Portfolio Optimization Analysis for Mutual Funds

Optimization Project2 - Group 18

Group Members:
Hayoung Kim (hk26786)
Mandeep S Burdak (msb4384)
Nicolas Rey(ner785)
Sanjana Nayak (svn378)

Table of Contents

1.Introduction and Problem

Equity money management includes active and passive strategies. Passive strategies, such as index funds, aim to replicate a market index's performance. Constructing an index fund that tracks a market index by purchasing all its constituent stocks is impractical and costly due to frequent rebalancing.

This project's objective is to create a more efficient index fund with a reduced number of stocks (m) to mirror the NASDAQ-100 index. We will utilize a similarity matrix (ϱ) representing relationships between stocks to select m stocks via integer programming. We will then use linear programming to determine the quantity of each selected stock for the portfolio. The fund's performance will be evaluated against the NASDAQ-100 index for various values of m.

2. Binary and Continuous Model

Problem Statement

This project involves a dual challenge: binary stock selection and continuous portfolio weight calculation, both optimized using Gurobi.

1. Binary Optimization (Stock Selection): Our objective is to select m stocks from n options, using binary variables (yj) to represent chosen stocks (yj = 1) or exclusions (yj = 0). Gurobi is employed to find the best stock combination aligning with the index, respecting binary constraints.

2. Continuous Optimization (Portfolio Weight Calculation): Calculating continuous portfolio weights (wi) for selected stocks involves mathematical optimization. Gurobi is leveraged to address this challenge by framing it as a continuous linear program. Additionally, we minimize the absolute difference between index returns and weighted stock returns using Gurobi's linear programming capabilities.

This project combines binary and continuous optimization techniques to construct an effective portfolio that mirrors the index's performance.

Methodology

Optimizer 1

Decision Variables

1. x: Stock Selection Matrix - "Matrix x, denoted as the Stock Selection Matrix, is a square $n*n$ matrix where each element xij is equal to 1 if stock j is determined to be the most suitable representation of stock i, and 0 if not.

```
# xij = 1 if stock j in index is the most similar stock i, 0 otherwise
x = finder.addMVar(shape=(n,n), vtype='B')
```

2. y: Selected Stocks: In the variable y, a binary array of length n is used to represent the selection of m stocks chosen for inclusion in the fund, where $y[i]$ = 1 indicates a chosen stock, and $y[i]$ = 0 signifies an excluded stock.

```
# yj = 1 if j is selected in the fund, 0 otherwise
y = finder.addMVar(n, vtype='B')
```

Objective

Maximize the product $rij*xij$, where rij represents the correlation between the returns of stock i and stock j.

```
# The objective of the model maximizes the similarity between the n stocks and their representatives in the fund.
finder.setObjective(gp.quicksum(phi[i][j]*x[i][j] for i in range(n) for j in range(n)),sense=gp.GRB.MAXIMIZE)
```

Constraints

1. Total number of selected stocks should be m: The constraint ensures that exactly m stocks are selected for inclusion in the fund.

```
# Max m(5 default) stocks in the fund
f1 = finder.addConstr(gp.quicksum(y[j] for j in range(n))==m)
```

2. Each stock i can be represented by only one stock j, meaning only a single representation per stock (i) is allowed.

```
# Each stock in NASDAQ should be represented by on of the stock's found
f2 = finder.addConstrs(gp.quicksum(x[i][j] for j in range(n))==1 for i in range(n))
```

3. Only a selected j^{th} stock can be used to represent a i^{th} stock: If a stock (j) is selected to represent stock (i), the constraint enforces that this selected stock (j) must be part of the fund as well, ensuring a one-to-one mapping between stocks in the fund and their representations.

$$x_{ij} <= y_i \text{ for } i=1,2,...n \text{ and for } j=1,2,...n)$$

```
# X-Y relationship
f3 = finder.addConstrs(x[i][j]<=y[j] for i in range(n) for j in range(n))
finder.optimize()
```

Optimizer 2

Decision Variables

1. a: Stock Weights: This array consists of n elements, with each element representing the proportion of the fund's total value allocated to a specific stock, determining its influence in the portfolio.

```
a = weigher.addMVar(n)
```

2. b: Absolute error in returns: The "absolute error in returns" is a one-dimensional array containing the absolute differences between the returns of the index and the returns generated by the weighted combination of the stocks we have selected. It quantifies the magnitude of the divergence between our portfolio's performance and the index's performance.

```
b = weigher.addMVar(t)
```

Objective

Optimize by minimizing the absolute disparity between the returns of our index fund and the aggregated returns generated by our carefully selected stocks.

```
weigher.setObjective(gp.quicksum(b[i] for i in range(t)))
```

Constraints

1. The dummy variable employed to represent absolute error, "returns," is required to exceed the absolute value of the difference in returns, ensuring a positive disparity.

```
w1 = weigher.addConstrs(q[i]-gp.quicksum(a[j]*ret[i][j] for j in range(n)) <= b[i] for i in range(t))
```

2. The dummy variable "returns," employed to indicate absolute error, should surpass the negative value of the return difference, thus preserving a positive difference.

```
w2 = weigher.addConstrs(gp.quicksum(a[j]*ret[i][j] for j in range(n))-q[i] <= b[i] for i in range(t))
```

3. Only the stocks selected in the preceding optimization process may be assigned a weight (where ai should be less than or equal to Mi for all i ranging from 1 to n).

```
# We are just using the stocks that we found
w3 = weigher.addConstrs(a[i]<=M[i] for i in range(n))
```

4. The assigned weights for the individual stocks must collectively sum to 1, denoting a total portfolio weight of ai =1.

```
# Weights should be equal to 1
w4 = weigher.addConstr(gp.quicksum(a[i] for i in range(n)) == 1)
weigher.optimize()
```

Results

1. The five best stocks to include in the portfolio along with their weights are as follows (when m= 5) :

	stock	weight	m
0	LBTYK	0.04886174835252490	5
1	MXIM	0.21038806005665600	5
2	MSFT	0.5803519807862960	5
3	VRTX	0.07119021516911040	5
4	XEL	0.08920799563541280	5

In constructing an optimized portfolio with 5 stocks, the selection of individual stocks and their associated weights plays a pivotal role in achieving the desired balance of performance and risk. After a rigorous evaluation, the portfolio management team has identified the five best stocks to include in the portfolio, each accompanied by their weight, with the objective of closely mirroring the index's performance.

1) Microsoft (MSFT) - Weight: 58%
Microsoft, a global leader in software and cloud services, holds the highest weight of 58% in the portfolio. Its strong market presence, continuous innovation, and reliable dividends contribute significantly to the portfolio's overall returns and stability.

2) Maxim Integrated Products Inc. (MXIM) - Weight: 21%
Maxim Integrated, a prominent semiconductor company known for its cutting-edge technologies, has a weight of 21% in the portfolio. It brings diversity to the tech sector, offering potential for substantial growth and stability.

3) Xcel Energy Inc. (XEL) - Weight: 9%
Xcel Energy Inc., a major utility company, constitutes 9% of the portfolio. Its presence adds an element of stability and predictability, as the utility sector is known for its consistent performance and dividends.

