Optimization Project 2- Integer Programming Report: Stock Portfolio

Tyler Cushing - tmc3295
Kai Zhang - kz3967
Meeth Yogesh Handa - mh58668
Jason Nania - jn28878

Goal:

In our project, we created an index fund with m stocks to track the NASDAQ 100 index. The index fund will have m stocks, where m is substantially smaller than the size of the target population, n.

Approach:

To begin, we formulated an integer program that picks exactly m out of n stocks for our portfolio. This integer program will take as input a 'similarity matrix', which we will call ϱ. The individual elements of this matrix, ϱ_{ij} , represent similarity between stock i and j. Similarity measure used for our approach is the correlation between the returns of stocks i and j.

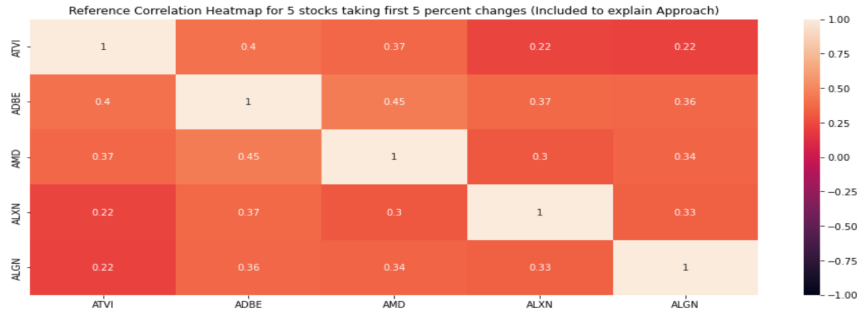
After finalizing the number of stocks to select and include, it is important to obtain the weights of individual stocks in the proposed portfolio. Finally, we evaluate how well our index fund does as compared to the reference index (NASDAQ-100 in this case) out of sample.

I. Correlation Matrix

To maximize our Fund's representation of the Index, we first created a correlation matrix for each stock pair. We did this by taking the daily change of each stock (R at T - R at T-1) and used the same to populate a n by n matrix with correlations of daily changes between each n stocks.

	ATVI	ADBE	AMD	ALXN	ALGN	GOOGL	GOOG	AMZN	AMGN	ADI	...	TCOM	ULTA	VRSN	VRSK	
ATVI	1.000000	0.399939	0.365376	0.223162	0.216280	0.433097	0.426777	0.467076	0.203956	0.329355	...	0.322906	0.128241	0.464850	0.316549	0.2
ADBE	0.399939	1.000000	0.452848	0.368928	0.363370	0.552125	0.540404	0.598237	0.291978	0.473815	...	0.360392	0.201151	0.711339	0.541243	0.4
AMD	0.365376	0.452848	1.000000	0.301831	0.344252	0.418861	0.417254	0.549302	0.151452	0.503733	...	0.332776	0.210623	0.498342	0.330900	0.2
ALXN	0.223162	0.368928	0.301831	1.000000	0.332433	0.315993	0.307698	0.363170	0.342022	0.317040	...	0.257143	0.408936	0.350581	0.191489	0.5
ALGN	0.216280	0.363370	0.344252	0.332433	1.000000	0.248747	0.250316	0.399281	0.264599	0.328280	...	0.175957	0.128559	0.360886	0.251855	0.3

Sample heat map taking a small number of stocks to depict varying levels of correlation.



