. A function called 'Find_worst_asset' is then used to identify the least contributing asset in the portfolio for potential replacement.

Next, another function, ‘Find_substitute_asset’, ranks all remaining assets in the pool based on their potential to improve the current portfolio, and the highest-ranked asset is selected as the substitution. The worst asset is replaced with this substitution, and the new portfolio is optimized using SLSQP. If the new Sharpe ratio exceeds that of the previous portfolio, the updated portfolio becomes the new base. If not, the algorithm continues testing the candidates from the ranked list. Once an improvement is found, the process repeats from the ‘Find_worst_asset’ step. If a new base portfolio cannot be established, it implies that the current portfolio is near-optimal. All base portfolios identified throughout the process are stored in a list called ’improvement portfolios’ for future use.

It should be noted that the current algorithm optimizes purely for the Sharpe ratio. However, if an investor prefers to prioritize either returns or volatility, the optimization criteria can be changed accordingly.

### 4.3.1 ‘Find Worst Asset’ Function

There are three main factors to consider when determining the least contributing asset within a portfolio: returns, volatility, and correlation. These factors combine to give a score to each asset in the portfolio, and the asset with the lowest score is considered the worst and selected for substitution.

Returns and volatility are straightforward, higher returns lead to a higher score, and lower volatility also leads to a higher score. Correlation is also considered because of diversification. As previously discussed, the Markowitz theory emphasizes that diversification enhances portfolio performance. Therefore, an asset that is highly correlated with other assets in the portfolio is less desirable. Correlation is measured using covariances between assets within the portfolio. Subsequently, assets that are highly correlated to others within the portfolio receive a lower score.

The scores from all three factors are combined to compute a final score for each asset. The asset with the lowest final score is considered the worst-performing and is selected for substitution.

### 4.3.2 ‘Find Substitute Asset’ Function

To rank assets from the asset pool for substitution, the same three factors — returns, volatility, and correlation — are considered. Since the returns and volatility of each asset have already been computed, calculating these respective scores is straightforward.

To compute the correlation score for each asset in the pool, the precomputed correlation matrix is used. For each asset within the pool, its correlation values with the existing assets in the portfolio are summed and stored. These values are then normalized to produce correlation scores. A lower correlation score indicates weaker correlation with the portfolio assets, leading to better diversification and makes the asset more desirable.

The three scores — return, volatility, and correlation — are equally weighted and combined to generate a final score for each asset in the pool. The assets are then ranked based on their scores and returned as an ordered list for substitution.

### 4.3.3 Demonstration

For demonstration, the S&P 500 assets and their adjusted closing data from 2024-2025 will be used. The following figure illustrates the progression of the convergence strategy. Each dot represents a portfolio saved in the list 'improvement_portfolios'. The initial random portfolio appears as the dot toward the bottom of the graph. As new base portfolios are found, the progression towards the portfolio with the highest Sharpe ratio can be observed. As mentioned, the current implementation optimizes for the Sharpe ratio. Therefore, some portfolios on the graph may have higher returns than the final output portfolio. But these portfolios have less favorable risk-to-return performance according to the Sharpe ratio. In this example, a total of 68 base portfolios were recorded, indicating the optimal portfolio was found after 67 portfolios.

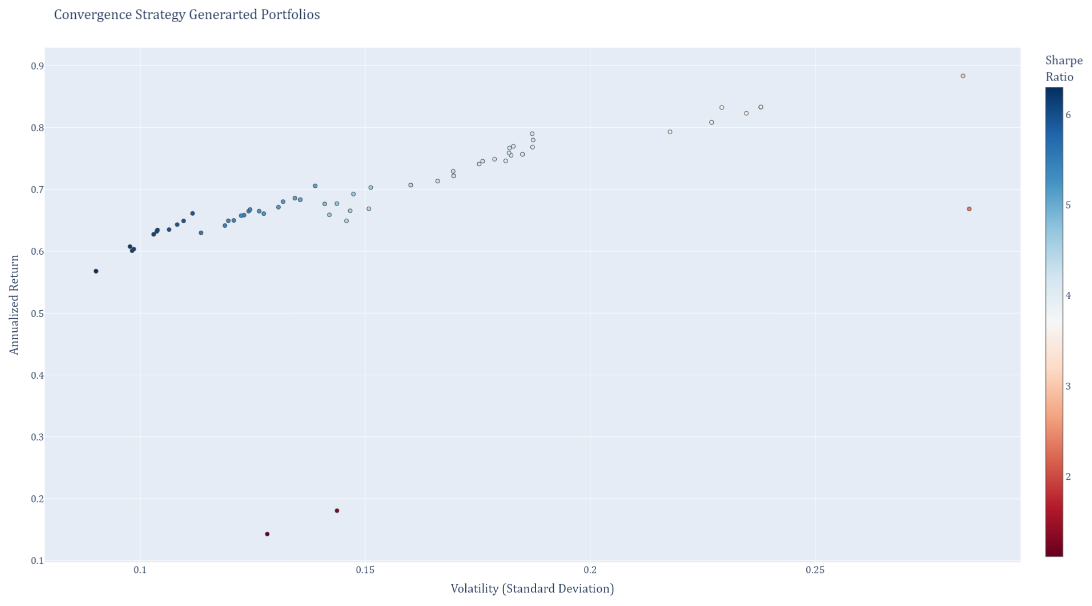


Figure 4.4: Demonstration of Convergence Strategy [3]

### 4.3.4 Comparison with Monte Carlo Strategy

The goal of using the convergence strategy over the Monte Carlo strategy was to address the main problem of finding the optimal or near-optimal portfolio within a finite number of iterations. The following figure 4.5 builds upon figure 4.2 and provides a comparison between the portfolios generated by the Monte Carlo strategy and those produced by the convergence strategy.

[3]Figure 4.4 for the Demonstration of the convergence strategy can be viewed at this link

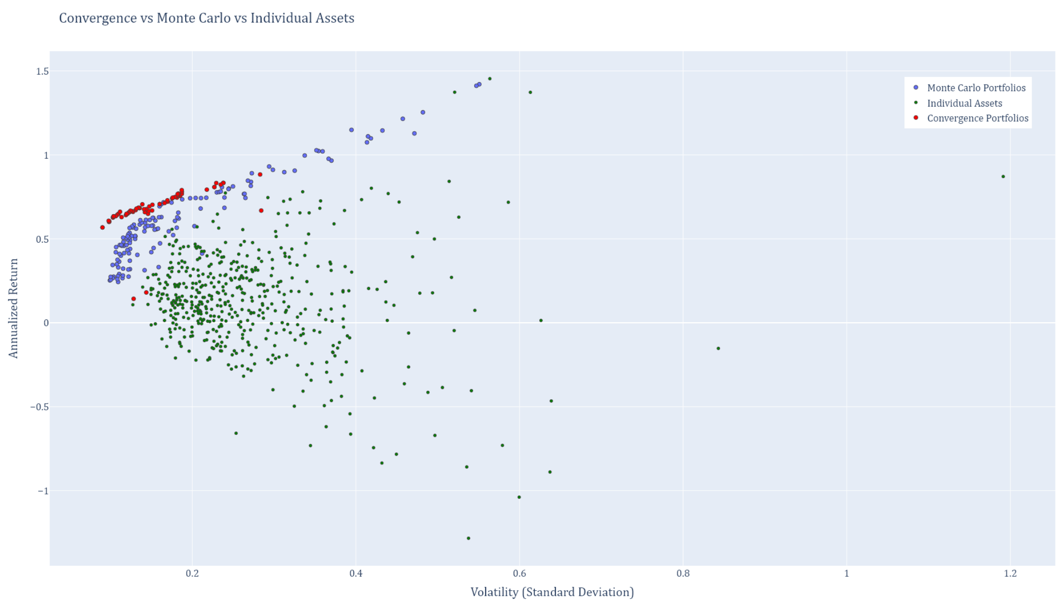


Figure 4.5: Comparing Convergence Strategy with Monte Carlo Strategy[4]

The red dots and blue dots represent the portfolios generated by the convergence and Monte Carlo strategy respectively. While the Monte Carlo strategy generates a greater variety of portfolios, the convergence strategy is able to find portfolios with higher Sharpe ratios. This is because the convergence strategy is designed to iteratively improve towards the portfolio with the highest Sharpe ratio, whereas the Monte Carlo strategy explores a wide range of possible portfolios without guaranteeing convergence on the optimal one.

It is important to note that the convergence method is highly flexible as it can be configured to converge on portfolios with higher returns, lower volatility, or any other metric. In contrast, the Monte Carlo method remains fixed as it relies on random sampling and cannot be directed to optimize according to investor preferences.

The main concern with the convergence method is its computation time. In the example above, the algorithm required over 3,500 iterations before converging on the optimal portfolio. Although this is much fewer than the preset of 10,000 portfolios evaluated by the Monte Carlo method, each iteration in the convergence strategy takes longer. This is due to the substitution process and the need to search through the ranked asset list. As a result, the overall efficiency of the convergence method is not necessarily better, and in some cases, even worse than the Monte Carlo method.