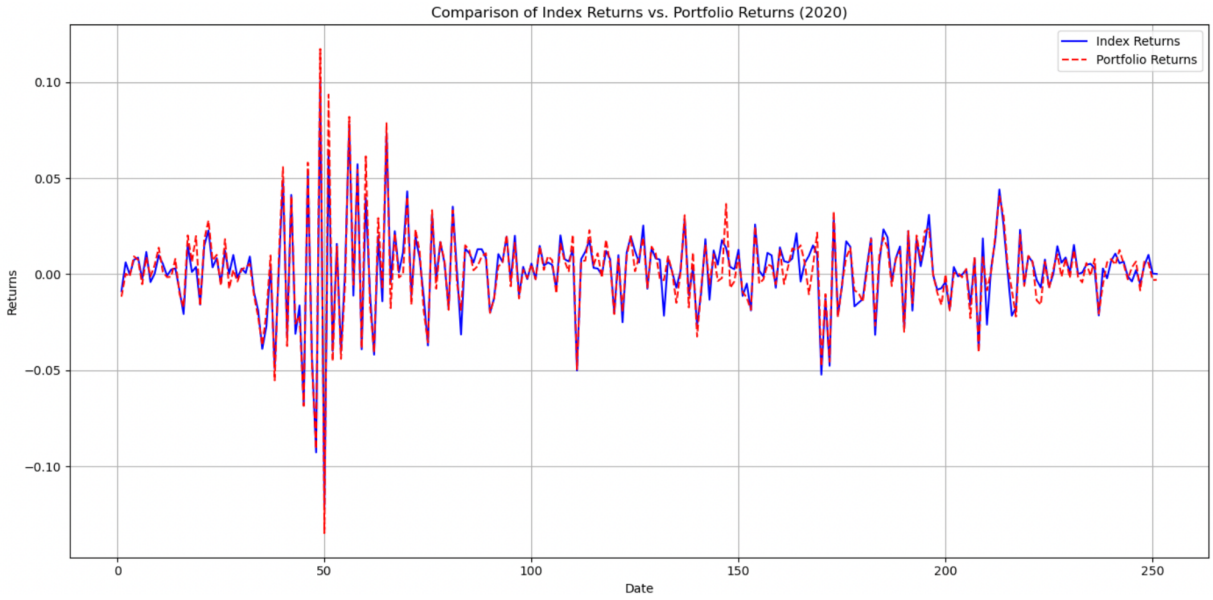
4) Vertex Pharmaceuticals Inc. (VRTX) - Weight: 7%
Vertex Pharmaceuticals, a biotechnology company specializing in creating transformative medicines, holds a 7% weight in the portfolio. It introduces an element of innovation and potential for high growth in the healthcare sector.

5) Liberty Global plc (LBTYK) - Weight: 5%
Liberty Global, a multinational telecommunications company, makes up 5% of the portfolio. It diversifies the portfolio into the telecommunications sector, offering potential for growth in a globally connected world.

These five selected stocks represent a balanced mix of growth potential, stability, and diversity. The allocation of weights reflects the portfolio's strategy to minimize tracking error while optimizing performance. With these stocks, the portfolio aims to replicate the returns of the index closely while efficiently managing risk. The specific weighting of each stock ensures that the portfolio maintains its desired characteristics, with the combination expected to deliver a well-rounded performance aligned with the investment objectives.

2. How well does this portfolio track the index in 2020 (when m= 5) ?

When m=5, the tracking error is 1.112437. This value represents how well the portfolio tracks the index, with lower values indicating better tracking performance. This can be seen in the graph below.



While both lines move in a generally similar pattern, there are clear differences between them, indicating a tracking error. This means that the portfolio does not perfectly replicate the index's performance. There are periods where the portfolio outperforms the index and vice versa.

- Similar Trends: Both the index returns and the portfolio returns generally move in the same direction, indicating that the portfolio's performance is somewhat correlated with that of the index.

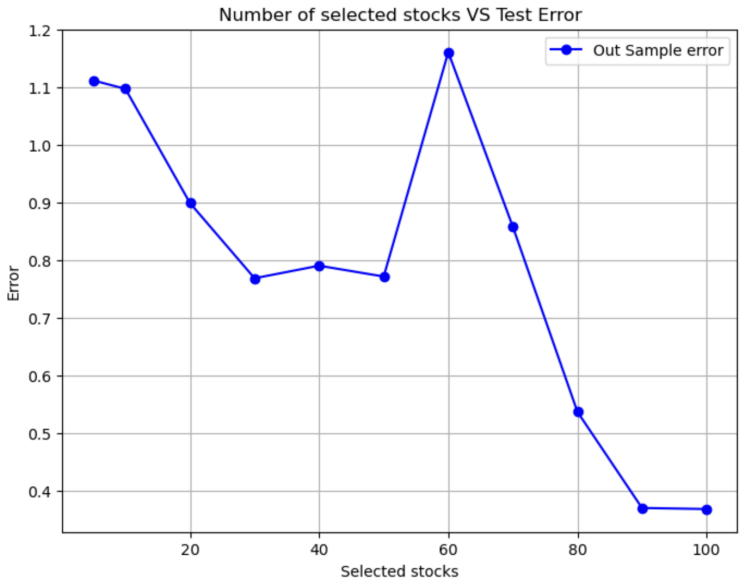
- Volatility Around Day 50-100: There is a significant spike in volatility for both the index and the portfolio around the range of Day 50 to Day 100. Both lines experience sharp fluctuations, but it's notable that the portfolio's returns exhibit more pronounced declines than the index during this period.

- Stabilization Post Day 100: After Day 100, both the index and the portfolio returns stabilize, with fluctuations being less severe compared to the earlier period. The portfolio returns seem to have a slightly higher volatility than the index returns during this period but not by a wide margin.

In conclusion, the portfolio does track the index to some extent in 2020, as evident from the similar movement patterns. However, despite having assets or strategies that tracks index's performance quite well, there are periods of divergence, indicating that constructing a portfolio with only 5 stocks is insufficient to fully reflect the index.

3. Analyze the performance of the portfolio for each value of m. How does the performance change?

	m	Error
0	5	1.112437
1	10	1.097709
2	20	0.899598
3	30	0.769110
4	40	0.790884
5	50	0.772100
6	60	1.161397
7	70	0.858428
8	80	0.537323
9	90	0.370506
10	100	0.368671



-Decreasing Trend (m=10 to m=30)
The error decreases significantly as m increases from 10 to 30. A noteworthy drop in tracking error occurs from m=10 to m=20, reaching 0.89. This trend continues as m further increases to 30, where the tracking error reaches at a relatively low value of 0.76. This indicates that the portfolio's performance improves with the addition of more diverse stocks.

- Sudden Spike (m=50 to m=60)
However, the tracking error experiences a temporary spike at m=60, rising to 1.16. This is markedly higher than the error rates for m=50 and m=70, which stand at 0.77 and 0.85, respectively. Such a spike is unexpected since the error rate is generally anticipated to decrease or at least remain relatively stable as m increases. This anomaly suggests that increasing m doesn't necessarily guarantee improved performance.

- Stabilizing Performance (m=60 to m=100)
Beyond this peak, the tracking error stabilizes and resumes its downward trend. It consistently remains below 1.0 from m=70 onwards, reaching a minimum of 0.36 at m=100.

- Performance Change with m:
The performance of the portfolio shows a notable change with different values of m. The tracking error initially tends to decrease significantly as m increases, indicating that including more stocks improves tracking performance. However, this trend is not linear, and there is a temporary increase in tracking error at a specific point (m=60) before it stabilizes.

