MOMENTUM-BASED PREDICTION IN LONG-SHORT STRATEGIES: THE CASE OF TAIWANESE ELEC. AND BKI INDEX FUTURES

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

July 19, 2023

ABSTRACT

This study introduces an innovative divergent pair trading strategy, also known as a long-short strategy, which is based on Taiwanese Electronic (Elec., Bloomberg ticker: TAIELEC Index) and Banking Industry (BKI, Bloomberg ticker: TAIBANK Index) Index Futures (Bloomberg tickers: TFEZ3 Index and TFBZ3 Index, respectively). These futures contracts are representative indicators of Taiwan's electronic and financial sectors, making them crucial investment and hedging tools in Taiwan's futures market.

The primary aim of this strategy is to anticipate the divergence in prices between Elec. and BKI futures, aiming to generate returns from taking a long position on the stronger future and short on the weaker one during the divergence. The study employs a unique approach to predicting the relative strength of futures based on a selected subset of their components, rather than the futures themselves. This approach was necessitated by the complexity of analyzing momentum factors directly due to the vast number of constituents contained in the futures indices.

To implement this approach, the study first identifies the stocks contributing approximately the top 25% and the bottom 25% to each index (calculated as a component's weight in the index times its five-day cumulative return) to serve as indicator components. These selected components are expected to exert the most significant pull towards strength and weakness in their respective indices.

Once the indicator components are identified, a predictive model, based on the linear regression method in Machine Learning, is trained to forecast future returns. The model uses the cumulative returns from every five-day window within a 200-day period as features to predict the cumulative return of the subsequent day (target). For example, it uses the cumulative returns from day 1-5, day 2-6, up to day 194-199 to predict the cumulative returns of day 6, day 7, up to day 200 respectively. This process allows for the prediction of day 201 cumulative return using the cumulative return of day 195-200.

The cumulative contributions of the Elec. and BKI indicator components form the Elec. and BKI indices respectively, which are then used as a measure of their relative strength. This method facilitates an efficient prediction of index futures strength, simplifying the investment decision-making process.

Keywords Asset Allocation · Long-Short Strategy · Divergent Pair Trading · Taiwanese Electronic and Banking Industry Index Futures · Momentum Factor · Machine Learning · Linear Regression · Predictive Modeling · Contribution Weight

1 Problem Formulation

The TWSE Electronics Index (commonly known as the Electronics Index), a value-weighted index based on electronics industry stocks listed on the Taiwan Stock Exchange (TWSE), is considered a key barometer of the overall performance of Taiwan's electronic sector. This index, encompassing a wide range of companies in high-tech, information software, and communications, often leads the market during periods of economic expansion and is closely watched by investors. Futures contracts based on the Electronics Index, known as "Electronics Futures," offer a mechanism to trade on this sector's overall performance.

The TWSE Financial and Insurance Index (known as the Financial Index), a similarly value-weighted index, reflects the performance of the financial and insurance sectors of the Taiwanese economy. Its futures, known as "Banking Industry (BKI) Futures," serve as a key trading and hedging instrument for this sector.

Given the significance of these two sectors and the prominent role they play in Taiwan's economy, developing an effective strategy for asset allocation between the Electronics and Banking Industry Futures could have considerable implications for investment decision-making. This study, therefore, aims to formulate a novel divergent pair trading strategy, or long-short strategy, based on these two futures.

The strategy's objective is to anticipate and capitalize on the price divergence between Electronics and Banking Industry Futures. However, predicting the futures' relative strength directly is challenging due to the numerous constituents contained in the indices, which can dilute the purity of momentum factors. The problem, thus, is to develop a strategy that uses a selected subset of the indices' components, those contributing significantly to the indices, to predict the relative strength of the futures.

In this new approach, the study identifies the stocks contributing to the top 25% and bottom 25% of each index. By selecting these components that exert the most significant pull towards strength and weakness on five -days cumulative return, the need for a comprehensive analysis of all constituents is avoided.

A further problem to solve is to train a predictive model on these high-contributing components' momentum to predict future returns. This task includes determining the ideal training data and target variables, as well as selecting a suitable machine learning model.

The problems formulated in this study, therefore, involve the design of an asset allocation strategy, the development of a predictive model, and the determination of how to effectively apply the model to drive investment decisions in the Taiwanese Electronics and Banking Industry Futures market.

2 The data of experiment

The experiment utilized data from four primary sources, namely EF_closing.csv, E_weights.csv, F_weights.csv, and index.csv. These files contain the daily closing prices of all the components in the Electronics and Banking Industry indices, the respective market shares of the components in these two indices, and the closing prices of the Electronics and Banking Industry indices themselves. The data spans from March 22nd, 2013 to July 10th, 2023. These comprehensive datasets provide the necessary inputs for formulating and testing the proposed asset allocation strategy, as well as calculating the hedge ratio for the strategy.

