

Dự án Module 6 Tuần 4 - LTSF-Linear Forecasting Challenge.

Time-Series Team

Ngày 6 tháng 12 năm 2025

Mục lục

PHẦN 1: KHỞI ĐỘNG (Foundation)	3
1 Đặt vấn đề	3
2 Dataset	6
2.1 Cấu trúc dữ liệu	6
2.2 Những hạn chế và thiếu hụt của Dataset	6
3 EDA	6
3.1 Phân tích hạn chế của Baseline và Kiểm chứng giả thuyết DLinear	7
3.2 Từ Phân tích đến Kiến trúc Đặc trưng (From EDA to Feature Engineering)	8
3.2.1 1. Xử lý Trend phi tuyến (Transformation Features)	8
3.2.2 2. Giải mã trạng thái thị trường (Volatility Features)	8
3.2.3 3. Tín hiệu từ hành vi giá và dòng tiền (Candlestick & Volume Features)	9
4 Feature Engineering	9
4.1 Feature Engineering Overview	9
4.2 Price Action Features from OHLC	9
4.3 Volume & Money Flow Features	12
4.4 STL Decomposition Features	14
4.5 Returns and Pattern-based Features	15
4.6 Trend Slopes, Volatility Stats and Z-scores	16
4.7 Feature List and Modeling Targets	17
PHẦN 2: TRÁI TIM CỦA MÔ HÌNH (The Core Engine)	20
1 Kiến trúc Pipeline 3 lớp	20
1.1 Lớp 1: Math Backbone (Trend)	20
1.1.1 Từ log-price đến đường xu hướng dài hạn	20
1.1.2 Mở rộng xu hướng vào tương lai: định nghĩa future _ret_math	22
1.1.3 Khi câu chuyện toán học bước vào thực thi code	23
1.1.4 Backbone kết hợp với residual: từ lý thuyết đến đường giá dự báo	24
1.1.5 Kết luận: một đường thẳng giữa thế giới xao động	25
1.2 Lớp 2: ML Residual	25
1.2.1 Vì sao XGBoost phù hợp với bản chất dữ liệu residual hơn các mô hình khác?	28
1.2.2 Câu chuyện cuối cùng: sự kết hợp giữa cái lớn và cái nhỏ	31
1.2.3 Thực thi mô hình XGBoost trên residual: từ ý tưởng đến code	31
1.3 Lớp 3: Pricing Layer	33

2 Deep Dive vào Pricing Layer	33
2.1 Cơ chế vật lý	33
2.1.1 Clipping	33
2.1.2 Damping	34
2.1.3 Mean Reversion	36
2.2 Regime-aware Pricing	38
2.2.1 Phân loại Regime từ dữ liệu: <code>detect_regime</code>	39
2.2.2 Regime-based scaling trong Pricing Layer	40
2.3 Optimization Strategy	41
2.3.1 Bài toán tìm tham số Pricing	42
2.3.2 Random Search	43
2.3.3 Cross Validation (CV)	44
2.3.4 Random Search + CV: ghép lại thành chiến lược tối ưu	45
PHẦN 3: HỘI ĐỒNG CHUYÊN GIA (Ensemble Strategy)	47
1 Định nghĩa các Chuyên gia (Experts)	47
1.1 Dynamic Experts (Mô hình Động)	47
1.2 Static Experts (Mô hình Tĩnh)	47
1.2.1 Độc lập với nhiễu – một góc nhìn “sạch” về FPT	47
1.2.2 Neo giá trị trong Ensemble – giảm phương sai, giữ kỷ luật	48
1.2.3 Fail-safe Mechanism – khi ML “đi lạc”, toán vẫn đúng vững	49
1.2.4 Từ Math Backbone đến “Static Agent” – quyết định mang tính quản trị, không phải sự lười biếng	49
1.3 Risk Experts (Mô hình Rủi ro)	50
2 Cơ chế Ensemble	50
2.1 Weighted Ensemble	50
2.2 Lý giải vai trò từng Expert	50
PHẦN 4: KẾT QUẢ & ĐÁNH GIÁ (Evaluation)	51
1 Kết quả dự báo	51
1.1 Thiết lập thí nghiệm dự báo 100 ngày	51
1.2 Bức tranh trực quan: Hybrid + Pricing + Trend + Uncertainty	51
1.3 Diễn giải kết quả cho người dùng cuối	53
2 Đánh giá độ tin cậy	53
2.1 Cross-validation theo thời gian: mô hình có bền vững qua nhiều pha thị trường?	53
2.2 Pricing Layer được tối ưu bằng Random Search + CV	54
2.3 Regime hiện tại: SIDEWAYS sau điều chỉnh	55
2.4 Dải bất định & quản trị rủi ro	55
2.5 Kết luận ngắn cho phần độ tin cậy	56

