REPLICATION OF TAIWAN TOTAL RETURN INDEX: TRACKING ERROR MINIMIZATION WITH MATHEMATICAL PROGRAMMING TECHNIQUE

Yu-Ching Liao
Futures Prop. Trading Dept., Capital Securities Corp.
Master of Science in Financial Engineering, UIUC

August 16, 2023

ABSTRACT

This research aims to replicate the Taiwan Stock Exchange Weighted Stock Total Return Index (TAIEX-Total Return Index) using a subset of its 963 constituent stocks. Due to the computational cost of exploring all possible combinations, the study seeks to minimize the tracking error while using the smallest possible subset of stocks. This problem is posed as an optimization of three variables: the length of the training period, the number of constituent stocks used, and the optimization method. The initial dataset comprises the daily closing prices and market-cap weightings of the constituent stocks from November 30, 2018, to July 20, 2023. After addressing missing values and other data irregularities, the study calculates daily returns. The research tests using 100, 200, 300, 400, 500-day training periods, and the top 100, 300, 500, 700, 900 or all stocks by market cap as features. Linear Regression, Lasso Regression, and Conjugate Gradient Research are utilized as optimization engines. The study then extends to incorporate different stock selection methods, such as proportional constituent selection and feature importance selection, instead of just taking the largest stocks by market cap. The results show that: (1) stock selection methods greatly influence model performance, necessitating a fourth dimension to the problem. (2) The best model uses feature importance selection with a random forest and optimization via linear regression to achieve the smallest mean square error. (3) The maximum absolute residual is 6.7 basis points.

Keywords: TAIEX-Total Return Index, Stock Index Replication, Optimization, Linear Regression, Lasso Regression, Conjugate Gradient Research, Feature Importance Selection

1 Problem Formulation

The problem this research aims to address is the replication of the Taiwan Stock Exchange Weighted Stock Total Return Index (TAIEX-Total Return Index) using a subset of its 963 constituent stocks. The successful replication of the Total Return Index can bring about numerous benefits. Not only can it significantly reduce the complexity and cost associated with managing a portfolio of 963 stocks, but it can also serve as a benchmark for the creation of index funds and Exchange-Traded Funds (ETFs). Furthermore, it can assist in the risk management process by allowing for accurate tracking and forecasting of market trends based on a manageable subset of stocks.

The replication of the index using a subset of constituent stocks introduces an inherent trade-off: the tracking error is expected to increase with a reduction in the number of stocks used. The primary challenge lies in finding a subset of stocks and an appropriate model that will achieve the most accurate replication of the Total Return Index, with an acceptable level of tracking error.

Specifically, the problem can be framed as a multi-dimensional optimization problem. The dimensions considered include: the training period length, the number of constituent stocks used, and the optimization method. The performance of the model is primarily evaluated based on the tracking error, with a goal to minimize this error while maintaining a reasonable number of stocks for practical considerations.

To this end, the problem is to identify an optimal subset of constituent stocks and an appropriate modeling approach that provide an accurate replication of the Total Return Index while minimizing the tracking error. The study will consider various lengths of training periods, a range of the number of top stocks by market capitalization, and different optimization methods including Linear Regression, Lasso Regression, and Conjugate Gradient Research. Furthermore, the research will also investigate the effect of alternative stock selection methods, such as proportional constituent selection and feature importance selection.

Ultimately, the research seeks to provide valuable insights and methodologies that could help practitioners to efficiently and accurately replicate the Total Return Index using a subset of its constituent stocks. The findings of this research can contribute significantly to the ongoing efforts in the optimization of index tracking strategies and have far-reaching implications for portfolio management, index funds, ETFs, and risk management.

2 The data of experiment

The experiment utilized daily closing prices and daily weights ranged from November 31th 2018 to July 20th 2023 as the original data of this research. We will then develop all of other kinds of the data from these two original data.

2.1 Description of data

We are provided with 2 csv files where contains 963 columns and 1125 rows, which are TX_Closing.csv and TX_Weights.csv as shown below:

Table 1: Original data-set overview ($TX\_Closing.csv$).

Date	**2330**	**2317**	**2454**	**2412**	…
2018/11/30	199.04	56.88	179.03	88.38	…
2018/12/3	207.43	58.55	185.06	87.55	…
⋮	⋮	⋮	⋮	⋮	⋮

Table 2: Original data-set overview ($TX\_Weights.csv$).

Date	**2330**	**2317**	**2454**	**2412**	…
2018/12/3	0.211463	0.000000	0.013670	0.030018	…
2018/12/4	0.211463	0.000000	0.013670	0.030018	…
⋮	⋮	⋮	⋮	⋮	⋮

2.2 Data pre-processing

Since some of the original data will contain some NaNs, due to the trading halt or unlisted during the time periods, we will have to deal with those NaNs before we calculate the daily return. We will demonstrate this process with setting number of components as 700 and number of days as 800 as an example.

We first extract out the latest 700 $\times$ 800 daily closing prices and the respective daily weights out from the original $TX\_Closing.csv$. After so doing, we first dot-multiply the closing price and the weights so that we can see clearly which rows to drop. With this tested dataframe, we first take the column of unlisted stocks (which first row is NaNs) out from the extracted data, and then drop the rows that contains NaNs, and check the length of the remained data. If the length matches the number of days that we want to test, we then return this the index and the columns so that we can get the closing prices and weights that do not contains the NaNs. If not, we then add one more previous data and redo the whole process until the length matches the number of days. The flow chart following showes the entire process of handling the data.

