**WIDS**

**PAIRS TRADING USING KALMAN FILTER**

**WEEK-2**

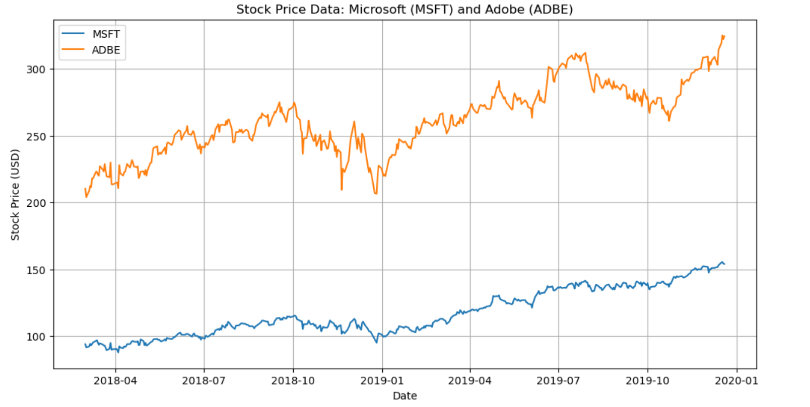
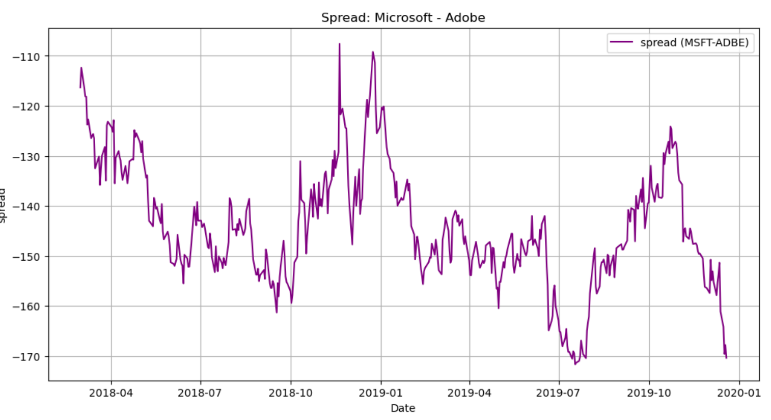
So, kicking this week off we firstly read about the Kalman Filter Algorithm, a recursive algorithm which estimates the state of a system taking into account noise, both in the system as well as in the observation. A lot of mathematics surrounds this algorithm, but long story short, with each new input entry to the system, it refines itself, and based on the previous results, estimates the next state.

