Optimization - Project 2

CREATING AN INDEX FUND IMITATING NASDAQ-100

A Data-Driven Approach Using Integer Programming

Group - 11

Meghavi Singhaniya, Abhijit Anil, Sai Lakshmi, Ali Sakhi Khan

Table of Contents

1.Introduction **3**
2.Data Preparation **5**
3.Methodology **7**
a) Stock Selection **7**
b) Weights Calculation **8**
4. Selecting top 5 stocks **11**
5. Optimal Number of Stocks **14**
6. Recommendations **18**

1.Introduction

The objective of this project is to design and implement an Index fund tracking the NASDAQ-100 index with a reduced number of stocks, m, selected from a larger universe of n stocks. This involves a multi-step process, including the formulation of an integer program for stock selection, solving a linear program for portfolio composition, and evaluating the fund's performance in comparison to the NASDAQ-100 index. The project aims to explore the concept of constructing a more manageable and cost-effective index fund, considering stock similarity based on a provided similarity matrix, ρij.

What is an Index Fund?

An index mutual fund or ETF (exchange-traded fund) tracks the performance of a specific market benchmark—or “index”, like the “popular” S&P 500 Index—as closely as possible. It is referred to as a “passive” investment strategy. Instead of hand-selecting which stocks or bonds the fund will hold, the fund's manager buys all (or a representative sample) of the stocks or bonds in the index it tracks.

What advantages does Index Fund offer?

1. **Diversification:** One of the main advantages of index funds is diversification. By investing in multiple stocks, they spread risk and reduce the impact of the poor performance of individual stocks. For the NASDAQ-100 index fund, it will attempt to hold a subset of the stocks from the NASDAQ-100 with similar weightings and correlation.

2. **Passive Management**: Index funds are passively managed, which means there is minimal human intervention in stock selection and portfolio management.

3. **Low Costs:** Index funds are known for their low expense ratios and trading costs. This is because they involve fewer buy and sell transactions compared to actively managed funds.

4. **Investor-Friendly:** Provides accessible, cost-effective market exposure for long-term investment and retirement planning.

Steps to be followed:

1. **Formulate Integer Program (Stock Selection):** Develop an integer program that selects exactly m out of n stocks for the portfolio. The program should use the similarity matrix ρ to choose stocks based on a correlation matrix between returns for each of the stock.

2. **Solve Linear Program (Portfolio Composition):** Once the stocks are selected, solve a linear program to determine how many shares of each chosen stock to buy to construct the portfolio. The linear program considers factors such as stock prices and portfolio constraints.

3. **Evaluate Performance:** After constructing the index fund portfolio, evaluate its performance by comparing it with the NASDAQ-100 index, using data from the year 2020.

4. **Vary m for Sensitivity Analysis:** To assess the impact of the number of selected stocks (m) on the index fund's performance, repeat the process for different values of m and analyze the results.

5. **Using a MIP for weight calculation:** The objective is to replace stock selection with weight selection through a Mixed-Integer Programming (MIP) approach. This MIP optimizes over all n stock weights while enforcing binary variables (y_i) to control stock inclusion in the portfolio.

By completing these steps, the project aims to demonstrate the feasibility of constructing a costeffective and manageable index fund that closely tracks the NASDAQ-100 index while leveraging a similarity matrix to make informed stock selections.

2.Data Preparation

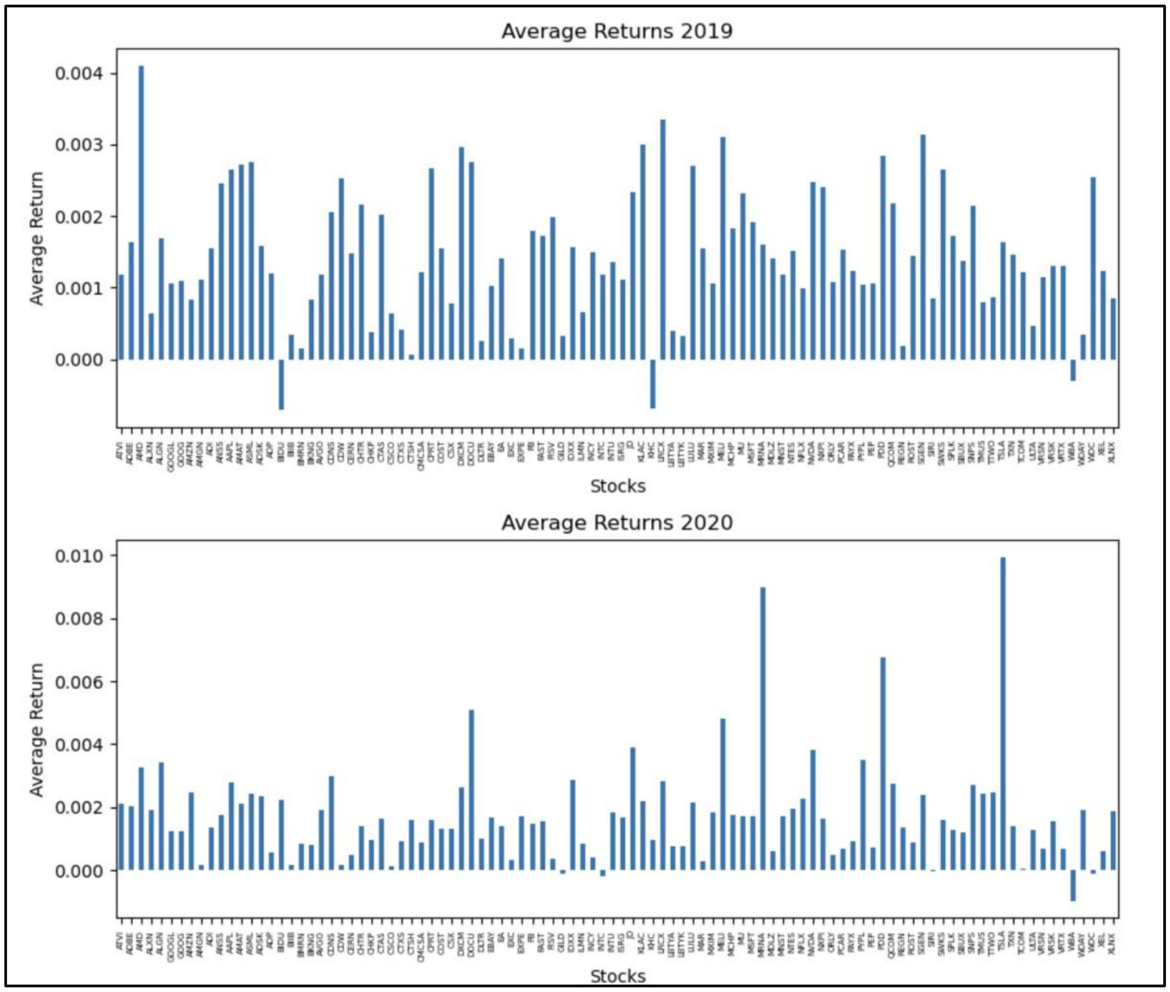


Figure 1: Average return of NASDAQ-100 stocks in the years 2019 and 2020

Let's delve into the performance of the NASDAQ-100 in the years 2019 and 2020. Figure 1 provides a visual representation of how the returns of most stocks within the index fluctuated. While some stocks witnessed a boost in their returns, over half of the stocks experienced a decrease.

