challenge: Quant (congrats congrats money money $ $ $ $)

Completed:

- task 1) data cleaning , feature extraction
- task 2) model making
- task 3) building a back testing engine
- task 4) trying to find cointegrated movement

Didnt complete:

- bonus) mapping the given dataset to real world assets

Summary

- wow, what an adventure this was!
- This document sumarises all the research , findings , results from all the experiments I ran for completing the task
- **1) Data cleaning, Feature extraction**
  - implemented multiple filters, data cleaning pipeline
  - dove really deep into traditional stock indicators, and quant indicators for ml modelling for financial data feature extraction
- **2) Models , Models, Models**
  - Tested traditional methods like indicators, candle stick patterns, heiken ashi, market structure indicators
  - Tested out many different ml frameworks for building various models like
    - Linear regression
    - Baysian models
    - Arima models
    - Decision trees
    - Random forests
    - Boosted random forests

        - Bagged decision trees
        - Cnns
        - Mamba models
        - Hierarchial graph nueral networks.
    - Tested feature decompostion and how it effects ml models
    - Tested 101 alphas just to see how they perform
- **3) Backtesting**
    - Built a multi agent weights based backtesting engine
    - Implemented different portifolio methods like 1/n , modern portifolio theory , neo modern portifolio theory
    - Implemented dynamic visualization using bokeh js, stop loss ability, sharpe , markdown indicator
- **4) Cointegrated movement**
    - used heiken ashi , kmeans for cointegration finding
    - ran tradition clustering algorithms like kmeans, gmm , dbscan
    - used monthly percentage gain with kmeans
    - Tried out Engle - Granger , Augmented Dickey Fuller

---

1) Data cleaning & feature extraction

Overview

Initially I thought data cleaning was just finding data in the wrong format , and removing rows with missing data , but it is a field of research on its own, and too vast to completly look into within the given time frame.

Pipeline steps (implemented)

1. Initial data quality assessment

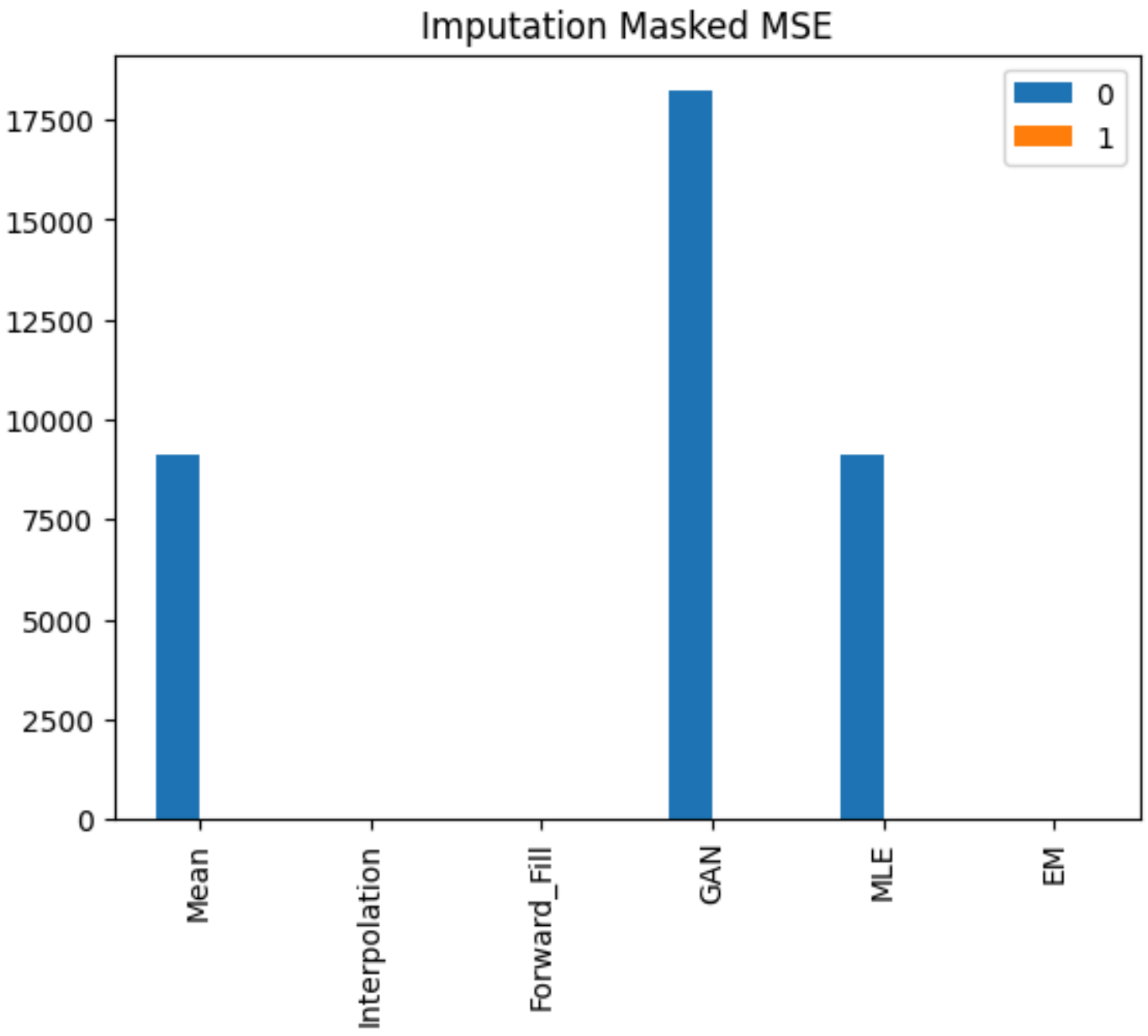
- We first want to get an overall idea of how the dataset is , before performing any cleaning activities. We can do that by analyzing the

following metrics

- **missing percentage:** trying to see how much of the data is missing
- **open high close low integrity:** seeing if they are logically sound
- **zero check:** to see how many zeros there are for values , as usually 0's arent present in stock data

2. Missing value handling / Imputation

- **mean imputation**: replace Nans with global average x
- **linear interpolation**: assume the asset moves at a constant velocity
  - $$x_t = x_a + (x_b - x_a)\frac{t-a}{b-a}$$
  - Variables: a, b are indices of known points; t is missing index.
- **Forward-fill**: runs of missing values for price series where appropriate; longer gaps left as NaN and removed from training windows.
  - $$x_t = x_{t-1}$$
- **maximum likelihood**: finds mean , variance of the data and uses a gaussian function to fill in the data
  - $$\mathcal{L}(\mu, \sigma) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$
- **Gan**: generator G learns to draw stock movements to fool a Discriminator D, Sounds cool , the formula looks even cooler, but its performance may not be the best.
  - $\min_G \max_D \mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - D(G(z)))]$
- **Expectation-Maximization (EM)**: An iterative two-step process.
  - E-Step: Calculate $E[x_{miss}|x_{obs}, \theta]$.
  - M-Step: Maximize $\theta = (\mu, \sigma)$ using the completed data.



3. Outlier detection & filtering

Intuition

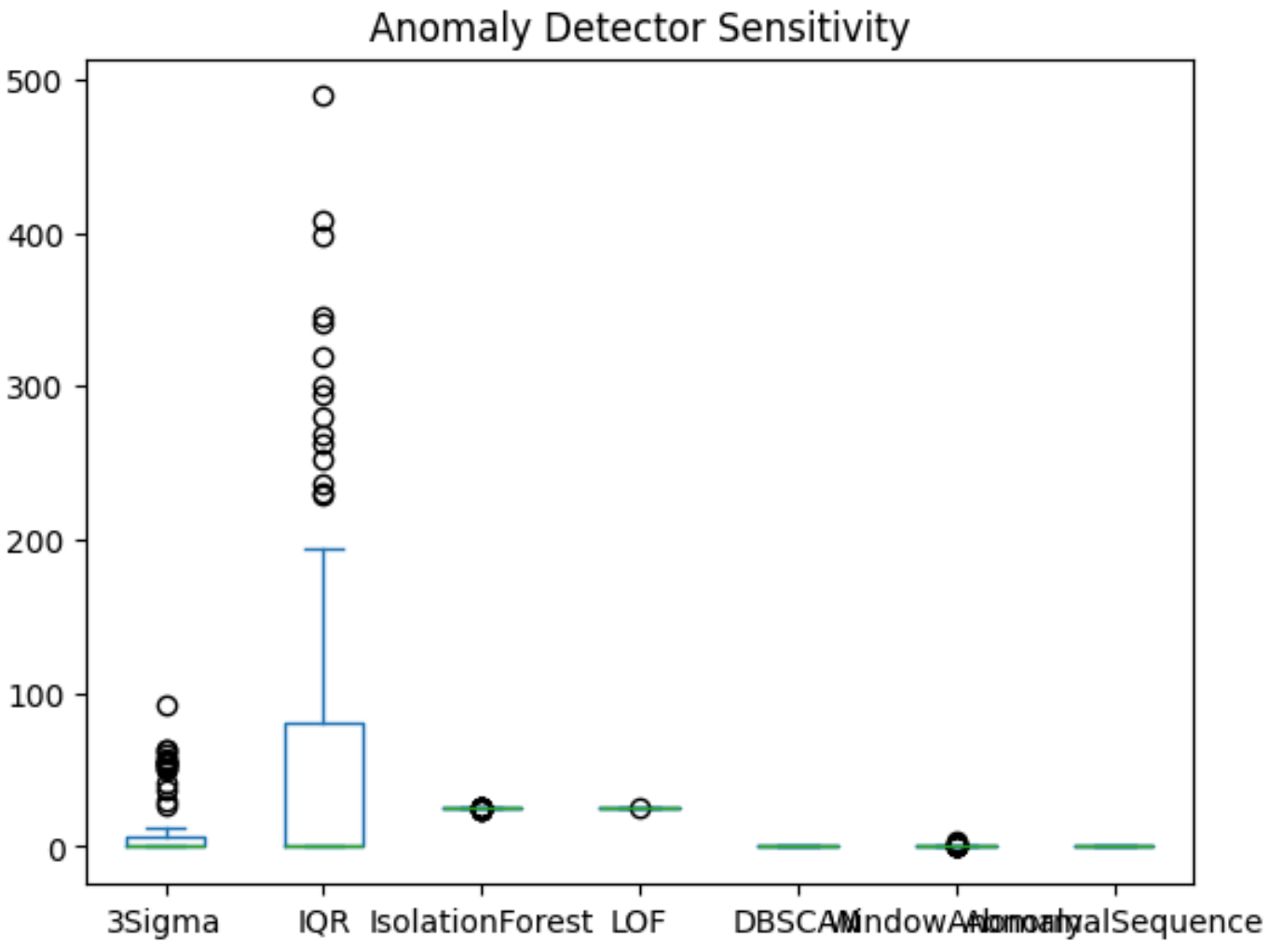
We separate signal from artifacts. Anomalies are points that deviate from expected statistical behavior.

Algorithm & Variable Dictionary

1. **3-Sigma**: Identifies values where $|x_t - \mu| > 3\sigma$.
    - Intuition: In a Normal distribution, this covers 99.7% of data. Anything outside is an extreme outlier.
2. **IQR (Interquartile Range)**: Uses quartiles to avoid being biased by the outliers.

- Math: Outlier if $x_t < Q_1 - 1.5 \cdot IQR$ or $x_t > Q_3 + 1.5 \cdot IQR$.

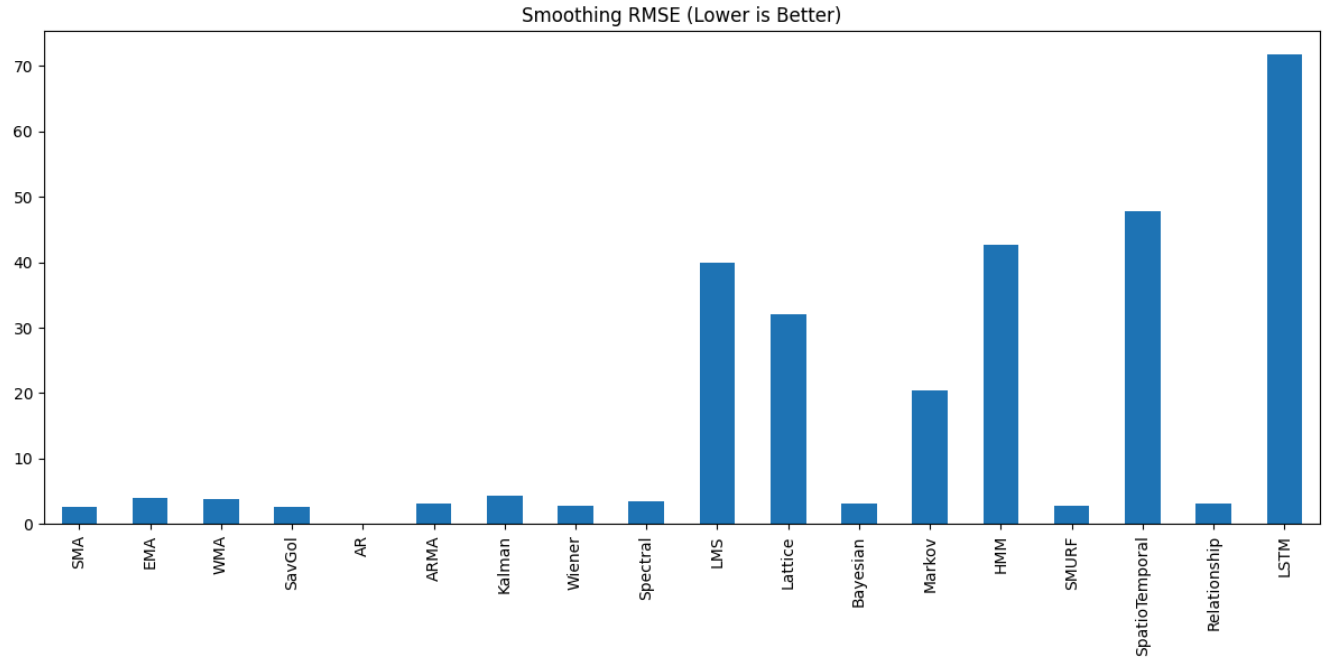
3. **Isolation Forest**: Built on the idea that outliers are few and different.
   - Mechanism: Randomly splits the data. Outliers end up in shorter branches (easier to isolate).
4. **LOF (Local Outlier Factor)**: Measures the density of a point compared to its neighbors.
   - Math: $LOF(k) = \frac{\text{avg neighbor density}}{\text{local density}}$. Score > 1 is sparse.
5. **DBSCAN**: Density-based clustering.
   - Variables: ϵ (radius), $MinPts$ (min points in radius).
6. **Window Anomaly**: Adapts to changing market conditions.
   - Math: $z_t = \frac{x_t - \text{roll\_mean}_k}{\text{roll\_std}_k}$.
7. **Abnormal Sequence**: Detects volatility clusters.
   - Math: Flag if $\text{Var}_{win} > \tau \cdot \text{Var}_{global}$.

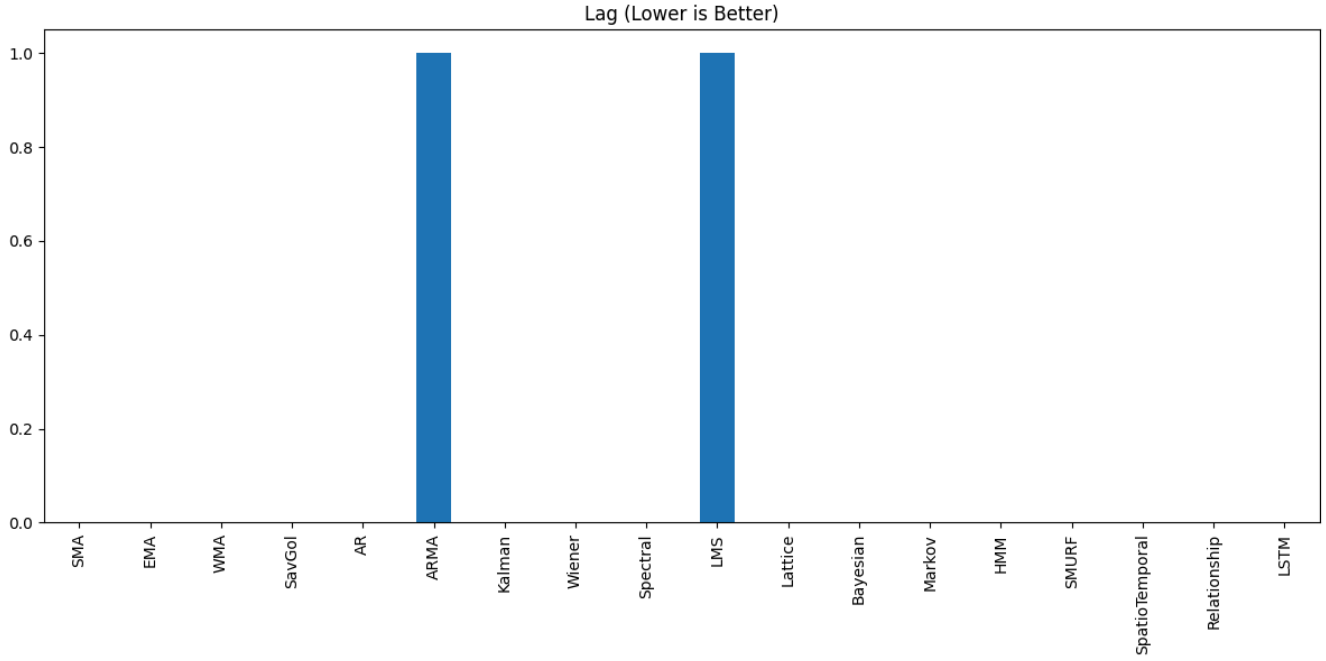


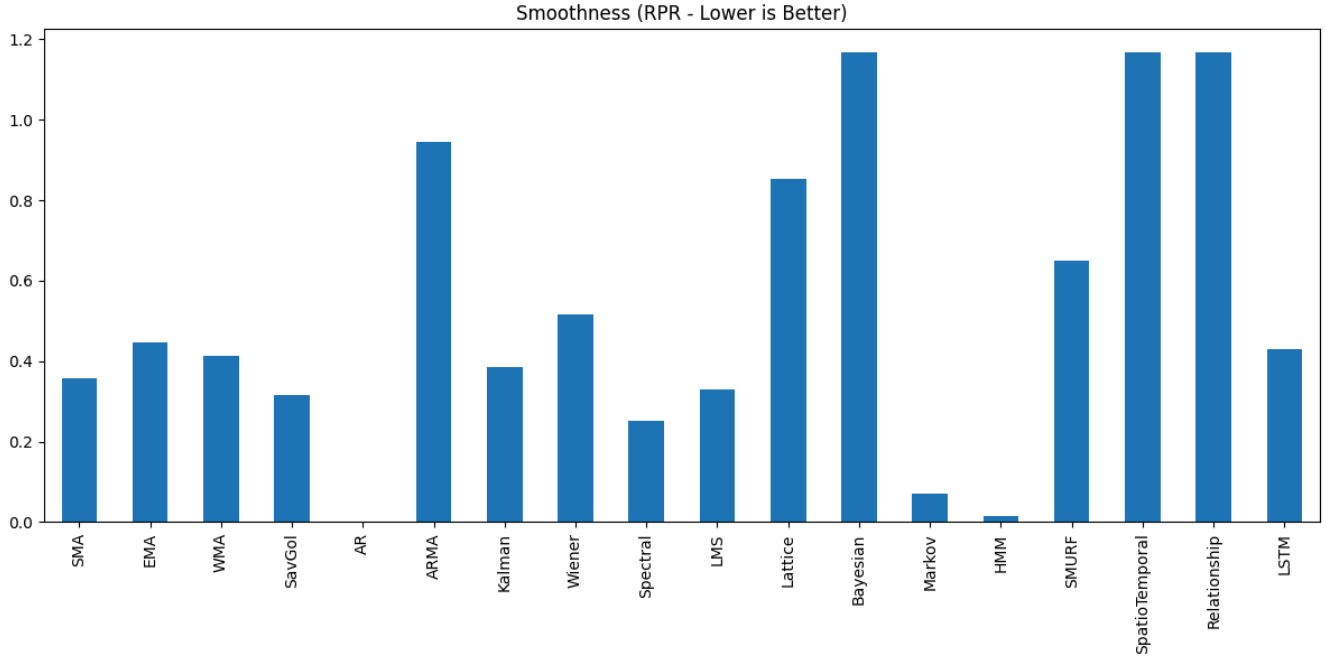
5. Smoothing & noise reduction

- I didnt end up implementing this , although i dug a deep rabbit hole into digital signal analysis and various smoothening techniques , i ended up not implementing any of the smoothening as that might skew the data for financial models

- But here are the results of some testing







Feature extraction

The most amount of time was spent here :)

1) Return Features (Simple, Log, Excess)

Intuition

Raw prices are not comparable across assets. Returns normalize by the price level: a 5 move on a 10 stock is large, but on a 1000 stock is small.

Math

- Simple return: $R_t = \frac{C_t}{C_{t-1}} - 1$
- Log return: $r_t = \ln\left(\frac{C_t}{C_{t-1}}\right) = \ln(C_t) - \ln(C_{t-1})$
- Excess return: $R_t^{excess} = R_t - r_{f,t}$ where $r_{f,t}$ is the per-day risk-free rate.

Notes

- Log returns are additive over time, which helps with modeling and aggregation.
- Excess returns isolate compensation over the risk-free baseline.

2) Lag Features (Momentum)

Intuition

Momentum effects suggest recent returns can contain predictive information. Lagged returns give the model direct access to recent history.

Math

For a series x_t, the k-lag feature is: x_{t-k}.

3) Rolling Statistics (Mean/Var/Min/Max/Std)

Intuition

Rolling statistics summarize local behavior: trend (mean), dispersion (variance/std), and extremes (min/max).

Math

For a trailing window of length w:

- Rolling mean: $\mu_t^{(w)} = \frac{1}{w}\sum_{i=0}^{w-1} x_{t-i}$
- Rolling variance: $\sigma_t^{2,(w)} = \frac{1}{w}\sum_{i=0}^{w-1}(x_{t-i} - \mu_t^{(w)})^2$
- Rolling min/max: $\min_{i=0..w-1} x_{t-i}$, $\max_{i=0..w-1} x_{t-i}$

Notes

Initial periods have NaNs until enough history accumulates.

4) Differencing (Trend Removal on Semilog Scale)

Intuition

Differencing removes slow-moving trends and makes a series more stationary. On a semilog plot, differencing log-price corresponds to log returns.

Math

- First difference: $\Delta x_t = x_t - x_{t-1}$
- Second difference: $\Delta^2 x_t = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2})$
  Applied to log price: $\Delta \ln(C_t) = \ln(C_t / C_{t-1})$

5) Rolling Z-Scores for Volume

Intuition

A volume z-score tells how unusual today's activity is compared to the recent past (liquidity and attention proxy).

Math

$z_t = \frac{V_t - \mu_t^{(w)}}{\sigma_t^{(w)}}$ where $\mu_t^{(w)}$ and $\sigma_t^{(w)}$ are rolling mean/std of volume over window w.

6) Volatility Indicators (Rolling Vol, ATR)

Intuition

Volatility measures dispersion of price moves and is central to risk-based strategies. ATR uses OHLC to quantify daily range volatility.

