Optimization Project 2: Index Fund Creation

Group Members:

Rianna Patel (rnp599)
Juhi Patel (jpp2464)
Samarth Mishra (sm79247)
Sreekar L (sl54387)

Introduction

Equity management is the process by which a company tracks and manages the ownership of the company in relation to stakeholders. They can mostly be categorized into strategies that are 'active' or 'passive'. A common passive strategy is called indexing (index fund) where the portfolio of stocks mirrors the general movement of the market index. The simplest approach would be to totally mimic the market index by purchasing all the stocks in the index having the same weights as that in the index. However, this isn't the most efficient approach and can get quite expensive.

In this report, we talk about a more practical and cost-effective way of creating an index fund using m stocks in the portfolio that closely follows the market index. We first identify the m stocks to be included and subsequently assign optimal weights to them so as to closely follow the market index in the most efficient way. Our analysis also covers how changing the number of stocks in our portfolio will impact the tracking error.

We aim to track the NASDAQ-100 index using our portfolio of m stocks and used the daily prices of the NASDAQ-100 index as well as the component stocks in 2019 to create our optimization models and check for the portfolio performance in the year 2020.

Methodology

We have selected the stocks and calculated their optimal weights using two approaches -

Approach 1:

In this approach, we select the best m stocks which are representative of the entire index by formulating a Binary Integer programming problem, and then for these selected stocks we find the optimal weights by solving a linear programming problem.

Step 1: Stock Selection

In order to find the best m stocks that are representative of the index, we solve the following Integer programming problem -

Objective function:

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

Here, we want to maximize the similarity between the stocks in the fund (n) and the stocks we want to select.

Decision Variables:

X_{ij} : (Binary variables) Represents if stock i is similar to stock j or not. Since we have 100 stocks, there will be a total of 100*100 = 10000 such variables for every i and j.
y_i : (Binary variable) Represents whether or not to select the stock i
ρ_{ij} = similarity metric between stock i and j. We define this by finding the correlation between the daily returns of stock i and j

Constraints:

1. Select exactly m stocks (1 constraint)

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

2. Stock i has only one representative stock j (Total constraints= n)

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

3. Stock i is best represented by stock j only if j is in the fund (Total constraints= n^2)

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

After solving this Binary integer problem on Gurobi we get the stocks we selected.

Step 2: Calculating weights for the stocks

We calculate the weights for the stocks by finding the difference between the summation of the return of the index at time period t and the summation of the return of the stock multiplied by the weight of the stock in the portfolio.

Objective function:

We formulate a linear program to identify the optimal weights to minimize the difference in returns between the index and portfolio (which are optimally chosen by considering maximum similarity with index stocks). The objective function can be written as:

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

Decision variables:

R_{it} : return of stock i in the portfolio at time t
w : weight of stock i in the portfolio
q_t : Return of the index at time period t

Constraints:

1. Sum of all the weights should be 1 (1 constraint)

$$s.t. \sum_{i=1}^{m} w_i = 1$$

2. We convert the absolute function into normal summation and add extra constraints.

   Our objective function now becomes a normal sum that we want to minimize -
   $y_1 + y_2 + y_3 + y_4 + \ldots . y_t$ - We'll have t terms - where t represents the number of time periods

   Additional constraints -

   $y_t > q_t$ - sum(w_i x r_{it}) and
   $y_t < q_t$ - sum(w_i x r_{it})

   Total number of such constraints: 2*(no. of periods of returns) as given in CSV file

3. All the weights should be greater than zero (m constraints)

$$w_i \geq 0.$$

After solving this Linear programming problem we arrive at the optimal weights for the stocks in our portfolio.

