The Application of Operations Research on the Maximization of Return Rate and the Minimization of Risk Rate in Portfolios

Software: Python

Author: 110304006 楊謹豪
Teacher: 陸行

2024/1/8

Abstract

The maximization of return rate and the minimization of risk rate are the two of the most essential problem in financial fields. The challenging part of dealing with these is how to get the optimal solution while there are multiple constraints, such as the trade-off between risk and return and the correlation between stocks in portfolio, and the complex calculation of quadratic form problem. This report is going to illustrate the method called Sequential Least SQuares Programming (SLSQP) that can help solve the target problem of this report and even make further construction of optimization of the construction of portfolio. After the analysis and organization, this report has achieved the goal of generating a standard form for the calculation of every combination of stocks in Taiwan.

Contents

1 Problem description

1.1 The goal and introduction of the main problems

In the financial field, the construction of portfolio is the key to make a successful financial management project. There are multiple ways to achieve the goal of finding the optimal proportion of stocks. Without a doubt, methods in operations research field are always terrific option for it since optimization is the core of operations research.

Therefore, the goal of this report is to try to find out a method in operations research that can help people solve the three problem below:

1 Maximization of Return Rate (with the constraint of risk rate)

2 Minimization of Risk Rate (with the constraint of return rate)

3 Maximization of Sharpe Ratio (with the constraint of risk rate and return rate)

1.2 Formulation of problems in operations research

First of all, the objective function for problem 1 is

$$max\ Z = \sum_{i=1}^{n} r_i x_i,\ i = 1, 2, \ldots, n, \tag{1}$$

where r_i stands for the return rate of stock i and x_i stands for the proportion of stock i in the portfolio. Then, there are two constraints. First one is that the combination of all the proportions of stocks should be 1. Hence, the equation of this constraint is

$$\sum_{i=1}^{n} x_i = 1,\ i = 1, 2, \ldots, n. \tag{2}$$

The second one is the constraint that should be decided by users. It is about the tolerance of standard deviation of portfolio, which is also seemed as the risk rate of portfolio. The standard form of the constraint is

$$\sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2 + 2 \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} \sigma_i \sigma_j x_i x_j} \leq l_{std},\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n,\ i \neq j, \tag{3}$$

where s_i is the standard deviation of stock i and ρ_{ij} is the coefficient of correlation between stock i and stock j as well as l_{std} is the limitation the users can tolerate on the standard deviation. For a easier calculation, the matrix form of the standard deviation forms the inequality is

$$\sqrt{x^T \, Cov \, x} \leq l_{std}, \tag{4}$$

where x is the vector of proportions and Cov is the covariance matrix of the stocks. By the formula above, we can know that this problem is not a simple linear problem anymore. It is a quadratic programming problem, and thus we need to find a solution that can be applied on the quadratic problem.

Second, the formation of problem 2 is actually similar to the formation of problem 2. The objective function is

$$min\ Z = \sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2 + 2 \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} \sigma_i \sigma_j x_i x_j} \leq l_{std},\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n,\ i \neq j, \tag{5}$$

where the indications are the same as above. The first constraint is 2, which is the same as the first constraint of problem 1. The second constraint is the inequality

$$\sum_{i=1}^{n} r_i x_i \geq l_r \; i = 1, 2, \ldots, n, \tag{6}$$

where l_r is the he limitation the users hope to exceed on return rate.

