TEXAS
The University of Texas at Austin

RM 294: Optimization Project 2 Report

Team Members:

- Aishwarya Sarkar - as99646
- Benjamin Kanarick - bjk2437
- Nicolay Huarancay - nh23865
- Rochan Nehete - rrn479

Index

Problem Overview

Question at Hand

In today’s world, the information available to both institutional and individual investors is unlimited, and everyone is trying to maximize returns while minimizing both risk and initial outlay. Currently, we see two common schools of investing, those that are passive and those that are active. Passive investing consists of many ‘hands-off’ approaches, one being ‘indexing’ where the name of the game is to create/choose a portfolio that reflects the actions of a market population or specific index.

The demanding aspect is constructing these tracking portfolios while avoiding both frequent and large transaction costs due to price movements and the need for rebalancing. Due to this reason, it is desirable to construct a replicating/tracking portfolio with (n) assets that capture most of the movement in the target portfolio/sector while including as few securities (m) as possible.

Analysis of Specifics

1.

Approach

In order to tackle this problem of replicating the NASDAQ-100 index with as few m stocks as possible, we have created an integer program that picks a designed m out of n securities in the index. Next, we will have the code optimized and solve for the 'best' securities on the basis of covering the overall movements of the NASDAQ-100. The code is made dynamic so that it can handle different csv files with potentially a different number of days and a different index with a different number of component stocks

Stock Selection

When selecting stocks, our objective has to be one that maximizes the similarity between the selected m and the index itself consisting of n securities:

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

Where pi,j is the correlation between the stock in the created fund and the index itself and xi,j is a binary constraint indicating if a security from the index is in the fund we created.

While maximizing the above equation, the three primary constraints we took into consideration while undergoing this process are as follows:

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

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

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

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

Constraint 1: Ensures only m stocks are included in the fund creation out of n securities in the index

Constraint 2: Imposes that each stock in the fund only is representative of one stock from the index itself

Constraint 3: Guarantees that each stock in the index is represent by each stock in the fund only if it is in the fund itself

Coding: Stock Selection

```
#---------------------------------------------------------------------------
## STOCK SELECTION
#---------------------------------------------------------------------------
# Model
mod = gp.Model()
mod.ModelSense = GRB.MAXIMIZE

# Decision Variables
x = mod.addVars(lst_stocks,lst_stocks,vtype=GRB.BINARY,name='x')
y = mod.addVars(lst_stocks,vtype=GRB.BINARY,name='y')

# First Constraints
mod.addConstr( sum(y[i] for i in lst_stocks) == m, "m")

# Second constrint
for i in lst_stocks:
    mod.addConstr( sum(x[i,j] for j in lst_stocks) == 1, "Mapping")

# Third constraint
mod.addConstrs( (x[i,j] <= y[j] for i in lst_stocks for j in lst_stocks), "Presence")

# Objective Function
mod.setObjective( sum(p[i][j]*x[i,j] for i in lst_stocks for j in lst_stocks))

mod.Params.OutputFlag = 0
# Solve
mod.optimize()
```

Portfolio Weights Calculation

Once we have selected which stocks will constitute our Fund portfolio, the next step is to calculate how much of each one we should allocate to have the best replication of the index. To calculate those weights we will match the weighted average returns of the portfolio with the index returns as closely as possible.

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

Then there are two constraints that should be satisfied. First, the sum of all weights add up to 1, and second, each weight is greater than or equal to 0 and less than or equal to 1.