2.1 Description of data

The study utilizes four primary data files, namely EF_closing.csv, E_weights.csv, F_weights.csv, and index.csv. These files provide comprehensive data for the components of the Taiwanese Electronic and Banking Industry Indices, as detailed below:

Table 1: Dataset Overview and Description.

$EF\_closing.csv$	Contains the closing prices of all the components in the Electronics and Banking Industry indices from March 22nd, 2013 to July 10th, 2023. Each row in the file corresponds to a specific date, and each column corresponds to a specific stock, with the cell value indicating the closing price of the stock on the respective date.

E_weights.csv	Contains the market shares of the components in the Electronics Index from March 22nd, 2013 to July 10th, 2023. The structure of the file is similar to that of EF_closing.csv, with each cell value indicating the market share of the respective stock on the respective date.
F_weights.csv	Contains the market shares of the components in the Banking Industry Index from March 22nd, 2013 to July 10th, 2023. The file structure is the same as that of E_weights.csv, with each cell value indicating the market share of the respective stock on the respective date.
index.csv	Contains the closing prices of the Electronics Index and the Banking Industry Index over the same time period. This data will be used to calculate the hedge ratio for the proposed strategy.

2.2 Data pre-processing

For the purpose of the experiment, the original closing price data for each constituent stock and the indices themselves were transformed into two different forms, namely, daily returns and cumulative returns.

First, we calculated daily returns by subtracting the previous day's closing price from the current day's closing price and then dividing the result by the previous day's closing price. Second, cumulative returns were calculated by summing up all daily returns from the start of the time series to the current day.

In addition, given that our interest lies in not only the top 25% of stocks that contribute most positively towards the movement of the indices but also the bottom 25% that contribute most negatively, we selected these stocks based on their contributions to the respective indices.

Below we provide an overview of the data transformation:

Table 2: Transformation of original closing price data to daily returns, cumulative returns and five-days return. (Sample: TSMC)

Date	Closing Prices	Daily Returns	Cumulative Returns	5 days Returns
2013-03-25	72.76	0.015350	0.015350	-
2013-03-26	72.39	-0.0005085	0.010265	-
2013-03-27	73.49	0.015195	0.025461	-
2013-03-28	73.12	-0.005035	0.020426	-
2013-03-29	73.49	0.005060	0.025486	0.025537
⋮	⋮	⋮	⋮	⋮

These transformations were performed on all the stocks' daily closing prices in the EF_closing.csv file and index.csv file, creating new data frames for daily, cumulative returns and five-days cumulative returns for each stock. The code to achieve so is shown as following:

```
def closing_in_return_out(path):
    with open(path, "r", encoding="big5", errors="replace") as f:
        daily_closing_EF = pd.read_csv(f, skiprows=1, index_col='Stocks Name')
    daily_closing_EF = daily_closing_EF.fillna(method='ffill')
    daily_closing_EF.index = pd.to_datetime(list(map(lambda x: str(x)[:-3],
        daily_closing_EF.index)))
    daily_return_EF = daily_closing_EF.pct_change()[1:-1]
    five_days_return_EF = daily_closing_EF.pct_change(5)[1:-1]
    cum_return_EF = daily_return_EF.cumsum(axis = 0)
    daily_closing_EF = daily_closing_EF[1:-1]

    return daily_closing_EF, daily_return_EF, cum_return_EF, five_days_return_EF

daily_closing_EF, daily_return_EF, cum_return_EF, five_days_return_EF =
    closing_in_return_out("Data/EF_closing.csv")
```

```
daily_closing_ID, daily_return_ID, cum_return_ID, five_days_return_ID =
    closing_in_return_out("Data/Index.csv")
```

Listing 1: Transforming the Closing Prices

By so doing, we can then obtain the dateframes of closing prices, daily return rates, cumulative return rates and five-days cumulative return rates for components of Elec. and BKI indexes and the indexes themselves respectively.

We as well import the weight data for the calculation of the contribution for following analysis. This can be achievable via following code:

```
def read_data(path):
    with open(path, "r", encoding="big5", errors="replace") as f:
        df = pd.read_csv(f, skiprows=1, index_col='Stocks Name')
    # Assuming df.index is your index
    new_index = [re.search(r'\d+', i).group() for i in df.index]

    # Assigning the new index to the dataframe
    df.index = pd.to_datetime(new_index, format='%Y%m%d')
    df = df[1:-1]
    df = df.fillna(method='ffill')
    return df

daily_weights_E = read_data("Data/E_weights.csv") / 100
daily_weights_F = read_data("Data/F_weights.csv") / 100
```

Listing 2: Read the weights data

Since the weights data was stored in percentage format, we divided them by 100 to convert them back to the percentile scale.

3 Methods Review

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 predict 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

$$y = \beta_0 + \beta_1 x + \epsilon \tag{1}$$

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.