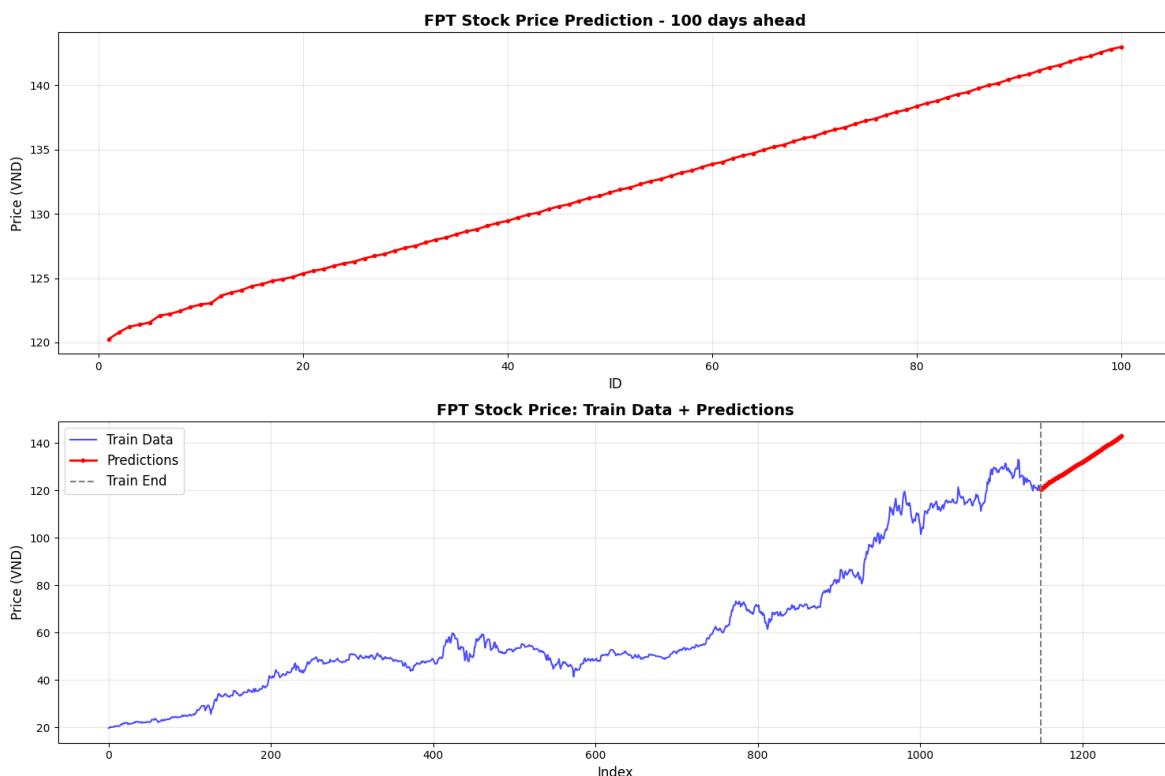
PHẦN 1: KHỞI ĐỘNG (Foundation)

1 Đặt vấn đề

1.1. Bức tường 100 ngày và *Cái chết của Phương sai*

Project 6.1 đặt ra bài toán nghe qua khá đơn giản: dự báo giá đóng cửa cổ phiếu FPT cho **100 ngày giao dịch tiếp theo** ($T+100$), chỉ sử dụng dữ liệu trong file `FPT_train.csv`. Trong ngắn hạn ($T+5$), các mô hình time-series kinh điển như ARIMA hay các biến thể Linear hiện đại (NLinear, DLinear) thường vẫn bám được xu hướng tương đối tốt.

Tuy nhiên, khi mở rộng sang **dự báo dài hạn**, đặc biệt là theo chiến lược *forecast cuốn chiếu* (recursive forecasting), baseline ban đầu của chúng tôi bộc lộ một vấn đề rất rõ: **đường dự báo 100 ngày gần như trở thành một đường thẳng tắp, không còn dao động** (hay hiện tượng "Cái chết của phương sai"). Hình 1 minh họa hiện tượng này:



Hình 1: *

Baseline Linear/NLinear: dự báo 100 ngày gần như trở thành một đường thẳng mượt, trái ngược với biến động mạnh của dữ liệu lịch sử.

Thoạt nhìn, rất dễ đổ lỗi cho **recursive forecasting** (dự báo cuốn chiếu) hay cho **MSE** (vì thường kéo dự báo về trung bình). Nhưng nếu dừng lại ở đó thì chưa đúng bản chất. Vấn đề thực sự nằm ở **cách thiết kế baseline và cách xử lý dữ liệu đầu vào**, cụ thể:

1. Bài toán mà baseline đang giải:

- Dữ liệu đầu vào là chuỗi log-price:

$$\text{close_log} = \log(\text{close})$$

- Mỗi mẫu huấn luyện lấy **14 ngày** làm input và **3 ngày** tiếp theo làm output. Bộ sinh dữ liệu được cài đặt như sau:
 - 14 giá log gần nhất → dự báo tiếp 3 giá log phía trước.
- Dự báo 100 ngày được tạo ra bằng cách lặp lại chiến lược này nhiều lần: sử dụng 14 ngày cuối (trong đó có cả dự báo trước đó) để dự đoán 3 ngày tiếp theo, cho đến khi đủ 100 ngày.

2. Kiến trúc mô hình: một tầng Linear trên chuỗi log-price

- Baseline sử dụng mô hình NLinear (nguyên bản sử dụng Linear thì kết quả còn tệ hơn trên public test Kaggle) với một tầng nn.Linear duy nhất, ánh xạ từ 14 giá trị log-price sang 3 giá trị log-price tiếp theo.
- Không có thành phần phi tuyến (nonlinearity), không có cơ chế nắm bắt cấu trúc thời gian phức tạp (như kernel, attention, hay recurrent).
- Về bản chất, mô hình chỉ học được một **hàm tuyến tính** trên đoạn log-price 14 ngày.

3. Bước chuẩn hoá của NLinear vô tình làm phẳng dao động:

- Trước khi đi vào tầng Linear, chuỗi đầu vào được chuẩn hoá theo:

$$x_{\text{norm}} = x - x_{\text{last}},$$

tức là toàn bộ 14 ngày được trừ đi giá trị ngày cuối cùng, rồi sau khi dự báo xong mới cộng lại.