II. Stock Selection

When selecting our stocks, our binary decision variable Y_j tells us which stocks, j, are in our fund. Y_j will be equal to 1 if the stock is included in our fund and Y_j will be equal to 0 if the stock is not chosen to be in our fund. Our binary decision variable X_{ij}, tells us which stock, j, best represents each stock i in the index. There are 100 Y_j variables and 10,000 X_{ij} variables.

a. Objective Function

The objective of our model is to maximize the similarity between n stocks and their representatives in the index fund. This is achieved by summing the product of similarities (P_{ij}) and binary X_{ij} result for each stock i in Index and each stock j in our Fund denoted as:

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

Our Python code looks like this:

```
def stock_selection (fin_matrix,m):
    sim_values = np.concatenate(fin_matrix.values)
    obj = np.array(list(sim_values) + [0]*len(fin_matrix))
    obj
```

b. Constraints

We have three types of constraints:

1. Total number of stocks m can be included for the Fund, denoted as: $\sum_{j=1}^{n} y_j = m.$ where Y_j is the binary decision variable.

   Our Python code for this constraint looks like this:

```
# first constraint
A[0][len(sim_values):] = 1
```

2. Each stock i in Index can only have one best representative j in the Fund denoted as:

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

where X_{ij} is the binary decision variable and n = 100.

Our Python code for this constraint looks like this:

```
# second constraint
for i in range(1,len(fin_matrix)+1):
    A[i][(i-1)*len(fin_matrix):(i*len(fin_matrix))] = 1
```

3. The third constraint ensures that stock i is best represented by stock j, but only if stock j is selected to the Fund, denoted as:

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

Our Python code for this constraint looks like this:

```
# third constraint
row = 1+len(fin_matrix)
for i in range(0,len(fin_matrix)):
    for j in range(0,len(fin_matrix)):
        A[row][i*len(fin_matrix)+j] = 1
        A[row][len(sim_values)+j] = -1
        row+=1
```

c. Results

The five stocks that are represented in our index fund based on the above constraints are shown below:

	stocks	chosen
56	LBTYK	1.0
59	MXIM	1.0
63	MSFT	1.0
94	VRTX	1.0
98	XEL	1.0

LBTYK (Liberty Global): Communications Sector
MXIM (Maximum Integrated) : Industrials
MSFT(Microsoft Corporation) : Information Technology
VRTX (Vertex Pharmaceuticals): Healthcare
XEL (XCEL Energy): Energy

III. Calculating Portfolio Weights

The next step we took is optimizing the proportion of the five stocks in the Fund while matching the returns of the index as closely as possible. We first calculated the daily return of the Index and five stocks we picked

X	NDX	LBTYK	MXIM	MSFT	VRTX	XEL
2019-01-03	-0.033602	-0.006280	-0.056845	-0.036788	-0.002133	-0.003940
2019-01-04	0.044824	0.068546	0.044004	0.046509	0.054785	0.009786
2019-01-07	0.010211	-0.042766	0.010388	0.001275	0.018240	-0.004330
2019-01-08	0.009802	-0.036122	-0.015026	0.007251	0.013307	0.011597
2019-01-09	0.007454	0.002959	0.027098	0.014300	0.023065	-0.007984
...	...	...	...	...	...	...
2019-12-23	0.002019	0.004617	-0.004213	0.000000	-0.003217	-0.017615
2019-12-24	0.000402	0.000919	-0.002604	-0.000191	0.002637	0.003913
2019-12-26	0.009058	-0.016988	0.006526	0.008197	-0.001406	-0.001114
2019-12-27	-0.000835	0.020551	-0.002431	0.001828	0.003269	0.005256
2019-12-30	-0.006983	-0.012357	0.001463	-0.008619	-0.009051	0.002693

250 rows × 6 columns

a. Decision Variable

W_i is the weight of each stock in the Fund. The total number of W_i variables changes with the change in number of stocks to be selected.

b. Objective

We want to minimize the total difference of daily Index return and Fund return across the time period, denoted as:

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

where q_t is the Index return on day t; W_i is the weight of each stock in the Fund; r_{it} is the return of stock i on day t; m = 5 stocks, T = 250 days for the initial formulation.

One obstacle that had to be overcome was to convert the above objective function into a linear program. This was done by introducing a placeholder variable y_i for each day in the time period which represented the difference between the index and fund returns. Additional constraints were added which restricted the value of y_i to be greater than both the positive and negative values of the index returns.

