Final Project: Hedge Fund Replication

Balboni Matteo, De Rosso Marco, Hanna Kamil, Rigamonti Matteo, Zhou Fude

Background

Traditionally, hedge funds (HF) have always been very secretive about their trading strategies. Especially for quant funds, the mysterious algorithms and electronic routines are aimed at generating the highest returns regardless of market conditions.

These are usually obtained combining different asset classes like equities, bonds, derivatives and taking long-short positions in different markets.

Moreover , hedge funds lack transparency and put investors into weak positions by charging high fees with high lock-up periods.

For these reasons, trying to mimic what the highest performing hedge funds are doing is of great interest to many investors.

Model objective

Given some artificial HF returns' trajectories and some available assets to trade, in this case, futures, the task will involve developing algos for combining the assets as to best replicate (or at least best approximate) the fund dynamics.

In other words, the model will provide the *Betas* (the weights) of the portfolio assets to associate to potentially each of the futures.

For the moment, the approach will try to optimally trade "near" the HF dynamics, considering* some potential regime-switching abilities (i.e. avoid replicating when the HF portfolio is plummeting).

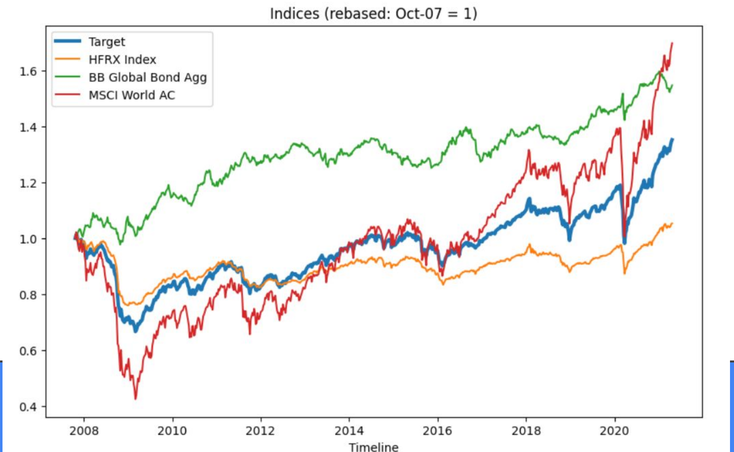
*Except for cropped OLS

Our Solution

The task of investment replica has a potential disadvantage. If we mimic the wrong funds or if the market crashes, our portfolio may suffer severe losses.
This is why ideally we want our models to learn only the behaviours that correspond to positive returns. While this idea is already present and is easily translatable in the domain of Reinforcement Learning, we propose a way to translate it and make it understandable by simpler algorithms.
In detail, we take the dataset of the returns and crop to zero the negative returns. This should force the models to learn positive returns while they have more freedom instead of approximating negative ones.
To push this idea to the limit we are going to build a third dataset made of the absolute values of the returns, and we are going to see if this improves the performances.

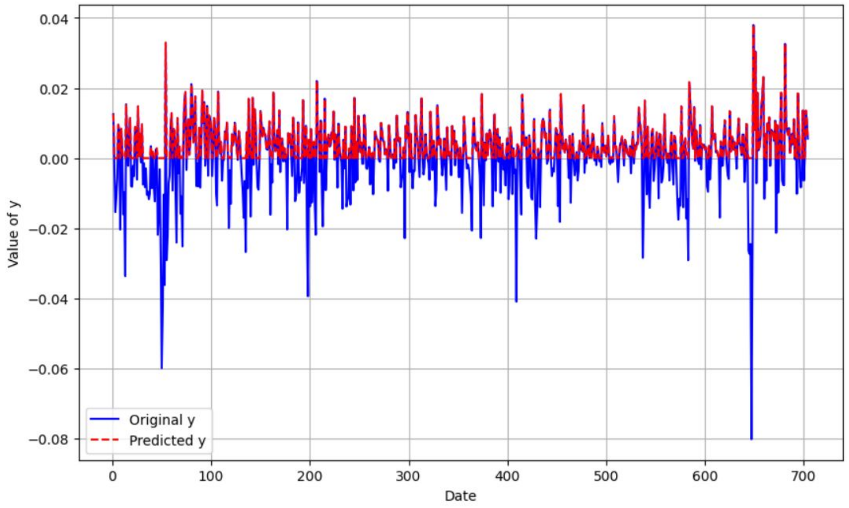
Data

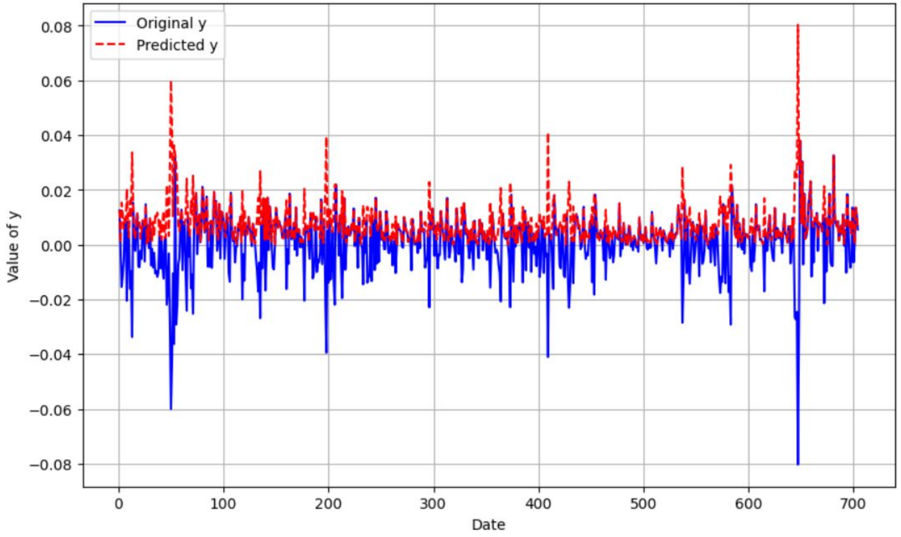
- Data contains time-series of hedge funds (we selected one) as well as time-series of futures prices.
- Missing values are removed from time-series
- Backtesting are done Out-of-Time (random train test split would imply total independence of asset returns → EMH→no systematic profit exist)
- Columns for HF dynamics and Futures prices are rescaled using percentage returns from previous day.
- Y feature represent HF path