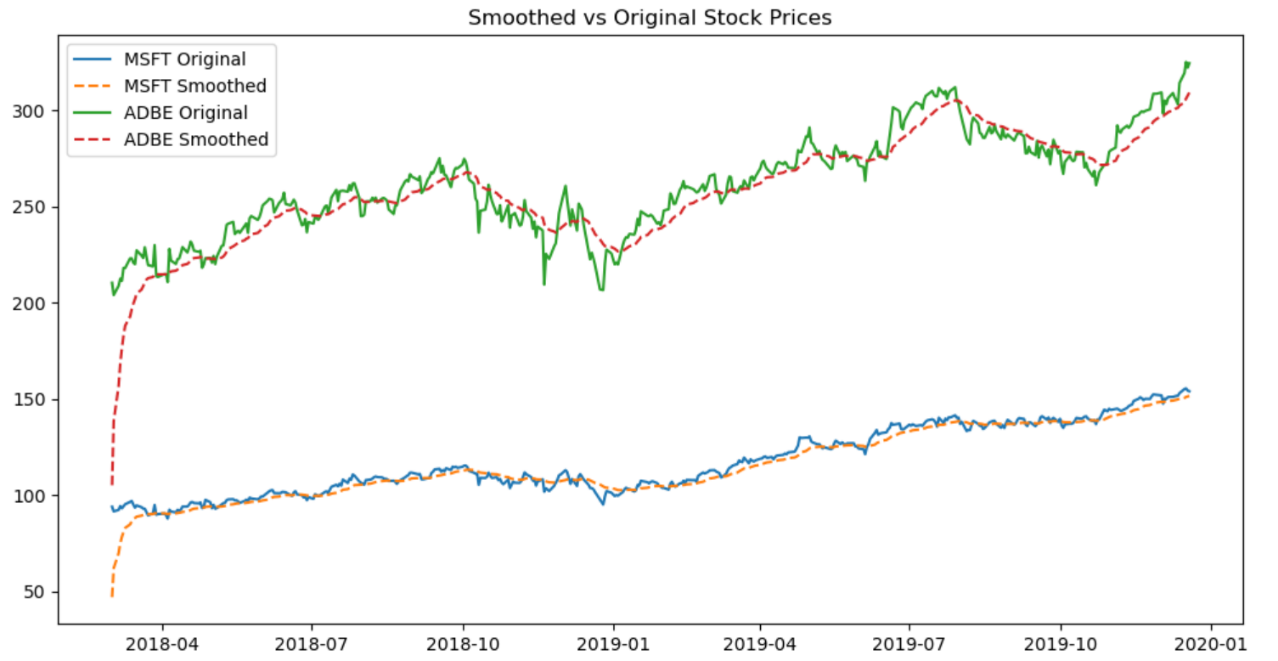
Alongside, there was introduction to new terms, which I will brief while explaining what I did in my assignment. So, the assignment demands from us, implementing a pairs trading strategy involving Kalman filter, on the MSFT and ADBE stocks. These are quite evidently cointegrated as proved earlier in the first assignment, but to complement the ADF test we can also use another ratio called the Hurst Ratio, which if is below 0.5 it implies that the respective time-series exibits mean reversion, and hence holds some predictability.

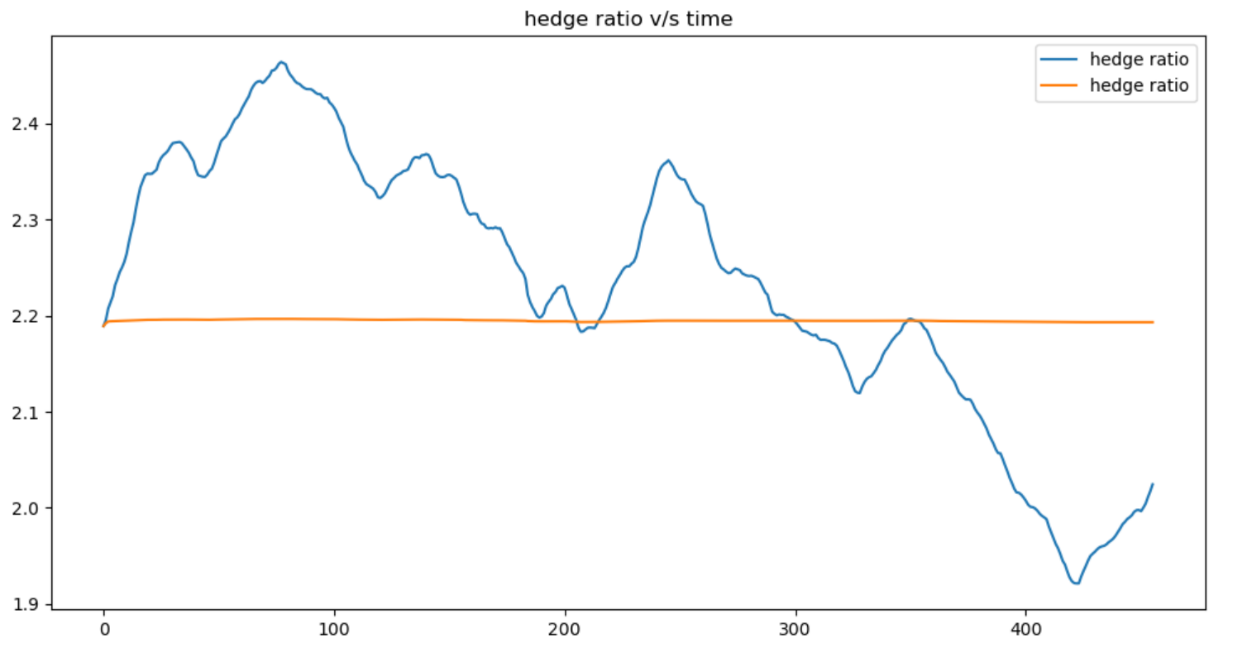
Also, this time we don’t work upon the price ratio of the two price-series of MSFT and ADBE, we work upon their spread. So, spread is basically defined as the price difference between a pair of assets. But, we use a ratio called hedge ratio which basically represents proptionality or the number of stocks of one asset to be held against another to minimize risk. The formula being:



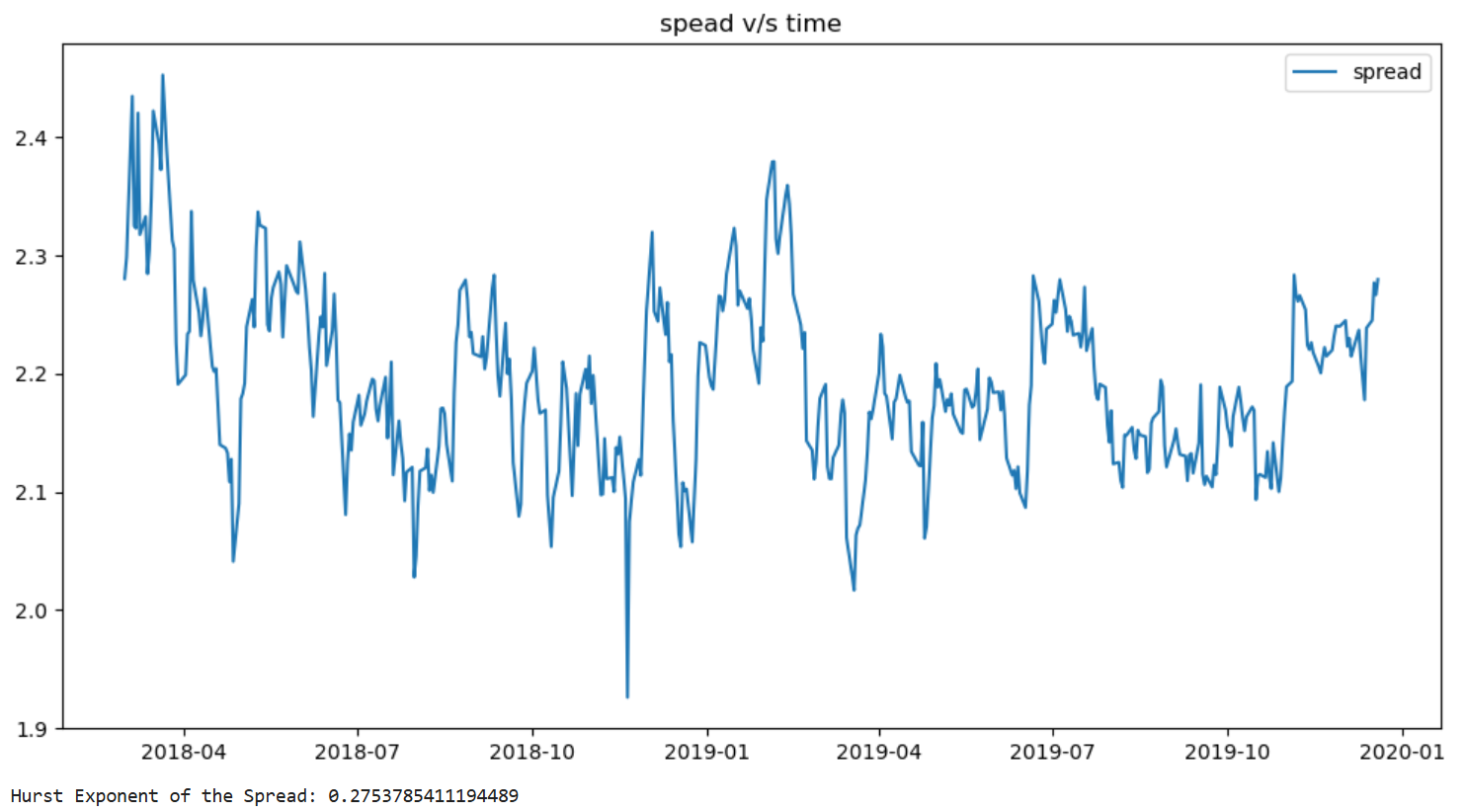
Beta being the hedge ratio, and y\_t and x\_t the respective price series for MSFT and ADBE.