In general, there is an evident trade-off between the size of the portfolio (m) and tracking error. While a larger number of stocks (higher m) may lead to more accurate tracking, it may also introduce complexities and potential inefficiencies. The optimal value of m depends on various factors, including the index being tracked, the market environment, and the available data. Analyzing this trade-off is crucial for portfolio optimization.

4. Is there some value of m, where there are diminishing returns of including more stocks in the portfolio?

Indeed, there is an observable point at which the inclusion of more stocks in the portfolio starts to exhibit diminishing returns in terms of improving tracking performance. This is evident from the provided tracking error values for different values of m.

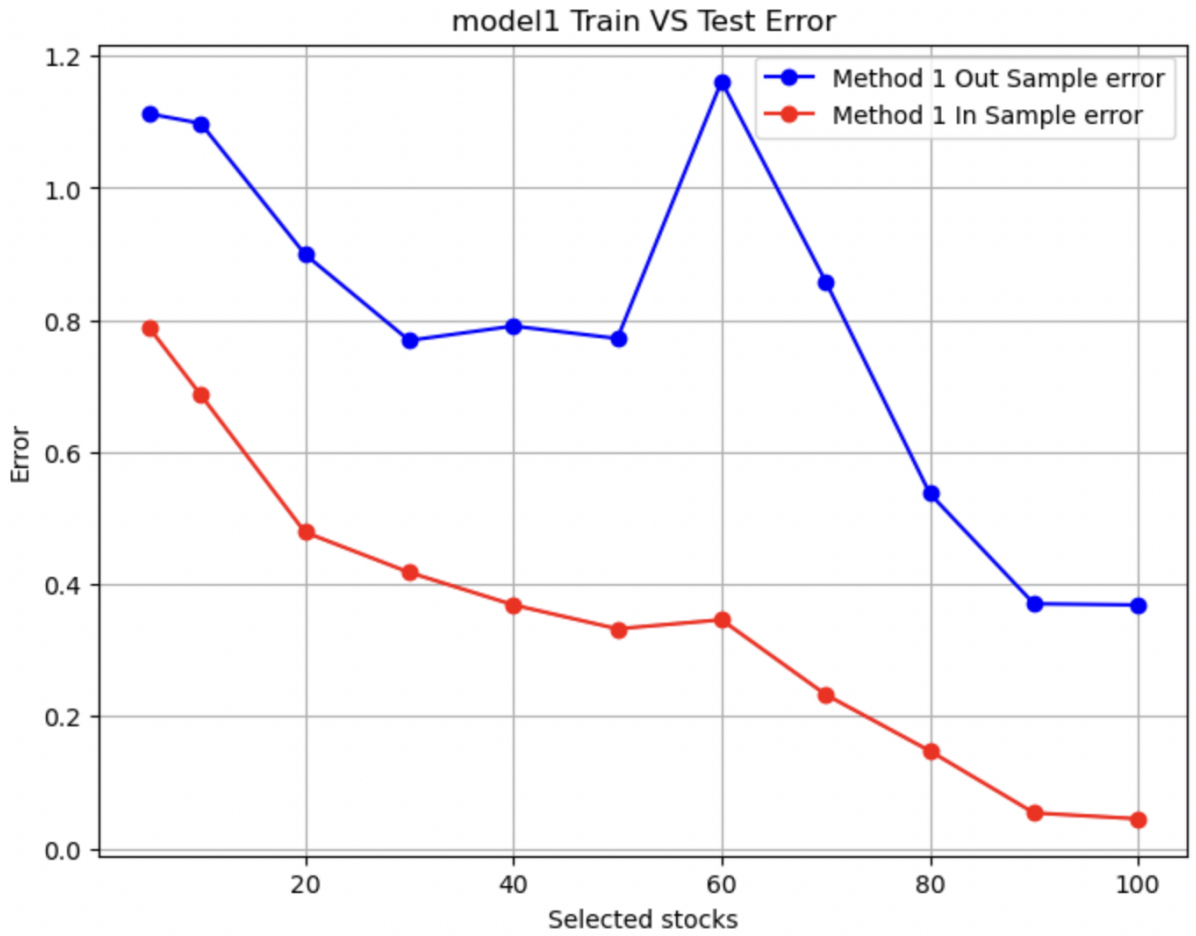
Initially, as m increases, the tracking error consistently decreases, suggesting that a larger number of stocks in the portfolio enhances tracking performance. However, there is a point of inflection, notably around m=50, where the trend reverses. Beyond this point, increasing m

results in a temporary spike in tracking error, indicating that additional stocks are not contributing to better tracking but may even introduce noise or inefficiencies.

Subsequently, after this temporary increase, the tracking error continues to decrease more gradually but remains below the levels observed before m=60. This shift in the trend demonstrates that there is indeed a point where including more stocks in the portfolio leads to diminishing returns regarding tracking performance.

In summary, the optimal value of m, where the portfolio achieves the most efficient tracking of the index, appears to be situated around m=30 to m=50, as the error rates are relatively low (0.77 to 0.79). Beyond this point, the benefits of adding more stocks to the portfolio become less significant, implying the presence of diminishing returns in terms of stock inclusion.

5. How is performance in 2019 different from performance in 2020? Why is it different?



The above graph illustrates the "Out Sample error (2020)" and "In Sample error (2019)" for our model1 against the number of selected stocks. Here's the interpretation based on the graph:

- In Sample Error (2019, Red Line)
As the number of selected stocks increases, the in-sample error steadily decreases. This suggests that the model's fit on the training data improves with the inclusion of more stocks in the sample. However, by the time we reach around 90 stocks, the decrease in error starts to plateau.

- Out Sample Error (2020, Blue Line)
The out-sample error behaves differently. Initially, as the number of selected stocks increases to around 40, the error decreases slightly. However, as we move from 40 to around 60 stocks, the error significantly increases, reaching its peak around 60 stocks. This suggests overfitting, where the model might be too closely tailored to the training data and fails to generalize well on new, unseen data. After 60 stocks, the error decreases again as the number of stocks increases, reaching a lowest level at around 90 stocks.

The ideal number of stocks for this model seems to be around 30 or slightly more, as both in-sample and out-sample errors are relatively low. As we increase the number of stocks further, the model starts to overfit around the 40-60 stock range, as evidenced by the spike in the out-sample error.

Then, why did the discrepancy in performance occur?

Several factors may contribute to this difference in performance between the two years:

1) Market Environment: The financial markets are subject to constant fluctuations and variations due to economic conditions, investor sentiment, and geopolitical factors. Differences in market dynamics between 2019 and 2020 could significantly influence the performance of the portfolio. For instance, the overall economic climate, interest rates, and market volatility can vary year by year.

2) Stock Composition: The specific stocks included in the portfolio may have performed differently in 2019 and 2020. Some stocks might have had exceptional years in terms of returns, while others may have faced challenges. These variations in individual stock performance would directly impact the overall tracking error.

3) External Events: Extraordinary events, such as the COVID-19 pandemic, political developments, or shifts in trade dynamics, can have substantial effects on stock prices. The occurrence of such events during 2020 could lead to increased volatility and uncertainty, influencing the portfolio's performance.