Our solution

- We show here the datasets that we have built (In red)





Absolute values

Cropped values

Algorithms

- We implemented several regression models: OLS, Lasso, Ridge and Elastic Net
- We delved into state space models with ARIMA, SARIMAX and Kalman Filters
- Finally we explored two solutions in the domain of Reinforcement Learning: a Random policy optimization and a Q-Learning approach.

Backtests

- In the next slides we will show the results of performing a training on the elements from index 200 to index 356 (rolling window) and then testing the models until the end of the serie.
- In the colab notebook we implemented tests every 100 samples, with both a fixed window and a rolling window of 156. The ones that we chose are the most representative of the behaviour of the model.
- We also computed several metrics for each time serie, such as volatility, upper and lower partial moments, value at risk, etc.
- We noticed that our modified cropped dataset effectively reduces volatility and lower partial moments.

Metrics

- Now we give a brief explanation of the metrics that we used
- Upper and Lower partial moments: expected value of positive (negative) deviations from a target return. This metric helps in understanding the potential for gains (losses) above a certain threshold.
- Semi-Deviation measures the dispersion of returns that fall below the mean or a target return. It is similar to standard deviation but focuses only on downside risk, helping investors understand the variability of negative returns.
- VaR estimates the maximum loss that could occur with a certain confidence level (e.g., 95%) over a specific period.
- CVaR, also known as Expected Shortfall, measures the average loss in the worst-case scenarios beyond the VaR threshold.

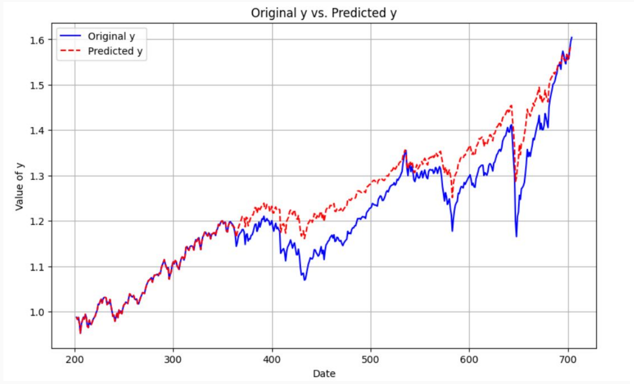
Metrics

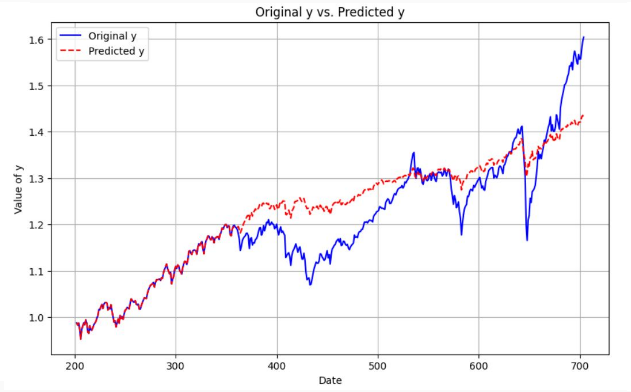
- Downside Deviation is similar to Semi-Deviation, focusing on negative returns. It helps investors assess the risk of losses by measuring the volatility of returns that fall below a specified threshold.
- LPSD measures the standard deviation of negative returns relative to a target return. It quantifies the risk of underperformance, helping investors understand the variability of returns below the target.
- The Omega Ratio compares the probability of gains to the probability of losses, considering different levels of target returns.
- The Sortino Ratio measures risk-adjusted returns, focusing only on downside risk. It is calculated by dividing the excess return over the risk-free rate by the Downside Deviation.
- CDaR measures the potential worst-case drawdown (decline from peak to trough) at a specific confidence level.

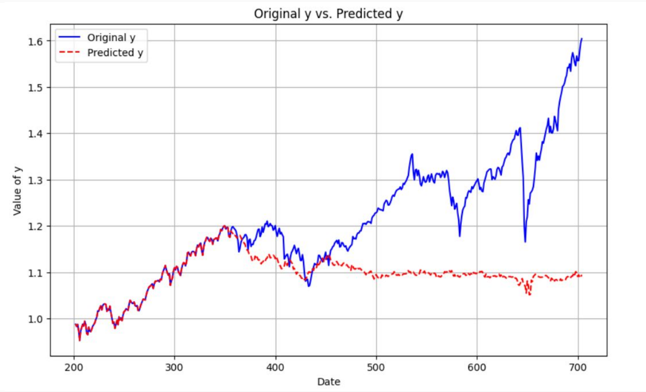
Backtesting

- Those metrics are computed for each test that we performed. For brevity, we leave them on the notebook. However it can be checked that, in the case of regressions, the lower partial moments are reduced significantly, along with the semi-deviation and other corresponding metrics.
- N.B. metrics that require the knowledge of the risk-free rate are computed with risk-free rate equal to zero. In a real world scenario this would not be the case, and they would be computed with the actual risk free rate.