Math

- Realized volatility (annualized): $\sigma_t^{(w)} = \sqrt{252} \cdot \mathrm{Std}(r_{t-w+1:t})$
- True Range: $TR_t = \max(H_t - L_t, |H_t - C_{t-1}|, |L_t - C_{t-1}|)$
- ATR: $ATR_t^{(w)} = \frac{1}{w} \sum_{i=0}^{w-1} TR_{t-i}$

MACD

- MACD line: $MACD_t = EMA_{12}(C)_t - EMA_{26}(C)_t$
- Signal line: $Signal_t = EMA_9(MACD)_t$
- Histogram: $Hist_t = MACD_t - Signal_t$
  EMA recursion: $EMA_t = \alpha C_t + (1-\alpha) EMA_{t-1}$ with $\alpha = 2/(n+1)$.

RSI (14)

RSI measures the balance of recent gains vs losses.

- $RS = \frac{AvgGain}{AvgLoss}$
- $RSI = 100 - \frac{100}{1+RS}$
  Wilder smoothing is implemented via exponential smoothing with $\alpha = 1/14$.

Bollinger Bands (20, 2)

- Mid: $Mid_t = SMA_{20}(C)_t$
- Upper/Lower: $Mid_t \pm 2 \cdot Std_{20}(C)_t$
  Also exported: Bandwidth and %B position within bands.

ROC (Rate of Change)

$$ROC_t^{(n)} = \frac{C_t}{C_{t-n}} - 1$$

Stochastic Oscillator (14,3)

$\%K_t = 100 \cdot \frac{C_t - L_t^{(14)}}{H_t^{(14)} - L_t^{(14)}}$, where $H_t^{(14)}$ is rolling highest-high and $L_t^{(14)}$ is rolling lowest-low.
$\%D_t = SMA_3(\%K)_t$.

7) Volume-Based Indicators (OBV, A/D Line)

OBV Intuition

Volume can precede price. OBV accumulates volume conditioned on price direction.

OBV Math

Let $OBV_0 = 0$. For $t > 0$:

- if $C_t > C_{t-1}$: $OBV_t = OBV_{t-1} + V_t$
- if $C_t < C_{t-1}$: $OBV_t = OBV_{t-1} - V_t$
- else: $OBV_t = OBV_{t-1}$
  We also export OBV rate-of-change because absolute OBV magnitude is less meaningful.

Divergence (OBV)

- Price rising while OBV falls/plateaus: potential distribution (selling pressure).

- Price falling/sideways while OBV rises: potential accumulation (buying pressure).

Limitations (OBV)

- A single day of extreme volume can dominate the series.
- OBV uses direction only; magnitude of price moves is ignored.
- Best used alongside other indicators (e.g., RSI/MFI).

Accumulation/Distribution (A/D) Line

Close Location Value: $CLV_t = \frac{(C_t - L_t) - (H_t - C_t)}{H_t - L_t}$
Money Flow Volume: $MFV_t = CLV_t \cdot V_t$
A/D line: $AD_t = \sum_{i \le t} MFV_i$

Trend Confirmation (A/D)

- If price rises and A/D rises, buying pressure supports the uptrend.
- If price falls but A/D rises, buyers may be stepping in (potential reversal).

8) Trend Strength (ADX, Aroon)

ADX

ADX measures trend strength (non-directional). We export +DI, -DI, and ADX.
Directional movement: $+DM_t = H_t - H_{t-1}$, $-DM_t = L_{t-1} - L_t$ with standard gating rules.
True range and ATR normalize these to +DI/-DI, then $DX_t = 100 \cdot \frac{|+DI_t - -DI_t|}{|+DI_t + -DI_t|}$, and ADX is a smoothed DX.

Interpretation (common heuristic):

- $ADX < 20$: weak/absent trend
- $ADX > 25$: trend strength increasing

AroonUp: $100 \cdot \frac{N - \text{periods since highest high}}{N}$
AroonDown: $100 \cdot \frac{N - \text{periods since lowest low}}{N}$

9) Ichimoku Cloud

We export conversion, base, leading spans A/B (shifted forward), and lagging span (shifted back).

- Conversion (9): $(HH_9 + LL_9)/2$
- Base (26): $(HH_{26} + LL_{26})/2$
- Span A: (Conversion + Base)/2 shifted +26
- Span B: $(HH_{52} + LL_{52})/2$ shifted +26
- Lagging: Close shifted +26 (Chikou value representing past price at current time)

10) Fibonacci Retracement Levels (Rolling Swing)

Using a rolling swing high/low over a window, we compute retracement levels: 38.2%, 50%, 61.8%.
If $H_t^{(w)}$ is rolling high and $L_t^{(w)}$ rolling low, range $R_t = H_t^{(w)} - L_t^{(w)}$.
Then level 61.8%: $H_t^{(w)} - 0.618 \cdot R_t$ (and similarly for 50%, 38.2%).

11) Market Structure Events (BOS / CHoCH / MSS)

We export three discrete event indicators (values in {-1, 0, 1}):

- `bos`: break of structure (bullish +1 when Close breaks above prior swing high; bearish -1 when breaks below prior swing low)
- `choch`: change of character (BOS opposite to the prevailing structure direction)
- `mss`: market structure shift confirmation (after CHoCH, the next BOS in the new direction)

Swing highs/lows are detected using a trailing local-extrema window (`swing_window=3`) and confirmed with a lag of `swing_window` periods.

12) Filter-Based Features (From the Cleaning Pipeline)

Why include filters as *features*?

Filtering can create alternative views of the same price process (trend + residual decomposition). In quant modeling, we often feed both the filtered signal and the residual (original minus filtered) to let the model learn which regime benefits from filtering.
Important: For stock data, aggressive filtering can blur true jumps/gaps and may degrade predictive features. This notebook keeps filtering strictly in the feature layer. The cleaned dataset is not filtered.

Features exported

For each filter we export three series:

- Filtered close: $\tilde{C}_t$
- Residual: $e_t = C_t - \tilde{C}_t$
- Filtered log return: $\ln(\tilde{C}_t / \tilde{C}_{t-1})$

Filters included (and math)

1. SMA (5): $\tilde{C}_t = \frac{1}{k}\sum_{i=0}^{k-1} C_{t-i}$ (trailing window).
2. EMA (12): $\tilde{C}_t = \alpha C_t + (1-\alpha)\tilde{C}_{t-1}$, $\alpha = 2/(k+1)$.
3. WMA (5): $\tilde{C}_t = \frac{\sum_{i=1}^{k} w_i C_{t-k+i}}{\sum_{i=1}^{k} w_i}$ with linear weights $w_i = i$.
4. Savitzky-Golay (11,2): fit a degree-2 polynomial over a trailing window and evaluate at the most recent point.
5. 1D Kalman filter: state model $x_t = x_{t-1} + w_t$, observation $z_t = x_t + v_t$; update via Kalman gain K_t.
6. Wiener filter: minimum MSE linear filter; causal implementation.
7. Spectral low-pass: Causal Butterworth recursive filter (4^{th} order).

8. LMS adaptive filter: $w_{t+1} = w_t + 2\mu e_t x_t$ (causal recursive update).
9. Lattice filter (demo): recursive forward/backward prediction error update using reflection coefficients k_m.
   AR/ARMA model-based filters are excluded because they often emit convergence/non-stationarity warnings on stock series.

hence we get a grand total of , 127 features.

Sources

1. IBKR quant
   https://www.interactivebrokers.com/campus/ibkr-quant-news/unlocking-financial-data-cleaning-preprocessing-guide/
2. Blue Chip Algos
   https://bluechipalgos.com/blog/cleaning-and-preprocessing-financial-data-for-trading/
3. ACM digital library
4. Time series data cleaning a survey (*research paper*)
   https://www.researchgate.net/publication/338166752_Time_Series_Data_Cleaning_A_Survey
5. Feature Extraction Methods in Quantitative Structure (*research paper*)
   https://ieeexplore.ieee.org/document/9078111
6. Study of market indicators used for technical indicators (*research paper*)
   https://indianjournals.com/article/ijemr-12-2-011
7. stock prediction based on technical indicators using deep learning model (*research paper*)
   https://www.academia.edu/56099366/Stock_Prediction_Based_on_Technical_Indicators_Using_Deep_Learning_Model
8. 7 best indicators
   https://www.investopedia.com/top-7-technical-analysis-tools-4773275
9. What are state space models
   https://www.geeksforgeeks.org/artificial-intelligence/state-space-models-ssms/

2) Backtesting engine

You might be wondering why this is before models even though its task 3. I felt its better to build the backtesting engine first before making any models , otherwise i would have no way to know if the model I tested was good or not.

Overview & architecture

The main goal for the backtester was to simulate a real market situation as closely as possible.

There are many considerations to take when playing the market like:

- **portifolio distribution:** you may want to distribute among different sectors , use mpt, use nmpt, or maybe make a model purely for portifolio building
- **macro indicators:** many people first zoom out and use boc , choch, trend , market structure , graph structure analysis before zooming in for the perfect entry point
- **micro indicators**: candle stick patterns, indicators corssovers, the entire 9 yards
- **stop loss**: stop losses are key to playing the marketing, again different people have different stop loss strategies. Making a model entirely for better stop loss functionality is a feat on its own

Hence keeping all this in mind the primary feature for my backtester was to be able to use multiple models together that fall in the above categories so they can act together to make better returns. Unfortuately due to time constraints i havent trained any models for the stop loss , macro indicators , but this is an area to be further looked into

- The backtester accepts signals / weights from model notebooks and runs period-by-period portfolio simulation using historical OHLCV.
- Key modules:
  - `engine`: orchestration of simulation (apply weights, compute PnL)

- `portfolio`: weight normalization, transaction cost & slippage application
- `metrics`: compute CAGR, Sharpe, Max Drawdown, drawdown series
- `report`: assemble summary table + Bokeh visualization outputs
- `bokeh_plots`: produces interactive visualizations saved as HTML

Execution flow (per backtest)

1. **Inputs:**
   - Price series for universe subset (close prices), model-produced signals or weight matrices (date x tickers).
2. **Preprocessing:**
   - Align price & weight matrices; trim initial dates where all weights are zero (fix applied so computed "Start" equals first active testing date).
3. **Apply weights → returns:**
   - Daily / periodic portfolio return $r_{p,t} = \sum_i w_{i,t-1} * r_{i,t}$ (weights usually lagged by one period to avoid lookahead).
4. **Portfolio accounting:**
   - Transaction costs applied on weight adjustments:
     $\text{cost} = \text{turnover} * \text{fee\_rate}$
   - Rebalancing schedule: daily or custom (notebooks use time-split rebalancing or per-strategy setting).
5. **Risk & performance metrics:**
   - Cumulative returns, NAV, drawdown series.

Visualization layer

- Bokeh used to produce interactive charts (equity curve, drawdown, per-asset weight heatmaps).
- Notebooks save Bokeh outputs (HTML) or static PNG depending on notebook settings; the report embeds or links to these.

Key metrics

- **CAGR (annualized return):**
  - $CAGR = (NAV_{end}/NAV_{start})^{(1/\text{years})} - 1$
  - Intuition: per-year constant return that would produce observed cumulative return.
- **Sharpe ratio (annualized):**
  - $Sharpe = (\text{mean}(r_p)/\text{std}(r_p)) * \text{sqrt}(\text{annualization\_factor})$
  - Where r_p are periodic returns and mean is excess over risk-free.
  - Intuition: risk-adjusted return per unit of volatility.
- **Max Drawdown (MDD):**
  - $\text{drawdown}_t = (NAV_t/\max_{s \le t}(NAV_s)) - 1$; $MDD = \min(\text{drawdown}_t)$
  - Intuition: worst peak-to-trough loss
  - a key risk metric.
- **Turnover:**
  - $\text{turnover}_t = \sum(|w_t - w_{t-1}|)/2$ (or similar)
  - captures trading intensity and affects transaction costs.

Portfolio theory & allocation

- **Modern Portfolio Theory (MPT):**
  - Asset-level mean returns μ and covariance Σ
  - Efficient frontier solves for weights w that optimize expected return for a given risk or minimize variance for target return: $\min_w w'\Sigma w$ subject to $w'\mu = R_{target}$, $\sum w = 1$.
- **1/N:**
  - 1/N: equal-weight portfolio as a robust baseline (less estimation error).
- **Neo Modern Portifolio Theory**
  - Incorporates robust optimization, shrinkage or hierarchical risk parity variants.
- **Stop-loss & risk overlays:**
  - Stop-loss implemented as conditional cap on drawdown per asset or portfolio-level rule that triggers liquidation / de-risking.
- **Agentic backtester behavior :**
  - Backtester designed to accept modular components:
    - Macro model (allocation overlays, e.g., regime detection)

    - Micro model (alpha model producing weights / signals)
    - Portfolio manager (rebalancing & constraints)
  - Implementation pattern: chain of components applied in engine; notebooks provide examples wiring these together.

Files / code references

- `src/backtester/engine.py`
- `src/backtester/report.py`
- `src/backtester/metrics.py`
- `src/backtester/bokeh_plots.py`
- `notebooks/*` (each notebook calls `run_backtest` or `run_portfolio` wrapper)

Sources

- looked into backtesters like:
  - backtester.py
  - zipline

---

3) Models

Training vs Testing Splits

- **Asset Split:** I first split the data on assets, training on first 70 assets , testing on 30. But this may lead to memorization as the model may learn overall market trends , so I then shifted to time wise splits.
- **Time-split convention:** training on the first ~7 years and testing on the last ~1.5 years (18 months) unless the notebook specifies otherwise.
- Models are trained to produce signals or class labels which are converted to weights per the backtester's adapter layer (e.g., score → rank →

long/short weights, or probabilistic outputs → expected return based weights).

Implemented model groups

Regular indicators

- i started off with just testing how regular indicators fair
- i tested double moving averages , and all the indicators whose features were extracted and they performed decently well
- on average they had a total return of 300 - 400% over 10 years
- these were tested on the entire 10 years of data

Strategy	total_return	cagr	sharpe	max_drawdown	trades
Aroon Trend (25)	**408.6689**	**17.7403**	**1.276504**	**-0.194689**	**739**
ATR Trailing Stop (14, SMA20 entry)	**445.4761**	**18.5690**	**1.236761**	**-0.303475**	**796**
Ichimoku Cloud	295.1193	14.7917	1.110760	-0.207840	660
rMACD Sign (12/26/9)	347.4280	16.2336	1.104115	-0.250960	5807
MACD Crossover (12/26/9)	324.9993	15.6350	1.060591	-0.275388	6105
A/D Line Trend (AD > SMA20)	386.9150	17.2247	1.047893	-0.355595	873
ADX Directional Trend (14, >25)	260.6546	13.7447	1.027710	-0.219342	613
Fibonacci Trend (R<0.382, 60)	261.0574	13.7574	1.020969	-0.260492	691
OBV ROC Momentum (10)	83.6499	6.2929	0.987873	-0.114098	571
EMA Trend (20)	252.0783	13.4702	0.971971	-0.214772	3907
Stochastic Mean Reversion (14,3)	296.4909	14.8317	0.921887	-0.333556	4928
CCI Mean Reversion (20)	253.9591	13.5309	0.915577	-0.291572	3825
OBV Trend (OBV > SMA20)	218.4736	12.3331	0.883422	-0.281130	5395

Strategy	total_return	cagr	sharpe	max_drawdown	trades
Fibonacci Level (C > 50%, 60)	223.9094	12.5241	0.881677	-0.299864	742
WROBV Momentum (20)	81.7476	6.1818	0.876630	-0.149039	121
SMA Trend (20)	177.1496	10.7765	0.806716	-0.242747	4168
RSI Mean Reversion (14, 30/70)	145.5908	9.4402	0.721637	-0.291424	895
Bollinger Mean Reversion (20,2)	130.9919	8.7689	0.630777	-0.312668	3555

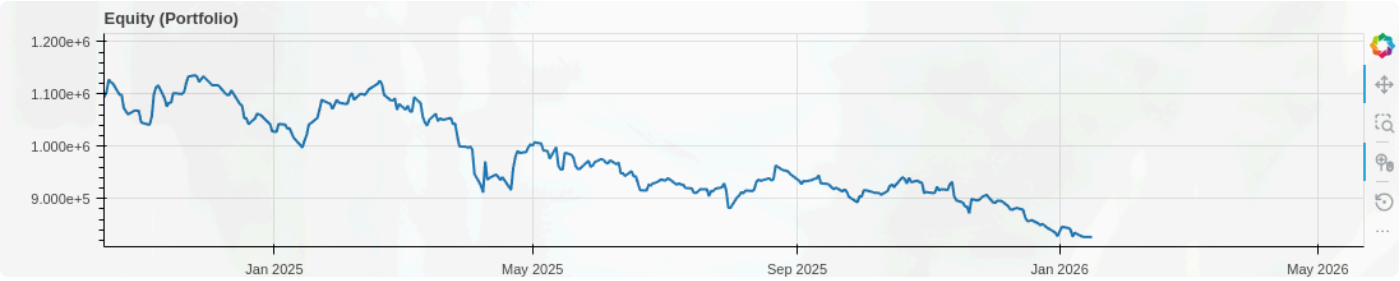
Candle stick patterns

- Back in the day , i used to trade purely based on candle stick patterns, i thought testing these out would net in good results , that turned out to be pretty wrong
- had horrible performance

The candle stick patterns were evaluated , by just using mathematical conditions instead of actually making a cnn recognize the pattern itself
Ex: A bullish engulfing can be mapped as current days High > prev days high and currend days low < prev days low
All the patterns use:

- 'bullish_engulfing'
- 'bearish_engulfing'
- 'morning_star'
- 'evening_star'
- 'three_white_soldiers'
- 'three_black_crows'
- 'piercing_line'
- 'hanging_man'
- 'hammer'
- 'inverse_hammer'

- 'tweezer_tops'
- 'doji'
- 'spinning_tops'



- goes to show , it is not wise to rely entirely on candle sticks , they are only a secondary indicator

Heiken Ashi

- this is one of my favourite strategies , and most profitable one based on my experiance
- it just clears up all the noise and gives an easy to read trend

Math

Close=1/4 (Open+High+Low+Close)
Open=1/2 (Open of Prev. Bar+Close of Prev. Bar)
High=Max(High, Open, Close)
Low=Min(Low, Open, Close)

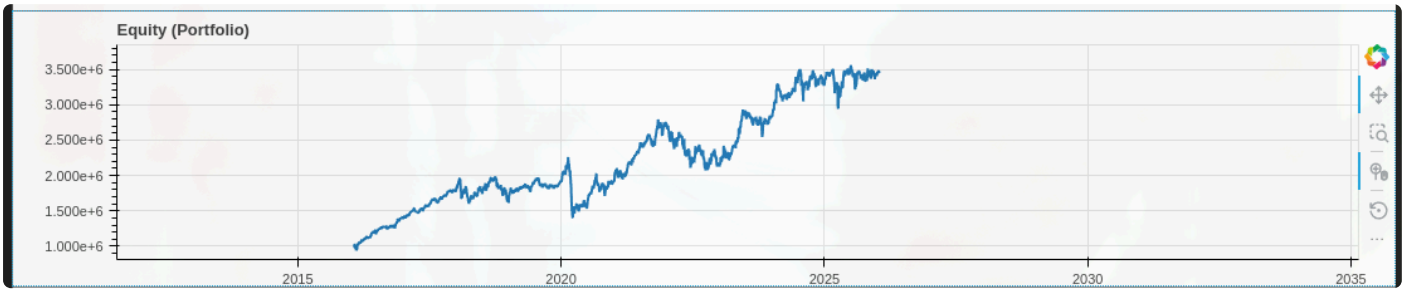
Strategy 1

buy on the first heiken ashi green candle
sell on the first heiken ashi red candle

- one my go to strategies when investing

```
Start                          2016-01-25 00:00:00
End                            2026-01-16 00:00:00
```

```
Duration                       3644 days 00:00:00
Initial Equity                        1000000.0
Final Equity                       3442629.38495
Equity Peak                        3558588.096368
Total Return [%]                       244.262938
CAGR [%]                                13.214728
Volatility (ann) [%]                    18.964477
Sharpe                                   0.749488
Sortino                                  1.178069
Max Drawdown [%]                       -37.680529
Calmar                                   0.350704
Best Day [%]                            10.973993
Worst Day [%]                          -12.142291
Avg Gross Exposure                       0.986287
Avg Net Exposure                         0.986287
Exposure Time [%]                       99.721227
Rebalance Days                               2123
Total Turnover                 2696623201.695777
Avg Daily Turnover                1073924.015012
```



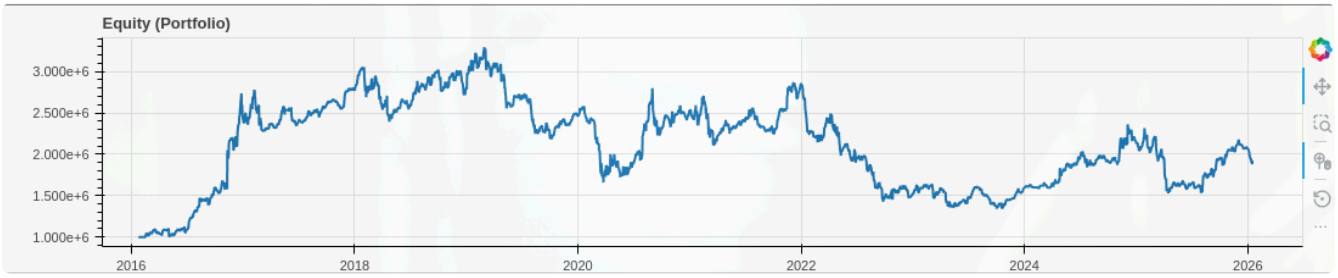
Strategy 2

- the biggest downside of this heiken ashi strategy is when its not paired with trend indicators
- hence we now pair it with MSS, BOS for trend confirmation
- And add an exit strategy when the red candles cross the fibonacci retracement levels instead of the first red candle

```
Start                          2016-01-25 00:00:00
End                            2026-01-16 00:00:00
Duration                        3644 days 00:00:00
Initial Equity                        1000000.0
```

Final Equity	1879313.237597
Equity Peak	3294251.077123
Total Return [%]	87.931324
CAGR [%]	6.539114
Volatility (ann) [%]	28.287175
Sharpe	0.363163
Sortino	0.597994
Max Drawdown [%]	-59.234152
Calmar	0.110394
Best Day [%]	29.787117
Worst Day [%]	-13.03212
Avg Gross Exposure	0.987655
Avg Net Exposure	0.987655
Exposure Time [%]	99.362804
Rebalance Days	499
Total Turnover	225685387.518696
Avg Daily Turnover	89878.688777

- lol , we did so much yet the returns dropped :(



Linear Regression Models

- Actually got better returns than expected for one of the most basic ml models
- trained on all 127 features

Math

- each feature is taken on a different dimension
- then we try to fit a hyperplane that has the least residual sum of squares (rss) distance from all the features
- y = mx + c

- here m represents the weights , c represents the bias
- we can extend this further for multiple features and that results in the above explanation
- now we can penalize the model, this is where ridge , lasso comes in. They prevent the model from over fitting
- hence now the model has to try to minimize rss + penalty
- Lasso
  - $$\text{Penalty} = \lambda \sum_{j=1}^{p} |\beta_j|$$
- Ridge:
  - $$\text{Penalty} = \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge - 1/n