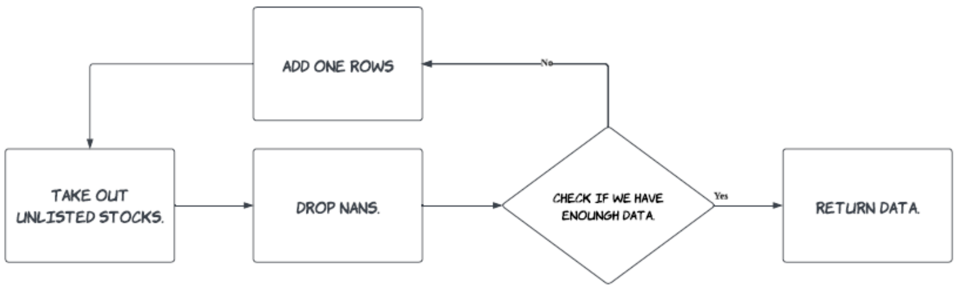


Figure 1: Deal with NaNs.

After so doing, we can obtain the closing price data that is without any NaN after we extract out the unlisted stocks, thus we can calculate the total return index. The reason of not using the true total return index, instead using the one we calculated by ourselves is that some of the components of total return index might change over time, so it is still better if we can calculate the total return index by the components of present.

We can obtain the self-made total return index by the calculation as shown below:

$$Total\_Return\_Index = \sum W_{stock\_id} \times R_{stock\_id} \tag{1}$$

Where R stands as the daily return of the stocks' closing prices, and W stands as the weights of the given stocks in the Taiwanese stock market for the respective date of closing prices. The calculated total return index is shown as below:

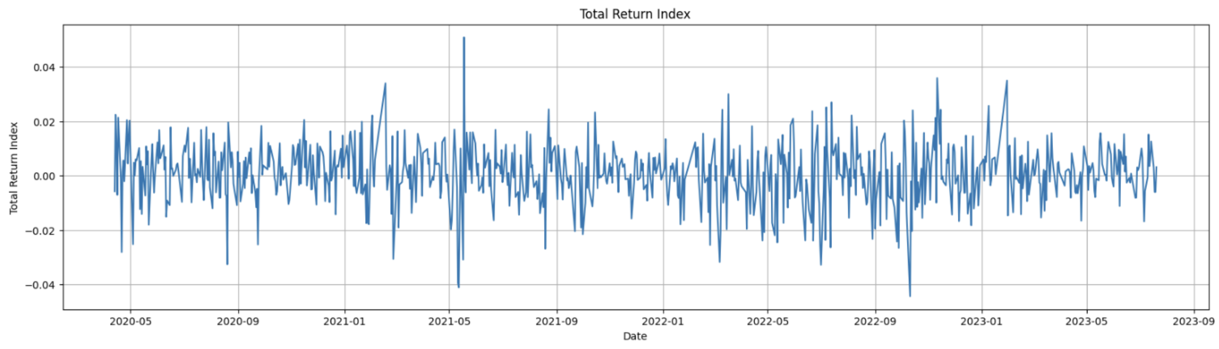


Figure 2: Total Return Index.

In dealing with the regression models (which are Linear Regression and Lasso Regression in our case), we will try the different way to deal with unlisted stocks. Let's use 800 stocks replication as example. We assume that, within these 800 stocks, the stock with stock id 6669 are unlisted in the beginning of the first day of our tested length. To be more specific, this 6669 is still unlisted on the date of the first row of our dateframe.

In dealing with the unlisted stocks, we first remove the return of 6669 from the features, and also substract the "contribution" of 6669 from the total return index. The contribution of the given stock is calculated as follow:

$$Contribution_{6669} = W_{6669} \times R_{6669} \tag{2}$$

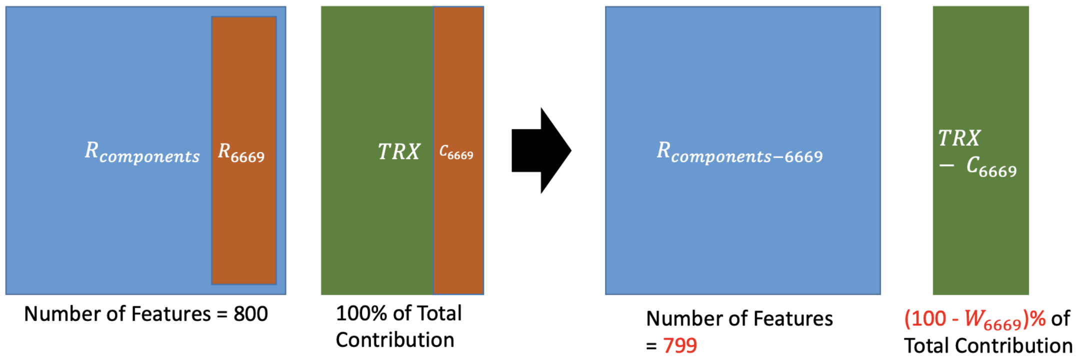


Figure 3: Dealing with unlisted (1)

As shown above, now we can thus have the total return index that is without the contribution of 6669. However, since the total return index is not containing the 6669, it is not summarized as 100%. We thus have to fatten this adjusted total return index to 100% by dividing it with 100-W_{6669}% as shown below:

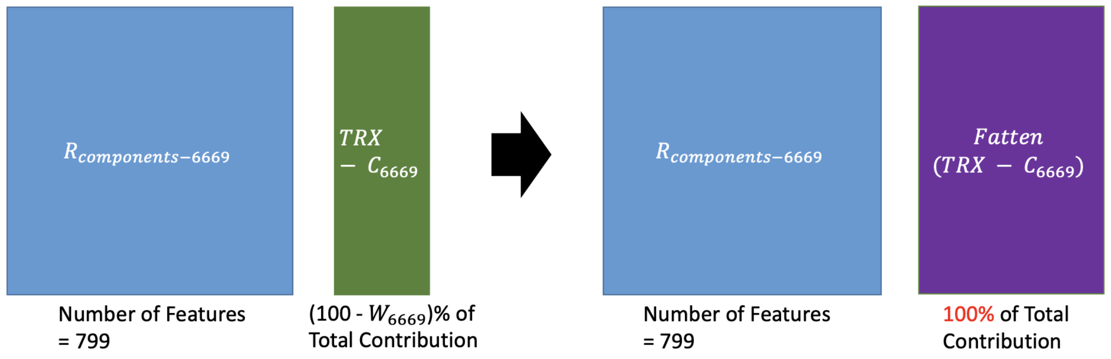


Figure 4: Dealing with unlisted (2)

With this features and fatten total return index without 6669, we can then implement the regression models to get the predicted total return index without 6669.

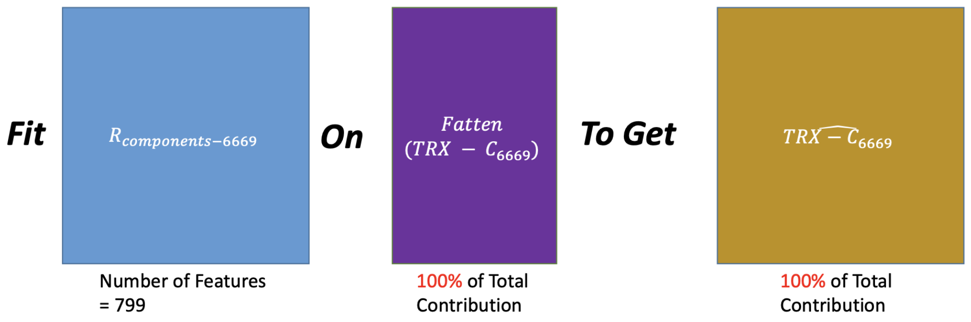

Fit
$R_{components-6669}$
Number of Features = 799
On
Fatten $(TRX - C_{6669})$
100% of Total Contribution
To Get
$\widehat{TRX - C_{6669}}$
100% of Total Contribution

Figure 5: Fitting the model on adjusted features and target.

After so doing, we can then add the contribution of 6669 back to the predicted total return index without 6669, and see the tracking error between our prediction and the true total return index so that we can compare performance of models.

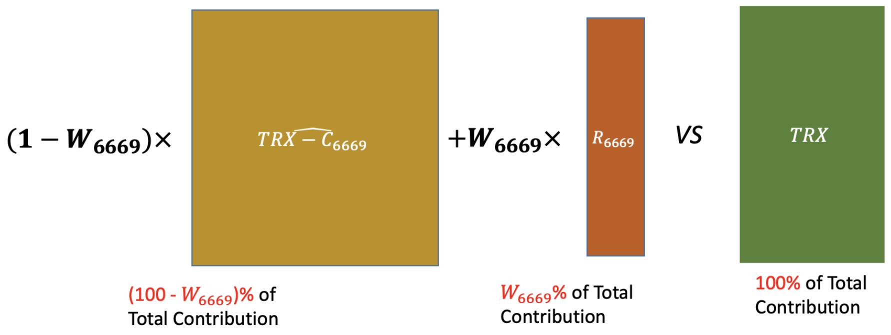

$(1 - W_{6669})\times$
$\widehat{TRX - C_{6669}}$
$+W_{6669}\times$
R_{6669}
VS
TRX
$(100 - W_{6669})$% of Total Contribution
W_{6669}% of Total Contribution
100% of Total Contribution

Figure 6: Comparing the performances of models.

We will implement this process on two regression models we are tested(not including Conjugate Gradient Research) to see if this way of evaluating the model is working. The best way to check the correctness of the method is somehow straightforward that we can see if the tracking error reduced with the number of features grows. If it does not, means that this methods will create too much noise, which can ruin the effectiveness of our models.

3 Methods Review

In this sections, we will introduce some models that we will utilize in this research, which are Linear Regression, Lasso Regression and Conjugate Gradient Research and Random Forest Feature-Importance.

3.1 Linear Regression

Linear regression is a straightforward and widely used statistical method that is prevalent in various fields. It models the relationship between two variables by fitting a linear equation to the observed data. The simplicity of linear regression, coupled with its power to effectively estimate dependent variables, makes it a valuable tool for forecasting and trend analysis.

The fundamental assumption behind the linear regression model is that there exists a linear relationship between the dependent and independent variables. Given a data set of n points, the equation for linear regression is

$$y = \beta_0 + \beta_1 x + \epsilon \tag{3}$$

where y is the dependent variable, x is the independent variable, β_0 and β_1 are parameters of the model which represent the intercept and slope of the line, respectively, and ϵ is the error term.

