Survivorship-Bias Free Nifty 50 Historical Data

The strategy implemented in the provided code aims to remove survivorship bias from the NIFTY 50 historical data and then test a simple momentum strategy on the survivorship bias-free data. It also performs an optimization of the momentum strategy by varying the lag, period, and number of stocks in the portfolio. The performance of the strategy is evaluated and visualized.

Importing Required Libraries

The code starts by importing the necessary libraries, including pandas, numpy, matplotlib, os, yfinance, warnings, seaborn, and tqdm. These libraries are used for data manipulation, visualization, fetching stock data, handling warnings, and providing progress bars.

Fetching Tickers

The code fetches the current and historical ticker symbols of the NIFTY 50 stocks from the Wikipedia page using pandas' read_html function. It combines the current and historical tickers, removes duplicates, and adds the '.NS' suffix to each ticker.

Fetching Shifts Data

The code fetches the shift data from the Wikipedia page, which represents the changes in the NIFTY 50 index constituents over time. The data is stored in the shifts DataFrame.

Fetching Stock Data

Using the yf.download function from the yfinance library, the code fetches the historical adjusted close prices for all the ticker symbols. The data is stored in the data DataFrame.

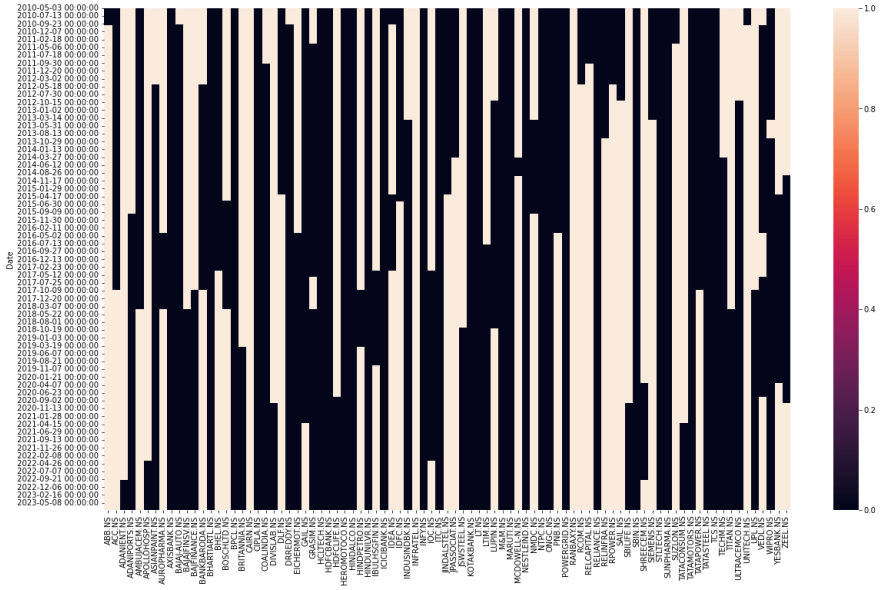
Handling Ticker Shifts

The code iterates over each historical ticker symbol and processes its corresponding shift data. It identifies the date ranges when the ticker symbol is added or removed from the NIFTY 50 index and updates the stock data accordingly.

```
SIEMENS
[':', Timestamp('2013-04-01 00:00:00')]


GAIL
[':', Timestamp('2021-03-31 00:00:00')]


RPOWER
[':', Timestamp('2012-04-27 00:00:00')]


IDEA
[':', Timestamp('2010-10-01 00:00:00'), Timestamp('2015-03-27 00:00:00'), ':', ':', Timestamp('2017-03-31 00:00:00')]


IOC
[Timestamp('2017-03-31 00:00:00'), ':', ':', Timestamp('2022-03-31 00:00:00')]
```

Visualizing Missing Data

The code visualizes the missing data in the data DataFrame using a heatmap generated by seaborn. This helps identify any missing values or gaps in the stock data.

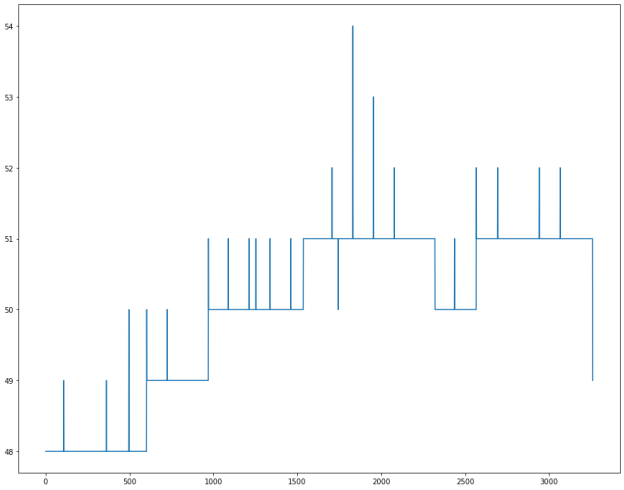


Calculating Number of Stocks

The code calculates the number of stocks available for each date by counting the non-null values in each row of the data DataFrame. The result is stored in the number_of_stocks list.