Code Snippet for our Objective Function and Constraint Matrices for Approach 1:

```
# Initializing and defining A, b, sense and obj matrices that go into into the optimal weights model
obj = np.concatenate([np.ones(no_of_periods),np.zeros(no_of_stocks_in_portfolio)])
A = np.zeros(((2*no_of_periods+no_of_stocks_in_portfolio+1),len(obj)))
b = np.concatenate([np.zeros(2*no_of_periods+no_of_stocks_in_portfolio),np.ones(1)])
sense = ['>']*(2*no_of_periods+no_of_stocks_in_portfolio)+['=']

for i in range(no_of_periods):
    A[2*i,i] = 1
    A[2*i+1,i] = 1
    for j in range(no_of_stocks_in_portfolio):
        A[2*i,(no_of_periods+j)] = df_return_stocks_in_portfolio.iloc[i,j]
        A[2*i+1,(no_of_periods+j)] = -df_return_stocks_in_portfolio.iloc[i,j]
    b[2*i]=stocks_2019_returns.iloc[i,0]
    b[2*i+1]=-stocks_2019_returns.iloc[i,0]
for k in range(no_of_stocks_in_portfolio):
    A[2*no_of_periods+k,no_of_periods+k] = 1
A[-1,:]=np.concatenate([np.zeros(no_of_periods),np.ones(no_of_stocks_in_portfolio)])
```

Approach 2:

In this approach, we ignored stock selection IP and reformulated the weight selection problem to be a single Mixed Integer Programming (MIP) that constrains the number of non-zero weights to be an integer. This can be done by replacing m with n so that optimization occurs over all the weights.

Objective Function

Our objective is to identify the optimal weights of stocks in our portfolio to minimize the difference in returns between the market index and our portfolio. Given below is the objective function:

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

Where:

q_t : Return of the NASDAQ-100 index at time t
r_{it} : Return of stock i (where stock i is one of the selected stocks) at time period t
w_i: Weight of stock i selected in our portfolio

The objective function is non-linear and can be reformulated in a linear way as below:

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

Where $z_t = |q_t - \sum_{i=1}^{n} w_i r_{it}|$

Decision Variables

w_i: Weight of stock i selected in our portfolio
z_t : Absolute difference between index return at t and weighted return of selected indexes (from objective function)
y_j : A binary variable that represents if stock j is present in our portfolio or not

Constraints

1. Selecting exactly m stocks in our fund (1 constraint)

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

2. Sum of weights of the m stocks in our portfolio equals 1 (1 constraint)

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

3. Constraints from reformulating non-linear objective function as linear:

$z_t > q_t$ - sum(w_i x r_{it}) and
$z_t < q_t$ - sum(w_i x r_{it})

Total such constraints: 2*(no. of periods of returns) as given in CSV file

4. Big M constraint that forces $w_i = 0$ if $y_i = 0$:

$$w_i - M y_i \leq 0$$

We took M=1 as our optimal weights for the stocks lie between 0 and 1. This is also the smallest value of M that we can take.

Total Big M constraints: m

5. All the weights should be greater than zero

$$w_i \geq 0.$$

Total such constraints: m

Code Snippet for our Objective Function and Constraint Matrices for Approach 2:

```
# Initializing and defining A, b, sense and obj matrices that go into into the MIP model
obj = np.concatenate([np.ones(no_of_periods),np.zeros(no_of_stocks),np.zeros(no_of_stocks)])

# Big M value taken to be M as the optimal weights lie between 0 and 1
M=1

A = np.zeros(((2*no_of_periods+2*no_of_stocks+2),len(obj)))
b = np.concatenate([np.zeros(2*no_of_periods+no_of_stocks),np.ones(1),np.zeros(no_of_stocks),m*np.ones(1)])
sense = ['>']*(2*no_of_periods+no_of_stocks)+['=']+['<']*(no_of_stocks)+['=']

for i in range(no_of_periods):
    A[2*i,i] = 1
    A[2*i+1,i] = 1
    for j in range(no_of_stocks):
        A[2*i,(no_of_periods+j)] = df_return_all_stocks.iloc[i,j]
        A[2*i+1,(no_of_periods+j)] = -df_return_all_stocks.iloc[i,j]
    b[2*i]=stocks_2019_returns.iloc[i,0]
    b[2*i+1]=-stocks_2019_returns.iloc[i,0]

for k in range(no_of_stocks):
    A[2*no_of_periods+k,no_of_periods+k] = 1

A[2*no_of_periods+no_of_stocks,:]=np.concatenate([np.zeros(no_of_periods),np.ones(no_of_stocks),np.zeros(no_of_stocks)])

for k in range(no_of_stocks):
    A[2*no_of_periods+no_of_stocks+k+1,no_of_periods+k] = 1
    A[2*no_of_periods+no_of_stocks+k+1,no_of_periods+no_of_stocks+k] = -M # Defining the Big M constraint

A[-1,:]=np.concatenate([np.zeros(no_of_periods),np.zeros(no_of_stocks),np.ones(no_of_stocks)])
```

