Project 2

Participants: Madeleine Cope, Anukul Kumar Singh, Arushi Sethi, Luis Villazon

Introduction

A common passive equity money management strategy is indexing. When indexing, the goal is typically to choose a portfolio (index fund) that has similar movements to that of a market index such as the NASDAQ-100. In theory, a way to make an index fund that accomplishes this goal would be to purchase all the stocks in this index with the same weights that the index provides. However, doing this is quite impractical and costly due to the amount of recalculating this would take on a regular basis. Moreover, a more efficient strategy would be to create an index with a much smaller number of stocks, and thus different weights for each stock, that does an efficient job at tracking the larger index with less complications.

Goal

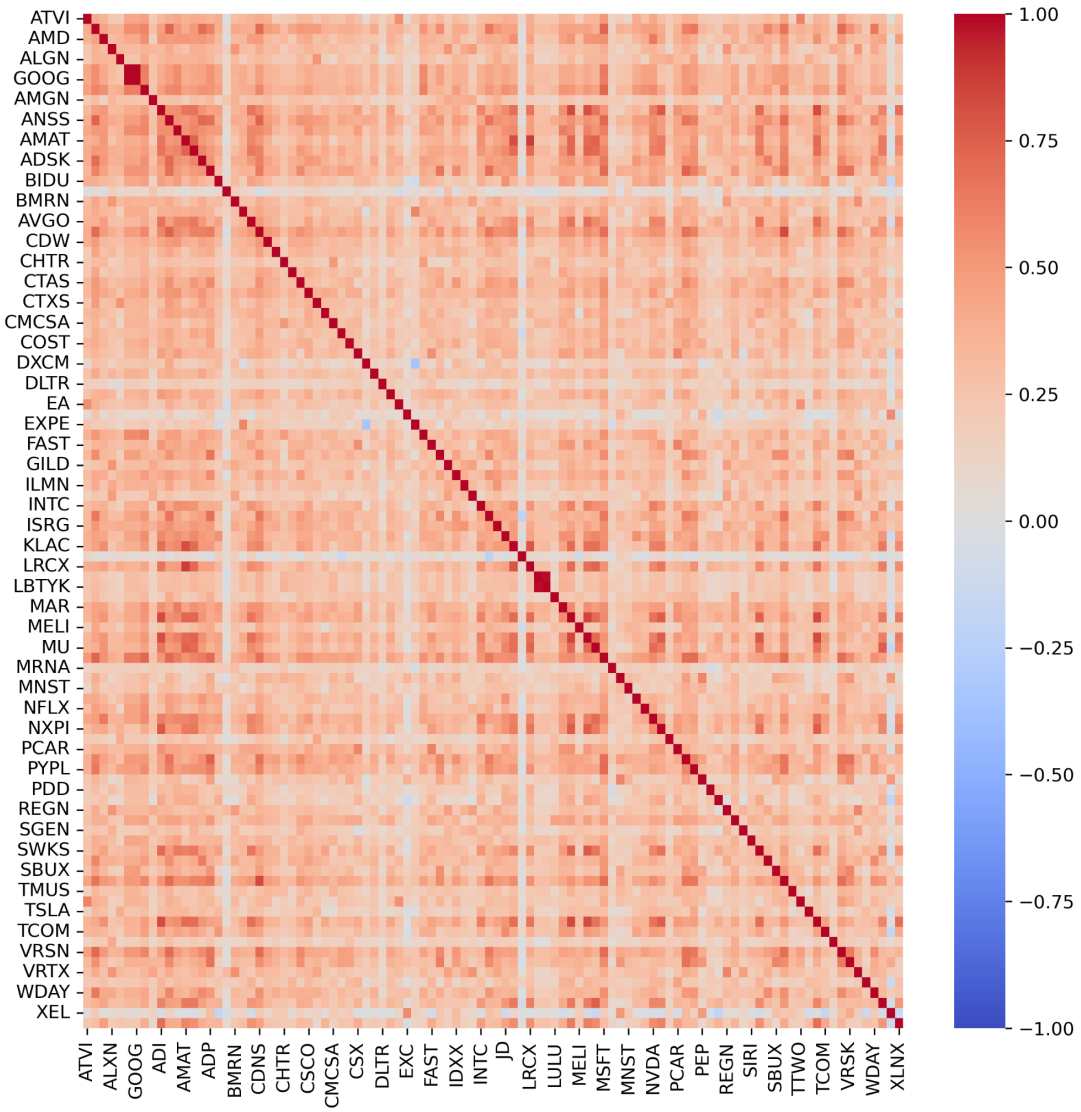
In this project, we've looked into the NASDAQ-100 index returns and its individual stocks from 2019 and 2020 as requested. Keeping in mind our limitations, we've put together a report that explains our analysis, decision-making, and weighted stock choices for the best possible Index Fund to follow the NASDAQ-100 closely. You'll find our code, charts, and some extra helpful information in the report.

Initial Method 1 (Integer + Linear)

Our first step in this process was to find a method to select a smaller number of optimal stocks for our portfolio. To do this, we used an integer program to maximize the similarity between the stocks in the NASDAQ-100 index and the stocks we will pick in our portfolio.

Below is the representation of similarity in stocks using a heatmap. The darker patches show a higher correlation within stocks. Many of the stocks seem to have moderate to high positive correlations with each other, indicated by the predominantly light-to-dark red colors. This suggests that a lot of these stocks tend to move in the same direction. This is common for stocks within the same sector or market.

From the names we can see that these stocks are mainly from the following sectors:
Here's a general breakdown of some of the notable companies from the heatmap:



1. **Technology:**

AMD - Advanced Micro Devices, Inc. (Semiconductors)
GOOG - Alphabet Inc. (Internet services and products, parent of Google)
AMZN - Amazon.com, Inc. (E-commerce and cloud computing)
AAPL - Apple Inc. (Consumer electronics)
MSFT - Microsoft Corporation (Software and hardware)
NVDA - NVIDIA Corporation (Semiconductors)
TSLA - Tesla, Inc. (Electric vehicles and energy)
ADSK - Autodesk, Inc. (Software)
INTC - Intel Corporation (Semiconductors)
NFLX - Netflix, Inc. (Streaming services)

PYPL - PayPal Holdings, Inc. (Online payments)
WDAY - Workday, Inc. (Software)

2. **Biotechnology & Pharmaceuticals:**

GILD - Gilead Sciences, Inc.
MRNA - Moderna, Inc.
REGN - Regeneron Pharmaceuticals, Inc.
VRTX - Vertex Pharmaceuticals Incorporated

3. **Consumer Services:**

MELI - MercadoLibre, Inc. (E-commerce)
EXPE - Expedia Group, Inc. (Travel services)
MAR - Marriott International, Inc. (Hospitality)

4. **Healthcare & Medical Devices:**

ILMN - Illumina, Inc. (Genomics)
ISRG - Intuitive Surgical, Inc. (Robotic surgical systems)

An example of cross sector similarities can be seen in ANSYS and Adobe which makes sense because semiconductors fuel technological companies.

