Optimization Project 2 report

Index Fund Creation and Performance Analysis

Group Members:
Anudeep Akkana(aa92799)
Muskan Agarwal(ma64547)
Tanvi Dalal (trd878)
Xinyu Liu (xl6598)

Objective

A market index consists of a specific number of stocks with assigned weights which assess market performance of a broad market population. A market index-like fund can be created as a portfolio by purchasing shares of all stocks in an index fund with the same weight as the index. However, the implementation can be impractical and expensive. The performance of the market index can be imitated by creating a **portfolio of 'm' stocks (Index Fund)** within a reasonable margin of a performance difference with the index.

This study aims to create a portfolio of 'm' stocks, an index fund that can track the NASDAQ-100 index through 'm' stocks, optimizing for portfolio performance. Our company will then decide how many of each chosen stock to buy for our portfolio to compare to the index.

Approach:

To create an index fund portfolio representing the NASDAQ-100 index, we need to minimize the difference between the returns of the m stocks and the index. Our company decided to try two methods:

1. Select m stocks first (IP), and then allocate optimal weights to selected stocks (LP).
2. Ignore stocks selection IP, reformulate weight selection problem to MIP that constrains the number of non-zero weights to be an integer.

Method 1:

Step 1: Stock Selection

We first calculate the daily returns for the index and the component stocks in both the 2019 and 2020 datasets, which is accomplished by:

```
: return2019 = stocks2019.iloc[:,1:].pct_change(axis='rows')[1:]
  return2020 = stocks2020.iloc[:,1:].pct_change(axis='rows')[1:]
```

Second, in order to know which stocks are most similar and able to represent the index, we create the similarity matrix to represent the correlation between each pair of stocks using the daily returns data frame from the previous step.

	ATVI	ADBE	AMD	ALXN	ALGN	GOOGL	GOOG	AMZN	AMGN	ADI	...
ATVI	1.000000	0.399939	0.365376	0.223162	0.216280	0.433097	0.426777	0.467076	0.203956	0.329355	...
ADBE	0.399939	1.000000	0.452848	0.368928	0.363370	0.552125	0.540404	0.598237	0.291978	0.473815	...
AMD	0.365376	0.452848	1.000000	0.301831	0.344252	0.418861	0.417254	0.549302	0.151452	0.503733	...
ALXN	0.223162	0.368928	0.301831	1.000000	0.332433	0.315993	0.307698	0.363170	0.342022	0.317040	...
ALGN	0.216280	0.363370	0.344252	0.332433	1.000000	0.248747	0.250316	0.399281	0.264599	0.328280	...
...	...	...	...	...	...	...	...	...	...	...	...
WBA	0.218149	0.228106	0.281950	0.192720	0.219595	0.232900	0.230603	0.288168	0.194490	0.347861	...
WDAY	0.311659	0.650430	0.407626	0.416396	0.308968	0.379493	0.371826	0.424748	0.211712	0.351734	...
WDC	0.303077	0.361516	0.438892	0.289908	0.284407	0.328619	0.322110	0.419620	0.172623	0.602935	...
XEL	0.043389	0.207403	0.017283	0.047947	0.088059	0.059930	0.052570	0.076724	0.137857	-0.047259	...
XLNX	0.249667	0.289497	0.478010	0.200356	0.253934	0.221983	0.213764	0.389871	0.092808	0.687646	...

We formulated an integer problem to solve the task of stock selection, the details are as follow:

Decision Variables:

Y_j - indicates stock j from the index are present in the fund or not (binary variable)
X_{ij} – indicates stock j in index is the most similar stock i (binary variable)
Total Decision variables - n^2+n (n: number of component stocks in the index)

Objective:

We need to select m stocks to be held in the fund. Our objective here is to maximize the similarity between all stocks in the index and their representatives in the fund, with the objective function:

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

Constraints:

1. Exactly **m** stocks to be held in the fund

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

2. Each stock **i** has exactly one representative stock **j** in the index.

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

3. Stock **i** is best represented by stock **j** only if **j** is in the fund

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

Total Constraints – n^2+n+1 (n: number of component stocks in the index)

Optimal Decision:

The component stock with the highest number of related stocks with a low value of m indicates the relative importance of holding them in an index fund.