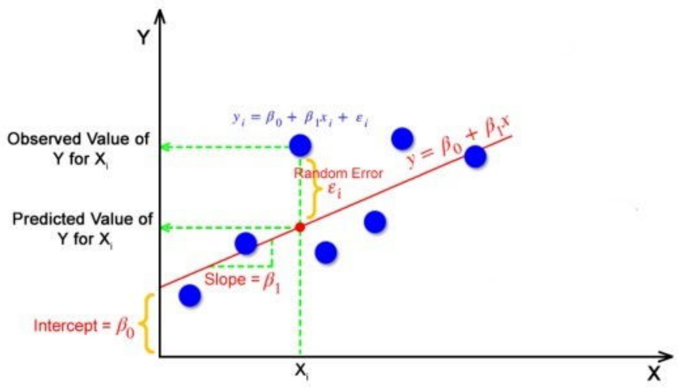

Y
X
$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$
$y = \beta_0 + \beta_1 x$
Observed Value of Y for X_i
Predicted Value of Y for X_i
Random Error ε_i
Slope = β_1
Intercept = β_0
X_i

Figure 7: A simple linear regression model.

3.2 Lasso Regression

Lasso Regression minimizes the sum of squared residuals with a constraint on the absolute value of the coefficients. The objective function is given by:

$$\min_{\beta} \frac{1}{2} \|y - X\beta\|_2^2 + \alpha \|\beta\|_1 \tag{4}$$

where y is the dependent variable (Total Return Index), X represents the constituent stocks, β is the vector of coefficients, and α is the tuning parameter controlling the amount of shrinkage.

3.3 Conjugate Gradient Research

Conjugate Gradient Research is an iterative method that can be used to solve the linear system:

$$Ax = b \tag{5}$$

where A is a symmetric positive-definite matrix, and b is a given vector. The method generates a sequence of conjugate directions and iteratively refines the solution. The update step can be written as:

$$x_{k+1} = x_k + \alpha_k d_k \tag{6}$$

where α_k is the step size, and d_k is the conjugate direction.

In this research, since CGR model require an initial condition to trigger, we will set the weights that is published by Taiwan Futures Exchange (adjusted to sum up to 1) as the initial weights, add set the constraints to sum the weights as 1 also, so that we can grantee that the result of optimization will definitely not worse than simply implementing the weight that is published my Taiwan Futures Exchange.

3.4 Random Forest Feature-Importance

Random Forest does not have a simple mathematical formulation as it is based on an ensemble of decision trees. The feature importance can be computed using metrics like Gini impurity, and for a given tree T and feature j, the importance can be defined as:

$$Importance(j) = \sum_{t \in T:split\_on\_j} (Gini(t) - Gini(t_{left}) - Gini(t_{right})) \tag{7}$$

where the sum is over all nodes t in the tree that split on feature j, and $Gini(t)$ is the Gini impurity of node t.

4 Design of experiments

As mentioned above, this whole experiment can be divided into three main problems waiting to be solved: (1)How much features should we use? (2)How much length of data should we include and (3)Which optimization technique should we implement?

We will thus test the performance of using 100, 300, 500, 700, 900 or 963 as the number of features, 100, 200, 300, 400, 500 as the length of data, and generate the heatmap of each combination of parameters for three models that we are testing.

Note that, in this section, we will pick the features by their weights in the total return index. For example, if I set the number of features as 100, I will then pick the features which weights are top 100 among all weights of all stocks.

4.1 Training and Testing Set Division

As above, we will try several length of data. However, those are the length of "training" dataset. Thus, the exact number of dataset we need will then be the length of training dataset plus the length of testing dataset, and also plus one row of dataset since we will need extra data for the calculation of daily return from closing price.

4.2 Result of Linear Regression

From the result below, we can notice that the tracking error is growing as the number of features grows. This is abnormal, which shows that our method of dealing with the unlisted stocks might be fallacious that will generate more noise as we enhance the number of features. We can as well look into the result of Lasso Regression to confirm this thought.

If this has been confirmed, we will then have to find the another way to deal with unlisted stocks so that we will not include to much noise in our analysis.

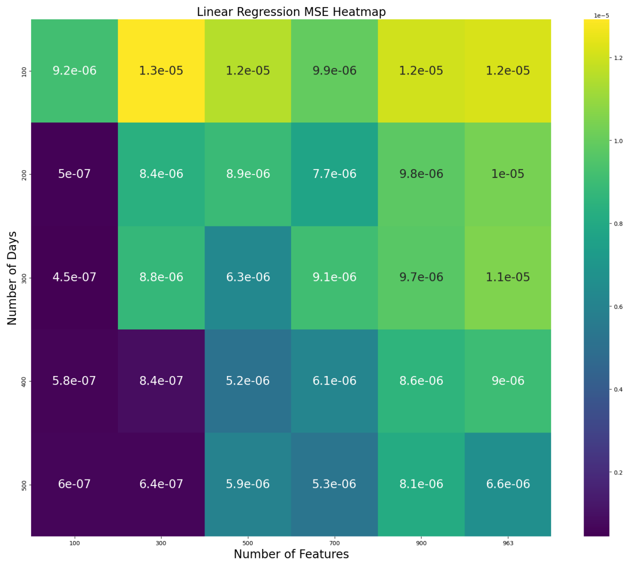


Figure 8: Result of Linear Regression.