Upon scrutinizing the average performance of the index for both years, it becomes apparent that the NASDAQ-100 encountered an average increase of 32.76% (see Figure 2). Notably, Cognizant Technology Solutions Corp (Ticker: CTSH) displayed the most significant average percentage increase at an astounding 2165.66%. On the other hand, Baidu Inc (Ticker: BIDU) recorded the most substantial average percentage decrease at -417.74%.

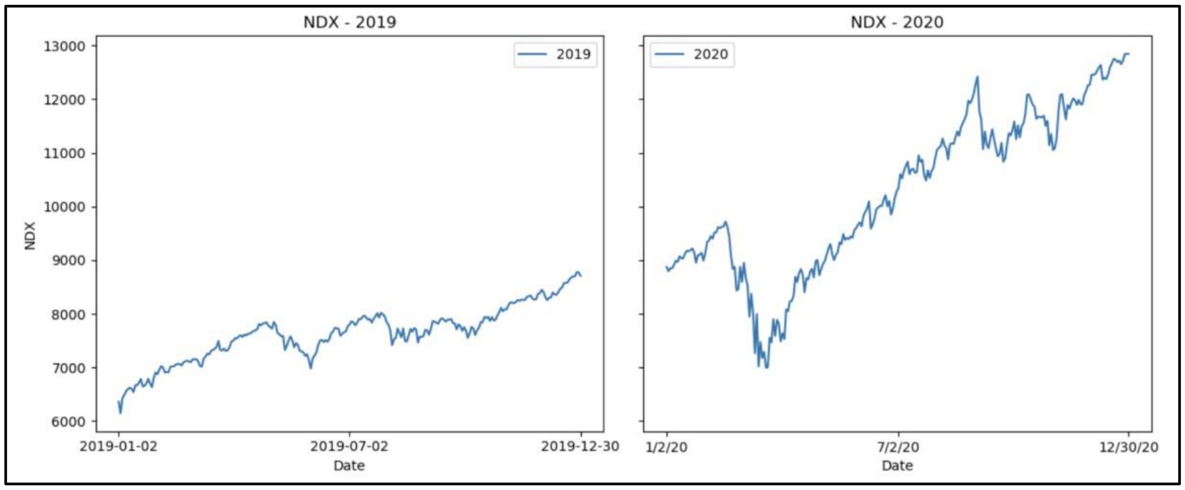


Figure 2: Overall trend of the NASDAQ-100 stocks in the years 2019 and 2020

Although the overall trend in 2020 suggests a decrease in average returns, the remarkable growth in certain stocks contributed to an overall increase of 20.75% in the NASDAQ-100's returns. This demonstrates the dynamic nature of the index, where exceptional performers can drive the index's returns even amidst challenging times. Refer to Figure 2

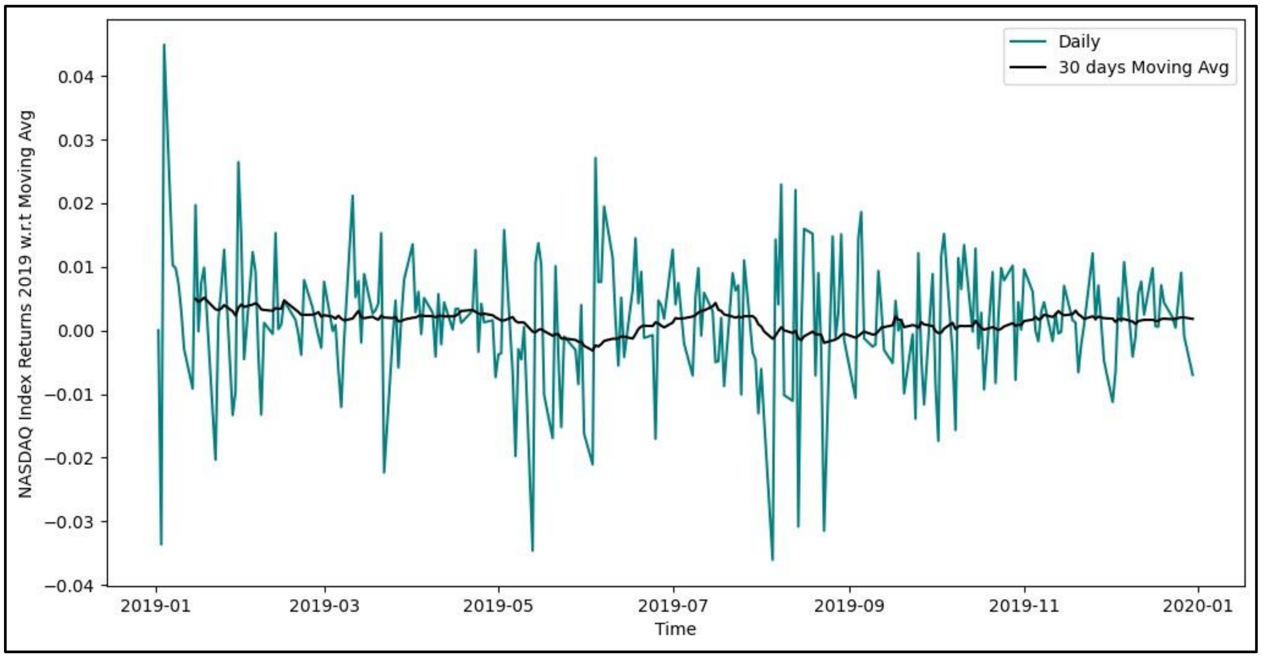


Figure 3: NASDAQ-100 returns vs the Moving average of returns in 2019

In addition, it's worth noting that the 30-day moving average for the NASDAQ-100 remained relatively constant throughout the observed period. This constant moving average suggests that despite the fluctuations in individual stock returns, the overall index maintained a level of consistency. Refer to Figure 3