As for the stock selection, the function used to do select similar stocks in the fund is as follows:

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

With this function, we added three constraints to the model as well. Our first constraint selects that only “m” number of stocks be used in our portfolio. Our second constraint ensures that each stock “i” has only one representative stock “j” in the index. Lastly, our third constraint ensures that each stock “i” is represented by stock “j” only if “j” is in the fund. These constraints follow these general formulas:

$$s.t. \sum_{j=1}^{n} y_j = m.$$

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

$$x_{ij} \le y_j \qquad for\ i,j = 1,2,\dots,n$$

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

Where “yj” is a binary decision variable that represents which stocks “j” are in the fund. In addition, for each stock “i” in the NASDAQ index, the binary decision variable “xij” shows us which stock “j” is the best representative of stock “i”. Our code for creating this overall stock selection method is as follows:

```
select_stocks(correlation_matrix, m):
n = correlation_matrix.shape[0]

# Initialize model
model = Model()

# Decision variables
y = model.addVars(n, vtype=GRB.BINARY, name="y")
x = model.addVars(n, n, vtype=GRB.BINARY, name="x")

# Objective
model.setObjective(sum(correlation_matrix.iloc[i, j] * x[i, j] for i in range(n) for j in range(n)), GRB.MAXIMIZE)

# Constraints
model.addConstr(sum(y[j] for j in range(n)) == m, "Constraint_1")

for i in range(n):
    model.addConstr(sum(x[i, j] for j in range(n)) == 1, f"Constraint_2_{i}")

    for j in range(n):
        model.addConstr(x[i, j] <= y[j], f"Constraint_3_{i}_{j}")


model.Params.OutputFlag = 0
# Solve the model
model.optimize()

# Extract selected stocks
selected_index = [j for j in range(n) if y[j].X > 0.5]
selected_stocks = [correlation_matrix.columns[i] for i in selected_index]
```

Our second step in the process for creating our portfolio was to establish a method to attribute weights to the “m” number of stocks in it. This method is pivotal in ensuring that our portfolio closely tracks the NASDAQ-100 index. The primary challenge here is to balance the returns of individual stocks in such a way that the aggregate return of our portfolio is as close as possible to the return of the NASDAQ-100 index.

To achieve this, we introduce a series of auxiliary variables, zt, for each time period t. These variables are designed to capture the absolute error between the return of the NASDAQ-100 index at time t and the return of our portfolio at that same time. Our goal is to minimize the cumulative absolute error over all periods. Mathematically, this is represented as:

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

To ensure “zt” captures the positive difference between the index return and the portfolio return at time t:

$$z_t \geq q_t - \sum_{i=1}^{m} w_i * r_{it}$$

Where “qt” represents the return of the NASDAQ-100 index at time period “t”. “Wi” is the weight of the stock in our portfolio and “rit” is the return of stock “i” chosen from one of the stocks in the above function.

This constraint ensures that “zt” captures the positive difference between the index return “qt” and the weighted return of our portfolio. If the portfolio's return is less than the index return at time t, this difference will be positive, and “zt” will capture that magnitude.

On the other hand, to ensure “zt” captures the negative difference:

$$z_t \geq -q_t + \sum_{i=1}^{m} w_i * r_{it}$$

Conversely, this constraint captures situations where the portfolio's return exceeds the index return at time t. If this difference is negative, “zt” will capture its absolute value. Together, these two constraints guarantee that “zt” always captures the absolute difference between the index return and the portfolio's return, regardless of whether the portfolio underperforms or outperforms the index.

Furthermore, the weights must constitute a valid portfolio distribution. For this purpose, a constraint should be added that ensures the cumulative sum of the weights of the stocks in the portfolio equals 1.

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

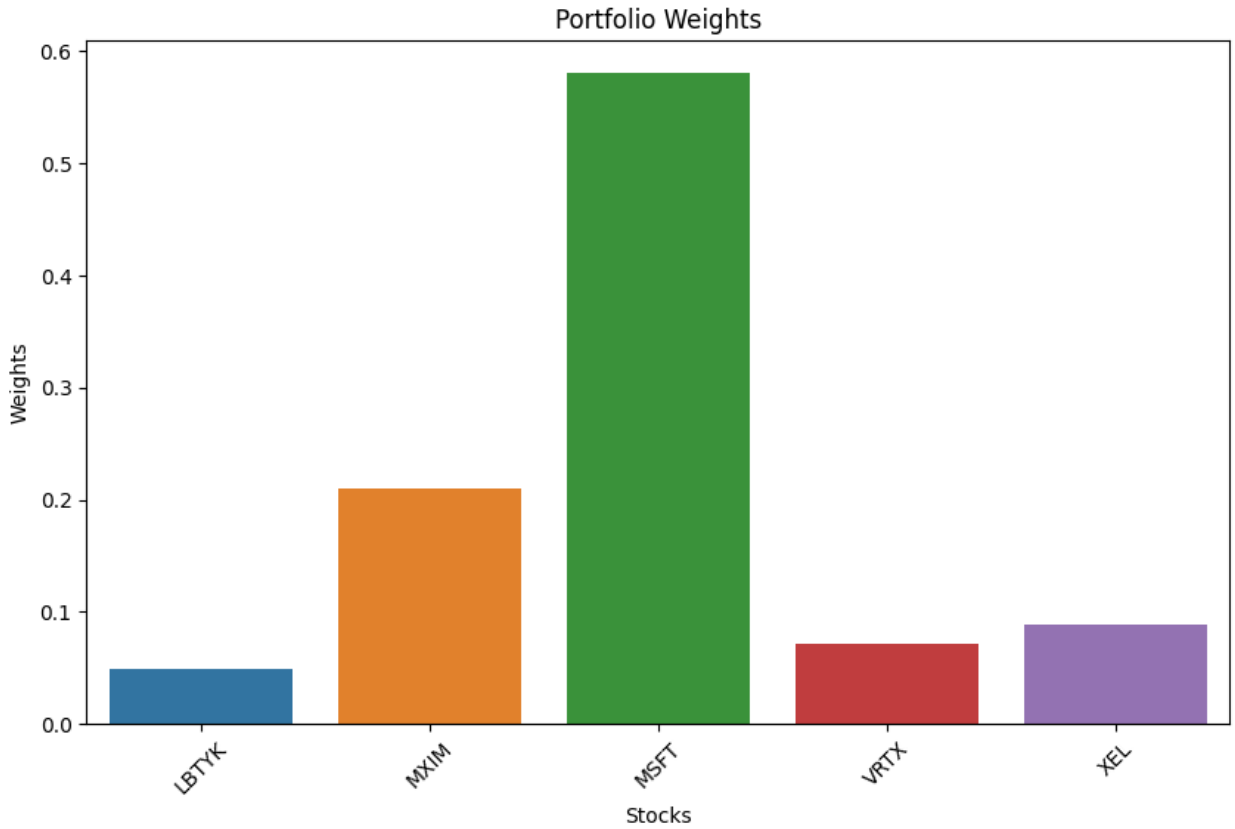
This last constraint ensures that no stock in the portfolio has a negative weight, which would imply short selling. By keeping all weights non-negative, the portfolio remains long-only.

