Project 2

Optimization of Stocks selection to replicate Nasdaq-100 Index

Team Members :

David Gong (dg38767)

Rathil Madihalli (rm63782)

Sameer Ahmed (sa49847)

Sian Cheng (sc65827)

Problem Setting

Equity management is the process of creating and managing owners in a company. They can be broadly categorized into ‘active’ and ‘passive’ strategies. A prevalent passive investment strategy involves employing an index fund designed to replicate the structure and returns of a particular index, such as the S&P 500. This can be accomplished by acquiring all the stocks within the index proportionally weighted. However, this approach can become cost-intensive, due to the expenses during the periodic rebalancing of the portfolio through trading.

So, we will create an index fund that best tracks the NASDAQ-100 with fewer stocks than the index. We are doing this to get a passive return using the success of a large and successful index while mitigating the downsides. This can be accomplished by creating an integer program that selects *m* out of the *n* available stocks for inclusion in the fund. Additionally, we will utilize a linear program to determine the most favorable weights for these selected stocks, with the goal of minimizing the absolute disparities between the portfolio returns and the index returns.

Methodology

Initially, we computed the daily stock returns for each stock in our 2019 and 2020 stock data frames. This operation was executed using the following code:

```
return19 = stock19.pct_change()  # returns for stocks in 2019
return20 = stock20.pct_change()  # returns for stocks in 2020
```

```
return19.head(5)
```

X	NDX	ATVI	ADBE	AMD	ALXN	ALGN
2019-01-02	NaN	NaN	NaN	NaN	NaN	NaN
2019-01-03	-0.033602	-0.035509	-0.039498	-0.094530	0.022030	-0.085791
2019-01-04	0.044824	0.039903	0.048632	0.114370	0.057779	0.010445
2019-01-07	0.010211	0.028196	0.013573	0.082632	0.018302	0.017192
2019-01-08	0.009802	0.030309	0.014918	0.008751	0.006207	0.015954

In [8]:

```
return20.head(5)
```

Out[8]:

Unnamed: 0	NDX	ATVI	ADBE	AMD	ALXN	
2020-01-02	NaN	NaN	NaN	NaN	NaN	
2020-01-03	-0.008827	0.000341	-0.007834	-0.010183	-0.013260	-0.0
2020-01-06	0.006211	0.018238	0.005726	-0.004321	0.001598	0.0
2020-01-07	-0.000234	0.010043	-0.000959	-0.002893	0.002533	-0.0
2020-01-08	0.007452	-0.007623	0.013438	-0.008705	0.016191	0.0

Next, we generated a similarity matrix that computes the correlation between the returns of all possible pairs of stocks, denoted as ρij. This matrix enables us to know which stocks exhibit the highest similarity and could potentially serve as representatives. (See the screenshot below for reference.)

```
# Correlation matrix for returns (this will be the coefficients in the objective function)
rho = return19.drop(columns='NDX').corr()
rho.head(5)
```

	ATVI	ADBE	AMD	ALXN	ALGN	GOOGL	GOOG	AMZN	AMGN	ADI	...	TCOM	ULTA	VRSN	VRSK	VRT
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	0.25967
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	0.40217
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	0.27298
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	0.52242
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	0.33497

For the selection process, we formulated an integer program to choose m stocks that

best represent the NASDAQ-100. Our decision variables, objective function, and constraints are shown below:

Decision Variables:

y_i : whether stock j is selected in the portfolio or not (1 if j selected in the fund, 0 otherwise)

x_{ij}: whether stock j found to be most similar to stock i (x_{ij}= 1 if stock j in index is the most similar to stock i, 0 otherwise)

Objective Function:

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

```
mod.setObjective(gp.quicksum(gp.quicksum(modX[i][j] * rho.iloc[i, j] for i
    in range(n)) for j in range(n)), sense=gp.GRB.MAXIMIZE)
```

Constraints:

1) Exactly m stocks in the fund

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

```
con1 = mod.addConstr(gp.quicksum(modY[j] for j in range(n)) == m)
```

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

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

```
con2 = mod.addConstrs(gp.quicksum(modX[i][j] for j in range(n)) == 1
    for i in range(n))
```

3) Stock *i* is represented by *j* only and only if *j* is in the fund

$$x_{ij} \leq y_j (i, j = 1, 2, \ldots, n)$$
$$x_{ij}, y_j \in \{0, 1\}$$

```
con3 = mod.addConstrs(modX[i][j]<=modY[j] for i in range(n) for j in
    range(n))
```

Next, for the portfolio weights selection, we formulated a linear program to identify the optimal weights of each of the stocks chosen in the portfolio.

optimal weights to minimize differences in returns between the portfolio and the index.

