Project 2 Report

Group 9: Shengxiang Wu, Ohyoon Kwon, Kaushik Kumaran, Raymond Li

Introduction:

We created a fund to select m stocks from the NDX index. Our goal was to make the index perform as similar to the NDX as possible. Therefore, we were trying to minimize the everyday difference in return between our fund and the NDX. We approached this goal using two different methods. Our first method was to select m stocks with the highest correlation to other stocks in 2019, assign weights to them, and then measure the performance of our fund using the 2020 data. The second method was to skip the stock selection, directly assign weights to m stocks, find the combination that could minimize the difference in returns compared to the NDX, and then measure the performance using the 2020 data.

Procedure

Method 1:
First, we selected m stocks from the index, so we have $\Sigma Y_j = m$ as our constraints, where Y_j is a binary variable representing whether or not stock j is included in the fund.

For every stock in the index, the stock is mapped to one and only one stock in our fund. Therefore, for every stock in the fund, there could be up to 100 maps. Hence, we will have 100 * 100 possible mappings, which leads to 10000 constraints in our constraints matrix: $\Sigma X_{ij} = 1$, for all X_{ij} mapping to Y_j.

Finally, we had $\Sigma X_{ij} < Y_j$ because the mappings will not happen if Y_j is not selected. Thus, we had 100 * 100 variables for X, and 100 variables for Y. The shape of our matrix A is (10101, 10100). Solved by Gurobi, we obtained m stocks representing most of the performance of stocks in the index with the highest correlation.

We created another optimization matrix to solve for the weights. We simply created 250 dummy variables to represent the absolute value of daily return differences. Based on the math,

$$min \sum |X - Return| = min \sum Y$$

Such that

$$Y > Return - X \; and \; Y > X - Return.$$

Therefore, we had 250 * 2 = 500 constraints and 250 dummy variables.

In addition, we needed m variables to represent the weights based on the m stocks selected, and one additional constraint because the weights for all m stocks sum up to 1. Therefore, our A matrix had the shape (501, 250 + m), and our objective function was trying to minimize the sum of the 250 dummy variables. Solving this with Gurobi, we then compared the

performance in 2019 to the performance in 2020 by assigning weights to the same stocks through a loop, where $m = 10, 20, 30, \ldots, 100$.

Method 2:
This method performed better, but took a long time to run.

In this method, we also had 100 Y variables to represent whether or not the stock was selected, 100 X variables to represent the weights of stocks, and 250 dummy variables to minimize the absolute value of daily return differences.

We had a total of 602 constraints. From the 602 constraints, just like in method 1, 250 * 2 = 500 constraints were dummy variables representing the absolute value. 100 constraints were used to limit $X < Y$ for all X_i and Y_i because X_j cannot have weights greater than 0 if stock i, represented by Y_i, was not selected. In addition, we had two more constraints because we needed to limit the sum of weights of all X to 1 and the sum of all Y to m. Our objective function was to minimize the sum of the 250 dummy variables.

Therefore, we had the constraint matrix A with the shape (602, 450), and spent approximately 11 hours solving it in a loop using Gurobi for $m = 5, 10, 20, \ldots, 100$. After solving the optimization problem, we compared the performance in 2019 to the performance in 2020 data under different m's.

Result

Method 1:
The result of $m = 5$ is shown below, with the selected stocks and assigned weights. We compared the performance in 2019 to the performance in 2020 using the sum of daily return differences for 250(251) days.

```
print('The performance of 2020, m = 5 is: ')
h
```

```
The performance of 2020, m = 5 is:

1.112437345507646
```

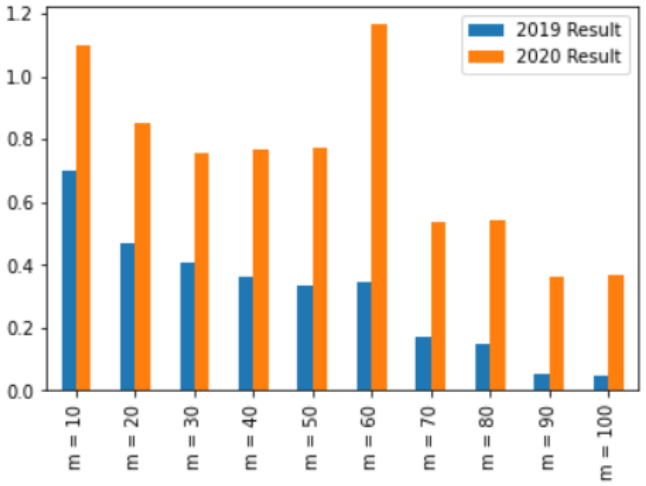
```
print('The performance of 2019, m = 5 is: ')
npvMod.objVal
```

```
The performance of 2019, m = 5 is:

0.7891782824631451
```

	Stock	Weights
0	LBTYK	0.048862
1	MXIM	0.210388
2	MSFT	0.580352
3	VRTX	0.071190
4	XEL	0.089208

The results for $m = 10, 20, \ldots, 100$ are shown below. The performance differences were measured by the sum of daily return differences.



