DSA4212 Portfolio Optimisation

Introduction

Objective

The primary objective of this portfolio optimisation project is to apply the mean-variance optimisation framework to construct an optimal portfolio of assets. I chose to maximise returns for a specified level of risk, as this approach aligns well with the practical goal of achieving higher returns while maintaining a controlled risk exposure. This method allows for a clear focus on enhancing potential returns without exceeding a predetermined risk tolerance, making it suitable for investors who prioritise growth but are mindful of risk limits. To achieve this, the project involves calculating the expected returns, volatilities, and correlations of a historical dataset, followed by optimisation to determine the portfolio weights that best achieve this risk-return balance.

Background

Mean-variance optimisation is central to modern portfolio theory. It seeks to build a portfolio that maximises expected return for a given risk level or minimises risk for a target return. By examining the trade-off between risk (measured by variance) and return, this framework helps investors identify efficient portfolios along the "efficient frontier."

In portfolio management, mean-variance optimisation supports informed asset allocation by incorporating expected returns, volatilities, and correlations, effectively aiding in risk diversification. However, it is also sensitive to input estimates and depends on historical data, which may not reliably predict future performance. This project explores these aspects by implementing the framework and testing solution stability under various scenarios.

Approach Overview

This project is structured into three stages:

1. **Data Collection**: Gather historical financial data for a diversified asset set.
2. **Optimisation**: Implement mean-variance optimisation using the CVXPY library to determine optimal portfolio weights. Visualise the results on an efficient frontier plot.
3. **Stability Analysis**: Test the robustness of the optimised portfolios and provide insights into how changes in market conditions may impact portfolio performance.

Data Collection

Data Source and Timeframe

The historical financial data for this project was obtained using the `yfinance` library in Python, which provides an efficient and reliable way to access stock price and other financial data. The time frame chosen spans from 1 January 2005 to 1 October 2024. This period incorporates data from various market cycles, including major events like the 2008 Global Financial Crisis and the 2020 COVID-19 pandemic, both of which had significant impacts on global markets. Including these periods ensures that the optimisation reflects a range of market conditions, providing a more robust basis for portfolio construction.

Asset Selection

The portfolio was constructed to include a diversified set of assets, focusing on U.S.-listed stocks. This selection aimed to cover a wide range of sectors, ensuring that the portfolio was not overly concentrated in any one industry. The chosen assets span the following sectors: Technology, Healthcare, Consumer Staples, Consumer Discretionary, Financials, Industrials, Utilities, Communication Services, Energy, Bonds, REITs, and Commodities. This sectoral diversity helps to manage risk by spreading exposure across different parts of the economy, which tend to perform differently in various market environments.

Data Processing

Before the data could be used for analysis, several preprocessing steps were taken:

1. **Data Cleaning**: Missing values or irregularities in price data, such as “NaN” entries or outliers, were handled. For missing values, forward or backward filling methods were used, as they can provide continuity without distorting the data.
2. **Adjusting for Dividends and Splits**: The data from `yfinance` already considers dividends and stock splits through adjusted closing prices. This adjustment ensures that the return calculations are accurate and reflect the true growth in value.

Calculations

Daily Returns and Covariance Matrix

- For each asset, its daily returns are calculated by taking the percentage change in adjusted closing prices, thereby capturing its day-to-day performance.
- Using the daily returns, I then calculated the covariance matrix. This matrix represents the degree to which each pair of assets moves in relation to one another, which is essential for understanding diversification benefits within the portfolio.

Expected Returns

- I calculated the expected daily return for each asset by taking the mean of its daily returns over the entire time frame.
- To annualize these expected returns, I multiplied each asset's expected daily return by 252, the approximate number of trading days in a year. This annualisation step provides a clearer view of each asset’s average yearly return, a key input for portfolio optimisation.

Portfolio Expected Returns

- The portfolio’s expected return was calculated by taking the weighted sum of the individual assets’ annual expected returns. This was achieved by multiplying each asset’s annualised expected return by its corresponding weight in the portfolio and then summing these values. This result reflects the overall expected return for the portfolio, based on the asset allocation.