4.3 Result of Lasso Regression

As shown below, the result of Lasso Regression also shows that the tracking error is growing as the number of features grows. This indicates that we have to find the another way to deal with unlisted stocks.

Thus, we will look into the method using Conjugate Gradient Research (CGR), which we do not have to deal with the unlisted stocks since we are using the nowadays weights as the initial conditions and regress the optimization based on the previous days' daily return of stocks.

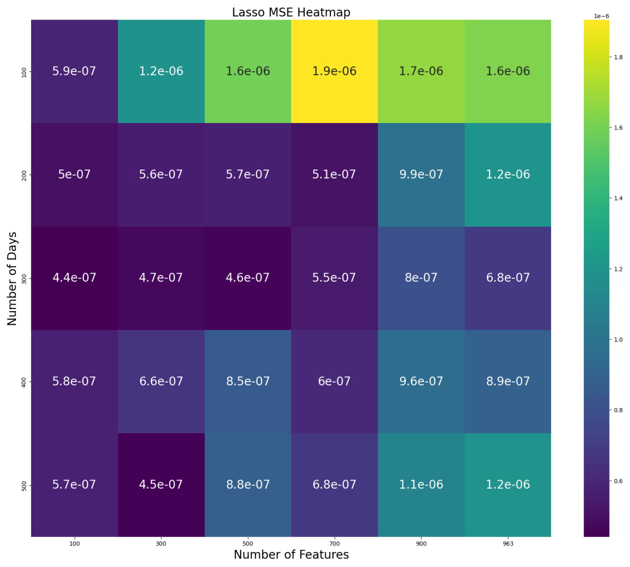


Figure 9: Result of Lasso Regression.

4.4 Result of Conjugate Gradient Research

As shown below, we can see that the result above are abnormal that indicates the fallacy of the method that we are dealing with unlisted stocks. Thus, we expect that for this approach that does not need to deal with unlisted stocks can be helpful in getting the result we desire.

For the recap, for this method, we are using the weight that is given by the Taiwan Futures Exchange as the initial condition, pick the number of features, extract the features and their weights out, adjust the weights of them so that they can be summed up to one, and then implement CGR by training set, and eventually test the parameters we get by applying it on the testing set.

For the result below, we can see that this result are way more rational that, as the number of the features grown, the tracking error as well reduced in their value.

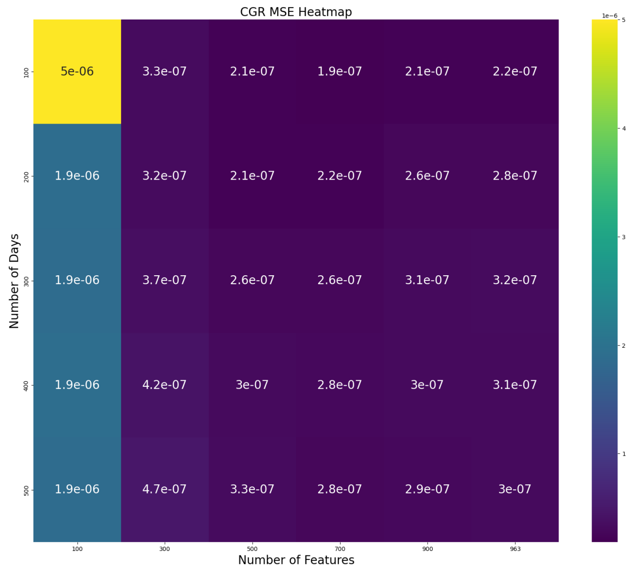


Figure 10: Result of CGR.

With 700 features and 100 data length, we can get the mean-square error of 0.0018 basis points.

Thus, for the following part, we will only utilize CGR as our optimization method.

5 Inspection into the Result and Modification

In this section, we will look into the result that we get, to see if there are any abnormality that should be modified.