Last but not least, we need to form the problem 3, which involves a financial term Sharpe Ratio. The equation of Sharpe Ratio is

$$\frac{R_p - R_f}{\sigma}, \tag{7}$$

, where R_p is the return rate of the portfolio, R_f stands for the return rate of risk-free asset such as T-bills and σ is the standard deviation of the portfolio. This ratio illustrates how much the excessive return rate the portfolio can earn where a unit of standard deviation seemed as the risk rate increases. Because the higher the Sharpe Ratio stands for the better performance the portfolio has, the objective function can be shown as

$$max\ Z = \frac{R_p - R_f}{\sigma}. \tag{8}$$

And for the better calculation, the R_p and $sigma$ will be replaced and shown as

$$max\ Z = \frac{\sum_{i=1}^{n} x_i - R_f}{\sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2 + 2\sum_{i=1}^{n}\sum_{j=1}^{n} \rho_{ij}\sigma_i\sigma_j x_i x_j}},\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n,\ i \neq j \tag{9}$$

. Then, there are three constraints of problem 3.

$$\sum_{i=1}^{n} x_i = 1,\ i = 1, 2, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n} r_i x_i \geq l_r\ i = 1, 2, \ldots, n \tag{11}$$

$$\sqrt{\sum_{i=1}^{n} \sigma_i^2 x_i^2 + 2\sum_{i=1}^{n}\sum_{j=1}^{n} \rho_{ij}\sigma_i\sigma_j x_i x_j} \leq l_{std},\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n,\ i \neq j \tag{12}$$

2 Software (Function) description

2.1 Description of Python and package used

The programming language used in the report for the solution is Python. Python is a famous programming language in worldwide. In Operations Research (OR), Python serves as a versatile tool for modeling and solving optimization problems. Libraries like SciPy (for numerical computations), NumPy (for handling arrays and matrices), and Pandas (for data manipulation and analysis) are particularly useful. Python's ability to integrate with optimization libraries, such as PuLP for linear programming and Pyomo for more complex optimization tasks, makes it a powerful tool in the OR toolkit. It allows for efficient formulation and solving of various OR problems, including resource allocation, scheduling, supply chain optimization, and decision analysis, offering a balance of ease of use and computational power. There are five packages used in the code for solution. They are requests, re, pandas, NumPy and SciPy. The first four packages are about the information gathering and processing. The last package is the key to calculation. The function called minimize provides a efficient and convenient way of solving optimization problems. The reason of using this function instead of PuLP or Pyomo mentioned above is that minimize function in SciPy contains a great advantage in solving quadratic programming problem which is the problem to be solved in this report.[1] The minimize function is constructed with few parts:

1. Objective function: The most essential part of the whole function. It needs user to provide a function object in Python. The function requires to have an argument that is the variables you want to use to calculate in the problem, which is mostly in a matrix form, and a return value that is referred to the final value to be optimized in the problem. Although the name of function is "minimize", users can still use it to

solve maximization problem by changing the return value into negative condition.

2. Bound: The bound is used to ensure that the variables will not exceed their own limitations.

3. Method: This part includes various methods for dealing with the problem. For example, it includes Nelder Mead, BFGS and SLSQP, the one used in this report.

4. Constraints: The constraints here is just like what we learn in operations research courses. However, the form of it is slightly different. First, it only has two type for users to choose. One is "eq" standing for "equality", another is "ineq" standing for inequality. These two types further result in the special form for inequality since there is no difference between $\geq$ and $\leq$. Therefore, the argument for inputting the function of inequality constraints has to be the form of

$$h(x) \geq 0 \tag{13}$$

, where $h(x)$ is the constraint.

After the introduction of basic functions, the method used eventually is also essential to be introduced here. The method finally used in this report is Sequential Least SQuares Programming (SLSQP). According to the introduction and the potential paper[2] behind the algorithm, we can know that it is a great algorithm for sequential quadratic programming problem and provides a fast as well as accurate result of approximation. The mathematical formulas are like below

- Objective Function and Constraints Given:

$$\begin{aligned} \text{Objective function:} \quad & f(\mathbf{x}) \\ \text{Equality constraints:} \quad & g(\mathbf{x}) = 0 \\ \text{Inequality constraints:} \quad & h(\mathbf{x}) \geq 0 \\ \text{Initial guess:} \quad & \mathbf{x}_0 \\ \text{Tolerance:} \quad & \text{tol} \end{aligned}$$