Results

Approach 1:

We ran the optimization models (stock selection and optimal weights models) for different values of m (ie. the number of stocks to be included) and checked for the in-sample (2019) and out-of-sample tracking errors/ performance. We varied m from 5 to 100.

Code Snippet for Stock Selection Model:

```
# Running the stock selection model
b_stock_selection=return_b_stock_selection(m)
ipMod = gp.Model()
ipMod_x = ipMod.addMVar(no_of_stocks_2019+no_of_stocks_2019**2,vtype=['B']*(no_of_stocks_2019+no_of_stocks_2019**2))
ipMod_con = ipMod.addMConstrs(A_stock_selection, ipMod_x, sense_stock_selection, b_stock_selection)
ipMod.setMObjective(None,obj_stock_selection,0,sense=gp.GRB.MAXIMIZE)

ipMod.Params.OutputFlag = 0 # tell gurobi |to shut up!!
ipMod.optimize()
```

Code Snippet for Finding the Optimal Weights Model:

```
# Running the optimal weights model
lpMod = gp.Model()
lpMod_x = lpMod.addMVar(len(obj))
lpMod_con = lpMod.addMConstrs(A, lpMod_x, sense, b)
lpMod.setMObjective(None,obj,0,sense=gp.GRB.MINIMIZE)

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

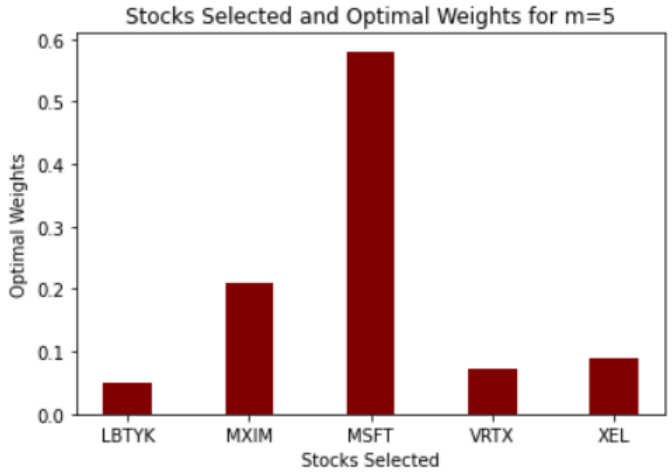
Number of Stocks in Portfolio (m) =5

For m=5, the best stocks to be included in our index fund from the Weight Selection IP model are outlined below:

- Liberty Global PLC Class C (LBTYK)
- Maxim (MXIM)
- Microsoft (MSFT)
- Vertex Pharmaceuticals Inc. (VRTX)
- Xcel Energy (XEL)

Optimal Weights

We then calculated the optimal weights that the 5 stocks should be assigned using the Optimal Weights Model. LBTYK was weighted the lowest at **4.89%**, with VRTX weighted at **7.12%**, XEL at **8.93%**, MXIM at **22.04%**. MSFT was assigned the maximum weight at **58.04%** among all the selected stocks



Detailed below are the in-sample (2019) and out-of-sample tracking errors for m=5:

2020 Tracking Error: **0.87**
2019 In-Sample Tracking Error: **0.79**

This indicates that the 2020 market returns and our portfolio returns have a high variation of ~0.87 or ~87%. To create an accurate tracking portfolio, our goal is to minimize variation as much as possible so our portfolio follows the market more closely.