Currently, the algorithm stops when all assets in the pool have been tested for substitution and no further improvement in the Sharpe ratio is found. However, during testing, I found that the main source of inefficiency is due to the need to search far down the ranked asset list before finding one that actually improves the Sharpe ratio. This can significantly increase computation time, especially with larger asset pools. To address this, I developed the learning convergence strategy.

[4]Figure 4.5 for comparison between Monte Carlo and convergence strategy can be viewed at this link

## 4.4 Learning Convergence Strategy

The main problem the normal convergence strategy faces is the need to search far down the ranked asset list before finding an asset that is desirable. This is because the asset ranking method in the 'find_substitute_asset' function uses fixed weights to combine the return, volatility, and correlation scores. Since return and volatility values are static, the only factor that changes between iterations is the correlation score-due to changes in assets in each base portfolio. As a result, the asset rankings tend to remain similar across iterations, causing the algorithm to frequently search deep into the list before finding an asset that improves the portfolio Sharpe ratio.

To address this problem, I applied reinforcement learning to enhance the convergence strategy. Instead of using fixed weights to combine the return, volatility, and correlation scores when ranking assets for substitution, the algorithm dynamically adjusts these weights with each successful substitution found.

Recalling how the convergence method works: when an asset is substituted and results in a higher Sharpe ratio, the new portfolio is set as the new base. With the learning convergence method, the substituted asset is then analyzed to adjust the score weights. For returns, if the asset has higher or lower returns than the average returns of all assets in the pool, the return weight is adjusted accordingly. Similarly for volatility, if the added asset has higher or lower volatility than the average, the volatility weight is updated. For correlation, the algorithm compares the average correlation among the portfolio assets before and after the substitution. If the new asset lowers the overall correlation – implying greater diversification – the weight given to the correlation score is increased. Otherwise, it is decreased.

After each update, the weights are normalized to so they sum to one. Additionally, all weights are adjusted proportionally using an adaptive learning weight. If the improvement in the Sharpe ratio is minimal, the learning rate is reduced to allow finer adjustments. If the improvement is large, the learning rate is increased to accelerate learning. The overall reinforcement learning enables the learning convergence strategy to learn which asset characteristics should be prioritized when creating the ranked assets list for substitution.

### 4.4.1 Comparison with Standard Convergence Strategy

The goal of the employing the learning convergence strategy is to achieve similar convergent results as the standard convergence strategy, but with significantly improved computational efficiency. The following figure 4.6 displays the portfolios generated by the learning convergence strategy alongside those from the Monte Carlo strategy. This graph mirrors figure 4.5 to show a clear comparison of performance between the two convergence strategies.

When comparing the figures 4.5 and 4.6, the portfolios produced by the learning convergence strategy are on par with those generated by the standard convergence strategy. Moreover, the final portfolio recommended by both strategies is the same, indicating that they converged towards the same optimal portfolio in this example.

Although both arrived at the same result, the learning strategy reached to it in just 550 iterations, compared to 4,000 iterations by the standard strategy. This demonstrates a significant decrease in computation time.

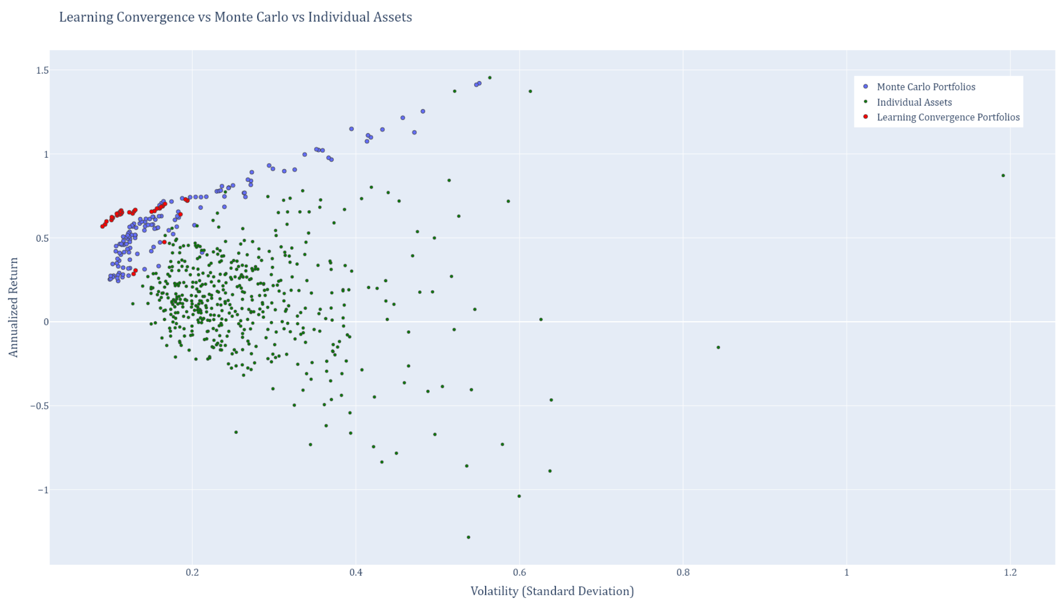


Figure 4.6: Comparing Learning Convergence Strategy with Monte Carlo Strategy

To accurately measure the efficiency improvement the learning strategy provided, an experiment was conducted to compare the performance of both approaches. Twenty random base portfolios were generated and used as starting points. The base portfolios were then inputted into both the standard convergence algorithm and the learning convergence algorithm to observe how they converged when given the same initial portfolio. For the learning strategy to demonstrate any meaningful improvement, we expect it to perform better on average in efficiency, while still producing a final portfolio that is on par with the standard model in terms of Sharpe ratio. The following figure 4.7 shows the results of the experiment.

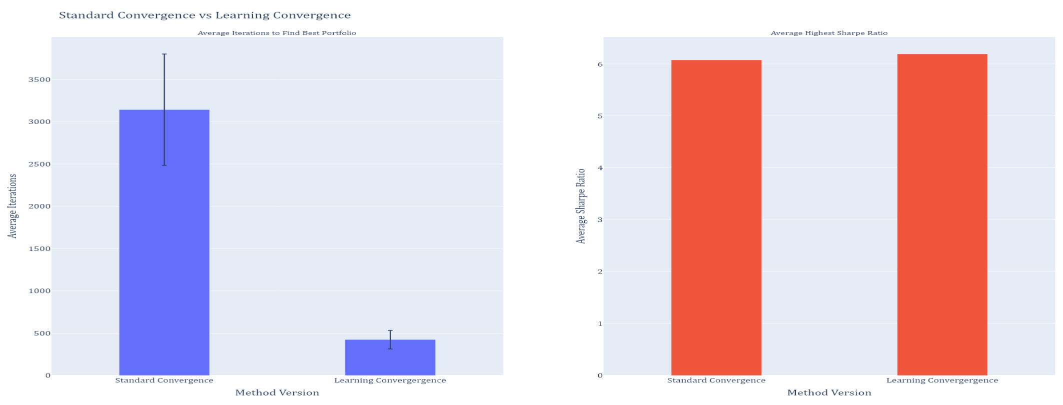


Figure 4.7: Performance of Learning Convergence vs Standard Convergence

[4]Figure 4.6 for comparison between Monte Carlo and learning convergence can be viewed at this link

As shown in right of figure 4.7, the average Sharpe ratios of the best portfolios found by each algorithm are nearly identical, meaning the output quality of the learning convergence strategy matches the standard approach. In the left graph, its shown that on average, the learning strategy requires much fewer iterations to reach the optimal portfolio and does so with much lower variation. This means that the learning strategy provides a consistent efficiency improvement without sacrificing quality.

## 4.5 Chapter Summary

To summarize this chapter, I detailed the four approaches used in developing my portfolio generation strategy: brute force method, Monte Carlo strategy, convergence strategy, and learning convergence strategy. I discussed the challenges of each approach and explained how subsequent strategies were designed to address those challenges.

The first portfolio generation strategy used a brute force method. The model evaluated all possible asset combinations and always returned the optimal portfolio based on the Sharpe ratio. However, its major drawback was efficiency. As the number of assets increased, the number of possible asset combinations grew factorially, making the approach impractical for large asset pools.

The second strategy addressed the computational inefficiency by using a Monte Carlo approach. It generated and evaluated random portfolios with randomly selected asset combinations, then returned the best portfolios from the sample. While this method improved efficiency, it could not guarantee that the final portfolio returned was optimal because the search was based on random sampling. Another drawback was its stochastic nature. Since it uses random sampling, there is no guarantee that the output portfolio will be the same each time the method is run, even when using the same dataset.

The third strategy uses the convergence strategy. Instead of relying on random sampling, this approach swaps assets from a base portfolio with assets from the pool. When a substitution leads to a higher Sharpe ratio, the new portfolio becomes the new base. This process continues until all potential asset candidates have been tested and no further improvement in the Sharpe ratio is found, implying that the current base is the optimal portfolio. Although this approach addresses the random walk problem from the Monte Carlo strategy, it does not necessarily provide better performance in terms of efficiency.

The fourth strategy incorporated reinforcement learning into the convergence strategy. The original convergence approach faced efficiency problems, as it often had to test many assets before finding one that improved the portfolio's Sharpe ratio. By using reinforcement learning, the learning convergence strategy learns to prioritize which assets from the pool are more likely to improve the portfolio. This greatly reduced the number of iterations it took to find the optimal portfolio and improved efficiency substantially without sacrificing output quality.