Plotting Number of Stocks Over Time

The code plots the number of stocks available over time using matplotlib. This graph helps visualize the changing number of stocks in the portfolio.



Preparing Combined DataFrame

The code creates a copy of the data DataFrame called combined_df for further analysis. It also initializes an empty list called stocks_list to store the top-performing stocks for each month.

Selecting Top Stocks for Each Month

Using the resampled combined_df DataFrame, the code selects the top-performing stocks for each month based on the previous returns. The number of stocks selected is determined by the parameter stocks_no. These stocks are added to the stocks_list.

Calculating Monthly Returns

The code calculates the monthly returns for each stock in the returns DataFrame by dividing the adjusted close prices by their previous month's prices.

Evaluating Performance

The code evaluates the performance of the portfolio by calculating the average monthly returns for each month. It stores the results in the perform DataFrame.

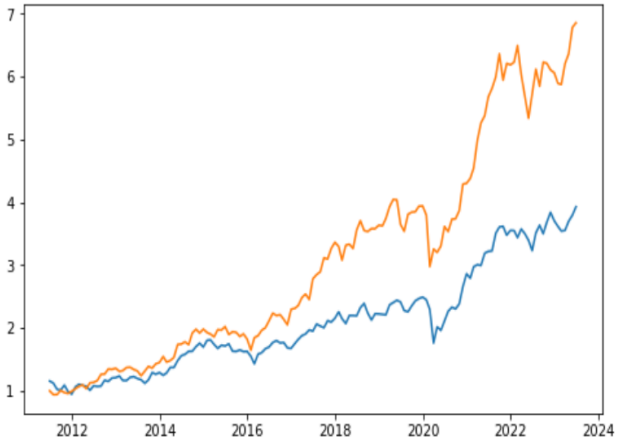
Date	Adj Close	Nifty	Date	Monthly_returns	Strategy
2011-06-30	1.031847	1.156768	2011-06-30	1.000303	1.000303
2011-07-31	0.970712	1.122889	2011-07-31	0.939211	0.939495
2011-08-31	0.912258	1.024365	2011-08-31	1.002103	0.941471
2011-09-30	0.988452	1.012536	2011-09-30	1.069082	1.006509
2011-10-31	1.077550	1.091058	2011-10-31	0.966364	0.972654
...	...	...	...	...	...
2023-02-28	0.979719	3.544402	2023-02-28	0.997254	5.879403
2023-03-31	1.003225	3.555832	2023-03-31	1.056422	6.211130
2023-04-30	1.040626	3.700290	2023-04-30	1.024817	6.365270
2023-05-31	1.025984	3.796438	2023-05-31	1.066256	6.787009
2023-06-30	1.035321	3.930532	2023-06-30	1.010708	6.859687

145 rows × 5 columns

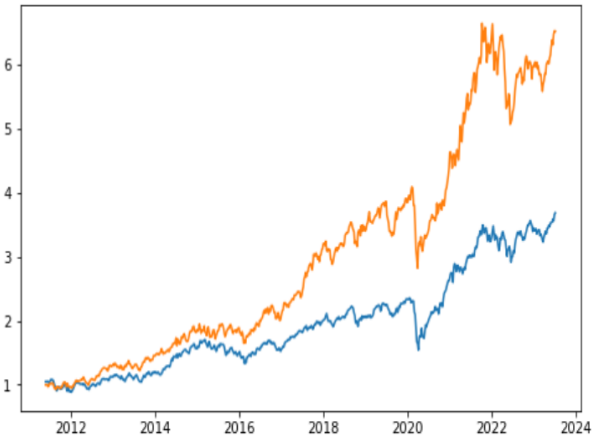
Comparing Performance with NIFTY 50 Index

The code fetches the NIFTY 50 index data from Yahoo Finance using the yf.download function. It resamples the data to monthly frequency, calculates the monthly returns, and compares the performance of the strategy with the NIFTY 50 index.

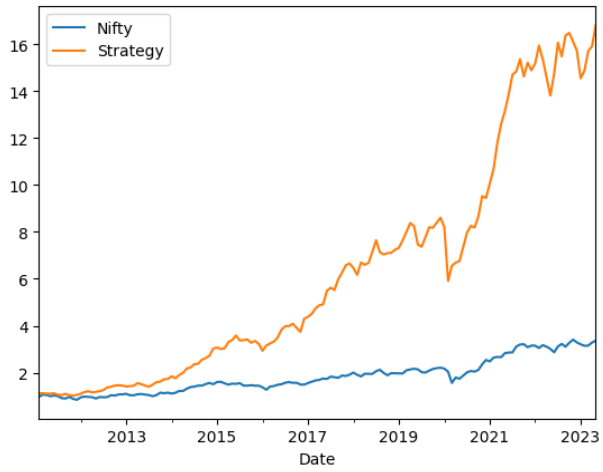
Monthly(Without Survivorship Bias)