5.1 Inspection into Result

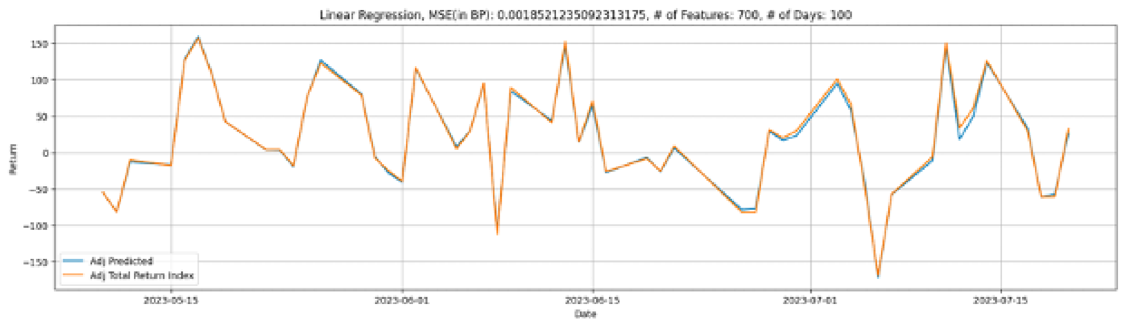


Figure 11: Fitted Value by CGR and Actual Value

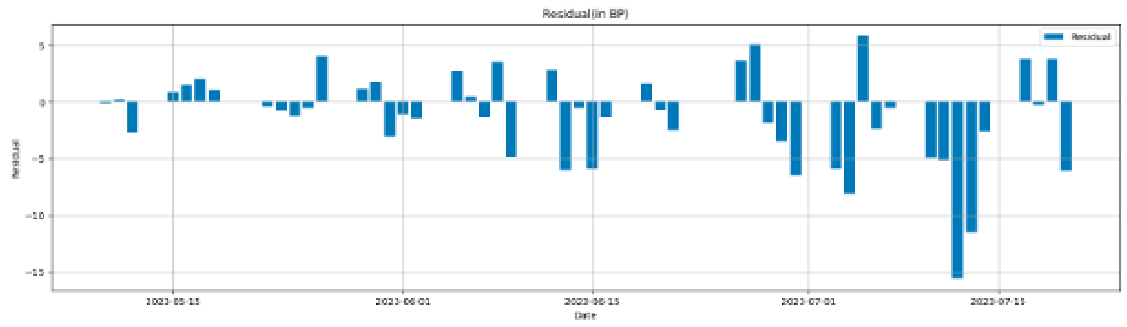


Figure 12: Residual of Testing Set

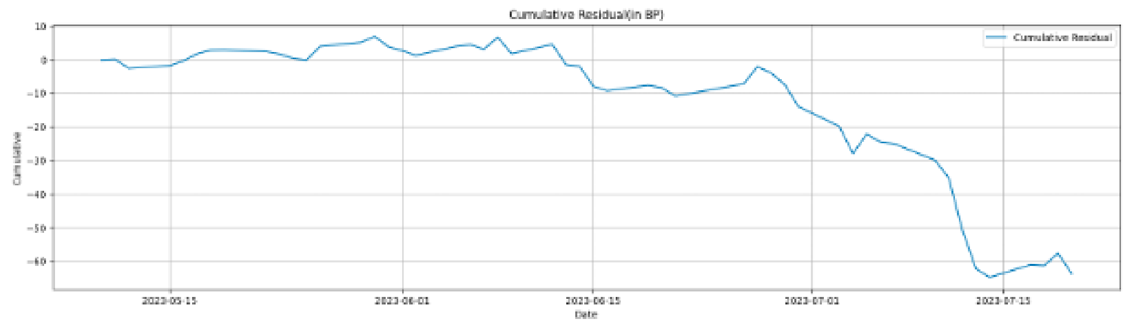


Figure 13: Cumulative Residual of Testing Set

We can see that the largest residual are created slightly before July 15. And the cumulative residual as well underwent the huge draw down during that period.

This can also be seen from other models, shows that our approach of picking the features might be wrong. In the previous research, we picked the stocks by their weights in the total return index. However, instead of picking via the weights, it is better if we can "replicate the look" of the return. That is, if we picked by the weights, since electronic stocks contributed the most value in Taiwanese market, which makes them enjoy more weights, we will instead pick numbers of electronic stocks rather than replicates the shape of the total return index.

Thus, we will try other way to pick the features rather than simply getting those are highly weighted. To be more specific, we add one more selection into our previous problem statements: How should we picked the stocks?

5.2 Modification

We modify the original three problems into four, which are: (1)How should we picked the stocks? (2) How much features should we use? (3)How much length of data should we include and (4)Which optimization technique should we implement?

For the approaches of picking the features, we will try the new three ways instead of selecting by the weights, which are (1)selecting by the section of stocks, (2) selecting by the feature importance and (3) combining two methods above, which specifically is to select the important features from three sections to at once replicate the shape and selecting the important features.

5.2.1 Selecting by the Shape

For this method, we are to try replicating the shape of the total return index. That is, we divided all stocks into "electronic", "banking" and "others", and look into their proportion in the whole Taiwanese market. We then create the portfolio by making the proportion of those three section as same as that of all stocks in Taiwanese market.

If we divide all stocks by those three sections, we can see that about 43% are electronic stocks, and about 4% banking stocks, and rest 53% are those not being included in previous two sections.

As a result, for taking the number of features as 100 as an example, we will then select the 43 from "electronic" stocks, 4 from "banking" and 53 from "others" sections, so that we can get the portfolio whose shape is same as the Taiwanese stock market.

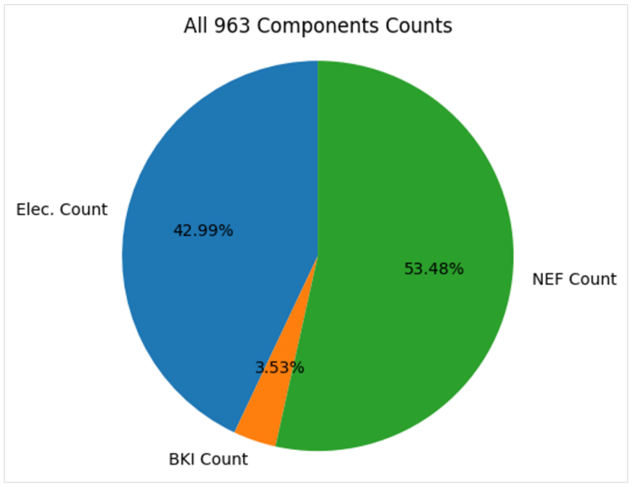


Figure 14: Proportion of all TW stocks divided by sections

By implementing this approach, we can get the result as shown below. Note that, since we already know that the way we deal with unlisted stocks is fallacious, I will only provide the result that we utilizing CGR in the following research.