- Bước này giúp mô hình ổn định hơn khi phân phối giá thay đổi, nhưng đồng thời cũng **giảm mạnh biên độ dao động tương đối** trong đoạn 14 ngày – đặc biệt khi log-price vốn đã khá mượt.

4. Dữ liệu log-price của FPT gần như tuyến tính theo thời gian:

- FPT có xu hướng tăng trưởng theo thời gian, gần giống một hàm mũ theo ngày giao dịch.
- Khi lấy log, chuỗi giá trở thành một đường có **trend gần tuyến tính**, tức là:

$$\log(FPT(t)) \approx at + b.$$

- Trong bối cảnh như vậy, một tầng Linear được huấn luyện bằng MSE trên rất nhiều cửa sổ 14→3 gần nhau sẽ tự nhiên học ra **một slope trung bình** cho log-price, chứ không học được volatility.

5. Tại sao recursive làm mọi thứ “thẳng tắp” hơn?

- Bản thân recursive forecasting không phải là thủ phạm giết phương sai – rất nhiều mô hình AR, RNN, LSTM vẫn dùng recursive bình thường.
- Nhưng trong baseline này, mỗi lần dự báo:
 - mô hình dùng **input đã bị làm mượt** (log + chuẩn hoá),
 - áp dụng **một hàm tuyến tính** với slope gần như cố định.

- Khi ta liên tục feed lại các dự báo mượt đó vào vòng sau, **mọi nhiễu nhỏ còn sót lại bị “là phẳng” dần**, và quỹ đạo dự báo hội tụ thành:

$$\hat{y}_{t+k} \approx \hat{y}_t + k \cdot \Delta,$$

tức là một đường thẳng với độ dốc gần như cố định.

Nói cách khác, hiện tượng **“Cái chết của phương sai”** ở đây không phải là một quy luật tự nhiên của recursive forecasting hay của MSE, mà là hệ quả tất yếu khi kết hợp:

- **Log transform** làm mượt chuỗi giá,
- **Chuẩn hoá giá trị cuối** trong NLinear làm phẳng thêm dao động,
- **Mô hình tuyến tính đơn tầng** chỉ học được trend tuyến tính,
- **Dự báo cuốn chiếu nhiều bước** khuếch đại tính tuyến tính này qua thời gian.

Kết quả cuối cùng chính là những gì ta thấy ở Hình 1: baseline có thể giữ được hướng tăng, nhưng **mất hoàn toàn đặc trưng volatility** của cổ phiếu FPT. Đây là lý do chúng tôi cần một pipeline mới, nơi phần *trend* vẫn được mô hình hoá bằng một backbone toán đơn giản, nhưng phần *dao động* và *hành vi thị trường* phải được xử lý bởi một lớp mô hình khác giàu năng lực hơn.

1.2. Vì sao Deep Learning không phải là *Chén Thánh?*

Nghe có vẻ hợp lý khi nâng cấp lên LSTM hoặc Transformer. Tuy nhiên, với dữ liệu chứng khoán Việt Nam (FPT là ví dụ điển hình), DL lại vấp phải hai rào cản lớn:

1. **Dữ liệu hạn chế:** vài nghìn điểm dữ liệu là quá ít. DL cực kỳ dễ overfit, học vẹt nhiều thay vì quy luật.
2. **Thiếu cơ chế “tự sửa lỗi”:** DL thuần tuý dựa trên dữ liệu. Khi thị trường chuyển chế độ (từ Bull sang Bear), mô hình có xu hướng **ngoại suy mù quáng** dựa trên xu hướng cũ.

Do đó, DL không phải lời giải tối ưu cho bài toán dự báo dài hạn với dữ liệu hạn chế như FPT.

1.3. Lời giải Hybrid: Khi Machine Learning gặp Financial Engineering

Thay vì cố xây một mô hình **siêu AI** làm tất cả, ta áp dụng triết lý **Decomposition – chia để trị**, để mỗi thành phần mô hình làm đúng thứ nó giỏi nhất.

1. **Math Backbone (Trend dài hạn):** Dùng Linear trên log-price để giữ hướng chính xác của quỹ đạo dài hạn, tránh dự báo trôi dạt vô lý.
2. **XGBoost Residual (Dao động ngắn hạn):** Linear không thể học hết nhiễu động. XGB học residual giúp:
 - tái tạo nhịp dao động ngắn hạn,
 - tăng realism cho đường giá,
 - ổn định hơn DL trên lượng dữ liệu nhỏ.
3. **Pricing Layer (Lớp kiểm soát hành vi thị trường):** Đây là phần nâng cấp quan trọng nhất của pipeline.

- **Mean Reversion:** giá sẽ có lực hút về vùng giá trị thực.
- **Regime Detection:** nhận diện Bull/Bear để điều chỉnh biên độ dao động.

Tinh thần của pipeline: Mục tiêu không phải dự đoán chính xác từng ngày (điều bất khả thi), mà là xây dựng một **trajectory giá hợp lý và vững** – kết hợp quán tính toán học và tâm lý thị trường.

2 Dataset

Dữ liệu được sử dụng trong dự án là tập tin `FPT_train.csv`, chứa thông tin giao dịch lịch sử của cổ phiếu FPT. Đây là dữ liệu chuỗi thời gian dạng bảng (Tabular Time-series) điển hình trong tài chính.

