Optimization Project 2 - Integer Programming

Sai Bhargav, Praneet Kumar, Anthony Moreno, Joe Morris

Fa22 OPTIMIZATION I

Table of Contents

I. Introduction **3**

II. Methodology **3**

A. Stock Selection (IP) and Weight Calculations (LP) 3

Integer Programming for stock selection 3

Linear Programming for weights 6

B. Stock Selection and Weight Calculations (MIP) 7

III. Results **9**

A. Stock Selection (IP) and Weight Calculations (LP) 10

B. Stock Selection and Weight Calculations (MIP) 13

IV. Conclusion **16**

I. Introduction

We were tasked with coming up with a way to pick stocks and their corresponding weights to create a fund that mirrors the NASDAQ-100 index. By choosing the stocks with the highest return correlations with others stocks in the portfolio and using them as representatives of the index we only have to use a fraction of the stocks within the NASDAQ-100 to track the entire index. We hope to find the stocks and corresponding weights that make up our funds portfolio as well as the number of stocks that is optimal to hold within our fund to minimize absolute deviation of returns in comparison to the underlying index. Our team will analyze the returns from the NASDAQ-100 in 2019 in order to create a passive strategy with our desired results. We then plan on using the returns from the index in 2020 as a baseline to better understand how our passive strategy performs. By first creating an integer program to select the stocks based on their correlations, then implementing a linear programming approach to calculate the weights of these stocks we will be able to achieve this goal. We then plan on trying a mixed integer program method in order to find the best possible solution.

II. Methodology

The first step in solving this problem is to convert the 2019 and 2020 prices data into returns data. The following formula is used to achieve that:

$$R_t = (P_t - P_{t-1})/P_{t-1}$$

Where,

R_t = Return of the stock at time t
P_t = Price of the stock at time t
P_{t-1} = Price of the stock in previous period (at time t-1)

Using the returns data of the individual stocks we can get the correlation matrix in which each element represents the correlation between the stocks. For example, the element in the row i and column j is the correlation between stock i and stock j. So, the diagonal elements of this matrix are going to be 1 as they represent the correlation between the same stock.

A. Stock Selection (IP) and Weight Calculations (LP)

This method involves solving an Integer Programming to select a predetermined number (m) of stocks within the index to best represent the index. After selecting the stocks we determine the weights of these stocks by solving a Linear Programming problem.

Integer Programming for stock selection:

The objective of this problem is to select a predetermined (m) number of stocks from the index in such a way that the selected stocks represent the index as closely as possible. So, each stock in

the index is represented by a stock in the portfolio. The stocks are selected in such a way that the sum of correlation between the stocks in the index and the corresponding representative stock in the portfolio is maximized.

Objective:

$$max \sum \varrho_{ij} x_{ij}$$

Constraints:

1. $\sum y_j = m$
2. $\sum x_{ij} = 1$ for i = 1,2,.....,n
3. $\sum x_{ij} \leq y_j$ for i, j = 1,2,.....,n
4. $x_{ij}, y_j = \{0,1\}$

Decision Variables:

- ϱ_{ij} = correlation between stock i in the index and stock j in portfolio
- x_{ij} = represents the mapping between stock i and j. Every stock i in the index can be mapped to every stock j in the portfolio but finally for every stock i in the index there will be one stock j from the portfolio which will be selected by our program to maximize the above mentioned objective function
- y_j = represents the presence of a stock j in the portfolio. If this value is 1 then stock j is selected for the portfolio else it is not

Constraints Explanation:

1. We initially select the number (m) of stocks we would like to have in our portfolio. This constraint is important from a business point of view as our transaction costs increase with increase in the number of stocks in our portfolio.
2. For each stock i in the index there should be exactly one stock j in the portfolio which is the closest, out of all the stocks in the portfolio, to stock i in terms of returns.
3. If a particular stock j is not present in the portfolio then it should not represent any stock i from the index. Which means, mathematically, there should be no mapping from index stocks to this stock. If the stock j is present in the portfolio it should be allowed to have mappings from index stocks.
4. All the mappings from stock i in index to stock j in portfolio (x_{ij}) and the indicators of the presence of stock j in the portfolio (y_j) are Binary i.e., 0 or 1.