	2019 Result	2020 Result
m = 10	0.701218	1.100511
m = 20	0.466233	0.853540
m = 30	0.407021	0.755453
m = 40	0.363281	0.767318
m = 50	0.334010	0.772208
m = 60	0.343167	1.164219
m = 70	0.169113	0.538227
m = 80	0.148219	0.543650
m = 90	0.053779	0.364608
m = 100	0.044911	0.365480

Method 2:

The result of $m = 5$ is shown below, with the selected stocks and assigned weights. We compared the performance in 2019 to the performance in 2020 using the sum of daily return differences for 250(251) days.

```
print('Result 2020 when m = 5: ')
h
```

```
Result 2020 when m = 5:
0.7773624845503466
```

```
print('Result 2019 when m = 5:')
npvMod.objVal
```

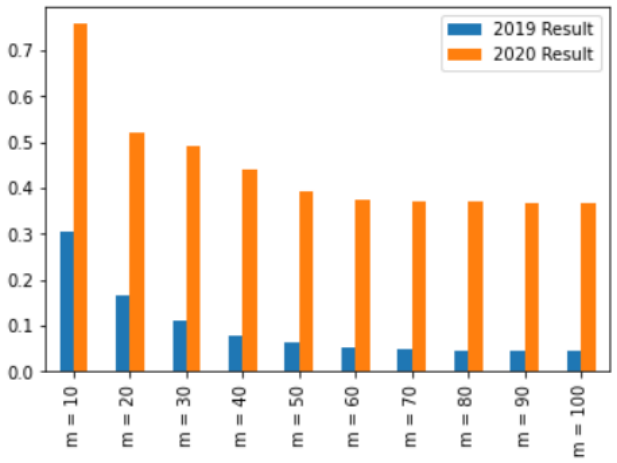
```
Result 2019 when m = 5:
0.4992586920546347
```

	Stock	Weights
7	AMZN	0.250123
9	ADI	0.113758
11	AAPL	0.191692
63	MSFT	0.289869
65	MDLZ	0.154558

The results for $m = 10, 20, \ldots, 100$ are shown below. The performance differences were measured by the sum of daily return differences.

```
df_compare.plot(kind = 'bar')
```

```
<AxesSubplot:>
```



	2019 Result	2020 Result
m = 10	0.304563	0.757166
m = 20	0.166750	0.521325
m = 30	0.111748	0.491365
m = 40	0.079414	0.438917
m = 50	0.064419	0.392651
m = 60	0.052752	0.374335
m = 70	0.048027	0.371565
m = 80	0.045227	0.370629
m = 90	0.044911	0.368682
m = 100	0.044911	0.368671

The results show that our portfolio was performing much better when we were using the 2019 data. The difference in daily returns between our portfolio and NDX, calculated using the 2020 data with weights extracted from the 2019 data, is quite large compared to our performance in 2019. We believe that the difference is caused by weights being assigned to the stocks. There is more than one way to assign weights to 100 stocks in order to have very similar daily returns in a known year, but things will change in an unknown year. From our results, we can see that as m increases, our portfolio performs closer and closer to the NDX. When $m = 100$, for both methods, our portfolio performs very much the same as the NDX (the average of daily return difference is 0.00018). However, it increased to around 0.0015, because the weights assigned to stocks in the fund were different from the weights assigned in the NDX. Thus, even if our fund performed similarly to the NDX in 2019, it can perform differently from the NDX in 2020.

Conclusion

Our goal was to build a fund with a selected amount of stocks to perform as similarly as possible to the NDX. Compared to using method 1 as our approach, our fund can have a more similar performance measured by both 2019 and 2020 data using method 2 as our approach. In addition, the performance measured by both 2019 and 2020 data merely increased by increasing the number of selected stocks after selecting 60 stocks. So, our

```
print(list(resultmm['Stock']))
```

```
['ATVI', 'ADBE', 'AMD', 'GOOG', 'AMZN', 'AMGN', 'AAPL', 'AMAT', 'ADSK', 'ADP', 'BIDU', 'BIIB', 'BMRN', 'BKNG', 'AVGO', 'CDN
S', 'CHTR', 'CTAS', 'CSCO', 'CMCSA', 'COST', 'CSX', 'DLTR', 'EBAY', 'EA', 'EXC', 'FB', 'GILD', 'ILMN', 'INTC', 'INTU', 'ISR
G', 'JD', 'KHC', 'LULU', 'MAR', 'MXIM', 'MCHP', 'MU', 'MSFT', 'MDLZ', 'NTES', 'NFLX', 'NVDA', 'ORLY', 'PYPL', 'PEP', 'QCOM',
'REGN', 'SIRI', 'SBUX', 'TMUS', 'TTWO', 'TSLA', 'TXN', 'ULTA', 'VRSN', 'VRTX', 'WBA', 'XLNX']
```

suggestion is to create a fund with 60 stocks selected from the index using method 2 as the approach.
These are the 60 stocks we should pick to build our fund. In addition, relatively larger weights will be assigned to the following 10 stocks.

```
resultmm1.index = resultmm1['Stock']
resultmm1.plot(kind = 'bar')
```

```
<AxesSubplot:xlabel='Stock'>
```