Portfolio Expected Risk

- Portfolio risk was calculated using the portfolio variance formula:

$$\sigma_p^2 = \omega^T \Sigma \omega$$

  where ω is the vector of asset weights in the portfolio, and Σ is the annualised covariance matrix of returns.
- Taking the square root of the portfolio variance provided the portfolio's standard deviation, representing its overall risk. This measure of risk reflects the combined volatility of all assets in the portfolio, adjusted for correlations, and helps assess the expected range of fluctuations in portfolio returns.

Optimisation Framework

Formulation of the Mean-Variance Optimisation Problem

The mathematical formulation of the mean-variance optimisation problem is as follows:

Objective: Maximise the expected return of the portfolio, given a target level of risk. Mathematically, we can express this as:

$$\textit{maximise } \omega^T \mu$$

subject to:

$$\omega^T \Sigma \omega \leq \sigma_{target}^2 \qquad \text{[Constrain risk to target]}$$

$$\Sigma \omega_i = 1 \qquad \text{[Sum of weights = 1]}$$

$$w_i \geq 0 \quad \textit{for all } i \qquad \text{[No short selling]}$$

where:

- ω represents the vector of asset weights.
- μ is the vector of expected returns for each asset.
- Σ is the covariance of asset returns.
- σ_{target} is the target risk level, set to 8% in this project.

This formulation ensures that the optimisation process seeks the highest possible return for a portfolio with an overall risk (standard deviation) not exceeding 8%.

Implementation

Objective and Constraints in CVXPY:

- **Objective**: The objective function is set to maximise the portfolio's expected return, defined as the dot product of the asset weights ω and the expected returns μ.
- **Constraints**:

- ○ **Risk Constraint**: The portfolio variance (calculated as $\omega^T \Sigma \omega$) is constrained to be less than or equal to σ_{target} , where σ_{target} is 8% (or 0.08 in decimal form). This constraint ensures that the portfolio's risk does not exceed the specified target.
- ○ **Weight Sum Constraint**: The sum of all asset weights is constrained to equal 1. This ensures that the entire budget is invested across the selected assets without any additional borrowing.
- ○ **Non-negativity Constraint**: All weights w_i are constrained to be non-negative, meaning no short-selling is allowed. This is a practical constraint for risk-averse investors or funds that prefer long-only positions.

- **Parameters**:
  - ○ **Target Risk Level**: The target risk for this optimisation is set at 8%, guiding the model to find the optimal return achievable within this risk constraint.
  - ○ **Expected Returns and Covariance Matrix**: The expected returns vector μ and the covariance matrix Σ serve as the primary inputs for defining the risk-return characteristics of the portfolio.

Efficient Frontier

The efficient frontier was constructed by adjusting the risk constraint in each optimisation run to identify the maximum achievable return at that specific risk level.

By plotting the resulting optimised portfolios across different risk levels, I generated the efficient frontier—a graphical representation of the trade-off between risk and return. Each point on the efficient frontier represents an optimal portfolio, where any increase in return is matched by an increase in risk.

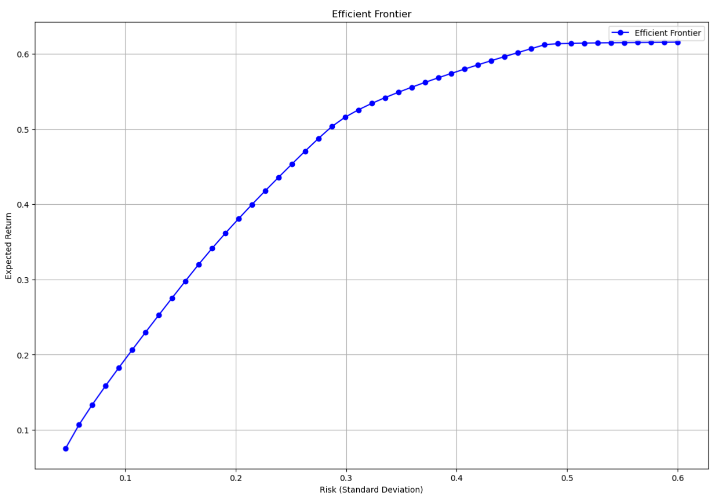

Figure 1: Efficient frontier plot at 8% risk target

The efficient frontier illustrates the set of optimal portfolios. Portfolios lying on the frontier are considered efficient because, for a given level of risk, they offer the maximum possible return. Conversely, for a desired level of return, they carry the minimum risk. Any portfolio below the frontier is suboptimal, as it would either have higher risk for a given return or a lower return for a given risk level.