2.1 Cấu trúc dữ liệu

Feature	Mô tả
<code>Open</code>	Giá mở cửa phiên giao dịch.
<code>High</code>	Mức giá cao nhất đạt được trong ngày (thể hiện áp lực mua/hưng phấn).
<code>Low</code>	Mức giá thấp nhất trong ngày (thể hiện áp lực bán/sợ hãi).
<code>Close</code>	Giá đóng cửa (thường được dùng làm target chính để dự báo).
<code>Volume</code>	Khối lượng giao dịch (động lực/nhiên liệu của xu hướng giá).

Bảng 1: Mô tả các biến trong tập dữ liệu FPT_train

Tập dữ liệu bao gồm các trường thông tin cơ bản (OHLCV) mô tả hành vi giao dịch theo ngày: Đặc trưng của cổ phiếu FPT là một mã cổ phiếu công nghệ đầu ngành tại Việt Nam, có xu hướng dài hạn là tăng trưởng (Uptrend). Tuy nhiên, trong ngắn hạn và trung hạn, nó chịu sự chi phối mạnh của các chu kỳ thị trường (Market Regimes) như Bull (Tăng), Bear (Giảm) và Sideways (Đi ngang).

2.2 Những hạn chế và thiếu hụt của Dataset

Dựa trên phân tích thực tế, tập dữ liệu này tồn tại những điểm "thiếu thực tế" gây khó khăn lớn cho việc dự báo chính xác 100 ngày nếu chỉ dùng model thuần túy:

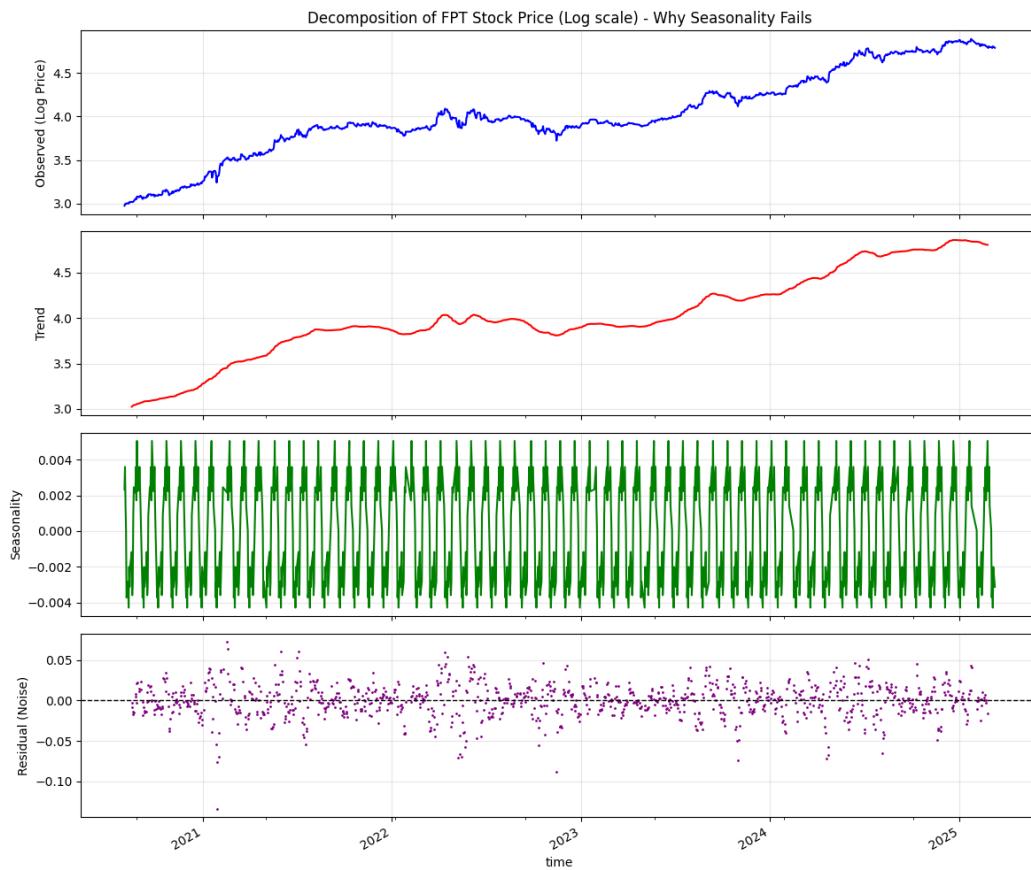
- **Thiếu thông tin ngữ nghĩa (No Semantic/News Data):** Dataset chỉ bao gồm các con số khô khan. Model hoàn toàn "mù" trước các thông tin vĩ mô hoặc tin tức doanh nghiệp (ví dụ: tin hợp tác AI, báo cáo tài chính). Như đã đề cập ở phần đặt vấn đề, việc thiếu phân tích cảm xúc khiến model không thể giải thích được các cú tăng/giảm đột biến do tin tức.
- **Độ dài dữ liệu so với tầm dự báo (Horizon Mismatch):** Dữ liệu huấn luyện chỉ khoảng hơn 1000 mẫu (tương đương khoảng 4.5 năm), nhưng yêu cầu dự báo xa tới 100 ngày (tương đương gần 1/4 năm giao dịch). Tỷ lệ này là quá lớn, khiến các mô hình dễ bị nhiễu và mất phương hướng ở các ngày cuối của chu kỳ dự báo.
- **Thiếu các biến phái sinh (Lack of Derived Features):** Dữ liệu thô chưa phản ánh được động lượng (Momentum) hay biến động (Volatility). Việc dự báo dựa trên dữ liệu thô mà không có Feature Engineering (như RSI, MACD, hay các tín hiệu dòng tiền) giống như việc lái xe chỉ nhìn công tơ mét mà không nhìn đường.