Stock Selection & Portfolio Weights for m=10-100 :

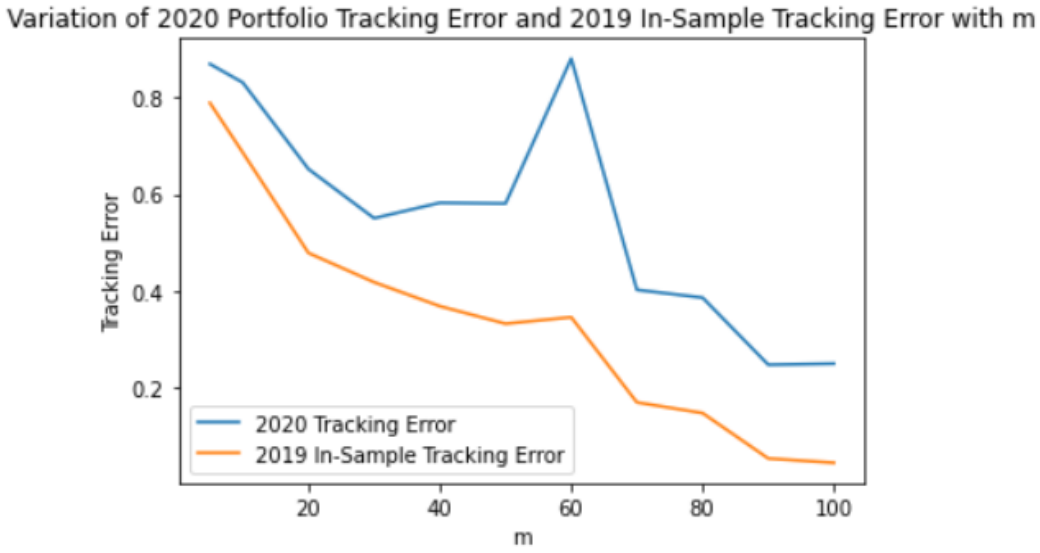
In this case, to construct an index fund that replicates the NASDAQ-100, we will select stocks to include in our portfolio in increments of 10 to analyze how the performance of the portfolio differs.

We will do this by setting the m value in the first constraint in our stock selection model varying it from m=10 to m=100, in increments of 10. When m=100, all stocks will be included in our index, rather than finding a smaller selection of the best 100 stocks. The 2nd and 3rd constraints remain the same, as each stock i still must only have one representative stock j in the index, and that stock j must be in the fund to best represent stock i.

Detailed below are the stocks selected, optimal weights, and the in-sample and out-of-sample tracking errors for each value of m:

	m	2020 Tracking Error	2019 In-Sample Tracking Error	Stocks Selected	Optimal Stock Weights
0	5	0.869670	0.789178	[LBTYK, MXIM, MSFT, VRTX, XEL]	[0.04886174835252491, 0.21038806005665553, 0.5...
1	10	0.831317	0.686533	[ATVI, ALGN, BKNG, KHC, LBTYK, MXIM, MSFT, ROS...	[0.04113108880018398, 0.012208224438274592, 0....
2	20	0.652338	0.478836	[ATVI, ALGN, GOOGL, ANSS, ADP, BIIB, CMCSA, DL...	[0.0246128451625659, 0.012773472041639327, 0.2...
3	30	0.550662	0.418279	[ATVI, ADBE, GOOGL, AMAT, ADP, BIIB, CTXS, CMC...	[0.016378555223330676, 0.038975319529575385, 0...
4	40	0.582591	0.368785	[ATVI, ADBE, ALGN, GOOGL, AMGN, AMAT, ADP, BII...	[0.022628982812142484, 0.023840141178068423, 0...
5	50	0.581148	0.332540	[ATVI, ADBE, ALXN, ALGN, GOOGL, AMGN, AMAT, AD...	[0.01945631641397821, 0.03434141488995112, 0.0...
6	60	0.880871	0.345927	[ATVI, ADBE, AMD, ALXN, ALGN, GOOGL, AMGN, AMA...	[0.01732165395289185, 0.05101686947636424, 0.0...
7	70	0.402497	0.169824	[ATVI, AMD, ALXN, ALGN, GOOG, AMGN, AAPL, AMAT...	[0.011191256355570224, 0.007895151456320913, 0...