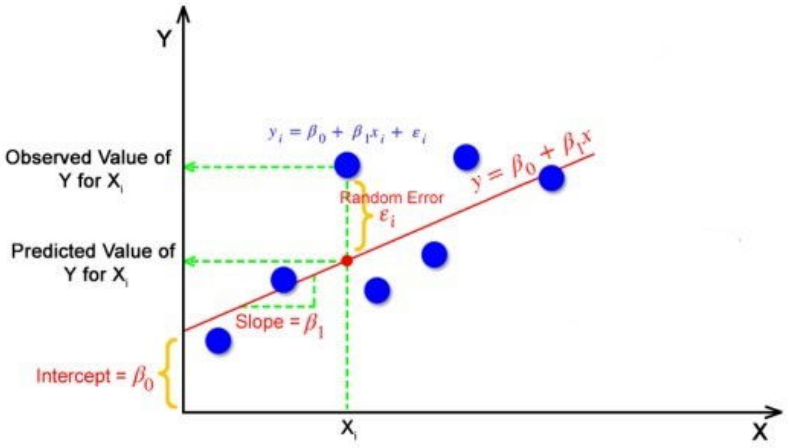

Figure 1: A simple linear regression model.

3.2 Hedge Ratio

The hedge ratio in finance refers to the ratio of the size of a position in a hedging instrument (such as a future or a derivative) to the size of the position being hedged. In other words, it is the proportion of an investment's risk that is removed by hedging. The hedge ratio is often calculated by the covariance between the returns of the portfolio and the returns of the hedging instrument divided by the variance of the returns of the hedging instrument:

$$Hedge\ Ratio = \frac{Cov(r_P, r_H)}{Var(r_H)} \tag{2}$$

where r_P is the return of the portfolio and r_H is the return of the hedging instrument.

3.3 Weighted Average

A weighted average is an average where each observation in the data set is multiplied by a predetermined weight before calculation. The weights reflect the relative importance of each observation. If all the weights are equal, then the weighted average equals the arithmetic mean. The formula for the weighted average is:

$$Weighted\ Average = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{3}$$

where w_i represents the weight for observation i and x_i represents the value of observation i.

4 Design of experiments

We will show how we design the experiments in this section. As the flow chart below, the main process of this experiment can be divided into four parts:

For each day in the dataframe $five\_days\_return\_EF$, we can calculate their contributions by multiplying them with their weights in either Elec. index and BKI index. After so doing, we pick those top and bottom 25% contributors and assign them as the representatives for Elec. and BKI indexes. Thirdly, we implement our moment-based predictor on those selected representatives to get the prediction of next day cumulative return. And lastly, we can then calculate the weighted average based on the re-scaled weights and the predictions of both classes of representatives so that we can observe the possible trends of Elec. and BKI indexes. We will discuss the details of each of these steps in the later sections.

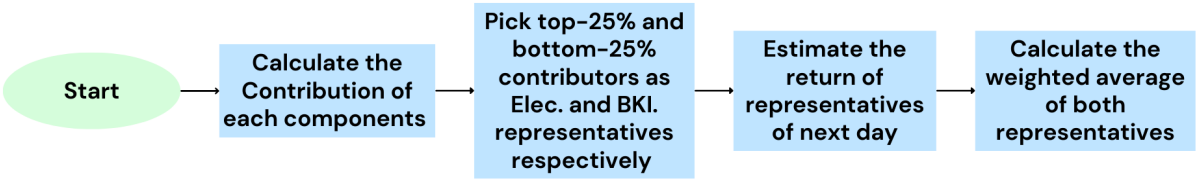

Figure 2: Flow Chart of the experiment.

4.1 Calculation of Contribution

For getting the contribution of each components each day, we multiply the five-days historical cumulative return by their weights on the market. Here are the code to achieve so:

```
E_list = daily_weights_E.columns
F_list = daily_weights_F.columns

five_days_return_E = five_days_return_EF[E_list.values]
five_days_return_F = five_days_return_EF[F_list.values]

contribution_of_momentum_E_last_five_days = five_days_return_E * daily_weights_E
contribution_of_momentum_F_last_five_days = five_days_return_F * daily_weights_F
```

Listing 3: Getting Contributions of Momentum

We first extract the columns from the previously imported weights data so that we can have the list of all components of either Elec.and BKI indexes and stored them in $E\_list.csv$ and $F\_list.csv$. We can then extract the five-days cumulative return of Elec. Index components and that of BKI index respectively. Eventually, we can then calculate the contribution of those components.

After so doing, we can then transform the this contribution of components to the percentile scale so that we can compare them with each other in the later steps. Following is the code to achieve so:

```
contribution_of_momentum_E_last_five_days_in_percent =
    contribution_of_momentum_E_last_five_days.apply(lambda row: row / row.sum(),
    axis=1)
contribution_of_momentum_F_last_five_days_in_percent =
    contribution_of_momentum_F_last_five_days.apply(lambda row: row / row.sum(),
    axis=1)
```

Listing 4: Getting Percentile-Contributions of Momentum

Below we provide an overview of the data transformation:

Table 3: Transformation of original 5 days return data to percentile-contribution of momentum in last five days. (Sample: TSMC)

Date	5 days Returns	Daily Weights in Elec. Idx.	Contribution	Percentile Contribution
2013-03-29	0.025537	(×) 0.234299	(=) 0.005983	0.297934
2013-04-01	0.014981	(×) 0.235792	(=) 0.003532	0.314699
2013-04-02	0.015195	(×) 0.234088	(=) 0.003557	0.399366
2013-04-03	0.000000	(×) 0.233452	(=) 0.000000	0.000000
2013-04-08	-0.014907	(×) 0.233712	(=) -0.03484	0.346648
⋮	⋮	⋮	⋮	⋮

With getting all the contributions for all components of the indexes, we can then selection those have top and bottom 25% of the contributions as the representatives.