3 EDA

Để xây dựng một chiến lược dự báo hiệu quả, trước tiên nhóm mình tiến hành phân tích thất bại của các phương pháp Baseline và khám phá đặc tính thống kê của dữ liệu FPT (2020-2024).

3.1 Phân tích hạn chế của Baseline và Kiểm chứng giả thuyết DLinear

Trong giai đoạn thử nghiệm ban đầu (Baseline), nhóm mình đã áp dụng các mô hình phân rã chuỗi thời gian phổ biến như DLinear. Giả thuyết ban đầu là tách chuỗi giá thành hai phần: **Trend** (Xu hướng) và **Seasonality** (Mùa vụ) để dự báo riêng biệt. Tuy nhiên, kết quả thực nghiệm và trực quan hóa dữ liệu đã chỉ ra những vấn đề nghiêm trọng:



Hình 2: Kết quả kiểm chứng DLinear: Sự thống trị của Nhiễu so với Mùa vụ

Insight 1: "Mùa vụ ảo" và sự thống trị của Nhiễu (Noise Dominance) Từ Hình ??, ta thấy thành phần Seasonality được trích xuất bởi DLinear có biên độ dao động rất nhỏ và không mang tính chu kỳ rõ rệt (khác với dữ liệu thời tiết hay doanh số bán lẻ).

- **Thực tế:** Biến động ngắn hạn của cổ phiếu FPT chủ yếu là **Nhiễu (Noise)** do tâm lý thị trường, không phải là tính mùa vụ tất định.
- **Hậu quả:** Việc ép mô hình học một "mùa vụ ảo" dẫn đến Overfitting. Mô hình cố gắng khớp các dao động ngẫu nhiên trong quá khứ thay vì học xu hướng thực sự.

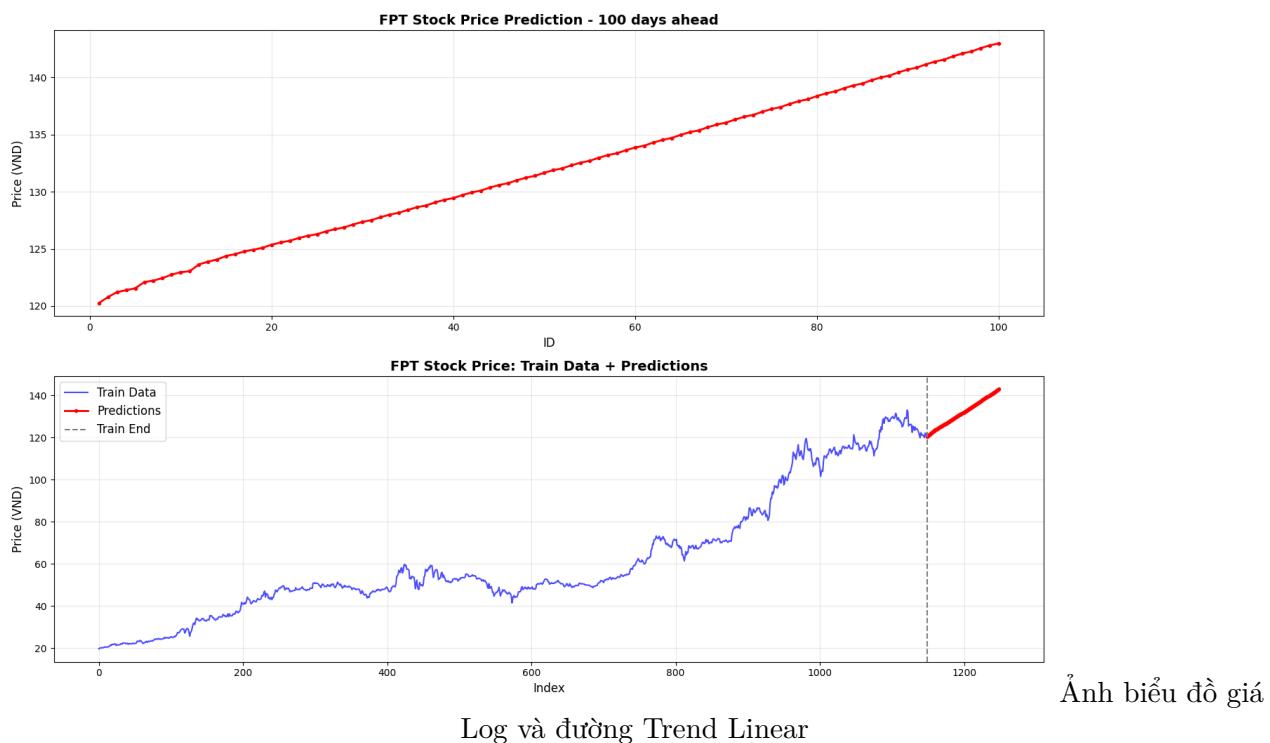
Insight 2: Bẫy ngoại suy tuyến tính (Linear Extrapolation Trap) Khi quan sát đường Trend của Baseline, mô hình có xu hướng kéo dài và tăng của năm 2021-2023 một cách vô hạn. Điều này khiến dự báo sai lệch hoàn toàn tại các điểm gãy (Structural Breaks) như đợt sụt giảm năm 2022.

→ **Kết luận từ Baseline:** Các phương pháp phân rã truyền thống (như trong DLinear) là không đủ. Chúng ta cần một phương pháp xử lý nhiễu tốt hơn (STL robust) và quan trọng hơn là cần thêm các biến giải thích (Explanatory Variables) để mô hình hiểu được "tại sao giá biến động" thay vì chỉ học vẹt chuỗi số.