The model previously described has been operationalized through the following Python code:

```python
def optimize_weights(m,selected_indices, returns_2019, index_returns_2019):
    selected_indices = select_stocks(correlation_matrix,m).loc[0, 'Indices']
    selected_returns = returns_2019.iloc[:, selected_indices]
    # Assuming the index returns are in a column named "NDX" in the original data_2019 dataframe
    index_returns_2019 = data_2019["NDX"].pct_change().dropna()
    a, T = m, selected_returns.shape[0]

    model_weights = gp.Model("PortfolioOptimization")
    w = model_weights.addVars(a, vtype=GRB.CONTINUOUS, name="w")
    z = model_weights.addVars(T, vtype=GRB.CONTINUOUS, name="z")

    model_weights.setObjective(gp.quicksum(z[t] for t in range(T)), GRB.MINIMIZE)

    for t in range(T):
        model_weights.addConstr(z[t] >= index_returns_2019.iloc[t] - gp.quicksum(w[i] * selected_returns.iloc[t, i] for i in range(a)))
        model_weights.addConstr(z[t] >= -index_returns_2019.iloc[t] + gp.quicksum(w[i] * selected_returns.iloc[t, i] for i in range(a)))

    model_weights.addConstr(gp.quicksum(w[i] for i in range(a)) == 1)

    for i in range(a):
        model_weights.addConstr(w[i] >= 0)

    model_weights.Params.OutputFlag = 0
    model_weights.optimize()

    performance = model_weights.objVal
```

Method 1 is like a smart shopping strategy for stocks. Instead of buying everything in the store (the NASDAQ-100), we're picking just a few items that give us the best value for our money. By selecting a smaller group of stocks, we can manage them easier and likely save on costs. The way we choose and weigh these stocks is to make sure our smaller basket moves similarly to the entire store's offerings. It's like making sure our mini-basket of items reflects the broader trends of what everyone else is buying. For investors, this is good news. They can get the benefits of investing in the big NASDAQ-100 without buying all the stocks. It's simpler, might be cheaper, and still aims to give the same kind of performance. This approach can make our fund stand out in a crowded market and attract smart investors looking for efficient options.

Using this approach, we started by trying to find the top five stocks that best match the 2019 NASDAQ-100 index. We set our sights on picking just five and, using our method, found that “LBTYK”, “MXIM”, “MSFT”, “VRTX”, and “XEL” were our best bets. These stocks were the closest in behavior to the bigger NASDAQ-100 list. We also figured out how much of each stock we should have in our mix, and here's how it turned out:



As we can see here, for our portfolio to have a minimized level of error, we should invest the most in “MSFT” and the least in “LBTYK”. In doing so, our error term will be decreased to 0.789 for the 2019 NASDAQ-100 index.

Error Terms at m=5 (1)