4) Data Tailoring: The portfolio was designed using data from 2019, with the objective of minimizing tracking error for that specific year. As a result, it might not necessarily be optimized to replicate the returns of the following year (2020). The differences in stock behavior between these years could be inadequately accounted for.

In summary, the variance in performance between 2019 and 2020 is an outcome of multifaceted factors. These include the evolving market environment, the individual stock dynamics, external events, and the tailored nature of the portfolio to the specific year for which it was constructed. Understanding these disparities is essential in evaluating the portfolio's adaptability and its ability to provide an accurate representation of the index's performance across diverse market conditions.

3. Mixed Model

Problem Statement

On the other hand, we can use a mixed model to find the ideal portfolio weights that closely track the NASDAQ-100 index. This approach, unlike the binary and continuous model, combines continuous variables (the weights of the stocks) with binary variables (the inclusion of stocks in the portfolio) in a single optimization problem. We define binary variables $y1, y2, y3, \ldots yn$ and impose constraints that force the $wi = 0$ if $yi = 0$, using the 'big M' technique. The objective is to minimize the difference between the index returns and the portfolio returns, while constraining the sum of binary variables to be equal to m.

Methodology

In our mixed model approach, we utilized the Gurobi optimizer to solve a mixed-integer programming (MIP) problem. This approach allowed us to precisely fine-tune the weights of the stocks in the portfolio, while also incorporating binary variables to indicate the inclusion or exclusion of specific stocks. Additionally, we introduced continuous variables to represent the absolute difference between the index returns and the portfolio returns. We implemented this mixed model as a function named 'model2', and below are the details of its composition.

Optimizer3

The decision variables we defined were as follows:
1) x: Stock Weights: A n-element array, which represents the weight assigned to each stock in our fund.
2) y: Selected Stocks: a n element array, that is 1 for the m stocks that are chosen to be included in the fund.
3) z: Absolute error in returns: A t-element array which represents the absolute difference between the index returns and the weighted sum of our selected stocks.

```
x = stocker.addMVar(n,vtype='C')
y = stocker.addMVar(n,vtype='B')
z = stocker.addMVar(t,vtype='C')
```

The objective function of our mixed model was to minimize the sum of the absolute differences between the index returns and the portfolio returns, ensuring that our portfolio mirrors the performance of the NASDAQ-100 index as closely as possible:

$$\min_{w,y,z} \sum_{t=1}^{T} z_t$$

```
stocker.setObjective(gp.quicksum(z[i] for i in range(t)))
```

We incorporated the following constraints into our model:
1) The total number of selected stocks should be equal to m: $\sum_{i}^{n} y_i = m$

```
# Max g stocks
c1 = stocker.addConstr(gp.quicksum(y[i] for i in range(n))==g)
```

2) If a stock is not selected (i.e., yi = 0), its weight must be zero (ensuring it is excluded from the portfolio). This was enforced using the 'big M' technique: $w_i \leq My_i \quad \text{for all} \quad i = 1, 2, \ldots, n$

```
# Big M constraint
c2 = stocker.addConstrs(x[i]<=y[i] for i in range(n))
```

The "big M" is set to 1 in our code, and this is a suitable choice because of the nature of our decision variables and the constraint $x[i]$ <= $y[i]$ ensures that:

- If $y[i]$ =0, then $x[i]$ <=0. Since $x[i]$ is a non-negative variable (as it represents the weight of a stock in the portfolio, and weights cannot be negative), this implies $x[i]$ = 0
- If $y[i]$ =1, then $x[i]$ <=1. This allows $x[i]$ to take any value between 0 and 1, inclusive.

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

3) The weights assigned to the stocks should sum up to 1:

```
# Sum up to 1
c3 = stocker.addConstr(gp.quicksum(x[i] for i in range(n))==1)
```

4) The variable Zt should be greater than or equal to the positive and negative values of the differences in returns: $z_t \geq q_t - \sum_{j=1}^{n} w_j r_{ij}$ and $z_t \geq \sum_{j=1}^{n} w_j r_{ij} - q_t$ for all $t = 1, 2, \ldots, T$

```
# Abs values
c4 = stocker.addConstrs(q[i]-gp.quicksum(x[j]*ret[i][j] for j in range(n)) <= z[i] for i in range(t))
c5 = stocker.addConstrs(gp.quicksum(x[j]*ret[i][j] for j in range(n))-q[i] <= z[i] for i in range(t))
```

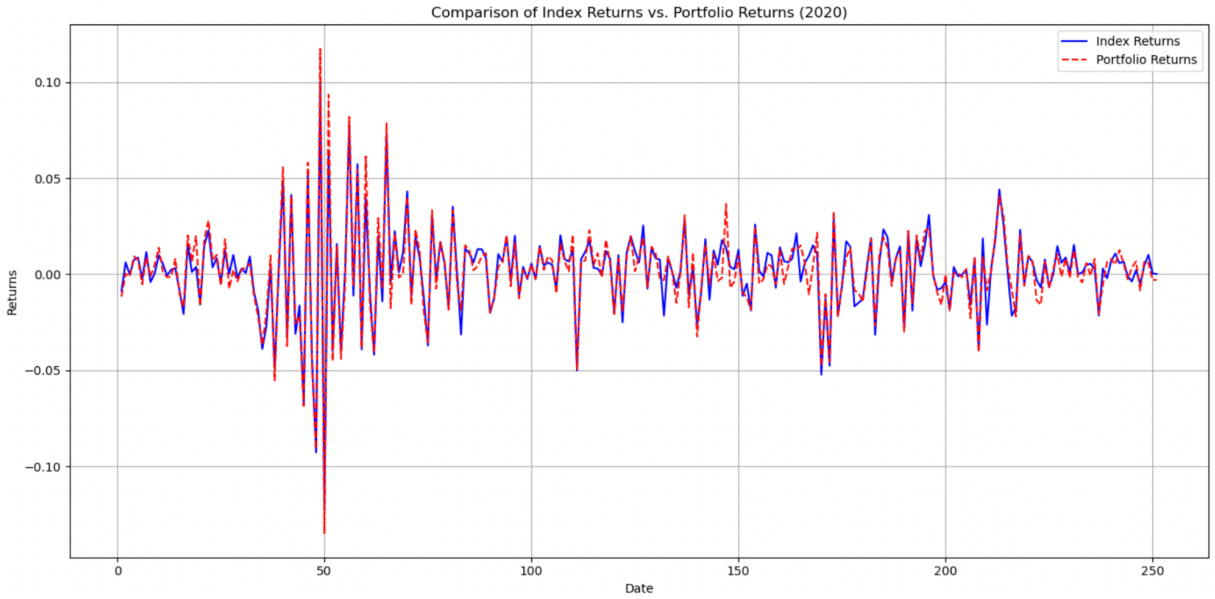
The results were analyzed for various values of m ranging from 5 to 100, using both in-sample data (2019 prices) and out-sample data (2020 prices). Through iterating over different values of m and utilizing the model_2 function, we computed the portfolio weights and corresponding returns. These were stored in lists and then transformed into a DataFrame, which was subsequently saved as 'weights2.csv'. Additionally, we calculated the tracking error for out-sample data (2020 prices) by comparing the sum of weighted returns of selected stocks with the index returns. This error was then stored in a list named error_test2 for further analysis.
The relationships between the number of selected stocks and the portfolio's tracking error were visualized using line charts. This analysis provided valuable insights into the optimal value of m that minimizes the tracking error while maintaining portfolio diversification.

