Indian Institute Of Technology

Ropar

MARKOWITZ PORTFOLIO OPTIMIZATION

The Report submitted

by

AVIJIT BISWAS

Motivation:

The search for a well-balanced investment strategy becomes critical in the dynamic world of financial markets, where risks are inevitable and uncertainties are significant. In the midst of financial instability, Harry Markowitz's revolutionary Portfolio Optimization model shines like a ray of reason. Our project's driving force is the simultaneous goal of reducing risk and optimizing returns—a precarious balance that characterizes investment success.

Because of the volatility of the financial markets, focusing only on profits might have dangerous consequences. On the other hand, taking risks too carefully can limit your potential rewards. Markowitz's model is brilliant because it can reconcile these two demands in a way that makes sense. Through the application of statistical analysis and matrices, the model enables investors to create portfolios that balance risk and maximize profits.

The knowledge that portfolio optimization is not just a theoretical concept but also a practical requirement for how to diversify your investment in different assets. By applying the Markowitz model and conducting thorough data analysis, our goal is to shed light on the complexities involved in building portfolios that may have the highest potential for financial success. Our project aims to provide investors with the tools they need to navigate the complicated terrain of investing decisions, which entails balancing risk and reward. This approach promotes financial stability and prosperity.

Overview:

Harry Markowitz Portfolio Optimization Model is a core framework in current portfolio theory that tries to optimize a portfolio's anticipated return for a given level of risk or reduce risk for a given level of expected return. This concept is predicated on the idea that an investor may build an optimum portfolio by carefully choosing and allocating assets based on their past performance and their connections with one another.

We used financial data covering the previous 20 years, from November 1, 2003, to November 14, 2023, in our model implementation. The initial step was to import the financial data into a DataFrame using Python's sophisticated pandas package. Following that, we went through a comprehensive data cleansing

procedure to confirm the information's quality and dependability. This included dealing with missing values, dealing with outliers, and refining the dataset to build a solid foundation for analysis.

With the cleansed data in hand, we calculated essential KPIs for portfolio optimization. The calculation of asset returns is a critical component of this procedure. Returns are important because they measure an investment's gain or loss over a specified time period. The average contribution of each asset to the portfolio was then estimated using mean weights. Simultaneously, estimated returns were generated to estimate each asset's projected future performance.

The covariance matrix, which reflects the degree of variability and the link between the returns of different assets, is the foundation of portfolio optimization. The covariance matrix is useful for evaluating the diversification effects of mixing different assets in a portfolio. We mainly used computed the covariance matrix in our model to get insights into how various assets move in relation to each other and this is an important aspect in building a well-diversified portfolio.

With these key indicators in hand, the optimization process may begin. The goal of the optimization is to discover the best asset weighting combination that maximizes the portfolio's expected return while reducing its risk, as measured by volatility. This procedure necessitates advanced mathematical tools, which are frequently performed using optimization algorithms.

To summarize, we used pandas to import and clean financial data from the previous two decades, compute returns, mean weights, and predicted returns, and compute the covariance matrix. The model is a useful tool for investors to use to strategically deploy their assets, taking into consideration past performance and linkages between different investments to attain an optimal risk-return balance.

Covariance Matrix:

A covariance matrix between different assets (say there are n-assets) is a $n \times n$-square matrix. The diagonal entries represents variance of ij^{th}-entry represents covariance between i^{th} and j^{th} asset. Consider A to be a matrix of order $m \times n$. Suppose i^{th} column of A can be viewed as vector, say x_i, of $\mathbb{R}^m$. Then co-variance matrix of A is given as follows:

$$C(A) = \frac{1}{n-1} B^T B$$

Where B^T is a $m \times n$ matrix whose i^{th} column is $x_i - \mu$, where $\mu = \frac{\sum_{i=1}^{n} x_i}{n}$.

A covariance matrix is a crucial component in the Harry Markowitz Financial Portfolio Optimization model, serving as the important tool for analysing the relationships between different assets within a portfolio. In the context of portfolio optimization, the covariance matrix captures the statistical measure of the degree to which the returns of two assets move in relation to each other. It is a square matrix that quantifies the strength and direction of the linear relationship between pairs of assets.

The statistical purpose of the covariance matrix lies in providing insights into the portfolio returns of the assets. A positive covariance indicates that the returns of two assets tend to move in the same direction, while a negative covariance suggests that they move in opposite directions. A covariance of zero suggests that there is no linear relationship. Understanding these relationships is pivotal in the context of portfolio optimization because it helps investors see the diversification benefits of combining different assets.

