Problem Statement

We executed 'passive' equity money management strategy where the goal is to choose a portfolio that mirrors the movements of the broad market population or a market index. Our goal was to choose a portfolio that mirrors the movements of NASDAQ-100 index as closely as possible.

Constructing an index fund that tracks a specific broad market index could be done by simply purchasing all n stocks in the index, with the same weights as in the index. However, this approach is impractical (many small positions) and expensive (rebalancing costs may frequently be incurred, price response to trading). An index fund with m stocks, where m is substantially smaller than the size of the target population, n, seems desirable.

We executed the following two methods to create the index fund –

Method (i): Solving one IP and one LP

We created the index fund in multiple steps. First, we formulated an integer program that picked exactly m out of n stocks for our portfolio. This integer program took as input a 'similarity matrix', which we call ρ. The individual elements of this matrix, ρ_{ij}, represent daily return correlation between stock i and j. Next, we solved a linear program to decide how many of each chosen stock to buy for your portfolio and finally evaluate how well our index fund does as compared to the NASDAQ-100 index, out of sample.

Method (ii): Solving one MIP

This approach involved creating the index fund in a single shot by solving a mixed integer programming problem. We re-formulated the weight selection problem to be an MIP that constraints non-zero weights to be an integer. This problem solves for both stock selection and weight allocation in a single attempt.

Method (I) – Consider stock selection and weight calculation as two separate problems.

I. Stock selection

The binary decision variables y_j indicates which stocks j from the index are present in the fund ($y_j = 1$ if j is selected in the fund, 0 otherwise). For each stock in the index, i = 1, . . . , n, the binary decision variable x_{ij} indicates which stock j in the index is the best representative of stock i ($x_{ij} = 1$ if stock j in index is the most similar stock i, 0 otherwise).

The first constraint selects exactly m stocks to be held in the fund. The second constraint imposes that each stock i has exactly one representative stock j in the index. The third constraint guarantees that stock i is best represented by stock j only if j is in the fund. The objective of the model maximizes the similarity between the n stocks and their representatives in the fund.

Constraints and objectives

The objective of this IP problem is to provide us with the best mapping:

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

Constraints for this problem are given below:

1. Selection of exactly m stocks:

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

2. Each stock with only one representative:

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

3. Best representation of stock with the ones in the fund:

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

Keeping all variables binary, i.e.,

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

Python Code for this problem

```
#creating objective array
N = len(df_corr)
obj = np.zeros(N**2 + N) # N**2 + N represents number of decision variables

#filling in the values in objective array
index = 0
for i in df_corr.columns:
    for j in df_corr.columns:
        obj[index] = df_corr.loc[i,j]
        index += 1
```

```
#initializing constraints, and directions
A = np.zeros((1 + N + N**2,N**2 + N))
b = np.zeros(1 + N + N**2)
direction = np.array(['']*(1 + N + N**2))
```

```
#1st constraint -- limiting the number of selected stocks to m
A[0,N**2:] = 1
direction[0] = '='
```

```
#2nd constraint -- limiting the similar stock count to 1
row = 1
ind_vec = np.array(range(N))
for i in range(N):
    A[row, i*N + ind_vec] = 1
    b[row] = 1
    direction[row] = '='
    row += 1
```

```
#3rd constraint -- making sure best stock representation is present in fund
A[N+1:,0:(N**2)] = np.diag(np.ones(N*N))

for i in range(N):
    A[(N+1+(i*N)):(N+1+(i*N)+100),N**2:] = -(np.diag(np.ones(N)))

b[row : row + N*N] = 0
direction[row : row + N*N] = '<'
```

II. Calculating weights for the selected stocks

To get the portfolio weights we matched the returns of the index as closely as possible. r_{it} is the return of stock i (where stock i is one of the chosen stocks from above) at time t, q_t is the return of the index at time t, and w_i is the weight of stock i in the portfolio.