- Quadratic Approximation of Objective Function

$$Q(\mathbf{x}, \Delta\mathbf{x}) = f(\mathbf{x}) + J_f(\mathbf{x})^T \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T H_f(\mathbf{x}) \Delta\mathbf{x}$$

- Linearization of Constraints

$$\begin{aligned} \text{Equality Constraints:} \quad & g(\mathbf{x}) + J_g(\mathbf{x})^T \Delta\mathbf{x} = 0 \\ \text{Inequality Constraints:} \quad & h(\mathbf{x}) + J_h(\mathbf{x})^T \Delta\mathbf{x} \geq 0 \end{aligned}$$

- QP Subproblem

Minimize $Q(\mathbf{x}, \Delta\mathbf{x})$ subject to the linearized constraints.

- Line Search and Update

Find an appropriate step size along the direction $\Delta\mathbf{x}$ and update the current solution.

- Convergence is checked by assessing the magnitude of the gradient of the Lagrangian,

the change in variable values, and the change in the objective function value.

2.2 Advantage and limitation of software and functions

For the software, Python, the advantage of it is obvious. It is flexible and has a incredibly wide range of packages as the supplement of solving any kind of problems. Furthermore, as a famous and popular programming language, basically the newest technology, especially methods about artificial intelligence, will always be able to be used in Python, which will help the calculation and progress in the research simplify the upgrading of old packages or technology in operations research fields too.

However, there are still some mere disadvantage in Python. First, it is sometimes hard to know the theory behind some complex packages or functions like the one used in this report. Another weakness of Python is that mastering and using it are still harder than using Excel. Therefore, for a tyro or a person who has never learned Python and operations research, Excel will be a better choice for them.

As for the function, minimize in SciPy, the biggest advantage, as mentioned, is that it is easy to be used in Python. In addition, the speed of calculation and the extension it can achieve are also the apparent advantages of it.

Nevertheless, just like what was mentioned in the introduction of the disadvantage of Python, the lack of explanation of the formula behind the functions is something undoubtedly bothering because only know the original formula can make a better usage of the function.

3 The method

3.1 Mathematical expressions and Its Application

The structure of mathematical expression in the problem are shown in the steps above. Furthermore, for the sake of making a better analysis and apply this thought on real world problem, a surface and a set of codes were designed1. For manifesting the application of those functions and the procedure of solving this problem, there are five stocks, 期街口布蘭特正 2(00715L), 國泰 20 年美債 (00687B), 元大台灣 50(0050), 台積電 (2330), 元大台灣 50 反 1(00632R), will be used in my example. These are the great choices for constructing a portfolio since the low or negative correlations between each stock. Besides the risk-free asset used here is 0.016, which is the interest rate of fixed deposit in Taiwan.

Step 1: Choosing the stocks you wanna use. Figure 2

Step 2: Choose which kind of problem (three types discussed in 1.2) you wanna solve and limitation user can decide. Figure 3

Step 3: Get the required information of the stocks chose by Web Scraping. Figure 4

Step 4: Make a calculation according the objective functions and constraints of correspond kind of problem in 1.2. Figure 5 6 7

Step 5: Show the result of the calculation.

4 Solutions

4.1 The solutions of the corresponding parameters

The parameters and the corresponding result of each questions are shown below.

1 Example with the constraints of the minimum expected return rate 0.14 and maximum

risk rate (standard deviation) 0.18.

a The Result in the Maximization of Sharpe Ratio (Figure 9)

b The Result in the Maximization of Return (Figure 10)

c The Result in the Minimization of Risk Rate Rate (Figure 11)

2 Example with the constraints of the minimum expected return rate 0.27 and maximum risk rate (standard deviation) 0.30.

a The Result in the Maximization of Sharpe Ratio (Figure 12)

b The Result in the Maximization of Return (Figure 13)

c The Result in the Minimization of Risk Rate Rate (Figure 11)

5 The results

5.1 Comparison and analysis of the result

The result of this problem is basically impossible to compare with other ways, such as the result in textbook or the calculation on my own, since the uniqueness of quadratic programming this problem belongs to. Therefore, the analysis below will only be the comparison between different problems and different situation.

