OPTIMIZATION PROJECT - 2 INTEGER PROGRAMMING

Portfolio Optimization with Gurobi: An Algorithmic Approach to Building an Index Fund

2023-10-21

GROUP - V

Manideep Telukuntla, Krittika Deshwal, Milind Bhatia, Chih-En Ko

Table of Contents

1. Introduction

An index fund is a type of mutual fund or exchange-traded fund (ETF) with a portfolio constructed to match or track the components of a financial market index, such as the NASDAQ-100, Standard & Poor's 500 Index (S&P 500) or other similar indexes. Constructing an index fund that tracks a specific broad market index can be done simply by purchasing all the stocks in the index, with the same weights as in the index. However, this approach is impractical (many small positions) and expensive. An index fund with q stocks, where q is substantially lower than the size of the target population (n) is much more ideal. In this project we try to create a smaller portfolio to track the NASDAQ-100 index.

2. Objective

The goal of this project is to create a portfolio of 'm' stocks, an index fund, that can track the NASDAQ-100 index by identifying those 'm' stocks and optimizing the difference in portfolio performance. To create an index fund, we need to decide the number of such stocks, identify those stocks and come up with weighted allocations that can resonate with the NASDAQ-100 index.

There are two questions to answer:

- how many stocks?
- and which stocks?

So initially, we plan to devise an optimization problem to find the 5 best stocks and their weights to create an index that has the lowest error (exact metric defined in the methodology) with the NASDAQ-100. Then we created a portfolio by varying the number of stocks in the index to find a point of diminishing return. The optimization will be done on data for the year 2019 and the performance will be evaluated on 2020's stock returns data.

3. Methodology

We evaluate two approaches to execute a successful portfolio performance calculation using optimization models:

1. **With Stock Selection:** Identify the stocks that best represent the value of all stocks in the index AND calculate the weightage for all 'm' stocks.

2. **Without Stock Selection:** Evaluate the approach where we only calculate weights without Stock Selection and run our model as a Mixed Integer Program

3.1 With Stock Selection: Selecting Stocks and Calculating Optimal Weights

For this approach, we first determined the correlation between each stock in the index and the fund to increase similarity between the index stock and the fund that represents it. For this the objective function and constraints are:

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

```
# Set objective to maximize total correlation
model.setObjective(gp.quicksum(correlation_matrix.iloc[i, j] * x[i, j]
    for i in range(n) for j in range(n)), sense=GRB.MAXIMIZE)
```

Code Block 1: Objective function for maximizing correlation defined using Gurobi in python

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

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

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

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

The objective here is to maximize the correlation that exists between our m-stock portfolio and the whole index. To achieve this, we consider 100 y_j variables and 100*100 = 10,000 x_{ij} variables.

- The y_j binary variables represent whether or not a stock is present in our m-stock fund.
- The x_{ij} binary variables represent whether or not a stock j is the most similar to stock i.

```
# Decision variables
x = model.addVars(n, n, vtype='B')
y = model.addVars(n, vtype='B')
```

Code Block 2: Decision variables defined using Gurobi in python

The constraints to be met:

- The first constraint $\sum_{j=1}^{n} y_j = m$ indicates the number of stocks that must be present in the m-stock fund.
- The second constraint $\sum_{j=1}^{n} x_{ij} = 1 \; for \; i = 1,2,\ldots,n$ states that each stock i can only be "most similar" to one stock j.
- The third constraint $x_{ij} \leq y_j$ for $\mathrm{i}, \mathrm{j} = 1,2,\ldots,\mathrm{n}$ indicates that the similarity between stock i and j can only occur if stock j is selected to be part of the m-stock fund.

- The last constraint denotes that x_{ij} and y_j can only be either 0 or 1. This constraint is not incorporated into the constraint matrix, instead, the variable type is set as 'Binary' in Gurobi.

```
# Add constraints
model.addConstr(gp.quicksum(y[j] for j in range(n)) == m)
model.addConstrs(gp.quicksum(x[i, j] for j in range(n)) == 1 for i in range(n))
model.addConstrs(x[i, j] <= y[j] for i in range(n) for j in range(n))
```