Constraints and objectives

Here the objective would be to minimize the training error, i.e.,

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

Constraints for this problem are given below:

1. <u>All non-negative weights adding up to 1:</u>

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

$$w_i \geq 0.$$

2. To re-formulate the above non-linear problem as an LP, T new variables, Y_1 ….. Y_T were added and the constraints were as follows:

$$Y_t + \sum_{i=1}^{m} w_i r_{it} \geq q_t \quad \forall\, t = 1 \ldots . T$$

$$Y_t - \sum_{i=1}^{m} w_i r_{it} \geq -(q_t) \quad \forall\, t = 1 \ldots . T$$

And the new objective is to minimize:

$$min \sum_{t=1} Y_t$$

Python Code for this problem

```
#creating the objective array
N = len(df_2019) - 1
obj = np.zeros(sum(selected_stocks.astype(int)) + N)
obj[0:N] = 1
```

```
# creating constraints
n_var = sum(selected_stocks.astype(int)) + N
A = np.zeros((N*2+1, n_var))
b = np.zeros(N*2+1)
direction = np.array(['']*(N*2+1))
```

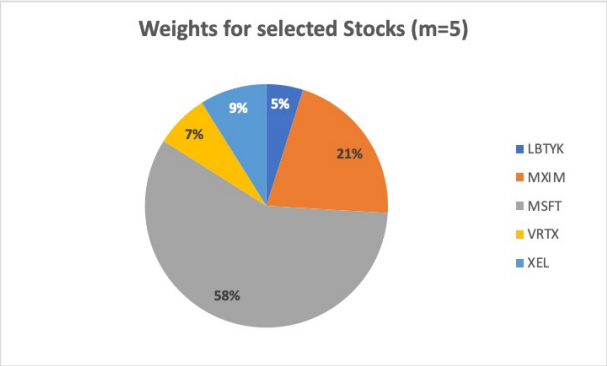
```
#constraint 1
A[0,N:] = 1
b[0] = 1
direction[0] = '='
```

```
# remaining constraints
q = df_2019_returns[['NDX']]
r = df_2019_returns[df_2019_returns.columns[1:].values[selected_stocks.astype(bool)]]

A[1:N+1,0:N] = np.diag(np.ones(N))
A[N+1:,0:N] = -(np.diag(np.ones(N)))
for i in range(N):
    A[i+1,N:] = r.reset_index().drop(columns = 'Date').iloc[i+1].values
    b[i+1] = q.reset_index().drop(columns = 'Date').iloc[i+1].values
    direction[i+1] = '>'
    A[N+1+i, N:] = r.reset_index().drop(columns = 'Date').iloc[i+1].values
    b[N+1+i] = q.reset_index().drop(columns = 'Date').iloc[i+1].values
    direction[N+1+i] = '<'
```

Insights from Method (I)

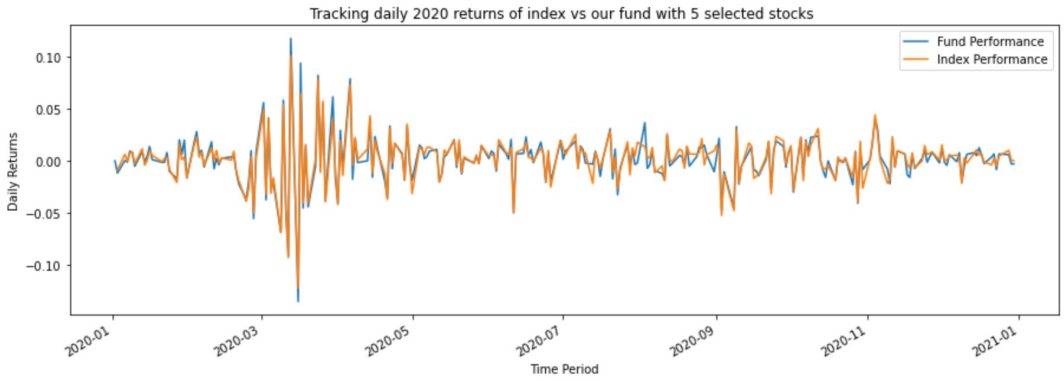
To begin this project, we tried executing method (I) for m = 5, i.e., selected 5 stocks and tried allocating weights to them. Below are the 5 selected stocks along with their respective weights that we obtained.