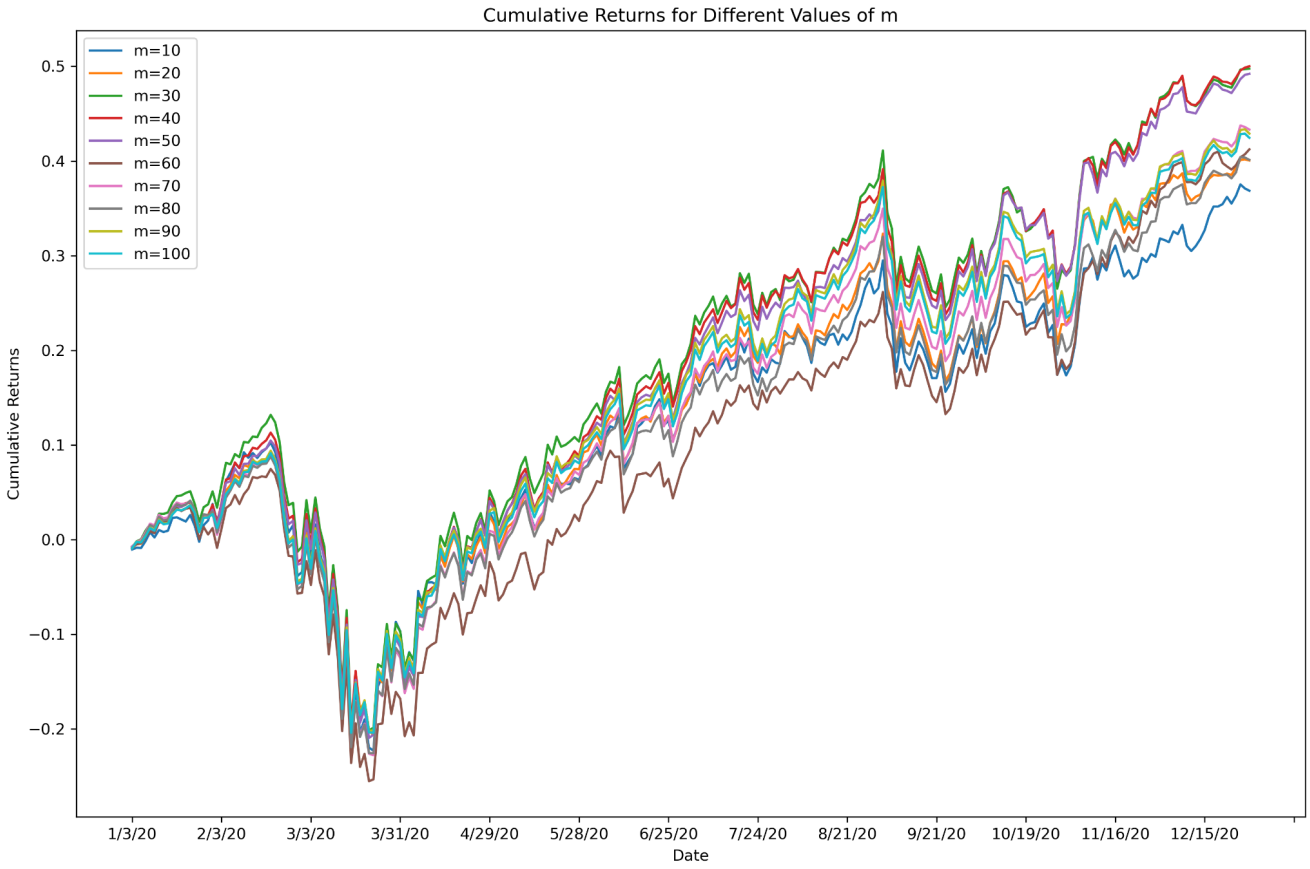
2019 Error	2020 Error
0.789	1.112

Knowing this, we wanted to compare our accuracy in sample performance to that of an out sample performance using 2020 NASDAQ-100 index information. With this, we used our same weights as calculated above with returns information from 2020. In doing so, our error term increased to 1.112 for our out of sample performance. Because this a large increase, it seems that our portfolio does not track the NASDAQ-100 index from 2020 quite as well as it did in 2019. This is likely due to various market shifts that occurred from the COVID-19 pandemic, however, it is still important for us to see if this trend still persists with modifications to the number of stocks in our portfolio.

To test this, we created 10 new portfolios with m=10,20,...90,100 stocks in them to see if the number of stocks used helped minimize error in our data and would help us better predict trends in the NASDAQ-100 index. To do this, we used the same tools as described above simply using different m values for each portfolio. In doing so, we calculated preferred stocks and weights on 2019 data, and then used those stocks and weights on the 2020 data in order to see our differences in error. A sample of our results were as follows:

	Number of stocks	Indices	Stocks	Weights	Performance (Insample)	Performance (Outsample)
0	10	[0, 4, 40, 53, 56, 59, 63, 79, 94, 98]	[ATVI, ALGN, EXPE, KHC, LBTYK, MXIM, MSFT, ROST, VRTX, XEL]	[0.04420008162515515, 0.02495587429403977, 0.02150098184560955, 0.023389314412142398, 0.03783494391598981, 0.15698580809468035, 0.4910305234120523, 0.0911518044588584, 0.046216031808782046, 0.06269463613269025]	0.701218	1.102404
1	20	[0, 4, 5, 10, 15, 17, 30, 36, 40, 51, 53, 56, 59, 63, 64, 72, 76, 91, 94, 98]	[ATVI, ALGN, GOOGL, ANSS, ADP, BIIB, CMCSA, DLTR, EXPE, JD, KHC, LBTYK, MXIM, MSFT, MRNA, PCAR, PDD, ULTA, VRTX, XEL]	[0.0246128451625659, 0.012773472041639327, 0.2043523158854835, 0.0744541929495624, 0.01941354139204288, 0.0065169485696624846, 0.04529662386544985, 0.01995678968025231, 0.007310069699686221, 0.03937154106780161, 0.03618476446490949, 0.0033070823213016036, 0.11154798202438464, 0.24394752145679896, 0.0052746982892368085, 0.0449096090976265, 0.01162030700704926, 0.013698175980572167, 0.03491081794021952, 0.040540701103754455]	0.478836	0.899598
2	30	[0, 1, 5, 12, 15, 17, 28, 30, 34, 35, 36, 40, 46, 51, 53, 56, 57, 59, 63, 64, 66, 72, 75, 76, 77, 86, 88, 91, 94, 98]	[ATVI, ADBE, GOOGL, AMAT, ADP, BIIB, CTXS, CMCSA, DXCM, DOCU, DLTR, EXPE, ILMN, JD, KHC, LBTYK, LULU, MXIM, MSFT, MRNA, MNST, PCAR, PEP, PDD, QCOM, TMUS, TSLA, ULTA, VRTX, XEL]	[0.016889901491185348, 0.031105451684456498, 0.18063401599195625, 0.03814396109159537, 0.0, 0.002714304247327306, 0.0, 0.05258331223545117, 0.024380141367161264, 0.0, 0.009086141155451102, 0.03287368959709584, 0.0024986343637322037, 0.02776205562219087, 0.007674981942931485, 0.015617808066050997, 0.01838412441856238, 0.06751392016117207, 0.2533016813412306, 0.00499871510589452, 0.0, 0.0370562161852007, 0.05328053052615635, 0.00272284331584433, 0.020303301653445422, 0.004600906065708365, 0.03379896623609255, 0.011132161512151094, 0.03666205666709325, 0.01428017795486262]	0.418015	0.769110

Given below are trends for cumulative returns of our stocks for 2020 differentiated for different values of m.