Results

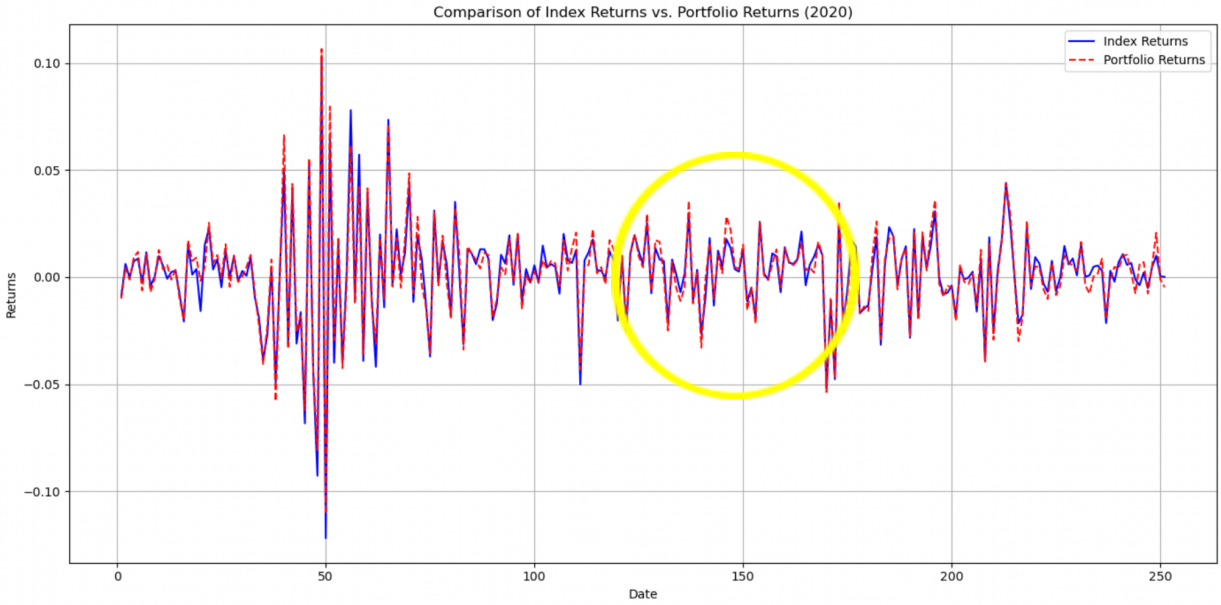
Tracking Error

Firstly, similar to what we did with model1 in question 2, we compared the returns of the mixed model (model2) with the 2020 index returns when m=5, and calculated the error rate. Consequently, we obtained an error rate of 0.777362, which is approximately 30% lower than the error rate of 1.112437 from model1. This is a positive signal indicating a significant improvement. The graph below illustrates this difference, particularly highlighting the areas where model1 had prominent errors, which have been significantly mitigated in model2 (marked with yellow circles)

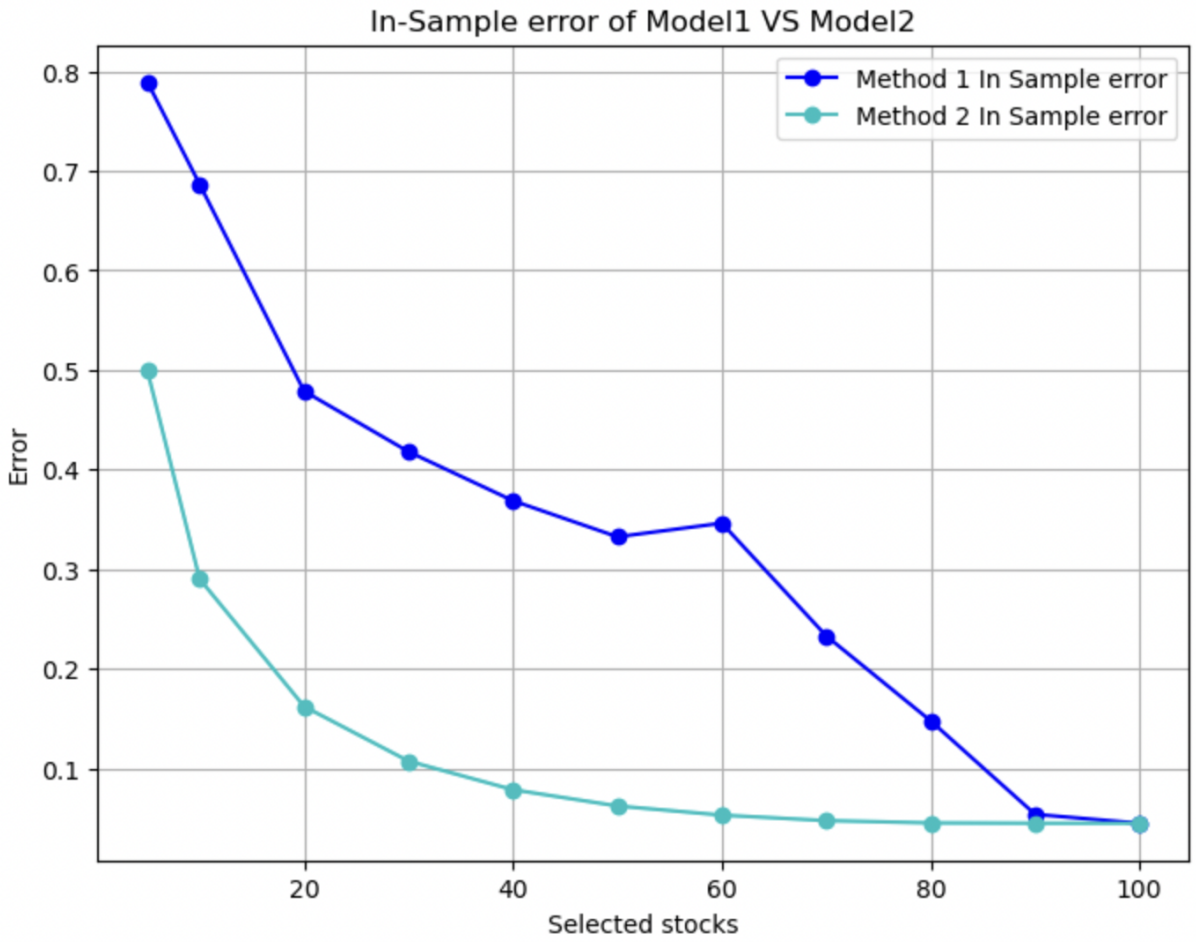
<model1,m=5 - index Returns vs Portfolio returns>