4.2 Selection of Representatives

After we have obtained the contributions for both of the indexes, we can then select those top and bottom 25% contributors as the representatives. The reason of picking both top and bottom 25% is because, by doing so, we both pulling and dragging force of indexes can be included in our predictions thus making our prediction to be more accurate.

This can be achievable via following code:

```
def top_and_bottom_contributors(df, threshold=0.25):
    result = []
    for date, row in df.iterrows():
        sorted_row = row.sort_values(ascending=False)
        cumulative_sum_desc = sorted_row.cumsum()
        over_threshold_desc = cumulative_sum_desc[cumulative_sum_desc > threshold]
        if not over_threshold_desc.empty:
            top_contributors = sorted_row.loc[:over_threshold_desc.index[0]]
        else:
            top_contributors = sorted_row

        sorted_row_asc = row.sort_values(ascending=True)
        cumulative_sum_asc = sorted_row_asc.cumsum()
        over_threshold_asc = cumulative_sum_asc[cumulative_sum_asc < - threshold]
        if not over_threshold_asc.empty:
            bottom_contributors = sorted_row_asc.loc[:over_threshold_asc.index[0]]
        else:
            bottom_contributors = sorted_row_asc

        contributors = top_contributors.index.tolist() + bottom_contributors.index
            .tolist()
        contributions = top_contributors.tolist() + bottom_contributors.tolist()
        result.append(pd.Series({'contributors': contributors, 'contributions':
            contributions}, name=date))

    return pd.DataFrame(result)
```

Listing 5: Getting Top 25% and Bottom 25% of contributors

Shown as following, we can create the dataframe storing all of the contributors and their contributions each day:

Table 4: Top and Bottom Elec. Contributors and their Contributions.
(Sample: Contributions of Elec. stocks on 2013-04-10)

Date	Contributors.	Contributions
2013-04-10	[TSMC, MTK, TPK-KY,...]	[0.4017349963794615, -0.0948408245465958, -0.0...
⋮	⋮	⋮

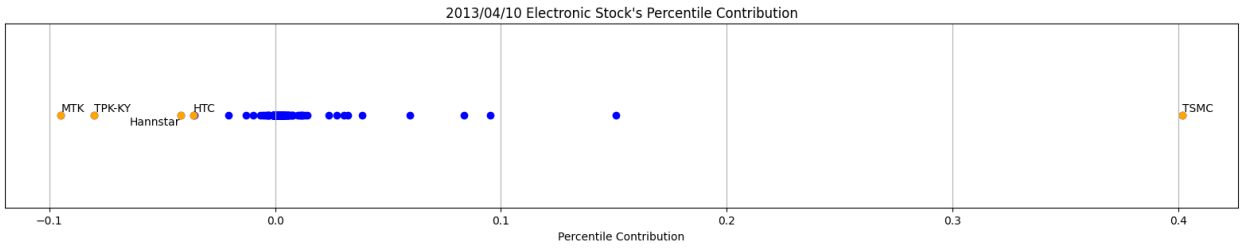

Figure 3: Percentile Contribution of Elec. on 2013/04/10

As shown above, we can notice that the contributors of the top- and bottom-25% can be selected out, which represent the main pulling and dragging force of the Elec. Index. Here I as well provide the sample contributors and contributions for BKI stocks.

Table 5: Top and Bottom BKI Contributors and their Contributions. (Sample: Contributions of BKI stocks on 2013-11-20)

Date	Contributors.	Contributions
2013-04-10	['CFH', 'MFH', 'ECB',...]	[0.5898557576667708, -0.18815531160708715, -0...
⋮	⋮	⋮

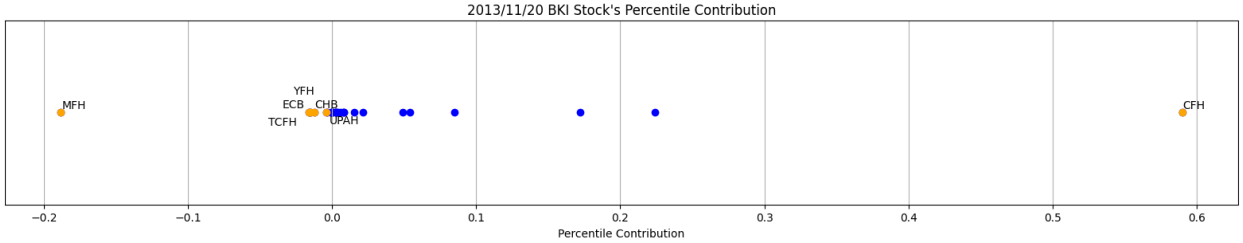

Figure 4: Percentile Contribution of BKI Index on 2013/11/20

After we had selected those representatives out we can then move on to the prediction of next day return. In the next section, we will discuss the methodology of our prediction and its performance.