From this point forward, the learning convergence strategy will serve as the standard portfolio generation method, as it consistently produces near-optimal portfolios within a realistic computational time frame.

# Chapter 5

# Portfolio Price Prediction

Utilizing the learning convergence strategy, we can generate an optimal portfolio based on the Sharpe ratio. However, since this portfolio is constructed entirely from historical data, leaving it unmanaged is insufficient for long-term growth. Due to the inherent volatility of investments and the financial market, a portfolio that performed well historically might not continue to do so. This could stem from various factors such as shifts in market conditions, unexpected under-performance of invested companies, or the rise of competitors. Consequently, it is crucial to periodically assess the portfolio's performance and predict its future potential to determine whether adjustments are needed. This chapter details the machine learning approach used to predict future portfolio performance and explains how these predictions influence portfolio modifications.

## 5.1 Machine learning models

To predict the future performance of an investment portfolio, I employed machine learning techniques. A machine learning model was first trained on historical data, and then used to predict portfolio price changes over an arbitrary time step, with a default setting of five trading days — equivalent to one week. These price predictions are then compared with the real price data and subsequently used to make future predictions. This iterative process mirrors real-world portfolio management practices, where portfolio performance is evaluated on a weekly basis.

The three machine learning techniques I found suitable for this prediction task were Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs).

### 5.1.1 Recurrent Neural Networks

The first approach I explored involved Recurrent Neural Networks(RNNs). RNNs are particularly effective for time series predictions, as they are designed to recognize patterns in sequential data and retain information from previous time steps. This makes them suitable for modeling financial trends and predicting how a portfolio might perform over time.

RNNs process input data sequentially, where each prediction is influenced not only by the current input but also by information retained from previous time steps. In the context of portfolio price prediction, the model leverages both the actual and predicted prices of a given day as inputs. These inputs are weighted, combined with a bias term, and passed through an activation function to produce the prediction for the next day. Notably, linear activation functions are generally avoided, as portfolio prices tend to follow complex, nonlinear patterns. Unlike tasks such as image classification or object detection, which involve more discrete outcomes, financial data exists within a continuous range and is less black-and-white.

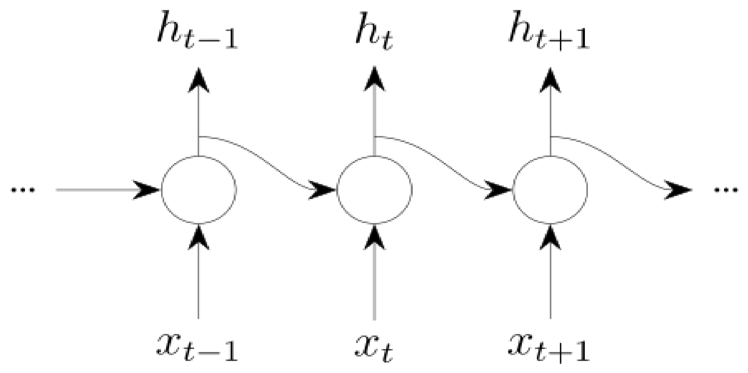


Figure 5.1: Diagram of Recurrent Neural Networks

Figure 5.1 is a basic demonstration of how RNNs work. There are 3 ‘blocks’ in the figure, the formula for each block is given as follows:

$$\forall_t, h_t = \tanh(h_{t-1}W_h + x_tW_x + b) \tag{5.1}$$

where $W_h$ is the weight matrix applied to output $h_{t-1}$ from the previous block, $W_x$ is the weight matrix applied to the current input $x_t$ and $b$ is a bias term. tanh is a common non-linear activation function used in recurrent neural networks. In the context of price prediction, the model combines both the predicted and actual prices of the given day as inputs to predict the price for the following day.

Although RNNs perform reasonably well over short time periods, their performance tends to degrade as time span increases. This is because standard RNNs often suffer from the vanishing or exploding gradient problem, which hinders their ability to learn long-term patterns in sequential data. Using the equation 5.1, if the weight matrix $W_h$ applied to output from the previous block is greater than 1, repeated applications over time can cause the gradients to grow exponentially — resulting in the exploding gradient problem. Conversely, if the weights are less than 1, the gradients diminish with each time step, eventually approaching zero — known as the vanishing gradient. This exploding or vanishing gradient makes it difficult for the network to retain information over long time periods as backpropagation becomes ineffective when gradients are either too large or small.

The following figure 5.2 demonstrates the exploding gradient problem encountered during training of an RNN for price prediction. As shown, the validation loss does not improve consistently across epochs but instead fluctuates significantly — often increasing sharply in one epoch and decreasing drastically in the next. This behavior is indicative of the exploding gradient problem. When gradients become too large during backpropagation, and the resulting weight updates between epochs are too extreme. As a result, the model struggles to converge and leads to a fluctuating loss rate as observed.

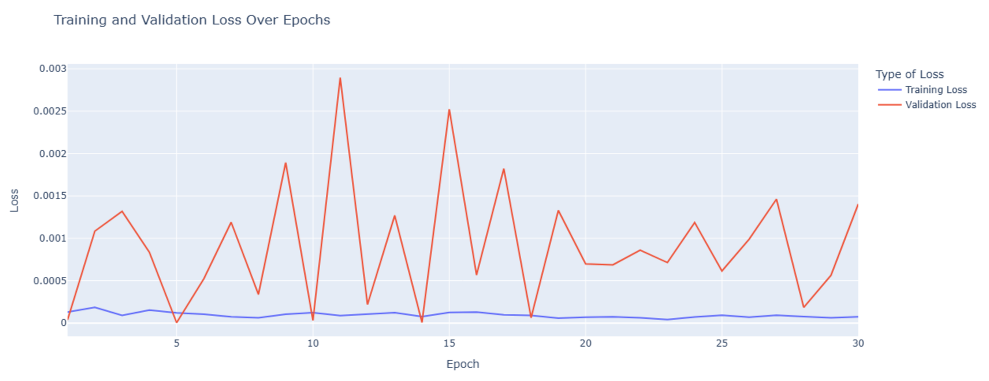


Figure 5.2: Loss rate of Recurrent Neural Network

### 5.1.2 Long Short Term Memory Neural Networks

To address the gradient issues, I employed a variant of RNNs known as Long Short-Term Memory (LSTM) networks. LSTMs introduce an additional component called memory, which is passed through each block as an additional input. This memory enables the network to retain and update information across longer sequences. Figure 5.3 below showcases the internal structure of a single LSTM block.

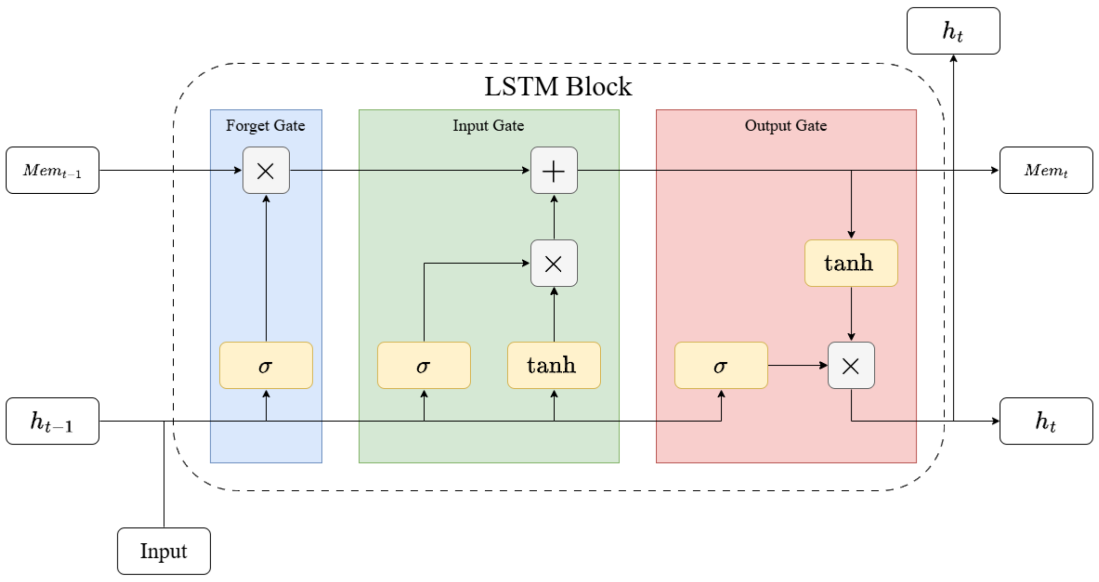


Figure 5.3: Diagram of Long Short Term Memory Network

Each LSTM block can be thought of as function used to predict the next time step in a sequence. It takes three inputs: the previous hidden state ($h_{t-1}$), the current input, and the previous memory state ($Mem_{t-1}$). It then produces two outputs: $h_t$, which represents the next hidden state, and an updated memory state $Mem_t$, which is passed on to the next block. In the context of price prediction, $h_{t-1}$ and $Input$ represent the predicted and actual price of a given day respectively. The memory ($Mem_{t-1}$) represents the historical information the model has learned in order to make future predictions. Using these inputs, the model outputs $h_t$, which is the predicted price for the following day.