First of all, according to the Figure 9 and Figure 10, we can see that the solutions of the maximization of Sharpe Ratio and the maximization of return rate are almost the same. However, we can still see that there is a mere different between the proportion of stocks. This consequence is because that the limitation of return rate, 14%, is lower than the value of return rate with the optimal set of variables when maximizing the return rate with the

limitation of risk rate is only 18%. Hence, the movement of setting the limitation of risk rate only on 18% will not impact the optimal value of the maximization of Sharpe Ratio and the maximization of return rate. However, the tiny difference between these two still exist. Although knowing the exact reason of this phenomenon is impossible, we can still guess that it might be caused by the numerical precision and tolerances, which is hard to be prevented.

Second, according to the Figure 12, Figure 13, and Figure 14, we can find out that the discrepancy between this three different objective might be obvious. Though all of them achieve similar results, there is no one be able to get the optimal values of all three objective, which indicates that users can truly use the system to achieve the goal they want and the independent calculation of each question.

6 Remarks and conclusion

6.1 Conclusion of the whole report

1. According to all the solutions in the examples, we can conclude that maximization of Sharpe Ratio is indeed the problem that can generate the most balanced result while requiring the most limitation, implying that the usage of it needs a deeper background about finance. Simultaneously, the maximization of the return rate and the minimization of the risk rate shows how they can help the user achieve the goal of constructing the best allocation of a portfolio in the method from the operations research field.

2. The quadratic programming can efficiently solve the problem in this report. The minimize function in SciPy manifests great ability of dealing with the maximization of return rate and the minimization of risk rate problems. The implementation of Python

in operations research field is effective and recommended.

3. The construction of the maximization of return rate and the minimization of risk rate problems are not as simple as expectation. However, the constraints of these three problems are much less than expected, which means it is complex for the construction of a single constraint but simple for the whole structure.

6.2 What I learned and future prospect

1. In the process of making this report, I learned how to solve quadratic programming problem in codes. Even though the formulas behind are still difficult to understand to me now, I still appreciate for the chance of learning the quadratic programming on my own. If it is possible, I hope I can still have a chance to learn it deeper and truly make a thorough system about it.

2. In the future, I wanna add the function of being able to short selling the stocks inside the portfolio since it is also an important part to achieve the most perfect allocation of portfolios. Simultaneously, if the short selling function is added, the consideration of price and the credit the user can use for the short selling will also be constraints as well. In this way, this code will be capable of solving the maximization of return rate and the minimization of risk rate problems in a more realistic form.

References

[1] 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.

[2] Dieter Kraft. “A software package for sequential quadratic programming.” In: *Tech. Rep. DFVLR-FB 88-28* (1988).