3.2 Từ Phân tích đến Kiến trúc Đặc trưng (From EDA to Feature Engineering)

Dựa trên những hạn chế đã xác định ở trên, nhóm mình đề xuất chuyển đổi từ việc sử dụng dữ liệu thô sang hệ thống đặc trưng ngữ nghĩa (Semantic Features). Các phân tích EDA dưới đây là cơ sở trực tiếp cho việc thiết kế 5 nhóm features trong mô hình Hybrid.

3.2.1 1. Xử lý Trend phi tuyến (Transformation Features)



Hình 3: Biểu đồ giá Log-Logarit và đường xu hướng dài hạn

Quan sát Hình 3, giá FPT tăng theo cấp số nhân. Việc sử dụng giá tuyệt đối (Pt) khiến mô hình thiên vị các dữ liệu gần nhất (giá cao) và coi thường dữ liệu quá khứ (giá thấp).

- **Giải pháp FE:** Chuyển đổi sang không gian Logarit (close_log) để tuyến tính hóa tốc độ tăng trưởng, giúp mô hình Linear Backbone hoạt động ổn định hơn.

3.2.2 2. Giải mã trạng thái thị trường (Volatility Features)

Hình ?? cho thấy độ biến động (Volatility) của FPT không cố định (Heteroskedasticity). Có giai đoạn đi ngang (Sideways) với biên độ hẹp, và giai đoạn bùng nổ (Crisis/Uptrend) với biên độ rộng.

- **Giải pháp FE:** Cần các features đo lường rủi ro như atr_14, park_vol (Parkinson Volatility). Đây sẽ là đầu vào cho bộ lọc Pricing Layer để điều chỉnh biên độ dự báo.

3.2.3 3. Tín hiệu từ hành vi giá và dòng tiền (Candlestick & Volume Features)

4 Feature Engineering

4.1 Feature Engineering Overview

Để mô hình có thể học được những gì ta quan sát trong EDA (xu hướng, biến động, volume spike, pattern nén, ...) dữ liệu OHLCV thô cần được biến đổi thành một bộ đặc trưng giàu cấu trúc. Hàm `add_stl_ohlc_features` thực hiện toàn bộ bước này cho cổ phiếu FPT:

- Chuẩn hoá giá về log (`close_log`).
- Trích xuất **price action** từ OHLC: body, range, shadows, gaps, ATR, Parkinson, ...
- Xây **volume & money flow features**.
- Áp dụng **STL decomposition** trên log-price.
- Tạo **returns 1/5/10 ngày** và **pattern up/down 3-streak**.
- Tính **trend slopes, acceleration, rolling stats, z-score**.

Cuối cùng, hàm `build_modeling_df` ghép các đặc trưng này với target dự báo: `future_ret`, `math_ret`, `resid_ret`, và `get_feature_cols` định nghĩa danh sách các feature sẽ đưa vào XGBoost.

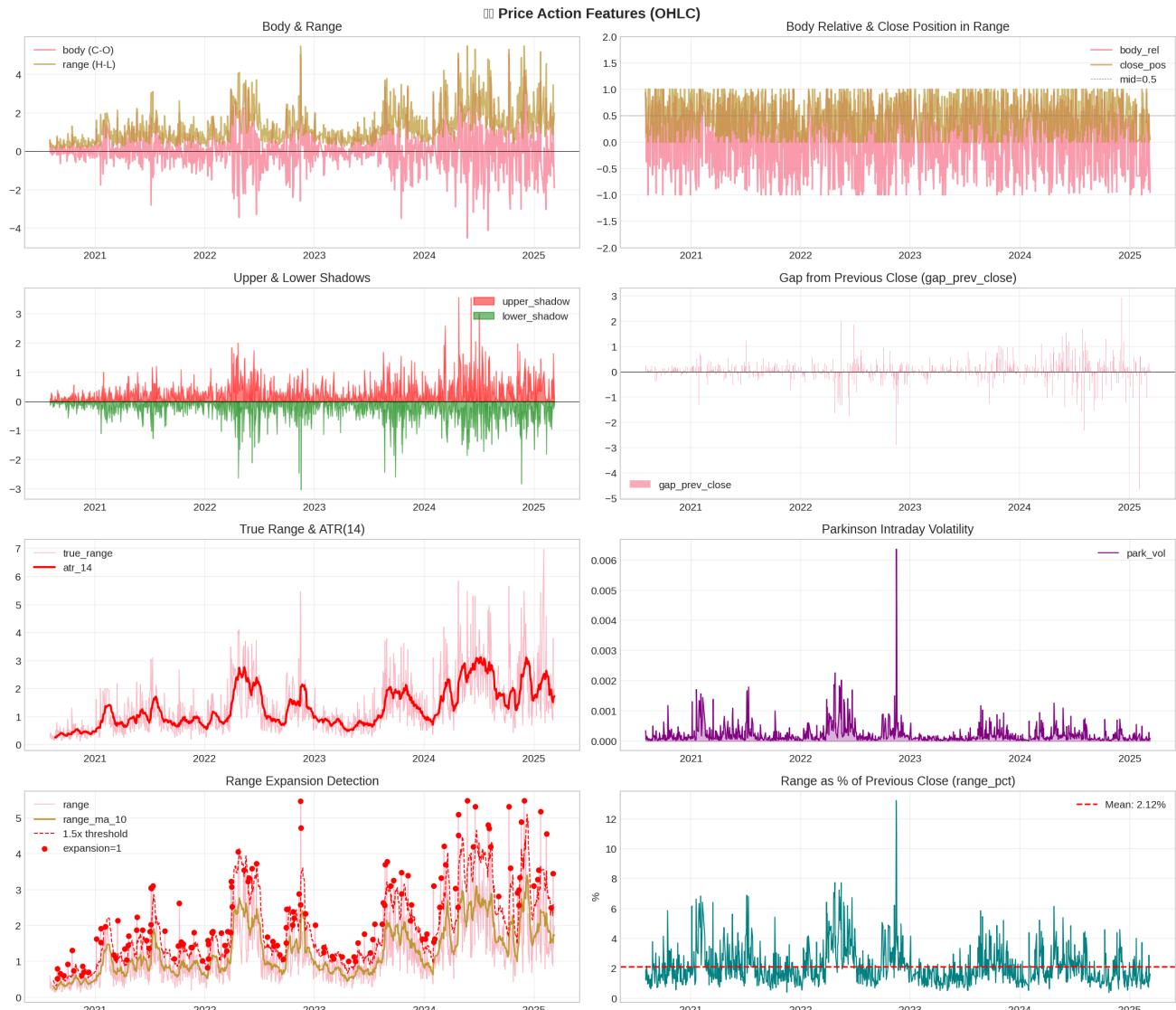
Dưới đây, ta lần lượt đi qua từng nhóm đặc trưng.