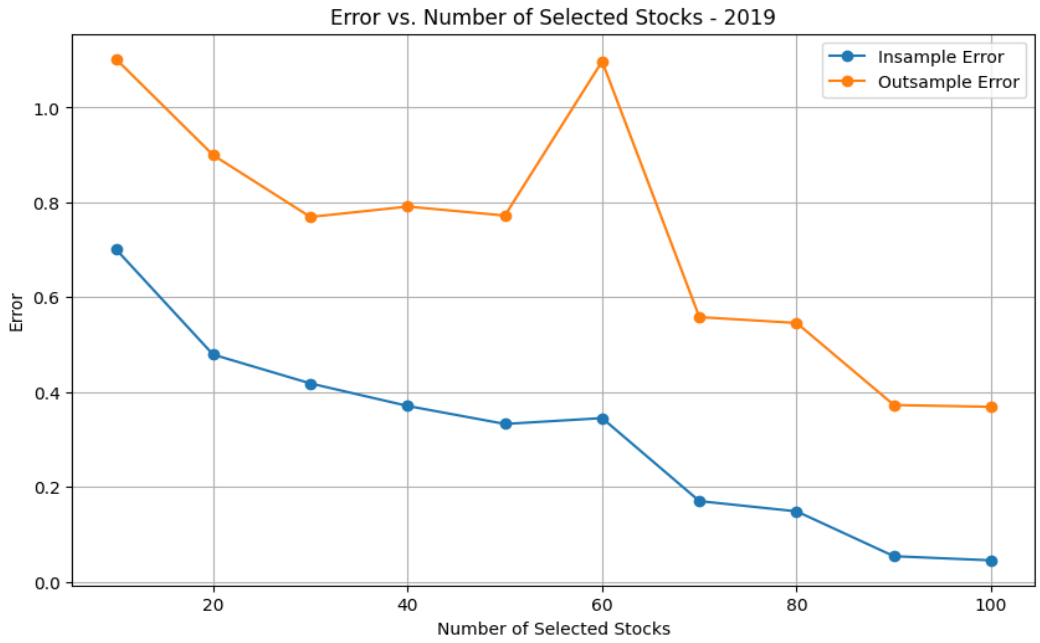
Our full list of error terms is as follows:

Error Terms for Method 1 over all M

M	In sample error (2019)	Out of sample error (2020)
5	0.789	1.112
10	0.701	1.102
20	0.479	0.900
30	0.418	0.769
40	0.371	0.791

50	0.333	0.772
60	0.345	1.100
70	0.170	0.556
80	0.148	0.546
90	0.054	0.372
100	0.045	0.369

The information regarding error for each value of m is as follows:



The provided graph depicts the relationship between error rates and the number of selected stocks for the year 2019. The data is segmented into in-sample and out-sample errors. Initially, as the number of selected stocks rises to 20, both error types see a decline. Particularly, the in-sample error consistently maintains a value lower than its out-sample counterpart, hinting at a better model performance on training data.

From 20 to 60 stocks, while the in-sample error continues its downward trajectory, the out-sample error exhibits volatility with a pronounced peak at around 60 stocks. Beyond this point, in-sample error stabilizes, whereas the out-sample error undergoes

fluctuations. Notably, the lowest error rates for both categories seem to emerge between the 20-40 and near the 80-stock mark.

The reason for this sudden spike in our out of sample data is likely due to overfitting in the 2019 dataset. If the data is too attuned to the original dataset, sudden spikes, such as this, can occur in the test sample. This makes it seem as though the model is not very well fit to the test data, but in actuality, it is likely too fit to the training data due to the weight calculations.

In addition, some of these differences in error could be due to the spread of the sectors that each company's stock is in. For instance, there seems to be a wide variety of industries represented in the spread of the chosen stocks for each "m", however there does also seem to be a very slight skew towards technology related companies. Moreover, if the training data was fit more to stocks in one particular sector, this could have shifted our results if the most popular industries shifted in 2020.

Thus, this information still shows us that the model is still not particularly robust at following the trends in 2020 as compared to 2019, even at multiple values of "m". This tells us that, in the investing world, we've got to be ready to switch things up based on what's happening around us. Sticking to just one game plan, might not always be the best bet. It's a good reminder to stay flexible and keep an eye on the changing market. Given these insights, we'll need to experiment with a new method to see if we can find a way to better follow the NASDAQ-100 trends in 2020 using our learnings from 2019 data.

Second Method 2 (Mixed Integer Program)

Because there were still significant error differences between the 2019 and 2020 portfolios we created, we wanted to test a new method to create portfolios for m=5,10,20,30…90,100 stocks. For this method, we chose to ignore our first integer program used to select stocks and instead modify our weight calculation linear program to be an MIP that constrains that number of non-zero weights to be an integer. To do this, we first modified the weight selection method to replace "m" with "n" in order to optimize over all weights as follows:

$$\min_{w} \sum_{t=1}^{T} |q_t - \sum_{i=1}^{n} w_i r_{it}|$$

Then, we defined binary variables "y1", "y2",..."yn" and added constraints to force "wi"=0 if "yi"=0 using a big M constraint. Here, our big M was equivalent to 1, seeing as the