Code Block 3: Constraints for selecting stocks defined using Gurobi in python

Our next step is to calculate the weighted return of each chosen stock to match the returns of the NASDAQ-100. After component stocks that maximize the similarity between the fund and the index have been chosen, the weights of these stocks also need to be determined to complete the fund set up. Therefore, the objective of the weights selection model is to ***determine the weight for each of the component stocks such that discrepancies between the index return and fund return over time are minimized.*** In other words, with q representing the index's return on a given day "t", "w_i" representing the weight of a selected stock "i", and "r_{it}" representing the return of stock "i" on a given day "t", the goal is to minimize the following:

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

```
# Set objective to minimize error
model.setObjective(gp.quicksum(z[t] for t in range(T)), sense=GRB.MINIMIZE)
```

Code Block 4: Objective function to minimize the error defined using Gurobi in python

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

```
model.addConstr(gp.quicksum(w[i] for i in range(m)) == 1)
```

Code Block 5: Constraint to ensure weights add up to 1 defined using Gurobi in python

While using the absolute discrepancies between index and fund returns prevent the negative and positive discrepancies from canceling each other out, it causes additional issues as there is no direct way to represent absolute values within the objective function. One way to cope with this issue is to break q up into two parts by defining an arbitrary decision variable:

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

up into two parts by defining an arbitrary decision variable y_i for each selected stock “i” such that:

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

$$y_i \geq -\left(q_t - \sum_{i=1}^{m} w_i r_{it}\right)$$

```
# Add constraints
model.addConstrs(z[t] >= (index_return.iloc[t] - gp.quicksum(w[i] * return_set.iloc[t, i] for i in range(m)))
for t in range(T))
model.addConstrs(z[t] >= -(index_return.iloc[t] - gp.quicksum(w[i] * return_set.iloc[t, i] for i in range(m)))
for t in range(T))
```

Code Block 6: Constraints for modelling non-linear constraint as linear constraint defined using Gurobi in python

With the above objectives, a model that optimizes weights for selected stocks can be created with formulations stated below:

Decision Variables:

a) w_i : weights of stock “i” selected for portfolio
b) y_i : max value of the possible difference between index return at t and weighted return of selected indexes

Constraints:

a) Sum of weights w_i for m stocks in the fund index equals 1
b) The difference between the index returns and the weighted return of the stocks should be less than y_i

Here $\boldsymbol{w_i \geq 0}$. The return of each stock, r_{it} , is calculated using the pandas ***pct_change()*** function which computes the percentage change from the previous day.

3.2 Without Stock Selection: Mixed Integer Program

With this approach, the stock selection integer programming problem was disregarded. Instead, a mixed integer programming problem where the number of non-zero weights is constrained to be an integer is used. To do this, we also introduced a set of binary variables $y_1, y_2, \ldots.\ y_n$ and added some bigM (smallest bigM set to 1 since our weights are between 0 and 1) constraints that force $w_i = 0$ if $y_i = 0$. The objective function and constraints for this method are as follows:

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

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

Code Block 7: Objective function to minimize the error in MIP problem defined using Gurobi in python

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

$$\sum_{i=1}^{n} y_i = m$$

$$\sum_{i=1}^{n} w_i \leq \mathrm{My_i}$$

```
# Add constraints
model.addConstr(gp.quicksum(w[i] for i in range(n)) == 1)
model.addConstr(gp.quicksum(y[i] for i in range(n)) == m)
model.addConstrs(w[i] <= M * y[i] for i in range(n))
model.addConstrs(z[t] >= (index_return.iloc[t] - gp.quicksum(w[i] * return_set.iloc[t,i] for i in range(n)))
for t in range(T))
model.addConstrs(z[t] >= -(index_return.iloc[t] - gp.quicksum(w[i] * return_set.iloc[t,i] for i in range(n)))
for t in range(T))
```

Code Block 8: Constraints for modeling the problem as MIP problem defined using Gurobi in python