4.2 Price Action Features from OHLC

Khối đầu tiên trong `add_stl_ohlc_features` kiểm tra dữ liệu có đủ cột `open`, `high`, `low` hay không:

```
1 has_ohlc = all(c in df.columns for c in ["open", "high", "low"])
```

Nếu có, ta trích xuất một loạt đặc trưng price action; nếu không, toàn bộ các cột liên quan (body, range, ...) sẽ được gán 0.0 để mô hình vẫn chạy được.



Hình 4: Price action

Cấu trúc nến: body, range, shadows, close_pos

$$\text{body} = C - O, \quad \text{range} = H - L,$$

$$\text{body_rel} = \frac{\text{body}}{|\text{range}| + \varepsilon}, \quad \text{close_pos} = \frac{C - L}{|\text{range}| + \varepsilon}.$$

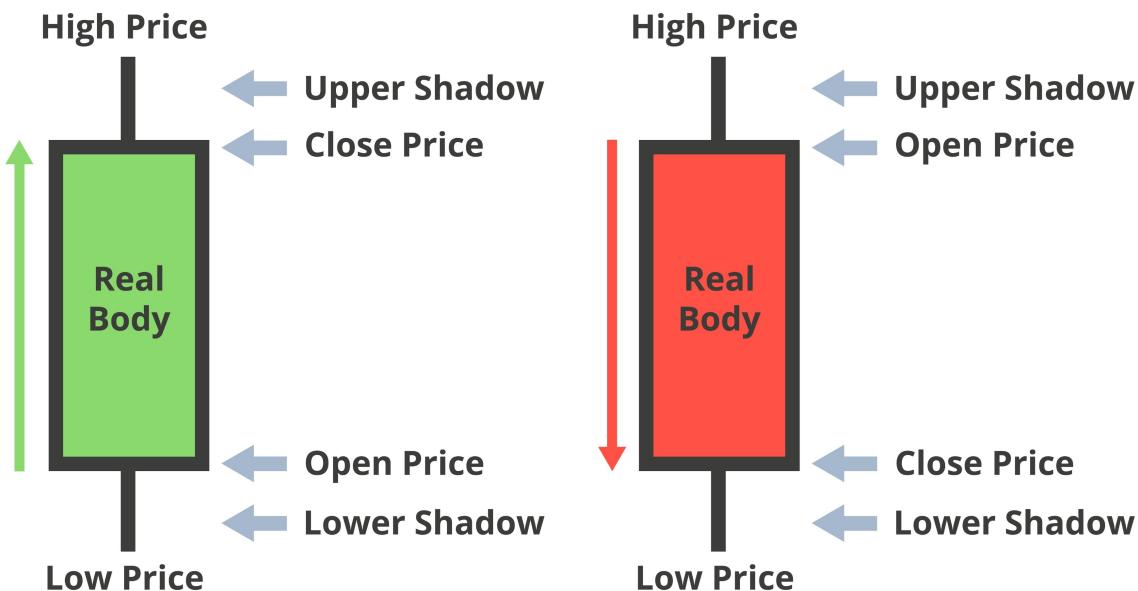
$$\text{upper_shadow} = H - \max(O, C), \quad \text{lower_shadow} = \min(O, C) - L.$$

Những biến này mã hoá hình dạng cây nến:

- body, body_rel: thân dài/ngắn, nến mạnh hay doji.
- range: độ biến động trong phiên.
- upper_shadow, lower_shadow: lực mua/bán bị đẩy lùi.

- `close_pos`: giá đóng cửa ở gần đỉnh, giữa hay đáy nến.

CANDLESTICK COMPONENTS



Hình 5: Cấu trúc nến

Gaps: `gap_oc`, `gap_prev_close`

$$\text{gap_oc} = O - C, \quad \text{gap_prev_close} = O - C_{t-1}.$$

Hai biến này đo khoảng cách giữa open và close hôm trước cho mô hình biết có gap-up/gap-down bất thường hay không—thường liên quan đến tin tức hoặc tâm lý mạnh.

`range_pct`: biên độ chuẩn hóa theo giá hôm trước

$$\text{range_pct} = \frac{\text{range}}{|C_{t-1}| + \varepsilon}.$$

Khi `range_pct` lớn, đó là một phiên “siêu biến động” so với mức giá trước đó. Đây là dạng tín hiệu mà ta đã thấy phản ánh rõ trong biểu đồ nến và volatility.