For **m=5**, the optimal stocks for the index found are **LBTYK, MXIM, MSFT, VRTX, and XEL.**

Step 2: Weight adjustment

Objective:

We formulated a linear program to identify the optimal weights and minimize the difference in returns between the index and portfolio. The objective function can be written as:

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

Where:

r_{it} : the return of stock i (where stock i is one of the chosen stocks) at time period t

q_t : the return of the index at time t

w_i: weight of stock i in the portfolio

Since the objective function is non-linear, we can translate the problem into

$$\min_{y} \sum_{t=1}^{T} y_t$$

Decision Variables:

w_i - weight of stock **i** selected for the portfolio

yt – difference between index return at t and weighted return of selected indexes

Total Decision variables - m + number of transactions in 2019 return dataset

Constraints:

1. Sum of weights for w_i for m stocks in the fund equals 1

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

2. New constraints added by transferring the non-linear program to the linear program

$$y_t \geq q_t - \sum_{i=1}^{m} w_i r_{it}$$

$$y_t \geq \sum_{i=1}^{m} w_i r_{it} - q_t$$

Total Constraints = 1 + 2 * number of transactions in 2019 return dataset.

Optimal Decision:

Here each weight w_i suggests to use the percentage of the portfolio that is dedicated to stock "i" within the fund. By investing in accordance with optimized w_i on selected stocks, the difference in returns between the index and the fund will be minimized.

Based on the 2019 return data, the weights for each stock selected are MSFT: 58.04%, MXIM: 21.04%, XEL: 8.92%, VRTX: 7.12%, LBTYK: 4.89%.

```
pd.DataFrame({'Stocks':stock_select[1:],'Weight':lpMod_x.x[-m:]})
```

	Stocks	Weight
0	LBTYK	0.048862
1	MXIM	0.210388
2	MSFT	0.580352
3	VRTX	0.071190
4	XEL	0.089208

So, when we have m=5, the stock selection and weights can be summarized as below:

S No.	Stock	Company	Sector	Weight	%Contribution	No. of stocks it is representing
1.	**LBTYK**	Liberty Global PLC Class C	Communication Services	0.04886	**5%**	3
2.	**MXIM**	Maxim Integrated Products Inc	Electronics	0.21038	**21%**	30
3.	**MSFT**	Microsoft Corporation	Information Technology	0.58035	**58%**	53
4.	**VRTX**	Vertex Pharmaceuticals Incorporated	Healthcare	0.07119	**7%**	10
5.	**XEL**	Xcel Energy Inc.	Utilities	0.08920	**9%**	4

We can see the sector information technology has 58% contribution, similar to 53% in the NASDAQ-100.

By using the 5 selected stocks and their corresponding weights, we get the difference in returns in 2020 as **0.8697**.

```
weight = lpMod_x.x[-m:]

sums = 0
for i in range(len(return2020_chosen)):
    sums += abs(return2020_chosen.iloc[i,0]-weight@return2020_chosen.iloc[i,1:])
print(sums)
```

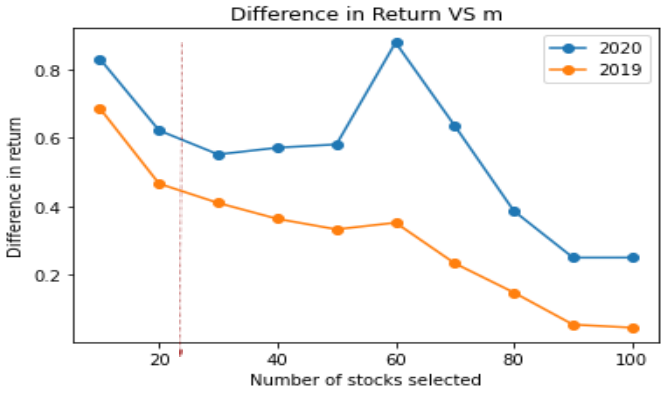
```
0.8696699433741902
```

The higher the difference, the lower the performance. The difference is high, because it is hard to use a low amount of stocks to represent a large market index of 100 stocks.

<u>Out-of-sample vs In-sample Performance by using different m</u>

We understand that changing the number of stocks in the portfolio, changes the performance. We need to identify the optimal number of stocks to be included in the portfolio to match the returns of the index as closely as possible.

By tracking the performance of the portfolio by using m from 10 to 100, the plot shows the relationship between the difference in the return and the number of stocks selected(m).



M = 30

Based on the graph:

1. **The error (difference) constantly decreases** till 50 stocks selected **in 2019 (in-sample data)**. After, the performance starts to decrease constantly from 60 stocks.