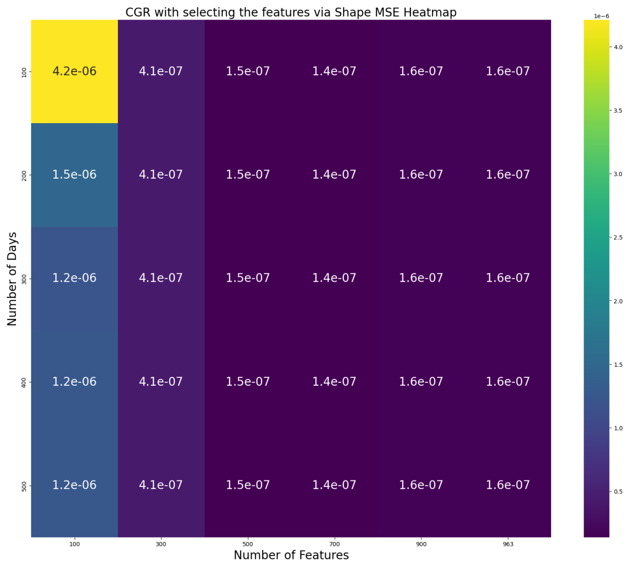


Figure 15: Result of CGR with selecting features via replicating shape of TW Total Index

With 700 features and 100 data length, we can get the mean-square error of 0.0014 basis points, which is lower than just simply picking the features according to their weights.

5.2.2 Selecting by the Feature Importance

For this approach, we will calculate the feature importance of each stock, and get the first "number of features" stocks ranked by the importance.

We are to utilize Random Forest Feature Importance model to get the importance for all stocks, which are being calculated by following procedure: (1) Train on the historical data to get the mean-squared error, (2) Randomly messed up the order of the feature we want to test (3) Train on the messed-up data to get the mean-squared error again (4)

Calculate the absolute value of the difference between two mean-squared errors. The higher difference means that the given feature is more important compare to others.

By selecting the features by their importance, we can get the result as shown below:

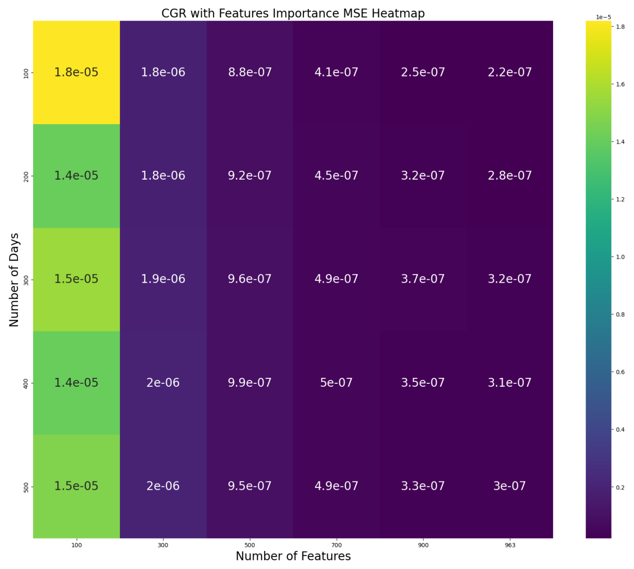


Figure 16: Result of CGR with selecting features via Features Importance

With 700 features and 100 data length, we can get the mean-square error of 0.0062 basis points.

5.2.3 Selecting by the Shape and the Features Importance

With this method, after replicating the shape of TW Total Return Index, we select the stocks that is important in each section.

The result is shown as below:

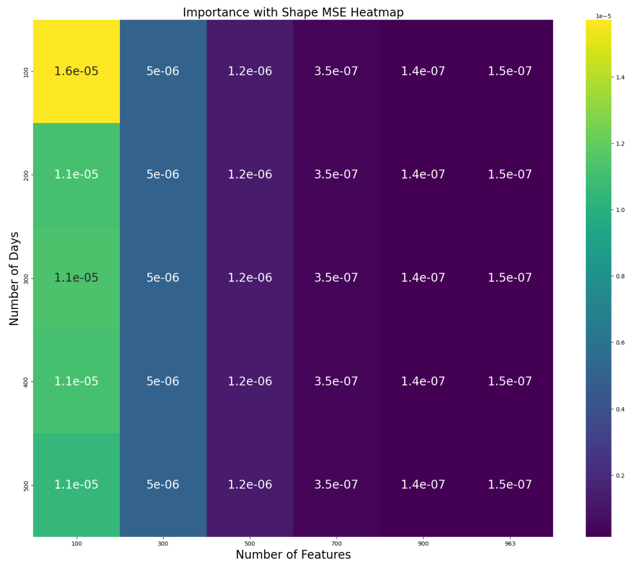


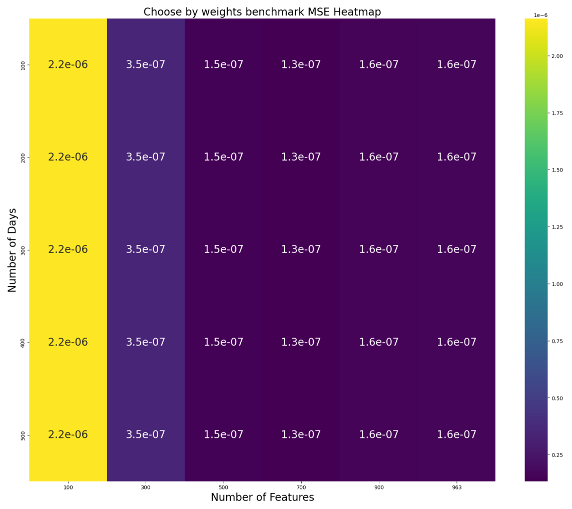
Figure 17: Result of CGR with selecting features via the Shape and Features Importance

With 700 features and 100 data length, we can get the mean-square error of 0.0035 basis points.