The LSTM has three main components: the forget gate, the input gate, and the output gate:

1. **Forget Gate**: This gate decides what information should be removed from the memory. It takes the previous hidden state, the current input and passes them through a sigmoid activation function, producing values between 0 and 1. The hidden state and current input are combined similarly to equation 5.1 where weight matrices are applied to both and added together with a bias. However, instead of using a tanh activation function, the Forget Gate uses a sigmoid activation function, which outputs values between 0 and 1. These values act as filters, where values closer to 0 indicate that the all information stored in memory should be forgotten, and values closer to 1 indicate that it should be retained. In essence, the Forget Gate determines what percentage of the long-term memory should be remembered and what should be discarded.

2. **Input Gate**: This gate determines what new information should be added to the memory. There are two activation functions used in this gate. First, a tanh function takes in the previous prediction $h_{t-1}$ and the current input to generate candidate values that could be added to the memory. Afterwards, a sigmoid function decides what percentage of the new values should be added to the memory. For example, if the predicted value is close to the actual value, the model would want to preserve this information in memory, so the sigmoid output would be close to 1. Conversely, if the prediction deviates significantly from the actual value, the model would want to discard it as unreliable, resulting in a sigmoid output closer to 0. The product of these two activation functions is added to memory. Once added, this updated memory serves as an input for the Output Gate and is also passed forward to the next LSTM block.

3. **Output Gate**: Finally, the Output Gate determines the next hidden state, which in this context represents the next predicted price. It uses the updated memory, the previous hidden state, and the current input to compute the output. First, the current input and previous hidden state are combined and passed through a sigmoid activation function. The updated memory is passed through a tanh function. The product of these two outputs forms the new hidden state $h_t$, which serves as the predicted price. Both the updated memory and this predicted price are then passed forward to subsequent LSTM blocks for predicting future prices.

In essence, each LSTM block takes in three main components: the current input, the previous hidden state, and the previous memory state. These are processed through the gates to update the memory and generate the next hidden state, which is the predicted price. LSTMs address the gradient problems by maintaining and selectively updating a memory. This allows the model to retain important information over long sequences, enabling it to learn from past predictions and use that info to make future predictions.

The following figure 5.4 demonstrates the effectiveness of the LSTM model in addressing gradient issues. This LSTM was trained on the same dataset as the RNN shown in Figure 5.2. However, its validation loss steadily decreases and approaches zero, implying that the model is not affected by exploding or vanishing gradients. In contrast to the unstable loss pattern observed in the RNN shown in figure 5.2, the LSTM exhibits significantly better performance, with a more stable and consistent reduction in loss rate across epochs. This implies that the LSTM is learning meaningful patterns suitable for price prediction tasks.

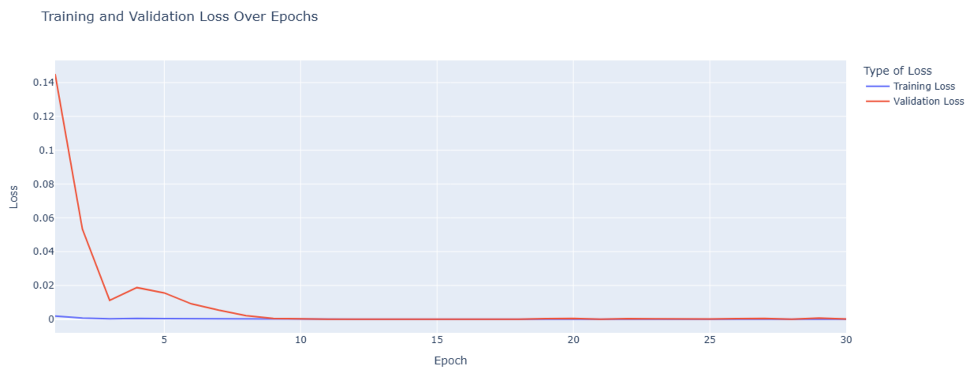


Figure 5.4: Loss rate of Long Short Term Memory Model

### 5.1.3 Gated Recurrent Units

The last machine learning technique explored was Gated Recurrent Units (GRUs). GRUs function similarly to LSTMs, as they also incorporate a memory unit designed to retain past information for use in future predictions. However, the key difference is that GRUs simplify the architecture by combining the forget and input gates found in LSTMs into a single ‘update gate’, thus reducing complexity and computational costs.

The main issue encountered when using GRUs is their diminished performance over longer time periods. Since GRUs combine the forget and input gates into a single update gate, their ability to precisely control memory updates at each block is reduced. Consequently, as the number of blocks increases — corresponding to a longer time span — the memory retention becomes less effective, leading to reduced accuracy in future predictions. In testing, while GRUs performed well when trained on shorter time periods, such as three months, their performance significantly declined when the training period was extended beyond six months.

As a result, although GRUs offered better efficiency than LSTMs due to their simpler architecture, their inability to effectively learn long-term dependencies made them a less suitable choice for portfolio price prediction.

## 5.2 LSTM Model setup

To set up an LSTM model for portfolio price prediction, a standard sequential model was imported from TensorFlow's Keras library and used as the foundation for building the LSTM architecture. Sequential models serve as a blueprint for constructing neural networks by allowing layers to be added one at a time in a linear order. Data flows from the input layer to the output layer in a single, forward direction, making this architecture well-suited for tasks where the order of data is important, such as price prediction. Building the LSTM architecture with the sequential model involved configuring three key components: the activation functions, the loss function, and the layers of the network.

The activation functions used within the LSTM model are the same as those illustrated in figure 5.3, where sigmoid ($\sigma$) and tanh functions are used to introduce non-linearity into the model.

For the task of portfolio price prediction, I developed a custom loss function called weighted mean squared error (WMSE). This loss function is designed to emphasize more on recent data by assigning progressively larger weights to later predictions and inputs. Doing so, the model learns to place greater emphasis on recent inputs, which are typically more useful in predicting price changes than older data.

In terms of architecture, the model consists of two stacked LSTM layers, both imported from TensorFlow's Keras library. The decision to use two layers was made to limit the risk of overfitting. The first LSTM layer contains 250 neurons and is configured to return the full sequence of hidden states for each time step in the input sequence. The output sequence from this layer is passed through a dropout layer with a rate of 0.2 to reduce overfitting. The second LSTM layer has 50 neurons and returns only the final hidden state, effectively summarizing the entire sequence into a single vector. Another dropout layer, also with a rate of 0.2, is applied for additional regularization. Finally, a dense layer maps the output vector from the previous LSTM layer to a single value, representing the predicted portfolio price. A basic outline of the model is as follows:

**Algorithm 4** Build and Compile LSTM Model

```
1: function INITIALIZE TENSORFLOW SEQUENTIAL MODEL
2:     Add LSTM layer with 250 units, tanh activation, return_sequences=True
3:     Add Dropout layer with rate 0.2
4:     Add LSTM layer with 50 units, tanh activation, return_sequences=False
5:     Add Dropout layer with rate 0.2
6:     Add Dense layer with 1 unit
7: end function
8: Compile model with weighted MSE loss function
```

## 5.3 Model Training and Price Prediction

With the LSTM model configured, the next step is to train it on historical price data and begin making price predictions. However, before training, determining the optimal length of training data is crucial. For instance, would a model trained on five years of data significantly outperform one trained on only two years? To investigate this, an experiment was conducted to evaluate how varying the length of the training period affects the model's accuracy.

The experiment aimed to predict portfolio prices over the course of one year, from 01-01-2024 to 01-01-2025. The model was tested using different lengths of historical training data. During testing, the model predicted prices for five consecutive days at a time, then compared those predictions with the actual prices. The differences between the predicted and actual values were averaged to determine the overall prediction accuracy. The following table 5.1 shows the results of the experiment. The "Training Time Period" refers to the length of historical data used to train the model, measured as the time prior to 01-01-2024. The experiment was conducted within a Google Colab Notebook environment to ensure a standardized and reproducible setup.

Table 5.1: Accuracy and Error of model trained on different time lengths

| **Training Time Period** | 3 Months | 6 Months | 1 Year | 2 Years | 5 Years |
|---|---|---|---|---|---|
| **Average Accuracy (%)** | 85.32 | 97.73 | 97.80 | 98.13 | 95.35 |
| **Average Error (%)** | 14.68 | 2.27 | 2.21 | 1.87 | 4.64 |
| **Training Time (s)** | 4.5 | 4.7 | 5.1 | 5.3 | 6.2 |

As shown, the model generally performs better as the training period increases. When trained on only three months of data, the accuracy is significantly lower, suggesting that the model lacks sufficient data to learn meaningful patterns. In contrast, when trained on five years of data, the prediction accuracy slightly declines, which may indicate that the model is overfitting to older, less relevant data. Overall, training periods between six months and two years appear to provide the model with enough data to learn effectively and achieve high predictive accuracy.

The following figure 5.5 demonstrates the LSTM model trained on one year of historical data, from 2023 to 2024, and subsequently used to predict portfolio prices over the following year, from 2024 to 2025. The model predicts one week of prices at a time, compare them with the actual values, and then use this this information to make predictions for the following week—repeating this cycle for an entire year.