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

$$w_i \geq 0.$$

Coding: Portfolio Weights Calculation

```
#------------------------------------------------------------------------
## PORTFOLIO WEIGHTS
#------------------------------------------------------------------------
# Model
w_mod = gp.Model()
w_mod.ModelSense = GRB.MINIMIZE

# Decision Variables
w = w_mod.addVars(lst_x_output,vtype=GRB.CONTINUOUS,ub=[1]*len(lst_x_output),name='w')
u = w_mod.addVars(lst_dindex,vtype=GRB.CONTINUOUS,name='u')

# Constraints
w_mod.addConstr( sum(w[i] for i in lst_x_output) == 1, "sum_w")
w_mod.addConstrs( (u[j] >=  (q[j] - sum(w[i]*r[i][j] for i in lst_x_output))  for j in lst_dindex), "abs_value_1" )
w_mod.addConstrs( (u[j] >= -(q[j] - sum(w[i]*r[i][j] for i in lst_x_output))  for j in lst_dindex), "abs_value_2" )

# Objective Function
w_mod.setObjective( sum(u[j] for j in lst_dindex) )

w_mod.Params.OutputFlag = 0
w_mod.Params.TIME_LIMIT = time_limit
# Solve
w_mod.optimize()
```

Finally the outcome of the weights times the returns of our portfolio and the index should be compared to evaluate the performance in both the in sample data (2019) and out sample data (2020).

2.

Start with m=5. Find the best 5 stocks to include in your portfolio and the weights of those 5 stocks, using the 2019 data. How well does this portfolio track the index in 2020?

Approach to the problem and code

Stock Selection

Our approach is a **2-step approach**. The **first step** in our analysis is to **find 5 stocks** that best represent the overall index of the NASDAQ-100.
Once these stocks are selected we **then assign portfolio weights** for each stock in order to best represent the index as a whole.

```
# Decision Variables
x = mod.addVars(lst_stocks,lst_stocks,vtype=GRB.BINARY,name='x')
y = mod.addVars(lst_stocks,vtype=GRB.BINARY,name='y')
```

In order to do this, we implemented a two-step approach in order to minimize the deviation of our fund's return where m = 5 with respect to the actual return of the index at time qt.

```
# Objective Function
mod.setObjective( sum(p[i][j]*x[i,j] for i in lst_stocks for j in lst_stocks))
```

The constraints we add are :

- To make sure we select a total of m stocks
- To make sure we represent only one stock for every stock in NASDAQ
- Make sure that a stock cannot be represented by a stock that is not in the portfolio

Coding

```
#--------------------------------------------------------------------------
## STOCK SELECTION
#--------------------------------------------------------------------------
# Model
mod = gp.Model()
mod.ModelSense = GRB.MAXIMIZE

# Decision Variables
x = mod.addVars(lst_stocks,lst_stocks,vtype=GRB.BINARY,name='x')
y = mod.addVars(lst_stocks,vtype=GRB.BINARY,name='y')

# First Constraints
mod.addConstr( sum(y[i] for i in lst_stocks) == m, "m")

# Second constrint
for i in lst_stocks:
    mod.addConstr( sum(x[i,j] for j in lst_stocks) == 1, "Mapping")

# Third constraint
mod.addConstrs( (x[i,j] <= y[j] for i in lst_stocks for j in lst_stocks), "Presence")

# Objective Function
mod.setObjective( sum(p[i][j]*x[i,j] for i in lst_stocks for j in lst_stocks))

mod.Params.OutputFlag = 0
# Solve
mod.optimize()

# Output Stocks
lst_x_output = []
for i in lst_stocks:
    for j in lst_stocks:
        if x[i,j].X == 1 and j not in lst_x_output:
            lst_x_output.append(j)
```

Observed Stocks

Using the 2019 data, we found the 5 best stocks to include in our fund and their corresponding weights to be as follows:

```
In [37]: # Running Stock Selection
         mod, lst_x_output = stock_selection(m=5)

In [26]: # Objective Value
         mod.objVal

Out[26]: 54.83990652229107

In [38]: # Stocks Selected
         lst_x_output

Out[38]: ['MSFT', 'VRTX', 'MXIM', 'LBTYK', 'XEL']
```

Stocks	Weights
LBTYK	0.048862
MXIM	0.210388
MSFT	0.580352
VRTX	0.07119
XEL	0.089208

Evaluation of our selected stocks

Furthermore, when using these 5 tickers to track the index in 2020, we found the mean absolute error out of sample to be **0.869670**.