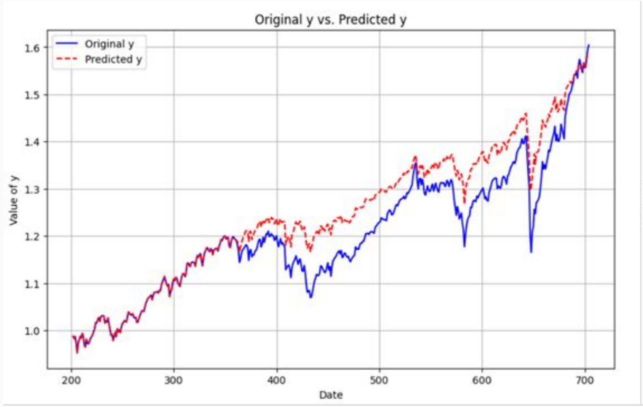
OLS Performance

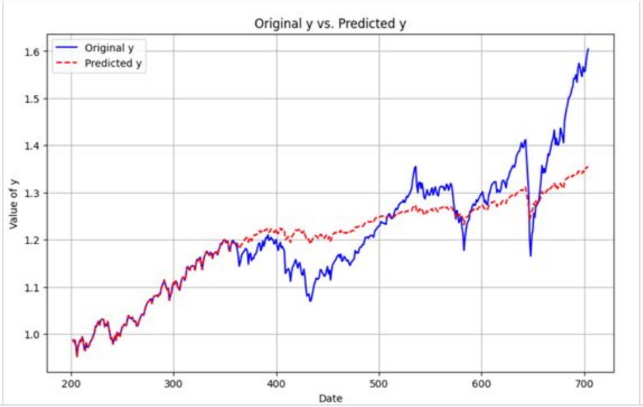


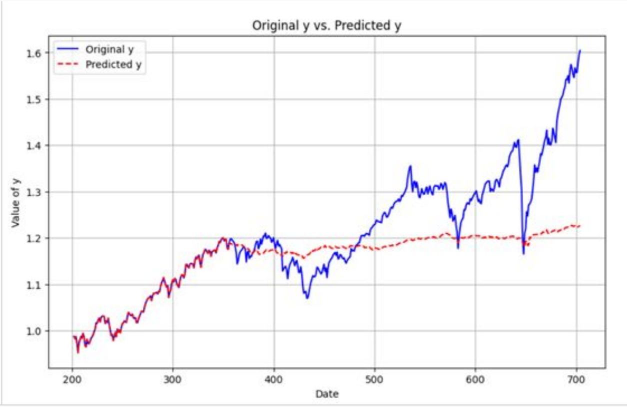




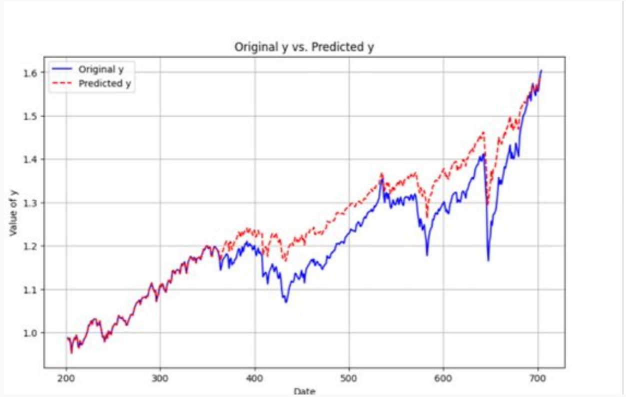
Lasso Performance

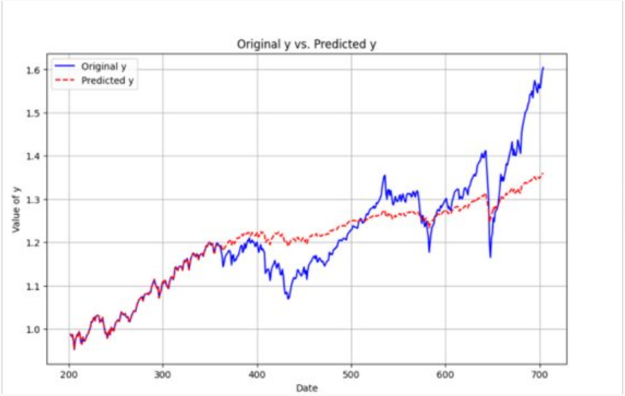


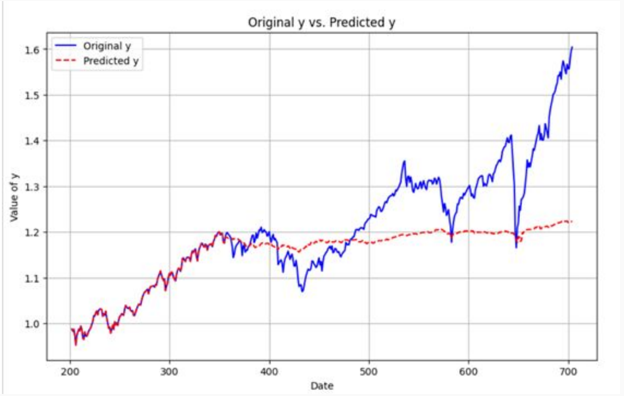




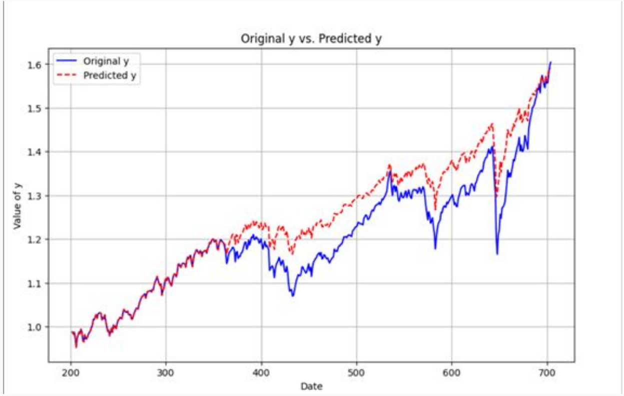
Ridge Performance

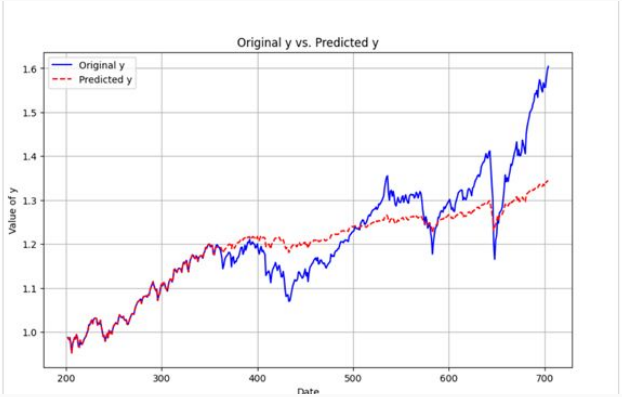


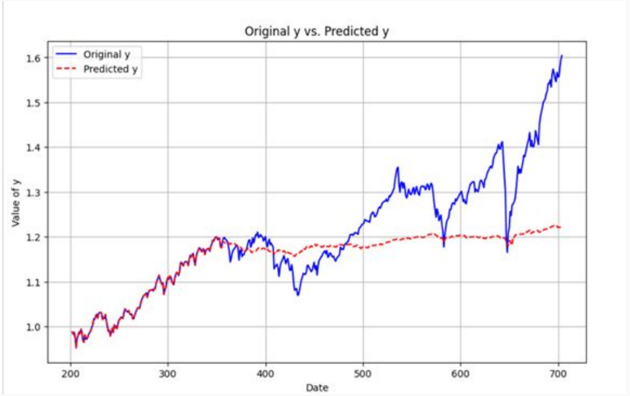