To evaluate the performance of this fund with m =5 stocks, we performed the following two analyses –

1. Tracked daily returns for index vs. fund with m=5 stocks in 2020

   We see that the fund with 5 selected stocks tracks the index closely in 2020 .

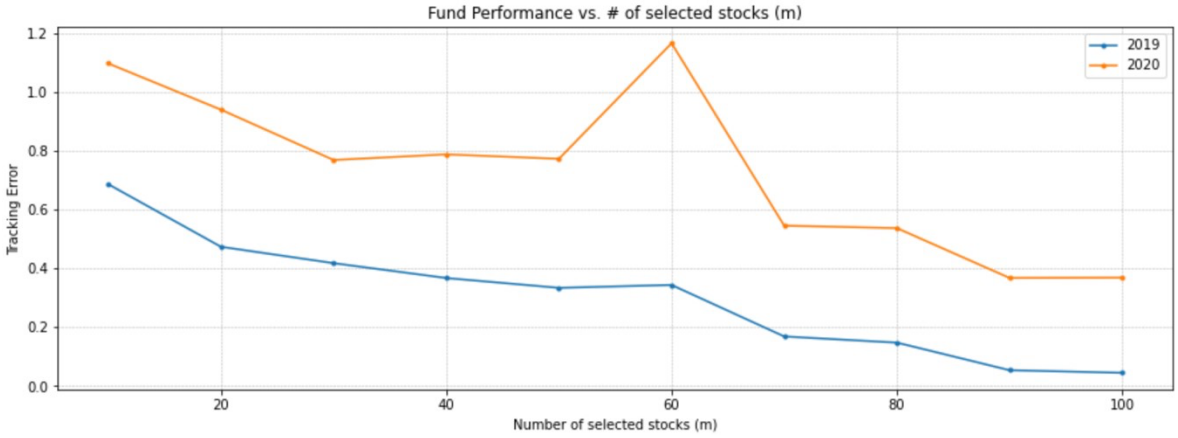


2. Calculated the absolute difference in daily returns for index and fund for 2019 (in-sample data) and 2020 (out of sample data)

$$\left|\sum_{t=1}^{T}\left|q_t - \sum_{i=1}^{5} w_i r_{it}\right|\right|$$

   The value of this tracking error is 1.11.

Next, we performed stock allocation and weight allocation for m = 10, 20, ...100. We have calculated the tracking error for all funds (m = 10, 20 ….100) for 2019 and 2020 data.



Approach (I)	m	5	10	20	30	40	50	60	70	80	90	100
Year	2019	0.79	0.69	0.47	0.42	0.37	0.33	0.34	0.17	0.15	0.05	0.04
	2020	1.11	1.10	0.94	0.77	0.79	0.77	1.17	0.55	0.54	0.37	0.37

We can see that tracking error generally decreases as well select more stocks for the fund, but there is an irregularity at m = 60 where we observe an increase in the tracking error before it quickly decreases to re-align with the general trend.

Method (II) – A single MIP to perform stock selection and weight allocation

Binary variables $y_1, y_2 \ldots. y_n$ represents whether stock y_i has been selected in the find or not. Also, the weight of the stock is 0 if it is not selected in the fund, i.e., $w_i = 0$ if $y_i = 0$. **Since y_i is already binary, our big M constraint can be 1**. Moreover, since we want to select 'm' stocks, sum of y_i can be equal to m.

Objective function and constraints

The objective:

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

Constraints are given below:

1. Sum of selected stocks:

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

2. To maintain consistency in case of selection of a stock:

$$w_i \leq y_i$$

3. Sum of weights = 1:

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

4. To reformulate for linear programming:

$$Y_t + \sum_{i=1}^{n} w_i r_{it} \geq q_t \quad \forall\, t = 1 \ldots.. T$$

$$Y_t - \sum_{i=1}^{n} w_i r_{it} \quad \geq -(q_t) \quad \forall\, t = 1 \ldots.. T$$