In this project, the efficient frontier plot also highlights the specific point corresponding to the 8% risk target, showing the optimal return achievable under this risk constraint.

Stability of Solutions and Limitations of CVXPY

Stability Across Time Periods

I evaluated the stability of the mean-variance optimised portfolio by testing it across five distinct time windows, each representing different market conditions. The selected periods were:

- **Pre-2008 Financial Crisis** (2005-2007)
- **Global Financial Crisis** (2008-2009)
- **Post-Crisis Recovery** (2010-2015)
- **COVID-19 Pandemic** (2020-2021)
- **Recent Period** (2022-2024)

To analyse the impact of changing volatility and return estimates, we compared the portfolio weights derived for each period. This allowed us to observe how the model's allocation preferences shifted under different economic conditions, especially during periods of high volatility like the Global Financial Crisis and the COVID-19 Pandemic. By examining these shifts, we gained insights into the sensitivity of mean-variance optimisation to market fluctuations.

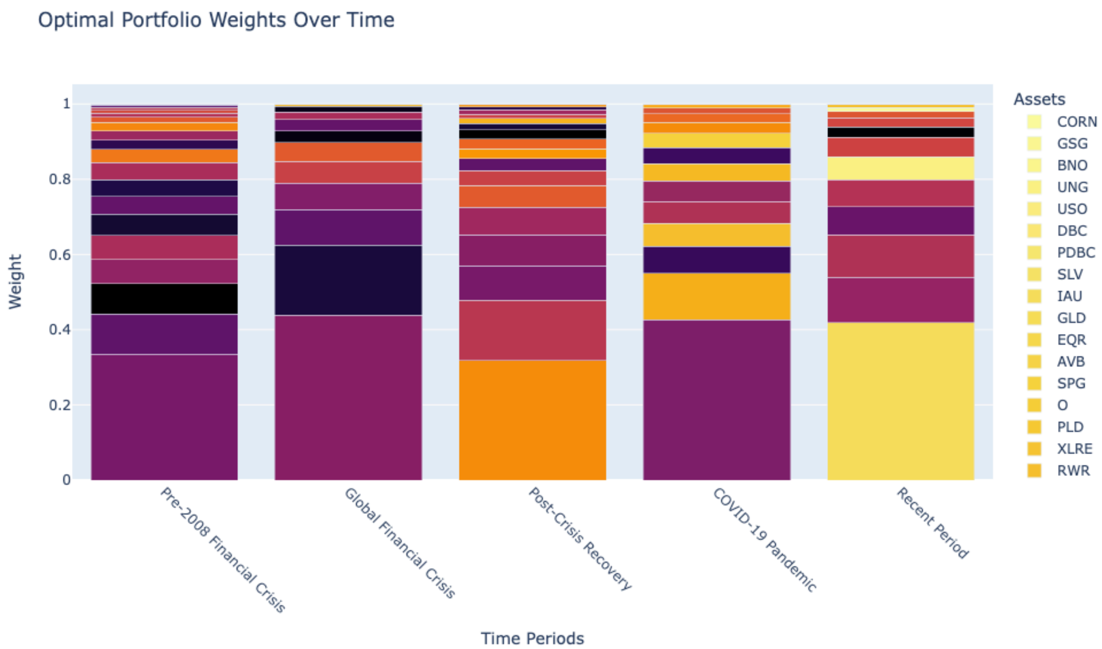

Figure 2: Optimal portfolio weights across various time periods

The portfolio weights displayed noticeable variability across periods, reflecting the model's responsiveness to changes in expected returns and covariances. For instance, during the Global Financial Crisis and COVID-19 Pandemic periods, there was a higher allocation to defensive assets, indicating a risk-averse approach in volatile conditions. In contrast, periods like the Post-Crisis Recovery showed a more balanced allocation, taking advantage of diversified growth opportunities.