<**model2**,m=5 - index Returns vs Portfolio returns>

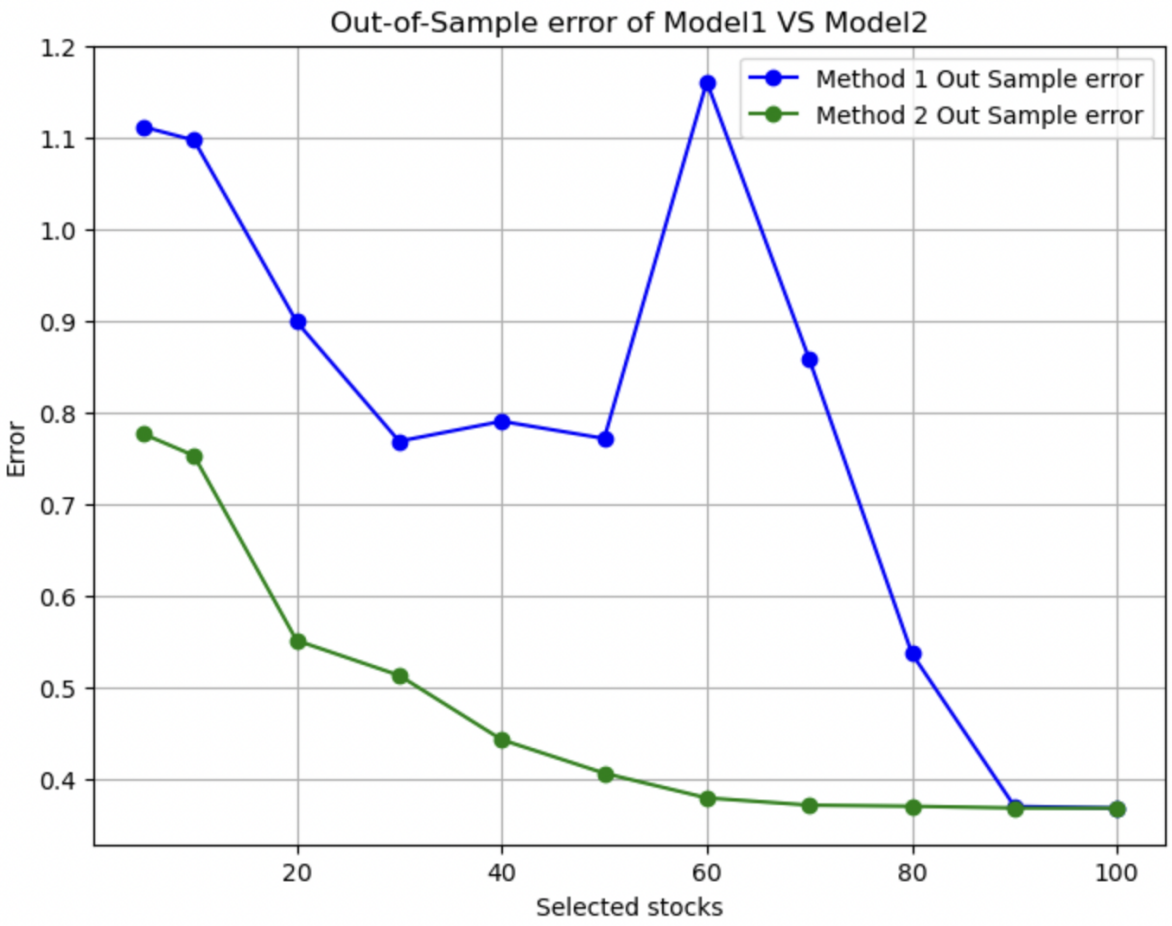


Subsequently, we investigated the tracking errors (absolute difference between index returns and selected stocks' returns) for model2 with various numbers of selected stocks, m = *10, 20, 30…100,* and obtained the following results.

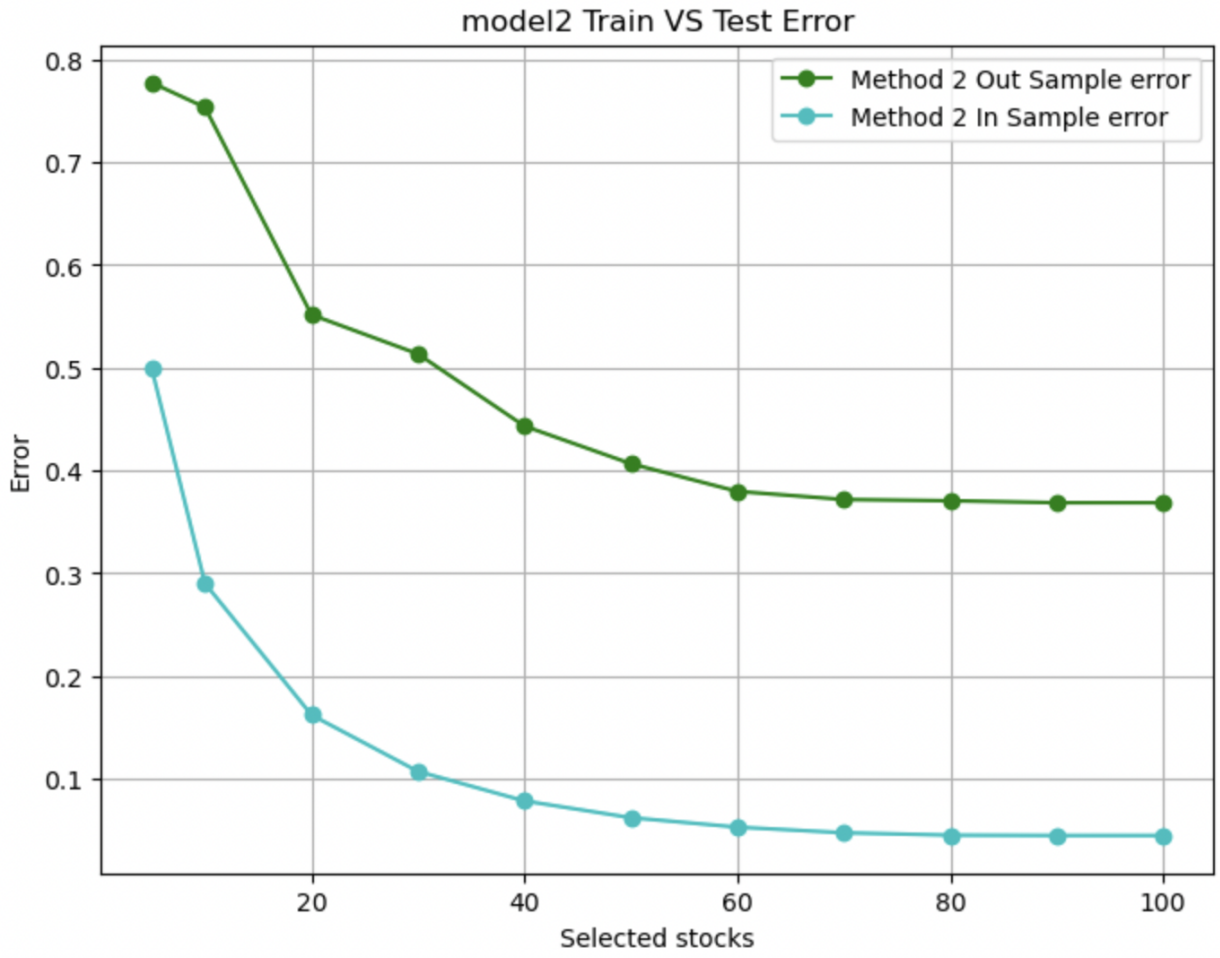


Generally, as the number of selected stocks increases, the error decreases. However, we observed a pattern where the rate of decrease in error diminishes significantly after m = *40* and from m = 60 onwards, adding more stocks doesn't significantly reduce the error.