5. The new objective is to minimize:

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

Python Code

```
#creating the objective array
N = len(df_2019) - 1 # number of time periods
n = len(df_corr) #number of stocks
obj = np.zeros(N + n + n) #function to be minimized + weights of stocks (w) + selection variable for each stock (y)
obj[0:N] = 1
```

```
# creating constraints
n_var = N + n + n
A = np.zeros((n+1+1+N*2, n_var))
b = np.zeros(n+1+1+N*2)
direction = np.array(['']*(n+1+1+N*2))
```

```
#1st constraint -- limiting the number of selected stocks to m
A[0,N+n:] = 1
direction[0] = '='
```

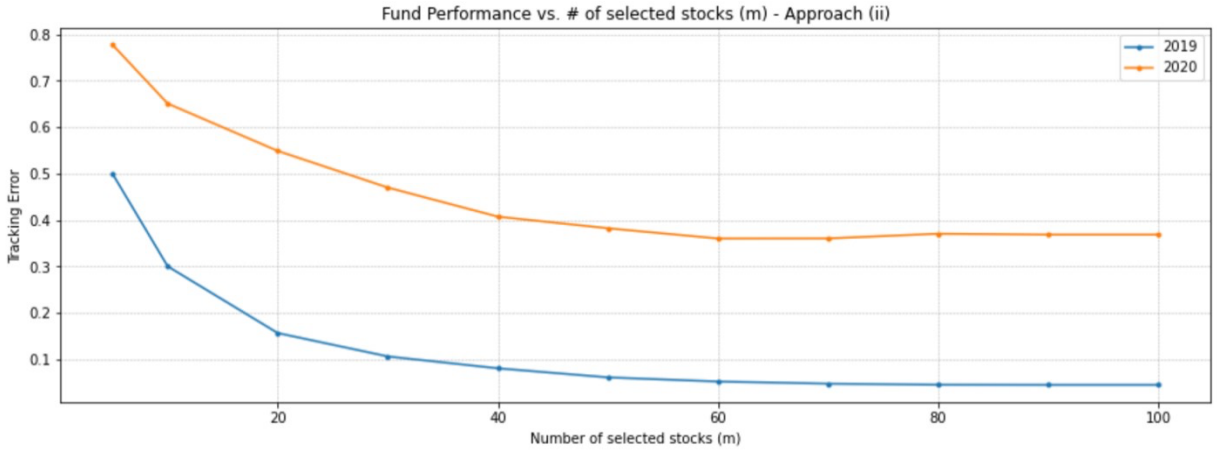
```
#2nd constraint - wi = 0 if yi = 0, big M = 1
A[1:n+1,N:N+n] = np.diag(np.ones(n))
A[1:n+1,N+n:] = -(np.diag(np.ones(n)))
direction[1:n+1] = '<'
```

```
# 3rd constraint - sum of w = 1
A[n+1,N:N+n] = 1
b[n+1] = 1
direction[n+1] = '='
```

```
# 4th Constraint - converting to LP
A[n+1+1:n+1+1+N,0:N] = np.diag(np.ones(N)) #coefficient of y
A[n+1+1+N:,0:N] = -(np.diag(np.ones(N))) #coefficient of y

for i in range(N):
    A[i+n+1+1,N:N+n] = df_2019_returns.reset_index().drop(columns = ['Date', 'NDX']).iloc[i+1]
    b[i+n+1+1] = q.reset_index().drop(columns = 'Date').iloc[i+1].values
    direction[i+n+1+1] = '>'
    A[i+n+1+1+N,N:N+n] = df_2019_returns.reset_index().drop(columns = ['Date', 'NDX']).iloc[i+1]
    b[i+n+1+1+N] = q.reset_index().drop(columns = 'Date').iloc[i+1].values
    direction[i+n+1+1+N] = '<'
```

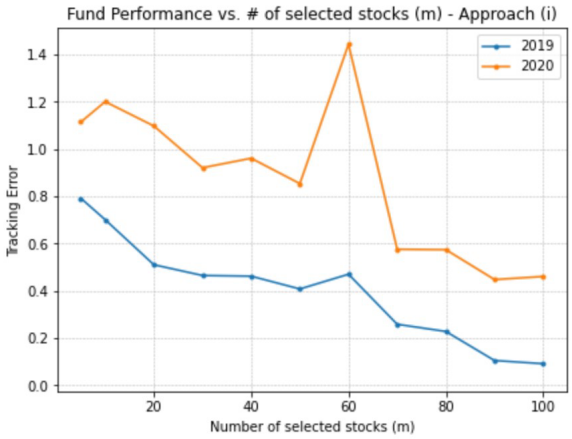
Insights from Method (II)