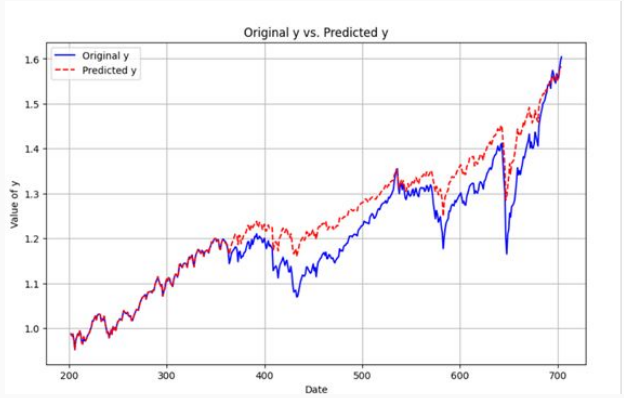
Elastic Net


Original y vs. Predicted y
Original y
Predicted y
Value of y
Date
1.6
1.5
1.4
1.3
1.2
1.1
1.0
200
300
400
500
600
700

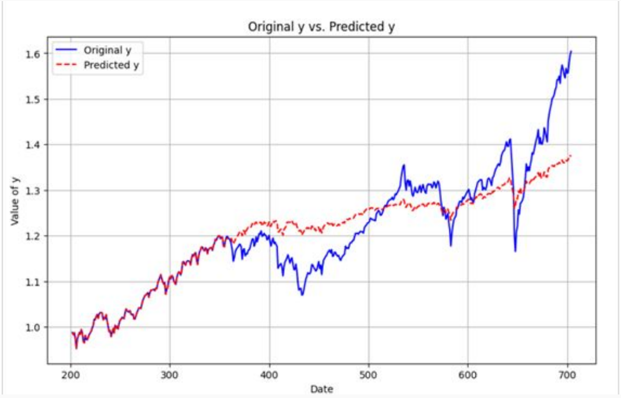

Original y vs. Predicted y
Original y
Predicted y
Value of y
Date
1.6
1.5
1.4
1.3
1.2
1.1
1.0
200
300
400
500
600
700


Original y vs. Predicted y
Original y
Predicted y
Value of y
Date
1.6
1.5
1.4
1.3
1.2
1.1
1.0
200
300
400
500
600
700