4. Analysis

4.1 Data Pre-processing:

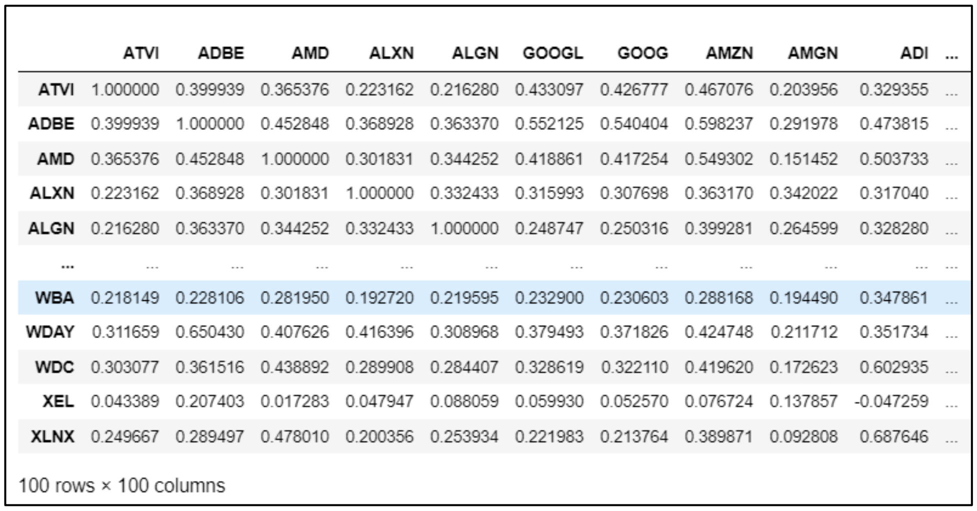
a. **Data Loading:** Two datasets, stocks2019.csv and stocks2020.csv, are loaded into the analysis. These datasets contain historical stock price data.

b. **Date Column Identification:** The project requires time-series analysis, and thus, it is crucial to identify the date column in each dataset. The date column is used as the index for time-series data. To perform time-series analysis, the code identifies the date column in each dataset (***identify_date_column()***).

c. **Calculating Returns:** Daily returns for each stock and the market index are calculated based on the loaded data. The ***pct_change*** method calculates the percentage change in stock prices between consecutive days. The significance of using it is that it helps transform price data into returns - it quantifies the daily percentage price movements and allows for better comparison and analysis of stock and index performance over time. These returns are essential for portfolio optimization and performance analysis.

4.2 Best 5 Stocks

The correlation between each stock in the index and the fund to increase similarity between the index stock and the fund that represents was computed as shown below:



	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	...

100 rows × 100 columns

Table 1: Stocks correlation matrix

After running the model with m = 5. It was found that the five most significant stocks of the index are Liberty Global PLC (multinational telecommunications company), Maxim Integrated (analog and mixed-signal integrated circuits company), Microsoft (multinational technology corporation), Vertex Pharmaceuticals Incorporated (a biopharmaceutical company), Xcel Energy Inc (utility holding company) as shown:

```
The stocks we chose are: LBTYK, MXIM, MSFT, VRTX, XEL
Max correlation total would be: 54.8399
```

Figure 1: Selected stocks and the correlation for m=5

The weights obtained for each of these stocks as a result of the portfolio weights optimization model. As observed, Microsoft has the highest weight, followed by Maxim Integrated.

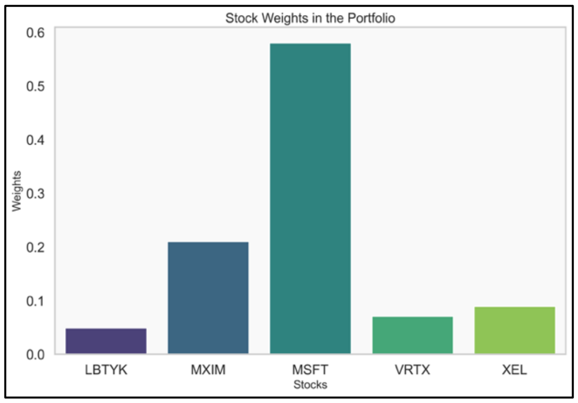
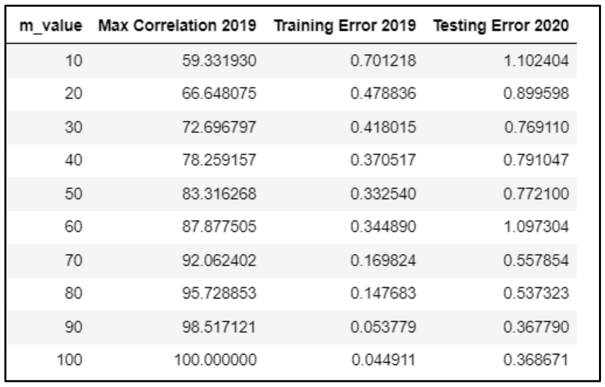


Figure 2: Stock weights for m=5

<u>Observation:</u> The training error using this approach was 0.7892 and the test error was 1.1124. This essentially means that our built portfolio tracks 2020 index decently given we only picked 5 stocks out of the 100 stocks present in the index.

