WIDS Project Report: Optimal Portfolio Allocation

This report documents my learning journey through the Optimal Portfolio Management project. Before starting this project, I had a general interest in finance, but I lacked a deep understanding . My main goal for the Optimal Portfolio Management project was to build something that could find the best combination of investments mathematically.

The project entailed creating Python scripts to get stock information, analyze it, and figure out the optimal combination of assets. Portfolio optimization is really important because it helps people make investment decisions. People do not have to pick stocks just because they feel like it. They can use the data and logic to make rational choices. This way, investors can try to maximize returns while avoiding excessive risk.

Going into this, I hoped to learn three specific things. First, I wanted to understand financial concepts like risk and volatility and how they actually work mathematically. Second, I needed to learn how to use Python libraries such as pandas and cvxpy to work with financial data. Third, I wanted to learn the logic of optimization well enough that I could explain it to someone else. This report covers what I learned, the technical work I implemented, and the insights I gained.

The first part of my project was about establishing the basics. So, I spent time reviewing resources and writing code to calculate fundamental metrics. I started by defining what a portfolio actually is. The resources explained that a portfolio is a collection of investments that represents an individual's total holding. The portfolio is made up of different asset classes, and balancing them is key.

I also learned about the different asset classes. Equities, or stocks, represent ownership in a company. When you buy a stock, you are buying a small piece of that business. Stocks offer high growth potential but are less safe. Fixed income assets, like bonds, are essentially loans to a government or corporation that pay back interest. These are safer but generally offer lower returns. Cash and equivalents are the safest place for money, but they do not grow much. Understanding these differences is important because the mathematical model relies on these assets behaving differently from one another.

One of the first technical concepts I had to grasp was measuring growth ,I learned that in financial modeling, we often use Log Returns instead of Simple Returns. Simple return is just the percentage difference from one day to the next. Log return is calculated as the natural logarithm of the price ratio between two days.I learned that log returns are additive. You can simply add them up to see performance over time, which makes the math much cleaner. In my code, I implemented this using the np.log function from the numpy library.

The financial definition of risk was a major shift in perspective for me. In finance, I learned that risk is actually defined as uncertainty. We use volatility to measure this risk. Volatility is a number that tells us how wildly a price swings up and down. Mathematically, it is the standard deviation of returns. Now I know that this high variance is exactly what makes it "risky," because the future price is uncertain. By calculating the standard deviation of the log returns, I could assign a concrete number to risk. I also learned about annualization. Since stocks trade about 252 days a year, I had to multiply my daily volatility by the square root of 252 to get a yearly risk number that makes sense for comparison.

Diversification was a concept I conceptually understood, but correlation made it mathematically clear. Correlation is a statistic that ranges from -1 to +1. It tells you how two assets move in relation to each other. If the correlation is +1, they move in perfectly together. If it is -1, they move in exact opposites. Now I understand that true diversification means buying assets that do not move together (low or negative correlation). If I own two tech stocks and they both crash at the same time, I haven't reduced my risk. I generated a correlation matrix in my project to visualize which stocks were connected.

The final foundational pieces were Expected Return and Covariance. Expected return is our best estimate of what an asset will earn in the future. To figure this out, I essentially calculated the average of past returns.Covariance was harder to grasp initially. It measures the directional relationship between two assets, similar to correlation, but it gives raw numbers that are necessary for calculation. The Covariance Matrix is the engine of portfolio optimization. It captures how every single stock interacts with every other stock in the portfolio.

After mastering the basics, I moved on to Modern Portfolio Theory (MPT). Proposed by Harry Markowitz, this theory is the standard for professional portfolio management. The core insight of MPT is that you shouldn't look at the risk and return of individual stocks in isolation. You must consider how they contribute to the risk and return of the portfolio as a whole.

The central concept of MPT is the Efficient Frontier. In one of my assignments, I generated thousands of random portfolios and plotted them on a graph. I put Risk (Volatility) on the X-axis and Expected Return on the Y-axis. The result was a cloud of points shaped roughly like a bullet or a sideways parabola.

The top edge of this cloud is called the Efficient Frontier. This line is significant because it represents the optimal set of portfolios, those that offer the highest possible return for a specific level of risk. It means that any portfolio located inside the cloud is suboptimal. There is always a better portfolio directly above it (more return for the same risk) or to the left (same return for less risk). The goal of optimization is effectively to find the weights that place our portfolio exactly on this line.