$y_i >= |q_t - w_i r_{it}|$

$y_i + w_i r_{it} >= q_t$

$y_i - w_i r_{it} >= -q_t$

c. Constraints

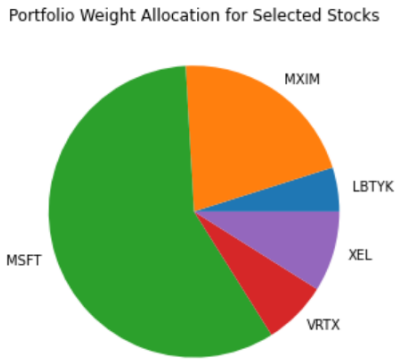
The five weights W_i are positive fractions and need to add up to 1, denoted as:

$$\sum_{i=1}^{m} w_i = 1 \qquad w_i \geq 0.$$

d. Results

We observed that MSFT (Microsoft) is a very large representative of other stocks, while other stocks, such as VRTX (Vertex Pharmaceuticals) only represent a small number of other stocks when m = 5.

	LBTYK	MXIM	MSFT	VRTX	XEL
0	0.048862	0.210388	0.580352	0.07119	0.089208



IV. Fund Performance on 2020 Index

Using the five stocks we have selected to be in our index fund from the 2019 data, we want to analyze how well our portfolio tracks the 2020 index. To begin, we use the percent changes of our five stocks in our index fund and start preparing our data.

The model did not perform well with selecting 5 stocks to track the 2019 index. The total error came out to be approximately 0.8969.

```
total_error = 0
for k in range(len(eval2020)):
    ind_val = eval2020.iloc[k][0]
    ret_val = eval2020.iloc[k][ss_list].to_list()

    total_error += abs(ind_val - np.dot(w2019,ret_val))
total_error
```

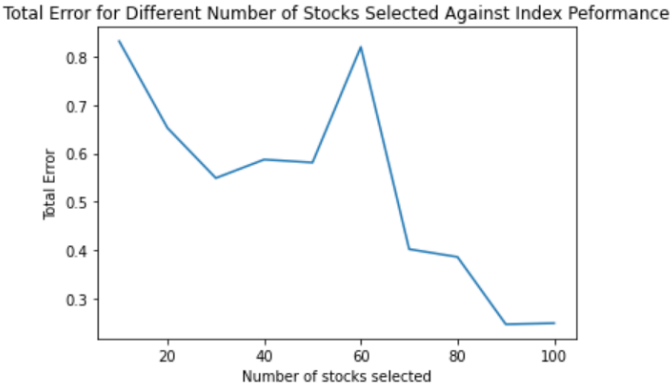
```
0.8696699433741908
```

V. Fund Performance with More Stocks

For this part we will add more stocks to our index fund to analyze how well our portfolio tracks the 2020 data. We first began with 5 stocks in our index fund. Now, we will see how our index performs with 10, 20, … , 90, and 100 stocks represented.

The lowest total error and the best performing model occurred when we had 90 stocks in our index fund. The total error where m=90 (90 stocks) was approximately 0.2475.