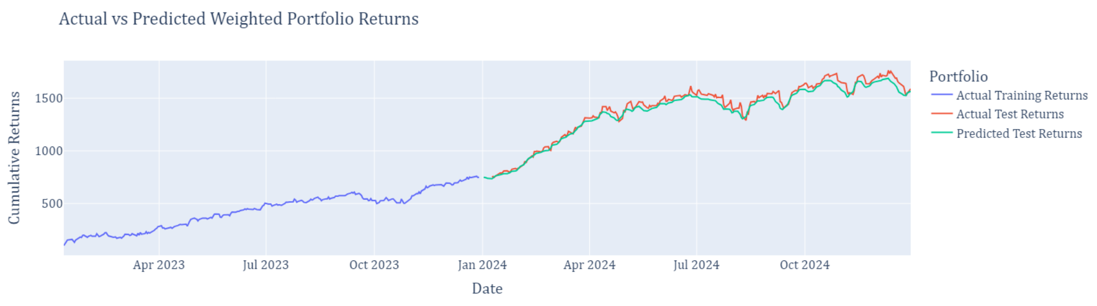


Figure 5.5: LSTM Model trained on 1 year to predict
Figure 5.5 for portfolio price prediction from 2024 to 2025 can be viewed at this link

### 5.3.1 Portfolio Grading Tool

With a portfolio price predictor in place, it may seem redundant to use the Sharpe ratio to evaluate portfolio performance, since the predictor is capable of forecasting future prices directly. However, the main issue is that the portfolio predictor is trained on the historical price data of a single portfolio. As a result, the model is specific to that portfolio and cannot be directly applied to others. This means a new LSTM model must be trained separately for each portfolio, which is computationally expensive. The average time taken to train an LSTM model can be seen in table 5.1 and training thousands of models to evaluate portfolios is not practical.

This is where the Sharpe ratio becomes useful as it serves as a grading tool that filters out portfolios that are unlikely to perform well according to historical data. The LSTM predictor can then be applied to the remaining portfolios with high Sharpe ratios, and determine which is likely to deliver the highest returns in the near future.

## 5.4 Chapter Summary

To summarize this chapter, I explained my rationale behind selecting LSTM models as the portfolio predictor, outlining why alternative techniques were less suitable. I also detailed the setup process and described the optimal training parameters used to achieve the best possible performance.

The three machine learning techniques explored were Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), and Long Short Term Memory models (LSTMs). RNNs were explored first but were quickly deemed unsuitable. Although they are designed to recognize patterns in sequential data and retain information from previous time steps, they suffer from a significant drawback: the exploding and vanishing gradient problem, reducing their effectiveness over longer sequences. GRUs and LSTMs address this issue by introducing a memory state that helps retain important information over time. Ultimately, LSTMs were chosen over GRUs, as they performed significantly better over long time sequences. While GRUs had a simpler architecture, their memory state is less precise, making them worse at capturing long-term dependencies, which is essential for price prediction.

Following this, I detailed the steps taken to set up the LSTM model. The model consists of two LSTM layers, a dense output layer, and dropout layers in between. A shallow architecture was chosen to reduce the risk of overfitting. Additionally, a custom loss function — Weighted Mean Squared Error — was implemented to place greater emphasis on recent data, enabling the model to better respond to recent market trends. Finally, non-linear activation functions such as sigmoid and tanh were used. Since price prediction is not a binary classification task but involves predicting values across a range, linear activation functions such as ReLU are less suitable.

Lastly, I outlined the training parameters used for the LSTM model. Through experimentation, I found that training the model between 6 months to 2 years of historical data prior to the prediction period resulted in the highest accuracy. The Robo-advisor will be standardized to using 2 years of training data as it yielded the highest accuracy.

# Chapter 6

# Demonstration and Testing

With both the portfolio generator and the portfolio price predictor in place, we can begin utilizing the Robo-advisor. This chapter provides a comprehensive demonstration of its functionality and details each step of the process.

## 6.1 Robo-advisor Procedure

The following pseudo-code is a simplified version of the Robo-advisor procedure:

**Algorithm 5** Robo-advisor Process

```
procedure ROBO-ADVISOR($Investment\ Period$, $Evaluation\ Window$)
    $Number\ of\ TimeSteps \leftarrow Investment\ Period / Evaluation\ Window$
    $Chosen\ Portfolios\ List \leftarrow \{\}$
    for t = 1 to $Number\ of\ TimeSteps$ do
        $Candidate\ Portfolios \leftarrow$ Learning Convergence Algorithm($Assets\ List$)
        $Filtered\ Candidate\ Portfolios \leftarrow$ Strategic Sampler($Candidate\ Portfolios$)
        $Highest\ Predicted \leftarrow -\infty$
        $Chosen\ Portfolio \leftarrow$ None
        for all $Portfolio \in Filtered\ Candidate\ Portfolios$ do
            $Predicted\ Price \leftarrow$ Portfolio Predictor($Portfolio$, t)
            if $Predicted\ Price > Highest\ Predicted$ then
                $Highest\ Predicted \leftarrow Predicted\ Price$
                $Chosen\ Portfolio \leftarrow Portfolio$
            end if
        end for
        if $Highest\ Predicted > 0$ then
            Append $Chosen\ Portfolio$ to $Chosen\ Portfolios\ List$
        end if
    end for
    return $Chosen\ Portfolios\ List$
end procedure
```

The first step of the Robo-advisor is to generate a base portfolio. This is done using the learning convergence algorithm, which identifies the optimal portfolio according to the Sharpe ratio. However, relying solely on this portfolio is not ideal, as it is derived entirely from historical data, and its future returns are uncertain. As a result, blindly investing in it may not lead to positive returns.

Instead, the Robo-advisor utilizes not just the final optimal portfolio according to the Sharpe ratio but also the list of 'improvement portfolios' returned by the learning convergence algorithm. This list includes all portfolios that had an improvement in Sharpe ratio during the optimization process. Notably, many portfolios toward the end of this list have Sharpe ratios that are very close to that of the optimal portfolio. This occurs because the algorithm initially finds portfolios with substantial improvements, but as optimization progresses, it begins to converge — resulting in small, incremental gains in Sharpe ratio. This phenomenon can observed in figure 6.1 where the increase in Sharpe ratio diminishes as more improvement portfolios are found.

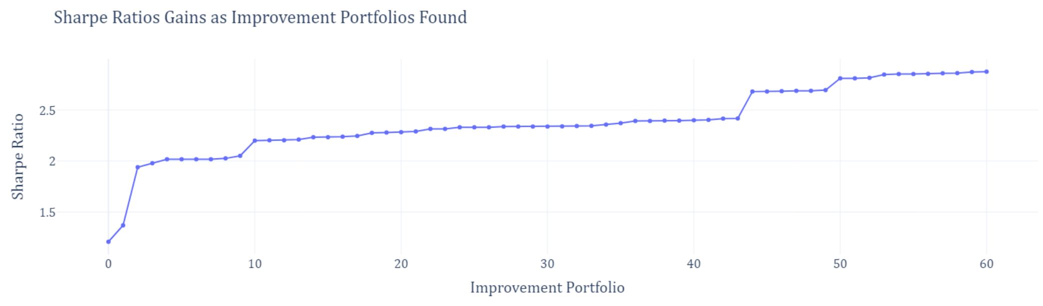


Figure 6.1: Sharpe Gains with Improvement Portfolios Found

Using the improvement portfolios list, the Robo-advisor selects portfolios with Sharpe ratios within 25% of the optimal portfolio's Sharpe ratio. A wide range of portfolios is selected because in many cases, portfolios with similar Sharpe ratios also have similar asset compositions. As a result, narrowing the selection to just the portfolios with the highest Sharpe ratios much may result in evaluating redundant portfolios. This range ensures that a diverse set of portfolios is considered during evaluation. In the case of figure 6.1, there are approximately 40 portfolios in the list with Sharpe ratios within 25% of the optimal portfolio. These selected portfolios, referred to as *candidate portfolios*, are then evaluated using the portfolio predictor to estimate their expected returns.

When the candidates list becomes too large, a strategic sampling method is used to determine which portfolios are to be evaluated. Candidate portfolios are divided into a fixed number of groups, each representing a segment of the overall Sharpe ratio range. From each group, two portfolios are randomly selected to serve as representatives of that specific segment. In figure 6.1, since there are 40 candidate portfolios with Sharpe ratios ranging from 2.3 to 2.86, a representative subset of eight is selected to cover the full range within that interval. This approach maintains computational efficiency while preserving diversity in portfolio evaluation.

Once the representative candidate portfolios are selected, the portfolio predictor is applied to estimate expected returns for each portfolio. The portfolio with the highest predicted return is selected as the base portfolio. If none of them are predicted to generate positive returns, no portfolio is held, as this implies that not investing and simply holding cash would be a better option during that period.

Regardless of whether a base portfolio has previously been established, the portfolio generation process is repeated each week. If a base portfolio exists, it is included in the list of candidate portfolios for that period. The portfolio predictor is again used to estimate expected returns for the candidates, and the portfolio with the highest predicted return is selected as the new base portfolio. This iterative process continues on a weekly basis until the end of the investment period.