We denote:

r_{it}: return of stock i at time t

q_t: return of index at time t

w_i: weight of stock i in portfolio

Objective Function:

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

Subject to:

$$\sum_{i=1}^{m} w_i = 1$$

We can rewrite this problem as:

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

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

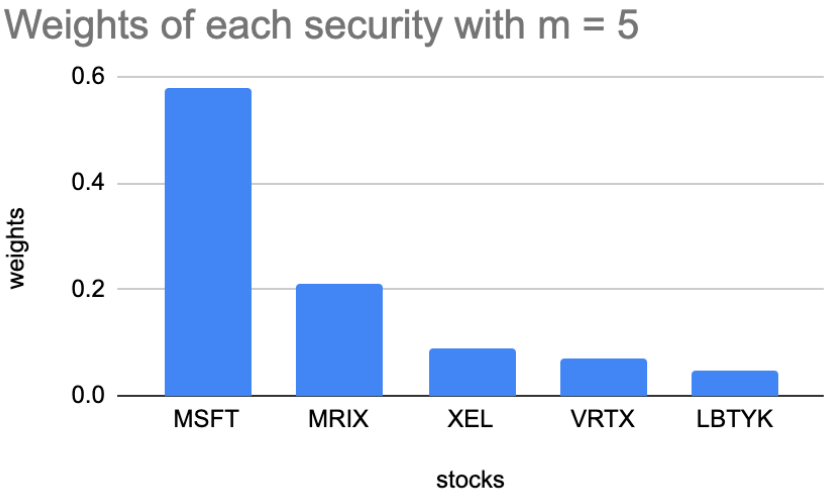
$$y_t \geq \sum_{i=1}^{m} w_i r_{it} - q_t \longrightarrow y_t - \sum_{i=1}^{m} w_i r_{it} \geq -q_t$$

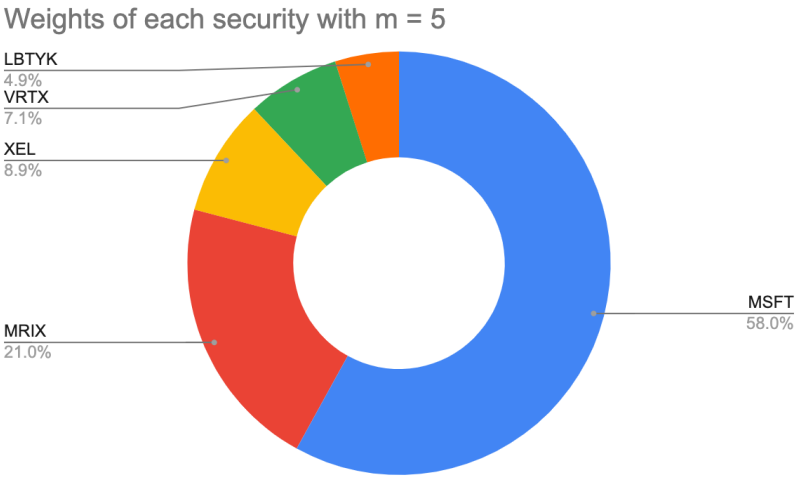
```
mod.setObjective(sum(mody), sense=gp.GRB.MINIMIZE)
```

```
y_t = [train_q[t]-gp.quicksum(selected_return.iloc[t,i] * modw[i] for i in
                              range(m)) for t in range(T)]
```

Having established all the requisite functions for stock selection, weight assignment, and performance calculation, we chose m = 5 and determined the top 5 stocks for inclusion in our portfolio, along with their corresponding weights. The selected 5 best stocks for our index fund are detailed below.

	LBTYK	MXIM	MSFT	VRTX	XEL
Weight	0.048862	0.210388	0.580352	0.07119	0.089208





When we choose 5 stocks to represent the portfolio with this method, the stocks

chosen are LBTYK, MXIM, MSFT, VRTX, and XEL. The highest weight is MSFT while the lowest weight is LBTYK. Using this method, all the other stocks are thrown out and not used at all.