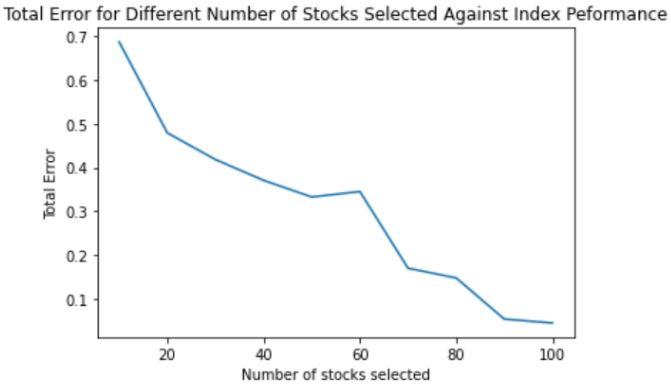
A graph below shows how the total error changes as the number of stocks selected, m , goes up from 10 to 100. We can see that when we selected 10 stocks, the total error was quite high, similarly to when we originally selected 5 stocks. Then as the number of stocks we selected increased, our model performed better as the total error began to decrease with each m. Suddenly, we can see a sharp increase in the total error when 60 stocks were selected. Following that, there was a drastic decrease in total error when 70 stocks were selected.




```
m = 10, total error is 0.8313165184500251
m = 20, total error is 0.6523378586362023
m = 30, total error is 0.5490849328816374
m = 40, total error is 0.5873120559530017
m = 50, total error is 0.5811479442370265
m = 60, total error is 0.8194235991971988
m = 70, total error is 0.40249695463776525
m = 80, total error is 0.3864307853572664
m = 90, total error is 0.24758163037063843
m = 100, total error is 0.24994327434030078
where m represents the number of stocks selected
```

Next, we did the same thing, but we will be using our index fund to analyze how well our portfolio tracks the 2019 data. The lowest total error and the best performing model occurred when we had 100 stocks in our index fund. This means that every stock was being represented by themselves as they were all chosen to be in our index fund. The total error where m=100 (100 stocks) was approximately 0.0449.

A graph below shows how the total error changes as the number of stocks selected, m , goes up from 10 to 100. We can see that when we selected 10 stocks, the total error was quite high once again. Then as the number of stocks we selected increased, our model performed better as the total error began to decrease gradually with each m. There is a small increase in the total error when m=60, but nothing as drastic as when we were comparing our fund to the 2020 data.




```
m = 10, total error is 0.686533011145812
m = 20, total error is 0.47883578791133574
m = 30, total error is 0.4180147550230183
m = 40, total error is 0.37051722255520575
m = 50, total error is 0.33254009291547576
m = 60, total error is 0.34488997906451385
m = 70, total error is 0.169823718275309
m = 80, total error is 0.1476825280009613
m = 90, total error is 0.05377918984547229
m = 100, total error is 0.044910816393603885
where m represents the number of stocks selected
```

The Total Error for 2019 and 2020 data

VI. Alternative Method for Fund Creation: Big M

In our implementation, we used a different approach as well which bypassed the aforementioned stock selection problem by adding extra binary variables Y_i for each stock and added a big M constraint including the Y_i and W_i with W_i representing the weight of each stock in the portfolio.

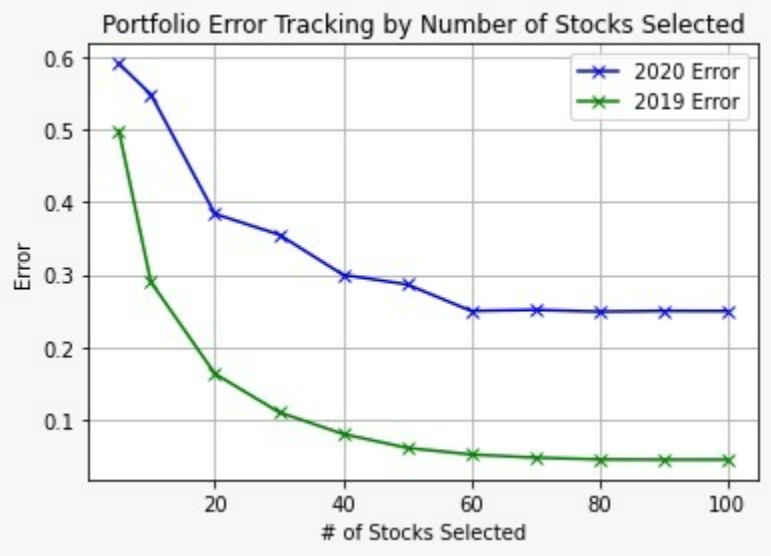
$$w_i <= M * Y_i$$
$$w_i - (M * Y_i) <= 0$$

Thus, we have n additional constraints when devising the optimization problem. M = 1 in our implementation since the weight of an individual stock cannot exceed the total sum of weights which should be 1.