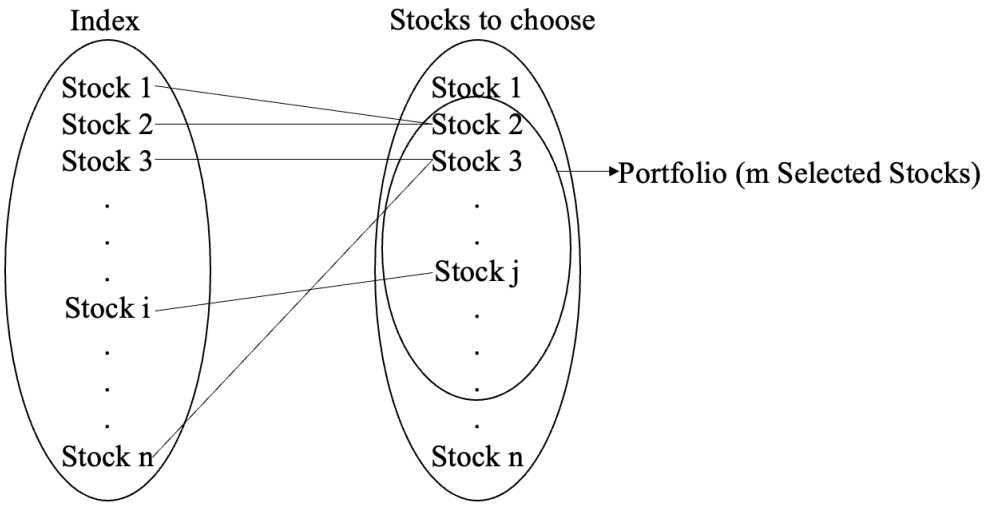
Code Snippet used for constructing the problem in Python:

```
A = np.zeros((1+num+num*num,num*num+num))
A[0,:] = np.array([0]*(num*num) + [1] * num)
for i in range(num):
    A[i+1,i*num:(i+1)*num] = 1
for j in range(num):
    A[(j+1)*num+1:(j+2)*num+1,num*num:num*num+num] = np.identity(num)*-1

A[num+1:1+num+num*num,0:num*num] = np.identity(num*num)

b = np.zeros(1+num+num*num)
b[0] = length
b[1:num+1] = 1
b[num+1:] = 0
sense = np.array(['=']*(num+1)+['<']*(num*num))
vtype = np.array(['B']*(num*num+num))
```

✓ 4.6s

- **A** is the Left Hand side of our constraints
- **b** is the Rights Hand side of our constraints
- **sense** is the relationship between the LHS and RHS of our constraints
- **vtype** has the information of whether our decision variables are Continuous, Integers or Binary

Pictorial Representation:



In the above picture we have our Index with all the stocks from 1 to i on the left. We have all the stocks list on the right. Out of these we choose m stocks to be in our portfolio. All the stocks in the Index will have a mapping to one of these m selected stocks based on how similar the two stocks are in terms of returns. For example, Stocks 2 and 3 in the index are mapped to

themselves which are present in the portfolio. Each stock is exactly similar to itself. But for Stock 1, it cannot be mapped to itself as it is not present in the portfolio. Hence, it will be mapped to some other stock based on returns similarity which in this case is stock 2.

Linear Programming for weights:

From the previous Integer Programming we have the stocks to be selected for our portfolio. Now we need to find the optimal weights for each of the selected stocks such that the difference between the returns of the index and the portfolio on each trading day is minimum.

Objective:

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

We have an absolute value function in our Objective. To convert this to Linear Programming, for each time t we are assigning a variable y_t. For example, for time t=1, we get the following:

$$|q_1 - \sum_{i=1}^{m} w_i r_{i1}| = y_1$$

This will give rise to two equations as below:

$$y_1 \geq q_1 - \sum_{i=1}^{m} w_i r_{i1}$$

$$\Rightarrow y_1 + \sum_{i=1}^{m} w_i r_{i1} \geq q_1$$

And:

$$y_1 \geq -1*(q_1 - \sum_{i=1}^{m} w_i r_{i1})$$

$$\Rightarrow y_1 - \sum_{i=1}^{m} w_i r_{i1} \geq -q_1$$

Constraints:

1. $\sum_{i=1}^{m} w_i = 1$
2. $w_i \geq 0$
3. $y_t + \sum_{i=1}^{m} w_i r_{it} \geq q_t$
4. $y_t - \sum_{i=1}^{m} w_i r_{it} \geq -q_t$

Decision Variables:

- w_i is the weight of the stock i present in our portfolio
- q_t is the return of the index at time t

- r_{it} is the return of the stock i at time t
- y_t is a pseudo variable assigned to the modulus value to convert the objective function to linear space

Constraints Explanation:

1. The sum of the weights of the stocks in our portfolio should be equal to 1.
2. All the weights should be greater than or equal to 0. The weights of the stocks which are not present in our portfolio are assigned the value of zero by restricting them in the code.
3. One of the constraints on y_t to be greater than the positive case of the absolute value function.
4. One of the constraints on y_t to be greater than the negative case of the absolute value function.