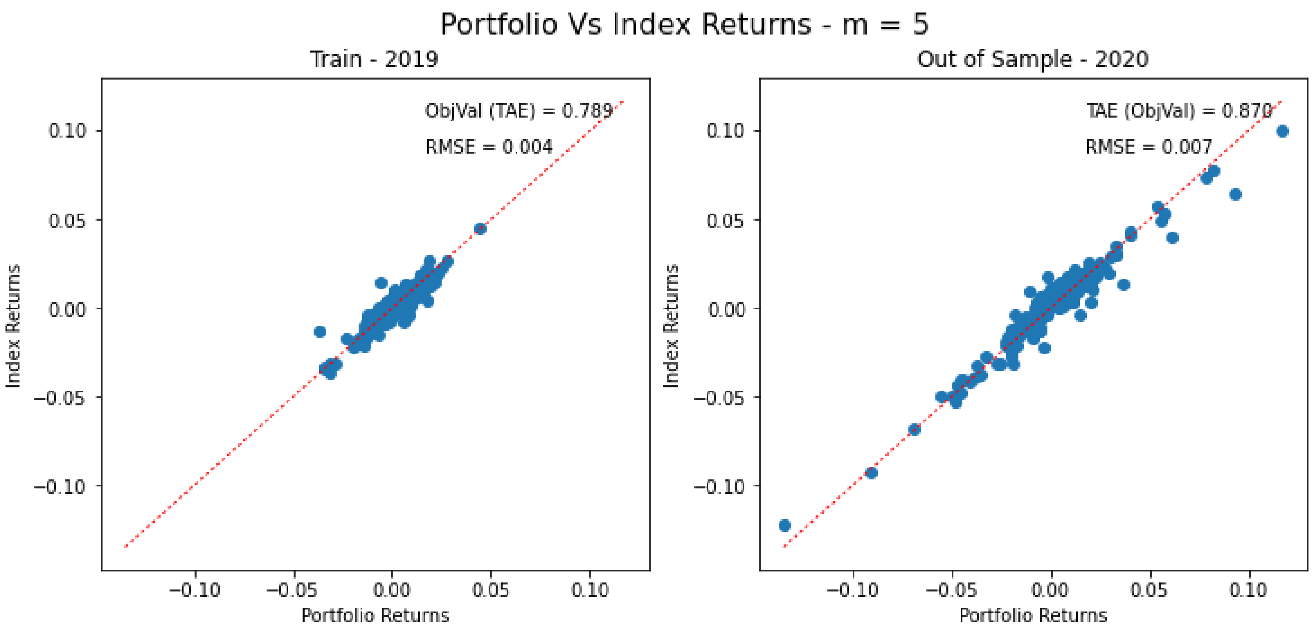
Code for Evaluation

```
    # Evaluation
    abs_error_train = w_mod.objVal
    abs_error_oos = sum(abs(q_oos-mod_ret_oos))

    return w_mod, weights_1, abs_error_train, abs_error_oos
```

```
In [63]: # Running Portfolio Weights Calculation
         w_mod, weights_1, abs_error_train, abs_error_oos = portfolio_weights(5,lst_x_output)
```

```
In [57]: # Objective Value (absolute error in Train)
         w_mod.objVal

Out[57]: 0.7891782824631471
```

```
In [53]: # Evaluation in Out of Sample(2020)
         abs_error_oos

Out[53]: 0.8696699433741903
```



(a) (b)

The above graph on the left(a) represents how our created portfolio of stocks performs on the 2019 index - which we call the Train data. The above graph on the right(a) shows how well our portfolio tracks the actual index in 2020.

We see that in (b), the portfolio tracks the index more or less accurately. This is evident from the fact that most of the points lie on the 45 degree line, so the index returns are approximately equal to the portfolio returns with some deviations. Because of these deviations, the TAE slightly increases from the Total absolute error measured on the 2019 returns.