3.Methodology

In order to determine which stocks should be picked in the fund, we have decided to take a two step approach. One, where we will find which of the stocks are the most similar to each other with the goal to maximize the similarity between the NASDAQ-100 stocks and their representatives in the fund. Two, where we determine the weight of the selected fund stocks such that it closely tracks the returns of the index. We will use the returns of the NASDAQ-100 from the year 2019 to select our stocks and determine their weights. We will then check to see how well our portfolio tracks the index of 2020.

a) Stock Selection

First and foremost, we need to find the daily returns of the index and find the correlation between the stocks. Next, we have to ensure that our selected stock is present in the index. And lastly, we want to identify which index stock best represents the fund stock. To establish this, we can implement an integer programming model with the following objective function and constraints:

$$max_{x,y} \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} x_{ij}$$

such that,

$$\sum_{j=1}^{n} y_j = m \text{ , where } m \text{ is the total number of stocks in the fund}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ , for } i = 1,2,\ldots,n$$

$$x_{ij} \leq y_j \text{ , for } i, j = 1,2,\ldots,n$$

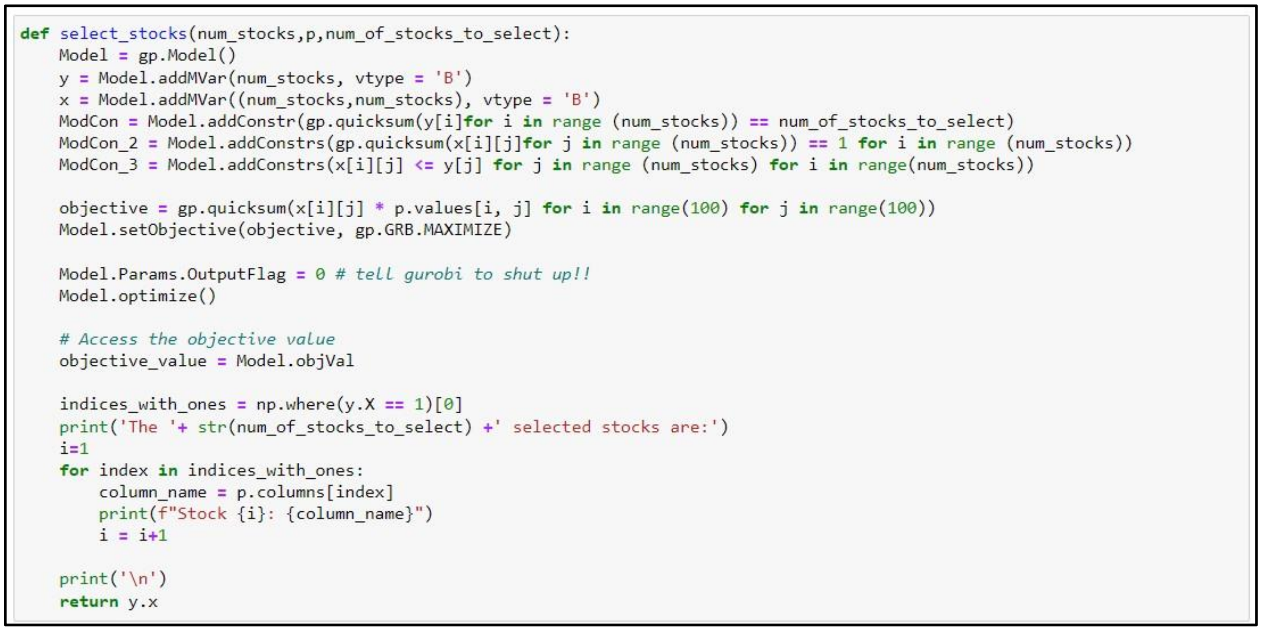
$$x_{ij} \leq y_j \in \{0,1\}$$

x_{ij} - *This variable is 1 if fund i is represented by fund j i.e if we pick a fund j and we have another fund that is highly correlated with fund j, it will be represented by this variable making it 1*

y_j - *This variable is 1 if that fund is selected*

Both these variables will be treated as binary variables

$max_{x,y} \sum^{n}{}_{i=1} \sum^{n}{}_{j=1} p_{ij}x_{ij}$ - *Moving on to the objective, we are trying to maximize the similarity between the n stocks that we have and their representative among the selected funds. The weight on the link between the selected fund* y_j *and the stock* x_{ij}*is the correlation pij which we are trying to maximize to obtain the best mapping between the stocks and the funds.*



```python
def select_stocks(num_stocks,p,num_of_stocks_to_select):
    Model = gp.Model()
    y = Model.addMVar(num_stocks, vtype = 'B')
    x = Model.addMVar((num_stocks,num_stocks), vtype = 'B')
    ModCon = Model.addConstr(gp.quicksum(y[i]for i in range (num_stocks)) == num_of_stocks_to_select)
    ModCon_2 = Model.addConstrs(gp.quicksum(x[i][j]for j in range (num_stocks)) == 1 for i in range (num_stocks))
    ModCon_3 = Model.addConstrs(x[i][j] <= y[j] for j in range (num_stocks) for i in range(num_stocks))

    objective = gp.quicksum(x[i][j] * p.values[i, j] for i in range(100) for j in range(100))
    Model.setObjective(objective, gp.GRB.MAXIMIZE)

    Model.Params.OutputFlag = 0 # tell gurobi to shut up!!
    Model.optimize()

    # Access the objective value
    objective_value = Model.objVal

    indices_with_ones = np.where(y.X == 1)[0]
    print('The '+ str(num_of_stocks_to_select) +' selected stocks are:')
    i=1
    for index in indices_with_ones:
        column_name = p.columns[index]
        print(f"Stock {i}: {column_name}")
        i = i+1

    print('\n')
    return y.x
```