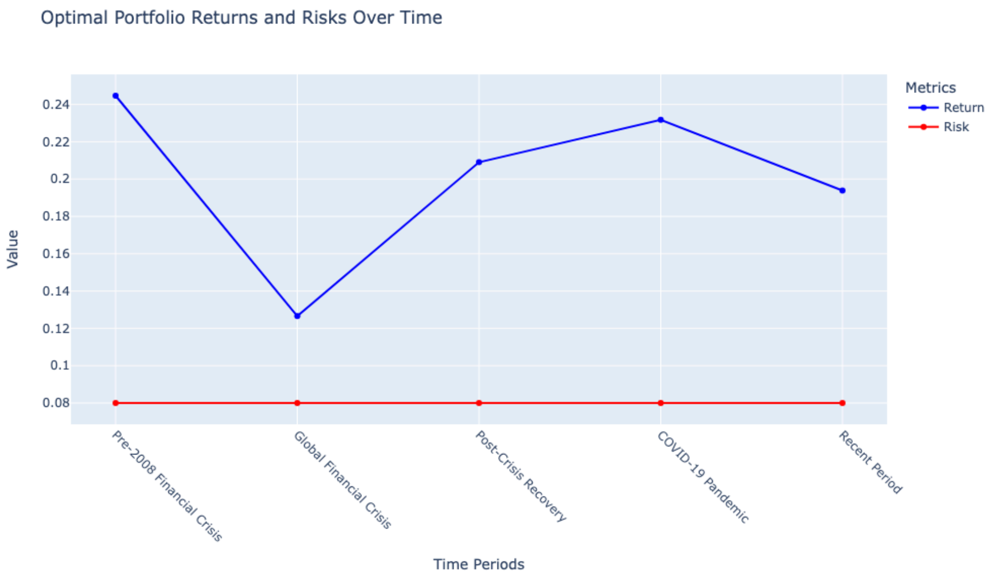

Figure 3: Annual returns and risk level across various time periods

While the target risk constraint of 8% was consistently met across all periods (as seen in the stable red line on Figure 3), expected returns fluctuated significantly. Lower expected returns were observed during high-volatility periods, such as the Global Financial Crisis, while periods with relatively stable market conditions, like the Pre-2008 Financial Crisis and Post-Crisis Recovery, showed higher expected returns. This variation demonstrates the model's limitation in achieving stable returns during turbulent markets.

These results suggest that while mean-variance optimisation can effectively control risk, it is highly sensitive to fluctuations in expected returns and covariances, requiring frequent rebalancing to maintain optimal allocations. This sensitivity implies that portfolios optimised with this approach may require active management, especially during volatile periods, to align with changing market dynamics.

Stability With Random Perturbations

To assess the stability of the mean-variance optimised portfolio, I tested it across various perturbation levels using Gaussian noise to introduce controlled randomness into the daily returns data. I applied Gaussian (normal) noise to the daily returns data by adding random values drawn from a normal distribution with a mean of zero and varying standard deviations (perturbation levels) which simulates unpredictable market fluctuations.

The perturbation levels were set to increase progressively, with each level representing a higher standard deviation in the Gaussian noise applied to the returns data. Lower levels simulate minor market noise, while higher levels introduce more substantial randomness, making the covariance matrix potentially less stable.

For each perturbation level, I recalculated the expected returns and covariance matrix from the perturbed returns data and re-ran the portfolio optimisation with the objective and constraints mentioned prior. Then I calculated the average and maximum change in weights between the new optimal weights and the optimal weights when there was no perturbation.