When comparing the performance of our model1 (binary & continuous model) with model2, the effectiveness of the mixed model approach becomes evident. Starting from $m = 5$, there is a substantial difference in errors, with model1 at about 0.8 and model2 at about 0.5. Between $m = 5$ and $m = 10$, model2 already shows a more rapid decrease in error than model1, indicating better performance. The difference between the two error graphs remains around 0.2 to 0.3 between $m = 20$ and $m = 60$, and it is not until around $m = 80$ that the gap narrows to less than 0.1. This trend demonstrates that the unlike model2, effectiveness of model1 's performance becomes apparent only after a significant increase in the number of selected stocks (m).



In the case of out-sample error, both model1 and model2 exhibited somewhat jagged (less smooth) graph shapes, and generally higher error values than in-sample error. However, it is important to note that in model2, the volatility of the out-sample error in model1 is markedly reduced, and there is a noticeable decreasing trend. This indicates that model2 maintains its better performance compared to model1. While model1 shows an error of around 0.8 up to m = 50, and even exhibits an exceptional phenomenon where the error suddenly spikes to its highest peak at m = 60, model2 presents a different scenario. In model2, the error sharply decreases initially, and after m = 40, it is slightly greater than 0.4. Then, from m = 60 onwards, the error further reduces to below 0.4, maintaining a similar error status up to m = 100.



The Mixed Model (model2) exhibits the same directional movements at all points m for both in-sample and out-sample error graph lines, indicating a consistent relationship between the two. However, there is a clear difference in volatility and smoothness between the out-sample and in-sample performances. While the scope of in-sample error ranges from 0.05 to 0.5, the scope of out-sample error is approximately 0.37 to 0.77, showing that out-sample errors are consistently higher by about 0.2 to 0.3 across all intervals. This sustained significant difference between in-sample and out-sample errors suggests that we should consider the following points:

- Model Generalization:
The model's ability to generalize from training data (2019 prices) to new, unseen data (2020 prices) is evident, although the decrease in performance is noticeable. This is reflected in the consistently higher out-sample errors, which indicates that while the model captures the underlying patterns in the data, it might not be fully adapting to new market conditions or external factors that could affect stock prices in 2020.

- Error Stability:

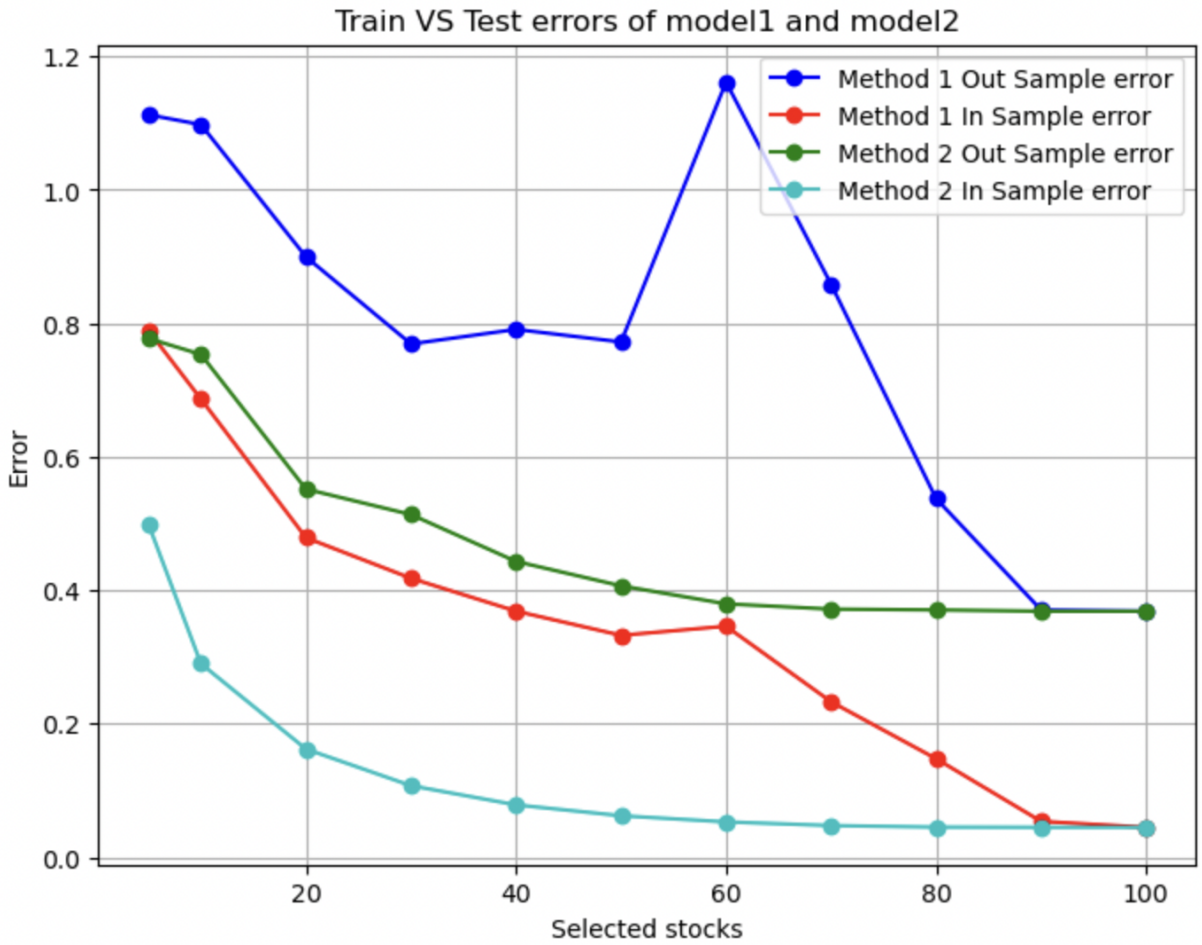
The stability in the error difference across all points of m suggests that the model has reliable performance. This predictability is crucial for making informed investment decisions based on the model's output.

- Potential Overfitting:

The discrepancy between in-sample and out-sample errors, especially with the low range of in-sample errors, suggests that the model might be slightly overfitting to the training data. While the model has been fine-tuned to capture the intricacies of the 2019 market, it might be less adept at handling the volatility and market shifts in 2020.

- Room for Improvement:

The consistent error difference, along with the higher range of out-sample errors, signals that there is room for further optimization. Refining the model to better adapt to new data and minimize the error difference could lead to a more robust and accurate model for portfolio management.