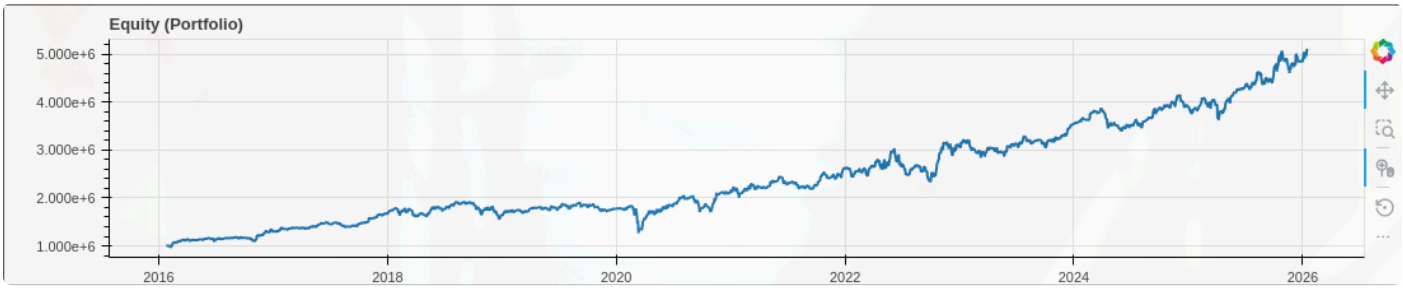
```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    4760608.744012
Equity Peak 4782795.775642
Total Return [%]    376.060874
CAGR [%]    16.959697
Volatility (ann) [%]    19.705881
```

```
Sharpe  0.894865
Sortino 1.374441
Max Drawdown [%]    -38.488
Calmar  0.440649
Best Day [%]    11.306162
Worst Day [%]   -18.954883
Avg Gross Exposure  0.994026
Avg Net Exposure    0.994026
Exposure Time [%]   99.402628
Rebalance Days  512
Total Turnover  372.673016
Avg Daily Turnover  0.148416
```

Ridge - MPT

```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    5082909.451176
Equity Peak 5099834.515544
Total Return [%]    408.290945
CAGR [%]    17.731467
Volatility (ann) [%]    19.402404
Sharpe  0.938796
Sortino 1.503368
Max Drawdown [%]    -34.178475
Calmar  0.51879
Best Day [%]    7.009781
Worst Day [%]   -12.957808
Avg Gross Exposure  0.994026
Avg Net Exposure    0.994026
Exposure Time [%]   99.402628
```

```
Rebalance Days  520
Total Turnover  480.569698
Avg Daily Turnover  0.191386
```



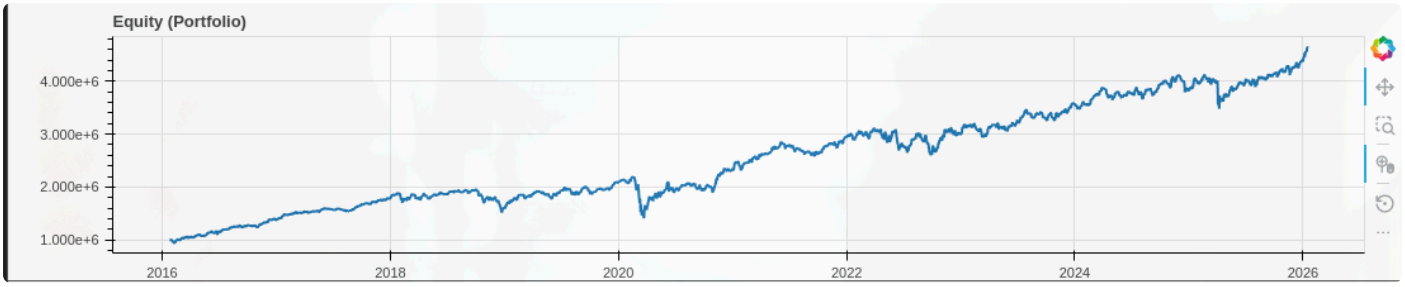
The other models

	model	Total Return [%]	CAGR [%]	Sharpe	Max Drawdown [%]
0	ridge	376.060874	16.959697	0.894865	-38.488000
1	lasso	366.072616	16.710969	0.925644	-35.774485
2	ols	329.067653	15.745624	0.841571	-37.055931

ElasticNet

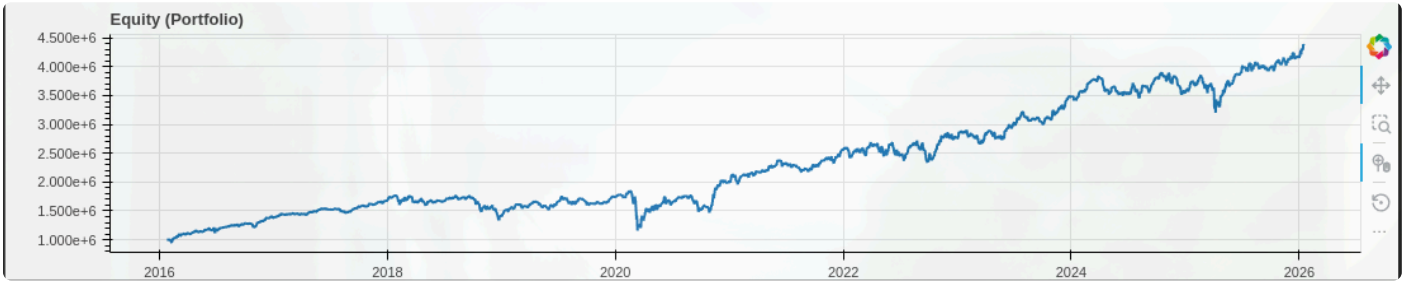
```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    4660726.161141
Equity Peak 4660726.161141
Total Return [%]    366.072616
CAGR [%]    16.710969
Volatility (ann) [%]    18.565528
Sharpe  0.925644
Sortino 1.444445
Max Drawdown [%]    -35.774485
Calmar  0.46712
Best Day [%]    10.251467
Worst Day [%]   -11.831363
Avg Gross Exposure  0.998009
Avg Net Exposure    0.998009
Exposure Time [%]   99.800876
```

```
Rebalance Days  2
Total Turnover  2.0
Avg Daily Turnover  0.000796
```



Bayesian Ridge Regression

```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    4368653.278401
Equity Peak 4389013.583917
Total Return [%]    336.865328
CAGR [%]    15.955106
Volatility (ann) [%]    19.759208
Sharpe  0.849282
Sortino 1.303927
Max Drawdown [%]    -37.144722
Calmar  0.429539
Best Day [%]    11.306162
Worst Day [%]   -18.954883
Avg Gross Exposure  0.994026
Avg Net Exposure    0.994026
Exposure Time [%]   99.402628
Rebalance Days  511
Total Turnover  377.622222
Avg Daily Turnover  0.150387
```



The above were all time based splits

Lasso | Ridge | orignal linear regression - Time split

	model	Total Return [%]	Sharpe	Max Drawdown [%]
0	ridge	54.010106	1.503062	-22.607705
1	ols	48.653071	1.358176	-22.183641
2	lasso	41.831591	1.396587	-18.490968

- a sharpe of 1.5 is actually crazy :))))

Ridge - time split

```
Start   2024-07-15 00:00:00
End 2026-01-15 00:00:00
Duration    549 days 00:00:00
Initial Equity  1000000.0
Final Equity    1540101.064611
Equity Peak 1546724.966447
Total Return [%]    54.010106
CAGR [%]    33.362219
Volatility (ann) [%]    20.499908
Sharpe  1.503062
Sortino 2.542599
Max Drawdown [%]    -22.607705
Calmar  1.475701
Best Day [%]    11.45801
Worst Day [%]   -7.482385
Avg Gross Exposure  0.989446
Avg Net Exposure    0.989446
Exposure Time [%]   98.944591
Rebalance Days  79
Total Turnover  110.7
Avg Daily Turnover  0.292084
```

Testing linear models on a smaller subset of indicators - BOS | CHoCH | Uptrend | Downtrend

- this is based on time split so only 2 years of testing

model Total Return [%] CAGR [%] Sharpe Max Drawdown [%]
lasso 41.831591 26.235639 1.396587 -18.490968
ridge 37.247918 23.501010 1.064191 -21.478081
ols 12.529357 8.187535 0.486278 -22.598123

here we see that with lesser no of features , lasso , ridge have a lot more significant effect on the model

Ridge on Heiken Ashi , Trend Features

```
Start                          2016-01-25 00:00:00
End                            2026-01-16 00:00:00
Duration                        3644 days 00:00:00
Initial Equity                         1000000.0
Final Equity                     4266614.210004
Equity Peak                       4363451.12814
Total Return [%]                     326.661421
CAGR [%]                              15.680289
Volatility (ann) [%]                  22.554975
Sharpe                                 0.759434
Sortino                                1.203814
Max Drawdown [%]                     -42.923471
Calmar                                 0.365308
Best Day [%]                          14.108949
Worst Day [%]                        -17.523651
Avg Gross Exposure                     0.999827
Avg Net Exposure                       0.999827
```

```
Exposure Time [%]                    99.960175
Rebalance Days                        2511
Total Turnover                 3171457517.046998
Avg Daily Turnover                1263025.693766
```

Bayesian Models

Conjugate Priors

math

- Objective: Model the probability p that the next-day return is positive.
- Likelihood: $y_t \sim \text{Bernoulli}(p)$, where $y_t = 1$ for up-days and $y_t = 0$ for down-days.
- Prior: $p \sim \text{Beta}(\alpha_0, \beta_0)$. This represents prior beliefs about the frequency of up-moves.
- Posterior: Due to conjugacy, the posterior is also a Beta distribution:

$$p|y_{1:n} \sim \text{Beta}(\alpha_0 + \sum y_i, \beta_0 + n - \sum y_i)$$

- Trading Signal: The posterior mean $\mathbb{E}[p|D] = \frac{\alpha_n}{\alpha_n+\beta_n}$ serves as the probability forecast.

Results

style Total Return [%] CAGR [%] Sharpe Max Drawdown [%]
1N 34.842081 22.053517 1.483627 -12.921751
MPT 16.694765 10.841123 0.750565 -13.418566

1/n

```
Start   2024-07-16 00:00:00
End 2026-01-16 00:00:00
Duration    549 days 00:00:00
Initial Equity  1000000.0
Final Equity    1348420.807267
```

```
Equity Peak 1348420.807267
Total Return [%]    34.842081
CAGR [%]    22.053517
Volatility (ann) [%]    14.069031
Sharpe  1.483627
Sortino 2.459612
Max Drawdown [%]    -12.921751
Calmar  1.706697
Best Day [%]    4.841407
Worst Day [%]   -5.886396
Avg Gross Exposure  0.992084
Avg Net Exposure    0.992084
Exposure Time [%]   99.208443
Rebalance Days  77
Total Turnover  17.9
Avg Daily Turnover  0.04723
```

3. Metropolis-Hastings MCMC

- **Method:** A random-walk Metropolis-Hastings algorithm to sample directly from the posterior $p(\beta|D)$.
- **Mechanism:** At each step, a proposal β' is sampled from a symmetric distribution (e.g., \mathcal{N}(\beta^{(t)}, \text{STEP_SCALE})). The proposal is accepted with probability:

$$\alpha = \min\left(1, \frac{p(D|\beta')p(\beta')}{p(D|\beta^{(t)})p(\beta^{(t)})}\right)$$

- **Inference:** After a burn-in period, the chain provides samples from the true posterior, allowing for a full predictive distribution.

```
Start   2024-07-16 00:00:00
End 2026-01-16 00:00:00
Duration    549 days 00:00:00
Initial Equity  1000000.0
Final Equity    1399095.648171
Equity Peak 1415358.920382
Total Return [%]    39.909565
CAGR [%]    25.092596
```

```
Volatility (ann) [%]    17.372862
Sharpe  1.371819
Sortino 2.343033
Max Drawdown [%]    -18.137274
Calmar  1.383482
Best Day [%]    9.790165
Worst Day [%]   -5.671871
Avg Gross Exposure  0.992084
Avg Net Exposure    0.992084
Exposure Time [%]   99.208443
Rebalance Days  65
Total Turnover  13.577812
Avg Daily Turnover  0.035825
```

4. Variational Inference (VI)

- **Objective:** Approximate the posterior $p(\beta|D)$ with a simpler distribution $q_\lambda(\beta)$ (e.g., a diagonal Gaussian) by maximizing the Evidence Lower Bound (ELBO):

$$\mathcal{L}(\lambda) = \mathbb{E}_{q_\lambda}[\log p(D, \beta) - \log q_\lambda(\beta)]$$

- **Optimization:** Uses stochastic gradient descent with the **reparameterization trick** ($\beta = \mu + \sigma \odot \epsilon, \epsilon \sim \mathcal{N}(0, I)$) to optimize the variational parameters μ and σ.
- **Benefit:** Much faster than MCMC for large datasets while providing a better uncertainty estimate than a point estimate.

```
Start   2024-07-16 00:00:00
End 2026-01-16 00:00:00
Duration    549 days 00:00:00
Initial Equity  1000000.0
Final Equity    1202721.961048
Equity Peak 1232920.871974
Total Return [%]    20.272196
CAGR [%]    13.095023
Volatility (ann) [%]    16.843844
Sharpe  0.81309
Sortino 1.255354
```

```
Max Drawdown [%]    -17.262441
```

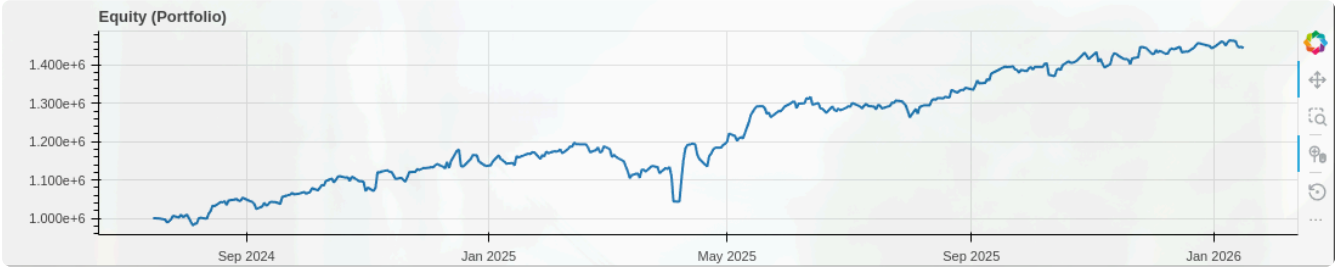
Bayesian Sharpe ratio asset selection

```
Start   2024-07-16 00:00:00
End 2026-01-16 00:00:00
Duration    549 days 00:00:00
Initial Equity  1000000.0
Final Equity    1337823.370352
Equity Peak 1337823.370352
Total Return [%]    33.782337
CAGR [%]    21.413185
Volatility (ann) [%]    15.64573
Sharpe  1.315634
Sortino 2.161809
Max Drawdown [%]    -13.072936
```

Bayesian Stochastic volatility particle filter

```
Start   2024-07-16 00:00:00
End 2026-01-16 00:00:00
Duration    549 days 00:00:00
Initial Equity  1000000.0
Final Equity    1443528.656117
Equity Peak 1464212.77093
Total Return [%]    44.352866
CAGR [%]    27.727263
Volatility (ann) [%]    15.175843
Sharpe  1.684742
Sortino 2.890453
Max Drawdown [%]    -12.852631
```

- CRAZY, a sharp of 1.68



Bayesian logistic regression mcmc nuts

```
Start   2024-07-16 00:00:00
End 2026-01-16 00:00:00
Duration    549 days 00:00:00
Initial Equity  1000000.0
Final Equity    1404824.280434
Equity Peak 1424496.884119
Total Return [%]    40.482428
CAGR [%]    25.433826
Volatility (ann) [%]    19.252787
Sharpe  1.269686
Sortino 2.139035
Max Drawdown [%]    -19.680988
```

Decision trees / Random forests

- Decision tree splits on features to partition feature space
- objective: minimize impurity (Gini/Entropy for classification) or MSE (regression).
- Random Forest: ensemble of trees trained on bootstrap samples with feature subsampling
- final prediction is averaged (regression) or majority-vote (classification).

Decision Tree Regression

- asset split
  For regression, a common impurity is the mean squared error (MSE):

$$\mathrm{MSE}(S) = \frac{1}{|S|} \sum_{j \in S} (y_j - \bar{y}_S)^2.$$

We choose (k,tau) to minimize the weighted impurity:

$$\frac{|L|}{|S|}\mathrm{MSE}(L) + \frac{|R|}{|S|}\mathrm{MSE}(R).$$

```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    4474722.441032
Equity Peak 4496049.034339
Total Return [%]    347.472244
CAGR [%]    16.234722
Volatility (ann) [%]    18.447066
Sharpe  0.90789
Sortino 1.44332
Max Drawdown [%]    -34.359566
```

Decision Tree Classification

- asset split
  For binary classification, the Gini impurity for class probability p (class 1) is:

$$G(p) = 2p(1-p) = 1 - (p^2 + (1-p)^2).$$

(Entropy is another option: H(p) = -p log p -(1-p) log(1-p).)
We choose the split minimizing the weighted impurity.

```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    5145366.45647
Equity Peak 5145366.45647
Total Return [%]    414.536646
CAGR [%]    17.875911
Volatility (ann) [%]    17.692325
Sharpe  1.018193
Sortino 1.633754
Max Drawdown [%]    -29.800315
```

Bagging

reduces variance by averaging bootstrapped model predictions.

bagged Decision Tree

Bagging trains B base learners on bootstrap resamples of the training data.

For regression with predictions f_b(x), the bagged predictor is the average:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x).$$

For classification, we typically use majority vote or average predicted probabilities.

Bagging primarily reduces variance by averaging many noisy, high-variance learners (like deep trees).

```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    5065715.778746
Equity Peak 5089859.055038
Total Return [%]    406.571578
CAGR [%]    17.691423
Volatility (ann) [%]    18.069835
Sharpe  0.99182
Sortino 1.60103
Max Drawdown [%]    -29.624449
```

- wow no difference between a regular dt , and a bagged dt even though the bagged one took 30 mins to train on my cpu :(

Random Forests Regressor

A random forest is bagging plus additional randomness at each split:

- Each tree is trained on a bootstrap sample.
- At each node split, only a random subset of features of size m is considered.

This reduces correlation between trees and further reduces variance.

$$\hat{f}_{\mathrm{RF}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x).$$

```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
Final Equity    4810454.864092
Equity Peak 4833381.563091
Total Return [%]    381.045486
CAGR [%]    17.082073
Volatility (ann) [%]    18.052236
Sharpe  0.963911
Sortino 1.547181
Max Drawdown [%]    -31.46595
```

- wow , not much difference again

Random Forest Classifier

- asset split
  For class 1 probability:

$$\hat{p}_{\mathrm{RF}}(y = 1 \mid x) = \frac{1}{B} \sum_{b=1}^{B} \hat{p}_b(y = 1 \mid x).$$

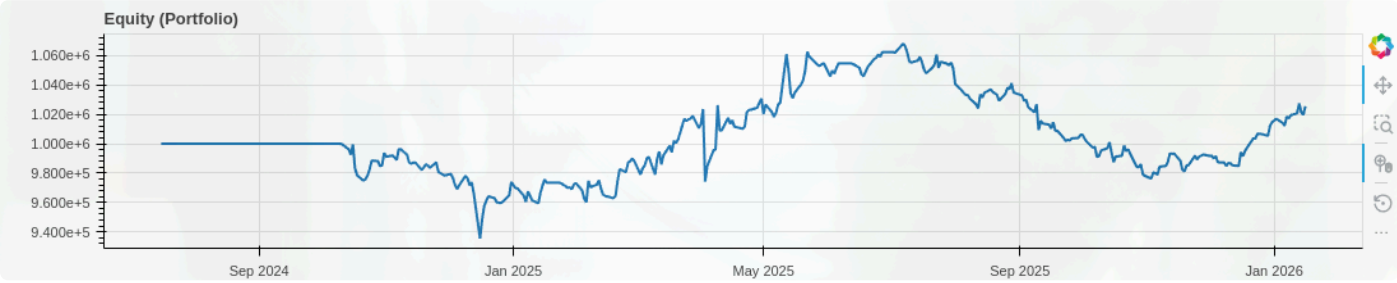
```
Start   2016-01-25 00:00:00
End 2026-01-16 00:00:00
Duration    3644 days 00:00:00
Initial Equity  1000000.0
```

```
Final Equity    4792965.058824
Equity Peak 4815302.888518
Total Return [%]    379.296506
CAGR [%]    17.039264
Volatility (ann) [%]    17.708952
Sharpe  0.977184
Sortino 1.562666
Max Drawdown [%]    -30.686674
```

Trying Decision Trees on Classifying Cointegration pairs for trading

- task 4 aswell

```
Start   2024-10-11 00:00:00
End 2026-01-16 00:00:00
Duration    462 days 00:00:00
Initial Equity  999500.0
Final Equity    1025525.466274
Equity Peak 1068130.55984
Total Return [%]    2.603849
CAGR [%]    2.071068
Volatility (ann) [%]    10.041391
Sharpe  0.249845
Sortino 0.37452
Max Drawdown [%]    -8.610758
```



Dimensionality Reduction

- The features so far are performing decently well , all the ml models are getting decent results

- The indicators are usually classified into **momentum , volatility , volume, trend**.
- what happens if we bucket these features and use PCA before training the model on them , to reduce the no of features

volume_features = [
'log_volume', 'volume_roll_mean_5', 'volume_roll_mean_20',
'volume_roll_mean_60',
'volume_roll_std_5', 'volume_roll_std_20', 'volume_roll_std_60',
'volume_zscore_5', 'volume_zscore_20', 'volume_zscore_60',
'volume_minmax_20', 'obv', 'obv_roc_10', 'wrobv_20', 'ad_line'
]
volatility_features = [
'logret_roll_var_5', 'logret_roll_var_10', 'logret_roll_var_20', 'logret_roll_var_60',
'logret_roll_std_5', 'logret_roll_std_10', 'logret_roll_std_20', 'logret_roll_std_60',
'atr_14', 'realized_vol_20', 'bb_bb_bandwidth'
]
trend_features = [
'sma_20', 'sma_50', 'ha_ha_open', 'ha_ha_high', 'ha_ha_low', 'ha_ha_close',
'sma_ratio_20', 'ema_ratio_20', 'bb_bb_mid', 'bb_bb_upper', 'bb_bb_lower',
'bb_bb_percent_b', 'adx_plus_di', 'adx_minus_di', 'adx_adx', 'adx_adx_raw',
'aroon_aroon_up', 'aroon_aroon_down', 'ichimoku_ichimoku_conv',
'ichimoku_ichimoku_base', 'ichimoku_ichimoku_span_a',
'ichimoku_ichimoku_span_b',
'ichimoku_ichimoku_lagging', 'bos', 'choch', 'mss'
]
momentum_features = [
'rsi_14', 'macd_macd', 'macd_macd_signal', 'macd_macd_hist',
'rmacd_12_26_9', 'stoch_stoch_k', 'stoch_stoch_d', 'roc_10', 'cci_20'
]

Results from PCA:

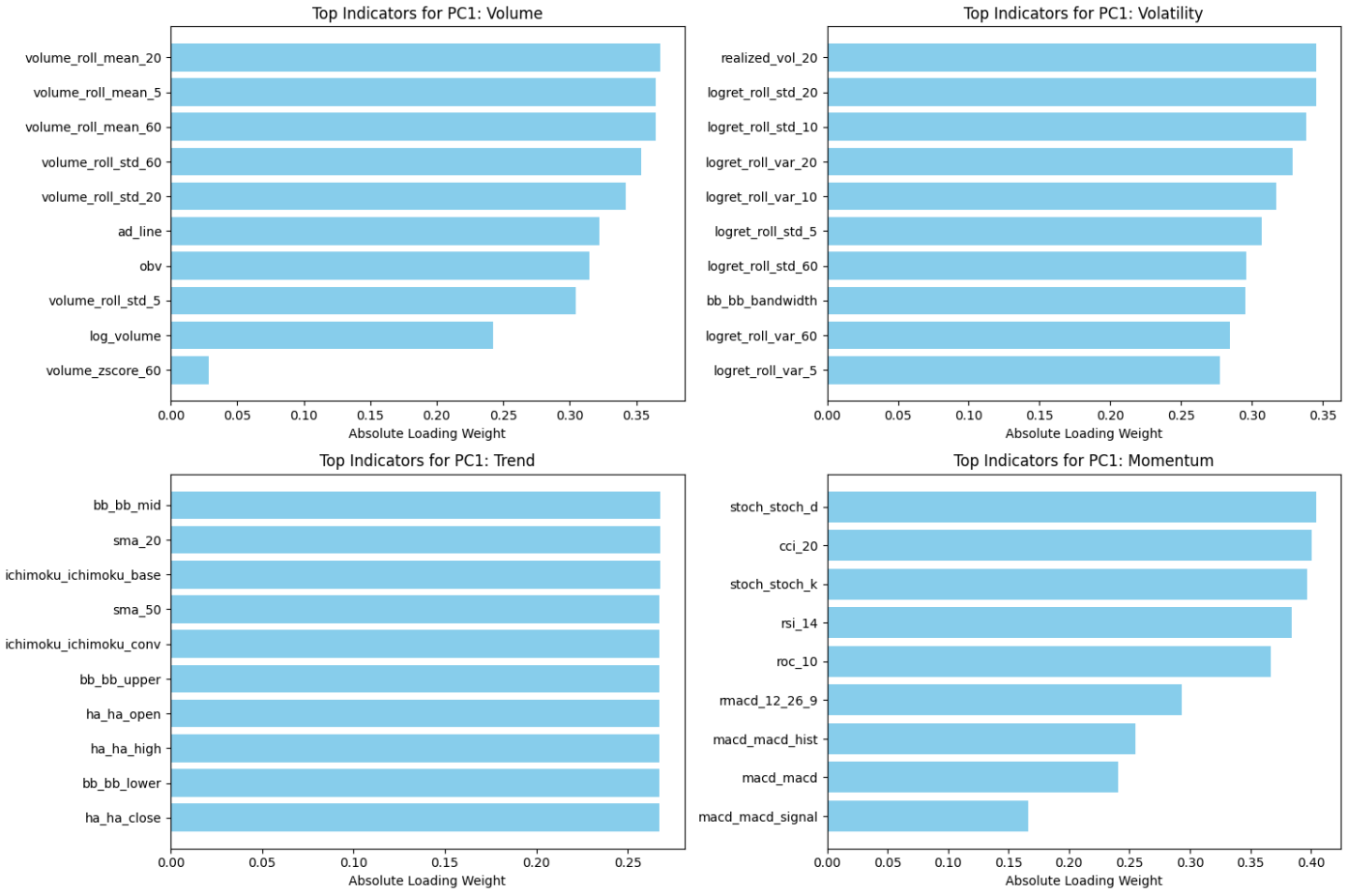
Processing Bucket: Volume
Explained Variance: 77.63%

Processing Bucket: Volatility
Explained Variance: 87.80%

Processing Bucket: Trend
Explained Variance: 81.35%

Processing Bucket: Momentum
Explained Variance: 87.46%

Final feature shape: (174800, 12)



Now training a random forest model on these reduced features

```
Start                          2023-01-03 00:00:00
End                            2026-01-16 00:00:00
Duration                        1109 days 00:00:00
Initial Equity                          999500.0
Final Equity                      2152021.721414
Equity Peak                       2272809.230218
Total Return [%]                      115.309827
```

CAGR [%]	28.868597
Volatility (ann) [%]	27.554454
Sharpe	1.055567
Sortino	1.839131
Max Drawdown [%]	-29.149494
Calmar	0.990364
Best Day [%]	16.650574
Worst Day [%]	-8.909668
Avg Gross Exposure	0.999002
Avg Net Exposure	0.999002
Exposure Time [%]	99.868938
Rebalance Days	763
Total Turnover	739976539.195652
Avg Daily Turnover	969825.084136

crazy , this led to a sizeable improvement

Other Dimensionality reducing metrics

ICA + random forest

Start	2023-01-03 00:00:00
End	2026-01-16 00:00:00
Duration	1109 days 00:00:00
Initial Equity	999500.0
Final Equity	2049173.70531
Equity Peak	2144827.764263
Total Return [%]	105.01988
CAGR [%]	26.798358
Volatility (ann) [%]	27.470266
Sharpe	0.999511
Sortino	1.65259
Max Drawdown [%]	-27.489511

LDA + Random Forest

```
Start                          2023-01-03 00:00:00
End                            2026-01-16 00:00:00
Duration                        1109 days 00:00:00
Initial Equity                          999500.0
Final Equity                      1421220.834308
Equity Peak                       1494934.910473
Total Return [%]                       42.19318
CAGR [%]                              12.346186
Volatility (ann) [%]                  22.917871
Sharpe                                 0.620082
Sortino                                1.082296
Max Drawdown [%]                     -20.156508
```

SVD + Random Forest

```
Start                          2023-01-03 00:00:00
End                            2026-01-16 00:00:00
Duration                        1109 days 00:00:00
Initial Equity                          999500.0
Final Equity                      1851947.513294
Equity Peak                       1968486.970031
Total Return [%]                       85.287395
CAGR [%]                               22.62498
Volatility (ann) [%]                  26.553381
Sharpe                                 0.898677
Sortino                                1.514845
Max Drawdown [%]                     -24.348286
```

NMF + Rnadom Forest

```
Start                          2023-01-03 00:00:00
End                            2026-01-16 00:00:00
Duration                        1109 days 00:00:00
Initial Equity                          999500.0
Final Equity                      2055876.266266
Equity Peak                       2148747.132721
Total Return [%]                      105.690472
```

```
CAGR [%]                                26.935366
Volatility (ann) [%]                    28.076136
Sharpe                                   0.987646
Sortino                                  1.643141
Max Drawdown [%]                       -28.548289
```

Kernel PCA + Random Forest

```
Start                     2023-01-03 00:00:00
End                       2026-01-16 00:00:00
Duration                   1109 days 00:00:00
Initial Equity                       999500.0
Final Equity                  2497625.985664
Equity Peak                   2604653.277231
Total Return [%]                  149.887542
CAGR [%]                           35.374725
Volatility (ann) [%]               27.114425
Sharpe                              1.250207
Sortino                             2.154106
Max Drawdown [%]                  -21.02126
Calmar                              1.682807
Best Day [%]                       14.881038
Worst Day [%]                      -7.275235
Avg Gross Exposure                  0.999064
Avg Net Exposure                    0.999064
Exposure Time [%]                  99.868938
Rebalance Days                           763
Total Turnover            803482180.929999
Avg Daily Turnover          1053056.593617
```

- KERNEL PCA THE GOAT
- *kernel pca took a random forest model which had a sharpe of around 0.9 to 1.2 , thats massive improvement*

Boosting

- AdaBoost: sequential reweighting of training examples; final predictor is weighted sum of weak learners.

- Gradient boosting (GBDT): sequentially fit models to residuals (negative gradient).
- HistGradientBoosting: histogram-based gradient boosting for speed.
- XGBoost / LightGBM / CatBoost: optimized GBDT implementations with system-specific enhancements (regularization, leaf-wise growth, categorical handling).
- Notebooks: `notebooks/boosted algorithms/*`

Adaboost

AdaBoost builds an additive classifier from weak learners (often stumps - decision trees with only leaves):

$$F_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x),$$

where each weak learner $h_t(x) \in \{-1, +1\}$ is trained on a reweighted dataset.
Initialize sample weights: $w_i^{(1)} = 1/n$.
At iteration t, fit h_t to minimize weighted error:

$$\varepsilon_t = \sum_{i=1}^{n} w_i^{(t)} \, \mathbb{1}[h_t(x_i) \neq y_i].$$

Set learner weight:

$$\alpha_t = \tfrac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}.$$

Update sample weights:

$$w_i^{(t+1)} \propto w_i^{(t)} \, \exp(-\alpha_t y_i h_t(x_i)).$$

This increases weight on misclassified points, focusing subsequent learners on hard examples.

```
Start   2024-07-19 00:00:00
End 2026-01-15 00:00:00
Duration    545 days 00:00:00
```

```
Initial Equity  999500.0
Final Equity    1191482.935094
Equity Peak 1216126.229841
Total Return [%]    19.207897
CAGR [%]    12.567775
Volatility (ann) [%]    17.220194
Sharpe  0.769042
Sortino 1.318385
Max Drawdown [%]    -15.924873
```

LightGBM

LightGBM is a gradient boosting tree framework emphasizing speed:

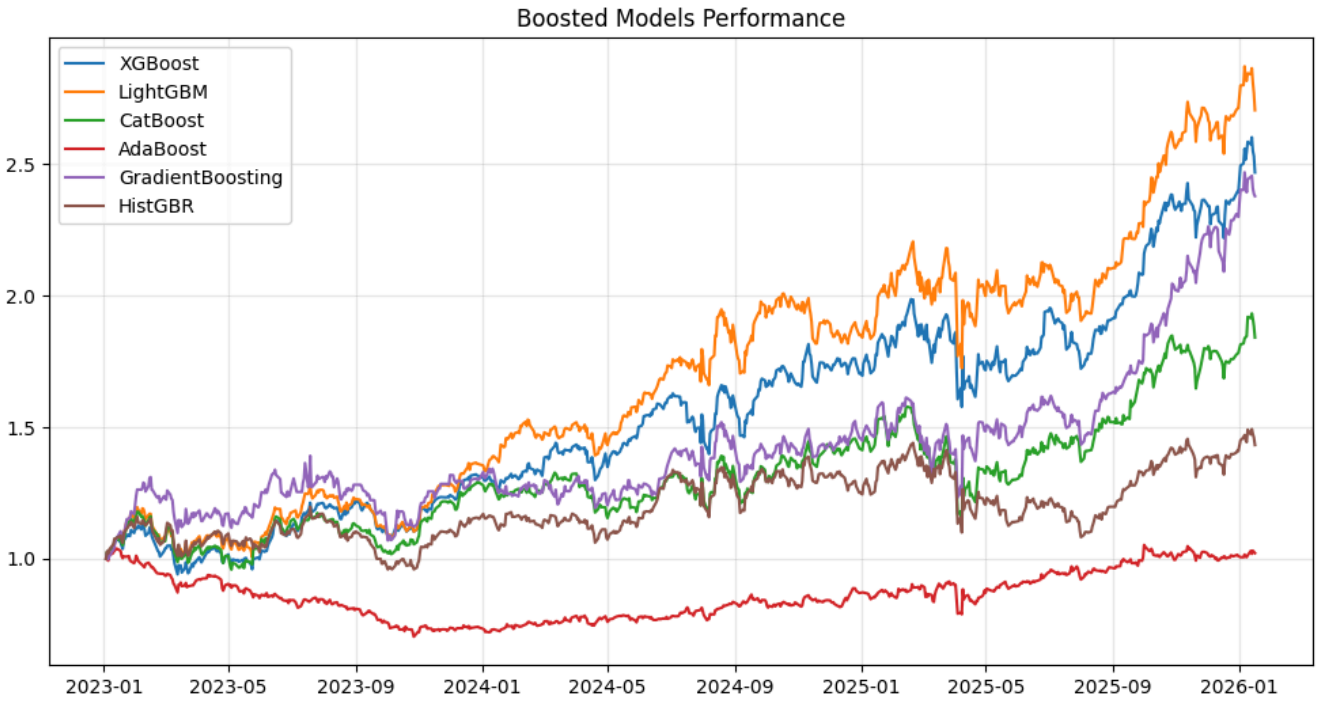
- Histogram-based splits
- Leaf-wise (best-first) tree growth
- Optional tricks like Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB)
  Model remains an additive boosted tree ensemble:

$$F_M = F_0 + \sum_{m=1}^{M} \nu h_m.$$

- didnt have time to run this on 127 features , as these were very computationaly expensive

Kernel PCA + Boosted Random Forests

- since kernel pca worked so well on regular random forests , lets try it on boosted random forests aswell



- LightGBM is getting phenomenal results

```
Start                          2023-01-03 00:00:00
End                            2026-01-16 00:00:00
Duration                        1109 days 00:00:00
Initial Equity                         1000000.0
Final Equity                     2704431.272386
Equity Peak                      2870954.648079
Total Return [%]                     170.443127
CAGR [%]                              38.960465
Volatility (ann) [%]                  26.180521
Sharpe                                   1.3856
Sortino                                2.478817
Max Drawdown [%]                     -21.823081
Calmar                                 1.785287
Best Day [%]                          14.970449
Worst Day [%]                         -8.727961
Avg Gross Exposure                     0.999199
Avg Net Exposure                       0.999199
Exposure Time [%]                     99.868938
Rebalance Days                              763
Total Turnover                1304452669.591913
Avg Daily Turnover               1709636.526333
```

- damn a sharpe of 1.38 , pretty good

Clustering + kernel PCA

if me doing dimensionality reduction on pre selected buckets improved results so much , lets see what happens when i let kmeans cluster the indicators into groups and then do kernel pca , then build lightgbm on top of that

Start	2023-01-03 00:00:00
End	2026-01-16 00:00:00
Duration	1109 days 00:00:00
Initial Equity	1000000.0
Final Equity	1067919.100706
Equity Peak	1138770.712949
Total Return [%]	6.79191
CAGR [%]	2.196937
Volatility (ann) [%]	17.503723
Sharpe	0.21132
Sortino	0.345136
Max Drawdown [%]	-21.124486
Calmar	0.104
Best Day [%]	7.828659
Worst Day [%]	-4.970383
Avg Gross Exposure	0.999572
Avg Net Exposure	0.999572
Exposure Time [%]	99.868938
Rebalance Days	763
Total Turnover	1332044640.569925

- damn a massive flop

CNNs

- 1D CNN for time-series: convolution over time axis to extract local temporal patterns.
- 2D CNNs over feature-grid: treat feature cross-section as spatial grid; use AdaptiveAvgPool2d to avoid hardcoding flatten sizes.
- LeNet/AlexNet-like architectures adjusted for small channels and adaptive pooling.

1d

```
Start   2024-10-11 00:00:00
End 2026-01-15 00:00:00
Duration    461 days 00:00:00
Initial Equity  999500.0
Final Equity    1053899.627255
Equity Peak 1056166.329386
Total Return [%]    5.442684
CAGR [%]    4.33095
Volatility (ann) [%]    18.502639
Sharpe  0.318121
Sortino 0.502091
Max Drawdown [%]    -19.777488
```

ta small

```
Start   2024-07-19 00:00:00
End 2026-01-15 00:00:00
Duration    545 days 00:00:00
Initial Equity  999500.0
Final Equity    1237306.161508
Equity Peak 1243026.74204
Total Return [%]    23.792512
CAGR [%]    15.466812
Volatility (ann) [%]    19.388288
Sharpe  0.833904
Sortino 1.347956
Max Drawdown [%]    -17.801731
```

- i probably didnt run them for enough epochs , hence the bad performance

LSTM / ALSTM

- LSTM: RNN variant capturing long-range dependencies using gated units.
- ALSTM: attention-augmented LSTMs (attention layer over sequence).
- Notebooks: `notebooks/hierarchical graph neural network/*` (LSTM variants included)

LSTM

```
Start   2024-07-19 00:00:00
End 2026-01-15 00:00:00
Duration    545 days 00:00:00
Initial Equity  999500.0
Final Equity    1283256.554754
Equity Peak 1285544.214803
Total Return [%]    28.38985
CAGR [%]    18.338928
Volatility (ann) [%]    20.686506
Sharpe  0.91287
Sortino 1.483994
Max Drawdown [%]    -19.959862
```

ALSTM

Trying to implement some research papers

Graph Neural Networks HGNN)

- HGNN: hierarchical graph neural network as per the paper implemented variants: HGNN_M, HGNN_I, HGNN_full with different hierarchy construction and pruning strategies. Edge pruning uses top-k significant connections; market-aware graphs included in FinMamba-like architectures.

```
Start   2024-07-19 00:00:00
End 2026-01-15 00:00:00
Duration    545 days 00:00:00
Initial Equity  999500.0
Final Equity    1280139.57151
Equity Peak 1285024.497098
Total Return [%]    28.077996
CAGR [%]    18.145175
Volatility (ann) [%]    21.344071
Sharpe  0.882607
Sortino 1.480118
```

```
12  Max Drawdown [%]    -19.594495
```

- maybe my implementation was wrong , but it wasnt that amazing

FinMamba (Paper-inspired implementation)

- Custom PyTorch module combining SSM-like blocks, graph attention aggregation, and multi-loss (MSE + pairwise ranking + GIB term).
- Trained for EPOCHS=3 in provided notebook; uses repo feature dataset

---

4) Cointegration experiments

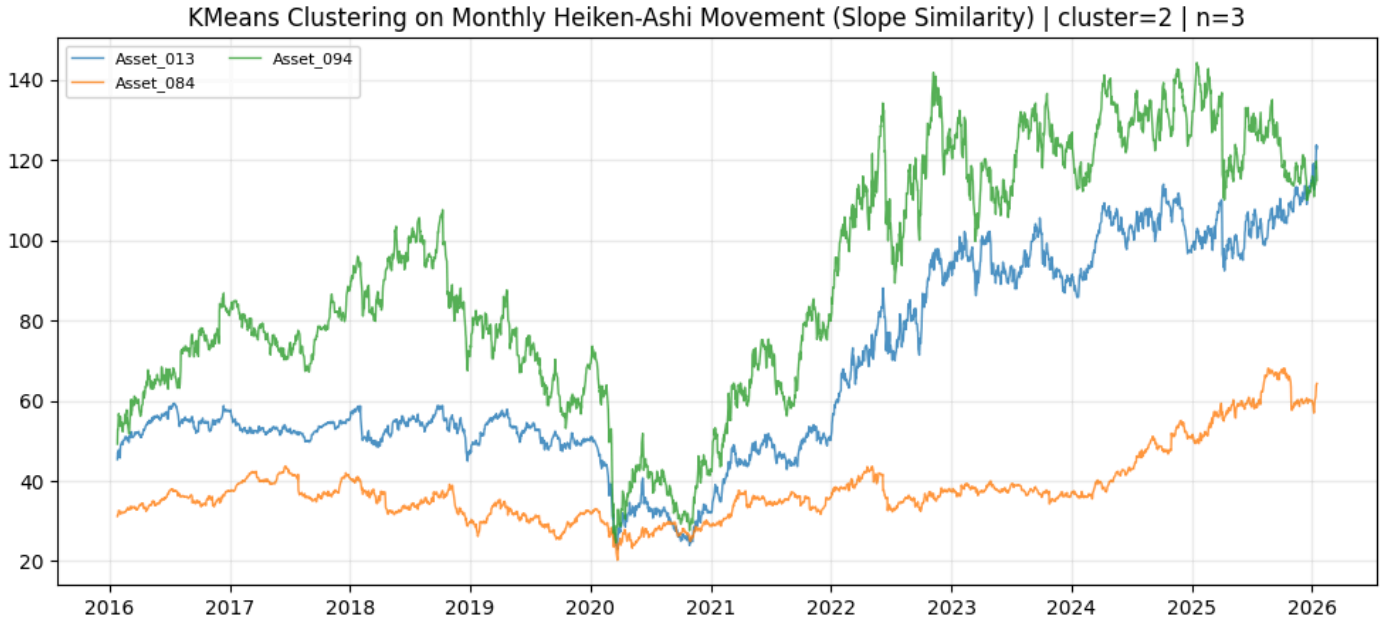
Goals

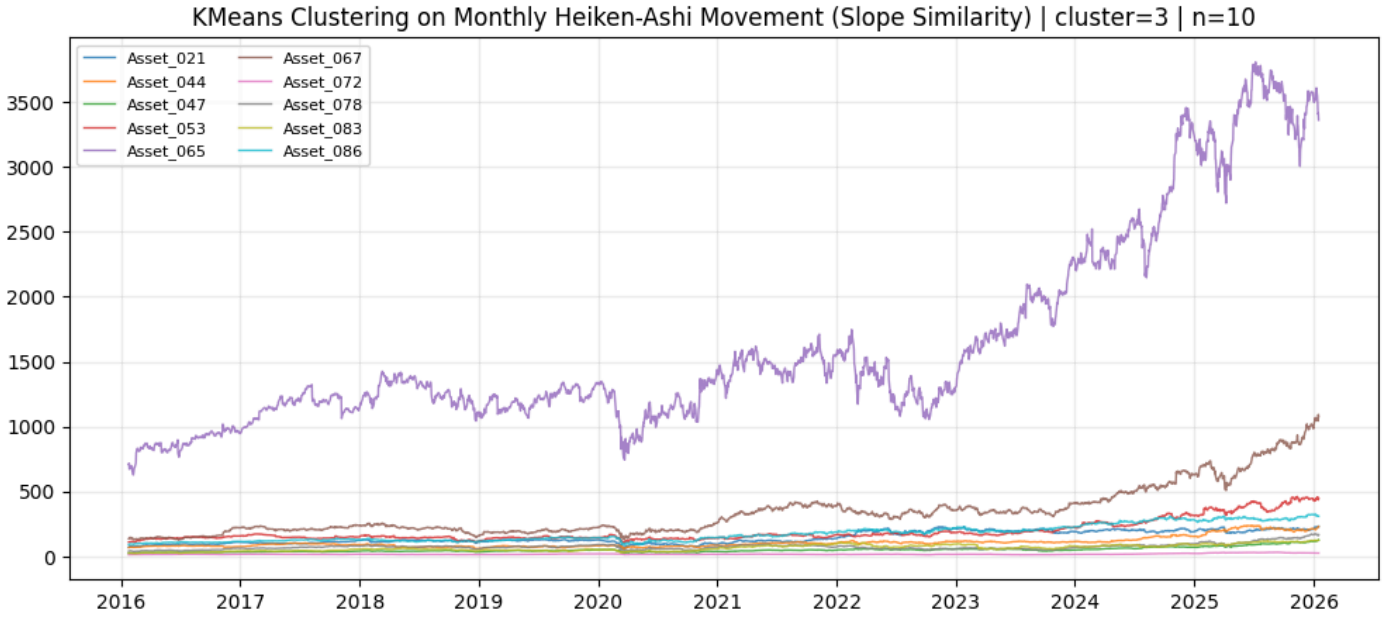
- Find groups of assets that move together (cointegrated or exhibit similar trend/shape) for pair/trade construction, statistical arbitrage, or portfolio construction.
- i think the best implementation was the simplest , just clustering the monthly gain percentage