Weekly(Without Survivorship Bias)

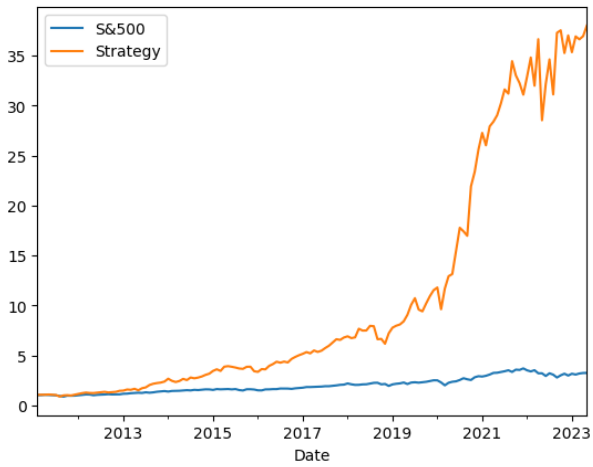


Monthly(With Survivorship Bias)

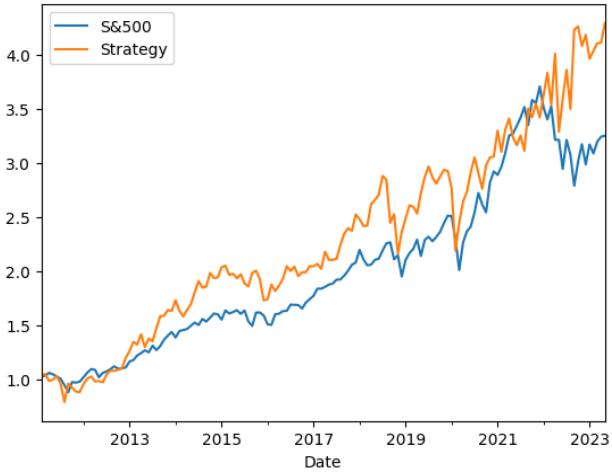


S&P 500 Backtest

With Survivorship Bias



Without Survivorship Bias



Visualizing Performance

The code plots the performance of the strategy and the NIFTY 50 index using matplotlib. It generates two graphs: one with linear scale and another with a logarithmic scale.

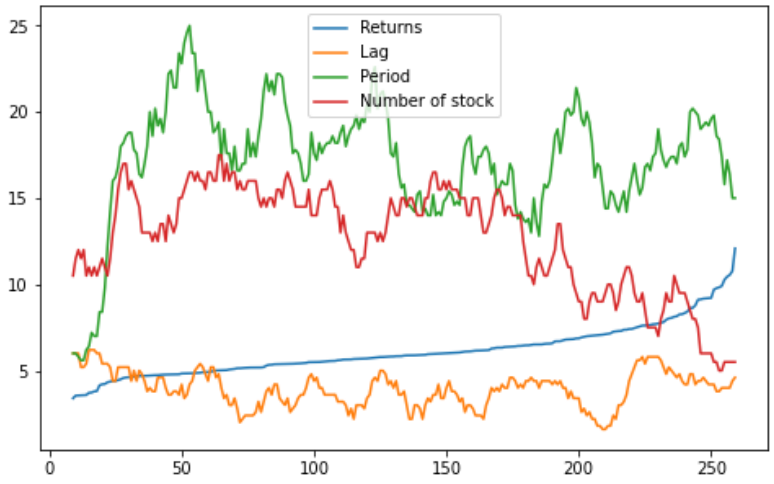
Optimization of Momentum Strategy

The code defines a function mom that implements the momentum strategy with varying lag, period, and number of stocks. It iterates through different combinations of these parameters using the product function from the itertools module. For each combination, the strategy's performance is calculated and stored in the opt DataFrame.

```
(5.135967688576338, 2, 7, 10)
(5.81858073200074, 2, 7, 15)
(6.157828178626516, 2, 7, 20)
(7.278934921713355, 2, 9, 5)
(6.56537687218268, 2, 9, 10)
(5.9120338589102905, 2, 9, 15)
(6.042264021315903, 2, 9, 20)
(6.496303903956242, 2, 11, 5)
(6.964547869731695, 2, 11, 10)
(6.032375885879424, 2, 11, 15)
(5.882689368745868, 2, 11, 20)
(4.847809784861032, 2, 13, 5)
(5.498522818021085, 2, 13, 10)
(7.302298822591083, 2, 13, 15)
(5.801450999925752, 2, 13, 20)
(8.782232521937946, 2, 15, 5)
(6.5478311576253665, 2, 15, 10)
```

Plotting Optimization Results

The code plots the optimization results by rolling averaging the parameters and plotting the performance over the parameter combinations.



Summary and Conclusion

The provided code removes survivorship bias from the NIFTY 50 historical data, tests a momentum strategy on the cleaned data, and performs an optimization of the strategy. The performance of the strategy is compared with the NIFTY 50 index, and the optimization results are visualized. This helps evaluate the performance and potential improvements of the momentum strategy.