2. **Performance of the portfolio in 2020** (out-of-sample data): The error (difference) decreases until the stock equals 30. After that, the difference in return starts to increase with a spike at 60 stocks, which implies that performance is diminishing by adding stocks from 30 to 60. This could be due to overfitting the in-sample data, hence

performance of out-of-sample data does not follow the trend of in-sample data. After 60 stock counts, the error (difference in return) decreases again till 100 stock counts.

Since 2019 stocks return data is used for the selection of stocks and the weights, the fund performance is much better for 2019 (in-sample) than for 2020 (out-of-sample) data.

Based on the analysis above, by using method 1, the optimal m is 30, with the result as follows:

```
When m = 30
[0, 1, 5, 12, 15, 17, 19, 28, 30, 34, 35, 36, 46, 51, 53, 56, 57, 59, 63, 64, 66, 72, 75, 76, 77, 86, 88, 91, 94, 98]
30 stocks selected: ATVI ADBE GOOGL AMAT ADP BIIB BKNG CTXS CMCSA DXCM DOCU DLTR ILMN JD KHC LBTYK LULU MXIM MSFT MRN
A MNST PCAR PEP PDD QCOM TMUS TSLA ULTA VRTX XEL
Weights for stocks:  [0.02098562 0.04838005 0.15151172 0.03455038 0.00081464 0.00837358
 0.05553826 0.         0.05300536 0.01758276 0.         0.01493762
 0.00731477 0.03093167 0.0081     0.00070229 0.02004791 0.08364164
 0.21767601 0.00672114 0.00405035 0.03831927 0.06464688 0.00559495
 0.00973464 0.00868326 0.03327205 0.01561865 0.02220974 0.01705479]
How well this portfolio tracks index in 2020:  0.5518333454890046
How well this portfolio tracks index in 2019:  0.4097923638636843
```

Method 2:

In this method, we found the weights for ALL stocks available in the index and re-formulated the problem into an MIP that constrains the number of non-zero weights equal to the stocks to be selected, i.e., m.

Objective:

We are aiming to minimize the sum of absolute differences between the return of the index and the return of stocks selected. The objective function is:

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

Where:

r_{it} : the return of stock i (where stock i is one of the chosen stocks) at time period t

q_t : the return of the market index at time t

w_i: weight of stock i in the portfolio

Similar to method 1, since the objective function is nonlinear, we can translate the problem into

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

Decision Variables:

w_i - weight of stock **i** selected for the portfolio (continuous variable)

z_t - difference between index return at t and weighted return of selected indexes (continuous var)

y_j - represent if stock j presents in the portfolio or not (binary variable)

Total Decision variables - 2p + q
Where:
p: number of transactions in 2019 return dataset
q: number of component stocks in the index (100 in our case)

Constraints:

1. Sum of weights for w_i for m stocks in the fund equals 1

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

2. Only m stocks in the index must be held in the fund

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

3. Big M constraint

$$w_i - M y_i \leq 0$$

4. Extra constraints added from translating the non-linear optimization problem to linear

$$z_t \geq q_t - \sum^{m} w_i * r_{it},$$

$$z_t \geq -(q_t - \sum_{i=1}^{m} w_i * r_{it})$$

Total constraints - 2p + q + 1 + 1

Where:
p: number of transactions in 2019 return dataset
q: number of component stocks in the index (100 in our case)

Optimal Decision:
For m = 5, based on the 2019 return data, the weights for each stock selected in the previous steps are MSFT: 28.99%, AMZN: 25.01%, AAPL: 19.17%, MDLZ: 15.46%, ADI: 11.38%

```
pd.DataFrame({'Stocks':stock_select,'Weight':weight})
```

	Stocks	Weight
0	AMZN	0.250123
1	ADI	0.113758
2	AAPL	0.191692
3	MSFT	0.289869
4	MDLZ	0.154558