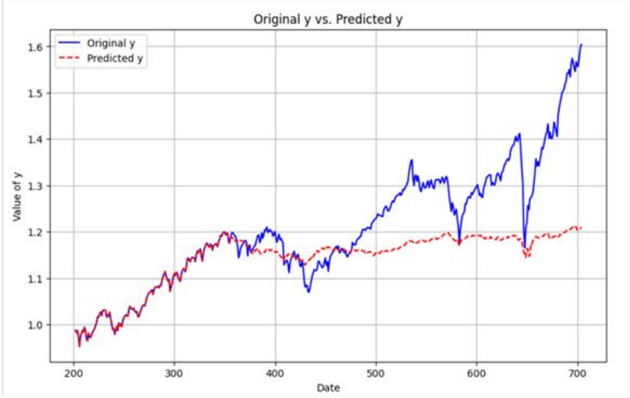
ARIMA



Original y vs. Predicted y
Original y
Predicted y
Value of y
Date

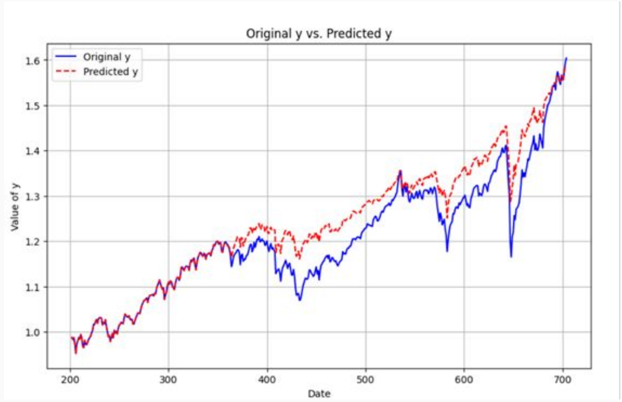


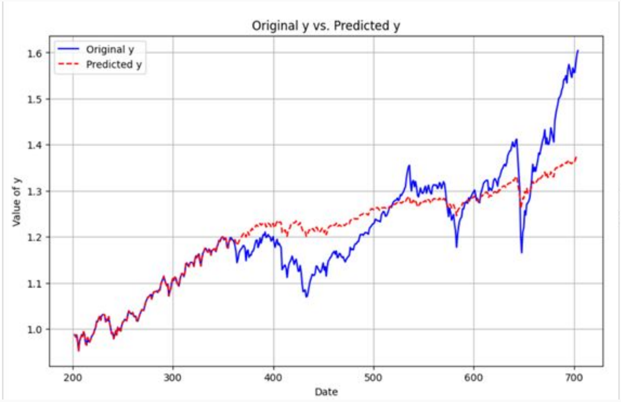
Original y vs. Predicted y
Original y
Predicted y
Value of y
Date

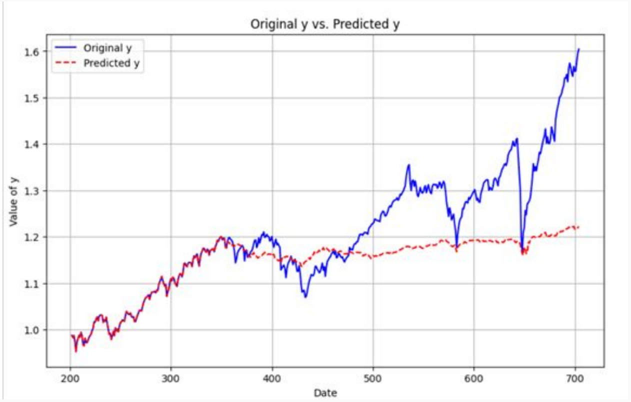


Original y vs. Predicted y
Original y
Predicted y
Value of y
Date

SARIMAX


Original y vs. Predicted y
Original y
Predicted y
Value of y
Date
1.0
1.1
1.2
1.3
1.4
1.5
1.6
200
300
400
500
600
700

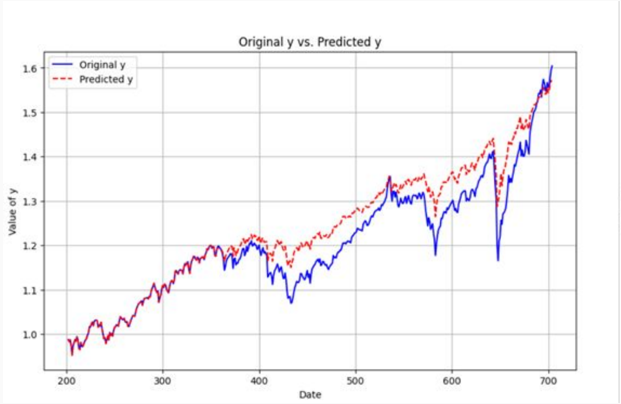

Original y vs. Predicted y
Original y
Predicted y
Value of y
Date
1.0
1.1
1.2
1.3
1.4
1.5
1.6
200
300
400
500
600
700

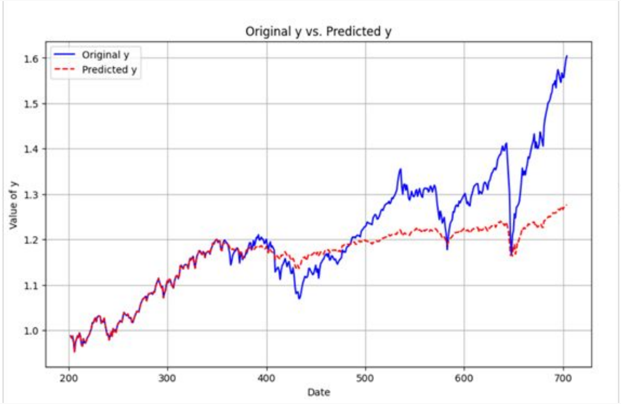

Original y vs. Predicted y
Original y
Predicted y
Value of y
Date
1.0
1.1
1.2
1.3
1.4
1.5
1.6
200
300
400
500
600
700