4.3 Analyze the performance of the portfolio for each value of m

Here we defined ***optimize_portfolio()*** function and redid step (2) with m = 10, 20, …, 90, 100. to compare the performance of the various portfolios created. We used the difference between the return of the portfolio and the return of the index as our comparison metric. Below are three figures that summarize our results:



m_value	Max Correlation 2019	Training Error 2019	Testing Error 2020
10	59.331930	0.701218	1.102404
20	66.648075	0.478836	0.899598
30	72.696797	0.418015	0.769110
40	78.259157	0.370517	0.791047
50	83.316268	0.332540	0.772100
60	87.877505	0.344890	1.097304
70	92.062402	0.169824	0.557854
80	95.728853	0.147683	0.537323
90	98.517121	0.053779	0.367790
100	100.000000	0.044911	0.368671

Table 2: Correlation for 2019 & Errors for different m values

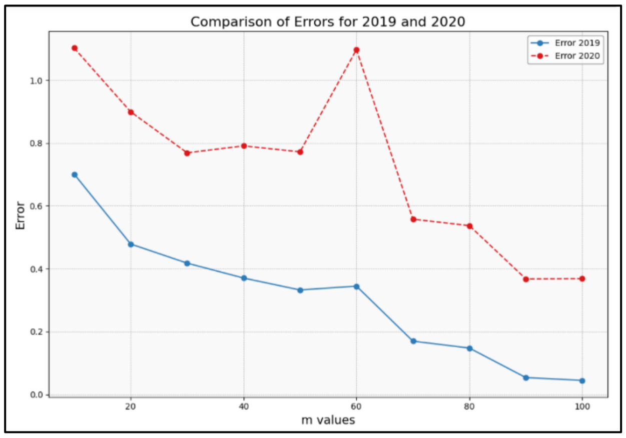


Figure 3: Error comparison between 2019 and 2020 using Method 1

Observations: Based on these results, we can state the following:

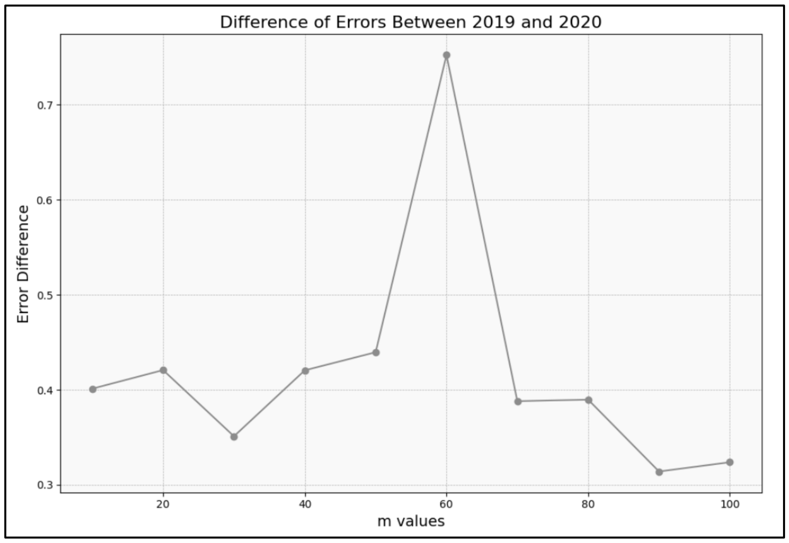


Figure 4: Error difference between 2019 and 2020 for different m values

1. **Max Correlation:** As 'm' increases, the Max Correlation tends to increase (Table 2). This indicates that with a larger number of stocks in the portfolio, the portfolio's correlation with the benchmark (or market) becomes stronger. This might imply that the portfolio tracks the market more closely.

2. **Training Error (2019):** The Training Error (in-sample) decreases as 'm' increases. This suggests that with more stocks in the portfolio, it becomes easier to fit the historical data

accurately. However, extremely low training error might indicate overfitting, where the model fits the training data too closely but doesn't generalize well.