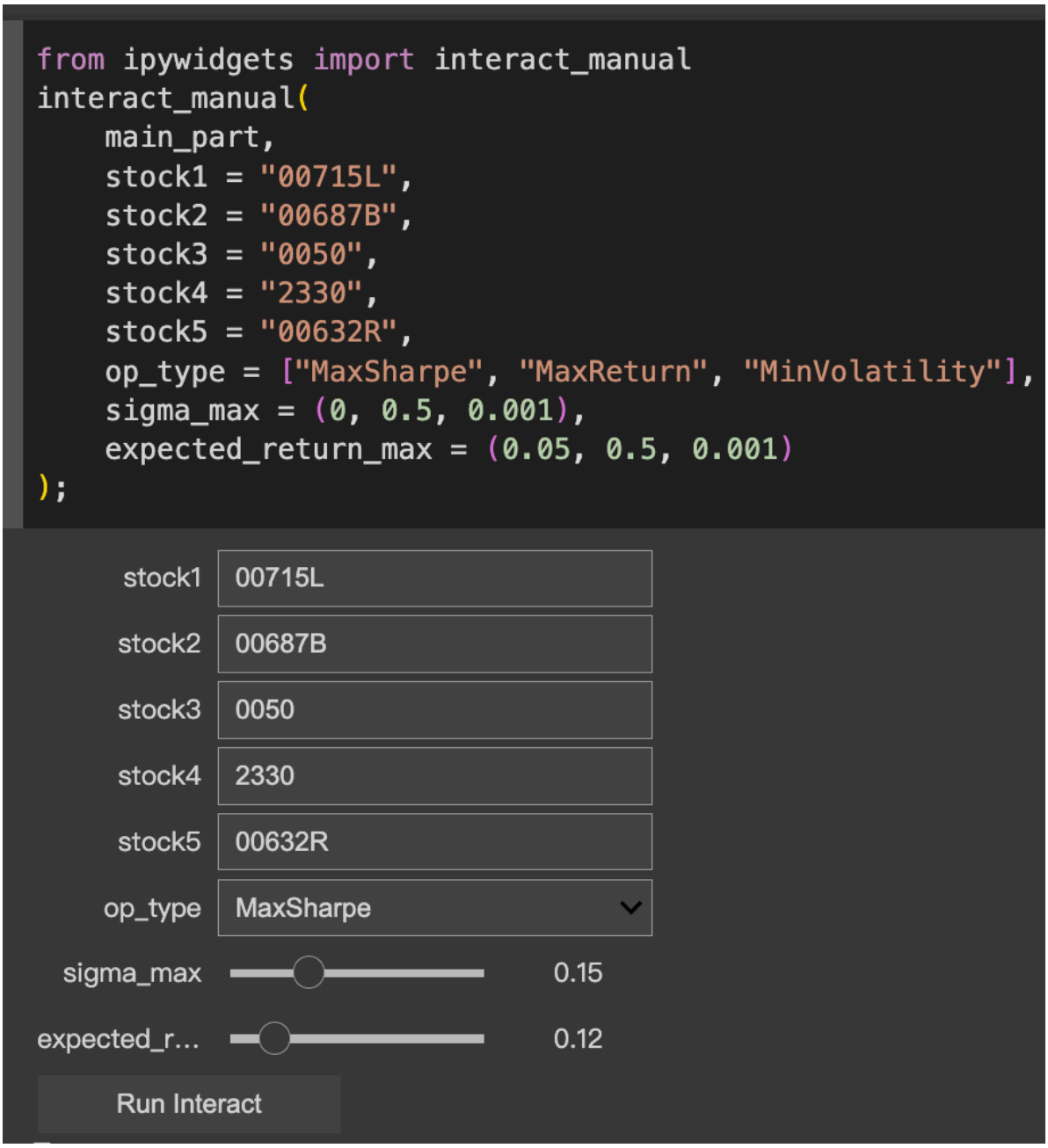

Figure 1: The above part is the main code for running the procedure of solving and the below part is the surface for inputting the information required.

```
True
Optimal proportions of  00715L : 0.094650938303873 87
Optimal proportions of  00687B : 0.1365632911363298
Optimal proportions of  0050 : 3.333921680392926e-17
Optimal proportions of  2330 : 0.5293779113014413
Optimal proportions of  00632R : 0.23940785925835512
Maximum Sharpe Ratio: 0.7569071804523781
Maximum expected return: 0.12953628347794394
Minimum volatility: 0.15000027270198057
```

Figure 2: The five stocks we determine to choose above.