It is important to note that this process more closely resembles portfolio reconstruction rather than modification. Since portfolio selection is entirely driven by the latest predictions, a completely different portfolio could be chosen each week, potentially resulting in substantial changes in portfolio composition.

## 6.2 Demonstration

To demonstrate the Robo-advisor, I simulated an investment period from January 1, 2023, to January 1, 2025 — a two-year time frame. The asset list included over 1,000 investment options, comprising of ETFs, bonds, and stocks. Historical price data spanning from 2021 to 2025 was collected, with data from the two years preceding the investment period used for portfolio construction based on the Sharpe ratio, and also for training the portfolio prediction model. The Robo-advisor was configured to evaluate and adjust the portfolio on a weekly basis. To measure its performance, the returns of the NASDAQ Index fund over the same period were included for comparison. The following are the results of the simulation:

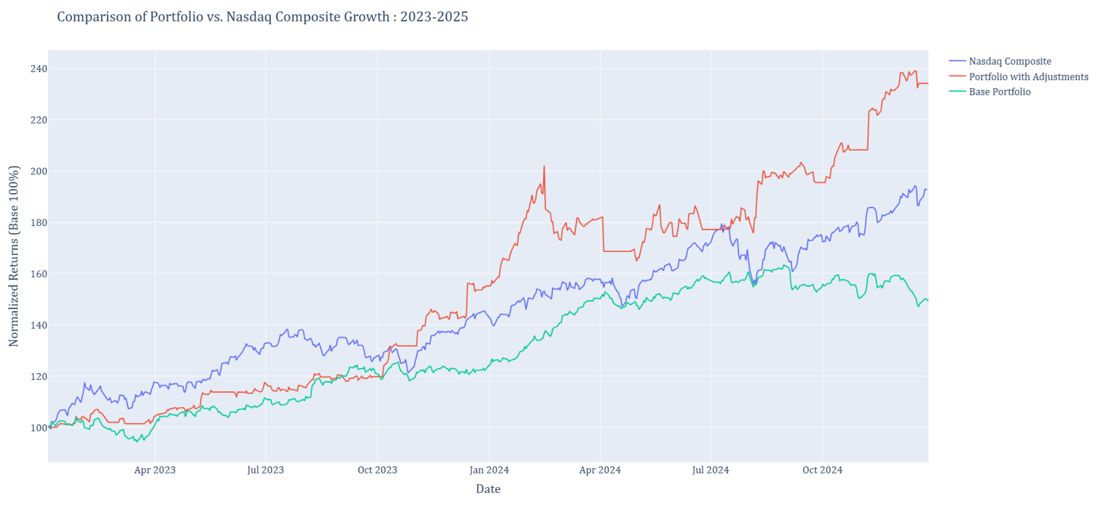


Figure 6.2: Comparing Robo-advisor with NASDAQ[1]

[1]Figure 6.2 for the performance of Robo-advisor portfolio can be viewed at this link

The figure 6.2 displays three lines: the blue line represents the NASDAQ Index fund, the green line shows the initial base portfolio, which remained unchanged over the two-year period, and the orange line represents the Robo-advisor-managed portfolio, which was periodically evaluated and adjusted. As shown, the Robo-advisor portfolio achieved the highest returns, followed by the NASDAQ Index, while the static base portfolio performed the worst. This highlights the importance of periodically adjusting a portfolio, as assets that performed well in the past may not continue to do so in the future. The flat segments in the orange line indicate periods where the Robo-advisor determined that not investing was the optimal decision based on its price predictions. During these periods, the portfolio was empty, resulting in no changes in returns.

Table 6.1: Percentage Returns of Each Investment at Different Points in Time

| **Returns at** | 6 Month | 1 Year | 1.5 Year | 2 Years | **Annual Returns** |
|---|---|---|---|---|---|
| Base Portfolio | 11% | 26% | 57% | 50% | 25% |
| Robo-advisor Portfolio | 16% | 55% | 77% | 134% | 67% |
| NASDAQ INDEX | 32% | 40% | 72% | 93% | 46.5 |

In table 6.1, the percentage returns of each investment are shown at 6-month, 1-year, 1.5-year, and 2-year intervals. While the base portfolio performed the worst of the three, it should be noted that an annual return of 25% is still exceptionally strong. For comparison, risk-free investments such as U.S. Treasuries typically yield around 4.2% annually. Additionally, it's worth highlighting that the NASDAQ Index performed exceptionally well during the two-year span from 2023 to 2024. Historically, the NASDAQ Index has had an average annualized return of 18.4% over the past 10 years. However, during 2023 and 2024, it averaged an impressive 46.5% annually, marking some of its best years in recent history.

The Robo-advisor portfolio returned 55% in the first year and 79% in the second year, averaging 67% annually. Notably, the Robo-advisor portfolio performed weaker than the NASDAQ Index in the first 6 months. Upon further investigation, this can be attributed to the US market decline in 2022, caused by factors such as historically high inflation and the Russia–Ukraine war, which lead to significant losses for many NASDAQ stocks. When the market rebounded in early 2023, these same stocks saw rapid growth. However, since the Robo-advisor initially generates portfolios using the Sharpe ratio — which is based entirely on historical performance — it excluded many of these stocks due to their poor returns in 2022. As a result, the Robo-advisor portfolio initially underperformed the NASDAQ Index.

Overall, while the Robo-advisor portfolio achieved the highest returns in the simulation, there is still room for further improvement. The simulation took roughly 90 minutes to run, with each weekly evaluation averaging around 50 seconds. To keep the runtime manageable, the number of candidate portfolios used for prediction was reduced, limited to just 8 portfolios per weekly evaluation. In a real-world scenario, where an investor could afford longer runtimes each week, the portfolio candidates list could be increased significantly. Evaluating a wider range of portfolios would significantly improve the likelihood of selecting one with higher return potential.

## 6.3 Testing and Results

To evaluate the overall performance of the Robo-advisor, I conducted simulations across various investment periods, ranging from short-term investments of 3 months to long-term investments spanning 3 years. These simulations were run across randomly selected time periods between 2010 and 2025. This chapter will detail the tests performed and analyze the results.

### 6.3.1 Simulation Standards

**Asset List** – An asset list comprised of over 1,000 investment options was used for all simulations. The asset list mainly consisted of U.S. based investments, including S&P 500 company stocks, American ETFs, and U.S. corporate bonds. Of the total assets, approximately 60% were stocks, 25% were ETFs, and 15% were bonds.

**Investment Period** – Simulations were run between 2010 and 2025, allowing investment scenarios at any point within this 15-year period. Although earlier time periods could technically be included, this time frame is sufficient for the purposes of testing.

**Periodic Evaluation** – The Robo-advisor was set to re-evaluate the portfolio weekly, reflecting typical real-world investment behavior.

**Risk Level** – A neutral risk level was used, meaning the portfolio generation method produced optimal portfolios based on the Sharpe ratio, which reflects the best historical risk-adjusted returns.

**Portfolio Predictor Training Period** – Two years of historical price data were used to train the LSTM model for the Portfolio Predictor across all simulations. Based on prior experiments, this two-year window was found to provide sufficient data for the model to learn effectively and achieve high predictive accuracy without overfitting.

### 6.3.2 Simulation Results

Short-term investments were defined as periods from 3 to 6 months, while long-term investments ranged from 1 to 3 years. Although testing periods of 5 years or more was considered, there were runtime concerns as running a single 3-year simulation already took approximately 4.5 hours, making it impractical to conduct multiple long-duration tests. The following tables show the results of the simulations:

Table 6.2: Average return of Robo-advisor across Different investment durations

| **Average Returns(%)** | 3 Months | 6 Months | 1 Year | 2 Years | 3 Years |
|---|---|---|---|---|---|
| **Robo-advisor Portfolio** | 10.18% | 19.04% | 52.04% | 143.8% | 236.8% |
| **Base Portfolio** | 6.54% | 10.90% | 11.96% | 30.60% | 61.03% |
| **NASDAQ Index Fund** | 4.16% | 4.40% | 17.32% | 37.82% | 51.00% |

Table 6.2 presents the average returns of the base portfolio, the Robo-advisor portfolio, and the NASDAQ Index across different investment durations: 3 months, 6 months, 1

year, 2 years, and 3 years. Detailed returns for each investment period can be seen in figures stored in the Appendix A.

Table 6.3: Average Number of unique assets held by the Robo-advisor

| | 3 Months | 6 Months | 1 Year | 2 Years | 3 Years |
|---|---|---|---|---|---|
| **Number of unique assets** | 18 | 38 | 77 | 186 | 259 |
| **Average adjustments per week** | 1.38 | 1.52 | 1.54 | 1.71 | 1.73 |

Table 6.3 presents the average number of unique assets that appeared in the Robo-advisor portfolio at any point during different investment periods, along with the average number of assets adjustments each week. For example, over a 2-year period, Robo-advisor portfolios on average held a total of 186 different assets at various points in time, with an average of 1.71 asset changes per week during that period.