1. Heiken Ashi monthly slope clustering

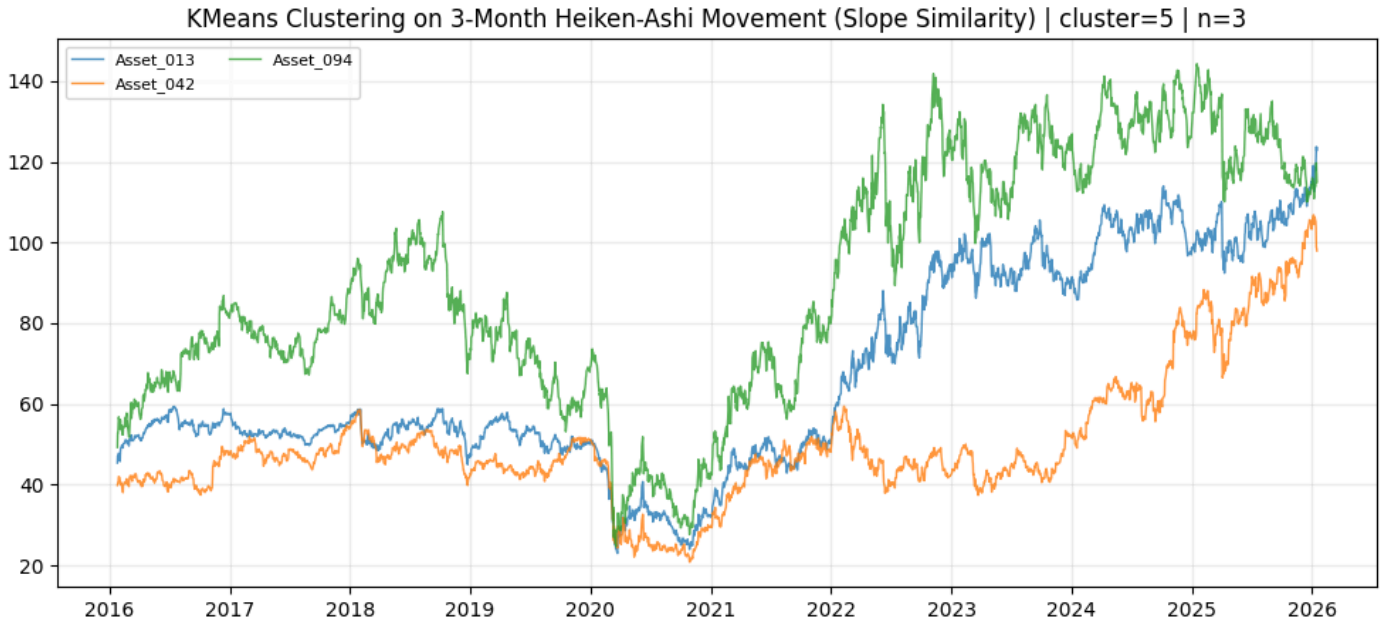
- Transform daily OHLC to Heiken Ashi monthly candles (month-end / custom period), compute per-period slope normalized by period length (slope = delta(close) / days).
- KMeans clustering on slopes across assets — clusters group assets with similar slope direction and profile, not magnitude.
- Notebooks:
  - `notebooks/cointegrated movement/01_kmeans_heiken_ashi_monthly_slope_clusters.ipynb`
  - `02_` (3-month) and `03_` (6-month) variants

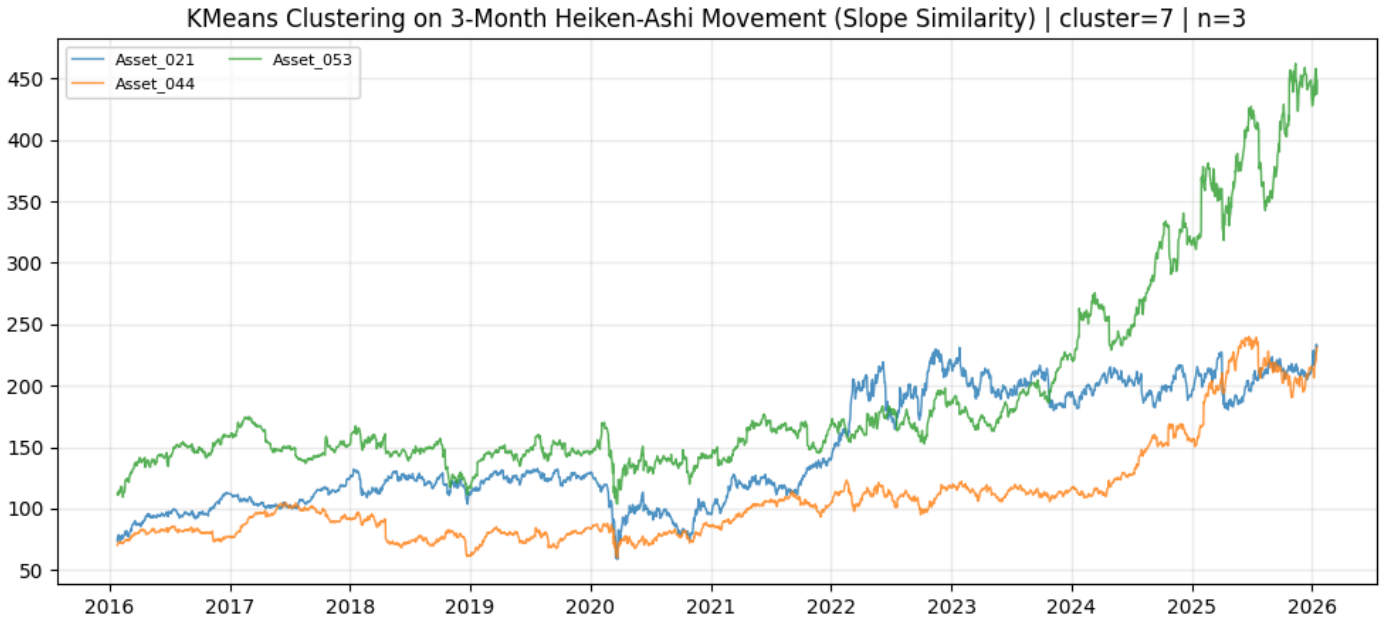
1 month = 1 heiken ashi candle





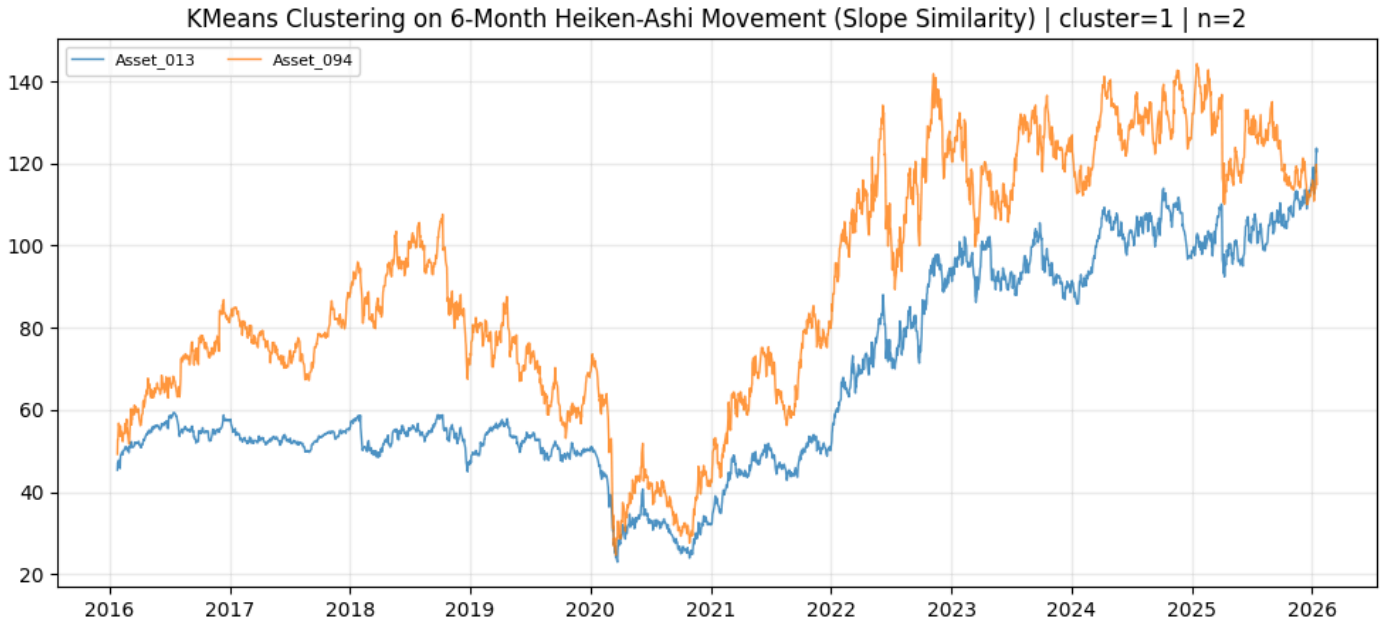
3 months = 1 heiken ashi candle

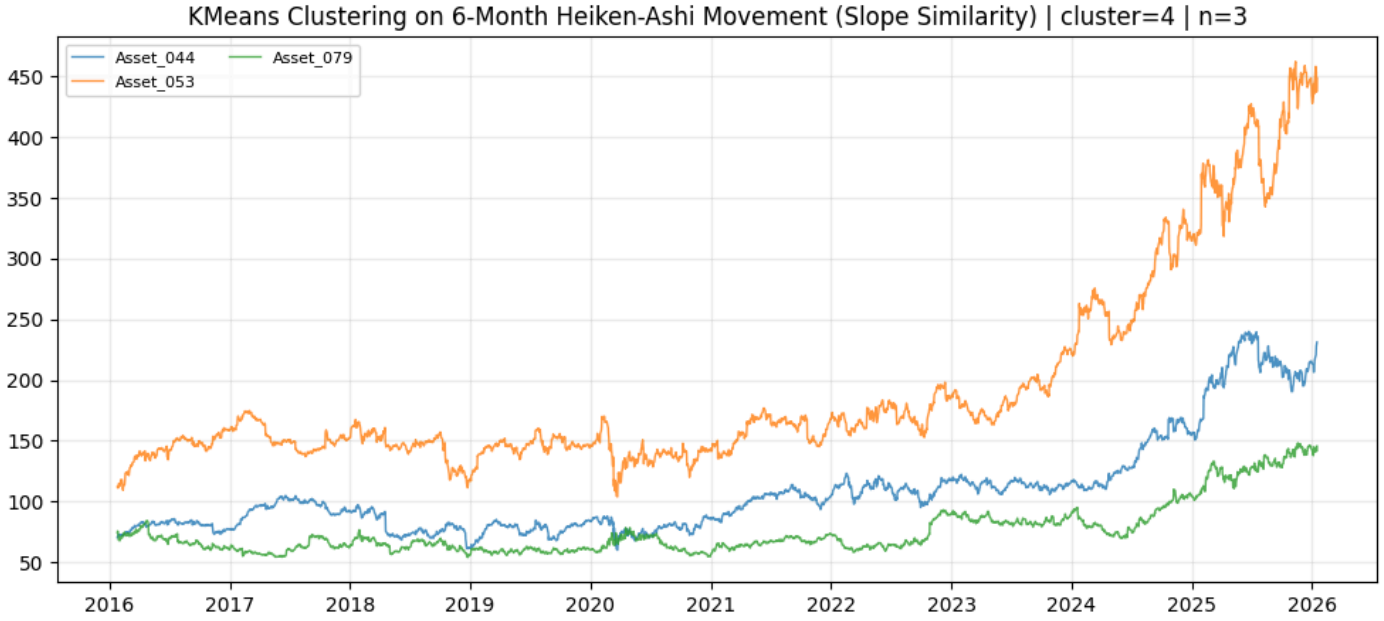




- we have to filter out noisy results where it groups together 10 or more stocks , the results with 2-5 stocks can be considered decently accurate

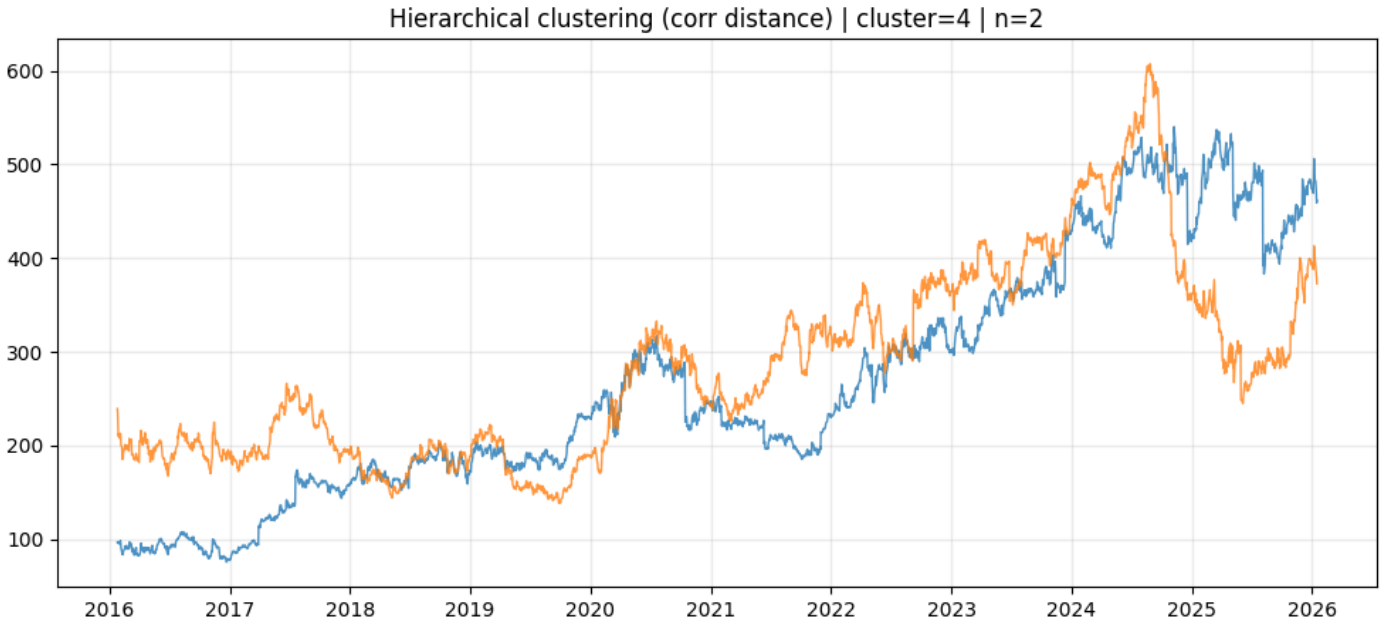
1 candle = 6 months

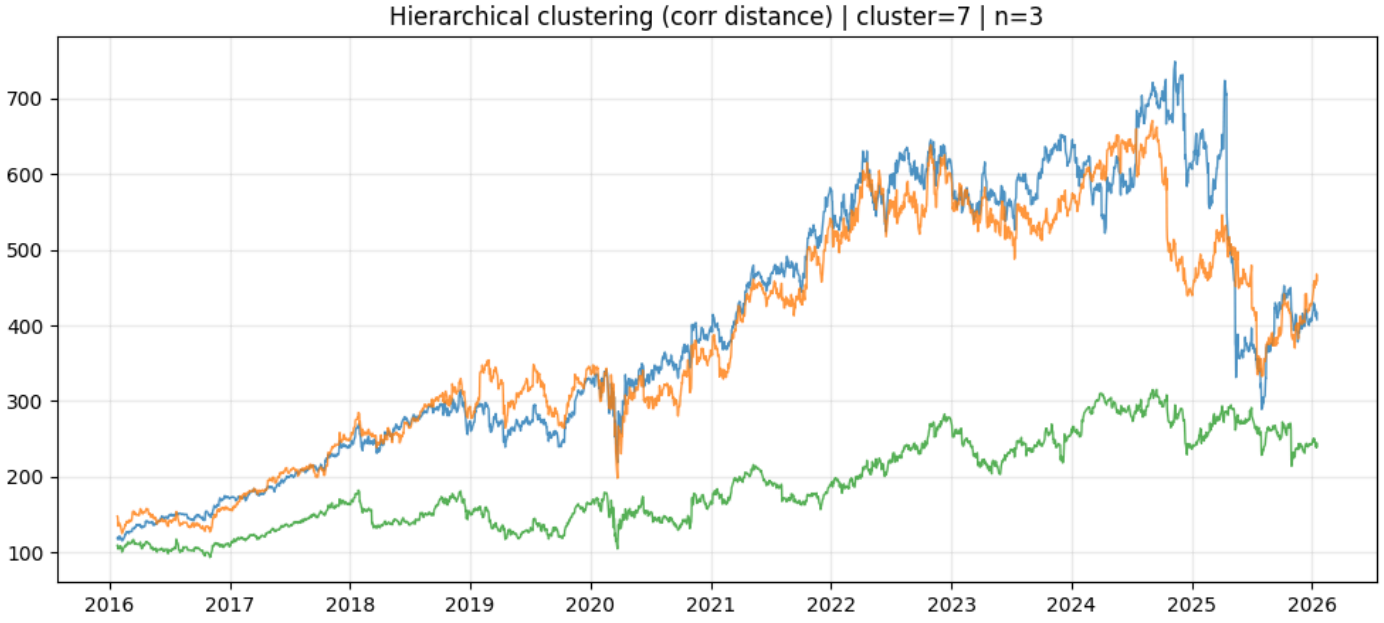


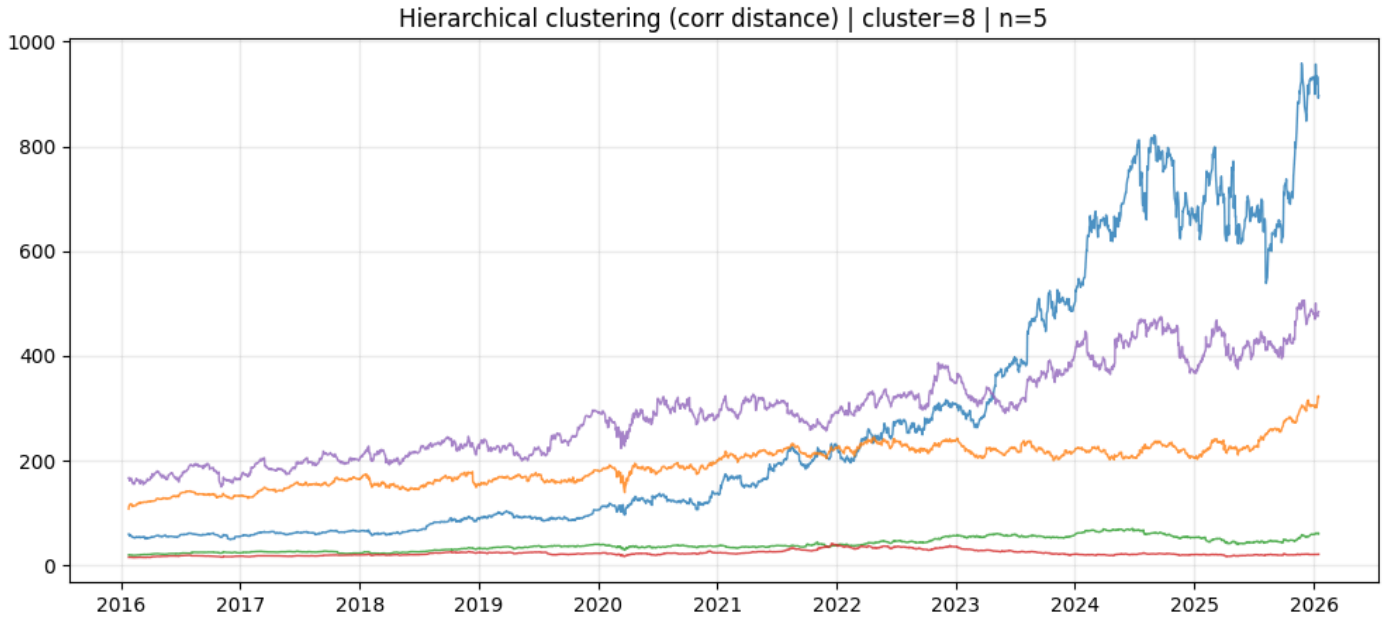


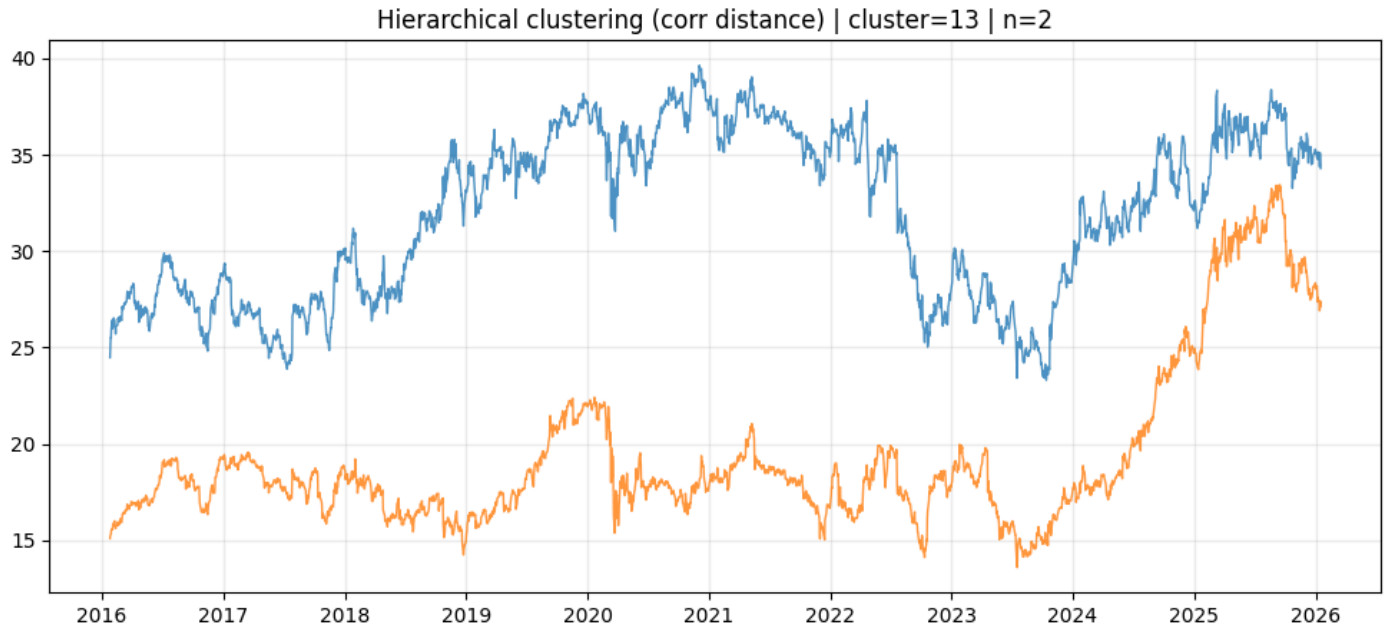
2. Hierarchical clustering

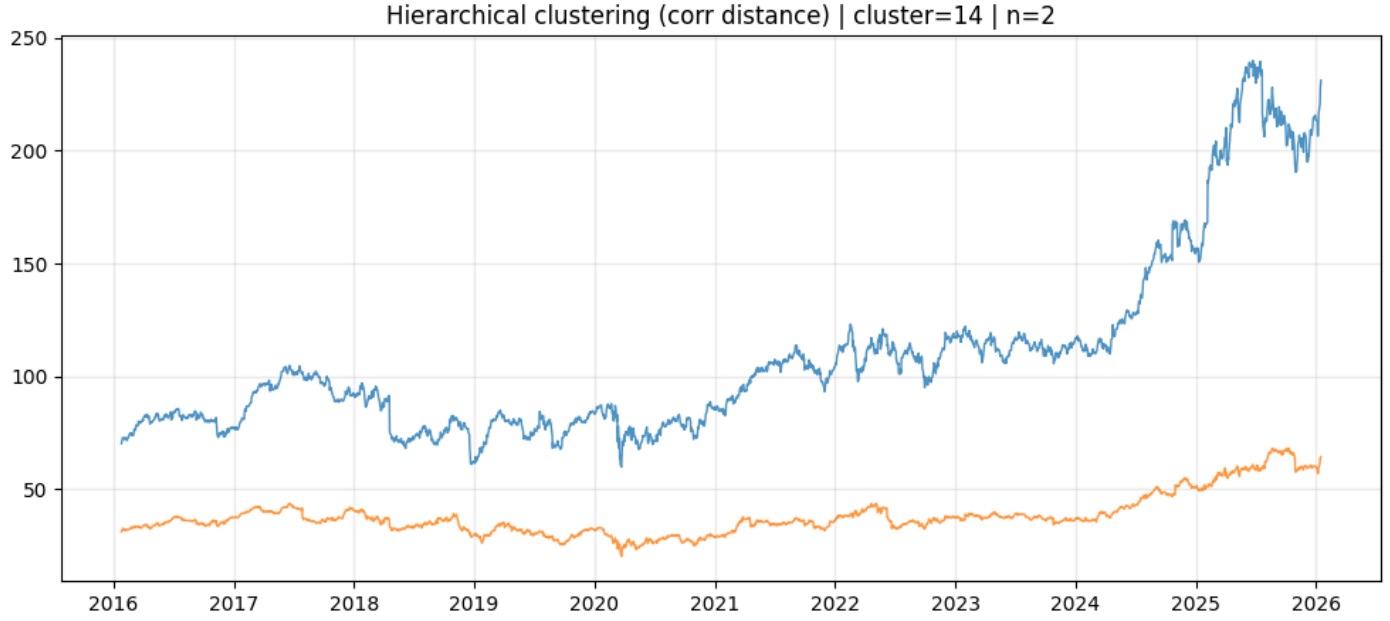
- Build distance matrix based on correlation or shape distance (DTW optional if implemented), apply scipy hierarchical clustering (Ward, average).
- Visualize dendrogram and cut at desired threshold to produce groups.