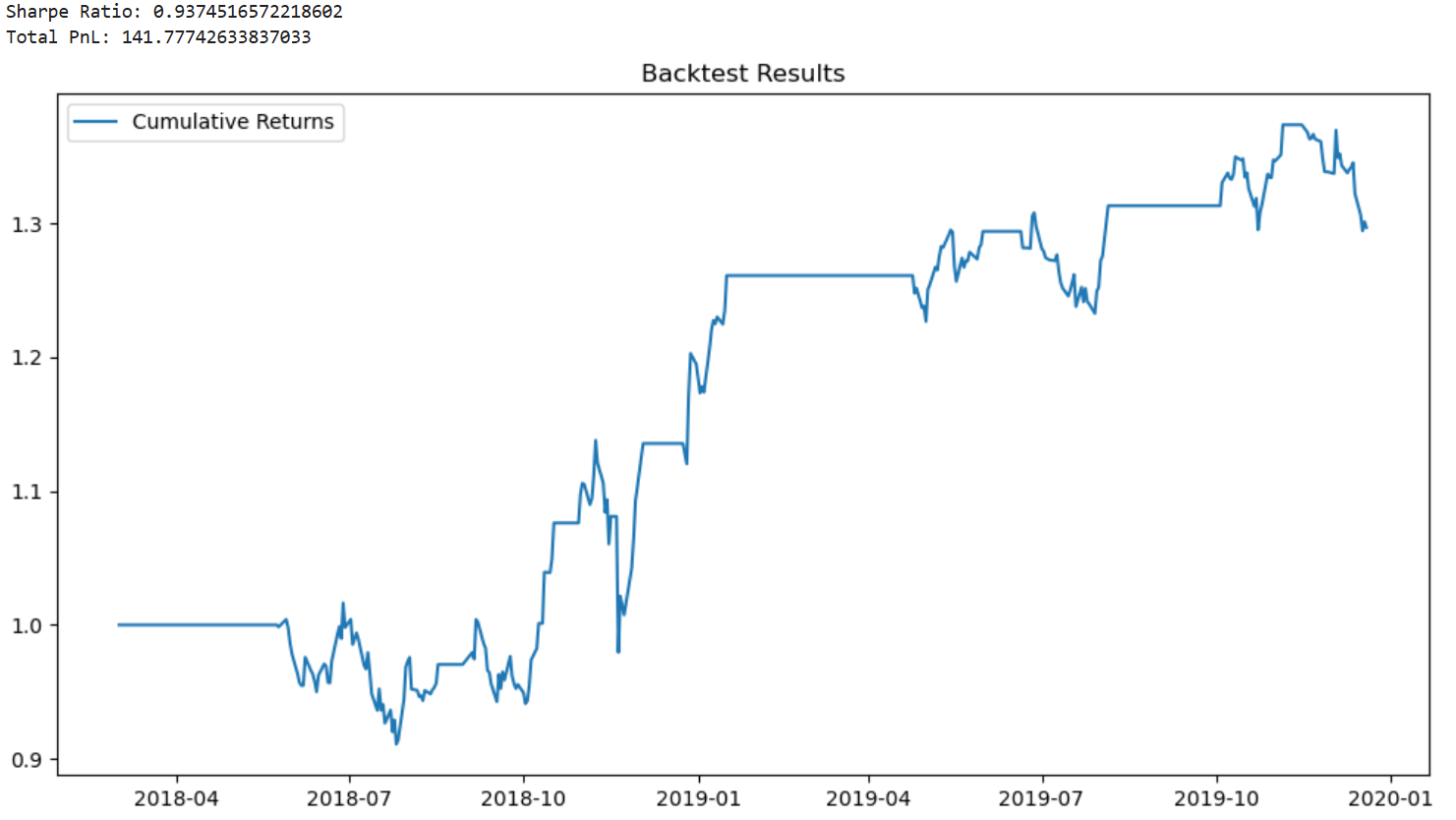
So, here is what the spread looked, hedging it with unity.