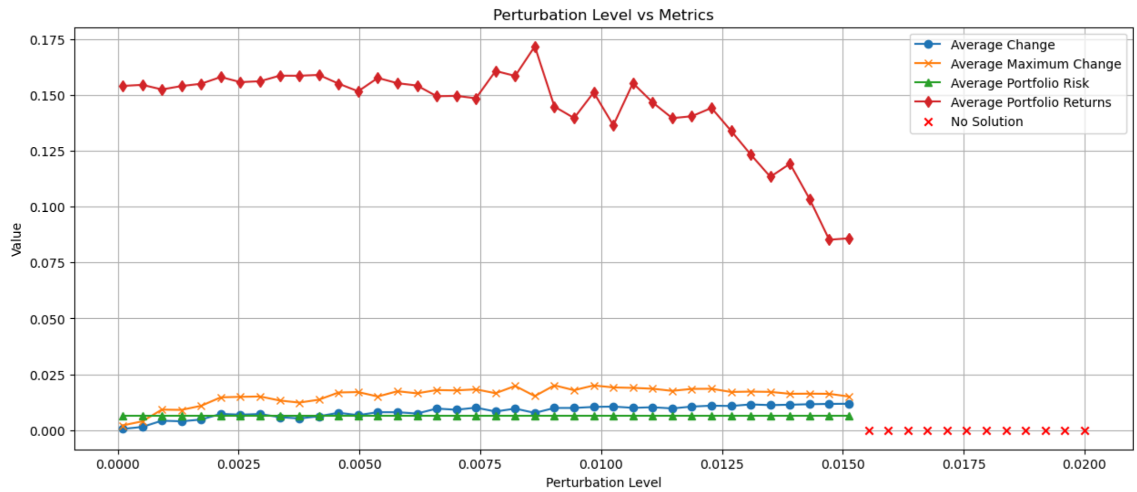

Figure 4: Impact of perturbation level on portfolio metrics

At low perturbation levels (below ~0.0125), the optimisation process performed reliably, with CVXPY consistently finding feasible solutions. In this range, the portfolio metrics—namely, **Average Change**, **Average Maximum Change**, **Average Portfolio Risk**, and **Average Portfolio Returns**—displayed minimal variation across different perturbation levels. This stability demonstrates the robustness of the mean-variance optimisation framework under minor market variations, which is desirable in real-world scenarios.

As the perturbation level increased beyond a certain threshold (~0.0150), the model started encountering infeasibility issues, marked by a series of red "X" symbols on the plot. These markers denote perturbation levels where CVXPY was unable to find a feasible solution.

At these higher perturbation levels, I observed the error message: `Forming a nonconvex expression quad_form(x, indefinite)`. This error indicates that the covariance matrix derived from the perturbed returns data was no longer positive semidefinite (PSD), a requirement for the convex optimisation framework in CVXPY. Once the covariance matrix became indefinite, the `quad_form` function could not process it to calculate portfolio risk, leading to an infeasible problem. This requirement limits CVXPY's usability under conditions of high market volatility or data noise, where non-PSD matrices are more likely.

There is a workaround to this by using the `cp.psd_wrap()` function which is designed to interpret a matrix as positive semidefinite (PSD) in the context of optimisation, but it does not make the matrix actually PSD. If the matrix is indefinite or has negative eigenvalues, `cp.psd_wrap()` will still lead to an error. This limitation means that CVXPY does not have a built-in feature to automatically "adjust" a matrix to be PSD, which is often necessary in portfolio optimisation when dealing with perturbed or noisy covariance matrices.

Discussion

CVXPY is ideal for:

- **Convex Problems**: CVXPY excels in stable, convex scenarios where the covariance matrix is PSD, such as periodic rebalancing in calm market conditions.
- **Low-Noise Data**: When data variability is minimal, CVXPY reliably finds feasible solutions.

In high-volatility or non-convex scenarios, alternative methods are more suitable:

- **High-Volatility Conditions**: For noisy or volatile data, consider robust optimisation techniques or frameworks that handle non-PSD matrices.
- **Complex Constraints**: For non-convex constraints (e.g., integer decisions, downside risk measures), alternative tools may be more appropriate.

Suggested Improvements and Alternatives

1. **Robust optimisation**: Using robust optimisation can enhance stability under uncertainty, making portfolios less sensitive to data noise.
2. **Data Smoothing**: Smoothed or aggregated data, such as moving averages, can improve matrix stability and reduce non-PSD issues.
3. **PSD Preprocessing**: Adjust the covariance matrix to be PSD before input, using eigenvalue correction or nearest PSD projection.

Conclusion

Overall, CVXPY provides a solid foundation for traditional portfolio optimisation tasks. However, adapting to real-world market conditions may require either robust optimisation techniques or alternative frameworks to ensure stability and flexibility. This study reinforces the need to carefully select optimisation tools based on specific market conditions and problem requirements, balancing the simplicity of convex frameworks like CVXPY with the adaptability needed for complex, volatile scenarios.

References

Zeng, S. (2020, September 10). *The efficient frontier in python*. Medium. https://medium.com/@zeng.simonl/the-efficient-frontier-in-python-a1bc9496a0a1

Welcome to CVXPY 1.5 - CVXPY 1.5 documentation. (n.d.). https://www.cvxpy.org/

Yfinance. PyPI. (n.d.). https://pypi.org/project/yfinance/