4.3 Momentum-Based Prediction on Representatives

After we get the representatives for both Elec. and BKI index, the next step for us is to predict the next day return via momentum. In this section, we will utilize the return of TSMC stock to demonstrate our methodology.

We first chop out the cumulative return with window of 200 as our historical training dataset. That is, we are to utilize the 200 historical data to predict the next day cumulative return.

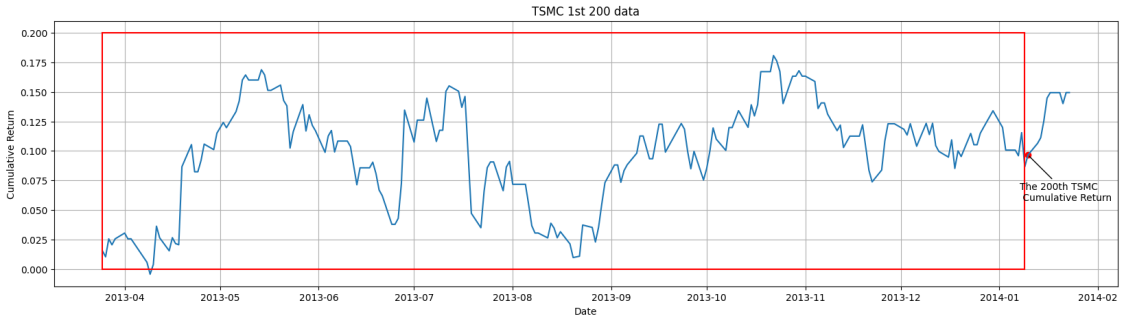

Figure 5: 1st 200 TSMC cumulative Return

After we obtain our historical data, we can then move on to the second step that fit the Linear Regression model on the historical data. Since we are to train on the momentum, it is of necessity to considerate "how does the momentum of previous days infect the future" and "will this infection maintain". Therefore, the most reasonable way is to chop the historical data with several minor moving window, and assign the all the "next days" of all minor moving window as the target we want to fit on.

To be more specific, let's say we are to chop all the first 200-days historical data via 5-data moving window, the features and targets will then be as shown below, where i represent the index of data in 200-days historical data, and t_i for date index and R_i for cumulative return of TSMC. Note that index i starts from 0, thus the R_{199} actually represent the 200$^{\text{th}}$ data.

Table 6: Training Features and Training Target of Momentum-Based Prediction Model (Sample: TSMC)

Index, i	**Date Index,** t_i	**Date**	**Training Features** (X_train)	**Training Targets**(y_train)
0	t_0	2013-03-25	$[R_0, R_1, R_2, R_3, R_4]$	R_5
1	t_1	2013-03-26	$[R_1, R_2, R_3, R_4, R_5]$	R_6
⋮	⋮	⋮	⋮	⋮
194	t_{194}	2014-01-02	$[R_{194}, R_{195}, R_{196}, R_{197}, R_{198}]$	R_{199}

As shown below, the green and blue windows are two of the features, and the green and blue dots beside them are their respective targets.

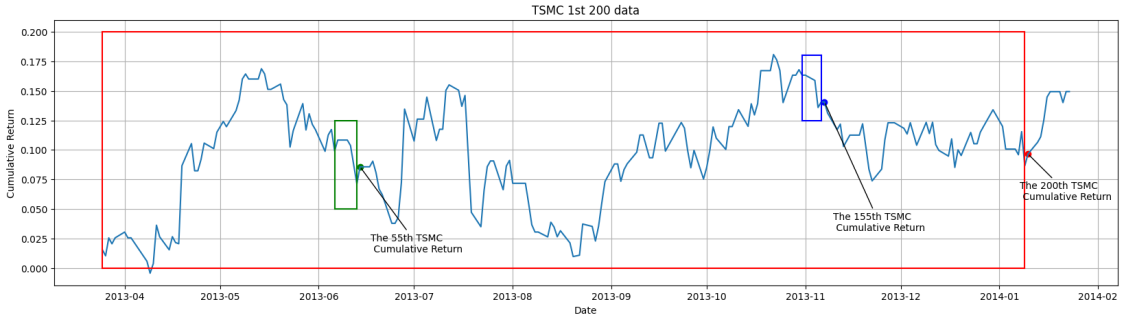

Figure 6: Features and Targets

After we have successfully trained our model, we can then implementing this model for predicting the 201th data. By repeating this whole process, we can then create the prediction for all the data we have to test the effectiveness of this methodology. As shown below, this way of prediction can effectively follow the trend and thus more fitted the actual return. On the data ranged from March 22nd, 2013 to July 10th, 2023, our model can reduce the mean square error down to 0.04732%.