8	80	0.386431	0.147683	[ATVI, AMD, ALXN, ALGN, GOOGL, AMGN, AAPL, AMA...	[0.008611353419442337, 0.006452940814983654, 0...
9	90	0.247582	0.053779	[ATVI, ADBE, AMD, ALXN, ALGN, GOOG, AMZN, AMGN...	[0.003440798812385665, 0.020014628603631023, 0...
10	100	0.249936	0.044911	[ATVI, ADBE, AMD, ALXN, ALGN, GOOGL, GOOG, AMZ...	[0.004222720908253049, 0.017137939539244973, 0...

Variation of in-sample (2019) and out-of-sample (2020) tracking errors with variation in m is plotted below:

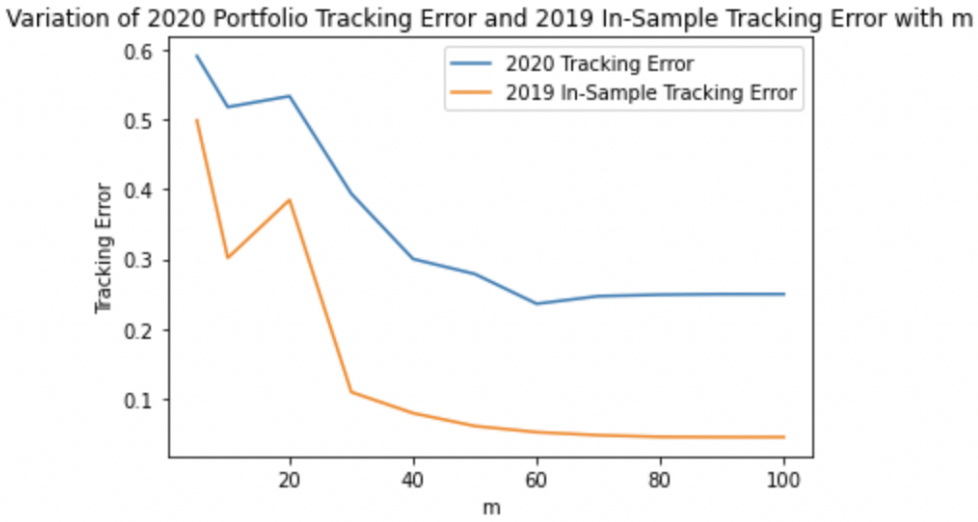


Specific Insights: In-sample vs Out-of-Sample Performance:

- Given the portfolio performances above in 2020 vs 2019, we can see that the 2019 in-sample performance has been consistently better than the performance of our portfolio in 2020. This is due to the fact that our optimization models have been designed considering the 2019 stock price data. However, we should keep in mind that there could have been many external economic factors affecting our outcomes in 2020 due to COVID-19.

- ***Variation of Tracking Errors with m:***
    - ***2020 Out-of-Sample Tracking Error:*** We see that the out-of-sample tracking error initially decreases until m=30 and then slightly increases to m=50, spiking up massively at m=60. Thus after m=30, we observe diminishing returns of including more stocks in the portfolio leading to bad performance. However, after m=60, the out-of-sample performance again increases and decreases until m=100. For higher values of m, it is intuitive that these will have higher performance and low tracking errors as we will be selecting the most representative stocks from our market index into our portfolio.
    - ***2019 In-Sample Tracking Error:*** In-sample performance is observed to decrease continuously as we increase m from 5 to 100

Approach 2:



We ran the MIP optimization models for different values of m (ie. the number of stocks to be included) and checked for the in-sample (2019) and out-of-sample tracking errors/ performance. We varied m from 5 to 100.

Code Snippet for MIP Model (Running each model for each value of m for 1 hr):