In the Markowitz model, the covariance matrix is instrumental in calculating portfolio volatility and return. The model is not just about the individual returns of assets but also about the way assets interact with each other. The covariance matrix allows for the quantification of the total risk of a portfolio, taking into account both the individual volatilities of assets and their pairwise relationships. By using this matrix into the optimization process, investors can construct portfolios that creates an optimal balance between volatility and return.

To calculate the expected return of a portfolio, the covariance matrix is used in conjunction with the weights assigned to each asset in the portfolio. The formula for portfolio return involves the weighted sum of the expected returns of individual assets, taking into account their pairwise covariances. This comprehensive approach to portfolio construction, grounded in statistical analysis using covariance matrices, empowers investors to make informed decisions that align with their risk tolerance and return objectives.

Markowitz Portfolio Theory:

Risk-Return

Examining historical performance to forecast future returns is a frequent approach, with expected return defined as the average of a probability distribution of probable returns. The computation of expected return, often known as the mean or average return, is a critical first step in Markowitz's portfolio selection process.

This entails calculating the historical average of a stock's return over a given time period. The expected return for portfolios is computed by taking the weighted average of individual expected returns. A significant disadvantage is the uncertainty surrounding the proper time range for collecting previous performance. Choosing between a five-year, ten-year, or longer time horizon increases difficulties due to market uncertainty and volatility.

The calculation of the anticipated return, which represents the average of a probability distribution of probable returns, is critical in Markowitz's portfolio selection model. This expected return, also known as the mean or average return, is calculated by taking the historical average of a security's returns over a particular time period. This first stage in the model lays the framework for portfolio optimization. The weighted average of expected individual returns for a portfolio of two or more securities is simple, with weights reflecting the proportion of each asset in the overall portfolio. The concept of expected return, which is based on historical performance, is a key statistic that guides investment decisions and risk-return calculations.

Portfolio Return Variance

The volatility or risk of a security's return is determined using two key measures: variance and standard deviation. Variance is a metric that measures the squared deviations of a stock's return from its expected return, expressing the average squared difference between actual returns and the average return. In the context of a portfolio, variance refers to the volatility of an item or collection of assets, with higher variance values suggesting greater volatility. When various assets are maintained in a portfolio, the variance is decreased since assets that decrease in value are generally countered by those that increase in value, reducing overall risk. Notably, the total variation of a portfolio is typically lower than a simple weighted average of individual asset variances, and thus as the number of assets increases, covariances influence the total variance more than individual variances. This emphasizes the need of knowing how assets move within a portfolio rather than focusing exclusively on the variations in the value of each individual asset.

Standard Deviation

Besides from variance, another widely used indicator of volatility or risk is a security's standard deviation. Markowitz's portfolio selection model assumes that investors make investment selections based on both returns and risk spread. For investors, the perceived risk in purchasing a securities is the fear of receiving lower-than-expected returns, essentially a divergence from the expected average

return. As a result, each security has its unique standard deviation from the average. A higher standard deviation signifies a higher level of risk and thus a bigger potential return. The square root of the variance is the standard deviation. Calculating the standard deviation of predicted returns is a statistical method that takes into account a variety of parameters to determine the volatility of the return. This metric is essential understanding and managing the uncertainty associated with investment returns in Markowitz's approach.

Covariance of Return

Variance and standard deviation are good measures of stock variability, but when it comes to evaluating the relationship between returns on different stocks, covariance and correlation become critical metrics. Covariance is a statistical measure that reveals the interdependence of two securities' returns. The covariance is positive if the returns are positively correlated; negative if the returns are negatively correlated; and zero if there is no observable relationship. This idea is important in Markowitz's portfolio selection model since he highlights the need of avoiding securities having significant covariances among themselves. The correlation, which will be explained further below, gives a normalized measure of the degree and direction of this association between two random variables.

Diversification

Diversification, a core premise of Markowitz's portfolio selection theory and Modern Portfolio Theory (MPT), entails distributing assets across different financial instruments, industries, and categories in order to reduce portfolio risk. It is exemplified by the phrase "don't put all your eggs in one basket," which emphasizes the risk reduction gained by diversifying investments. Diversification includes asset classes other than stocks, such as bonds, real estate, and commodities. The goal is to maximize returns while minimizing risk by investing in assets that react differently to the same events, taking advantage of the diversification effect. Diversification can lower unsystematic risk or diversifiable risk connected to certain companies, but it cannot eliminate all risk. Diversification has no effect on systematic risk, which stems from external variables influencing the majority of organizations.