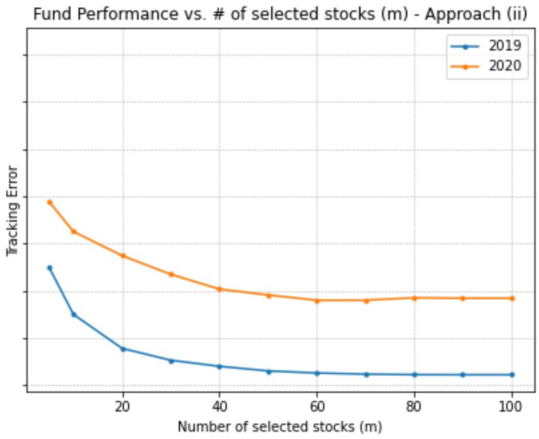


Approach (II)	m	5	10	20	30	40	50	60	70	80	90	100
Year	2019	0.50	0.30	0.16	0.11	0.08	0.06	0.05	0.05	0.05	0.04	0.04
	2020	0.78	0.65	0.55	0.47	0.41	0.38	0.36	0.36	0.37	0.37	0.37

As with approach (I), we calculated the tracking error for all funds (m = 10, 20 ….100) for 2019 and 2020 data. The tracking error here seems to be lower and better than approach (I).

Comparing the results from Method (I) and Method (II)

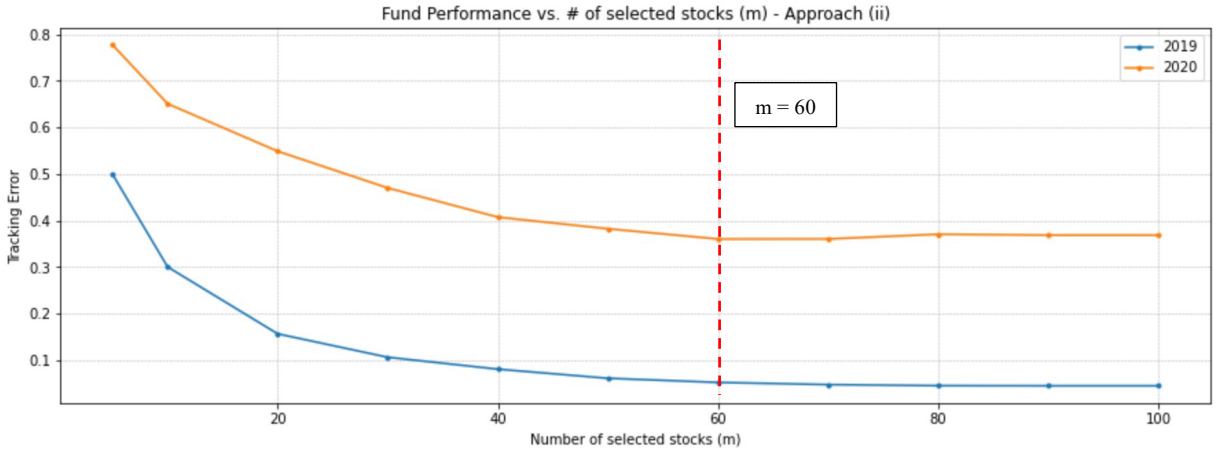




The main difference between the two approaches is that the first approach calculates stock selection and the weights separately, while the second approach ignores the stock picking aspect and calculates the weights and stocks used in one step. The graphs below depict the tracking error with the index to the number of stocks picked in the tracking portfolio over a year. **The second approach has a lower tracking error in general** since the first approach starts with about 0.7/1.1 tracking error while the second approach starts with 0.3/0.65 error in 2019/2020 respectively. Both approaches have the same general trend with additional stocks decreasing the tracking error, but at the 60-stock mark the graphs differ. **For the second approach m=60 marks the point in which additional stocks do not improve the tracking error, while in the first approach it represents a massive spike in tracking error.**