3. **Testing Error (2020):** The Test Error (on 2020 data) initially decreases as 'm' increases, indicating better performance as more stocks are included. However, there seems to be a turning point where ***Test Error starts to increase again post 'm' = 50 until ‘m’ = 70*** (Table 2 and Figure 3 & 4). This suggests that there are temporary ***diminishing returns beyond this point as error does not significantly decrease at the cost of adding more stocks***. Including more stocks doesn't necessarily lead to better out-of-sample performance. This could be due to increased complexity and noise in the portfolio, making it harder to predict future performance.

4. **In-Sample Performance (2019) vs. Out-of-Sample Performance (2020):** In 2019, the portfolio performs well with low Training Error, indicating a good fit to the historical data. Performance in 2019 is usually better because the portfolio is constructed using 2019 data. The model "knows" this data and performs well in-sample. In 2020, the portfolio's performance is relatively worse, indicating that the portfolio didn't perform as well.

5. **Why is the Performance Different in 2019 and 2020?**
   In 2020 there could be several factors like unforeseen market events, economic changes, or shifts in stock behaviors that could cause a difference in performance. The change in performance in 2020 for financial portfolios can be attributed, at least in part, to the COVID-19 pandemic. COVID-19 had a significant and unprecedented impact on global financial markets, causing disruptions and volatility. Investors often responded to these uncertainties by selling off stocks, which led to sharp market declines and increased volatility. This sudden and extreme volatility could have affected the performance of portfolios, causing unexpected losses or gains.

4.4 Let’s Ignore Stock Selection

The code begins by defining the objective: Minimize a measure of error or risk associated with a portfolio. This error is often referred to as the ***"tracking error,"*** which quantifies how closely the portfolio follows the performance of a chosen index or benchmark (in this case, ***index_return***). We used a similar approach as the 1st Method wherein we selected 5, 10, 20,…, 100 stocks for the second method and used the same absolute difference metric. Thus, the smaller the value of the metric the close is the tracking of the portfolio to the index. Through the graph below we can observe the expected i.e., as the number of stocks in the portfolio basket increases, the performance also improves. The ***output from our new approach seems to be better*** than our original approach as our ***test error is much lower***. Moreover, our ***test error almost consistently tracks the training error in its pattern*** which means that our portfolio is doing a good job in tracking the returns.

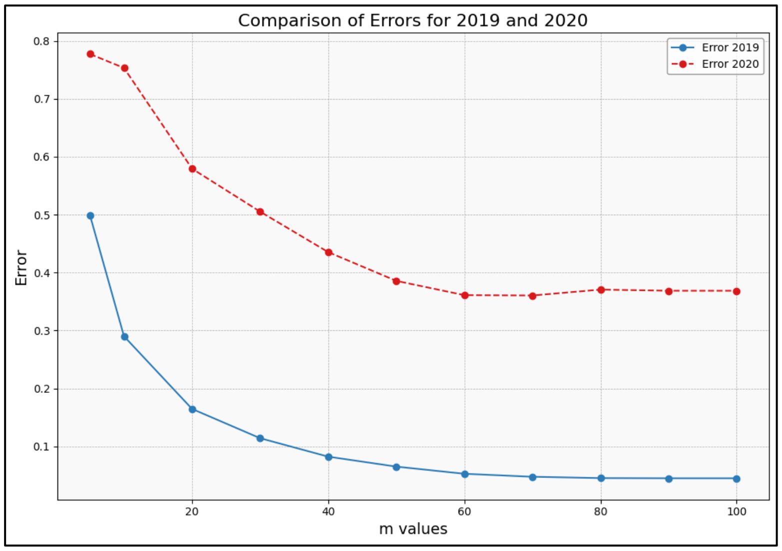


Figure 5: Error comparison between 2019 and 2020 for different m values using method 2

This relation holds true for both 2019 and 2020. Diminishing returns start to settle in after 40 stocks, however the performance stabilizes as well, which indicates that the ***performance of the 40-stock basket*** is quite similar to the portfolio with 100 stocks (Figure 5 & 6).