The MVP sits at the very leftmost tip of the Efficient Frontier curve. This point represents the mathematically lowest possible risk you can achieve with your set of assets.I found it interesting that the MVP is not simply a portfolio of 100% cash or the single safest stock. Because of correlation, the MVP is usually a mix of assets. Even risky assets can help lower the total portfolio risk if they move in opposite directions to other holdings.

I also learned about risk aversion. This concept acknowledges that not all investors are the same. Some are conservative and prioritize safety while others are aggressive and willing to accept wild price swings if it means potential for higher gains.

We model this preference using a parameter called Lambda (λ). I used Lambda in my code to find different optimal points along the Efficient Frontier. When I set Lambda to a high number, the model penalized risk heavily, resulting in a safe portfolio near the MVP. When I set Lambda low, the model chased returns, resulting in a riskier allocation. This taught me that there is no single "best" portfolio for everyone, the optimal choice depends entirely on the investor's specific risk tolerance.

Moving from theory to code required understanding optimization mathematics. I learned that every optimization problem consists of three core components: decision variables, an objective function, and constraints.

The **Decision Variables** are what we are trying to solve for. In this case, they are the portfolio weights (how much capital to allocate to each stock). If I have five stocks, I have five variables to determine.

The **Objective Function** is the goal we want to minimize or maximize. For the Minimum Variance Portfolio, the goal is to minimize risk. I used the matrix formula $w^T \Sigma w$, where w is the weight vector and Σ (Sigma) is the covariance matrix. This compact formula sums up every variance and covariance interaction in the portfolio to give a single risk number.

Constraints are the rules the solution must follow. The most fundamental one I used is the budget constraint. This rule states that the sum of all weights must equal 1 (100%). It ensures we invest exactly the money we have.Also every weight must be greater than or equal to zero. I did this to prevent the model from short-selling stocks. While shorting is a valid strategy, I wanted to keep my first project simple and focused on traditional buying strategies.

I encountered the concept of convexity, which is crucial for optimization. A convex optimization problem has a solution space shaped like a bowl. Because portfolio optimization is convex, I didn't have to worry about the computer getting stuck on a "fake" best answer. To solve for a specific risk preference, I set up an objective function to Maximize: $(\text{ExpectedReturn}) - (\lambda \times \text{Risk})$. This equation forces the computer to mathematically balance the desire for profit against the penalty of risk.

I started with yfinance, a library that downloads stock prices from Yahoo Finance. Next, I used pandas for data manipulation. I used .dropna() to remove incomplete rows. I learned that calculating returns always creates a NaN (Not a Number) value in the first row because there is no previous day to compare it to, so that row must be removed to prevent errors.After cleaning, I used numpy for mathematical transformations. I converted raw prices into Log Returns using the np.logfunction. This prepared the data for statistical analysis.I then used pandas to calculate the mean returns and the covariance matrix. As mentioned earlier, raw daily numbers aren't very intuitive, so I annualized them. I multiplied the mean returns by 252 and the covariance matrix by 252. This step was vital to ensure the final outputs represented yearly expectations, which are much easier to interpret.

In an earlier assignment, I tried to find the best portfolio using nested loops. It was slow and inefficient. Switching to cvxpy was a helpful. It allows you to write the problem in code almost exactly as it appears in math notation. I defined a variable w for weights. I defined risk using cp.quad_form and return using matrix multiplication (@). I then created a cp.Problem object containing my objective and constraints. When I called .solve(), cvxpy handled the heavy calculus in the background and returned the optimal weights instantly.

I also gained a practical understanding of linear algebra. I calculated portfolio variance using the matrix notation w.T @ Sigma @ w. It multiplies the weights by the covariance matrix, and then by the weights again. This single line of code efficiently sums up all the volatility and correlation interactions in the portfolio.

This project was a significant educational step for me. I used to think portfolio management was about reading news headlines and picking winners. Now I see it is really about statistics and probability. The fact that we can quantify "fear" with a parameter like Lambda is fascinating ,it bridges the gap between human psychology and hard mathematics.

This project has changed how I will approach problems in the future. Now, when I face a decision with multiple variables and constraints, I will look for an optimization approach rather than guessing. I have developed a toolkit of Python skills that I can apply to other data science problems. Moving forward, I would like to expand this project by using real-time data feeds or adding more complex constraints. Overall, this was a challenging but highly rewarding experience that gave me a solid foundation in both finance and programming.