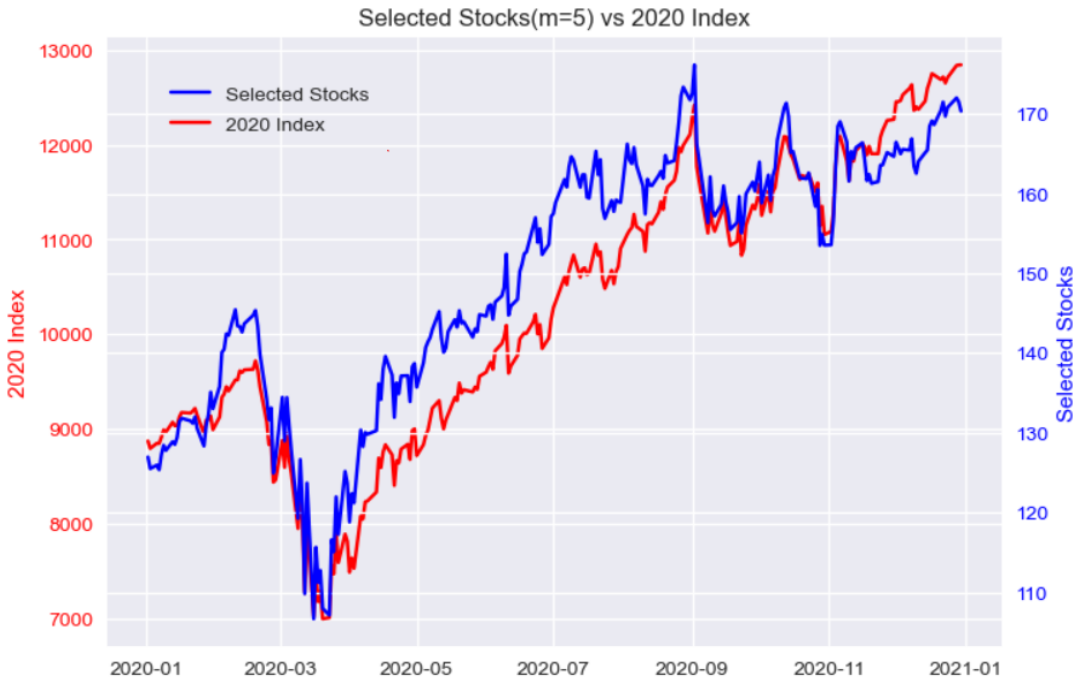
Validation on 2020 Returns:

In order to validate on the 2020 returns we used the following formula where w_i is the weights we calculated above.

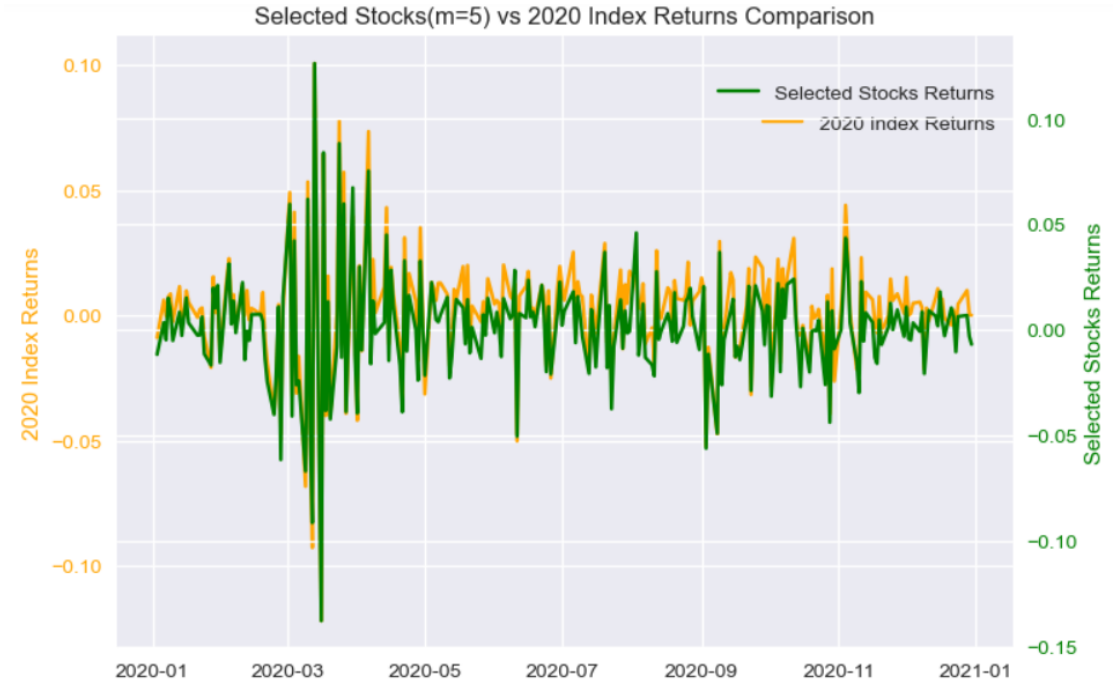
$$\sum_{t=1}^{T} \left| q_t - \sum_{i=1}^{5} w_i r_{it} \right|$$

```
# Calculate the error

T = len(evaluate_df) # Number of trading days

error = 0

for t in range(T):

qt = evaluate_df.iloc[t,:]['NDX'] # Daily index return

ret = evaluate_df.iloc[t,:][selected_stocks].to_list() # Daily portfolio
returns

error += abs(qt - np.dot(weight, ret)) # Absolute difference between the
returns
```