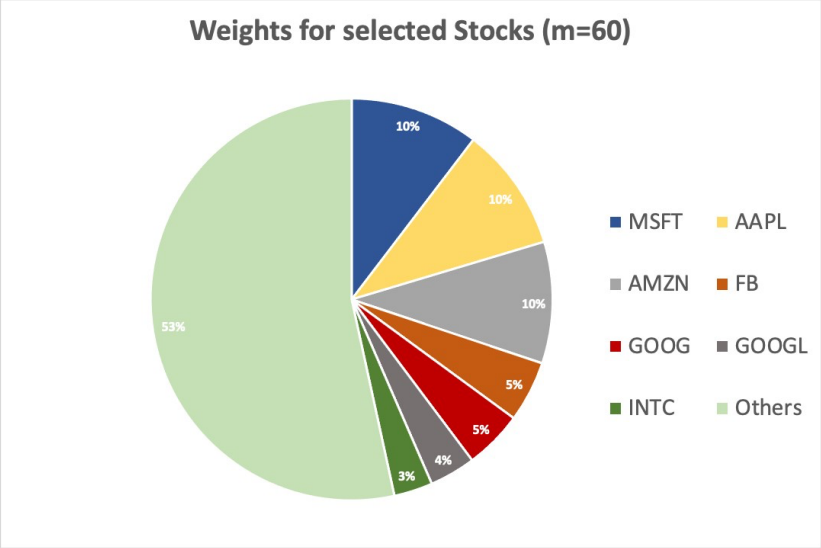
Recommendation

Based on our analysis, **using 60 stocks from Method (II) is the most efficient way of matching the NASDAQ-100 performance**. Looking at the tracking error of the portfolio, 60 stocks is the point at which adding additional stocks does reduce the tracking error with the index. Hence, we should be using Method (II) to select stocks and calculate their weights.



Below are the stocks with highest weightage among the 60 selected stocks based on Approach (II).

- Selected stocks with most weightage are Microsoft, Apple, Amazon, Facebook, and Google
- All the stocks clubbed under others have weightage < 3% each



Another thing to mention to our boss, is how often do we plan to run the models? If we plan to run our modes on a daily basis, method 1 would be preferred as it's less resource intensive and can be calculated in a matter of minutes, as method 2 takes 8+ hours to run (even on the latest hardware). If the boss is focused on running the model on less stocks, and resources is a nonissue, then method 2 is optimal.

Appendix

Below heatmap shows weights assigned to various stocks across different funds (m = 5, 10,…..100). This helps identify the important stocks, which are selected across most of the funds. **Majority of the big tech company stocks like Apple, Microsoft, Amazon, Facebook, Cisco, and Google are selected across most of the funds (m = 10, 20….100) with high weightages** which shows that these stocks are a good representation of the index.

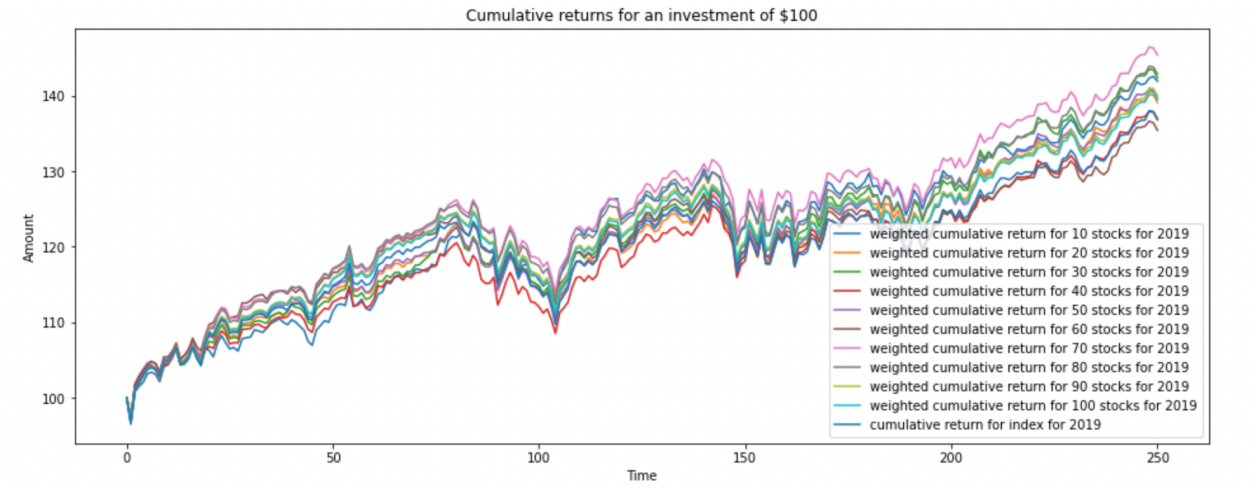
Green – High weightage of given stock for 'm' stock fund | Red – Low weightage of given stock for 'm' stock fund