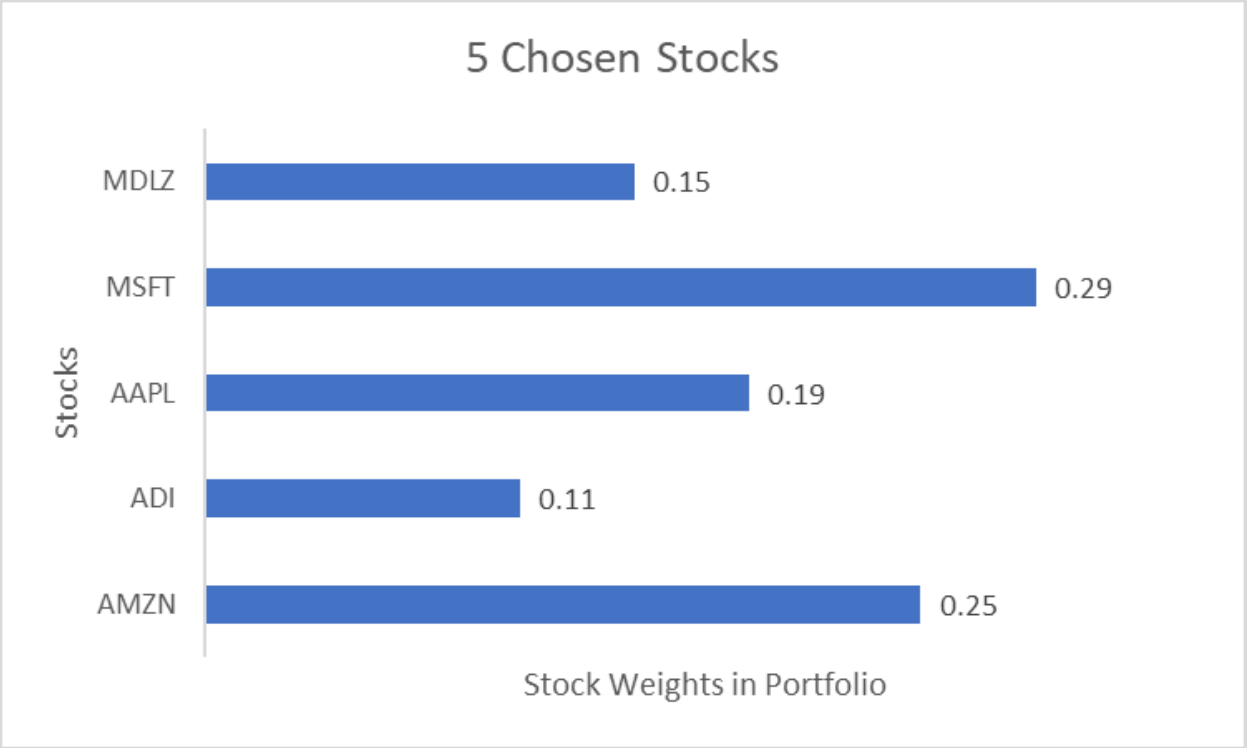
sum of all weights needed to sum to this amount anyway. In addition, a final constraint that the sum of all “y”s needed to be equal to “m” was added.

Method 2 offers a tighter focus in portfolio creation, aiming to get more from fewer stocks. This strategy can lead to higher potential returns but also comes with greater risks due to its concentrated nature. It's a reminder for investors that in finance, finding the right balance between risk and reward is crucial. This approach requires keen market observation and readiness to adapt based on performance results.

Our code for the program of this method is as follows:

```
#making second method
def second_mod(m):
  method2=gp.Model(env=env)
  method2.Params.OutputFlag=0

  #stock is either selected or not
  var1=method2.addMVar(n, vtype='B')

  #weight of stocks
  var2=method2.addMVar(n)

  #store difference in NASDAQ index and portfolio index
  var3=method2.addMVar(p)

  #objective

  method2.setObjective(gp.quicksum(var3[i] for i in range(p)), gp.GRB.MINIMIZE)

  #constraints

  #weight of all choices=1
  method2.addConstr(gp.quicksum(var2[i] for i in range(n))==1)

  #set number of stocks
  method2.addConstr(gp.quicksum(var1[i] for i in range(n))==m)

  #big M
  method2.addConstrs(var2[i]<=var1[i] *bigM for i in range(n))


  #more constraints

  #absolute value
  method2.addConstrs((var3[i]>= index_return.iloc[i]-gp.quicksum(var2[j]*returns_2019.iloc[i,j] for j in range(n))) for i in range(p))

  #another absolute value
  method2.addConstrs((var3[i]>= gp.quicksum(var2[j]*returns_2019.iloc[i,j] for j in range(n)) - index_return.iloc[i]) for i in range(p))
```

Because this problem is difficult for our software to solve, each of our 11 portfolios of different m values was run for an hour each in order to find optimal solutions. In doing so we initially found different stocks and weight values chosen for our first run through a portfolio of 5 stocks. The weights and value were as follows:



As we can see, different stocks were chosen in 2019 in comparison to 2020. Here, we see that the best stocks to use in order to maximize the similarity in trends between our portfolio and the NASDAQ-100 index are “AMAZN”, “ADI”, “AAPL”, “MSFT”, and “MDLZ”. In addition, in comparison to 2019, our error terms were as follows for this new method:

Error Terms at m=5 (2)

2019 Error	2020 Error
0.499	0.777

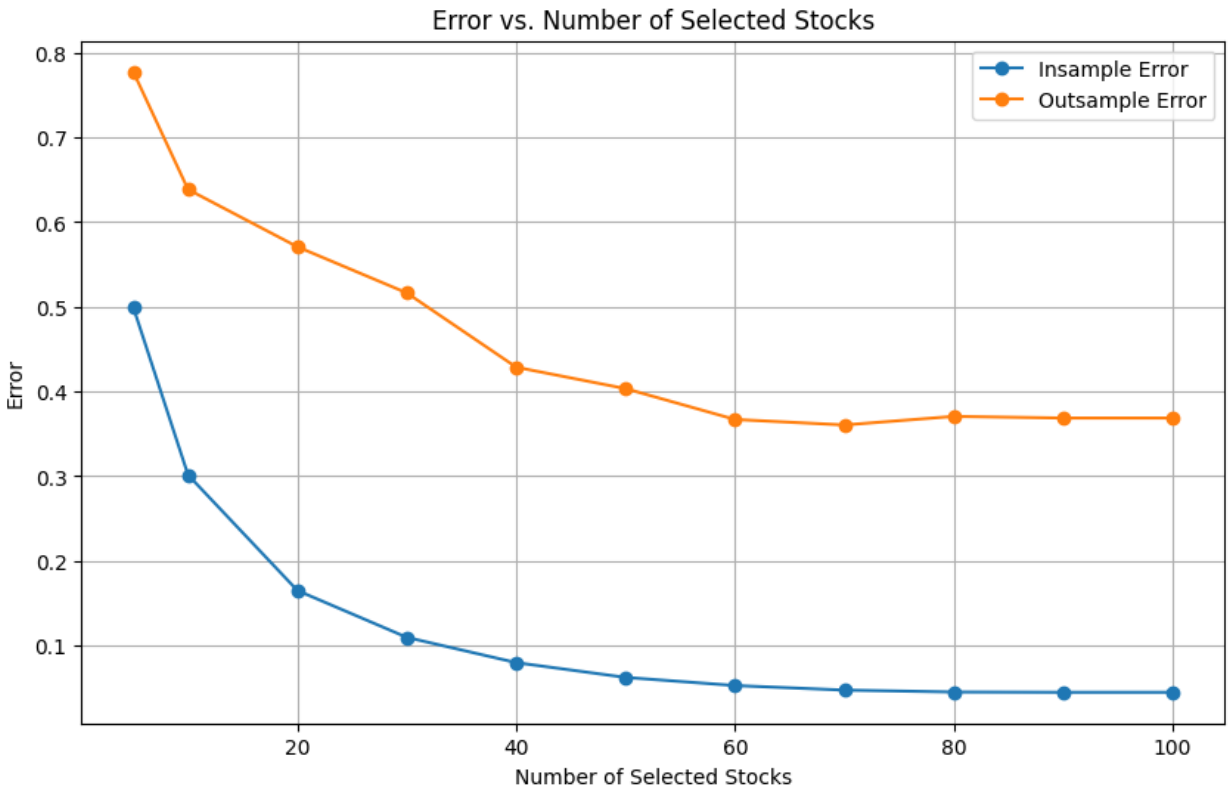
As we can see, this new method using an MIP significantly reduced the level of error we are seeing in both the in sample performance and the out of sample performance. However, there is still a significant difference in the error between in the in sample and out of sample performance for 2019 and 2020. Thus, we used the program to still run an error analysis over m=5,10,20,30…90,100 stocks to see if larger numbers of stocks reduce the level of error over time. An example of the results is as follows in the code snippet below:

	Number of stocks	Stocks	Weights	Performance (In sample)	Performance (Out sample)
0	5	['AMZN', 'ADI', 'AAPL', 'MSFT', 'MDLZ']	[0.25012259799984415, 0.11375807105291444, 0.1916922061629509, 0.2898692800135151 4, 0.15455784477077586]	0.499259	0.777362
1	10	['GOOG', 'AMZN', 'AAPL', 'CHTR', 'FB', 'GILD', 'MSFT', 'PCAR', 'TXN', 'VRSK']	[0.0916577050983396, 0.12775583419456224, 0.14548359786649365, 0.053330673924296725, 0.06453430070092206, 0.05782932147065084, 0.2148537933243309, 0.06732189322720131, 0.10201402754175896, 0.07521885265144371]	0.301498	0.638550
2	20	['ADBE', 'GOOG', 'AMZN', 'AAPL', 'AVGO', 'CSCO', 'CMCSA', 'FB', 'ILMN', 'INTC', 'KLAC', 'MSFT', 'NFLX', 'ORLY', 'PAYX', 'PEP', 'ROST', 'TXN', 'VRTX', 'WBA']	[0.06132477035577345, 0.0920113291902568, 0.10575174936081824, 0.09995754597023827, 0.04124980346029167 4, 0.0289139899553455, 0.046453228283709375, 0.05801960798282517, 0.02058805112849524, 0.04403580445473051 5, 0.026294835026418877, 0.09417385697736941, 0.024254109248858922, 0.022802351224962342, 0.045931284637530316, 0.03978260098828672, 0.04466799857974514, 0.04449381071531414, 0.030587245335733357, 0.028706027123296392]	0.165028	0.570820
3	30	['ADBE', 'GOOGL', 'AMZN', 'AMGN', 'AAPL', 'ADP', 'BIIB', 'BMRN', 'BKNG', 'AVGO', 'CHTR', 'CSCO', 'CMCSA', 'COST', 'CSX', 'FB', 'GILD', 'ILMN', 'INTC', 'LRCX', 'MU', 'MSFT', 'MDLZ', 'NFLX', 'PAYX', 'PYPL', 'QCOM', 'SBUX', 'TXN', 'WBA']	[0.033434489290841175, 0.0847507266678296 9, 0.10612321346639482, 0.011413431782997474, 0.11603650502057371, 0.03119043578148157, 0.007388370237603 53, 0.010157029793084369, 0.011685921704409847, 0.020990270421464 28, 0.016421499622241955, 0.03517326928791517, 0.026733620194036883, 0.030683999212714 22, 0.02182421574071606, 0.051207407742366 65, 0.027834448018038752, 0.012999141892947331, 0.036867481778458 37, 0.011931118488439637, 0.017476804462097352, 0.102349965939071 43, 0.022075685969708, 0.022655613858619066, 0.032760024922918 05, 0.02499606201095837, 0.014360292590061514, 0.014198307605446074, 0.028246856731444572, 0.016026033139568057]	0.109731	0.516360

Our results are also shown in the table and graph below:

Error Terms for Method 2 over all M

M	In sample error (2019)	Out of sample error (2020)
5	0.499	0.777
10	0.301	0.639
20	0.165	0.571
30	0.110	0.516
40	0.080	0.429
50	0.062	0.403
60	0.053	0.366
70	0.047	0.361
80	0.045	0.371
90	0.045	0.369
100	0.045	0.369