Use of Matrices in MPT:

Expected Portfolio Return:

Here we are using vector w to represent weights of allocation of each asset and we are considering μ as vector of expected returns. To optimize we consider μ as

weights of mean returns. We will calculate portfolio returns using the following formula:

$$R = w^T \cdot \mu$$

Portfolio Variance:

Portfolio variance is how much deviation in the returns of a portfolio. Let us denote covariance matrix of different set of assets by Σ. Then Portfolio variance is calculated by formula given below:

$$\sigma^2 = w^T \cdot \Sigma \cdot w$$

Portfolio Volatility (Risk):

Portfolio volatility is nothing but square root of portfolio variance, portfolio volatility is portfolio standard deviation. We denote portfolio risk by

$$\sigma_p = \sqrt{w^T \cdot \Sigma \cdot w}$$

Now what Markowitz suggested is to minimize σ_p and maximize R. So we take into consideration of maximization of the function

$$w^T \cdot \mu - \lambda\sqrt{w^T \cdot \Sigma \cdot w}$$

Here λ is risk aversion factor, which will create a balance between portfolio return and portfolio risk. While maximizing this function we actually want to find optimal weights w. So, we will be finding the weights w such that the value of α is maximized. The optimization often depends upon risk aversion factor and initial weights we are considering. Also, it depends upon outliers in the given data and assets value we have chosen.

Sharpe Ratio:

The Sharpe Ratio is a commonly employed metric in financial circles to gauge how well an investment or collection of investments is doing. It's a creation of William Sharpe and plays an integral role in Markowitz's Portfolio Optimization model. What purpose it serves? It is used to help investors figure out the adjusted for risks return of an investment and weigh up various investment chances. Sharpe ratio is calculated using the following formula:

$$sharpe\ ratio = \frac{\mu^T w - r_f}{-\sqrt{w^T \Sigma w}}$$

Here r_f is risk-free rate, rest all notations are as above.

Here, we used matrix multiplication and matrix inverse operations to calculate Sharpe ratio. Sharpe ratio is used in plotting Efficient Frontier.

Significance in the Markowitz Portfolio Optimization Model:

1) Risk-adjusted performance:

The Sharpe ratio provides a measure of the risk-adjusted performance of a portfolio. It allows investors to assess whether a fund's excess return is sufficient due to its level of risk.

2) Comparison of Portfolios:

Investors can compare different portfolios using the Sharpe Ratio. A higher Sharpe ratio indicates better risk-adjusted performance.

3) Portfolio selection:

In the Markowitz model, where portfolios are sorted in the order of efficiency, the Sharpe ratio is the main metric for selecting the best portfolio. A portfolio with the highest Sharpe ratio is considered to be the most efficient in terms of risk benefits have been adjusted.

4) Risk and benefit trade-offs:

The Sharpe ratio clearly captures the trade-off between risk and return. High-return portfolios are preferred, but excess returns must be weighed against increased risk.

Code:

Importing the libraries

```
import pandas as pd
import numpy as np
import pandas_datareader as pdr
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.optimize as opt
from scipy.optimize import minimize
from tabulate import tabulate

import warnings
warnings.filterwarnings("ignore")
```

Loading the datasets

```
reliance_data = pd.read_csv("reliance.CSV")
adani_data = pd.read_csv("adani.CSV")
infosys_data = pd.read_csv("infosys.CSV")
tcs_data = pd.read_csv("tcs.CSV")
airtel_data = pd.read_csv("airtel.CSV")
icici_data = pd.read_csv("icici.CSV")

Closing_Price = pd.concat([reliance_data[["Date", "Close"]], adani_data[["Close"]], infosys_data[["Close"]],
tcs_data[["Close"]], airtel_data[["Close"]], icici_data[["Close"]]], axis=1)
Closing_Price.columns = ["Date", "Reliance", "Adani", "Infosys", "TCS", "Airtel", "ICICI"]
Stocks = ["Date", "Reliance", "Adani", "Infosys", "TCS", "Airtel", "ICICI"]
Closing_Price
```

EDA & Data Cleaning

```
Closing_Price.info()
# Changing the date column to datetime format.
Closing_Price['Date'] = pd.to_datetime(Closing_Price['Date'])
print(Closing_Price.info())
Closing_Price.shape
# checking for null values
Closing_Price.isnull().sum()
# treating the null values
Closing_Price = Closing_Price.dropna()
stock_data = Closing_Price.copy()
stock_data = stock_data.reset_index(drop = True)
stock_data = stock_data.drop("Date", axis = 1)
stock_data
```

```
print(stock_data.isnull().sum())
print("*"*20)
print(stock_data.shape)
stock_data
```