Code Chunk 1: Function for selection of stocks

b) Weights Calculation

Once the stocks are selected, we have to ensure that the weight of each individual stock is determined in such a way that it reflects the returns of the index fund, in this case the NASDAQ100. This can be achieved by minimizing the absolute difference between the selected fund stock and the index's return during a certain time period. In order to solve this, we can opt for a linear programming model with the following objective function and constraints:

$$min_w \sum_{i=1}^{T} |q_t - \sum_{i=1}^{m} w_i r_{it}|$$

$$such\ that,$$

$$\sum_{i=1}^{m} w_i = 1$$

$$w_i \geq 0$$

$min_w \sum^{T}{}_{i=1} |q_t - \sum^{m}{}_{i=1} w_i r_{it}|$ - *Here we are trying to minimize the loss in returns by matching them as closely as possible with the index.*

r_{it} - *This is the return of stock i at time t*

w_i - *Weight of the stock i in the portfolio*

q_t - *Return of the Index at time t*

However, given that the objective function is non-linear, we will have to reformat it into a linear program. This can be achieved by rewriting $|q_t - \sum_{i=1}^{m} w_i r_{it}|$ as z_t. Now, our objective function and our constraints can be written as follows:

$$\min \sum_{t=1}^{T} z_t$$

such that,

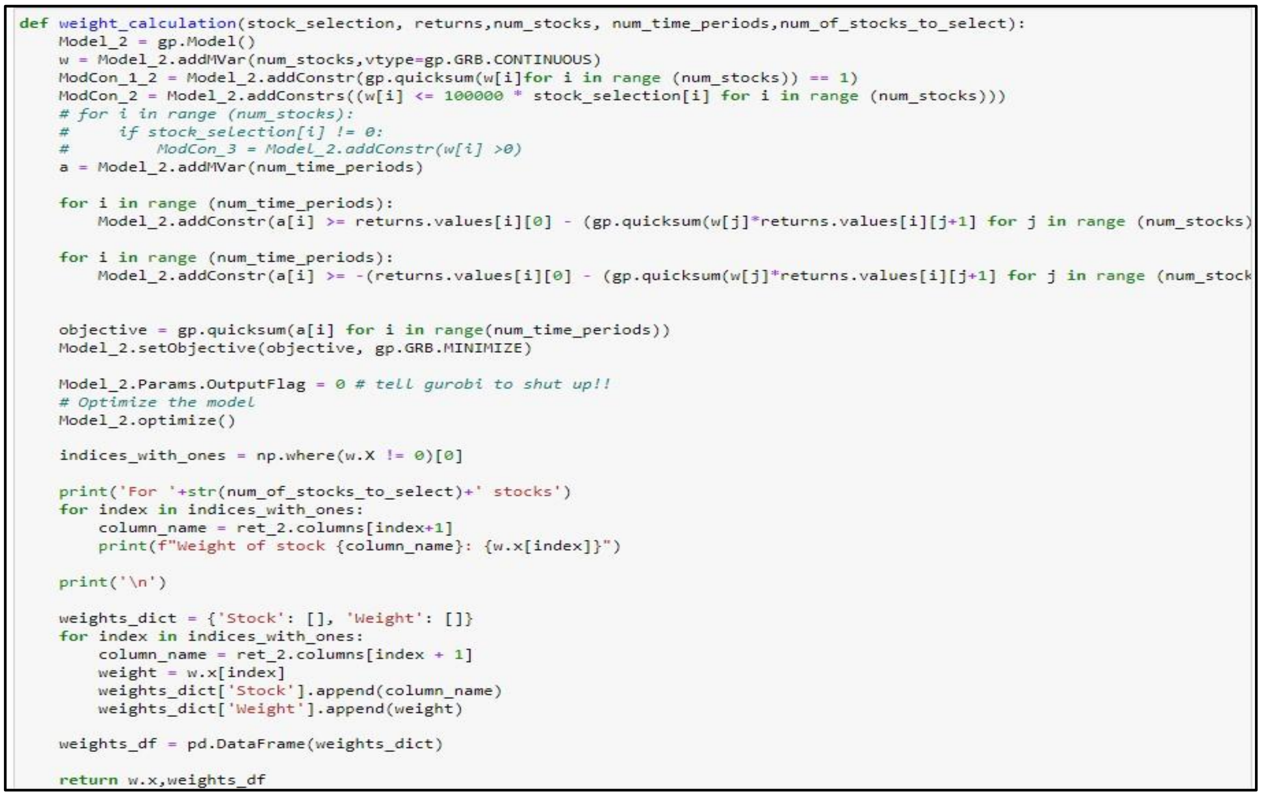
$$z_t \geq q_t - \sum_{i=1}^{m} w_i r_{it}\, , for\ all\ values\ of\ t$$

$$z_t \geq -\left(q_t - \sum_{i=1}^{m} w_i r_{it}\right), for\ all\ values\ of\ t$$

$$\sum_{i=1}^{m} w_i = 1$$

$$w_i \geq 0$$

In doing so, we can approach this as a linear program and easily construct the portfolio weight.



```python
def weight_calculation(stock_selection, returns,num_stocks, num_time_periods,num_of_stocks_to_select):
    Model_2 = gp.Model()
    w = Model_2.addMVar(num_stocks,vtype=gp.GRB.CONTINUOUS)
    ModCon_1_2 = Model_2.addConstr(gp.quicksum(w[i]for i in range (num_stocks)) == 1)
    ModCon_2 = Model_2.addConstrs((w[i] <= 100000 * stock_selection[i] for i in range (num_stocks)))
    # for i in range (num_stocks):
    #     if stock_selection[i] != 0:
    #         ModCon_3 = Model_2.addConstr(w[i] >0)
    a = Model_2.addMVar(num_time_periods)

    for i in range (num_time_periods):
        Model_2.addConstr(a[i] >= returns.values[i][0] - (gp.quicksum(w[j]*returns.values[i][j+1] for j in range (num_stocks)

    for i in range (num_time_periods):
        Model_2.addConstr(a[i] >= -(returns.values[i][0] - (gp.quicksum(w[j]*returns.values[i][j+1] for j in range (num_stock


    objective = gp.quicksum(a[i] for i in range(num_time_periods))
    Model_2.setObjective(objective, gp.GRB.MINIMIZE)

    Model_2.Params.OutputFlag = 0 # tell gurobi to shut up!!
    # Optimize the model
    Model_2.optimize()

    indices_with_ones = np.where(w.X != 0)[0]

    print('For '+str(num_of_stocks_to_select)+' stocks')
    for index in indices_with_ones:
        column_name = ret_2.columns[index+1]
        print(f"Weight of stock {column_name}: {w.x[index]}")

    print('\n')

    weights_dict = {'Stock': [], 'Weight': []}
    for index in indices_with_ones:
        column_name = ret_2.columns[index + 1]
        weight = w.x[index]
        weights_dict['Stock'].append(column_name)
        weights_dict['Weight'].append(weight)

    weights_df = pd.DataFrame(weights_dict)

    return w.x,weights_df
```