From this we determined that our overall **error is 1.12.** To better display this error we plotted the daily stock value vs the 2020 index change.



The graph presents the performance of our selected stocks (m=5) in relation to the 2020 Index over a year. While there are periods of divergence, overall, our chosen stocks tend to follow the trajectory of the 2020 Index. However, a better comparison would be returns. This is shown below.



Similarly to the previous graph, our five stock selections largely reflect the 2020

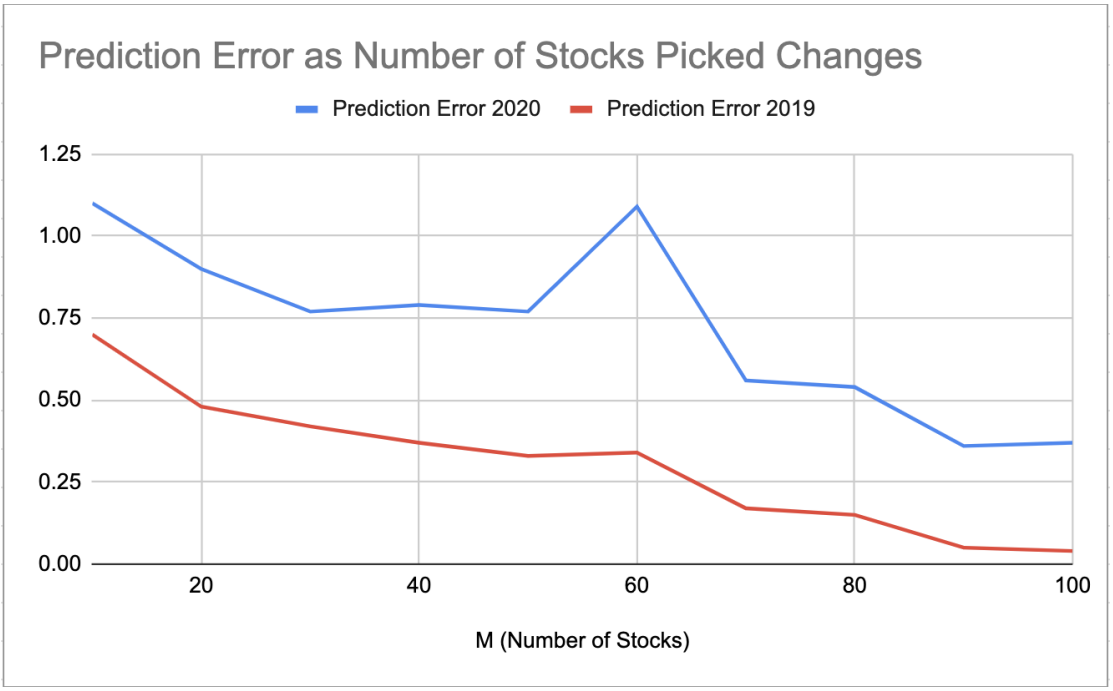
Index's return trend, indicating effective alignment with broader market movements. With just 5 stocks, we remarkably capture a significant amount of the returns variance, demonstrating the model's strength.