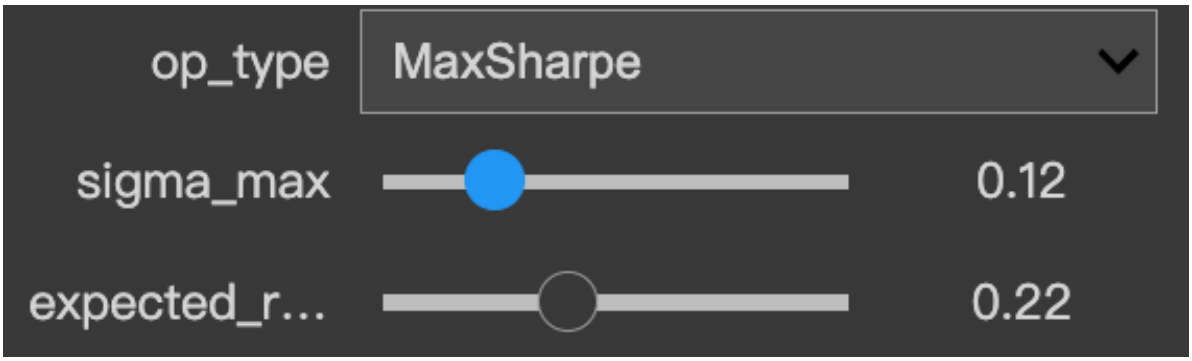

Figure 3: The choices for operation type and the limitation.

```python
def future_new(tp, code):
    global stock_all_data
    if code in stock_all_data.columns:
        return stock_all_data[code]
    print("正在抓取", code, "的資料。")
    res = requests.get('https://tw.quote.finance.yahoo.net/quote/q?type=ta&perd=' + str(tp) +
    m = re.findall('{"t":(\d+),"o":([\d.]+),"h":([\d.]+),"l":([\d.]+),"c":([\d.]+),"v":([\d.
    res_df = pd.DataFrame(m)
    res_df.columns = ['Date', 'open', 'high', 'low', 'close', "volumn"]
    # res_df['Date'] = pd.to_datetime(res_df['Date'], format = '%Y%m%d%H%M')
    res_df = res_df.set_index("Date")
    convert_dict = {"open":float, "high":float, "low":float, "close":float, "volumn":float}
    res_df = res_df.astype(convert_dict)
    stock_all_data[code] = res_df["close"]

    return res_df["close"]

def expected_return_and_std(code, stock, shift):
    global div_all_data
    global div_codes
    stock_shift = stock.shift(shift)
    stock = stock[shift:]
    stock_shift = stock_shift[shift:]
    stock_return = (stock - stock_shift) / stock_shift
    code1 = div_all_data[div_all_data["股票代號"] == str(code)]
    month = [transform_date_to_month_count(i) for i in code1['資料日期']]
    if str(code) in div_codes:
        mean_div = div_codes[str(code)]
    else:
        div_month_return = []
        for i in range(1, len(month)):
            div_month_return.append( (1 + (code1["權值+息值"].iloc[i] / code1['除權息參考價'].il
        if not div_month_return:
          return np.mean(stock_return), stock_return[::-1]
        mean_div = np.mean(div_month_return)
        div_codes[str(code)] = mean_div
    # print(code, " mean_div: ", mean_div)
    stock_return = stock_return + mean_div
    expected_return = np.mean(stock_return) + mean_div
    return expected_return, stock_return[::-1]
```

Figure 4: The code for stock data scraping.

```python
import numpy as np
from scipy.optimize import minimize

def max_return(expected_returns, cov_matrix, sigma_max, names):

    # Number of stocks
    n = len(expected_returns)  # Get the length of stocks

    # Define the objective function (negative expected return)
    def objective(x):
        return -np.dot(expected_returns, x)

    # Define the constraint for standard deviation
    def constraint(x):
        std_portfolio = np.sqrt(np.dot(x.T, np.dot(cov_matrix, x)))
        return sigma_max - std_portfolio

    # Constraints and bounds
    cons = [{'type': 'eq', 'fun': lambda x: np.sum(x) - 1},  # Sum of proportions equals 1
            {'type': 'ineq', 'fun': constraint}]              # Standard deviation constraint

    bounds = [(0, 1)] * n  # Proportions between 0 and 1 for each stock

    # Initial guess
    x0 = np.array([1/n] * n)

    # Solve
    result = minimize(objective, x0, method='SLSQP', bounds=bounds, constraints=cons)
    print(result.success)
    # Results
    optimal_proportions = result.x
    max_return = -(result.fun)
    min_volatility = np.sqrt(np.dot(optimal_proportions.T, np.dot(cov_matrix, optimal_proportions)))

    for i, name in enumerate(names):
        print("Optimal proportions of ", name, ":", optimal_proportions[i])
    print(f"Maximum expected return: {max_return}")
    print(f"Minimum volatility: {min_volatility}")
```