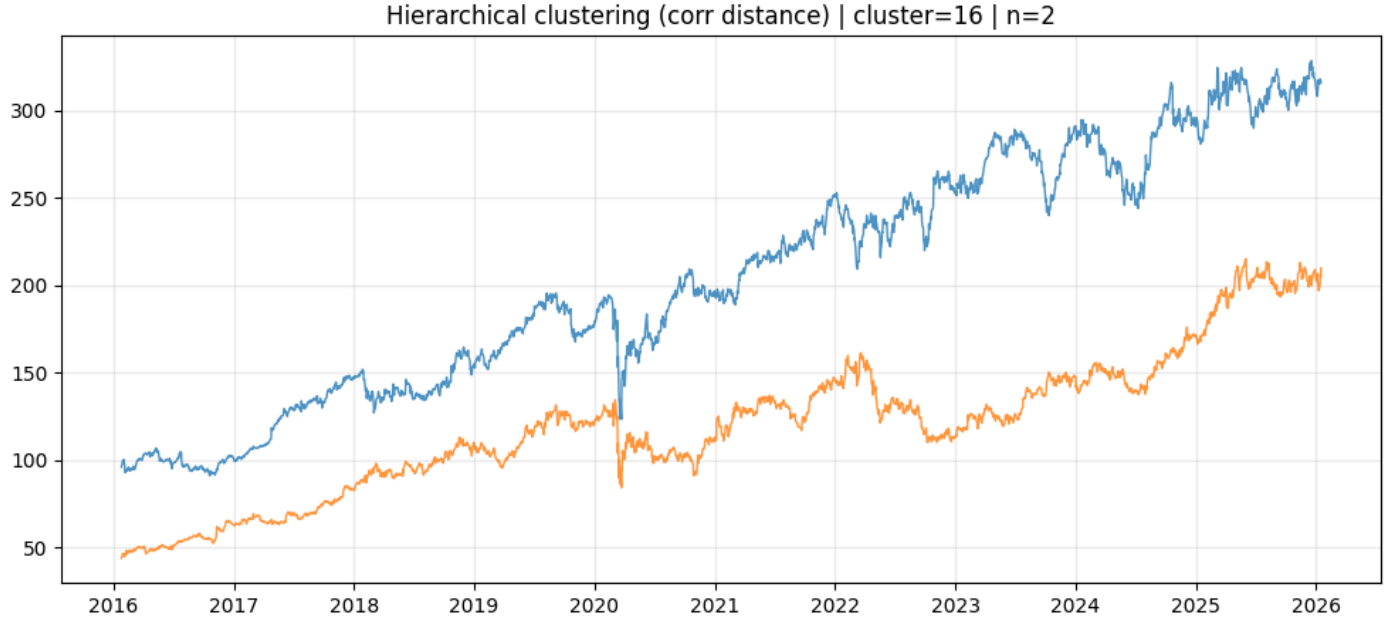

Hierarchical clustering (corr distance) | cluster=4 | n=2
600
500
400
300
200
100
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026

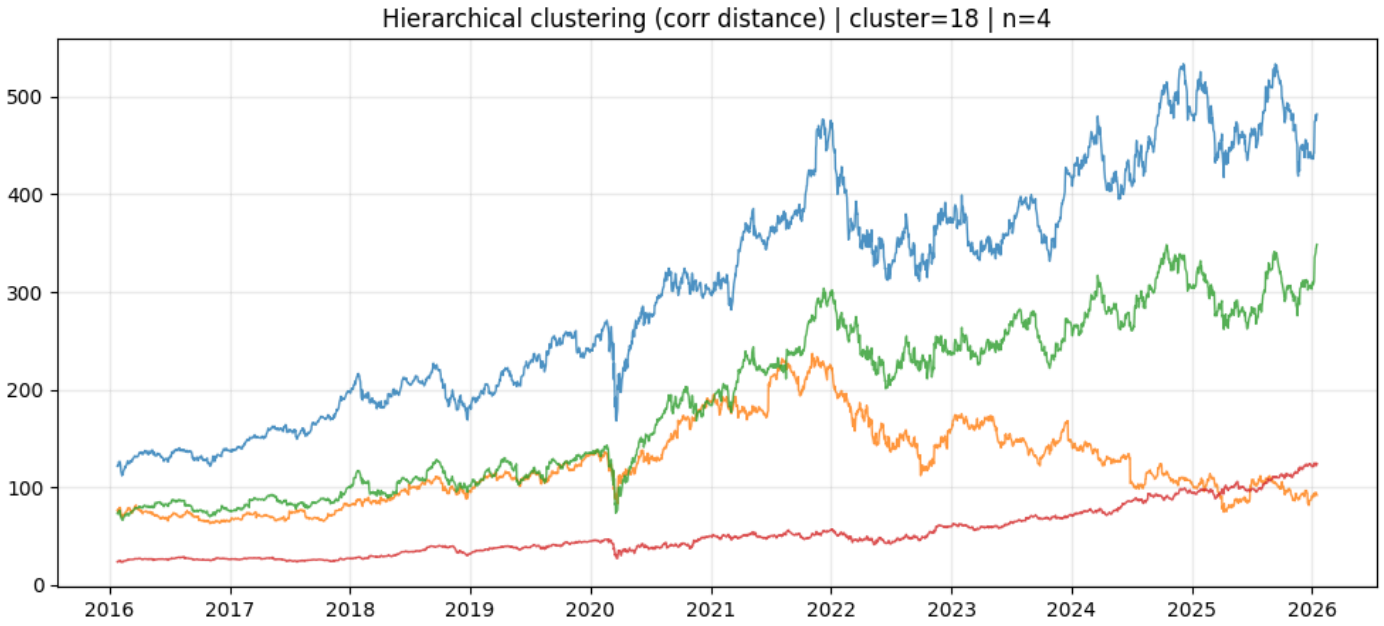

Hierarchical clustering (corr distance) | cluster=7 | n=3
700
600
500
400
300
200
100
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026


Hierarchical clustering (corr distance) | cluster=8 | n=5
1000
800
600
400
200
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026


Hierarchical clustering (corr distance) | cluster=13 | n=2
40
35
30
25
20
15
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026

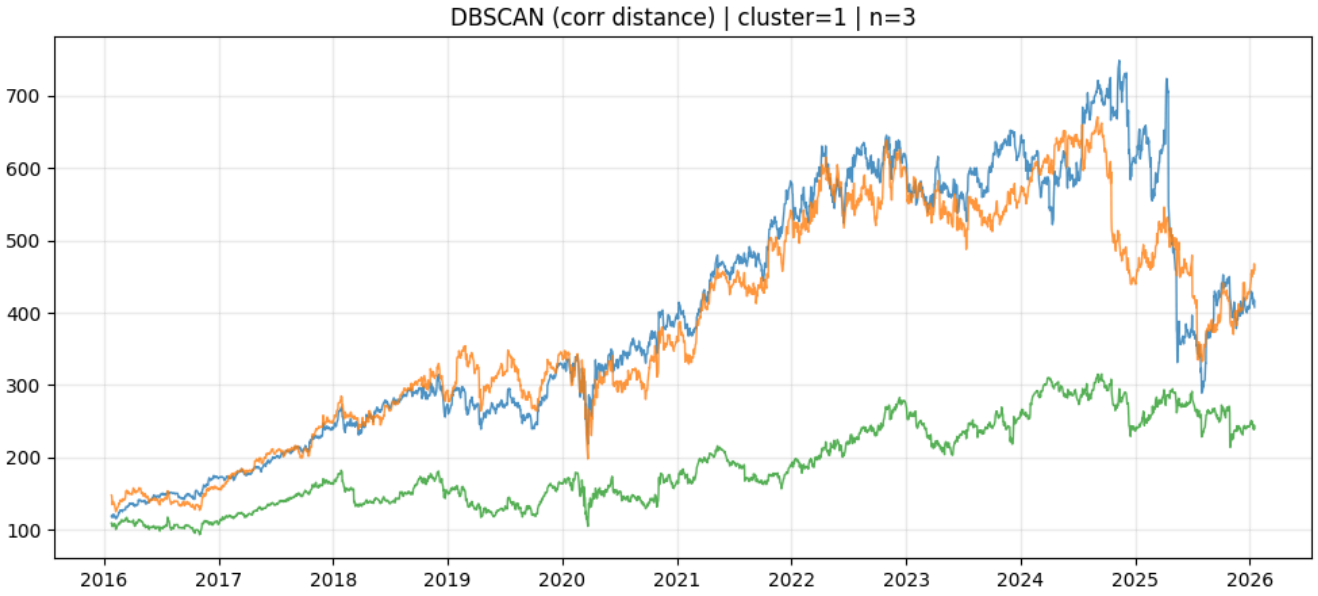

Hierarchical clustering (corr distance) | cluster=14 | n=2
250
200
150
100
50
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026

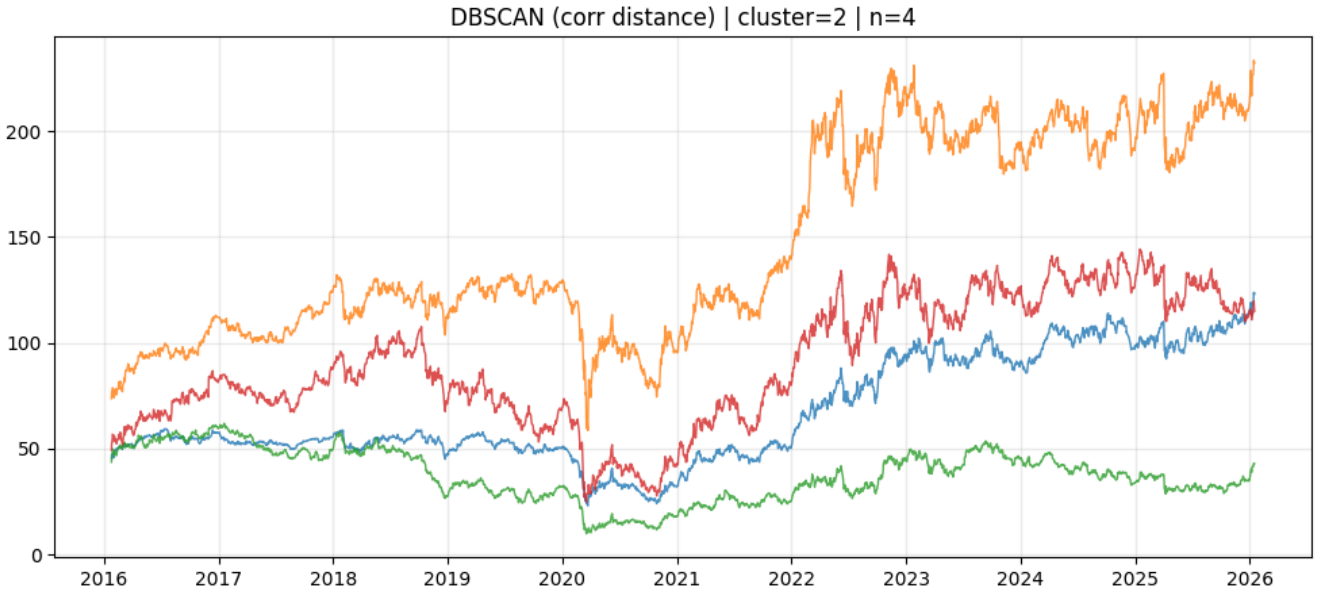

Hierarchical clustering (corr distance) | cluster=16 | n=2
300
250
200
150
100
50
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026

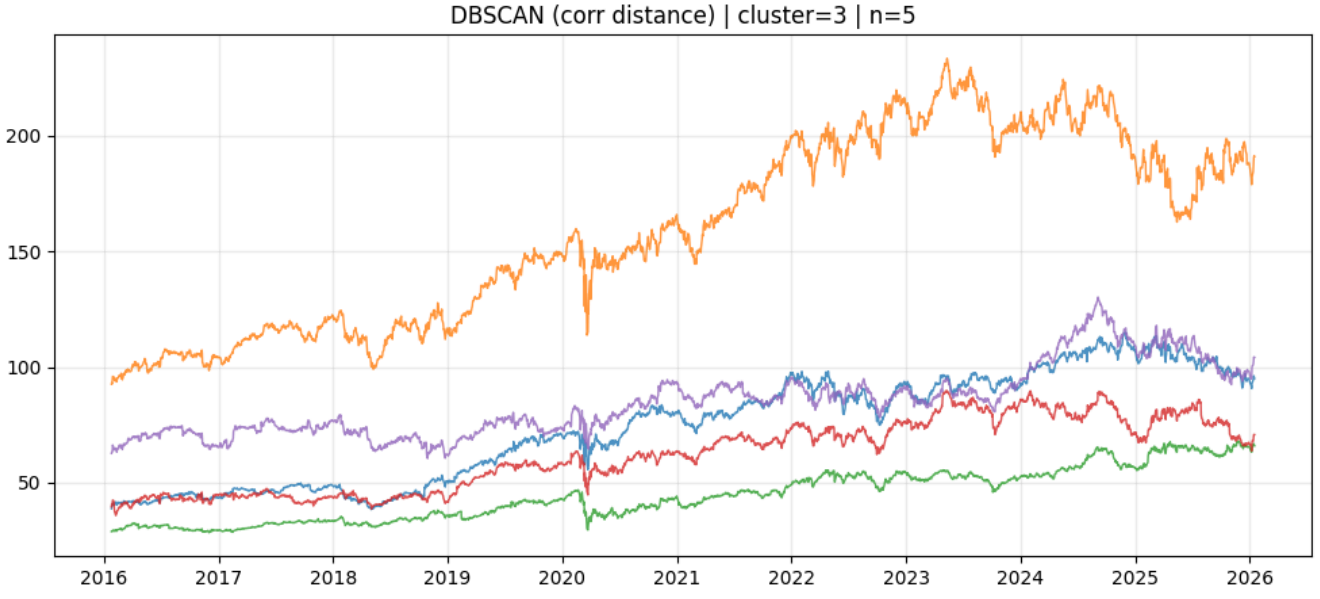


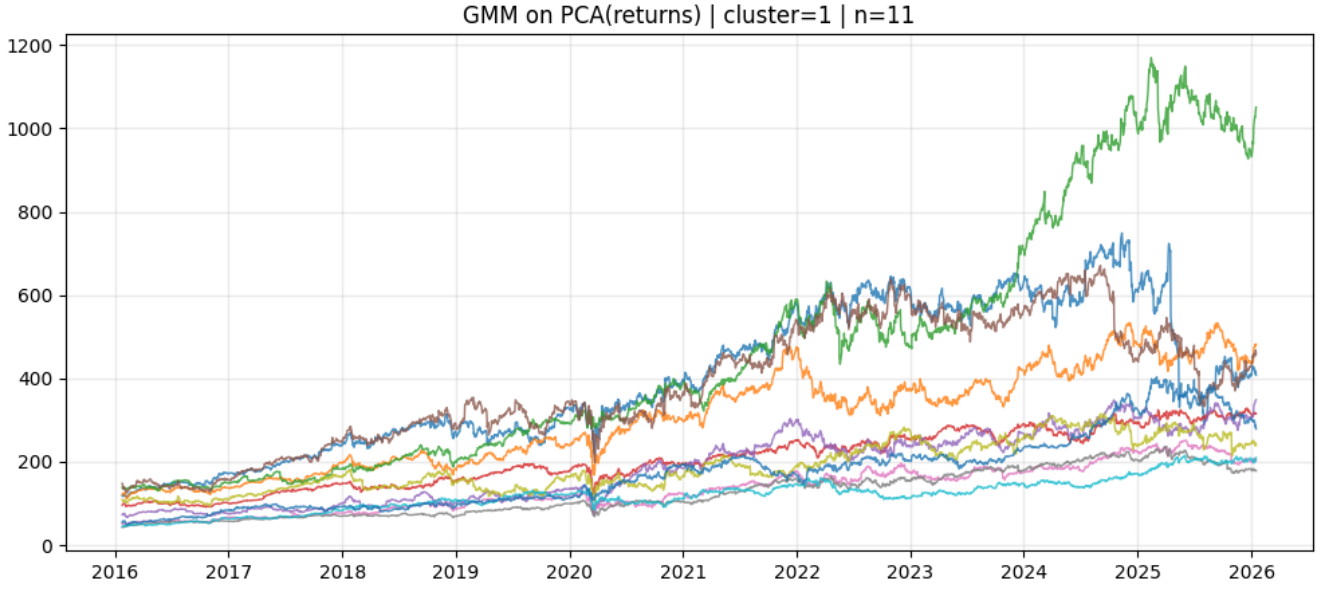
3. Density-based & mixture models

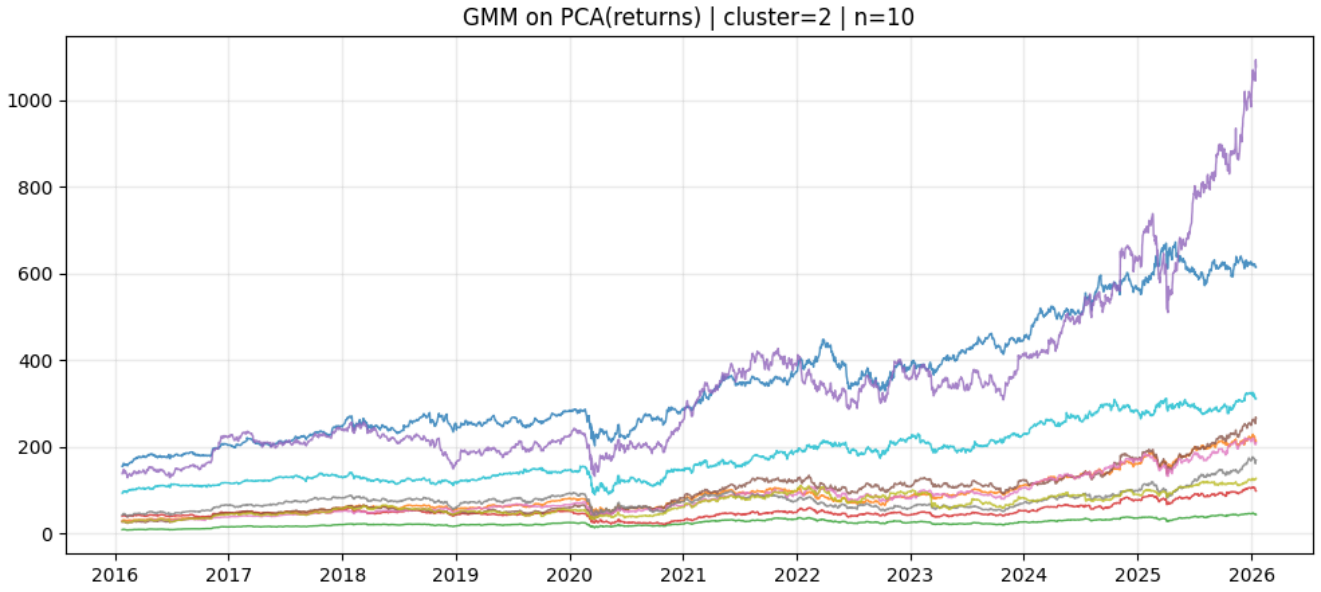
- DBSCAN: clusters dense regions in slope/feature space to isolate similar movers.
- Gaussian Mixture Models (GMM): fit mixtures and assign posterior cluster responsibilities.