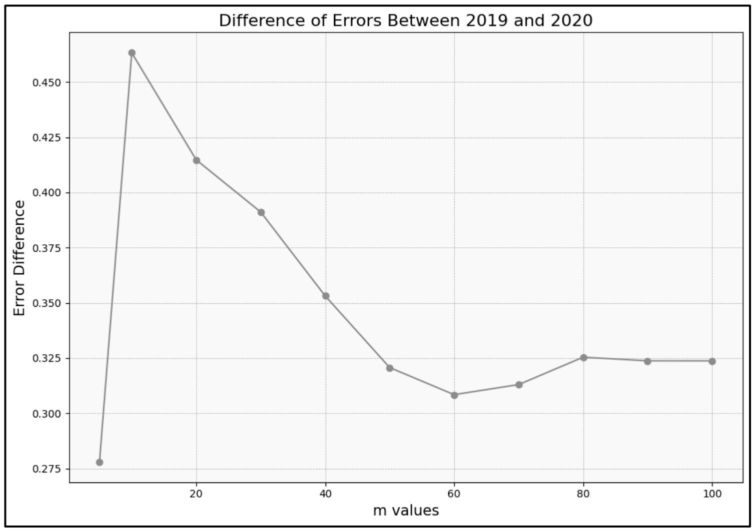


Figure 6: Error difference between 2019 and 2020 for different m values using method 2

4.5 Recommendations & Comparison

In order for us to make a recommendation for how many component stocks to select and how to calculate their weights, we can compare the two approaches and also validate our observations from the nature of the data. Comparing the errors for 2019 and 2020 data from both the approaches, we see:

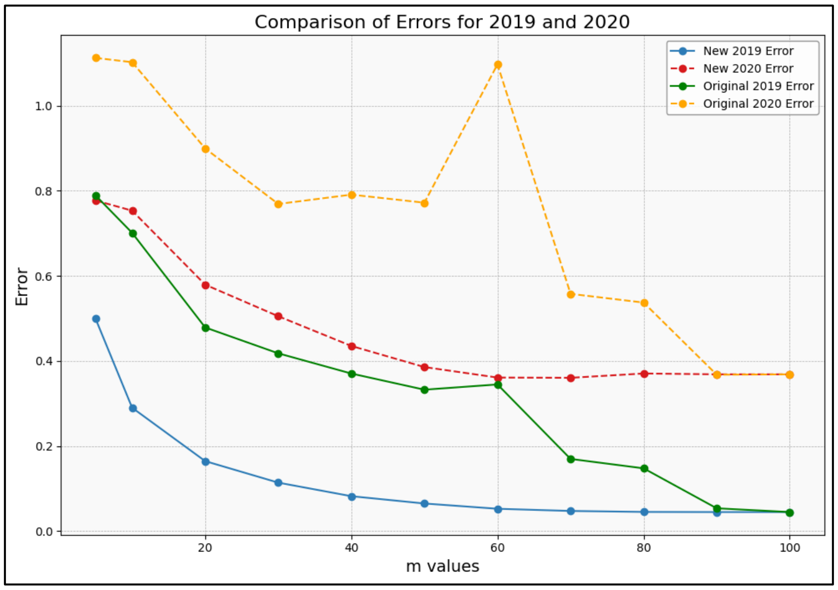


Figure 7: Comparison between errors for 2019 and 2020 from both methods

Method 2 clearly performs better at almost all values of 'm' with lower error. Its performance effectively ***peaks at around m = 50*** (or between 40-50), after which it diminishes. Method 1 has a higher variance, and its performance continuously improves as 'm' increases.

The plot below visually compares the ***incremental performance improvement of our new model*** (with varying 'm' values) in comparison to an original model. The x-axis represents different 'm' values, and the y-axis shows the percentage increase in performance of the new model relative to the original model. The plot helps visualize how changes in the model affect performance:

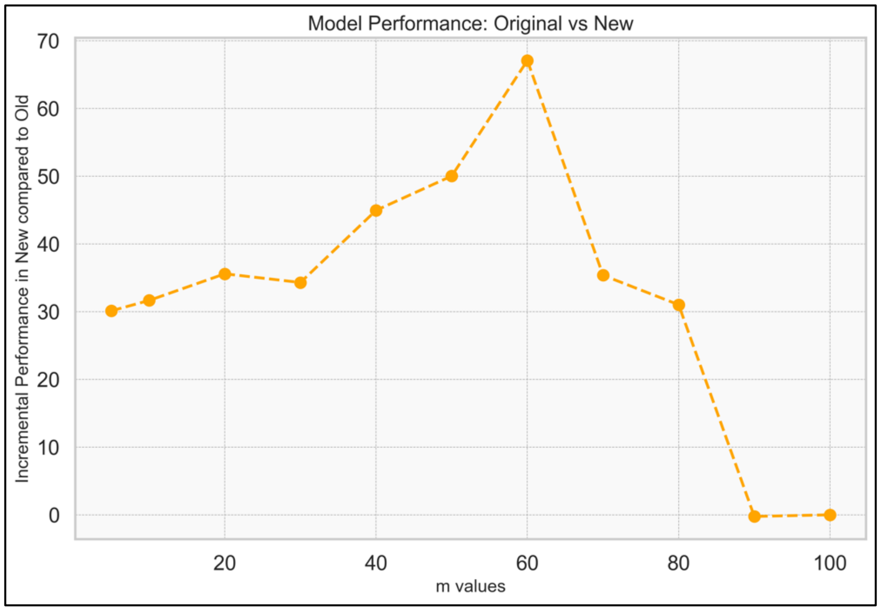


Figure 8: Incremental performance in New Approach compared to original

Hence, we ***recommend using Method 2 for modeling our Index fund*** to mirror the NASDAQ-100. We recommend using ***'m' to be around 40, where we see an 'elbow' in the graph***. This is a small enough number of stocks in our fund that will reasonably mirror the index. We could increase the value of 'm' further if the cost of adding stocks to our portfolio is not too high. The exact value of m can be refined by taking into consideration:

- the cost of adding more stocks to our portfolio (rebalancing costs, price response to trading, etc.) vs. the cost of error while mirroring the NASDAQ-100.
- Another key point is that Method 2 requires considerably higher computing power and time. Only if resources are unavailable, should we use Method 1 for modeling our Index fund. We have currently run the optimization model for ~10 hours.

As far as the data is concerned, when we look at the index returns from 2019 and compare them with 2020 data, we see that ***returns in 2020 have way more volatility as compared to 2019***. The increased volatility in 2020 is likely a result of the uncertainty and economic disruption caused by the COVID-19 pandemic. ***That also explains why our portfolio trained on the 2019 data performed slightly worse on the 2020 data.*** Financial markets react to significant global events, and the pandemic was one of the most impactful events in recent history. In 2020, there were periods of rapid declines (market crashes) followed by periods of recovery. These reactions are reflected in the volatility of index returns.

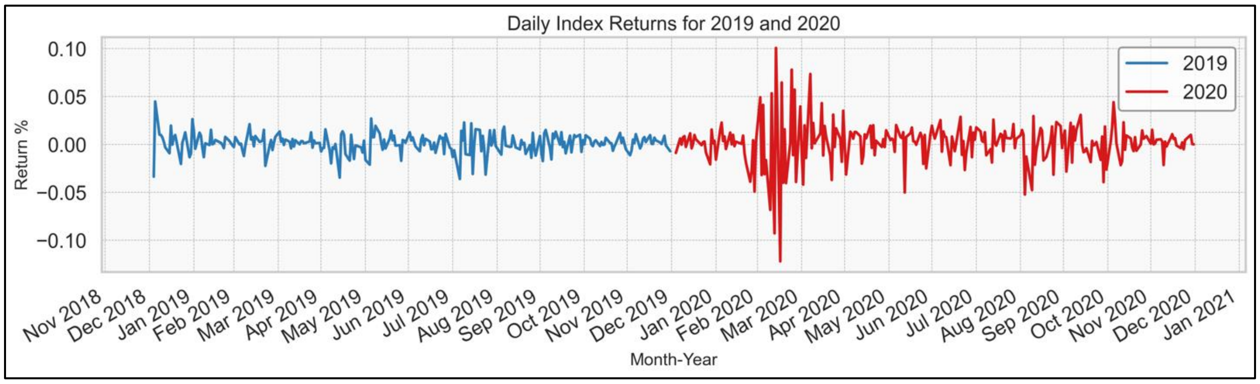


Figure 9: Daily index returns for 2019 and 2020

Note: While the x-axis in the above plot is labeled with "Month-Year" for clarity and ease of interpretation, the data is presented on a daily basis.

Therefore, this highlights the importance of diversification and risk management in investment strategies, as well as the need for a long-term perspective when evaluating market performance.

4.6 Summary

In summary, while there are performance improvements with more stocks ('m') in the portfolio up to a certain point, there are diminishing returns beyond that point. It's crucial to find the right balance in portfolio construction to avoid overfitting and ensure good out-of-sample performance. The difference in performance between 2019 and 2020 highlights the importance of building models that can adapt to changing market conditions.