3.

Redo step (2) with m = 10, 20, …, 90, 100 (obviously when m=100 you don't need to solve for which stocks to include, because they're all included). Analyze the performance of the portfolio for each value of m. How does the performance change? Is there some value of m, where there are diminishing returns of including more stocks in the portfolio? You can also look at the in-sample performance. That is, evaluate the performance in 2019 using 2019 portfolio construction and 2019 data. How is the performance in 2019 different from than performance in 2020?

Why is it different?

Approach

Redoing the stock selection with different values of m moving in steps of 10 from 10 to 100. These steps can be changed by the user by changing the values in the list.

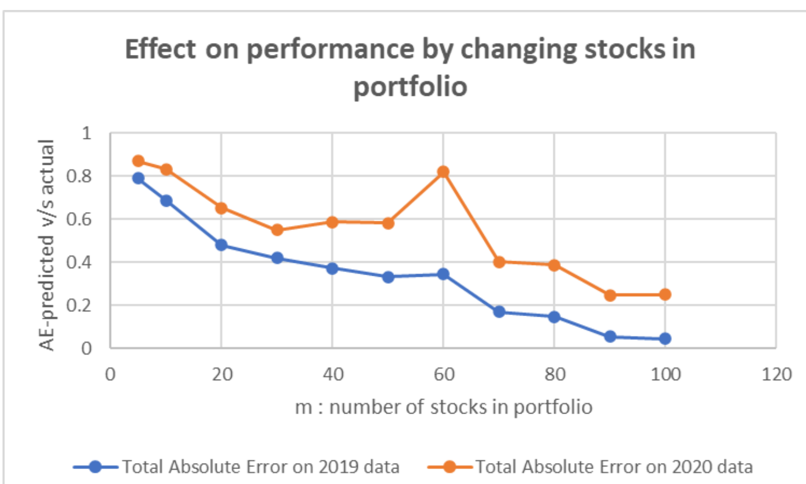
```
lst_ms = [5] + [*range(10,n_stocks,10)] + [n_stocks] # The list depends on number of stocks

for m in lst_ms:

    #-------------------------------------------------------------------------
    ## STOCK SELECTION
    #-------------------------------------------------------------------------
    # Model
    mod = gp.Model()
    mod.ModelSense = GRB.MAXIMIZE

    # Decision Variables
    x = mod.addVars(lst_stocks,lst_stocks,vtype=GRB.BINARY,name='x')
    y = mod.addVars(lst_stocks,vtype=GRB.BINARY,name='y')
```

Observations and Evaluation

We check the performance of the portfolio for increasing values of m in the table below:

Following is a visualization of the performance in 2019 and 2020 indices with change in the number of stocks in our portfolio.

m stocks in fund	Total Absolute Error on 2019 data	Total Absolute Error on 2020 data
5	0.789178	0.86967
10	0.686533	0.831317
20	0.478836	0.652338
30	0.418015	0.549085
40	0.370517	0.587312
50	0.33254	0.581148
60	0.34489	0.819424
70	0.169824	0.402497
80	0.147683	0.386431
90	0.053779	0.247582
100	0.044911	0.249936