Table 7: Testing Features and Prediction of Momentum-Based Prediction Model (Sample: TSMC)

Index, i	**Date Index,** t_i	**Date**	**Testing Features** (X_test)	**Prediction** (y_pred)
195	t_{195}	2014-01-03	$[R_{195}, R_{196}, R_{197}, R_{198}, R_{199}]$	$\hat{R}_{200}$

This methodology is achievable via following code:

```
def LR_predictor(df):
    y_pred = []

    for date_index in range(200, len(df.index)):
        data = df.iloc[date_index - 200: date_index]

        # Convert the data within the range of [0, 1]
        scaler = MinMaxScaler(feature_range=(0, 1))
        scaled_data = scaler.fit_transform(data.values.reshape(-1, 1))

        # Convert the data into the format of samples, time steps, features
        X, y = [], []

        # Using historical 5 days to predict the return of next day
        for i in range(5, len(scaled_data)):
            X.append(scaled_data[i-5:i, 0])
            y.append(scaled_data[i, 0])
        X, y = np.array(X), np.array(y)

        # Creating Training and Testing Sets
        X_train, X_test = X, X[-1:]
```

Figure 7: Actual Return and Prediction

```
        y_train = y

        # Build and Trained the model
        lr_model = LinearRegression()
        lr_model.fit(X_train, y_train)

        prediction = lr_model.predict(X_test)
        prediction = scaler.inverse_transform(prediction.reshape(-1, 1))

        y_pred.append(prediction[0])
    return df.index[200:], y_pred
x, y = LR_predictor(tsmc)
```

Listing 6: Momentum-Based Predictors

We can then proceed this on all representatives, so that we can obtain all the prediction for getting the estimated trend for next step.

4.4 Calculate the Weighted Average for Predicted Trend

After we have all prediction for all representatives, we can calculate the weighted average of both Elec. and BKI representatives with the daily weight of representatives.

Table 8: Calculation of Estimated Trend of Index(Sample: 2013/07/10 Data)

Elec. Representatives	Weights in Elec.	Weights in all Elec. Representatives	Predicted Return	Scores
TSMC	0.453963	0.940162	(×) 2.391962	(=) 2.248832
Wistron	0.008003	0.016574	(×) 2.107840	(=) 0.034936
LARGAN	0.008640	0.017894	(×) 2.147432	(=) 0.038425
GIGABYTE	0.004704	0.009742	(×) 3.541750	(=) 0.034504
Wiwynn	0.007546	0.015628	(×) 1.745218	(=) 0.027274
Total:		100%	Total:	2.383971

We then regress this on all the daily data we have, thus we can get all of the scoring everyday.

Table 9: Scoring Pool

Date	Score of Elec. Stocks	Score of BKI Stocks
2014/01/10	0.152452	0.192100
2014/01/13	0.132931	0.222698
2014/01/14	0.138823	0.183980
2014/01/15	0.180186	0.175229

⋮	⋮	⋮

In here, since we are applying the long-short strategy based on the hedge ratio, we have to calculate the hedge ratio of overall time. We as well utilize the 200-days windows to get the hedge ratio of everyday in our dataset. To be more specific, we utilize the historical 200-days of the return data to estimate the hedge ratio of next day. Following is the Python code to achieve so:

```
def hedge_ratio_calculator(daily_return_ID, x_name = "Elec",
                           y_name = "BKI", outside_window = 200):

    model = LinearRegression()
    hedge_ratio = []

    for index in range(len(daily_return_ID)):
        if index < outside_window:
            hedge_ratio.append("nan")
        else:
            X = daily_return_ID[x_name].iloc[index-outside_window:index].values
            y = daily_return_ID[y_name].iloc[index-outside_window:index].values

            X = X.reshape(-1, 1)
            model.fit(X, y)
            hedge_ratio.append(model.coef_[0])

    hedge_ratio = pd.DataFrame(hedge_ratio, index=daily_return_ID.index, columns=[
        'Hedge Ratio'])
    return hedge_ratio
Ratio = hedge_ratio_calculator(daily_return_ID)
```

Listing 7: Momentum-Based Predictors

We utilize the daily return of previously imported Elec. and BKI index, to calculate the best hedge ratio for everyday. After we have done so, we can then move on to the calculation of estimiated trend of Elec. and BKI indexes.