Visualization

Plotting line graphs between the "closing Price" of each stocks and its corresponding "dates".

```
# Iterating over each column except the 'Date' column
for i in Closing_Price.columns[1:]:
    plt.figure(figsize=(10, 7))
    sns.lineplot(data=Closing_Price, x='Date', y=i, label="Line Graph")
    plt.title(f"Line Graph for {i}")
    plt.legend()

# Set x-axis ticks for 21 evenly spaced dates
num_labels_to_display = 21
xticks_positions = plt.xticks()[0]

# Ensuring that there are enough ticks to display
if len(xticks_positions) > num_labels_to_display:
    selected_xticks_positions = xticks_positions[::len(xticks_positions) // num_labels_to_display]
    selected_xticks_labels = pd.to_datetime(Closing_Price['Date']).dt.strftime("%Y-%m-%d").tolist()[::len(C
losing_Price['Date']) // num_labels_to_display]

    plt.xticks(selected_xticks_positions, selected_xticks_labels, rotation=45)

# Display the plots
plt.show()
```

Returns Matrix

Calculating daily returns for each stock using "pandas".

```
daily_returns = stock_data.pct_change()
daily_returns = daily_returns.dropna()
daily_returns
```

Mean Returns and Covariance Matrix

```
# Calculate mean return and covariance matrix
mean_returns = daily_returns.mean()
mean_returns_array = mean_returns.values
cov_matrix = daily_returns.cov()*252*20
cov_matrix_array = cov_matrix.values
cov_matrix_df = pd.DataFrame(cov_matrix_array)

# Convert mean_returns to a NumPy array
mean_weights = mean_returns.values

# Set the risk aversion parameter (lambda)
# risk_aversion = 0.8
cov_matrix_df.columns = ["Reliance", "Adani", "Infosys", "TCS", "Airtel", "ICICI"]
new_index = ["Reliance", "Adani", "Infosys", "TCS", "Airtel", "ICICI"]
cov_matrix_new_index = cov_matrix_df.set_index(pd.Index(new_index))

# Display the DataFrame with the new index
cov_matrix_new_index

# Printing mean weights

v = stock_data.columns
l = mean_weights
mydata = []
for i in range(len(v)):
  a = []
  x = v[i]
  y = l[i] * 100
  a = [x,y]
  mydata.append(a)
head = ["Company", "Mean Returns"]

# display table
print(tabulate(mydata, headers = head, tablefmt = "grid"))
```

Portfolio Optimization

Optimising without using Target returns.

```
#Portfolio Optimization code without using target return

risk_aversion=6.0
#Defining objective function for calculating portfolio return and portfolio volatility
def objective_function(weights, mean_weights, cov_matrix, risk_aversion):
  portfolio_return = np.dot(mean_weights, weights)
```

```
    #print(portfolio.return)
    portfolio_volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    #portfolio volatility=portfolio_volatility
    #print(portfolio_volatility)
    return -(portfolio_return - (risk_aversion*portfolio_volatility))

# Define constraints (weights sum to 1)
constraints = ({"type": "eq", "fun": lambda weights: np.sum(weights) - 1})

# Define bounds for weights (between 0 and 1)
bounds = tuple((0, 1) for _ in range(len(v)))

# Initialize weights equally for all variables
initial_weights = [3. /len(v)] * len(v)

# Running the optimization model
result = minimize(
    objective_function,
    initial_weights,
    args = (mean_weights, cov_matrix, risk_aversion),
    method = "SLSQP",
    bounds = bounds,
    constraints = constraints
)

# Extract the optimal weights from the result
optimal_weights = result.x
# Display the optimal weights
print("Optimal Weights:", optimal_weights)

v = stock_data.columns
l = optimal_weights
mydata = []
for i in range(len(v)):
    a = []
    x = v[i]
    y = l[i]*100
    a = [x,y]
    mydata.append(a)
head = ["Company", "Portfolio Percentage weights"]

# displaying Portfolio Percentage Weights through the table
print(tabulate(mydata, headers = head, tablefmt = "grid"))
```