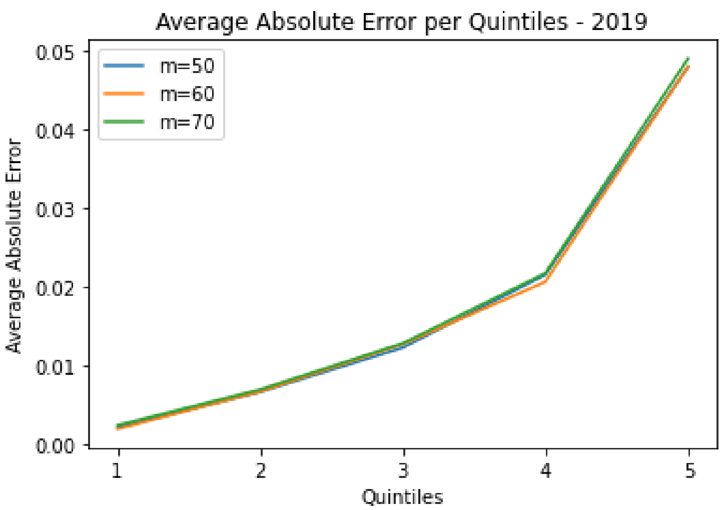
The above graph shows the performance of 2019 vs 2020 data for changing values of m against Total Absolute Error. We see that the portfolio performance is better in 2019 than in 2020. This is because the portfolio of stocks was created out of the 2019 data which forms our training data. So expectedly, the performance of the portfolio with the 2019 data is much better.

Is there some value of m, where there are diminishing returns of including more stocks in the portfolio?

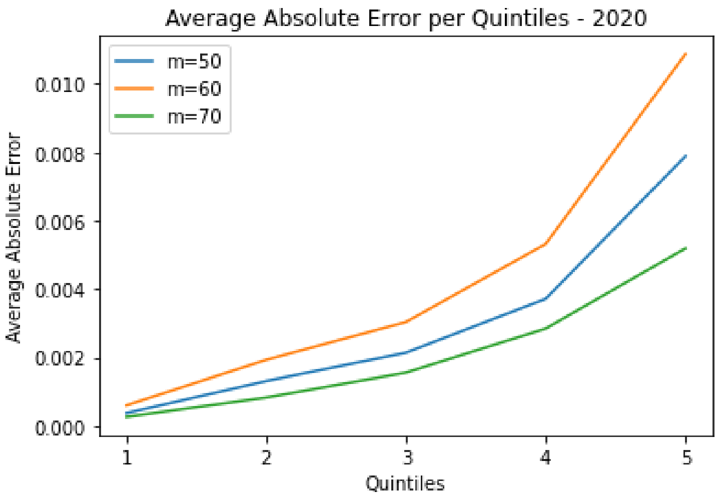
Referencing to the abovementioned graph, we see that the out-of-sample Total Absolute Error (TAE) for 2020 data decreases monotonically from m=5 to m=50 but shoots up drastically when the value of m changes from 50 to 60, especially at **m=60**. So for a range between m=50 and m=60, there are diminishing returns of including more stocks in the portfolio. The total absolute error for 2020 is given by $|q_{2020} - \sum w_i r_{it}|$ (similar for 2019) and as this value increases, it means the accuracy of tracking the index by our portfolio decreases.

Why is it different?

Suppose we take quintiles for daily errors for both 2019 and 2020 data for m=50,60 and 70. We then plot the 2019 and 2020 trends of error quintiles on charts with the x axis representing the quintiles and y axis representing the average absolute error in each quintile we get the following two graphs:



Errors by quintiles for 2019 for m=50,60,70. Almost consistent as we move up quintiles



Errors by quintiles for 2020 for m=50,60,70. Less consistent as we move up quintiles

We see a stark difference between the consistency of the Average Absolute errors as we move up the quintiles for the 2 years. This indicates that for 2020, there were some days for which our portfolio exhibited more errors for m=60 than those for m=50 and m=70.

4.

Another way you could solve this problem is to completely ignore the stock selection IP and reformulate the weight selection problem to be a MIP that constrains the number of non-zero weights to be an integer. Which method works better on the 2020 data, the original method or this new method?

Approach to the problem

In this problem, we take a different approach compared to our first approach. The second approach is more accurate than the first approach. In the first approach, first we selected the stocks and then we assigned the weights to the selected stocks. However, in the second method, we reformulate the weight selection problem to be a MIP that places a constraint on the number of non zero weights such that they have to be an integer.