Code Chunk 2: Function for weight calculation

4. Selecting top 5 stocks

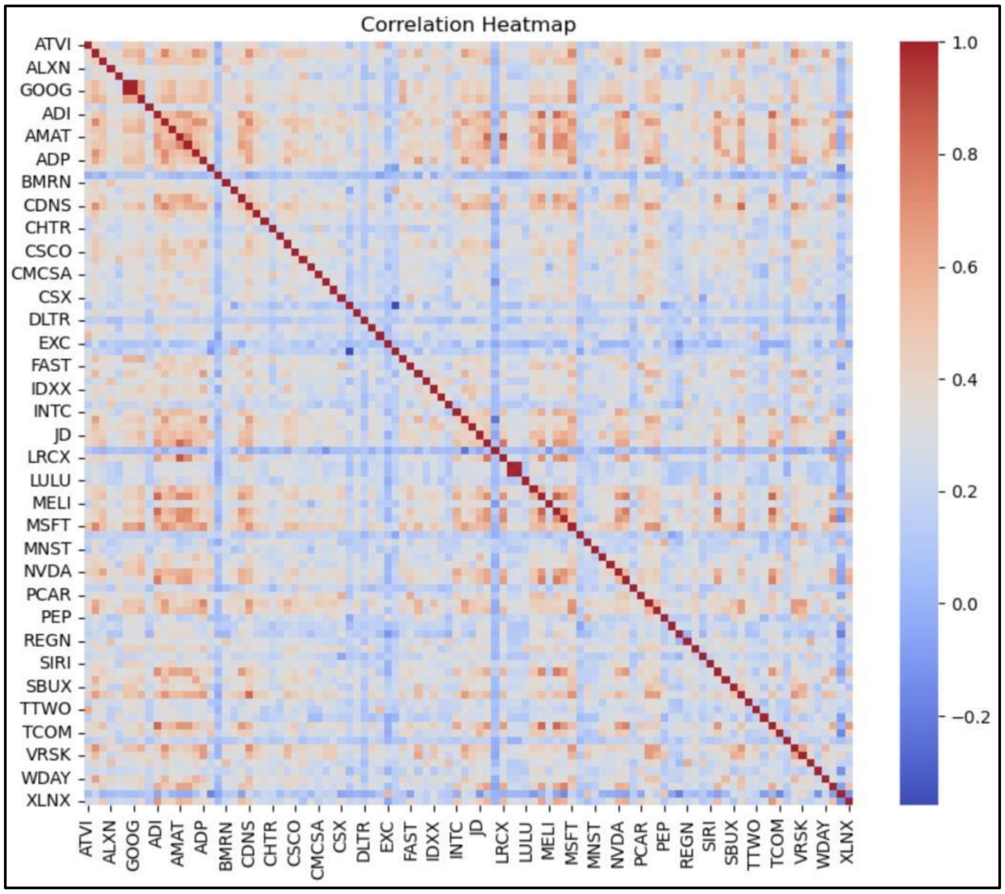
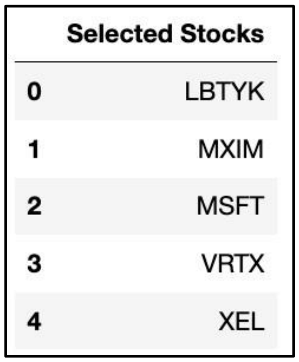


Figure 4: Correlation heatmap between the NASDAQ-100 index stocks for the year 2019

We wanted to take small steps forward before proceeding to find the optimal number of stocks for the fund. Initially, our focus was on identifying the top 5 stocks and determining their respective weights. To achieve this, we calculated the percentage return percentage return for each stock and subsequently evaluated the correlations among their returns (refer to Figure 4).

Leveraging these correlation values, we integrated them into our Integer Program, and the outcome unveiled the 5 most promising stocks that exhibited the potential for our portfolio. It was found that significant stocks of the index are Liberty Global PLC (multinational telecommunications company), Maxim Integrated (analog and mixed-signal integrated circuits company), Microsoft (multinational technology corporation),Vertex Pharmaceuticals Incorporated (a biopharmaceutical company), Xcel Energy Inc (utility holding company).



	Selected Stocks
0	LBTYK
1	MXIM
2	MSFT
3	VRTX
4	XEL

Table 1: Top 5 selected stocks

Following this, we calculated the optimal weight for each of the selected stocks. We employed the Linear Program to calculate these weights, resulting in the following allocations:

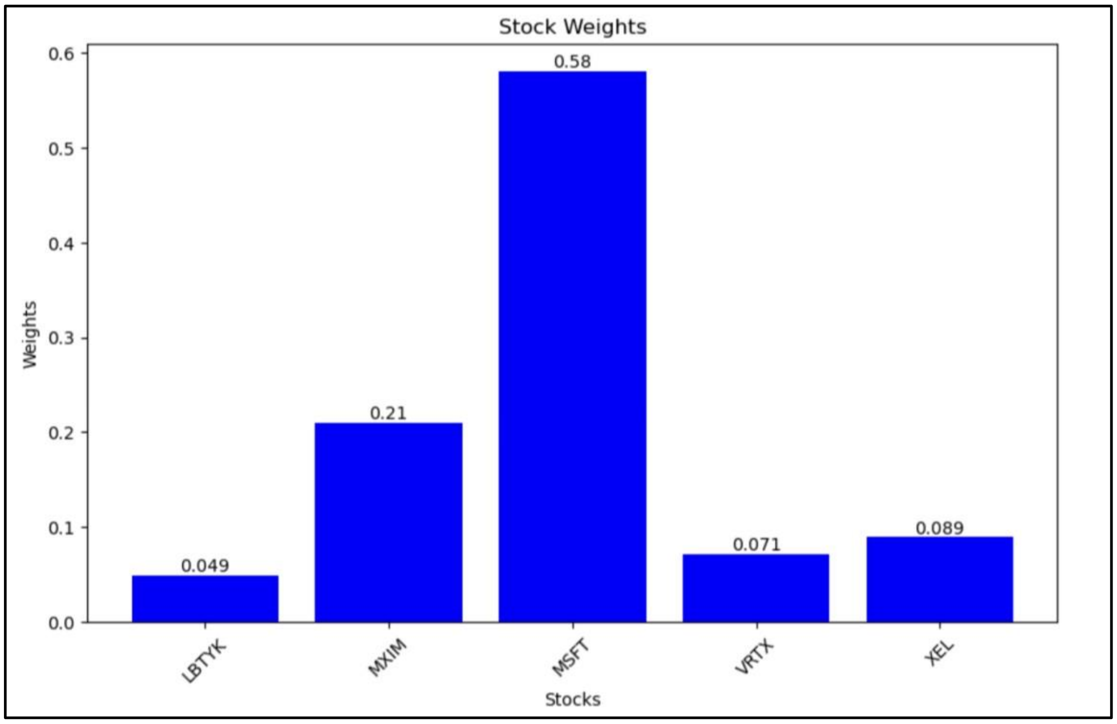
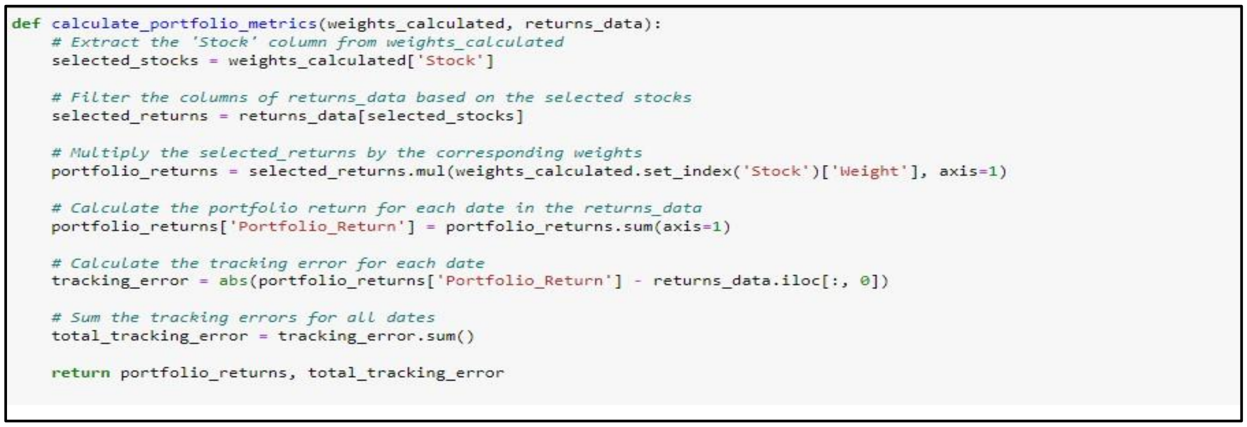


Figure 5: Weights of the Top 5 selected stocks from the NASDAQ-100 for the year 2019



```python
def calculate_portfolio_metrics(weights_calculated, returns_data):
    # Extract the 'Stock' column from weights_calculated
    selected_stocks = weights_calculated['Stock']

    # Filter the columns of returns_data based on the selected stocks
    selected_returns = returns_data[selected_stocks]

    # Multiply the selected_returns by the corresponding weights
    portfolio_returns = selected_returns.mul(weights_calculated.set_index('Stock')['Weight'], axis=1)

    # Calculate the portfolio return for each date in the returns_data
    portfolio_returns['Portfolio_Return'] = portfolio_returns.sum(axis=1)

    # Calculate the tracking error for each date
    tracking_error = abs(portfolio_returns['Portfolio_Return'] - returns_data.iloc[:, 0])

    # Sum the tracking errors for all dates
    total_tracking_error = tracking_error.sum()

    return portfolio_returns, total_tracking_error
```

Code Chunk 3: Function to calculate returns and error of portfolio

With weights in hand, we applied it to our selected stocks for the year 2020. This allowed us to gauge how the portfolio would perform concerning the broader market index. Our results show a variance of 1.11 which is promising and highlights a substantial correlation between our fund and the index. This can be further illustrated from Figure 6.

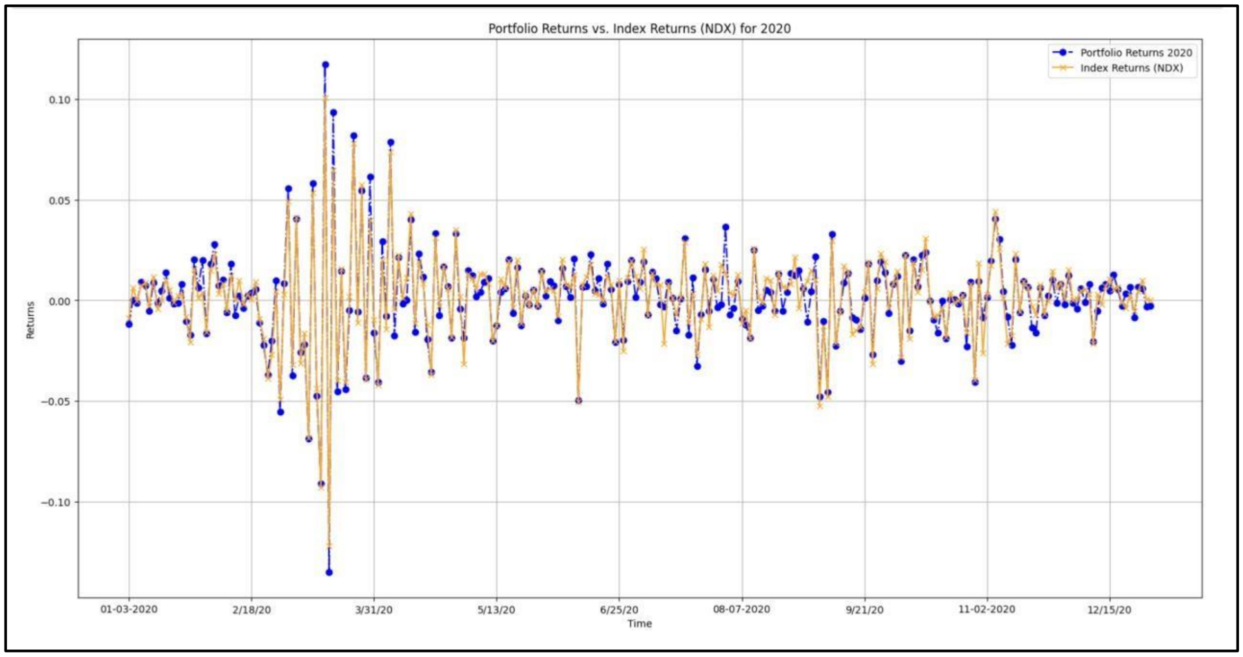
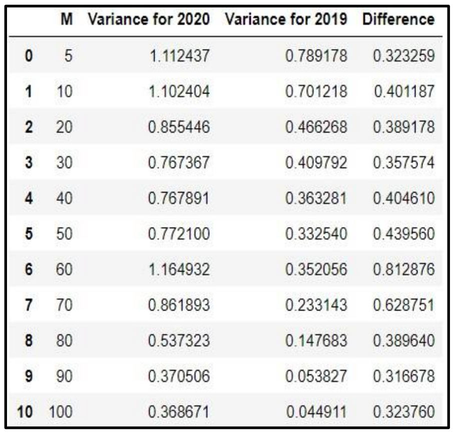