Code Snippet used for constructing the problem in Python:

```
A_2 = np.zeros((1+times*2,num+times))
A_2[0,times:] = 1
for i in range(times):
    A_2[i+1,i] = 1
    A_2[i+1,times:] = port_r[i]
    A_2[i+1+times,i] = 1
    A_2[i+1+times,times:] = -1*port_r[i]

b_2 = np.concatenate((np.array([1]),index_r,-1*index_r))
sense_2 = np.array(['='] + ['>']*2*times)
obj_2 = np.array([1]*times + [0]*num)
ub = np.concatenate((np.array([np.inf]*times),weight_con))
```

✓ 0.2s

- **A_2** is the Left Hand side of our constraints
- **b_2** is the Rights Hand side of our constraints
- **sense_2** is the relationship between the LHS and RHS of our constraints
- **obj_2** is the coefficients of our decision variables in the Objective function
- **ub** contains the max values each our decision variables can take. In this case, we are restricting the weights of those stocks which were not selected in the previous IP problem

B. Stock Selection and Weight Calculations (MIP)

In the previous method we found which stocks to select and then calculated the weights of the selected stocks in two consequent steps. In this method, we are going to directly assign the weights to all of the available stocks (n) in such a way that the sum of absolute difference

between the returns of the Index and the portfolio is minimized. The stocks with 0 weight will not be included in the portfolio.

Objective:

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

Similar to the Linear program in the previous method we are going to assign a pseudo variable y_t to the absolute value which will give rise to two constraints per each time t

Constraints:

1. $\sum_{i=1}^{n} w_i = 1$
2. $w_i \geq Mz_i$ (M=1)
3. $y_t + \sum_{i=1}^{n} w_i r_{it} \geq q_t$
4. $y_t - \sum_{i=1}^{n} w_i r_{it} \geq - q_t$
5. $\sum_{i=1}^{n} z_i = m$
6. $z_i = \{0,1\}$

Decision Variables:

- w_i is the weight of the stock i in the portfolio
- y_t is the return of the index at time t
- r_{it} is the return of the stock i at time t
- y_t is a pseudo variable assigned to the absolute value to convert the objective function to linear space
- z_i is a binary variable which represents whether stock i is present in the portfolio or not

Constraints Explanation:

1. The sum of the weights of all the stocks should be equal to 1.
2. All the weights should be greater than or equal to Mz_i. Here, as mentioned above z_i represents the presence of stock i in our portfolio. If the stock is present in our portfolio ($z_i = 1$) then the weight of the stock (w_i) will be less than or equal to 1. Else, the weight of the stock (w_i) will be less than or equal to 0 which will lead to that particular stocks' weight being 0 as weights cannot be negative. In the equation, M is equal to 1 as weights can never be greater than 1.
3. One of the constraints on y_t to be greater than the positive case of the absolute value function.

4. One of the constraints on y_t to be greater than the negative case of the absolute value function.
5. The number of stocks present in the portfolio should be equal to the predetermined number of stocks (m).
6. z_i should either be 0 or 1.

Code Snippet used for constructing the problem in Python:

```
M=1 # Big M is 1. Weights cannot and do not have to be greater than 1
A_3 = np.zeros((1+times*2+num+1,2*num+times))
A_3[0,times:times+num] = 1 # weights sum = 1
for i in range(times):
    A_3[i+1,i] = 1
    A_3[i+1,times:times+num] = port_r[i]
    A_3[i+1+times,i] = 1
    A_3[i+1+times,times:times+num] = -1*port_r[i]

for i in range(num):
    A_3[2*times+1+i,times+i] = 1
    A_3[2*times+1+i,num+times+i] = -1*M

A_3[1+times*2+num,times+num:] = 1 # no. of stocks in our portfolio constraint

b_3 = np.concatenate((np.array([1]),index_r,-1*index_r,np.zeros(num),np.array([0])))
b_3[1+times*2+num] = length
sense_3 = np.array(['='] + ['>']*2*times + ['<']*num + ['='])
obj_3 = np.array([1]*times + [0]*num*2)
#ub = np.concatenate((np.array([np.inf]*times),weight_con))
vtype_3 = np.array(['C']*(times+num) + ['B']*num)
```

✓ 0.1s

- **A_3** is the Left Hand side of our constraints
- **b_3** is the Rights Hand side of our constraints
- **sense_3** is the relationship between the LHS and RHS of our constraints
- **obj_3** is the coefficients of our decision variables in the Objective function
- **vtype_3** has the information of whether our decision variables are Continuous, Integers or Binary

III. Results

For the following analysis we used 2019 and 2020 prices data of NASDAQ-100 Index and the stocks present in this index. There are actually 103 stocks in the NASDAQ-100 Index but we only choose stocks for our portfolio from 100 of these as the other 3 do not have data in part of 2019. We used 2019 data for finding out the weights of the stocks selected for the portfolio and then used these weights to compare the returns of the index and the portfolio in 2020.