	# Selected Stocks										
Stocks	5	10	20	30	40	50	60	70	80	90	100
AAPL	0%	14%	12%	12%	11%	10%	10%	10%	10%	10%	10%
MSFT	0%	16%	9%	10%	11%	11%	10%	10%	10%	10%	10%
AMZN	0%	12%	10%	10%	10%	10%	10%	10%	10%	10%	10%
FB	0%	9%	5%	5%	5%	5%	5%	5%	5%	5%	5%
CSCO	25%	0%	4%	3%	3%	3%	3%	3%	3%	3%	3%
GOOG	0%	9%	0%	9%	8%	8%	5%	0%	3%	3%	3%
MDLZ	19%	0%	5%	3%	3%	3%	2%	2%	1%	1%	1%
GOOGL	0%	0%	11%	0%	0%	0%	4%	9%	6%	6%	6%
INTC	0%	7%	4%	3%	3%	3%	3%	3%	3%	3%	3%
ADP	11%	0%	5%	4%	3%	2%	2%	2%	1%	1%	1%
CTAS	29%	0%	0%	0%	0%	1%	1%	0%	0%	0%	0%
CMCSA	0%	0%	4%	3%	3%	2%	3%	3%	2%	2%	2%
ADBE	0%	0%	3%	4%	2%	2%	2%	2%	2%	2%	2%
PYPL	0%	0%	4%	3%	2%	2%	2%	2%	2%	2%	2%
AMGN	0%	8%	0%	0%	2%	2%	1%	1%	1%	1%	1%
NFLX	0%	0%	3%	3%	2%	2%	2%	2%	1%	2%	2%
KLAC	15%	0%	0%	0%	0%	0%	0%	0%	0%	1%	1%
GILD	0%	0%	4%	3%	2%	2%	1%	1%	1%	1%	1%
PAYX	0%	13%	0%	0%	0%	1%	0%	0%	1%	1%	1%
COST	0%	0%	4%	3%	2%	1%	1%	2%	1%	1%	1%
AVGO	0%	0%	3%	2%	1%	1%	1%	1%	1%	1%	1%

This heatmap shows the stocks which are not selected across any fund (and hence assigned 0% weight) corresponding to m = 5,10,20….100 stocks. These below stocks might be well represented by few other stocks which are able to track the index more closely.

ALGN	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
ADI	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
ANSS	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
CDW	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
CERN	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
CPRT	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
DOCU	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
LBTYA	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
SGEN	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
SWKS	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
SPLK	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
WDC	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

A few additional visualizations:


Cumulative returns for an investment of $100
Amount
Time
140
130
120
110
100
0
50
100
150
200
250
weighted cumulative return for 10 stocks for 2019
weighted cumulative return for 20 stocks for 2019
weighted cumulative return for 30 stocks for 2019
weighted cumulative return for 40 stocks for 2019
weighted cumulative return for 50 stocks for 2019
weighted cumulative return for 60 stocks for 2019
weighted cumulative return for 70 stocks for 2019
weighted cumulative return for 80 stocks for 2019
weighted cumulative return for 90 stocks for 2019
weighted cumulative return for 100 stocks for 2019
cumulative return for index for 2019