Figure 6: Comparing the returns of the stock fund with 5 best stocks and the NASDAQ-100 for the year 2020

5. Optimal Number of Stocks

We investigated the variance $(\sum_{i=1}^{T}| \quad \sum_{i}^{m} q_t - {}_{=1} w_i r_{it}|)$ n returns between the years 2020 and 2019 across different numbers of selected stocks for the fund, denoted as 'm’. We varied ‘m’ from 5 all the way up to 100. The results, as depicted in Figure 8, exhibit an interesting trend. The variance decreases as 'm' increases up until 'm' equals 50, and subsequently, it peaks at ‘m’ equals 60 after which it decreases before stabilizing at 'm' equals 90. Our primary objective was to minimize variance while selecting the fewest number of stocks, corresponding to an optimal 'm' value.



	M	Variance for 2020	Variance for 2019	Difference
0	5	1.112437	0.789178	0.323259
1	10	1.102404	0.701218	0.401187
2	20	0.855446	0.466268	0.389178
3	30	0.767367	0.409792	0.357574
4	40	0.767891	0.363281	0.404610
5	50	0.772100	0.332540	0.439560
6	60	1.164932	0.352056	0.812876
7	70	0.861893	0.233143	0.628751
8	80	0.537323	0.147683	0.389640
9	90	0.370506	0.053827	0.316678
10	100	0.368671	0.044911	0.323760

Table 2: Total Variance w.r.t NASDAQ for 2019 and 2020 for various values of m

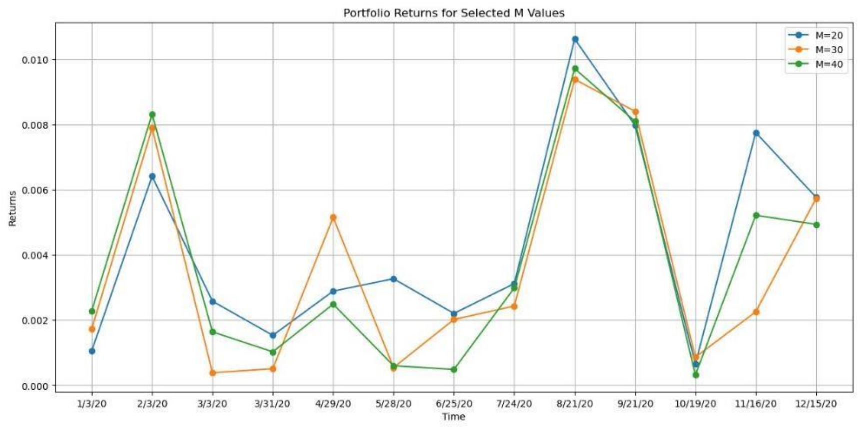


Figure 9: Returns for m = 20, 30 and 40

We also wanted to look at the trend for the returns of the fund for different values of M especially the ones that showed the smallest difference in the variance (M= 20, 30, 40). We didn't opt for M

= 70, 80, 90 and 100 since our goal is to build a more manageable and cost-effective index fund. The same can be observed in figure 9 above.

An 'm' value of 30 would be a reasonable choice, with a variance of 0.767367 for the year 2020 and 0.409792 for the year 2019. However, it's worth noting that the variance remains low when 'm' is set to 90 and 100. Nevertheless, we advise against opting for such a high number of stocks, as it may be more advantageous to invest in the entire NASDAQ-100 at that point.