A. Stock Selection (IP) and Weight Calculations (LP)

We first selected our m to be 5 and checked which stocks are selected by our model using the 2019 data. The following stocks were selected:

Stocks Selected	Company	Industry
LBTYK	Liberty Global PLC	Telecommunications
MXIM	Maxim Integrated	Semiconductors
MSFT	Microsoft Corporation	Information Technology
VRTX	Vertex Pharmaceuticals Inc.	Healthcare
XEL	Xcel Energy Inc	Utilities

If we observe the stocks selected by our Integer program we realize that each of them belong to a different sector. With only 5 stocks in our portfolio as a limitation, we have to select the stocks in such a way that our portfolio is diversified. Using the results from the Integer program we move on to find the weights of the selected stocks by solving the linear problem explained above. The following are the results:

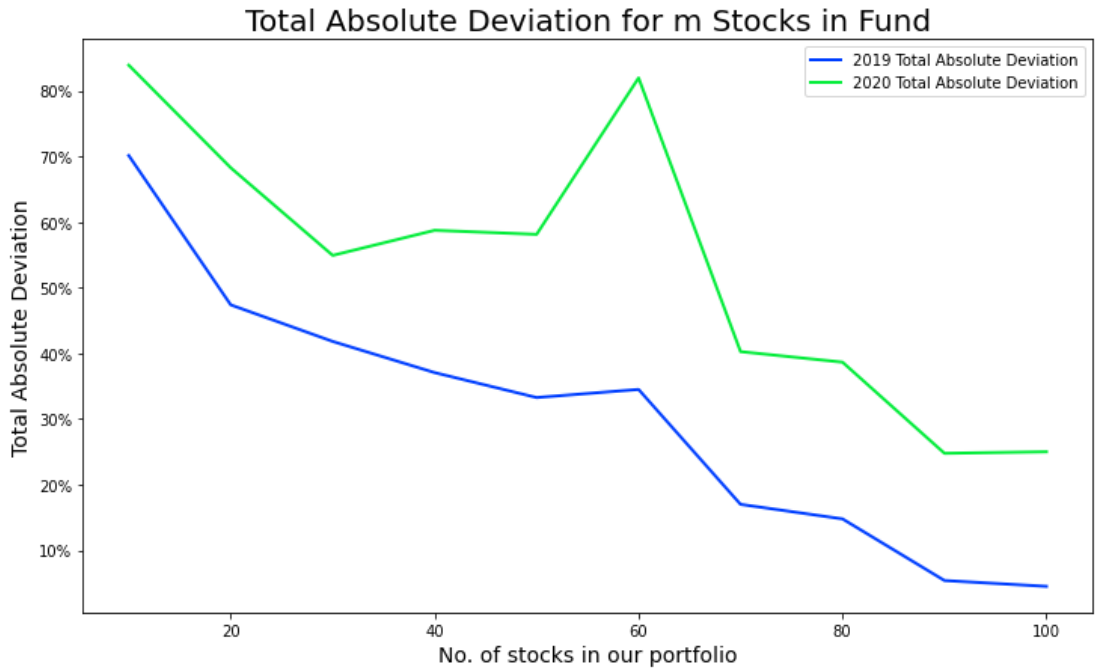
Stocks Selected	Weights
LBTYK	0.048862
MXIM	0.210388
MSFT	0.580352
VRTX	0.071190
XEL	0.089208

In the NASDAQ-100 index majority of the weight is constituted of Technology giants like Apple, Google, Microsoft, etc. In fact, Microsoft has the second highest weight in the actual index following Apple. Therefore, the weight given to Microsoft in the portfolio is actually making it mimic the Index. The following values are the sum of the absolute differences between the index returns and the portfolio returns with the above 5 stocks in it:

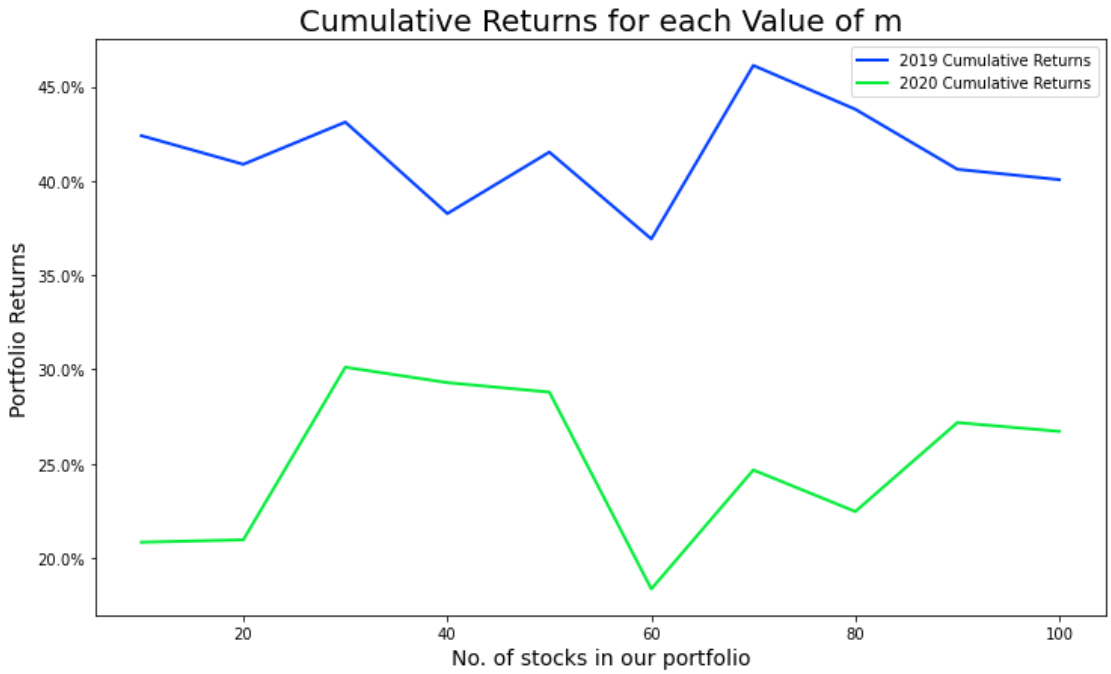
2019	2020
0.7892	0.8697

The differences are high considering that a return is of the order of 1/100th. But we only selected 5 stocks to be in our portfolio. Next, we repeated the same steps but kept changing the number of stocks from 10 to 100 with an increment of 10 to see how our model performs. The following is the result:

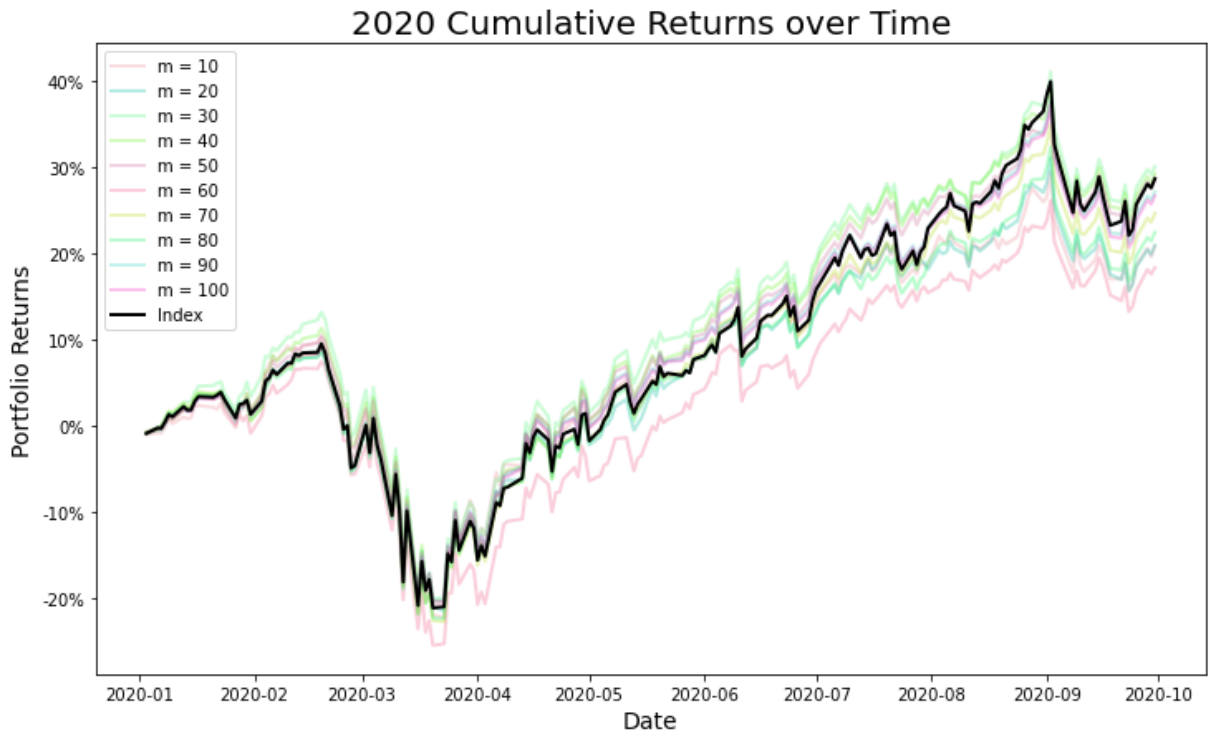
Number of Stocks Selected	Sum of absolute difference between the returns of the Index and Portfolio in 2019	Sum of absolute difference between the returns of the Index and Portfolio in 2020	Cumulative Portfolio Returns in 2019	Cumulative Portfolio Returns in 2020
10	0.686533	0.831317	0.426601	0.19981
20	0.473736	0.682573	0.408796	0.20959
30	0.418015	0.549085	0.431176	0.301145
40	0.370517	0.587312	0.382654	0.292968
50	0.33254	0.581148	0.415273	0.287943
60	0.34489	0.819424	0.369201	0.183519
70	0.169824	0.402497	0.461226	0.246624
80	0.147683	0.386431	0.437979	0.22461
90	0.053779	0.247582	0.406154	0.271778
100	0.044911	0.249943	0.400651	0.267112