Figure 5: The code of applying operations research on maximization on return rate.

```python
def min_volatility(expected_returns, cov_matrix, expected_return_max, names):
    # Number of stocks
    n = len(expected_returns)  # Get the length of stocks

    # Define the objective function (negative expected return)
    def objective(x):
        return np.sqrt(np.dot(x.T, np.dot(cov_matrix, x)))

    # Define the constraint for standard deviation
    def constraint(x):
        e = np.dot(expected_returns, x)
        return e - expected_return_max

    # Constraints and bounds
    cons = [{'type': 'eq', 'fun': lambda x: np.sum(x) - 1},  # Sum of proportions equals 1
            {'type': 'ineq', 'fun': constraint}]               # Standard deviation constraint

    bounds = [(0, 1)] * n  # Proportions between 0 and 1 for each stock

    # Initial guess
    x0 = np.array([1/n] * n)

    # Solve
    result = minimize(objective, x0, method='SLSQP', bounds=bounds, constraints=cons)

    # Results
    optimal_proportions = result.x
    max_return = -(result.fun)
    min_volatility = np.sqrt(np.dot(optimal_proportions.T, np.dot(cov_matrix, optimal_proportions)))

    for i, name in enumerate(names):
        print("Optimal proportions of ", name, ":", optimal_proportions[i])
    print(f"Maximum expected return: {max_return}")
    print(f"Minimum volatility: {min_volatility}")
```

Figure 6: The code of applying operations research on minimization on risk rate.

```python
def max_sharpe(expected_returns, cov_matrix, sigma_max, expected_return_max, names):
    risk_free_rate = 0.016    # Risk-free rate
    n = len(expected_returns) # Get the length of stocks

    # Define the negative Sharpe ratio (since we're minimizing)
    def negative_sharpe_ratio(x):
        Rp = np.dot(x, expected_returns)
        sigma_p = np.sqrt(np.dot(x.T, np.dot(cov_matrix, x)))
        return -(Rp - risk_free_rate) / sigma_p

    def constraint_std(x):
        std_portfolio = np.sqrt(np.dot(x.T, np.dot(cov_matrix, x)))
        return sigma_max - std_portfolio

    def constraint_e(x):
        e = np.dot(expected_returns, x)
        return e - expected_return_max

    cons = [] # Constraints and bounds
    cons.append({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
    cons.append({'type': 'ineq', 'fun': constraint_e})
    if sigma_max != 0:
        cons.append({'type': 'ineq', 'fun': constraint_std})
    bounds = tuple((0, 1) for _ in expected_returns)

    initial_guess = np.array([1 / n] * n) # Initial guess

    # Perform the optimization
    result = minimize(negative_sharpe_ratio, initial_guess, method='SLSQP', bounds=bounds, constraints=cons)
    print(result.success)
    # Results
    optimal_proportions = result.x
    max_sharpe_ratio = -result.fun
    max_return = np.dot(expected_returns, optimal_proportions)
    min_volatility = np.sqrt(np.dot(optimal_proportions.T, np.dot(cov_matrix, optimal_proportions)))

    for i, name in enumerate(names):
        print("Optimal proportions of ", name, ":", optimal_proportions[i])
    print("Maximum Sharpe Ratio:", max_sharpe_ratio)
    print(f"Maximum expected return: {max_return}")
    print(f"Minimum volatility: {min_volatility}")
```