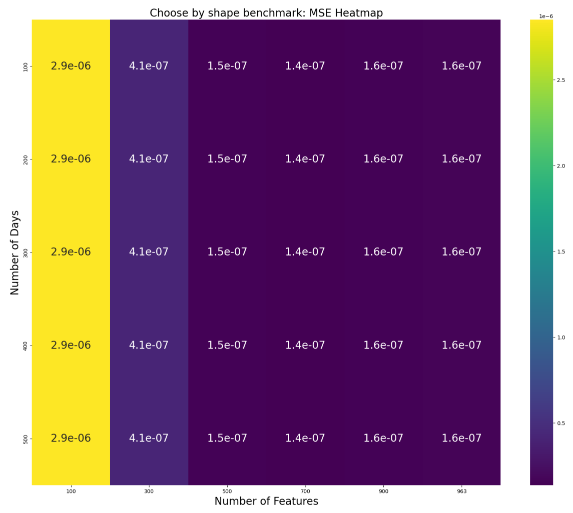
6 Numerical Results

We can compare the result with simply compare to not doing optimization, but instead simply using the initial weights as the weights for the components we selected.

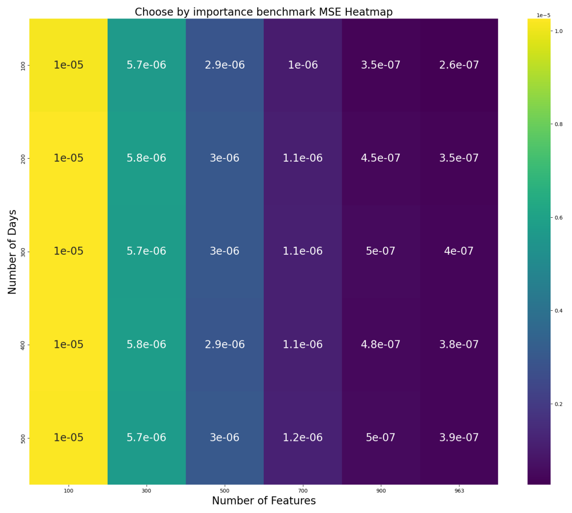
The result of the benchmark is shown as below:



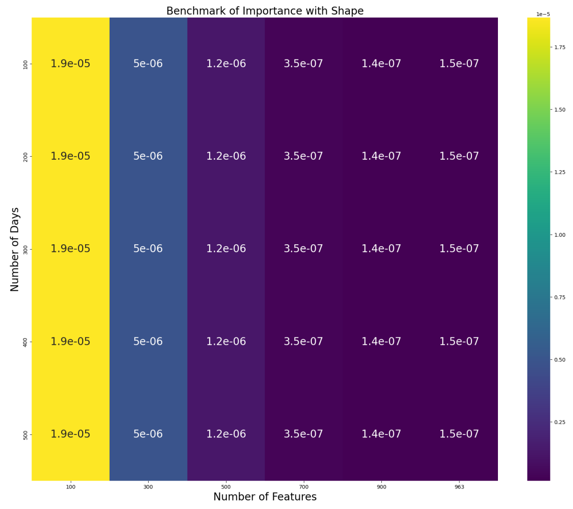
(a) Benchmark of selecting by weights



(b) Benchmark of Replicating Shape



(c) Benchmark of Features Importance



(d) Benchmark of Importance with Shape

Figure 18: Benchmark results

We can compare the result of taking 700 features and 100 length of data to look into the performance of our model.

Table 3: Numerical Result Comparison (700 numbers and 100 length of data)

MSE (in BP)	**By Weight**	**By Replicating Shape**	**By Importance**	**By Importance with Shape**
Benchmark	0.0013	0.0014	0.01	0.0035
CGR	0.0019	0.0014	0.0041	0.0035

With the result above, we can see that the mean-squared error of CGR can provide better result in general. And selecting the features by weights provides the best result among all if we only look into the result with 700 features and 100 length of data.

7 Conclusions

In conclusion, we can see that the way that we deal with the unlisted stocks is fallacious. Thus, it is better if we can find the other way to deal with so, or just simply find the another approach that we do not have to deal with them.

The research shows that the method that utilized CGR, we do not require us to deal with the unlisted stocks, can provide the most reasonable result compare to others.

We also notice that, instead of picking the features by the weights, selecting by the importance actually can provide more accurate result.

Therefore, it is way better if we design our model by selecting the features by replicating the shape of Taiwan Total Return Index and using CGR as the optimizer. For the number of features, we can modify it by the practice, such as how much stocks we can handle at the same time, and the optimal length of the data will be different based on how much number of features that we decided to traded on.