We see a downward trend of the sum of absolute difference between returns for both 2019 and 2020 with an exception at m = 60. The difference between the Index and the Portfolio is always lesser in 2019 than in 2020 as we have constructed our weights by using 2019 data.



The returns in 2019 are consistently higher than the returns in 2020. 2020 returns are lower at all values m as the market in general was down due to the Pandemic. Also, variation in the 2020 returns is much more erratic than in 2019 with change in number of stocks.



B. Stock Selection and Weight Calculations (MIP)

Similar to the previous method we first selected our m to be 5 and checked which stocks are selected by our model and what weights are assigned to each of them using the 2019 data. The following stocks were selected:

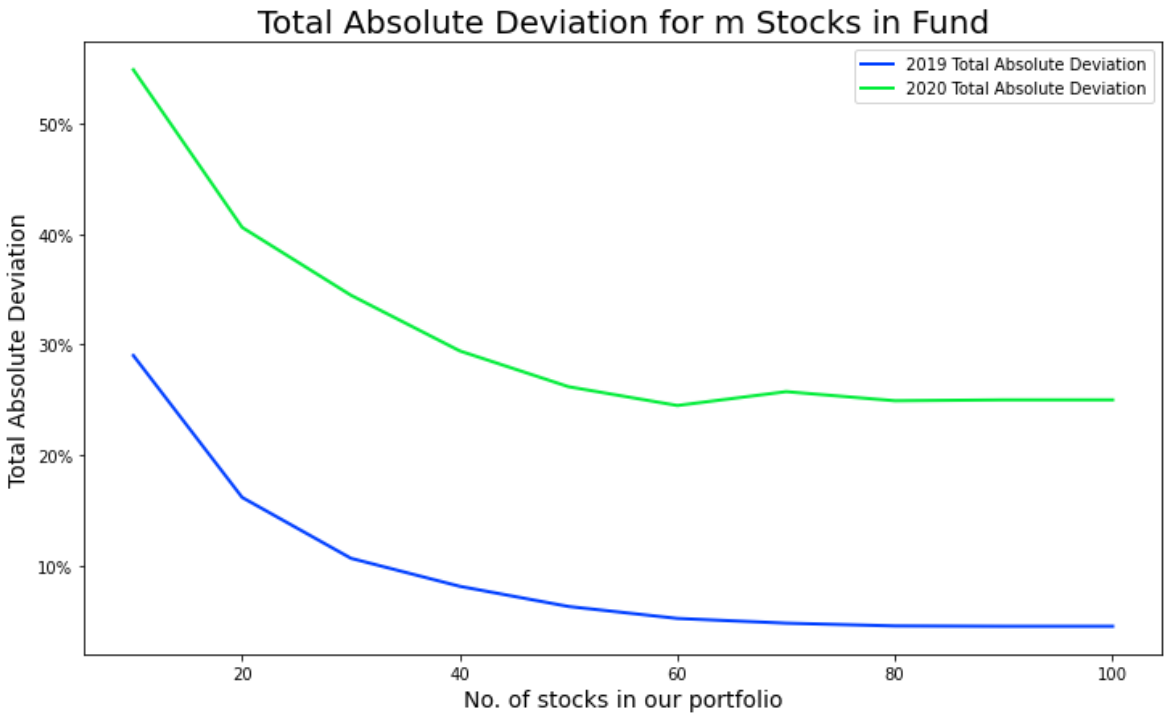
Stocks Selected	Company	Industry	Weights in Portfolio
AMZN	Amazon.com, Inc.	Multiple	0.250123
ADI	Analog Devices, Inc.	Semiconductors	0.113758
AAPL	Apple Inc.	Electronics	0.191692
MSFT	Microsoft Corporation	Information Technology	0.289869
MDLZ	Mondelez International Inc.	Food/ Beverage	0.154558