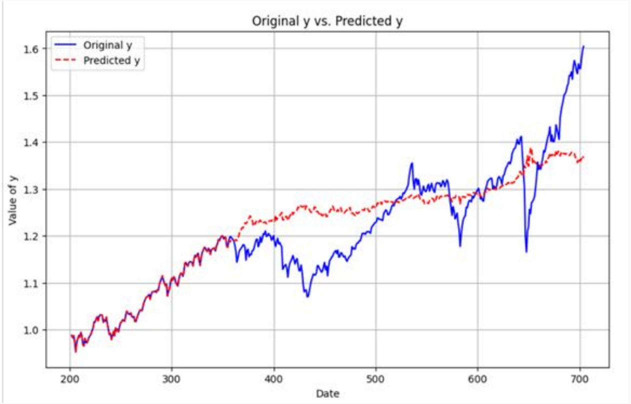
Kalman Filter

- Two version of Kalman filters were implemented in our project: one built from scratch and another using the prebuilt filter from the FilterPy python library.
- Both filters were integrated with other algorithms such as regression and backtesting.
- All the parameters for both filters, were derived from the previous implementations (regressions, for instance ridge regression and other…), since initializing things from zero; wouldn’t be so clever. For example, some of whom we can mention; the state vector for instance was initialized using beta weights obtained from a regression problem, the P error covariance matrix was initialized from the standard errors of the beta weights.

Kalman Filter







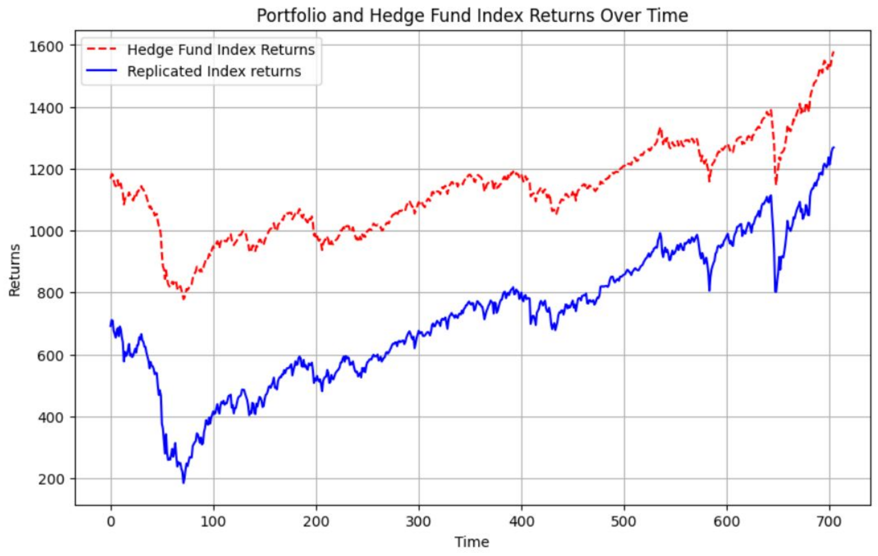
Comparison Kalman Filters

Our “built from scratch” Kalman filter has performed better than the prebuilt Kalman filter in the filterpy python library. This may due to many reasons.

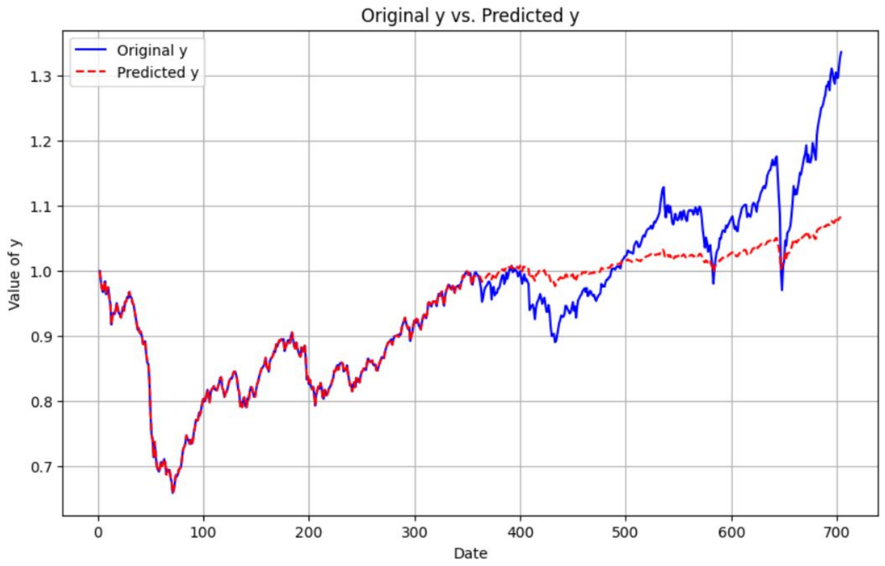
- The prebuilt Kalman filter (from the filterpy library) includes numerous parameters. Some of which are not applicable to our problem, therefore they are set to their default values which might impact the filter’s performance.
- The prebuilt filter is designed/optimized for different types of applications, there for it is not perfectly aligned with our needs.
- We encountered a problem in computation due to the covariance matrix of the futures returns R having high coefficients, which let for it to be a singular matrix, causing an issue with an inverse matrix computation. We solved this issue by addressing the covariance shrinkage problem by Ledoit Wolf (2004).