METHOD 1

Different Stock Selection Size (m) and Impact on Performance

We conducted an analysis to assess the impact of varying stock selection sizes on the optimization problem's performance. Utilizing a consistent performance metric - the absolute value difference between the selected stocks' performance and that of the index - allowed for a standardized evaluation.

	M (Number of Stocks)	Prediction Error 2020	Prediction Error 2019
0	10.0	1.100511	0.701218
1	20.0	0.898691	0.478836
2	30.0	0.769042	0.418015
3	40.0	0.790233	0.370517
4	50.0	0.771502	0.332540
5	60.0	1.093723	0.344890
6	70.0	0.555813	0.169824
7	80.0	0.535013	0.147683
8	90.0	0.364608	0.053779
9	100.0	0.365480	0.044911



A key observation is the pronounced stability in 2019's prediction error as the number of stocks increases, in stark contrast with 2020's fluctuating trend. Given that the model's weight calculations are based on the absolute difference between a selected parameter q_t and the sum of individual stock returns up to m stocks, it suggests that the model was well-fitted to the 2019 data. However, its generalization to the out-of-sample data of 2020 seems suboptimal. The sharp peak in prediction error for 2020 around the 60-stock mark could indicate overfitting during training, or it might suggest that 2020 introduced market factors or events that weren't captured or anticipated by the 2019-trained model. When forecasting out-of-sample for stock selection for the subsequent year, such as 2021, it would be crucial to consider the model's limitations highlighted by these discrepancies and potentially retrain or adjust the model to account for unforeseen market changes or shocks. The point that it starts having diminishing returns is around M = 70. M = 30 also has a decent decrease in error before it starts plateauing and going back up at 60, but at 70 it starts having a meaningful decrease in error again.

Nonetheless, a pattern is evident that the error diminishes as the variety of stocks in the selection expands. This is logically consistent, given that increasing the stock assortment enhances its performance, aligning it more closely with the normal index. It’s noteworthy, however, that the error margin remains relatively minimal. To pinpoint an exact M (number of stocks) that optimizes the portfolio, the introduction of a cost function is imperative. This function will impose a penalty for each additional stock incorporated, thereby ensuring a balance between diversity and precision.

METHOD 2

Instead of solving two linear optimization problems, we can solve one by using Big-M constraints and modifying our objective function. Our objective function will now be optimizing over ALL weights:

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

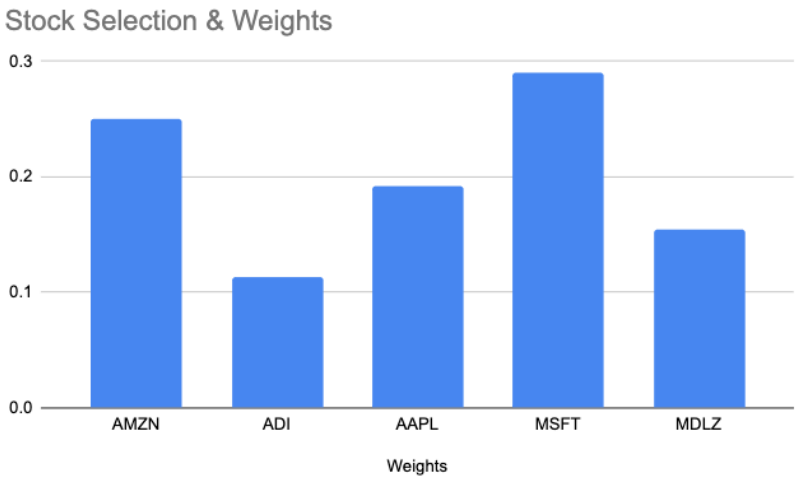
We convert the Absolute Value function in the following way:

```
mip_model.setObjective(gp.quicksum(y_abs_diff[t] for t in range(T)), sense=gp.GRB.MINIMIZE)

for t in range(T):

portfolio_return_at_t = gp.quicksum(w[i] * returns_2019.iloc[t, i+1] for i in range(n))

mip_model.addConstr(y_abs_diff[t] >= returns_2019.iloc[t, 0] - portfolio_return_at_t)

mip_model.addConstr(y_abs_diff[t] >= -returns_2019.iloc[t, 0] + portfolio_return_at_t)
```

The drawback of this approach is its NP-Hard nature. We allowed it to operate for 1 hour per run and tested various stock selection sizes.

```
GUROBI_TIME_LIMIT = 3600
```

For m=5, the following stocks and their weights are displayed below.