## 6.4 Evaluation

Firstly, across all investment periods, the Robo-advisor consistently achieves higher average returns than the NASDAQ Index fund. The comparison with NASDAQ is especially important, as the fund is one of the most popular and widely trusted investor funds. Consistently outperforming it by a significant margin demonstrates the strong performance of the Robo-advisor. Furthermore, a closer examination reveals that during periods when both the NASDAQ Index and a static base portfolio experience negative returns, the Robo-advisor often manages to lower losses. This is largely due to the portfolio predictor component, which, when anticipating negative returns across all portfolios, prompts the Robo-advisor not to invest and avoid unnecessary losses.

An interesting observation from the simulation results is the Robo-advisor's ability to amplify gains during strong NASDAQ performance. Since both operate within the U.S. market, a certain level of correlation is expected. However, the results show that the Robo-advisor often outperforms NASDAQ substantially during market upswings. This phenomenon can be seen in tables A.3, A.4, and A.5, where the Robo-advisor amplifies gains during periods of strong NASDAQ performance. This is likely due to its ability to identify and allocate heavily towards top-performing assets. While NASDAQ benefits from these high earners as part of a diversified index, the Robo-advisor strategically concentrates its positions on them, thus capturing even more upside.

For shorter investment periods — such as 3 and 6 months — the static base portfolio performs comparably to the Robo-advisor portfolio, and in some cases even outperforms it. However, as the investment period extends, the base portfolio begins to fall behind. This suggests that while past performance can continue over in the short term, it is not reliable over longer periods, as it is difficult to predict how an asset or company will evolve. This emphasizes the importance of periodically assessing and adjusting the portfolio for long-term performance, and that a static portfolio alone is insufficient.

Longer investment periods are where the Robo-advisor truly shines. This may be because, over extended time horizons, markets tend to experience greater fluctuations

— both upward and downward — offering more opportunities for the Robo-advisor to demonstrate its effectiveness. During periods of market growth, it can identify and capitalize on high-performing assets to maximize returns. Conversely, during market downturns, its predictive algorithms can help reduce losses by adjusting the portfolio accordingly or opting to not invest altogether. In contrast, short-term investment periods often lack the market and price movement needed for the Robo-advisor to fully demonstrate its capabilities, as limited price fluctuations make it harder to capture gains or mitigate losses.

### 6.4.1 Portfolio Assets Evaluation

In table 6.3, we observe that the average number of unique assets that appeared in the Robo-advisor portfolio at one point increases with the investment period. Similarly, the average number of asset changes per week also increases with duration, indicating more frequent adjustments over longer durations. This suggests that over longer durations, the Robo-advisor detects more shifts in market conditions and adjusts the portfolio accordingly. This reflects the dynamic nature of the market, where investment opportunities change frequently, and so the Robo-advisor adapts to find these opportunities. In contrast, shorter investment periods show fewer asset swaps per week. This is likely because market conditions remain relatively stable over those shorter windows, reducing the need for reallocation.

Another interesting observation is that the Robo-advisor strongly favors stocks over ETFs and bonds. On average, the unique assets held in the portfolio consisted of approximately 96.4% stocks, 2.6% ETFs, and only 1% bonds[2]. This indicates that most portfolios during the investment period were primarily composed of stocks, with ETFs and bonds appearing occasionally. After further analysis, it was found that this preference emerges during the portfolio generation step, where assets are selected based on whether they improve the overall portfolio Sharpe ratio. While many individual bonds and ETFs display higher Sharpe ratios in isolation, they are less frequently selected when constructing a portfolio optimized for overall Sharpe performance. This aligns with the Markowitz model, which suggests that diversification reduces portfolio risk while potentially increasing expected returns. In this context, even if individual stocks have lower Sharpe ratios, their collective contribution to diversification can result in better overall risk-adjusted returns than portfolios including bonds or ETFs.

Additionally, the Robo-advisor appears to favor stocks even more heavily in shorter investment periods compared to longer ones. In table A.6, 98% of the unique assets held by the Robo-advisor portfolio during shorter time periods were stocks. In contrast, during longer time periods, stocks made up around 95% of the unique assets. This may be explained by the fact that, over longer periods, there is a higher likelihood of market downturns and increased overall volatility. During these periods, the algorithm may favor more stable assets like bonds and ETFs, which offer lower volatility and help reduce overall portfolio risk. In contrast, significant downturns are less likely to occur within shorter time frames, making stocks more favorable due to their typically higher average returns.

[2]Detailed percentages of asset types can be viewed in table A.6 in Appendix section A.2

# Chapter 7

# Discussion and Conclusion

## 7.1 Project Objective

The main objective of this project was to explore the feasibility and challenges of developing an investment Robo-advisor. This was achieved by using freely available sources and open-source tools to design and integrate individual components into a fully functioning investment Robo-advisor system. In addition to data collection and processing, the system was comprised of two main components: **Portfolio Generation** and **Portfolio Prediction**.

**1. Portfolio Generation: Learning Convergence Algorithm**
Portfolio Generation involves constructing portfolios from a list of assets and optimizing them to achieve the highest possible Sharpe ratio. This component underwent several iterations during development, evolving from a brute-force strategy to a Monte Carlo approach, then to a convergence strategy, and ultimately to the learning convergence strategy. The function begins by generating a random base portfolio and iteratively swaps assets in and out, progressively optimizing toward the portfolio that yields the highest Sharpe ratio. However, the portfolio with the highest historical Sharpe ratio proved insufficient, as this metric is based entirely on past performance and offers no indication of future returns or market behavior.

**2. Portfolio Prediction: Long Short Term Memory Model**
The Portfolio Prediction component addresses the problem of having no indication of future behavior by using a Long Short-Term Memory (LSTM) model to estimate how the price of a portfolio may change over time. Given the list of candidate portfolios generated by the Portfolio Generation component, the predictor evaluates their potential future performance and identifies the one with the greatest upside potential.

**3. Integration of Components**
The Robo-advisor system integrates both components to perform portfolio creation, performance evaluation, and asset adjustments. First, a starting portfolio is established by using the Portfolio Generation function to produce a list of candidate portfolios, which are then evaluated by the Portfolio Predictor to identify the most promising one. For evaluation, at each time step—defaulting to one week—the Robo-advisor reruns

the algorithm to assess whether the current portfolio is expected to deliver favorable returns and whether any adjustments are necessary. This iterative process continues until the end of the investment period.

This project successfully demonstrated the steps and challenges involved in building an investment Robo-advisor, detailing each component and ultimately integrating them into a cohesive and functional system. Furthermore, after testing the Robo-advisor in U.S. markets, it outperformed one of the most trusted investor funds in NASDAQ. This indicates that the objective of achieving consistent, positive returns was met, and that the core algorithm of the Robo-advisor performs effectively in real-market conditions.

## 7.2 Limitations and Constraints

This section outlines key limitations of the current iteration of the Robo-advisor, along with potential solutions to address them. The primary issues include the portfolio evaluation metric during portfolio generation, as well as the omission of trading costs.

### 7.2.1 Portfolio Evaluation Metric

One of the main limitations of the Robo-advisor is that it does not account for recent market trends in its portfolio generation process. As a result, trending assets are often excluded from candidate portfolios and are not evaluated by the portfolio predictor. This issue arises from the evaluation metric used in the portfolio generation algorithm.

Currently, the algorithm evaluates portfolios using the Sharpe ratio. If substituting an asset results in a higher portfolio Sharpe ratio, the change is considered an improvement. However, this method can overlook assets that are currently performing well but have weaker historical performance. When such an asset is substituted into the portfolio, historical data may lower the overall portfolio Sharpe ratio, even though its recent momentum could enhance portfolio returns. As a result, the algorithm may reject positive substitutions due to its reliance on historical metrics and its inability to account for the latest trends or asset momentum.

To address this limitation, I experimented with applying weights to the historical price data, putting greater emphasis on recent returns. This was designed to help the algorithm better capture recent price changes and increase its sensitivity to current market trends. However, while weighting historical returns is a step in the right direction, it does not fully resolve the underlying issue as the Sharpe ratio still fundamentally relies on historical data. As a result, if an asset has only recently started to perform well but has a long history of under-performance or high volatility, its contribution to the portfolio Sharpe ratio may still appear unfavorable.

I believe the most effective way to address this issue is to introduce an additional portfolio evaluation metric called **momentum**, alongside the Sharpe ratio. This momentum score would be designed to capture recent market trends, short-term performance, and other factors that signal an asset's current trajectory. By incorporating this score, the algorithm could better identify assets with strong recent momentum, even if their long-term historical performance is weaker. Ultimately, using momentum alongside the Sharpe ratio as evaluation metrics could lead to more balanced portfolio generation.

### 7.2.2 Trading Costs

Another limitation of the Robo-advisor is its current oversight of trading costs. With each evaluation step, the Robo-advisor reruns its algorithm using the latest price data, identifies the best predicted portfolio among the candidate options, and reallocates the entire portfolio to match that selection. As a result, this process resembles portfolio reconstruction more than adjustments. Although the simulation results indicate that on average, each weekly evaluations lead to only one or two asset changes, a complete portfolio overhaul is still possible. This is unrealistic in real-world investing, as it ignores the impact of trading costs, which can significantly affect overall performance.

Addressing this oversight is particularly difficult. The current data collection method, which relies on the Yahoo Finance API, only provides asset-related data and does not contain any information on trading costs. While some APIs may offer trading cost estimates, they are often unreliable due to the significant variability across brokers and investment platforms. As a result, incorporating trading cost data into the Robo-advisor as a factor for evaluation remains an unresolved challenge.