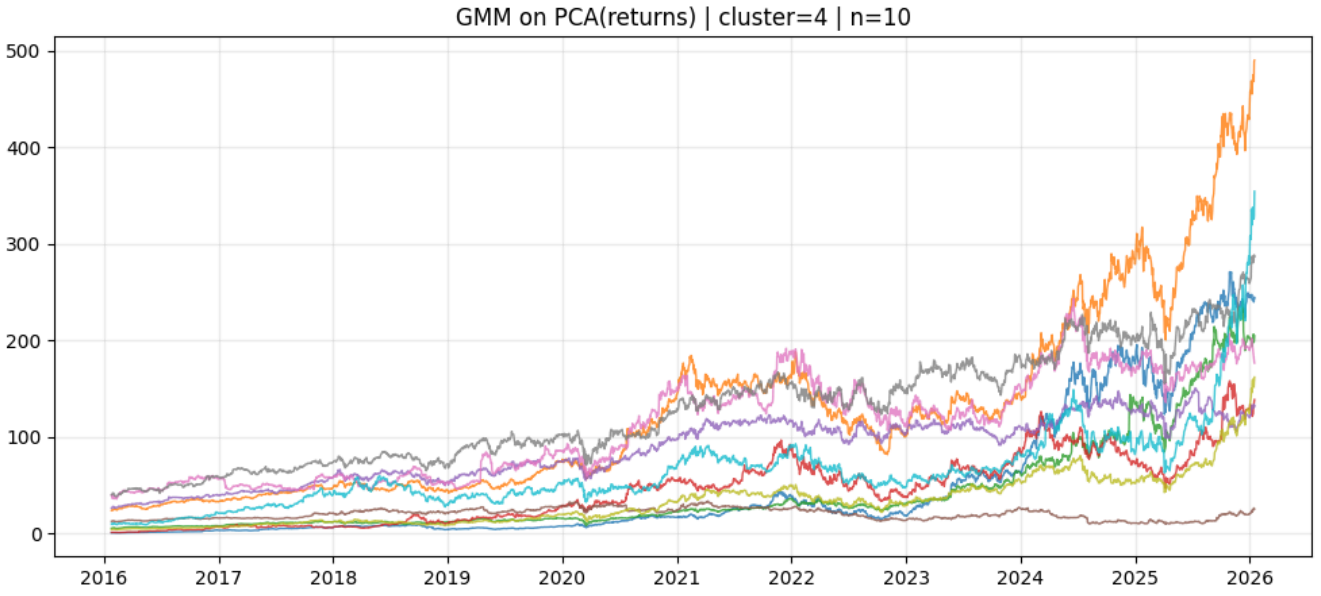


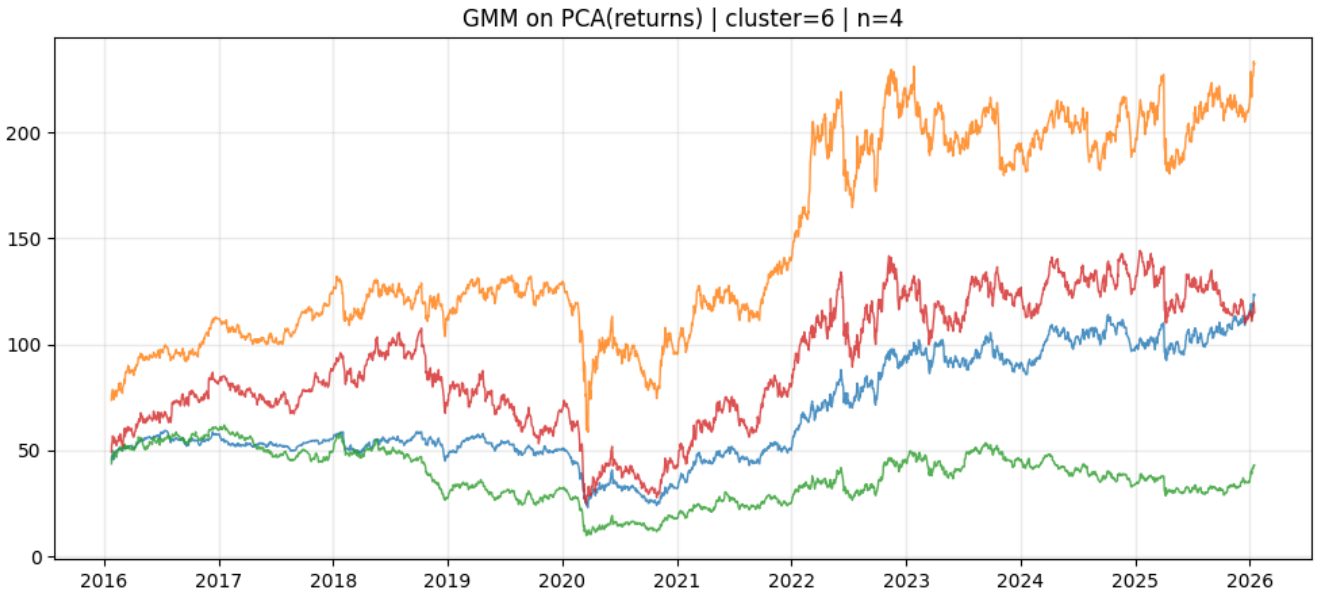

DBSCAN (corr distance) | cluster=2 | n=4
200
150
100
50
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026


DBSCAN (corr distance) | cluster=3 | n=5
200
150
100
50
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026

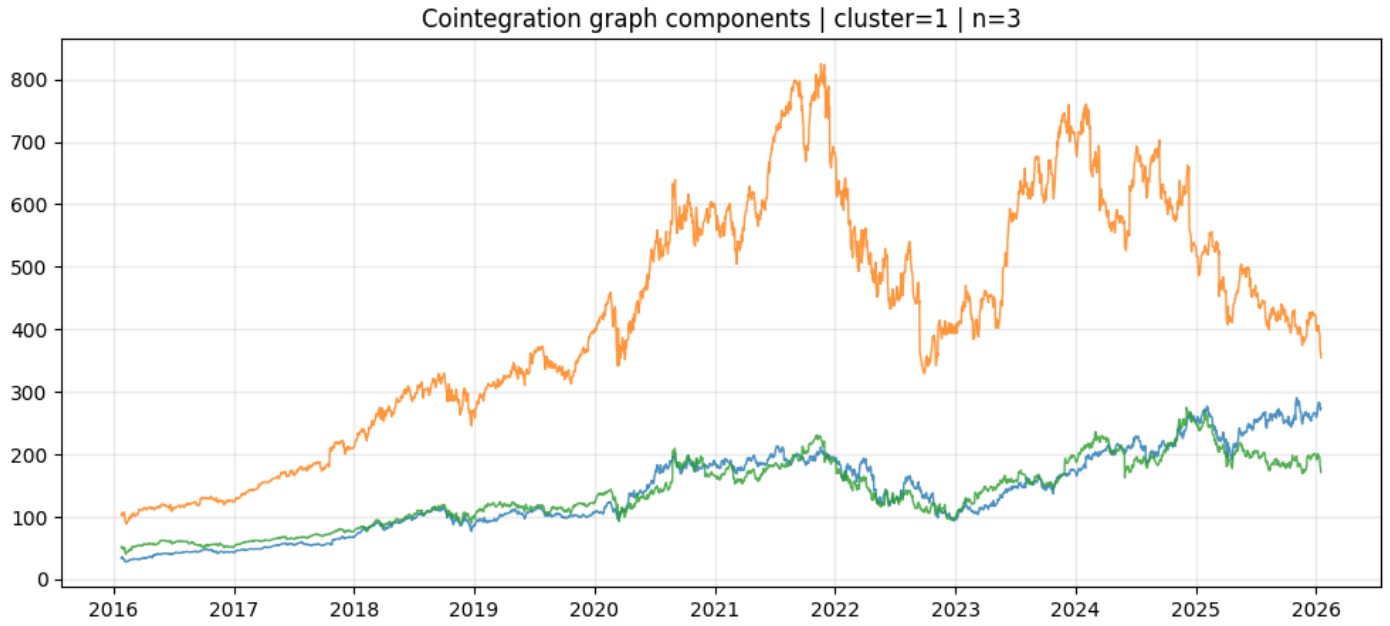

GMM on PCA(returns) | cluster=1 | n=11
1200
1000
800
600
400
200
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026

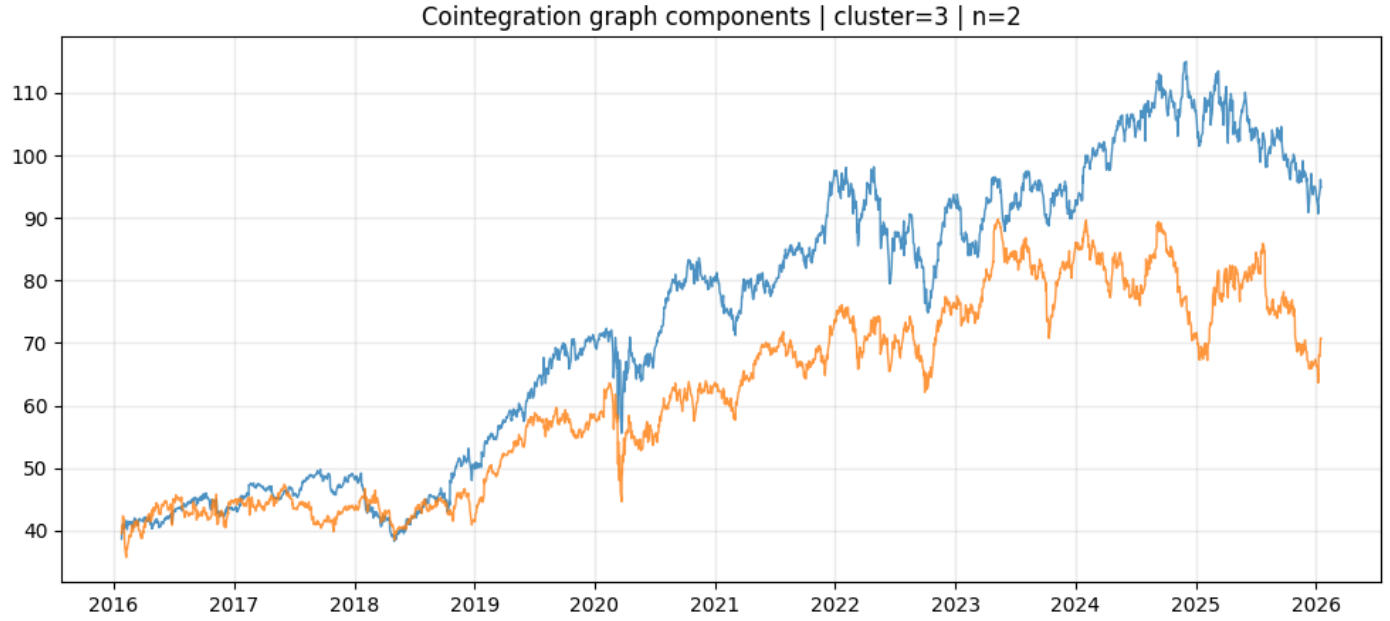






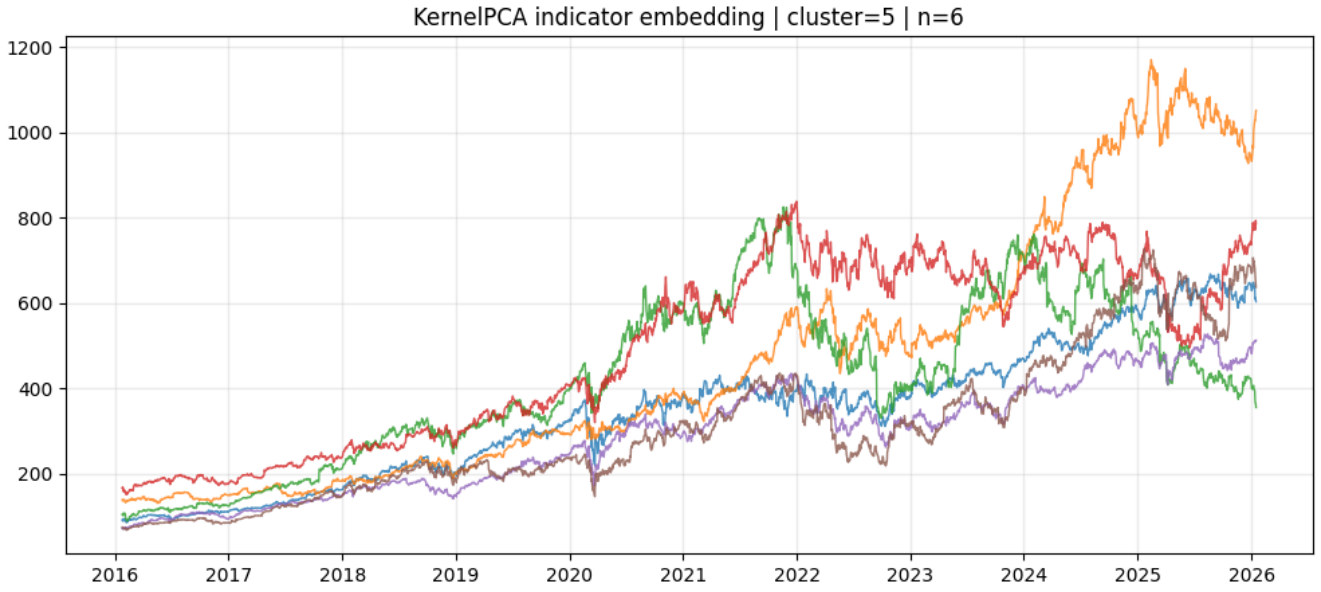
Cointegrated Graph components

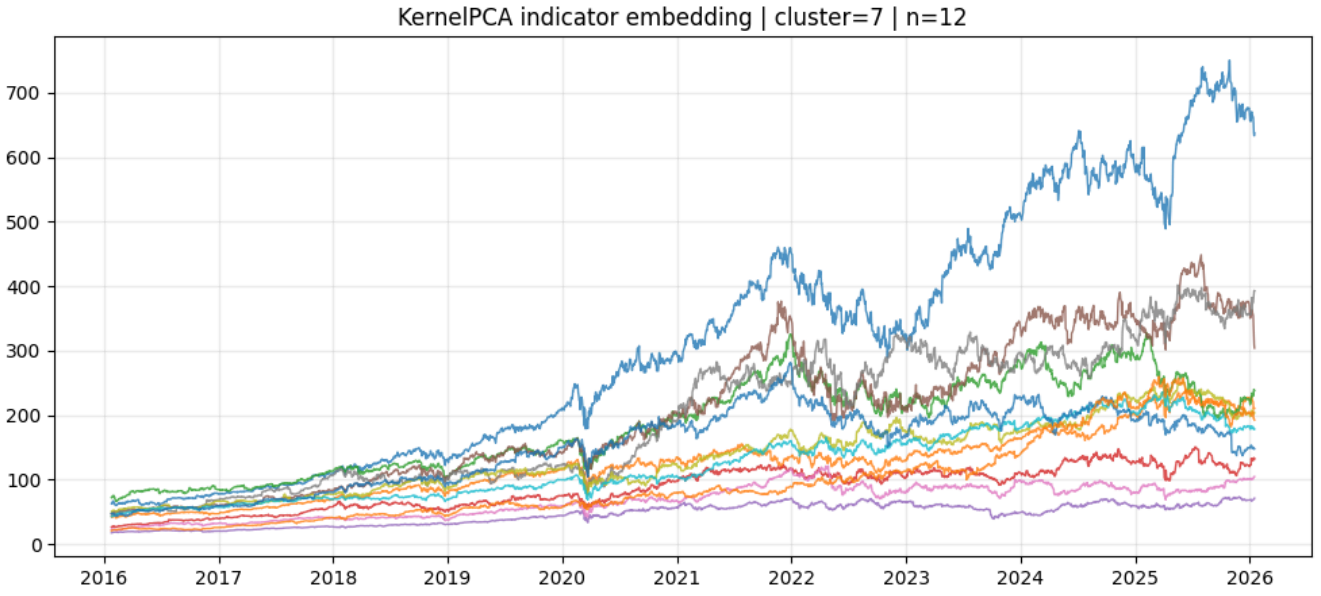




4. Kernel PCA on indicators

- KernelPCA for nonlinear dimensionality reduction of indicator space; cluster in reduced space to find indicators that co-move assets.
- Notebook: `notebooks/cointegrated movement/05_kernel_pca_indicators_comovement.ipynb`

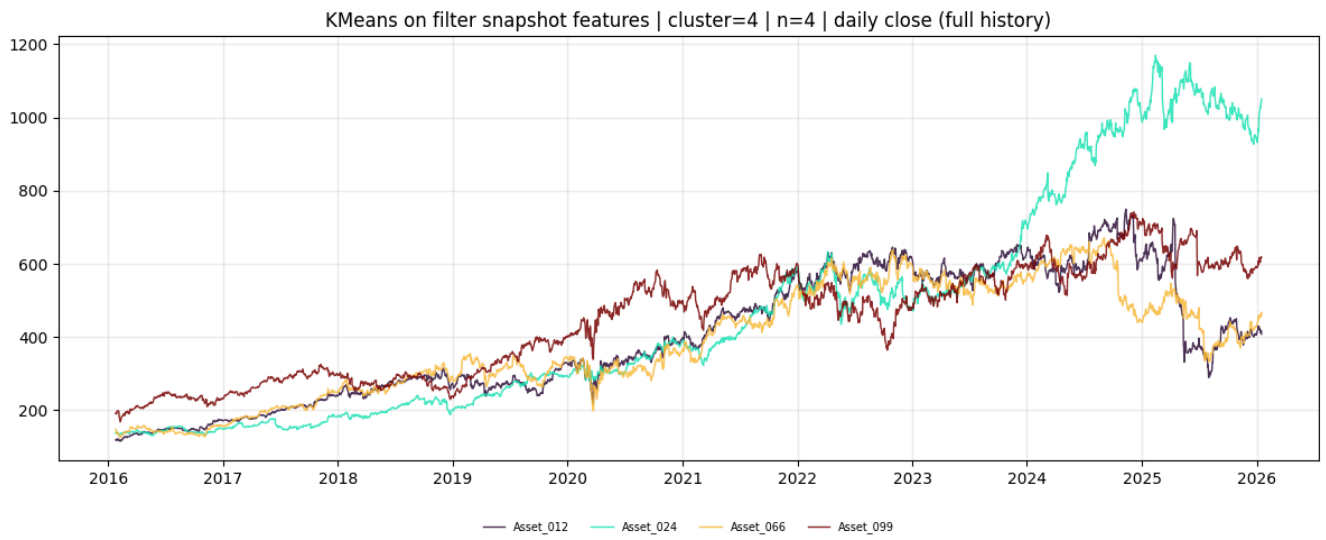


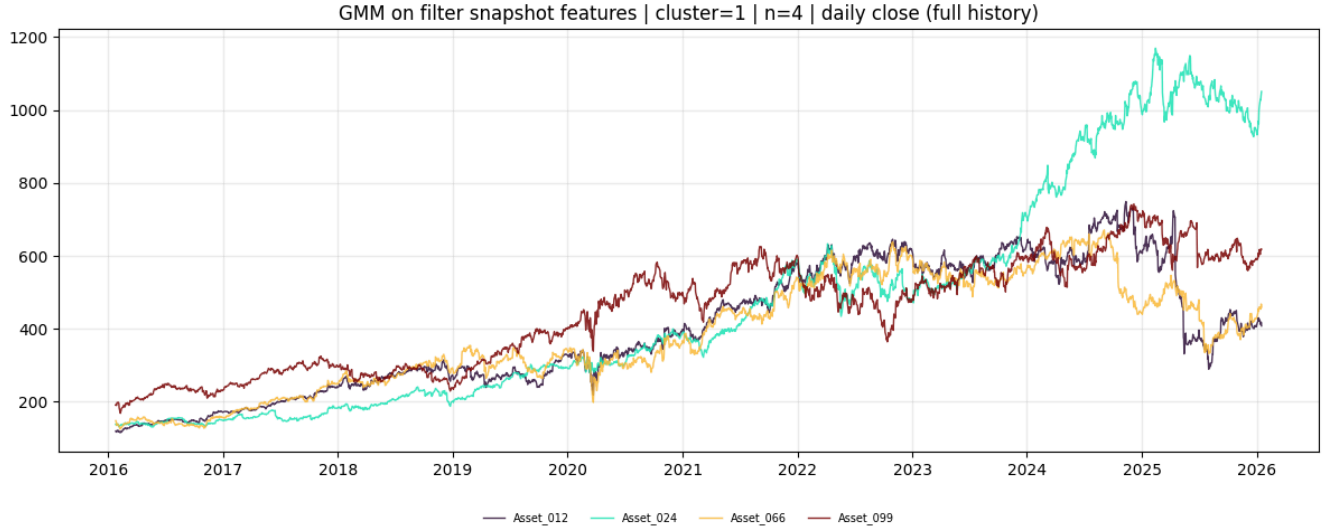


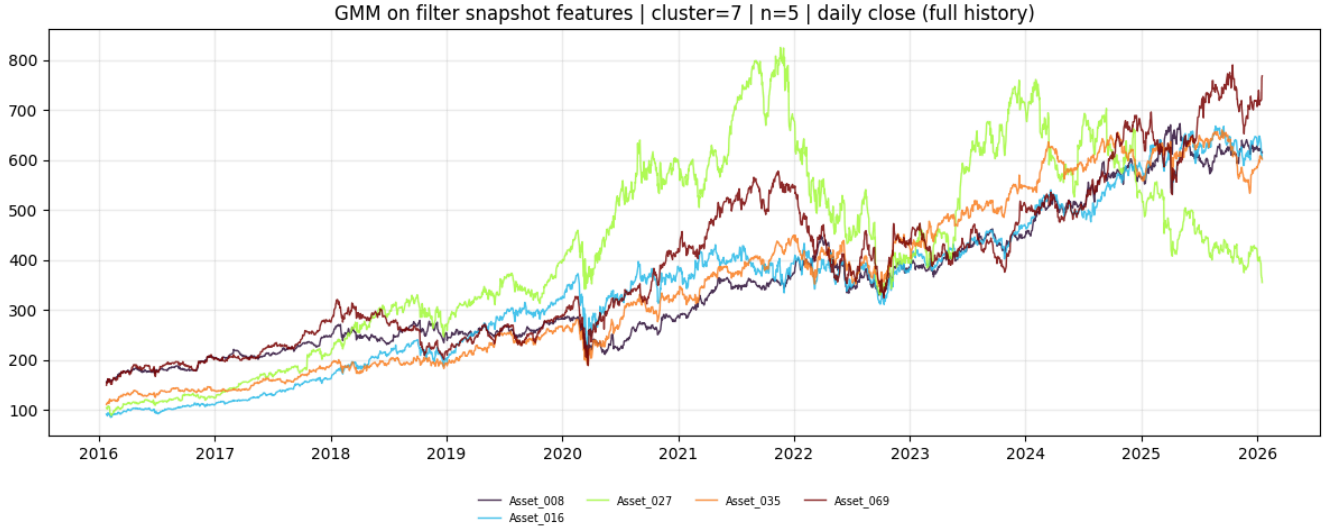
- very messy , not as good as the other methods

5. Kalman/Wiener/SavGol/SMURF snapshots + clustering

- Snapshot features (signal value, slope, residual) per asset per timestamp; KMeans/DBSCAN/GMM clustering to find potential cointegrated groups.
- Notebook: `notebooks/cointegrated movement/07_filter_snapshot_clustering.ipynb`



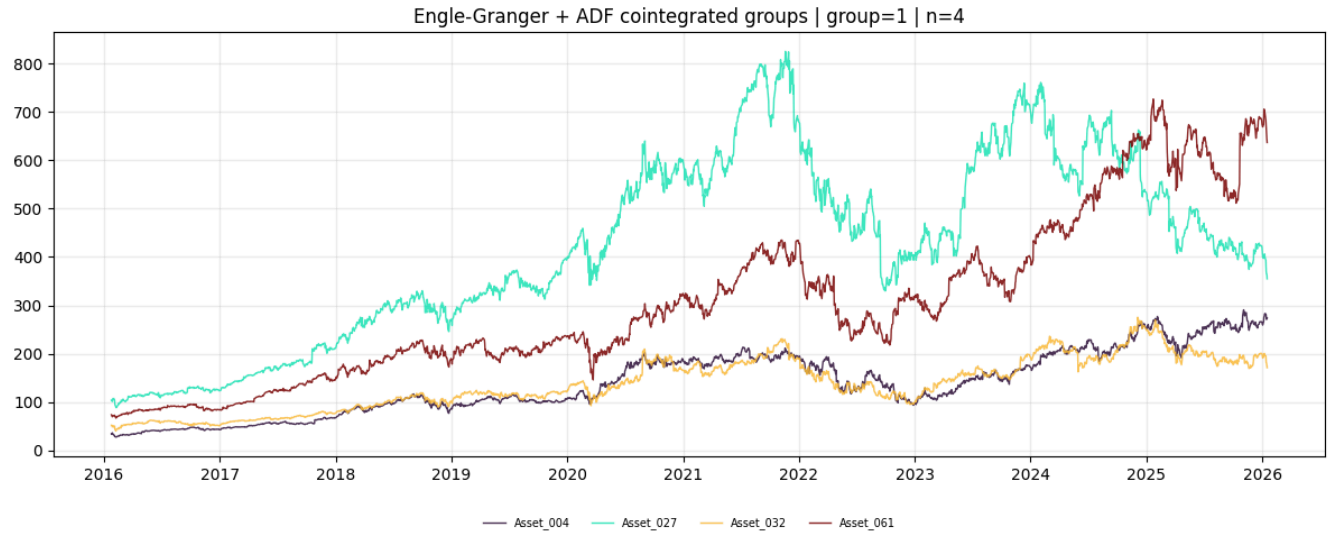




- again very messy

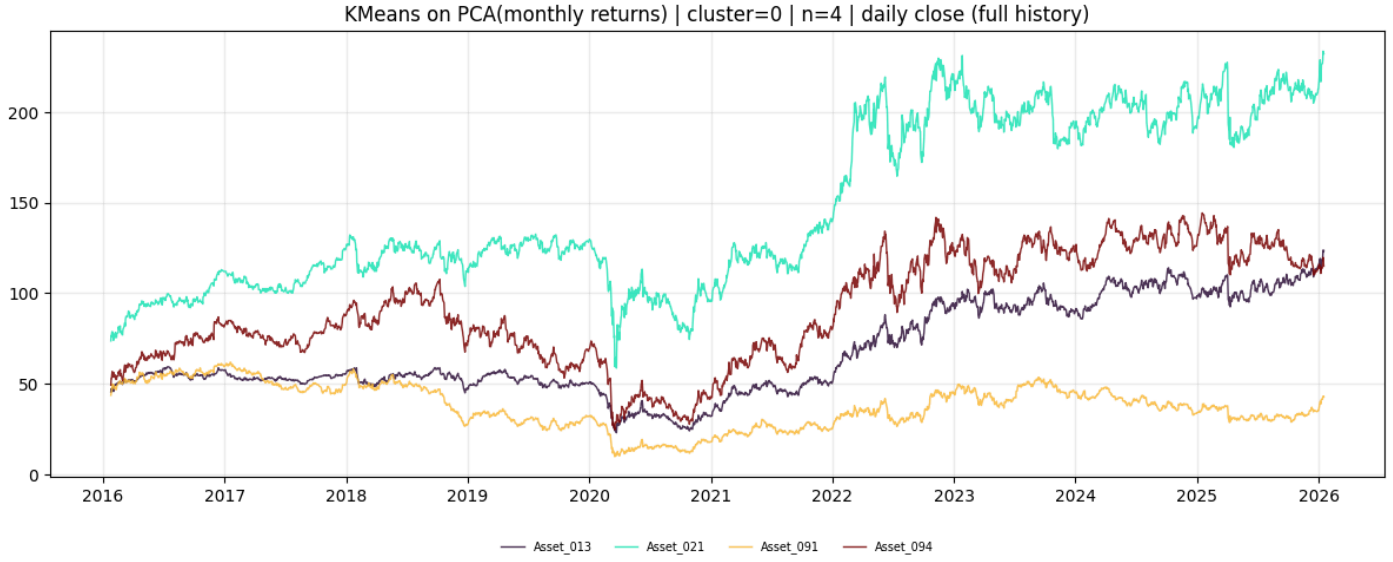
6. Engle-Granger (pairwise cointegration) + ADF residual test