Now, we apply Kalman Filter to the two price series. We use the inbuilt pykalman library, and a function called KalmanFilterAverage which basically smoothens out the two price time series, so that they can be integrated together using the KalmanFilterRegression function. We ultimately aim to mak epredcitions using Kalman Filter and then use the newly obtained input to estimate values of alpha and beta in the above mentioned equation. More the number of inputs, better the prediction of hedge ratio beta and calculation of spread. One thing to keep in my mind, is that everything being done is to ultimately end up with a time-series (using spread) which shows predictable mean reversion, to make a good strategy.

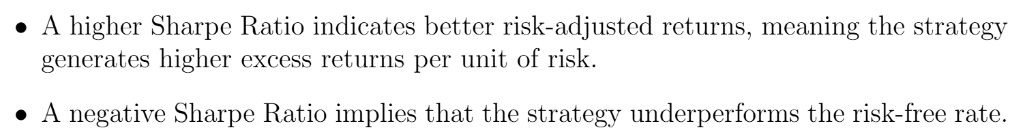
Now, using these smoothened series, we calculate the hedge ratios v/s time, the graph of which looks like:

Kind of means that we need to buy twice as many MSFT stocks compared to ADBE. This is what the spread looks like with the changing (improving) value of hedge ratio (over time):



The resulting hurst exponent is 0.27<0.5 and hence we proved that the spread-time series holds stationarity. So, till now what we have achieved is the value of hedge ratio, ‘beta’ which we calculated dynamically using KalmanFilter. Next, we do what we did previously in assignment 1, use this ‘mean-reverting’ time series and apply the pairs trading strategy using z-scores. If the z-score<-2, it means we need to go ‘Long’ on our pair (buy the first and sell the second stock) and similarly, go short if z-score>2. Applying this strategy gives us the following total return/cumulative return graph:

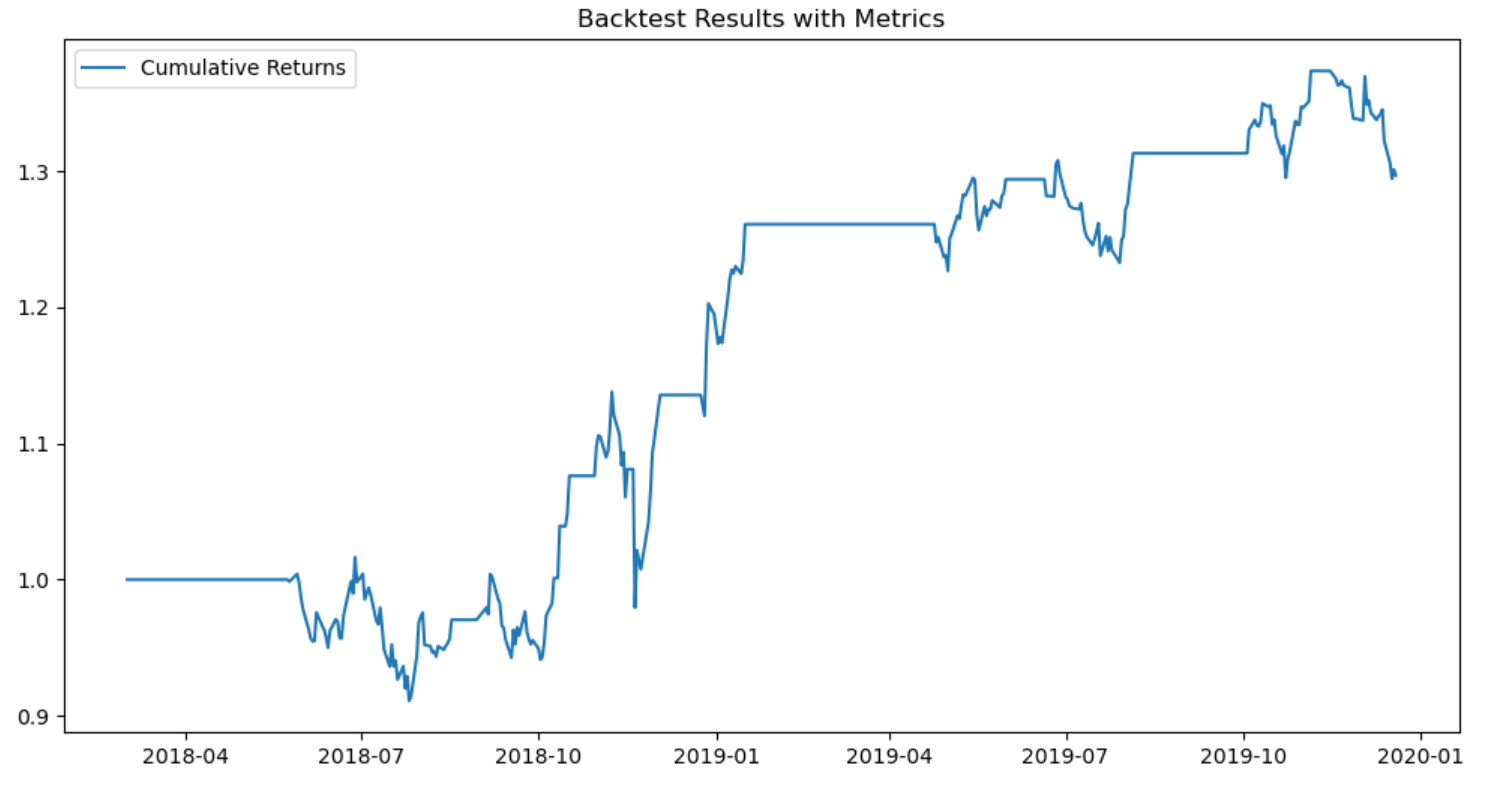
Another metric we mentioned is the ‘sharpe ratio’.