Table 10: Scoring Pool with Hedge Ratio

Date	Score of Elec. Stocks	Score of BKI Stocks	Hedge Ratio	Estimated LS Strategy Return
2014/01/10	0.152452	0.192100	0.610984	0.098954
2014/01/13	0.132931	0.222698	0.605758	0.142174
2014/01/14	0.138823	0.183980	0.607040	0.099709
2014/01/15	0.180186	0.175229	0.604424	0.06632
⋮	⋮	⋮	⋮	⋮

We can then plot the estimated trend out and compare them with actual return of long-short strategy as follow:

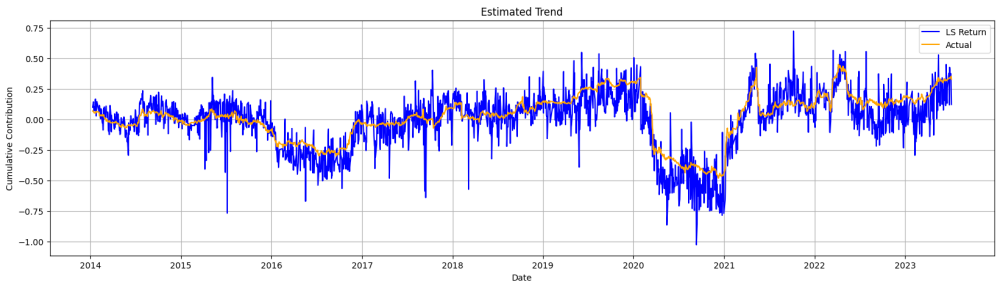

Figure 8: Estimated Trend V.S. Actual Return of LS Strategy

From this plot, we can see that our estimated trend, despite containing some noise, it still flows the trend of actual return of our LS strategy. We thus process this estimated return to exact out the signals.

We first calculate the difference between each return:

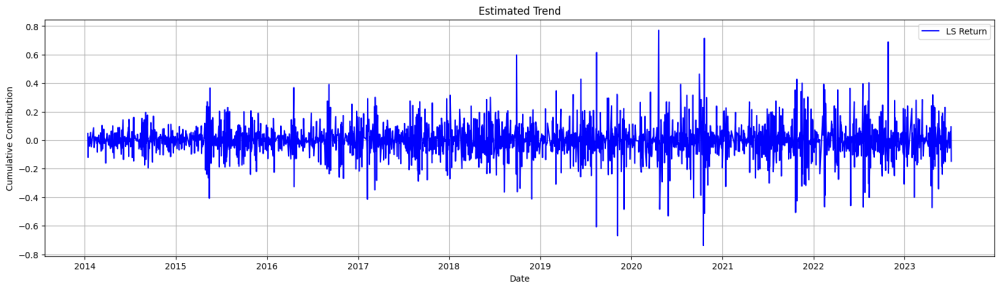

Figure 9: Signals from Estimated Trend

We can see that, if we can optimize the threshold of this signals based on minimizing the maximum draw down and still maintaining the positive PnL, we can then obtain the optimize the entry and exit points.

Following I have provided the code and the plot of the optimized strategy:

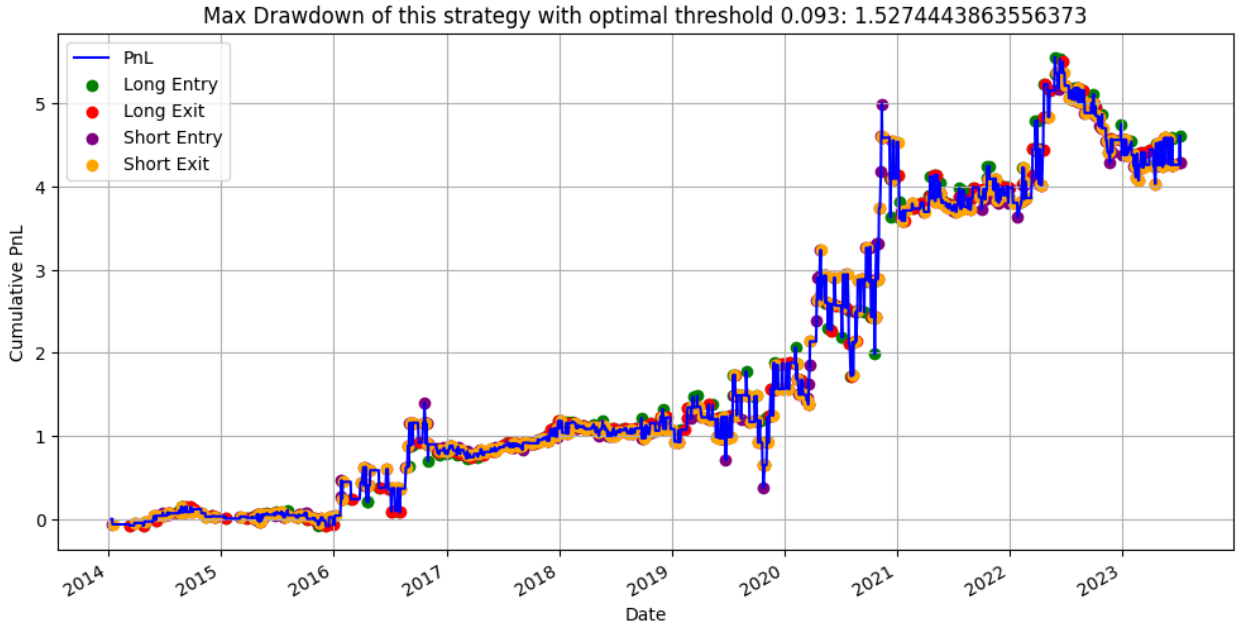

Figure 10: Optimized Strategy

```python
# Assume df is your dataframe
df = scoring_pools.copy()

def calculate_pnl_mdd_counts(df, threshold):
    df['pnl'] = 0
    df['position'] = 0  # Position status
    df['entry_exit'] = 0  # Mark entries and exits

    for i in range(1, len(df)):
        # Check if we need to exit
        if df['position'].iloc[i-1] == 1 and df['Signals DIFF'].iloc[i] <=
            threshold:
            df['position'].iloc[i] = 0
```

```
            df['entry_exit'].iloc[i] = -1
        elif df['position'].iloc[i-1] == -1 and df['Signals DIFF'].iloc[i] >= -
            threshold:
            df['position'].iloc[i] = 0
            df['entry_exit'].iloc[i] = 1
        # Or enter a position
        elif df['position'].iloc[i-1] == 0 and df['Signals DIFF'].iloc[i] >
            threshold:
            df['position'].iloc[i] = 1
            df['entry_exit'].iloc[i] = 1
        elif df['position'].iloc[i-1] == 0 and df['Signals DIFF'].iloc[i] < -
            threshold:
            df['position'].iloc[i] = -1
            df['entry_exit'].iloc[i] = -1
        # Or continue holding
        else:
            df['position'].iloc[i] = df['position'].iloc[i-1]

        # Calculate PnL
        df['pnl'].iloc[i] = df['position'].iloc[i] * df['Actual Return'].iloc[i]

    df['cumulative_pnl'] = df['pnl'].cumsum()
    df['running_max'] = np.maximum.accumulate(df['cumulative_pnl'])
    df['drawdown'] = df['running_max'] - df['cumulative_pnl']

    max_drawdown = df['drawdown'].max()
    final_pnl = df['cumulative_pnl'].iloc[-1]

    # Count the number of long and short entries and exits
    long_entries = sum((df['entry_exit'] == 1) & (df['position'] == 1))
    long_exits = sum((df['entry_exit'] == -1) & (df['position'] == 0) & (df['
        position'].shift() == 1))
    short_entries = sum((df['entry_exit'] == -1) & (df['position'] == -1))
    short_exits = sum((df['entry_exit'] == 1) & (df['position'] == 0) & (df['
        position'].shift() == -1))

    return final_pnl, max_drawdown, long_entries, long_exits, short_entries,
        short_exits

# Grid search over potential threshold values
thresholds = np.arange(0.001, 0.2, 0.001)  # Adjust the range and step size as
    necessary
results = []
for threshold in thresholds:
    pnl, mdd, le, lx, se, sx = calculate_pnl_mdd_counts(df.copy(), threshold)
    results.append((threshold, pnl, mdd, le, lx, se, sx))

# Convert results to a DataFrame for easier manipulation
results_df = pd.DataFrame(results, columns=['threshold', 'pnl', 'mdd', '
    long_entries', 'long_exits', 'short_entries', 'short_exits'])

# Filter for scenarios where pnl is positive
results_df = results_df[results_df['pnl'] > 0]

# Find the threshold that minimizes mdd
optimal_threshold = results_df.loc[results_df['mdd'].idxmin(), 'threshold']

# Calculate PnL, MDD, and counts using the optimal threshold
final_pnl, max_drawdown, long_entries, long_exits, short_entries, short_exits =
    calculate_pnl_mdd_counts(df, optimal_threshold)
```

Listing 8: Optimized Strategy

With the optimization, we can then maintain the positive return yet still minimizing the maximum draw down.

5 Numerical Results

After the optimization, we can obtain the result as follow:

Table 11: Numerical Analysis

Optimal Threshold	0.093
Final PnL	4.2872507479249915
Max Drawdown	1.5274443863556373

With optmization, we can gain the final PnL with about 400% of return with bearing 152% of the draw down.

6 Conclusion

This study demonstrated a promising approach to predict the trend of indices by employing momentum-based strategies on the representative stocks. We used a decade's worth of data to test the methodology, which achieved promising results in predicting the movement of the Taiwanese Electronics and Banking indices.

Our analysis showed that stocks contributing the most and the least to the movement of the indices have a substantial impact on future index returns. Our momentum-based predictor on these representative stocks managed to approximate the actual index trends effectively. Moreover, the prediction was improved by considering the hedge ratio, which provided a balance between the two studied indices.

However, it's worth noting that financial markets are highly complex and influenced by countless factors. Although the results from this study are promising, they do not guarantee future performance. Furthermore, the strategy should be backtested over different market periods and with various assets for a more robust evaluation.

Future work may explore different prediction models, employ machine learning algorithms, or consider more variables and factors. Another possible direction is to analyze other indices from different markets and sectors to test the versatility of this strategy.

To summarize, this study offered a fresh perspective on asset allocation and prediction in financial markets, and it opens new avenues for future exploration.