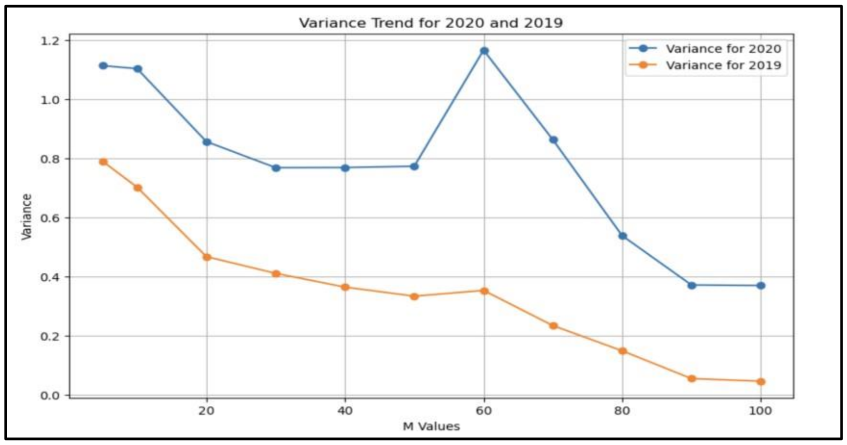


Figure 7: Variance trend for the years 2019 and 2020

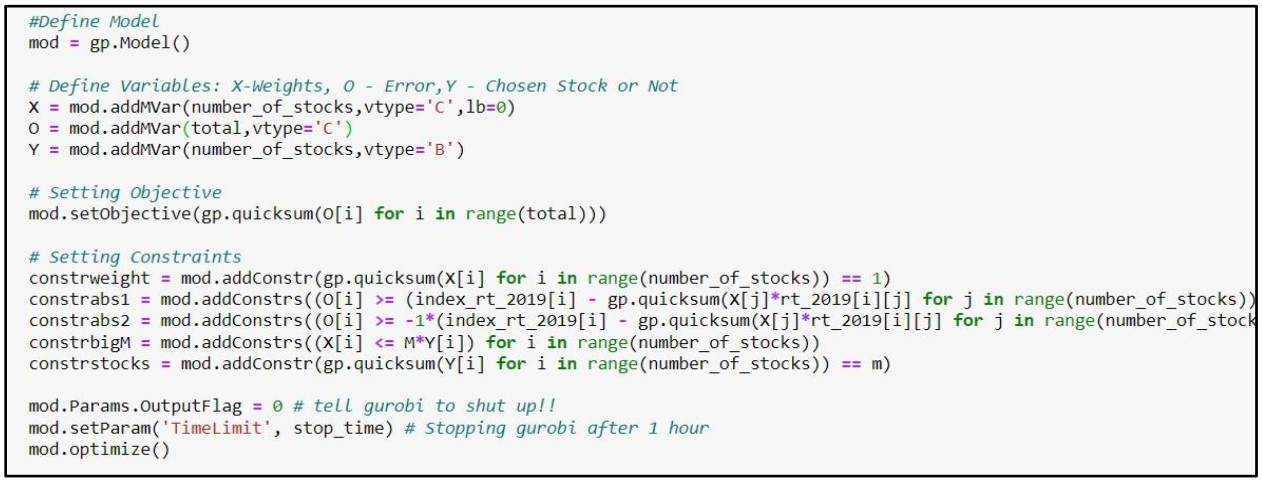
Mixed Integer Programming

We designed a MIP model which had a mix of binary and integer variables. The objective function and the constraints remain the same - except for an addition of "Big M" constraint where we make sure x_{ij} is zero if y_j is zero - meaning that if the stock is not chosen no weight should be assigned to that stock.

The Big - M constraint:

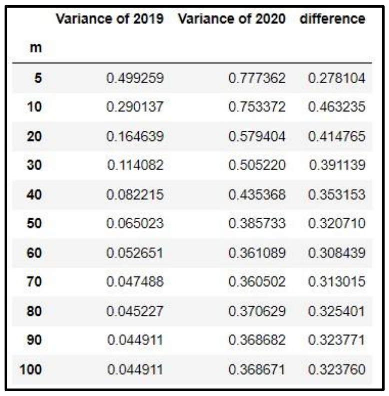
$$x_{ij} \leq M \times y_j$$

We can consider M to be a sufficiently large value, but the smallest big value of M can be 1 because the sum of weights can be 1.



```
#Define Model
mod = gp.Model()

# Define Variables: X-Weights, O - Error,Y - Chosen Stock or Not
X = mod.addMVar(number_of_stocks,vtype='C',lb=0)
O = mod.addMVar(total,vtype='C')
Y = mod.addMVar(number_of_stocks,vtype='B')

# Setting Objective
mod.setObjective(gp.quicksum(O[i] for i in range(total)))

# Setting Constraints
constrweight = mod.addConstr(gp.quicksum(X[i] for i in range(number_of_stocks)) == 1)
constrabs1 = mod.addConstrs((O[i] >= (index_rt_2019[i] - gp.quicksum(X[j]*rt_2019[i][j] for j in range(number_of_stocks))
constrabs2 = mod.addConstrs((O[i] >= -1*(index_rt_2019[i] - gp.quicksum(X[j]*rt_2019[i][j] for j in range(number_of_stock
constrbigM = mod.addConstrs((X[i] <= M*Y[i]) for i in range(number_of_stocks))
constrstocks = mod.addConstr(gp.quicksum(Y[i] for i in range(number_of_stocks)) == m)

mod.Params.OutputFlag = 0 # tell gurobi to shut up!!
mod.setParam('TimeLimit', stop_time) # Stopping gurobi after 1 hour
mod.optimize()
```

Code Chunk 4: Model for MIP



	Variance of 2019	Variance of 2020	difference
m			
5	0.499259	0.777362	0.278104
10	0.290137	0.753372	0.463235
20	0.164639	0.579404	0.414765
30	0.114082	0.505220	0.391139
40	0.082215	0.435368	0.353153
50	0.065023	0.385733	0.320710
60	0.052651	0.361089	0.308439
70	0.047488	0.360502	0.313015
80	0.045227	0.370629	0.325401
90	0.044911	0.368682	0.323771
100	0.044911	0.368671	0.323760

Table 3: Total Variance w.r.t NASDAQ for 2019 and 2020 for various values of m for MIP

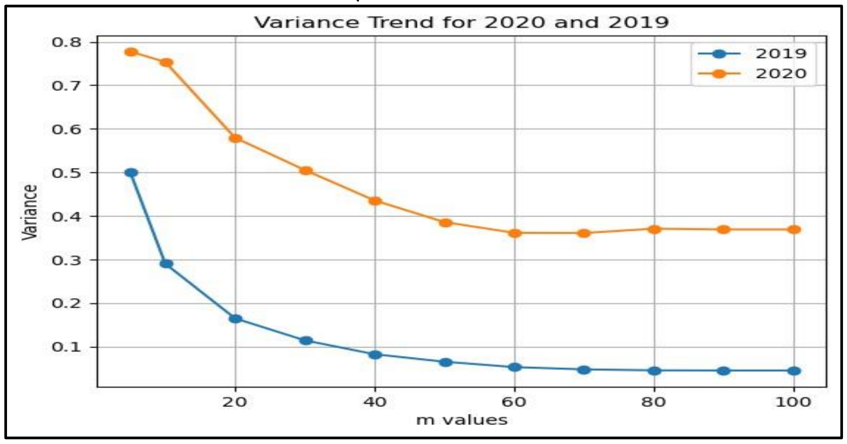


Figure 8: Variance trend for the years 2019 and 2020 using MIP.

As depicted in Figure 8, the performance stabilizes at approximately m=50, indicating that further additions of stocks do not significantly enhance performance. Beyond m=40, we observe a declining trend in performance. Therefore, we can opt for m=30, consistent with our previous choice. This selection aligns with economic considerations, as the diminishing variance reduction offered by m=40 may not be justified.

6. Recommendations

Examining the above graph, it is prudent to **incorporate 30 stocks** in our portfolio to emulate most of the stocks in the NASDAQ-100 Index and thereby reduce potential loss in returns. To determine a more precise value for 'm,' we should assess the trade-off between the cost of adding more stocks and the associated error.

Selecting Stocks and Assigning Weights:

- In our initial model, we observed an enhancement in performance as we increased the number of included stocks. This positive correlation between the count of stocks and performance implies that a larger selection of stocks results in more favorable outcomes

- However, the second method surpasses the first model, delivering superior and consistently strong performance. It exhibits smaller deviations in performance, signifying increased stability in returns. Nevertheless, this enhanced performance does entail greater computational requirements

Consequently, we recommend employing the second model for superior returns. It's worth noting that our analysis was conducted under a time constraint of 1 hour, suggesting potential for further improvement if we extend the analysis duration to better monitor returns.