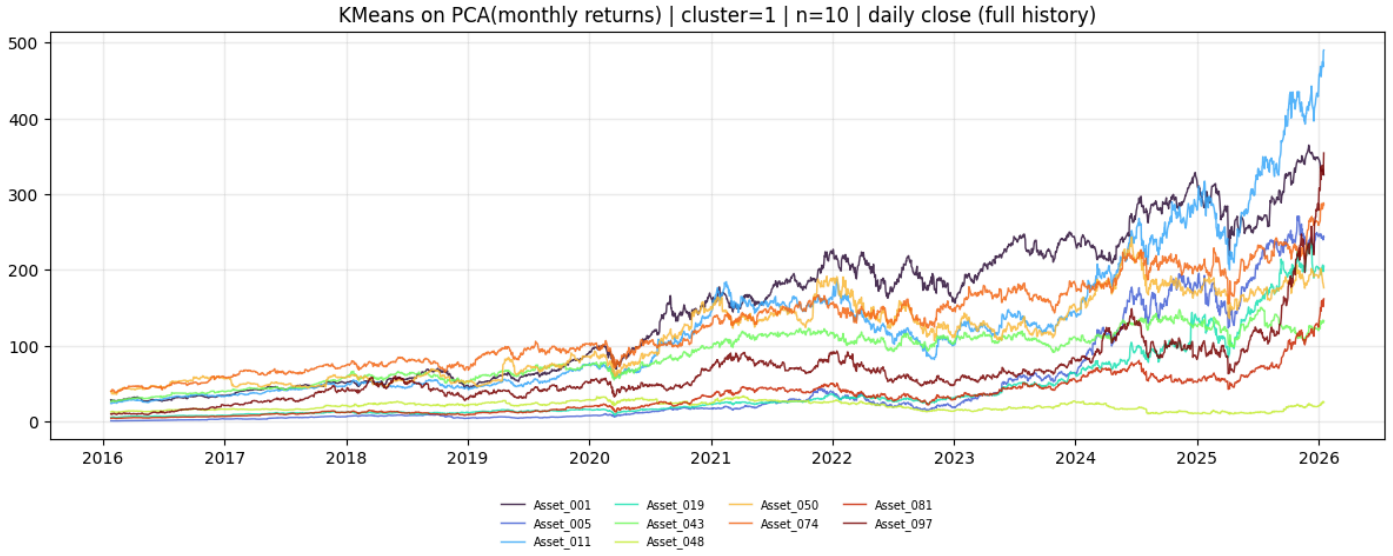
- For candidate pairs/groups, run Engle-Granger cointegration: regress series A on B to get residuals, test residuals with Augmented Dickey-Fuller (ADF) — stationarity implies cointegration.
- Notebook: `notebooks/cointegrated movement/08_engle_granger_adf_cointegration.ipynb`

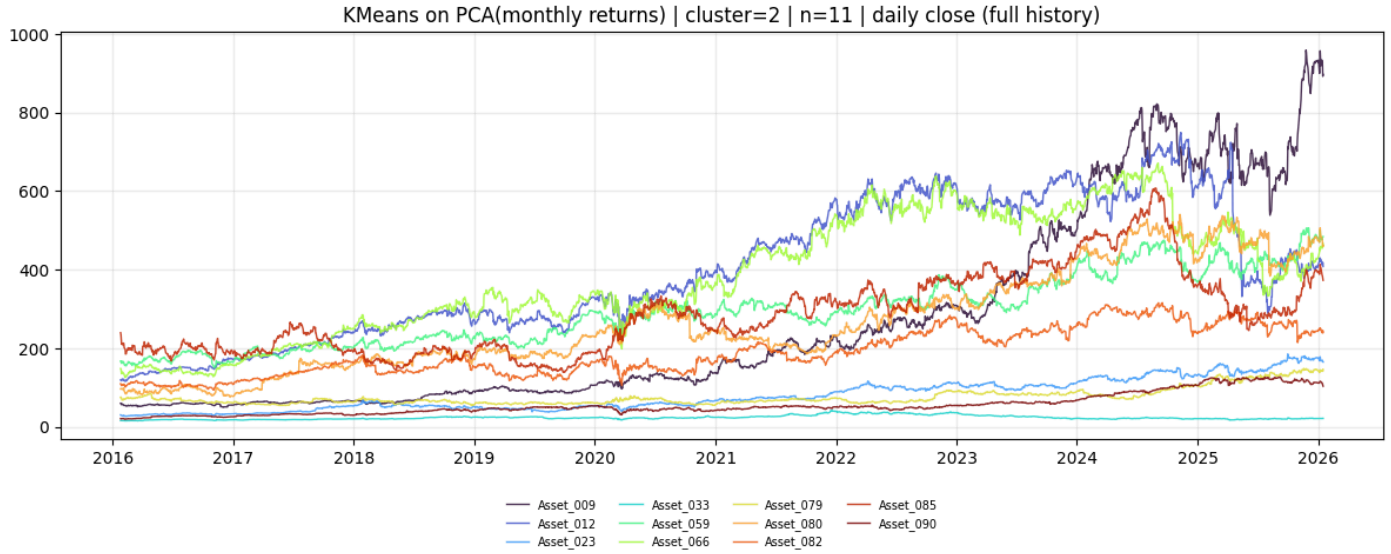


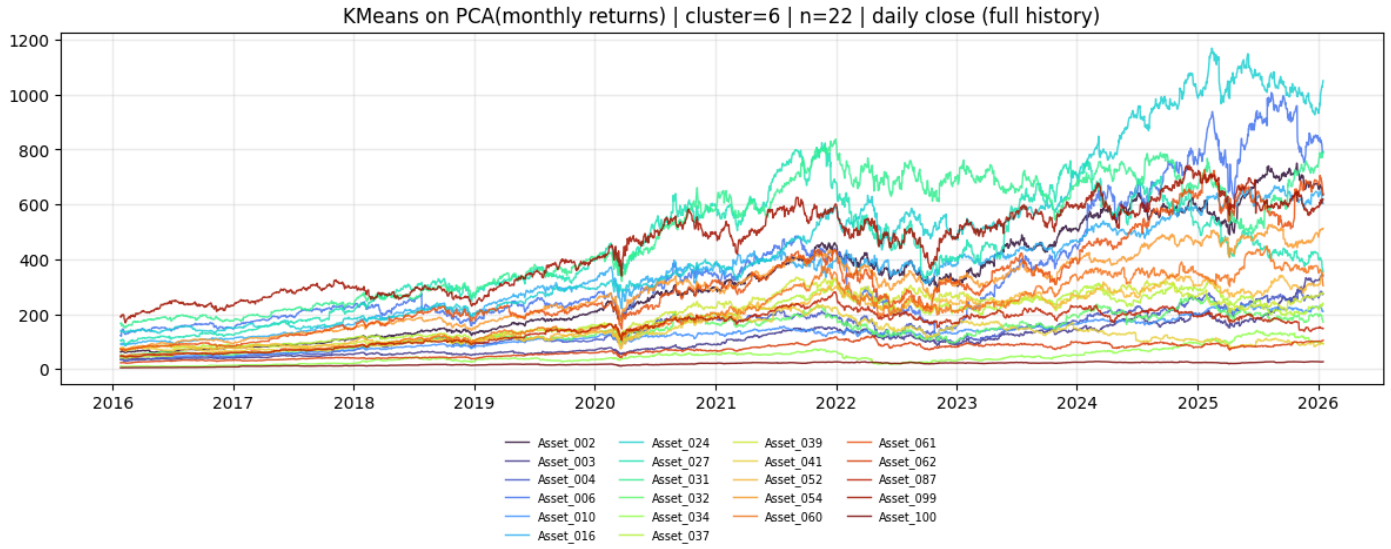
7. Monthly % Gain Clustering

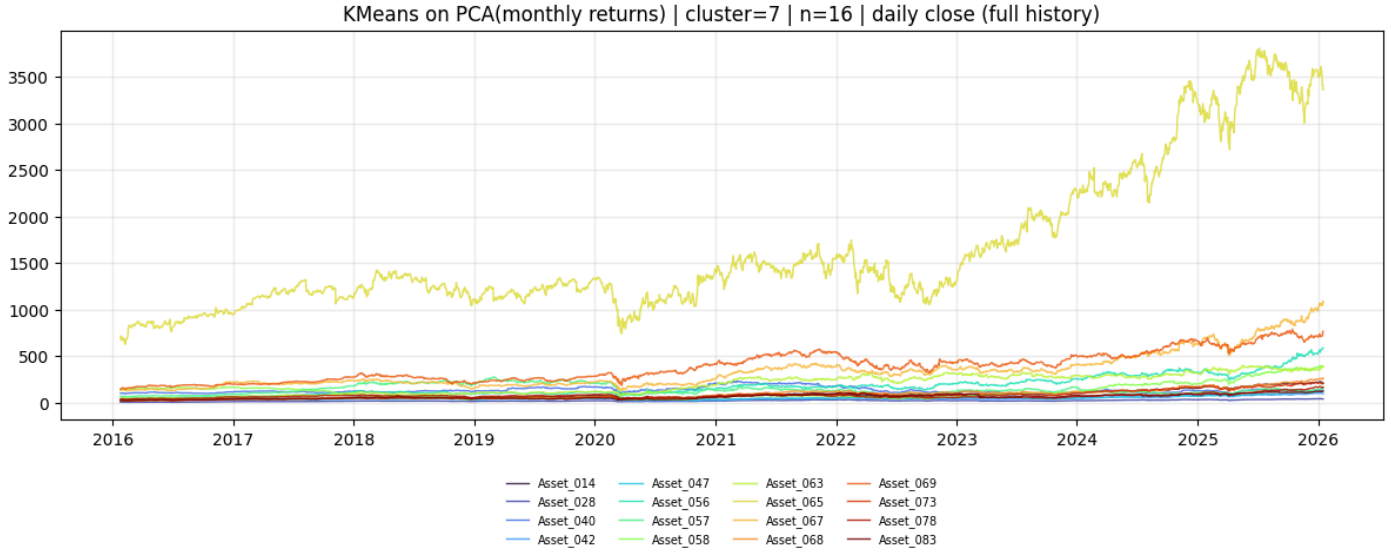
- Take the data of time , monthly percentage gain and cluster the data to see the relationships

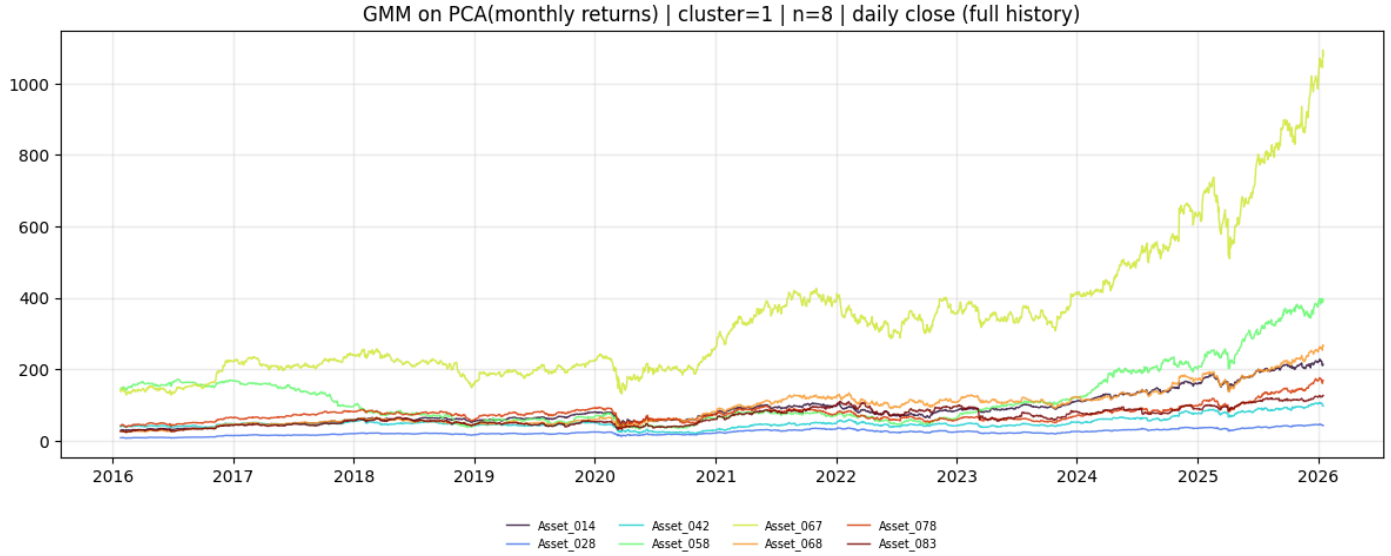


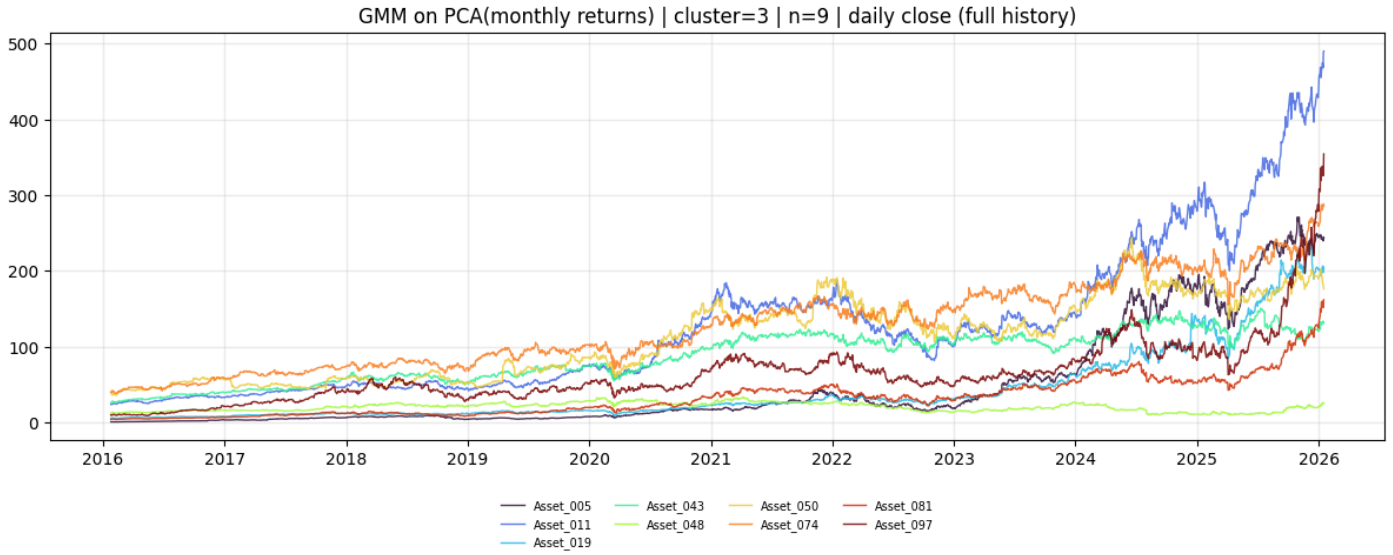


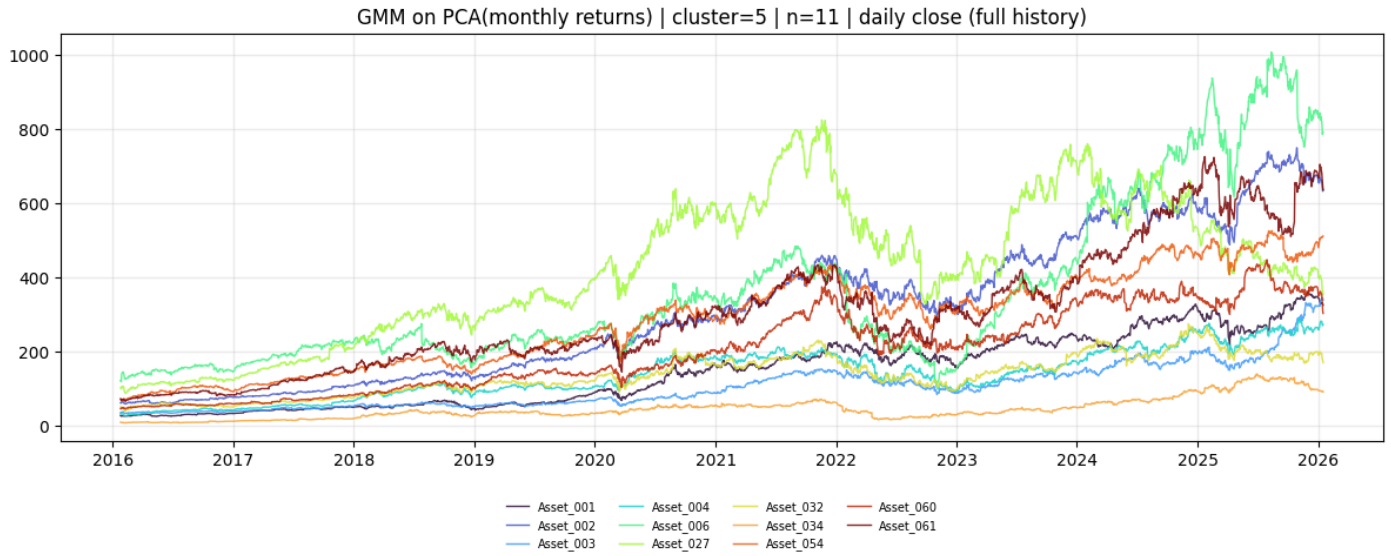

KMeans on PCA(monthly returns) | cluster=2 | n=11 | daily close (full history)
1000
800
600
400
200
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
Asset_009
Asset_012
Asset_023
Asset_033
Asset_059
Asset_066
Asset_079
Asset_080
Asset_082
Asset_085
Asset_090

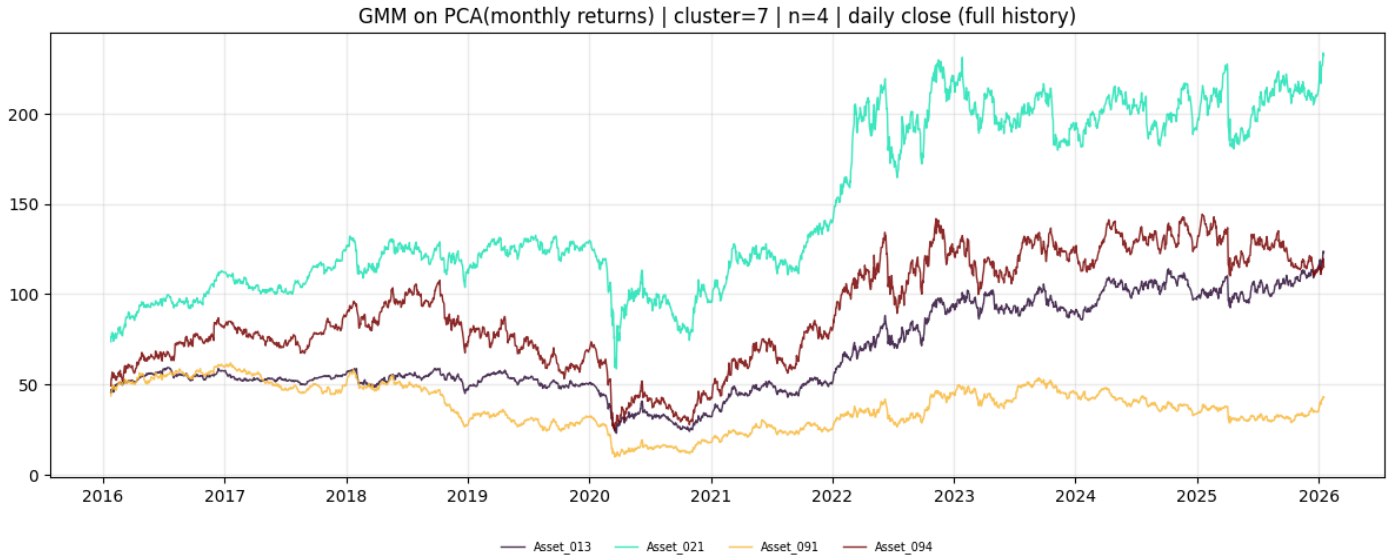

KMeans on PCA(monthly returns) | cluster=6 | n=22 | daily close (full history)
1200
1000
800
600
400
200
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
Asset_002
Asset_003
Asset_004
Asset_006
Asset_010
Asset_016
Asset_024
Asset_027
Asset_031
Asset_032
Asset_034
Asset_037
Asset_039
Asset_041
Asset_052
Asset_054
Asset_060
Asset_061
Asset_062
Asset_087
Asset_099
Asset_100


KMeans on PCA(monthly returns) | cluster=7 | n=16 | daily close (full history)
3500
3000
2500
2000
1500
1000
500
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
Asset_014
Asset_028
Asset_040
Asset_042
Asset_047
Asset_056
Asset_057
Asset_058
Asset_063
Asset_065
Asset_067
Asset_068
Asset_069
Asset_073
Asset_078
Asset_083


GMM on PCA(monthly returns) | cluster=1 | n=8 | daily close (full history)
1000
800
600
400
200
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
Asset_014
Asset_028
Asset_042
Asset_058
Asset_067
Asset_068
Asset_078
Asset_083


GMM on PCA(monthly returns) | cluster=3 | n=9 | daily close (full history)
500
400
300
200
100
0
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
Asset_005
Asset_011
Asset_019
Asset_043
Asset_048
Asset_050
Asset_074
Asset_081
Asset_097





- this actually gives a pretty good result , even though its the most simple out of all of them

Use case

- arbitrage trading
- finding pairs reduces risk, you can make money in situations where markets go up , down. Only loose when markets are sideways
- more useful to find opposite pairs

fin