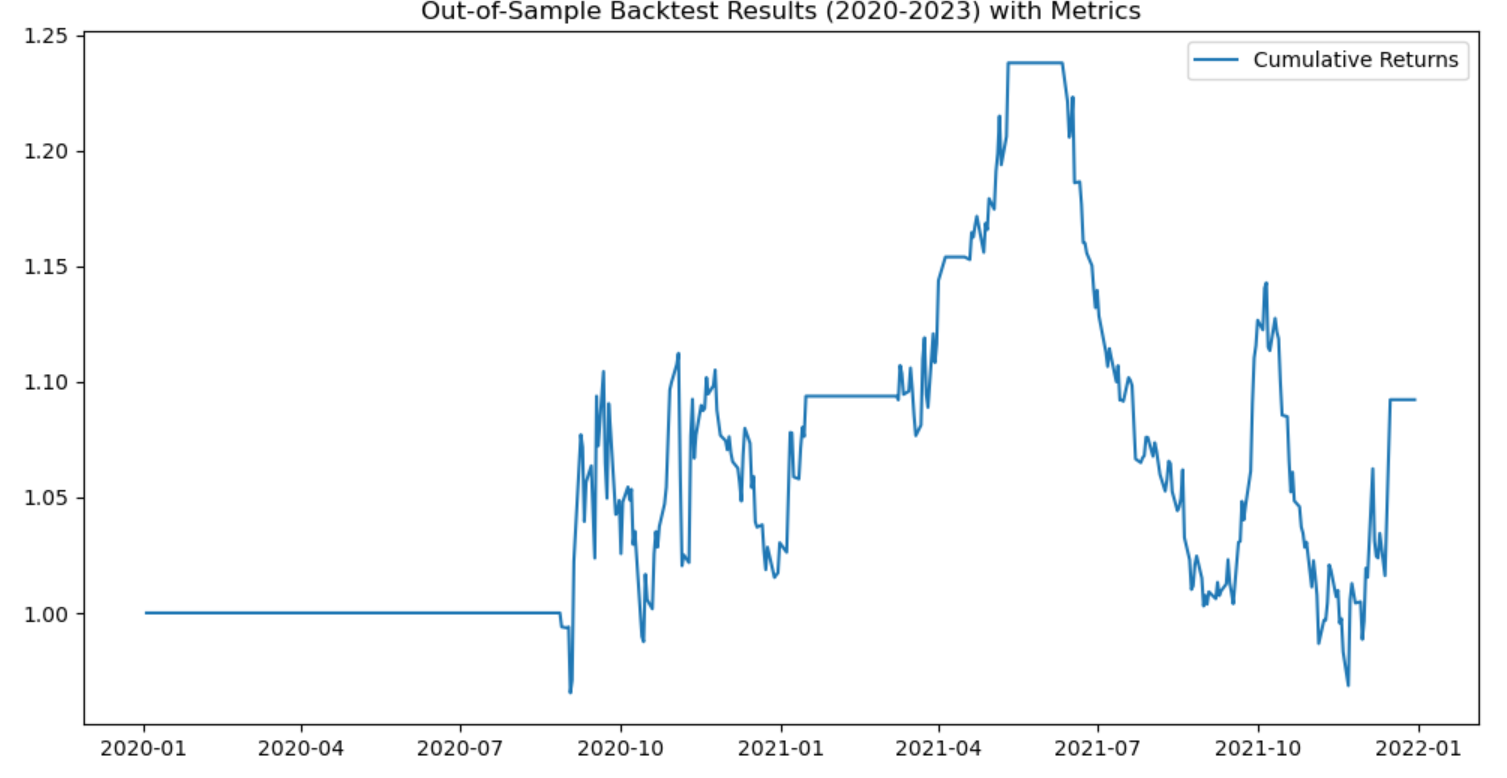


We get a sharpe ratio of 0.93, not ideal (ideal would be >1) but since it is close to 1 (and not negative fortunately) we can say the strategy applied is not awful. What we have done here is “in-sample backtesting”, which basically means that we have used the available stock data “till now” to get the best assumption of our alpha (neglected due to good co-integration) and beta (hedge ratio) parameters. It is when we use these parameters, on unseen future data, that we call “on-sample backtesting”.

Insample backtesting:

We get a CAGR of almost 16% (very decent, on par with small cap stock performace, but yes can be better) and a calmar ratio of 1.1 which basically indicates that the strategy has a reasonably good balance between its returns and the risk associated with drawdowns. We also get a profit of 142$ using this strategy.

Out-of-sample testing:



What we did here, is train the data from data of 2018 march to 2019 December. This data is then applied to data from 2020 Jan to Dec 2021 (not suggested as the training data should be more to give better sharpe ratios and reliability). Still we get a CAGR of around 4% (at par with government bonds :( but yes, more or less, what we needed to implement was successfully implemented.