Efficient Frontier

```
# Calculate the volatility of the optimal portfolio
portfolio_volatility = np.sqrt(np.dot(optimal_weights.T, np.dot(cov_matrix, optimal_weights)))

# Simulate portfolios for the efficient frontier
num_portfolios = 10000
results = np.zeros((3, num_portfolios))
risk_free_rate = 0.01  # replace with your risk-free rate
for i in range(num_portfolios):
    weights = np.random.random(len(v))
    weights /= np.sum(weights)

    # Calculate portfolio return using the Sharpe ratio formula
    portfolio_return = np.dot(mean_returns_array, weights)
    excess_return = portfolio_return - risk_free_rate
    portfolio_sharpe_ratio = excess_return / np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

    portfolio_volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))

    results[0, i] = portfolio_return
    results[1, i] = portfolio_volatility
    results[2, i] = portfolio_sharpe_ratio

# Plot the efficient frontier
plt.scatter(results[1, :], results[0, :], c=results[2, :], cmap="viridis", marker="o")
plt.title("Efficient Frontier with Color Gradient for Sharpe Ratio")
plt.xlabel("Volatility")
plt.ylabel("Return")
plt.colorbar(label="Sharpe Ratio")
plt.show()
```

Backtesting

```
#Code for backtesting without including target return

# Daily returns
returns = daily_returns

# Calculating mean returns and covariance matrix
mean_returns = returns.mean()
mean_weights = mean_returns.values
cov_matrix = returns.cov()*252*20

# Define risk aversion parameter
risk_aversion = 0.2
stocks_names = ["Reliance", "Adani", "Infosys", "TCS", "Airtel", "ICICI"]
```

```
stocks = stocks_names

# Define the objective function for optimization
def objective_function(weights, mean_weights, cov_matrix, risk_aversion):
    portfolio_return = np.dot(mean_weights, weights)
    portfolio_volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    return -(portfolio_return - risk_aversion*portfolio_volatility)

# Define constraints (weights sum to 1)
constraints = ({"type": "eq", "fun": lambda weights: np.sum(weights) - 1})
stock_names = Stocks

# Define bounds for weights (between 0 and 1)
bounds = tuple((0, 1) for _ in range(len(stocks_names)))

# Initializing weights equally
initial_weights = [1. / len(stocks_names)] * len(stocks_names)

# Run the optimization
result = minimize(
    objective_function,
    initial_weights,
    args = (mean_returns.values, cov_matrix.values, risk_aversion),
    method = "SLSQP",
    bounds = bounds,
    constraints = constraints
)

# Extracting the optimal weights from the result
optimal_weights = result.x

# Calculate the returns and volatility of the optimized portfolio
portfolio_return = np.dot(mean_returns, optimal_weights)
portfolio_volatility = np.sqrt(np.dot(optimal_weights.T, np.dot(cov_matrix, optimal_weights)))

# Displaying the results
print("Optimal Weights:", optimal_weights)
print("Optimal Portfolio Return:", portfolio_return*100)
print("Optimal Portfolio Volatility:", portfolio_volatility)
stocks_names = ["Reliance", "Adani", "Infosys", "TCS", "Airtel", "ICICI"]

v = stocks
l = optimal_weights
mydata = []
for i in range(len(stocks)):
    a = []
    x = v[i]
```

```
    y = l[i]*100
    a = [x,y]
    mydata.append(a)
head = ["Company", "Portfolio Percentage weights"]

# displaying Portfolio Percentage Weights through table
print(tabulate(mydata, headers = head, tablefmt = "grid"))


# Backtesting: Calculating the portfolio value over time
initial_investment = 100000  # Assuming initial investment amount
portfolio_value = np.zeros(len(daily_returns))
sum=0
for i in range(len(daily_returns)):
    daily_returns_today = daily_returns.iloc[i]  # Corrected this line
    portfolio_value[i] = initial_investment * (1 + np.dot(optimal_weights, daily_returns_today))
    #sum=portfolio_value[i]
#final_value_of_fund=sum
print(sum)
# Creating a DataFrame to store the portfolio value over time
portfolio_value_df = pd.DataFrame(portfolio_value, index = daily_returns.index, columns = ['Portfolio Val
ue'])

# Plot the portfolio value over time
plt.figure(figsize = (10, 6))
plt.plot(portfolio_value_df["Portfolio Value"], label = "Portfolio Value")
plt.title("Portfolio Value Over Time")
plt.xlabel("Date")
plt.ylabel("Portfolio Value")
plt.legend()
plt.show()
```

References:

1. Fabozzi, Frank J., Harry M. Markowitz, and Francis Gupta. "Portfolio selection." *Handbook of finance* 2 (2008).
2. Mangram, Myles E. "A simplified perspective of the Markowitz portfolio theory." *Global journal of business research* 7, no. 1 (2013): 59-70.
3. Z Bodie., A. Kane, and A. Marcus, Investments. Boston: McGraw-Hill/Irwin, 2009.