```
# Running the MIP model
lpMod = gp.Model()
lpMod_x = lpMod.addMVar(len(obj),vtype=['C']*(no_of_periods+no_of_stocks)+['B']*(no_of_stocks))
lpMod_con = lpMod.addMConstrs(A, lpMod_x, sense, b)
lpMod.setMObjective(None,obj,0,sense=gp.GRB.MINIMIZE)

lpMod.Params.OutputFlag = 0 # tell gurobi to shut up!!
lpMod.Params.TimeLimit = 3600
lpMod.optimize()
```

Detailed below are the stocks selected, optimal weights, and the in-sample and out-of-sample tracking errors for each value of m:

	m	2020 Tracking Error	2019 In-Sample Tracking Error	Stocks Selected	Optimal Stock Weights
0	5	0.591398	0.499259	[AMZN, ADI, AAPL, MSFT, MDLZ]	[0.2501225979998439, 0.1137580710529145 1, 0.19...
1	10	0.518220	0.301862	[GOOGL, AMZN, AAPL, CTXS, FB, INTC, MSFT, PAYX...	[0.10107942442492181, 0.13062393741220038, 0.1...
2	20	0.533836	0.385102	[ATVI, ADBE, AMD, ALXN, GOOGL, GOOG, AMZN, AMG...	[0.011485932796080116, 0.059569856056554, 0.0,...
3	30	0.393863	0.109405	[ADBE, GOOGL, AMZN, AMGN, AAPL, ADP, BIDU, BKN...	[0.029247917295816484, 0.08665842835568552, 0....
4	40	0.300330	0.079215	[ATVI, ADBE, GOOGL, AMZN, AMGN, AAPL, ADP, BII...	[0.005928919458291493, 0.0205553196367539, 0.0...
5	50	0.278776	0.060758	[ADBE, AMD, GOOGL, AMZN, AMGN, AAPL, ADP, BIDU...	[0.02254720386448382, 0.004169356020540 7765, 0...
6	60	0.236082	0.051953	[ATVI, ADBE, AMD, GOOG, AMZN, AMGN, AAPL, AMAT...	[0.006200078580536943, 0.01938727231578326, 0....
7	70	0.247045	0.047568	[ATVI, ADBE, AMD, GOOGL, AMZN, AMGN, AAPL, AMA...	[0.0022014181770870726, 0.01929402349358076, 0...
8	80	0.249124	0.045227	[ATVI, ADBE, AMD, ALXN, GOOGL, GOOG, AMZN, AMG...	[0.0031196129505436294, 0.017842007035128473, ...
9	90	0.249943	0.044911	[ATVI, ADBE, AMD, ALXN, GOOGL, GOOG, AMZN, AMG...	[0.004223289593193305, 0.017140049274027642, 0...
10	100	0.249936	0.044911	[ATVI, ADBE, AMD, ALXN, ALGN, GOOGL, GOOG, AMZ...	[0.004222720908253049, 0.017137939539244973, 0...