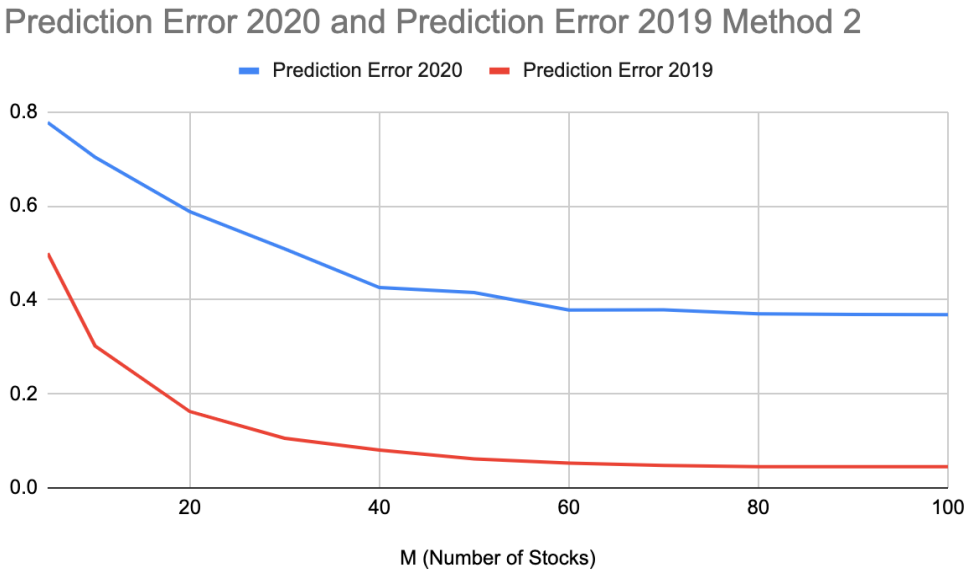


As you can see, the stocks selected are AMZN, ADI, AAPL, MSFT, and MDLZ. MSFT has the highest weight while ADI has the lowest weight. As indicated by the method, all other stocks are set to 0, so these are the only ones with a weight. These

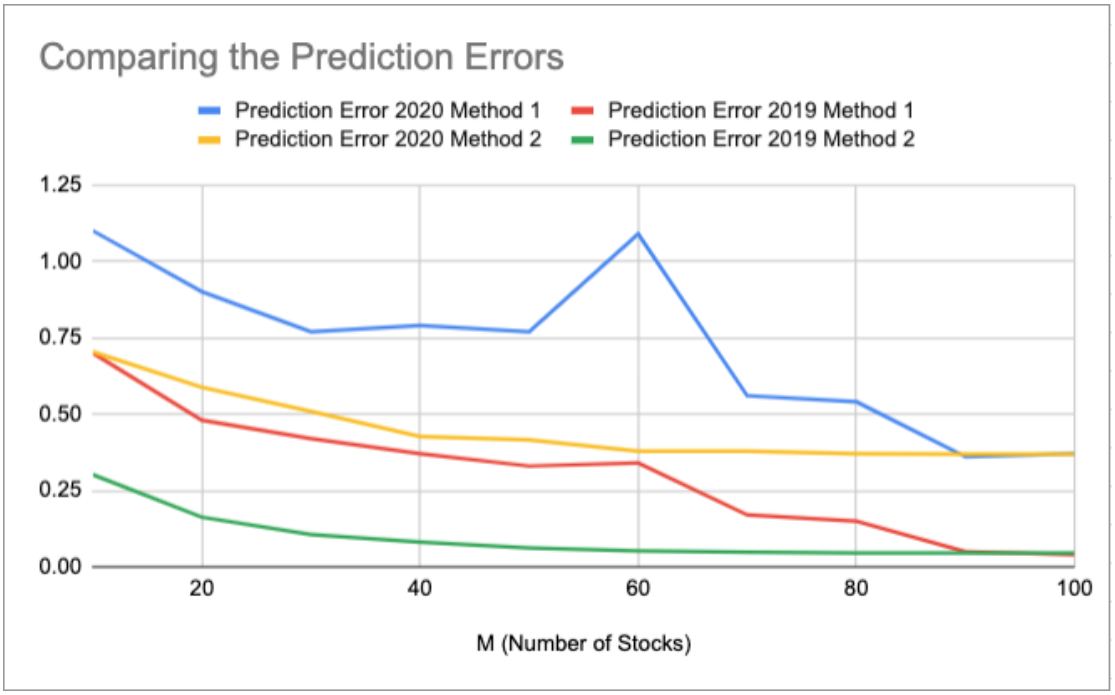
stocks selected show that the differing methods result in different stocks that the program believes are the best stocks.