Our approach is a **1-step approach**. In this method, we try to find out **which stocks to select** (Y) in a binary format and get the **weights**(w) in a continuous format **at the same time**. While optimizing, we try to minimizing a variable u such that

$$u_t = |q_t - \sum_i w_i r_{it}|$$

```
# Decision Variables
w = w2_mod.addVars(lst_stocks,vtype=GRB.CONTINUOUS,ub=[1]*len(lst_stocks),name='w')
y = w2_mod.addVars(lst_stocks,vtype=GRB.BINARY,name='y')
u = w2_mod.addVars(lst_dindex,vtype=GRB.CONTINUOUS,name='u')
```

Our objective function would be to minimize u over time period t such

$$obj = min \sum_t u_t$$

```
# Objective Function
w2_mod.setObjective( sum(u[j] for j in lst_dindex) )
```

The way we optimize a non-linear objective function as above is that we declare u as a variable to optimize that would be defined by 2 constraints: $u \geq q - \sum w_i r_{i,j}$ and $u \geq -(q - \sum w_i r_{i,j})$

Why do we do this? Because as u should represent an absolute function, we need u to be a positive value, which would only be possible by introducing the above 2 constraints.

```
# Constraint
w2_mod.addConstr( sum(w[i] for i in lst_stocks) == 1, "sum_w")
w2_mod.addConstrs( (u[j] >=  (q[j] - sum(w[i]*r[i][j] for i in lst_stocks))  for j in lst_dindex), "abs_value_1" )
w2_mod.addConstrs( (u[j] >= -(q[j] - sum(w[i]*r[i][j] for i in lst_stocks))  for j in lst_dindex), "abs_value_2" )
w2_mod.addConstrs( (w[i] <= y[i]*bigM for i in lst_stocks), "force_w0_y0")
w2_mod.addConstr( sum(y[i] for i in lst_stocks) == m, "m")
```

Other constraints we add are:

- We add big M constraints to ensure that we do not have portfolio weights with positive values for stocks which are not being selected.
- We add a constraint that makes sure all weights add up to 1
- We add a constraint that makes sure we are selecting only m stocks by limiting y

With these constraints, we run the Gurobi Optimization.

Observations and Evaluation

Following is the time required to run the optimization for m number of stocks

```
m: 5 - Time: 60.01 minutes
m: 10 - Time: 60.01 minutes
m: 20 - Time: 60.02 minutes
m: 30 - Time: 60.01 minutes
m: 40 - Time: 60.01 minutes
m: 50 - Time: 60.01 minutes
m: 60 - Time: 60.01 minutes
m: 70 - Time: 60.01 minutes
m: 80 - Time: 1.71 minutes
m: 90 - Time: 0.03 minutes
m: 100 - Time: 0.01 minutes
```

Upon doing so we obtain the following values for absolute error, RMSE of 2019 and 2020

Out[8]:

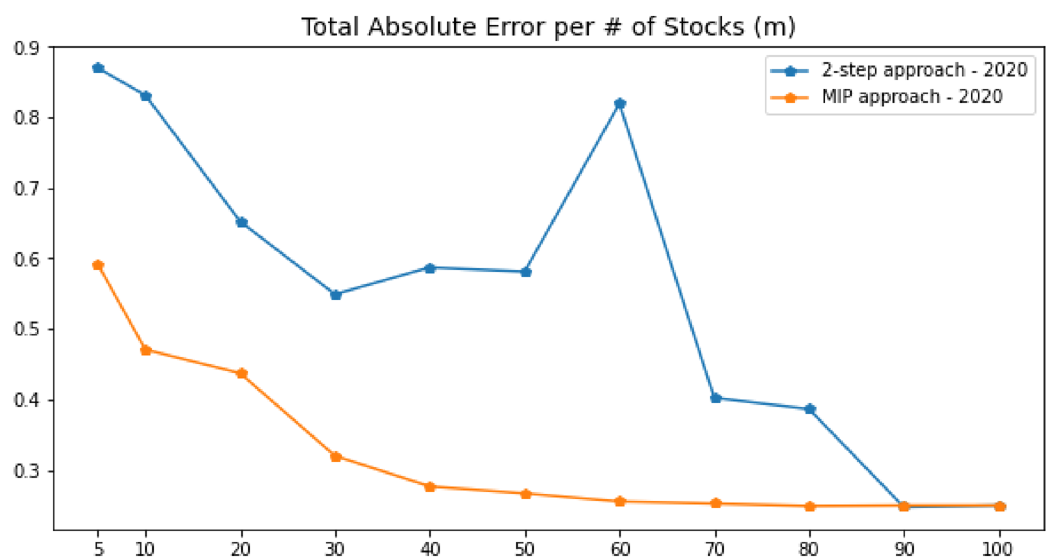
	m	#stocks	objVal_train	rmse_train	objVal_oos	rmse_oos
1	5	5	0.499259	0.002859	0.591398	0.004353
2	10	10	0.303673	0.001612	0.470866	0.003525
3	20	20	0.164830	0.000928	0.437215	0.003156
4	30	30	0.109360	0.000618	0.320199	0.002279
5	40	40	0.078134	0.000460	0.276861	0.002081
6	50	50	0.061592	0.000410	0.266989	0.001988
7	60	60	0.052299	0.000337	0.255418	0.001934
8	70	70	0.047529	0.000309	0.252523	0.001948
9	80	80	0.045227	0.000296	0.249124	0.001894
10	90	88	0.044911	0.000299	0.249943	0.001899
11	100	88	0.044911	0.000299	0.249936	0.001899

According to the question, if we run this optimization problem, there is a possibility that it may run for over 24hrs. So we add a time limit beyond which we force Gurobi to stop looking for a solution.

Setting: Time Limit --This number should be set!!

```
[ ] # TIME LIMIT
    time_limit = 3600
```

When we select 5/10/20/30/40/50/60/70 stocks, Gurobi runs until the time limit. However, expectedly, when we select more stocks, the optimization solution is generated within a minute.



Which Method works better

The above plot shows how the absolute out-of-sample error varies with the number of stocks selected in the two approaches. We can clearly see that the second approach performs much better on the 2020 data and has a much lower value of absolute error at most of the data points.

Even when we select just 5 stocks, we have an absolute error of just 0.6 with the second MIP approach, while with the first approach the error value is close to 0.9. So, the accuracy of the second approach is much higher. This is because the first approach has some inherent bias. Because first we select the stocks on the basis of highest correlation scores and then assign portfolio weights. So the selected stocks in the first approach may not be the ones which also minimizes the error between the actual return of the index and the weighted average return of the portfolio. Also, with the second approach, we do not observe any sharply diminishing returns when more stocks are included in the portfolio.

5.

Pretend you are an analyst at a mutual fund. Your boss has asked your team to come up with a recommendation for how many component stocks to include in the fund and how to pick their weights going forward. Write this project as if this is what you're going to deliver to your boss. Your boss is pretty technical and understands optimization, so don't be afraid to include quantitative material. Your boss is also busy, so be sure to include some visualizations to get the important points across. For the purpose of your recommendations, you can assume that your boss is interested in the data posted with the project.

Data Description

Stocks2019:

	X	NDX	ATVI	ADBE	AMD	ALXN	ALGN	GOOGL	GOOG	AMZN	...	TCOM	ULTA	VRS
0	2019-01-02	6360.870117	46.350380	224.570007	18.830000	98.050003	202.119995	1054.680054	1045.849976	1539.130005	...	27.590000	247.970001	147.7599
1	2019-01-03	6147.129883	44.704514	215.699997	17.049999	100.209999	184.779999	1025.469971	1016.059998	1500.280029	...	26.959999	243.360001	142.5899
2	2019-01-04	6422.669922	46.488358	226.190002	19.000000	106.000000	186.710007	1078.069946	1070.709961	1575.390015	...	28.549999	255.029999	148.9700
3	2019-01-07	6488.250000	47.799141	229.259995	20.570000	107.940002	189.919998	1075.920044	1068.390015	1629.510010	...	29.180000	271.000000	151.3999
4	2019-01-08	6551.850098	49.247898	232.679993	20.750000	108.610001	192.949997	1085.369995	1076.280029	1656.579956	...	29.480000	276.000000	156.9199
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
246	2019-12-23	8696.009766	58.505219	328.950012	45.459999	110.459999	278.140015	1350.630005	1348.839966	1793.000000	...	34.660000	253.020004	192.4299
247	2019-12-24	8699.509766	58.425743	329.640015	46.540001	110.279999	277.890015	1344.430054	1343.560059	1789.209961	...	34.470001	252.490005	192.7500
248	2019-12-26	8778.309570	58.505219	331.200012	46.630001	108.930000	278.260010	1362.469971	1360.400024	1868.770020	...	34.570000	251.330002	193.7100
249	2019-12-27	8770.980469	58.803261	330.790009	46.180000	108.550003	277.640015	1354.640015	1351.890015	1869.800049	...	34.610001	253.169998	194.0500
250	2019-12-30	8709.730469	58.495289	328.339996	45.520000	107.339996	275.630005	1339.709961	1336.140015	1846.890015	...	34.200001	251.350006	192.3300

Stocks2020:

	X	NDX	ATVI	ADBE	AMD	ALXN	ALGN	GOOGL	GOOG	AMZN	...	TCOM	ULTA	VR
0	2020-01-02	8872.219727	58.266792	334.429993	49.099998	107.839996	283.679993	1368.680054	1367.369995	1898.010010	...	36.970001	254.550003	196.729
1	2020-01-03	8793.900391	58.286655	331.809998	48.599998	106.410004	280.440002	1361.520020	1360.660034	1874.969971	...	36.180000	250.169998	200.880
2	2020-01-06	8848.519531	59.349670	333.709991	48.389999	106.580002	285.880005	1397.810059	1394.209961	1902.880005	...	35.689999	250.949997	202.740
3	2020-01-07	8846.450195	59.945747	333.390015	48.250000	106.849998	283.059998	1395.109985	1393.339966	1906.859985	...	37.330002	253.089996	203.210
4	2020-01-08	8912.370117	59.488754	337.869995	47.830002	108.580002	286.000000	1405.040039	1404.319946	1891.969971	...	36.869999	258.000000	204.149
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
184	2020-09-24	10896.469727	79.690002	467.670013	75.820000	112.019997	311.880005	1422.859985	1428.290039	3019.790039	...	27.750000	215.589996	202.009
185	2020-09-25	11151.129883	80.980003	479.779999	78.059998	114.430000	317.929993	1439.060059	1444.959961	3095.129883	...	30.150000	216.830002	205.580
186	2020-09-28	11364.450195	81.949997	488.510010	79.480003	113.300003	322.859985	1458.660034	1464.520020	3174.050049	...	32.000000	225.740005	204.720
187	2020-09-29	11322.950195	80.779999	489.329987	81.769997	112.459999	320.170013	1466.020020	1469.329956	3144.879883	...	31.270000	222.880005	204.910
188	2020-09-30	11418.059570	80.949997	490.429993	81.989998	114.430000	327.359985	1465.599976	1469.599976	3148.729980	...	31.139999	223.979996	204.850

Both files have daily level data for 100 stock prices for the mentioned year. The columns represent stock tickers with NDX as the NASDAQ index. For 2019, we have the entire year's data. For 2020, we have data till 30th of September

Recommendation

We can minimize the absolute error by using the **1-step approach** with **80 stocks** and it also does not take too much time to figure out which stocks to take. If we use the recommended second approach, we can make a **tradeoff between** choosing **40 or 50 stocks** by **tracking the index on a lesser amount of expenses** and it can also be easier to manage fewer stocks.