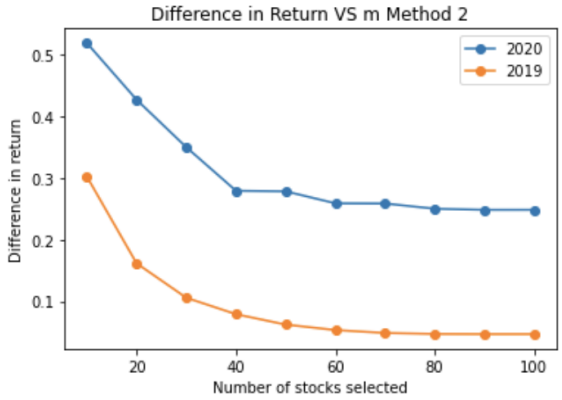
By using the 5 selected stocks and their corresponding weights, the difference in returns in 2020 we get is 0.5914, which is significantly lower than the result from method 1.

```
sums = 0
for i in range(len(return2020_chosen)):
    sums += abs(return2020_chosen.iloc[i,0]-weight@return2020_chosen.iloc[i,1:])
print(sums)
```

```
0.5913979244173762
```

Out-of-sample vs In-sample Performance by using different m

By tracking the performance of the portfolio by using m from 10 to 100, the plot below shows the relationship between the difference in the return and the number of stocks selected in the portfolio.



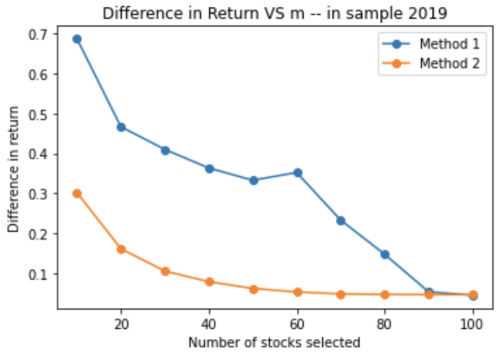
Based on the graph:

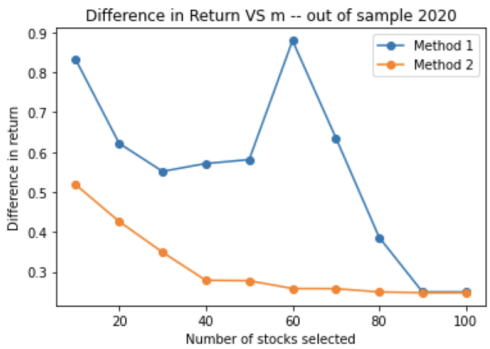
1. The in-sample error (2019 data) continues to decrease as m increases until 60 stocks and then remains almost constant.

2. The out-of-sample error (2020 data) decreases dramatically from m=10 to 40. The performance does not change much after as m continues to increase.

Since here also 2019 stocks return data is used for the selection of stocks and weights, so the fund performance is better for 2019 than for 2020. Even though the error keeps decreasing as m increases, the optimal m shouldn't be too large. Besides that, the error rate does not decrease much after around 40 to 50 stocks, we can conclude that by using method 2, the optimal number of stocks picked is around 40 to 50.

Comparison between Method 1 and Method 2





By looking at in-sample and out-of-sample performance separately, we can see that **method 2 generates better and smoother performance than method1.** The detailed performance of in-sample and out-of-sample by using method 1 and 2 follows:

	m	Method1_Eval2020	Method1_Eval2019	Method2_Eval2020	Method2_Eval2019
0	10	0.831317	0.686533	0.518220	0.301862
1	20	0.622036	0.466268	0.426428	0.161396
2	30	0.551833	0.409792	0.349524	0.105653
3	40	0.571485	0.363281	0.278946	0.079009
4	50	0.581148	0.332540	0.278010	0.062122
5	60	0.879273	0.352056	0.258568	0.053343
6	70	0.635728	0.233143	0.258282	0.048553
7	80	0.386431	0.147683	0.249828	0.046996
8	90	0.249824	0.053827	0.247986	0.046745
9	100	0.249943	0.044911	0.248013	0.046745

Conclusion and Recommendations:

1. **Method Choice**
   Between the two methods, method 1 overall has a higher error rate with more fluctuations than method 2, hence **method 2 (BigM method) provides more consistent and better performance**, mainly because method 2 solves the problem by trying to find the stocks selected and the weights for each stock at the same time.

   If processing time and computing power is a concern, Method 1 should be used.

2. **Number of stocks picked and weights for each stock**
   Considering the results above, we believe picking 40-50 stocks generates the best performance. Since the error rate does not decrease much after, the extra cost of adding extra stocks to the portfolio is not worthwhile.

Stocks recommended: MSFT, AMZN, AAPLE, GOOGL, and FB. The weights for these stocks are always at the top among all the stocks, holding a large portion of the NASDAQ-100 index, representing it well.