The above optimization problem takes extremely long for Gurobi to solve and hence, we set a cut off time limit of 1 hour (3600 seconds) for each value of m (number of stocks in Fund).

Number_c	ATVI	ADBE	AMD	ALXN	ALGN	GOOGL	GOOG	AMZN	AMGN	ADI	ANSS	AAPL	AMAT	ASML	ADSK	ADP	BIDU	BIIB	BMRN	BKNG	AVGO	CDNS
5	0	0	0	0	0	0	0	0.250123	0	0.113758	0	0.191692	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0.093503	0	0.125808	0.076823	0	0	0.138716	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0.094677	0.103361	0.030434	0.034716	0	0.118955	0	0.022275	0	0	0	0	0	0	0	0
30	0.009001	0.021307	0	0	0	0.085988	0	0.092432	0.021561	0	0	0.111675	0	0	0	0.028602	0.007902	0	0	0.014356	0.015289	0
40	0	0.022593	0.006223	0	0	0.084434	0	0.101973	0.012594	0	0	0.109932	0	0.014897	0	0.031622	0.009891	0.00649	0	0.015669	0.013547	0
50	0.005319	0.023629	0.004325	0	0	0.08357	0	0.098042	0.012086	0	0	0.106664	0	0	0	0.028845	0.003298	0.006427	0.009396	0.013831	0.016852	0
60	0	0.01621	0.002794	0	0	0.086696	0	0.095965	0.016079	0	0	0.102822	0	0	0.005467	0.01501	0.005101	0.006034	0.002621	0.013929	0.012766	0
70	0.003455	0.01823	0.003072	0	0	0.088768	0	0.097141	0.014023	0.004316	0	0.100068	0.007393	0	0.005614	0.017737	0.006496	0.007654	0.003296	0.012445	0.012629	0.006639
80	0.003119	0.017845	0.003566	0.001275	0	0.057147	0.027985	0.096184	0.01427	0	0	0.102581	0.004063	0.002597	0.003408	0.014886	0.005873	0.006792	0.004444	0.013507	0.012524	0.00431
90	0.004279	0.017198	0.004016	0.002467	0	0.056237	0.030318	0.095571	0.013615	0	0	0.102006	0.003509	0.001401	0.003818	0.014701	0.005131	0.006661	0.003984	0.01137	0.012906	0.003836
100	0.004223	0.01714	0.004037	0.00245	0	0.056829	0.029717	0.095613	0.013637	0	0	0.102011	0.003552	0.001353	0.003923	0.014759	0.00513	0.00666	0.003962	0.011347	0.012871	0.003818

The results give us a fairly accurate representation of the most optimal solution. We saved the output of the stock weights into a csv file and then used the data to test the performance against the Index for 2019 and 2020.



When testing the performance of the Fund obtained from this method against the index, we found that while the Error (Sum of Absolute Error) reduced to values close to 0 on the train 2019 data and plateaus just below 0.3 for the test 2020 data.

VII. Recommendation

The maximum Total Error for both 2019 and 2020 stock data is less when the Alternative Big M method is used for portfolio creation. The alternative approach tends to perform better for most values of m when compared to the initial approach which included 2 separate optimization problems: 1 for stocks selection and 1 for the selected stocks portfolio weights.

However, it is observed that the minimum test error on the 2020 data is just below 0.3 for both methods indicating that as the value of m approaches the total number of stocks in the data, the errors tend to reach a threshold and flatten out.

Our recommendation is to go with the alternative big M method for Fund creation to track the Index performance based on the results obtained from our analysis. The ideal number of stocks to select in this case is close to half of the total number of stocks in the data (m = 50/60 in our case).