Comparison Kalman Filters

Example (Ridge Regression) : (trained from observations [1~356], prediction after observation 356

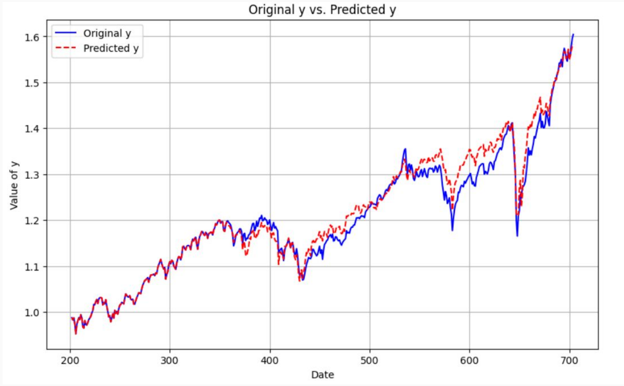


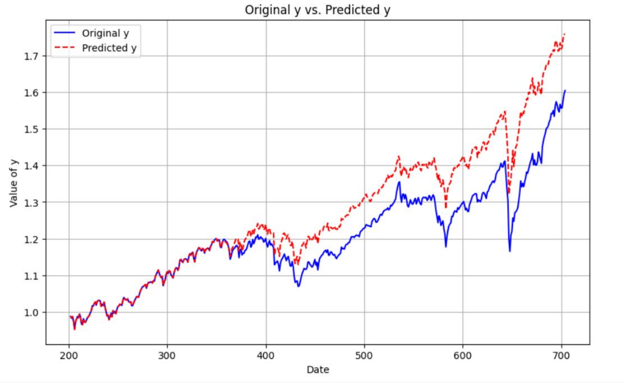
Prebuilt (From filterpy)

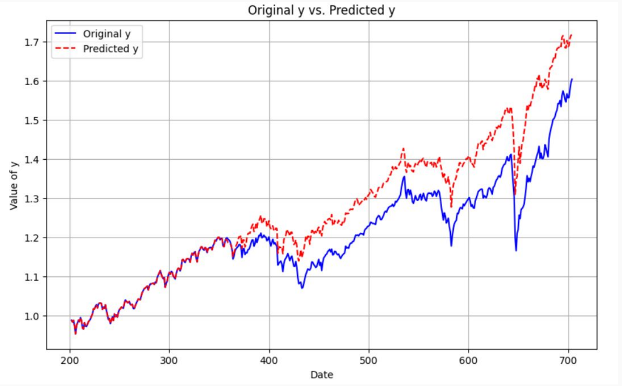


Built from scratch

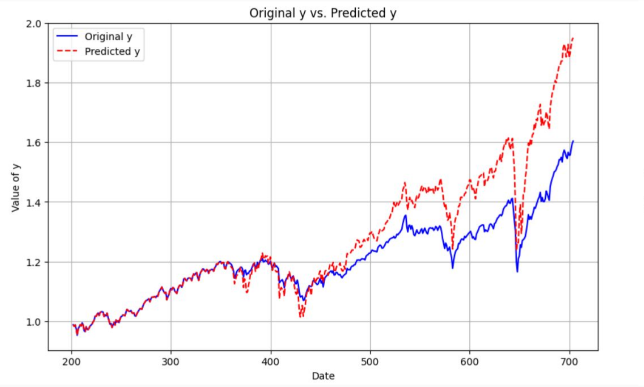
Reinforcement Learning - Random Agent

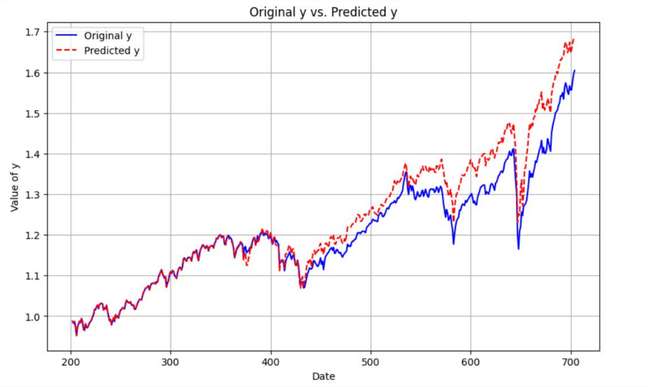


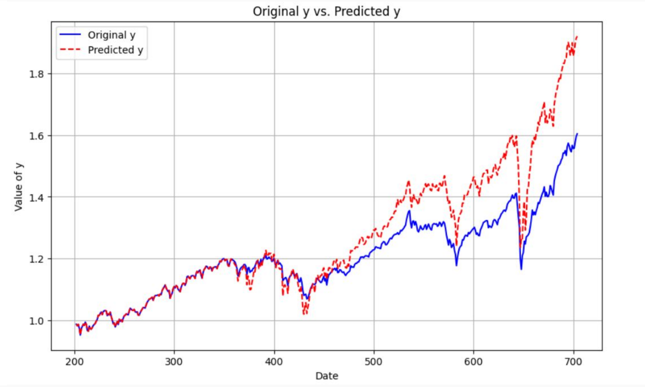




Reinforcement Learning - Q-Learning







Conclusions

From the exploration of all these models we found that:

- With the original dataset we can achieve good replicas of the target index
- With the cropped dataset we can obtain low variance replicas of the target index (significant reduction of lower partial moments). We noticed that whenever the training happened on a window where the index was growing the replica tended to maintain the growth trend of that window.
- With the absolute values dataset we obtained almost flat or bad performing replicas

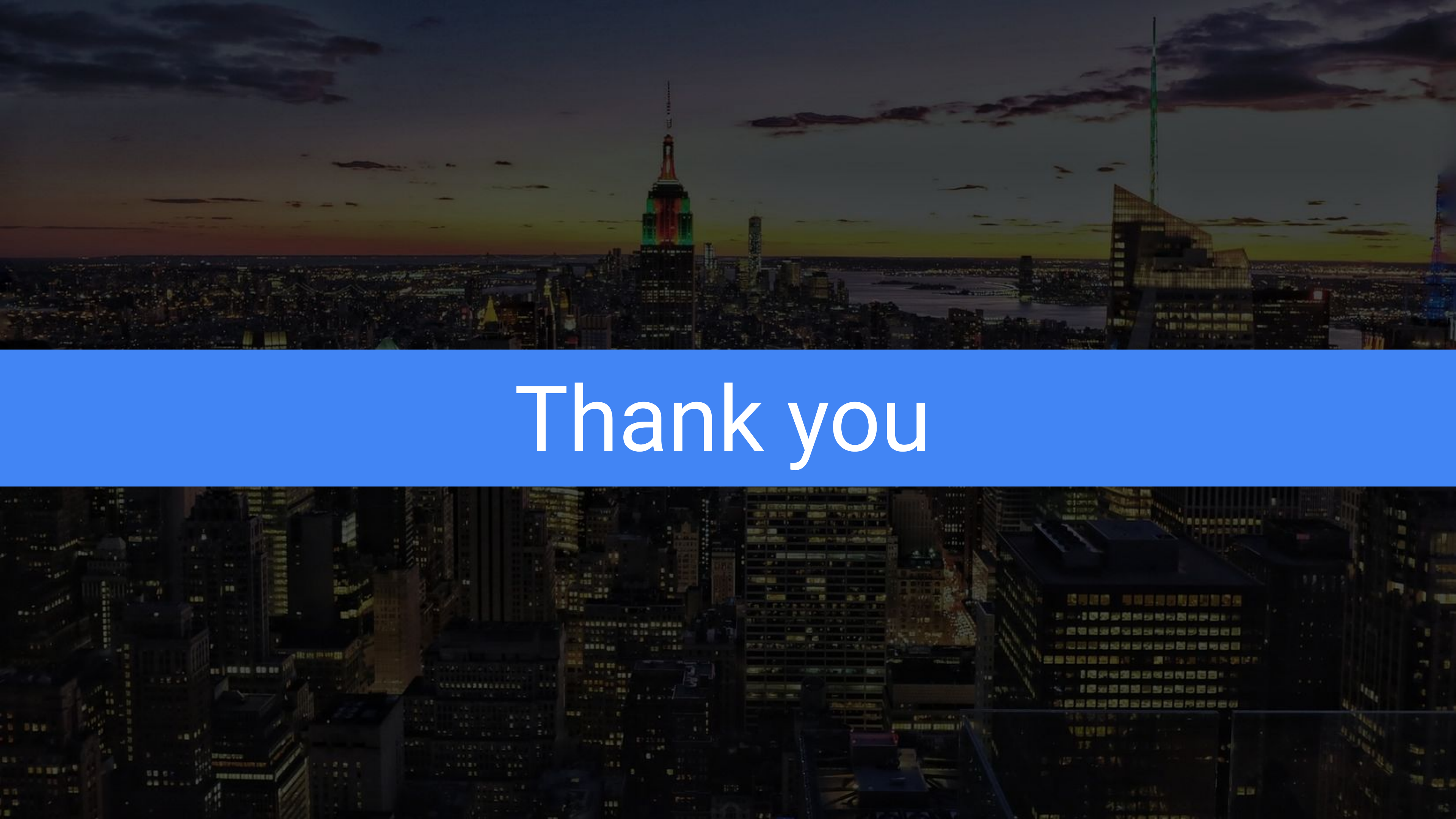
The only case in which the absolute values dataset performed better than the original and cropped values was with the Kalman Filter.

Conclusions

The algorithms that achieved the best results in terms of replicating the target index were the Reinforcement Learning algorithms, that showed a tendency to outperform the index by showing generally higher returns for a more consistent period of time. The negative side effect is that they increase the variance of the replicated index. When utilizing the cropped dataset, the variance of the replicated serie becomes comparable with the one of the target index and keeps the tendency to have higher returns. When utilizing the absolute values dataset the variance is increased.

References

- Ledoit, O., & Wolf, M. (2004). A Well-Conditioned Estimator for Large-Dimensional Covariance Matrices. *Journal of Multivariate Analysis*, 88(2), 365-411.
- Wei, Wang Chun. "Modelling and Replicating Hedge Fund Returns." SSRN, March 1, 2010.
- Roncalli, Thierry, and Weisang, Guillaume. "Tracking Problems, Hedge Fund Replication, and Alternative Beta." SSRN, December 24, 2008.



Thank you