This time we see more technology companies with the weights more equally distributed. As the NASDAQ-100 Index has more weightage for technology companies it is reasonable to select the highest weighted Industry sector to imitate the Index. The following values are the sum of the absolute differences between the index returns and the portfolio returns with the above 5 stocks in it:

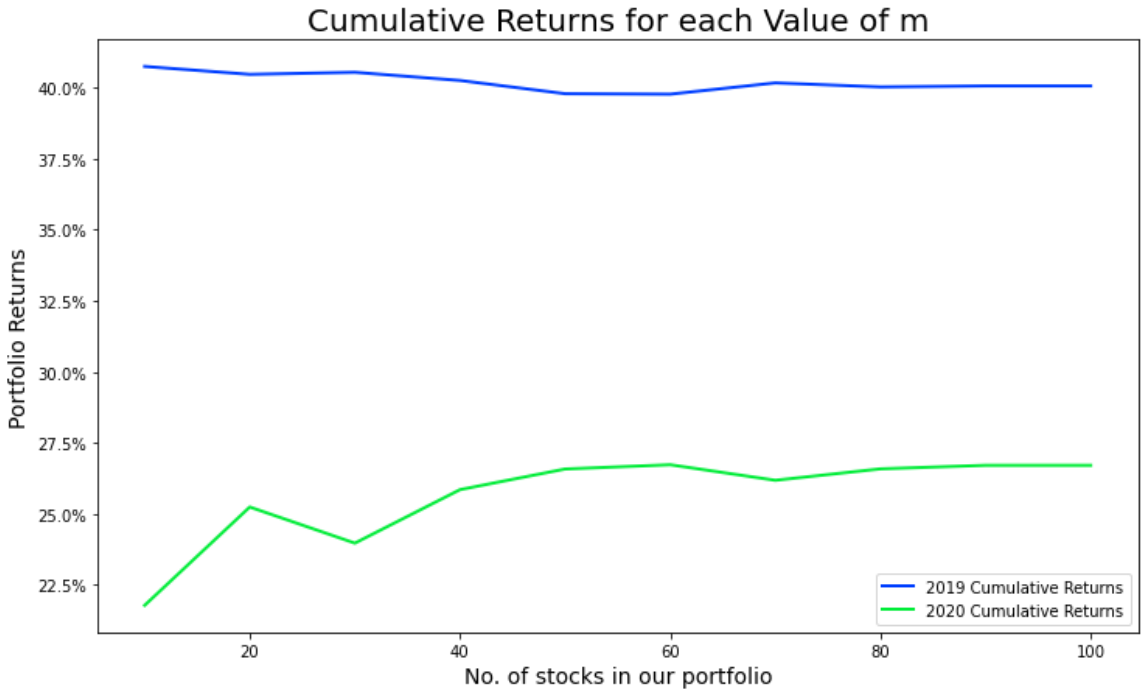
2019	2020
0.4993	0.5914

The differences are high considering that a return is of the order of 1/100th. But we only selected 5 stocks to be in our portfolio. Next, we repeated the same steps but kept changing the number of stocks from 10 to 100 with an increment of 10 to see how our model performs. The following is the result:

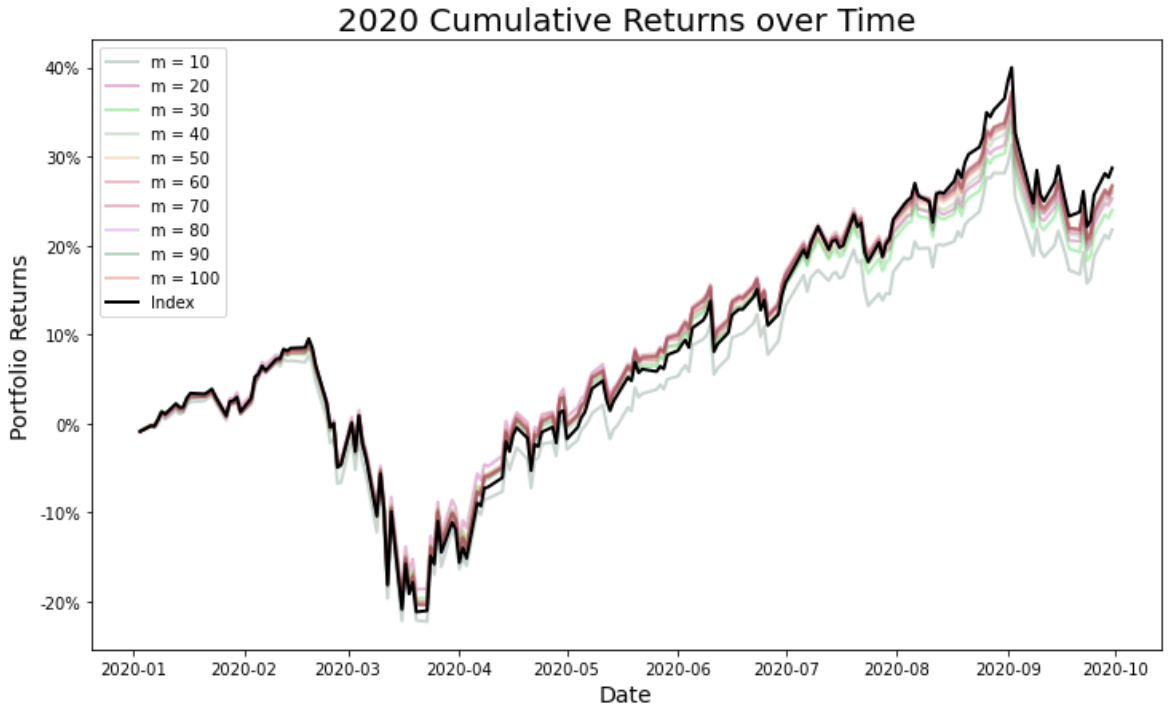
Number of Stocks Selected	Sum of absolute difference between the returns of the Index and Portfolio in 2019	Sum of absolute difference between the returns of the Index and Portfolio in 2020	Cumulative Portfolio Returns in 2019	Cumulative Portfolio Returns in 2020
10	0.290137	0.548939	0.407514	0.217832
20	0.161601	0.406139	0.404279	0.252433
30	0.106465	0.344686	0.405453	0.239714
40	0.0811	0.29428	0.402576	0.258543
50	0.06282	0.261847	0.397918	0.265812
60	0.052008	0.244844	0.397756	0.267311
70	0.047723	0.257388	0.401726	0.261843
80	0.045227	0.249143	0.400299	0.265862
90	0.044911	0.249943	0.400651	0.267112
100	0.044911	0.249943	0.400651	0.267112



We see a downward trend of the sum of absolute difference between returns for both 2019 and 2020. The difference between the Index and the Portfolio is always lesser in 2019 than in 2020 as we have constructed our weights by using 2019 data.



The returns in 2019 are consistently higher than the returns in 2020. 2020 returns are lower at all values m as the market in general was down due to the Pandemic. Also, variation in the 2020 returns is much more erratic than in 2019 with change in number of stocks.



IV. Conclusion

Our recommendation to maximize cumulative returns would be the MIP method for several reasons. Firstly, looking at the plots of cumulative returns for IP/LP and the MIP, we see that for the majority of m stocks selected, the MIP method tracks the index more closely, and this is corroborated when we take a look at the the plots of total absolute deviation for both methods, as the curves for both 2019 and 2020 are smooth over the range of m for MIP, where as it is very jagged for the IP/LP method. From these plots we can also see that IP/LP doesn't reach its lowest deviation until more than 80 stocks have been selected, where the MIP method begins to level out somewhere between 40 and 60 stocks selected. This accuracy comes at a cost though, as MIP is extremely time consuming and takes much longer than IP/LP.

Now that we have recommended a method for selecting weights, we need to determine how many stocks to select using MIP. Again looking at the total absolute deviation for the MIP method, we can see that somewhere between 40 and 60 stocks selected the deviation from the index levels out and from then on there isn't anything else to be gained from adding more stocks to the portfolio. Thus our recommendation would be to at most add 60 stocks to the MIP model to follow the index as closely as possible with the least computational time and energy required.