Figure 7: The code of applying operations research on maximization on Sharpe Ratio.

```
True
Optimal proportions of  00715L : 0.039347390934903 27
Optimal proportions of  00687B : 8.776215473858975e-12
Optimal proportions of  0050 : 0.0
Optimal proportions of  2330 : 0.42964528355399456
Optimal proportions of  00632R : 0.5310073255077543
Maximum Sharpe Ratio: 2.014290008771451
Maximum expected return: 0.049999999991396774
Minimum volatility: 0.016879396632729138
```

Figure 8: The result of the calculation.

```
Success: True
Optimal proportions of  00715L : 0.10795623004172215
Optimal proportions of  00687B : 0.14448326720685947
Optimal proportions of  0050 : 0.0
Optimal proportions of  2330 : 0.5835224565294442
Optimal proportions of  00632R : 0.16403804622197413
Maximum Sharpe Ratio: 0.79424
Maximum expected return: 0.15896
Minimum volatility: 0.18
```

Figure 9: The result of maximization of Sharpe Ratio with the constraint of minimum expected return rate 0.14 and maximum risk rate 0.18.

```
Success: True
Optimal proportions of  00715L : 0.107952096593410 18
Optimal proportions of  00687B : 0.1445225002442207
Optimal proportions of  0050 : 8.0709979071423 32e-15
Optimal proportions of  2330 : 0.58351467457136 99
Optimal proportions of  00632R : 0.16401072859099106
Maximum Sharpe Ratio: 0.79424
Maximum expected return: 0.15896
Minimum volatility: 0.18
```

Figure 10: The result of maximization of return rate with the constraint of maximum risk rate 0.18.

```
Success: True
Optimal proportions of  00715L : 0.099308406872733 65
Optimal proportions of  00687B : 0.141918635363358 16
Optimal proportions of  0050 : 3.955169525227 12e-16
Optimal proportions of  2330 : 0.5479653168399472
Optimal proportions of  00632R : 0.2108076409239606
Maximum Sharpe Ratio: 0.77193
Maximum expected return: 0.14
Minimum volatility: 0.16064
```

Figure 11: The result of minimization of risk with the constraint of minimum expected return rate 0.14.

```
Success: True
Optimal proportions of  00715L : 0.14704716726267753
Optimal proportions of  00687B : 4.0115480381963664e-17
Optimal proportions of  0050 : 1.452830911130576e-17
Optimal proportions of  2330 : 0.8529528327373224
Optimal proportions of  00632R : 0.0
Maximum Sharpe Ratio: 0.86018
Maximum expected return: 0.27251
Minimum volatility: 0.2982
```

Figure 12: The result of maximization of Sharpe Ratio with the constraint of minimum expected return rate 0.27 and maximum risk rate 0.30.

```
Success: True
Optimal proportions of  00715L : 0.1304560871473859
Optimal proportions of  00687B : 5.249774681810593e-16
Optimal proportions of  0050 : 3.6177793616767873e-16
Optimal proportions of  2330 : 0.8695439128526128
Optimal proportions of  00632R : 3.932333516969144e-16
Maximum Sharpe Ratio: 0.85873
Maximum expected return: 0.27362
Minimum volatility: 0.3
```

Figure 13: The result of maximization of return rate with the constraint of maximum risk rate 0.30.

```
Success: True
Optimal proportions of  00715L : 0.148146123662478 17
Optimal proportions of  00687B : 0.007844205653673282
Optimal proportions of  0050 : 0.0
Optimal proportions of  2330 : 0.8440096706838487
Optimal proportions of  00632R : 0.0
Maximum Sharpe Ratio: 0.85956
Maximum expected return: 0.27
Minimum volatility: 0.2955
```

Figure 14: The result of minimization of risk with the constraint of minimum expected return rate 0.27.