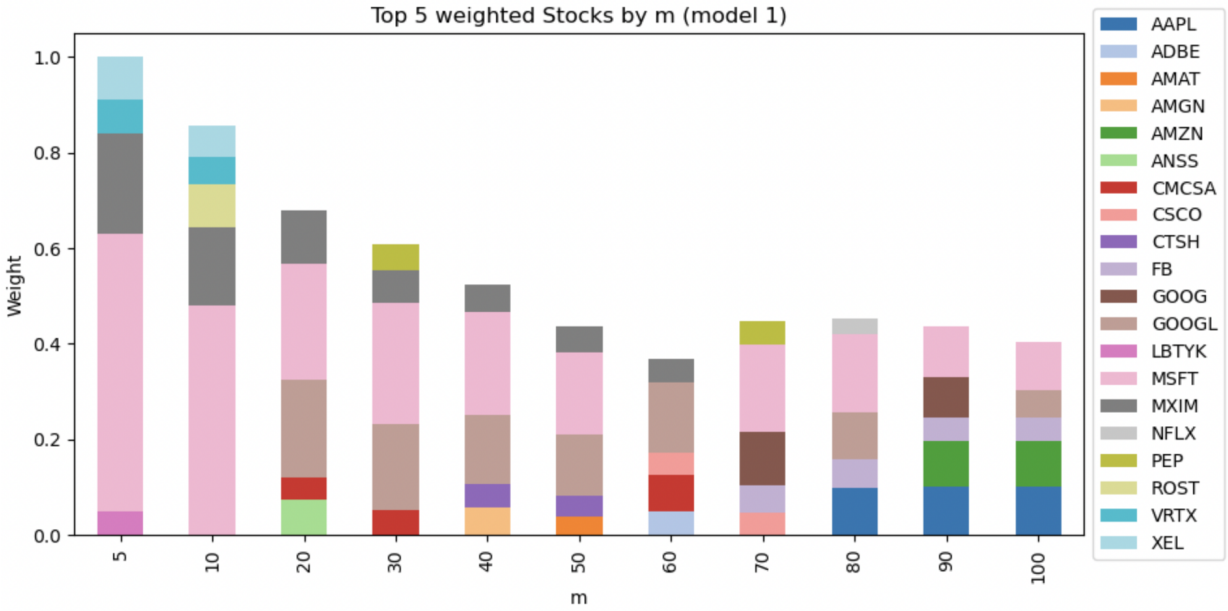


Overall, both model 1 and model 2 exhibit a decreasing trend in in-sample and out-sample errors, and the lowest ending error values at $m = 90$ and $m = 100$ are nearly identical for both models' in-sample and out-sample cases respectively. However, the difference between in-sample and out-sample errors for model 1 is approximately 0.3 to 0.4, whereas for model 2, it is around 0.2 to 0.3, confirming that model 2 performs better. Nonetheless, in cases where almost all stocks are selected, such as at $m = 90$ and $m = 100$, there is no significant performance difference between the two models. Therefore, when considering the substantial difference in computational costs between the mixed model and the binary & continuous model, strategically opting for model 1 could be more advantageous.

Weights

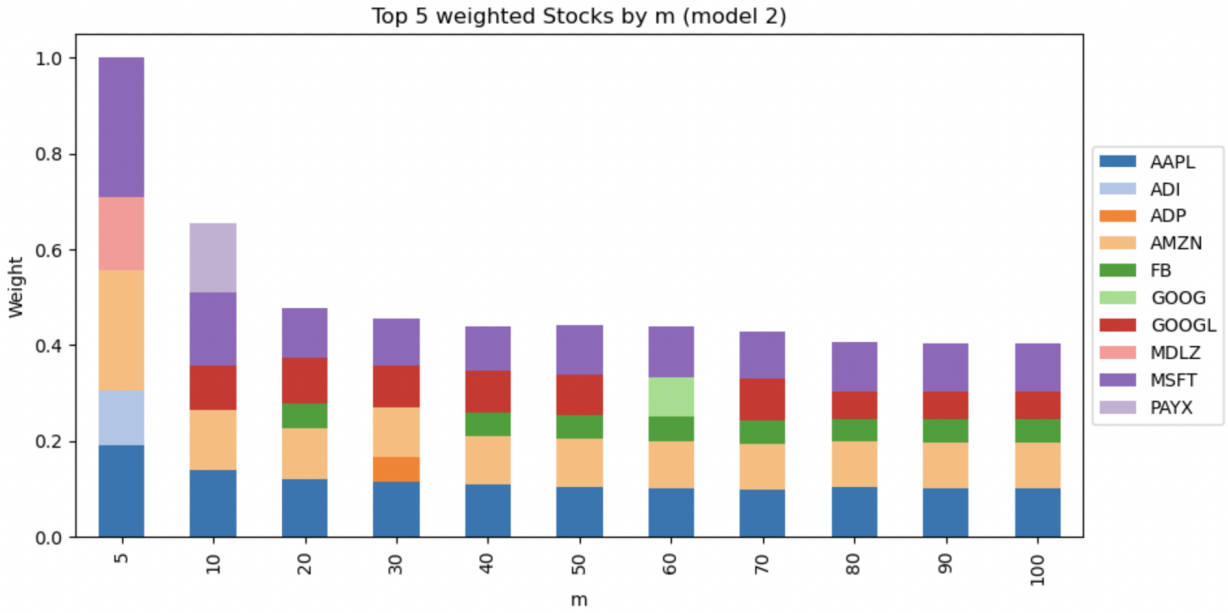
In addition to tracking errors, we also examined whether there are meaningful differences in stock selection between the two models. For effective analysis and visualization, we represented the top five stocks with the highest weights for each value of *m* on a graph.

<model1 - Top 5 weighted stocks >



In the case of model1, out of a total of 55 potential stock candidates, there were only 20 unique stocks selected. This indicates that a few key stocks were repeatedly chosen, resulting in overlapping selections. Among these, MSFT consistently received the highest weight across almost all values of *m*, standing out prominently. This was followed by GOOGL, MXIM, AAPL, AMZN, AAPL, and FB, which were consistently selected across certain intervals.

<model2 - Top 5 weighted stocks >



Conversely, in the case of model2, out of a total of 55 potential stock candidates, only 10 unique stocks were selected, which is 50% of the level in model1. This suggests that model2 focused much more on specific stocks, resulting in less diversification compared to model1. Interestingly, while model1 had significant variance in stock weights, with MSFT being a dominant stock, model2 exhibited relatively even weight distribution. Nevertheless, the selected stocks in both models overlapped by 6 stocks: AAPL, AMZN, FB, GOOG, GOOGL, and MSFT. This confirms that these 6 stocks are key components of our portfolio, regardless of the model used.

4. Conclusion

Through our comprehensive analysis of the two models, it is evident that both model1 (Binary & Continuous) and model2 (Mixed-Integer Programming) have their unique strengths and weaknesses. The decision on which model to choose depends on various factors, such as the desired number of component stocks in the fund, the computational costs, and the tolerance for error.

Model Comparison

When comparing the two models, it is important to consider the specific needs of the fund. If the goal is to minimize tracking errors and achieve a portfolio that closely mirrors the NASDAQ-100 index, model2 is the clear winner. However, if simplicity, flexibility, and lower computational costs are more important, then model1 may be the better choice.

Recommendation

For situations where almost all stocks are selected (e.g., m = 90 and m = 100), there is no significant performance difference between model1 and model2. However, considering the substantial difference in computational costs, we recommend opting for model 1 in such cases.
In cases where fewer stocks are selected, model2 outperforms model1, as indicated by the consistently lower tracking errors. Therefore, for m values less than 90, model2 is the preferred choice.

Future Alternatives

We suggest exploring other optimization techniques that may offer a balance between the simplicity of model1 and the accuracy of model2. Machine learning algorithms, such as neural networks, could be an avenue for future research.

In conclusion, our recommendation is to opt for model2 for m values less than 90, as it provides a more accurate and optimal solution with lower tracking errors. However, for situations where almost all stocks are selected, model1 is the more cost-effective and strategically advantageous option. As we move forward, we should continue to explore other optimization techniques and models to find the most effective and efficient way to manage our fund's portfolio.