## 7.3 Future Work

One of the best ways to further improve the Robo-advisor is to incorporate natural language processing (NLP). NLP would allow the system to analyze textual data, such as financial news, earnings reports, and social media sentiment. By combining this textual information with traditional market data, the Robo-advisor could gain a deeper understanding of market trends and investor sentiment. For example, NLP can help detect early signals of market shifts not yet reflected in price movements and uncover emerging patterns related to specific assets or sectors. Ultimately, incorporating NLP makes the Robo-advisor more adaptive and responsive to market trends.

Additionally, after extensive testing within the U.S. investment market, it is evident that the core algorithm of the Robo-advisor performs effectively. The next step is to adapt and evaluate the system across other global markets. Doing so will test its generalizability and help determine whether the Robo-advisor can be applied and optimized in different economic environments, asset classes, and market conditions.

# Bibliography

[1] Patrick Bolton, Xavier Freixas, and Joel Shapiro. “Conflicts of interest, information provision, and competition in the financial services industry”. In: *Journal of Financial Economics* 85.2 (2007), pp. 297–330.

[2] Lukas Brenner and Tobias Meyll. “Robo-advisors: A substitute for human financial advice?” In: *Journal of Behavioral and Experimental Finance* 25 (2020), p. 100275. ISSN: 2214-6350. DOI: `https://doi.org/10.1016/j.jbef.2020.100275`. URL: `https://www.sciencedirect.com/science/article/pii/S2214635019301881`.

[3] Jacques Bughin et al. “Artificial intelligence the next digital frontier”. In: (2017).

[4] Crispin Coombs and Alex Redman. “The impact of robo-advice on financial advisers: a qualitative case study”. In: (2018).

[5] Paulo Costa and Jane E Henshaw. *Quantifying the investor’s view on the value of human and robo-advice*. Tech. rep. Vanguard Research Report. 1, 2022.

[6] Francesco D’Acunto, Nagpurnanand Prabhala, and Alberto G Rossi. “The promises and pitfalls of robo-advising”. In: *The Review of Financial Studies* 32.5 (2019), pp. 1983–2020.

[7] Frank J Fabozzi et al. “Mean-Variance Model for Portfolio Selection”. In: *Encyclopedia of Financial Models* (2012).

[8] Jill E Fisch, Marion Laboure, and John A Turner. “The Emergence of the Robo-advisor”. In: *The disruptive impact of FinTech on retirement systems* 13 (2019), pp. 13–37.

[9] Wilfredo Graterol et al. “Emotion detection for social robots based on NLP transformers and an emotion ontology”. In: *Sensors* 21.4 (2021), p. 1322.

[10] Toni Hilton et al. “Adopting self-service technology to do more with less”. In: *Journal of Services Marketing* 27.1 (2013), pp. 3–12.

[11] Karl G Jöreskog. “Structural analysis of covariance and correlation matrices”. In: *Psychometrika* 43.4 (1978), pp. 443–477.

[12] Pegah Kolbadi and Hamed Ahmadinia. “Examining Sharp, Sortino and Sterling ratios in portfolio management, evidence from Tehran stock exchange”. In: *International Journal of Business and Management* 6.4 (2011), pp. 222–236.

[13] Matthew L Meuter et al. “Self-service technologies: understanding customer satisfaction with technology-based service encounters”. In: *Journal of marketing* 64.3 (2000), pp. 50–64.

[14] “Robo-advisory: From investing principles and algorithms to future developments”. In: *Machine Learning and Data Sciences for Financial Markets: A Guide to Contemporary Practices*. Cambridge University Press, 2023, pp. 60–85.

[15] Patrick Schueffel. “Taming the beast: A scientific definition of fintech”. In: *Journal of Innovation Management* 4.4 (2016), pp. 32–54.

[16] BPT Team. *What are the top operating costs for robo-advisors?* Nov. 2024. URL: https://businessplan-templates.com/blogs/running-costs/robo-advisor#:~:text=Technology%20and%20Software%20Licensing%3A%20One,from%20%24100%2C000%20to%20%24500%2C000%20annually..

[17] Matthias W Uhl and Philippe Rohner. “Robo-advisors versus traditional investment advisors: An unequal game”. In: *The Journal of Wealth Management* 21.1 (2018), p. 44.

[18] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[19] Anna Warchlewska and Krzysztof Waliszewski. “Who uses robo-advisors? The Polish case”. In: (2020).

[20] Lixuan Zhang, Iryna Pentina, and Yuhong Fan. “Who do you choose? Comparing perceptions of human vs robo-advisor in the context of financial services”. In: *Journal of Services Marketing* 35.5 (2021), pp. 634–646.

# Appendix A

# Figures and Tables

All detailed figures for the simulation results can be viewed at: `https://github.com/LinusTC/LinusTC.github.io/tree/main/backtests`

## A.1 Robo-advisor Simulation Results

### A.1.1 3 Month Investment Period

Table A.1: Robo-advisor Performance for 3 month investment period

| Investment Period | Robo-advisor Portfolio | Base Portfolio | Nasdaq |
|---|---|---|---|
| Mar 2014 to June 2014 | 2.0% | 7.3% | -2.1% |
| Nov 2016 to Feb 2017 | 0.2% | 12.0% | 10.9% |
| Feb 2019 to May 2019 | 18.4% | 7.1% | 11.0% |
| Jul 2023 to Jan 2024 | 21.7% | 5.5% | 11.2% |
| Jan 2025 to Apr 2025 | 8.6% | 0.8% | -10.2% |
| **Average Returns** | **10.18%** | **6.54%** | **4.16%** |

### A.1.2 6 Month Investment Period

Table A.2: Robo-advisor Performance for 6 month investment period

| Investment Period | Robo-advisor Portfolio | Base Portfolio | Nasdaq |
|---|---|---|---|
| Jan 2011 to July 2011 | 26.6% | 15.5% | 1.8% |
| Sep 2013 to Mar 2014 | 12.6% | 24.9% | 19.3% |
| Feb 2015 to Aug 2015 | 16.7% | 12.3% | 9.3% |
| June 2018 to Dec 2018 | -1.8% | -7.5% | -13.2% |
| Jul 2023 to Jan 2024 | 41.1% | 9.3% | 4.8% |
| **Average Returns** | **19.04%** | **10.90%** | **4.40%** |

### A.1.3 1 Year Investment Period

Table A.3: Robo-advisor Performance for 1 year investment period

| Investment Period | Robo-advisor Portfolio | Base Portfolio | Nasdaq |
|---|---|---|---|
| Jan 2012 to Jan 2013 | 18.6% | 9.4% | 15.2% |
| Apr 2016 to Apr 2017 | 27.5% | 8.9% | 20.1% |
| Sep 2019 to Sep 2020 | 112.0% | 20.2% | 48.1% |
| Sep 2021 to Sep 2022 | 2% | -16.4% | -21.7% |
| Sep 2023 to Sep 2024 | 100.1% | 37.7% | 24.9% |
| **Average Returns** | **52.04%** | **11.96%** | **17.32%** |

### A.1.4 2 Year Investment Period

Table A.4: Robo-advisor Performance for 2 year investment period

| Investment Period | Robo-advisor Portfolio | Base Portfolio | Nasdaq |
|---|---|---|---|
| Jan 2013 to Jan 2015 | 92.6% | 71.0% | 53.6% |
| Jun 2015 to Jun 2017 | 24.7% | -2.2% | 22.1% |
| Sep 2017 to Sep 2019 | 112.5% | 23.5% | 23.2% |
| Sep 2020 to Sep 2022 | 213.0% | 11.0% | -2.5% |
| Jan 2023 to Jan 2025 | 276.2% | 49.7% | 92.7% |
| **Average Returns** | **143.80%** | **30.60%** | **37.82%** |
| **Average Annual Returns** | **71.90%** | **15.30%** | **18.91%** |

### A.1.5 3 Year Investment Period

Table A.5: Robo-advisor Performance for 3 year investment period

| Investment Period | Robo-advisor Portfolio | Base Portfolio | Nasdaq |
|---|---|---|---|
| Jan 2013 to Jan 2016 | 201.6% | 89.2% | 64.1% |
| Jan 2018 to Jan 2021 | 372.4% | 84.1% | 62.4% |
| Jan 2022 to Jan 2025 | 136.4% | 9.8% | 26.5% |
| **Average Returns** | **236.8%** | **61.03%** | **51%** |
| **Average Annual Returns** | **78.93%** | **20.34%** | **17%** |

## A.2 Preferred Assets of Robo-advisor

Table A.6: Asset types of Unique Assets of Robo-advisor

| **Investment Type** | **Stocks** | **Bonds** | **ETFs** |
|---|---|---|---|
| 3 Months | 98.1% | 0.7% | 1.17% |
| 6 Months | 97.7% | 0.82% | 1.48% |
| 1 Year | 96.7% | 1.23% | 2.07% |
| 2 Years | 94.3% | 1.29% | 4.41% |
| 3 Years | 95.2% | 0.95% | 3.85% |
| **Average** | **96.4%** | **1.00%** | **2.60%** |