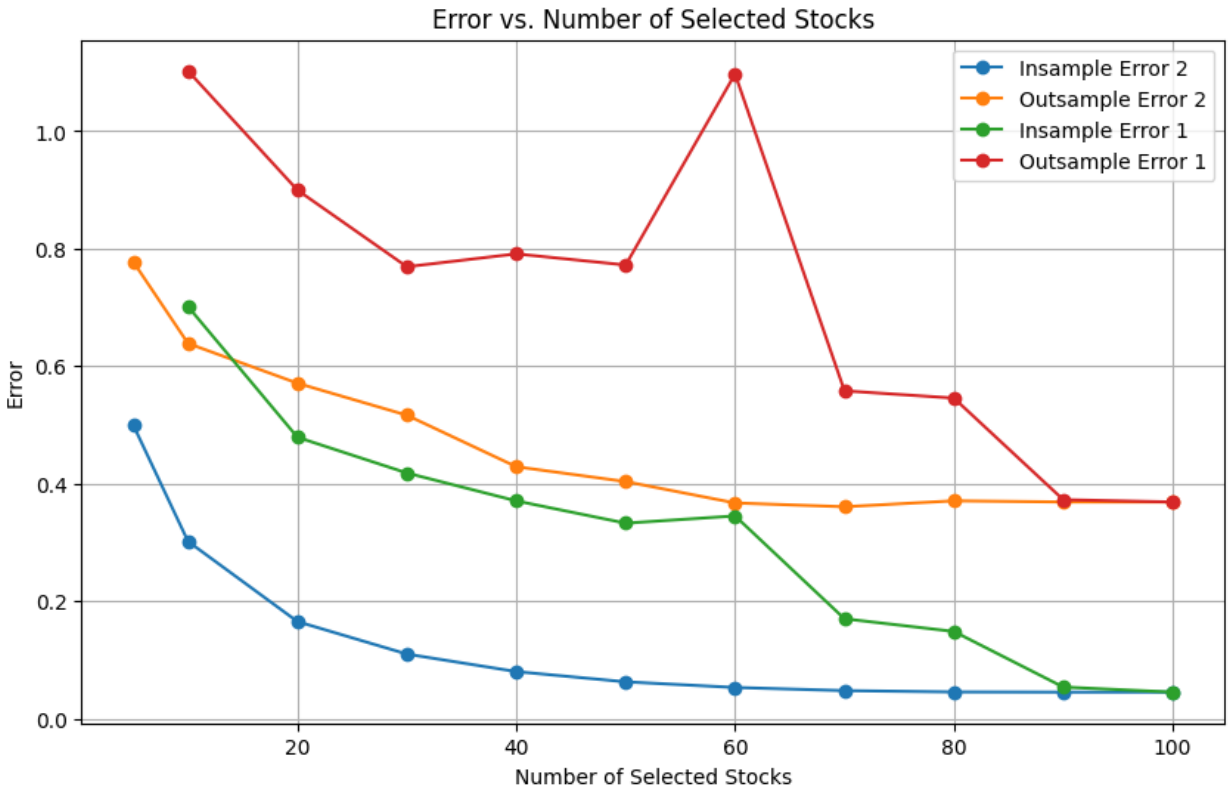
As we can see from the graph and values above, generally speaking, this method produces are more stable decline in error for both in and out of sample performance. Specifically, we can see that the insample performance decreases to a very small value below 0.1 as more stocks are added to our portfolio. Our out of sample performance follows a similar trend, however, its error term as more stocks are added to the portfolio converges to a value around 0.35 rather than below 0.1. Moreover, there still seems to be a difference in the level of predictability of 2020 NASDAQ-100 trends in comparison to 2019, however the error in this is significantly reduced.

In addition, we also see here that the slopes of these curves are much more stable and standardized. This showcases the effects of using the weights differently in the MIP to more accurately track the NASDAQ-100. Moreover, this keeps the training data from being over fit and keeps there from being a spike in the error at m=60 like we saw in the method 1 graph. The MIP is beneficial here because weight optimization is a part of stock selection here rather than doing it in two different steps. Hence there is no overfitting. Moreover, the Big M constraint ensures the continuous values of weights are not calculated for stocks that are not selected.

Lastly, an explanation for this fit could be due to a more similar skew of industries represented in each portfolio. For instance, in method 1, there was a large variety of

stocks represented in each portfolio. However, there seems to be an even stronger skew towards the technology industry in the selections using method 2. Because the trendlines for both 2019 and 2020 seem similar, this means that the MIP, using its different weight function, may have been able to maintain specificity over both years. This showcases that similar industries' stocks were chosen over both years using this method. However, if there is a general skew using this method, this could pose issues in creating a diverse portfolio. Moreover, this method seems to track the NASDAQ-100 well, but taking a closer look at the chosen stocks, could pose issues in future years for consumers trying to invest diversely and safely.

Final Analysis and Recommendation



The graph above shows both our error trends in 2019 (in sample) and 2020 (out of sample) error using methods 1 and 2. Here, we can see the following trends:

Insample Error: In both methods, the insample error starts at a relatively high value when the number of selected stocks is low. As the number of selected stocks increases, the insample error decreases steadily and then remains fairly constant or slightly fluctuates.

Outsample Error: The outsample error exhibits a more variable behavior. Initially, it starts high, similar to the insample error, but decreases at varying rates. In the first method, there's a noticeable spike around the 60-80 stocks range (potentially due to overfitting) before it drops again. In the second graph, the outsample error steadily decreases and then levels off, showing a more stable pattern after a certain number of stocks.

The results from the second method, the Mixed Integer Program, stand out. Not only does it do a solid job in keeping errors low for both in and out of sample data, but it also shines when we look at the tricky year of 2020. Even with all its unpredictability, this method brings the 2020 errors close to what we saw with the 2019 data using the first method. This shows that we've found a more reliable way to follow the big trends of the NASDAQ-100 index, giving investors a clearer path and more confidence in their investment choices.

***Recommendation*:** In terms of a recommendation from an investment or portfolio optimization perspective, it might be prudent to consider a number of stocks where both insample and outsample errors are low, ensuring a robust model fit on unseen data. In the context of these graphs, it would be in the range where both errors are minimized and stabilized. Moreover, given the trends we are seeing we recommend using the Mixed Interger Program method (Method 2) to select proper stocks for a portfolio seeing as it tracks the NASDAQ-100 index the best, and with minimal error. Using this, we also recommend choosing a number of stocks “m” between 40-60. This is because using the MIP we see a did in error that stabilizes at these numbers of stocks.

However, if cost and efficiency are also being considered, method 2 may be slightly more cumbersome seeing as the program takes a long time to run. Moreover, if time and efficiency are strongly preferred, method 1 may be more suitable.

In conclusion, while both strategies have their merits, Method 2, with the Mixed Integer Programming, stands out as a potential game-changer for portfolio managers and investors. Its ability to consistently track the NASDAQ-100 index, especially in a challenging year like 2020, showcases its potential to offer stability in volatile markets. From a business standpoint, adopting such a method could mean enhanced trust from clients, better investment outcomes, and a competitive edge in the marketplace. However, like any financial strategy, it's essential to continuously review and test it against real-world scenarios to ensure it remains the best fit for the ever-evolving market landscape.

Summary:
To create a portfolio mirroring the NASDAQ-100 index, our objective was to identify a select group of stocks and weights, ensuring efficiency without incurring excessive costs or complexity.

To achieve this, we explored two distinct strategies. First, we utilized the Integer and Linear program method (Method 1) to align closely with the NASDAQ-100 movements while curbing discrepancies. We then delved into the Mixed Integer Program (Method 2) targeting similar objectives but employing a different computational technique.

Our findings indicated that Method 2 was more effective in reducing errors. Furthermore, for those seeking an ideal portfolio composition, our data suggests focusing on a range of 40-60 stocks, as the error rates level off within this range.