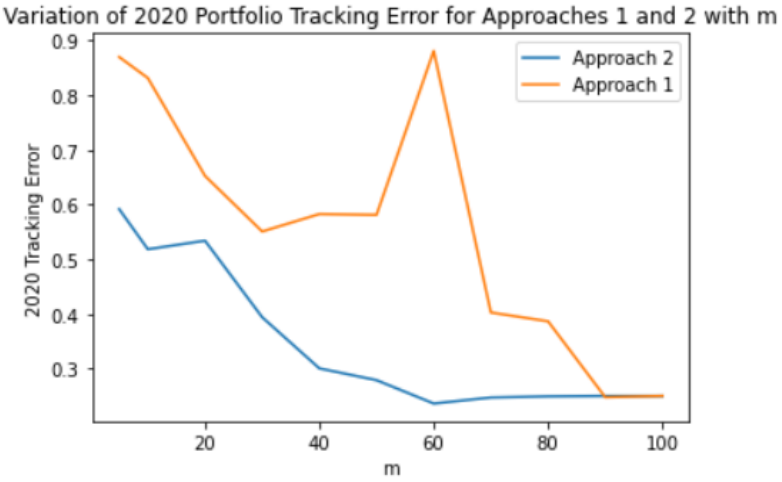
Variation of in-sample (2019) and out-of-sample (2020) tracking errors with variation in m is plotted below:

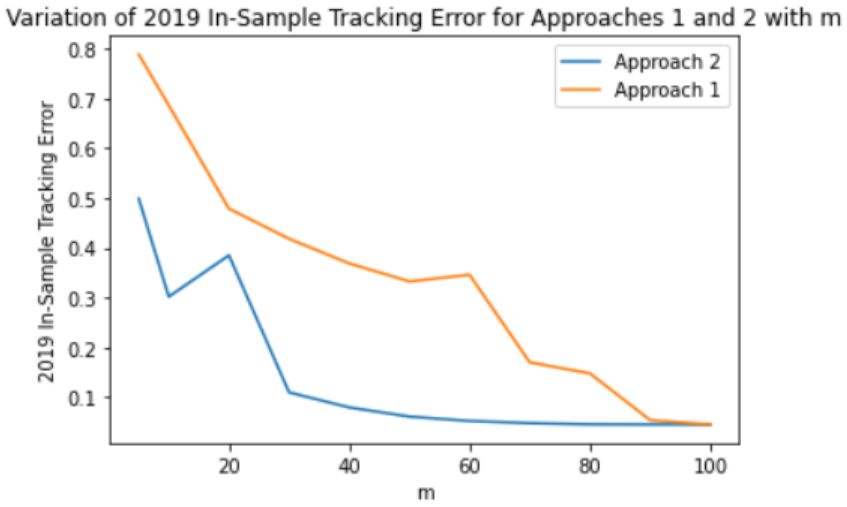
Specific Insights: In-sample vs Out-of-Sample Performance:

Looking at the graph we can see that the in-sample error (2019) is less than that of 2020 and the tracking error for both gradually decreases as we increase m. The tracking error decreases until m = 60 and post that the graph flattens out. The error decreases as we increase the value of m but it doesn't make sense to increase m to a high value. Somewhere around 40-50 would be a good number to pick.

Conclusion and Recommendations

Approach 1 & Approach 2 Comparison





As we see in approaches 1 and 2, the tracking error decreases as more stocks are added to the fund. The number of stocks selected is influenced by the cost of having more stocks in the fund. The number of stocks is unknown. We need to explore whether lower differences in returns or having fewer stocks is more important to our boss for the given fund.

Recommendations:

- If the goal is to have more stocks in the model, then the first approach would make more sense. In the first approach, the optimal would be 70 stocks. In the second approach, the optimal would be around 40 stocks.
- We should also consider how often the models are being run. If the model needs to be run on a more frequent basis (daily for example) then approach 1 would be better than approach 2 as that takes more than 8 hours to run. Also, approach 1 uses less computing power and has about a 20-30% decrease in the difference in the rate of return.
- But if computing power is not an issue then approach 2 is better than approach 1 because the error is lower at every value of m for both insample and out of sample.

After discussing the factors with limitations in terms of computing power and time we should decide on the approach.