Note that the stock selection itself is different from the first method. This makes it incumbent that we also check the error for the new method and compare it with the prior.

	M (Number of Stocks)	2019	2020
0	5	0.499259	0.777362
1	10	0.301862	0.703715
2	20	0.162433	0.587937
3	30	0.105641	0.508643
4	40	0.080632	0.426427
5	50	0.061594	0.415462
6	60	0.052605	0.378358
7	70	0.047577	0.378687
8	80	0.045227	0.370615
9	90	0.044911	0.369088
10	100	0.044911	0.368671



Method Two actually outperforms Method One. In order to better demonstrate it, we plotted both errors.



The error rate of Method 2 is markedly lower compared to Method 1. This substantial difference in prediction accuracy proves the superiority of Method 2. Therefore, if time permits, opting for Method 2 would be the more advantageous choice due to its enhanced precision.

Conclusion:

In this report, our goal is to use a passive strategy to choose a small selection of stocks to mirror one of the most successful indexes, the NASDAQ-100. We achieved this goal and created our portfolio through *two different methods*.

Our first method was choosing an m number of stocks that are most representative of the entire stock set using a similarity matrix called p. We then use an objective function to maximize the sum of the similarities between the chosen stocks and the total stocks. After choosing m = 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 stocks, we find the weights of each stock chosen to mirror the index's return as closely as possible. We do this through an objective function to minimize the absolute value difference of our chosen portfolio's return and the return of the index.

The second method is to go through all the stocks but limiting the number of non-zero stock weights to m. We do the same thing with m = 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100. Then we create an objective function where we minimize the absolute value difference of our chosen portfolio's return and the return of the index again. This method is NP Hard and it takes much longer to run. We had to limit each run to 1 hour and iterating through all these took around 10 hours to do.

We found that for both methods the error goes down as we include more stocks into our portfolio, which is expected. The second method minimizes the error a lot more than the first method, but again, with the NP Hard complexity it makes it much more time consuming to run.

In order to decide the optimal number of stocks to include in our portfolio, we propose utilizing a cost function. The function will penalize each additional stock, ensuring we reach an optimal point. This strategy will be based on our specific business needs and will be developed with insights from our senior executives. An example of the updated optimization function would be for method 1 is below, where *f(x)* is the cost function.

$$\min_{w} \sum_{t=1}^{T} |q_t - \sum_{i=1}^{m} w_i r_{it}| + f(x)$$

Through the first method (without a cost function), we get an ideal number of stocks, or m = 30. This is the point that has an error that continuously decreases until this point. At m = 60, there is a spike in the error upwards and then goes downwards after. At 70, it goes down again but we did not think that it was worth buying that many stocks to reduce the error by a small amount. We think this point should generate the closest return without spending too much on buying more stocks.

Through the second method, we get an ideal number of stocks of m = 40. The error for this graph decreases at a pretty rapid rate until m= 40 where it still decreases, but it plateaus and decreases very slowly. Another point could be m = 60, as the error does increase a little at m= 70, but we think that it is not worth it as the decrease in error from 40 to 60 is almost negligible.

Additionally, we suggest considering Method 2 when time allows. Our findings indicate that it offers a lower error rate, making it a potentially more accurate approach for stock selection. This combination of strategies is aimed at optimizing our portfolio while balancing risk and return efficiently.

While Method 2 provides a much better error rate and is probably more accurate, the computation is much heavier and takes more time to run. If time is a constraint, then we propose Method 1. However, you can see that there are possible spikes in error on the first method, so that is something you will definitely have to watch out for when using the first method.

The main thing we learned is that it is possible to match a smaller portfolio to a large index, such as the NASDAQ-100. Going forward, if we have a lot of time available,

using Method 2 to find the ideal stocks and weights is better. If time is short, using Method 1 is better. It seems that using an m of around 40 is ideal for Method 2, while Method 1 has an ideal m of 30. This could be helpful to shorten the time to run as instead of running the program with all 100 stocks, it would be easier to run just up to m = 40.