True Range, ATR và Parkinson volatility True range được tính theo kiểu Wilder:

$$\begin{aligned} \text{true_range}_t &= \max\{|H_t - L_t|, |H_t - C_{t-1}|, |L_t - C_{t-1}|\}, \\ \text{atr_14} &= \text{MA}_{14}(\text{true_range}). \end{aligned}$$

Song song, ta có volatility kiểu Parkinson:

$$\text{park_vol}_t = \frac{1}{4 \ln 2} [\ln(H_t/(L_t + \varepsilon))]^2.$$

Hai biến này cho mô hình một đo lường ổn định về **biến động nội phiên** không chỉ nhìn vào close-to-close.

range_ma_10 và range_expansion

$$\text{range_ma}_10 = \text{MA}_{10}(\text{range}), \quad \text{range_expansion} = \mathbb{1}(\text{range} > 1.5 \cdot |\text{range_ma}_10|).$$

`range_expansion` là flag (0/1) cho biết hôm nay có “bùng nổ biến động” so với 10 ngày gần nhất hay không—rất phù hợp với các phiên breakout / breakdown mà ta thấy trong candlestick chart.

Lag features: body_lag1, body_lag2, close_pos_lag1 Cuối cùng, ta thêm một chút “trí nhớ” cho mô hình bằng:

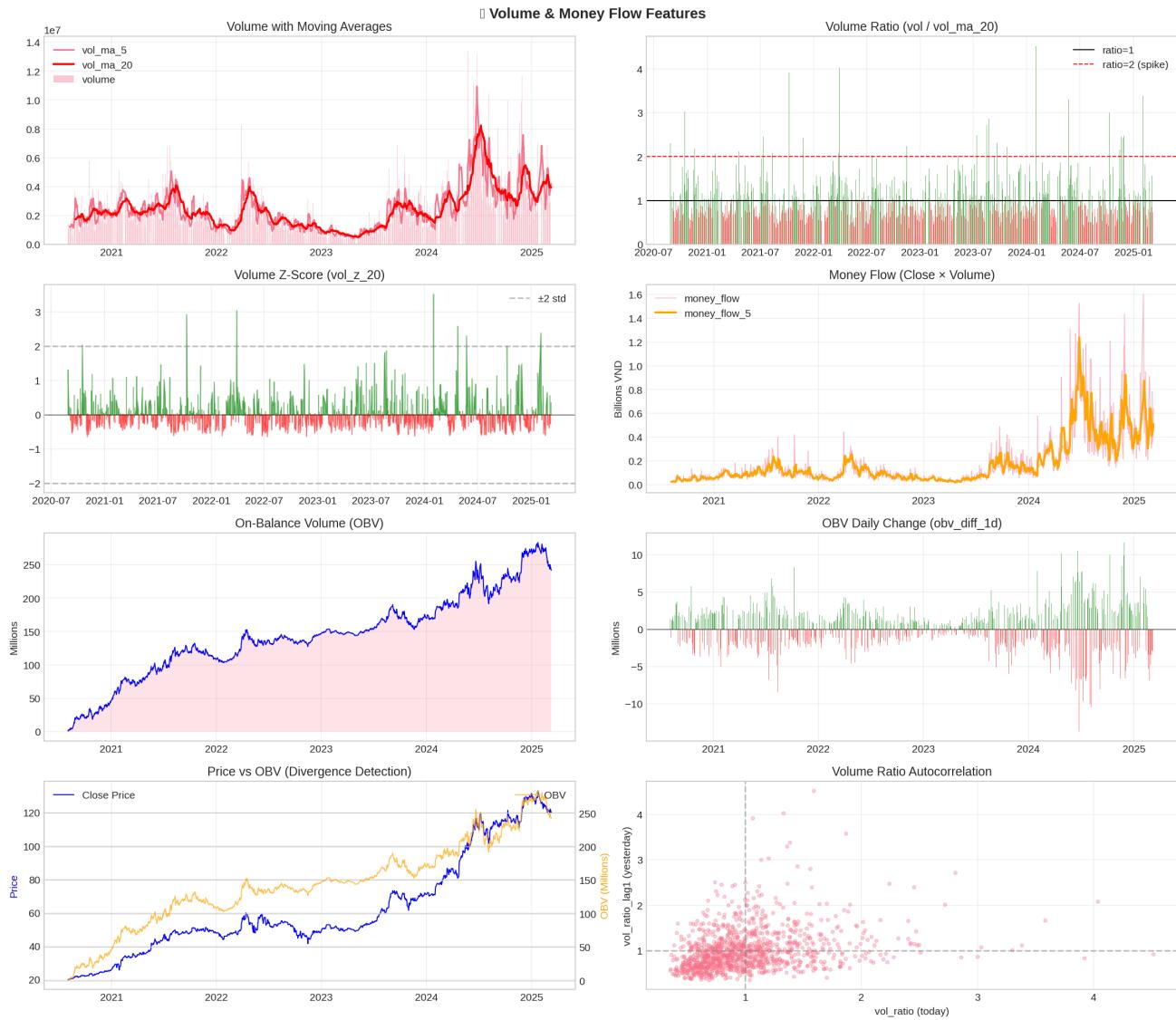
$$\text{body_lag1} = \text{body}_{t-1}, \quad \text{body_lag2} = \text{body}_{t-2}, \quad \text{close_pos_lag1} = \text{close_pos}_{t-1}.$$

Những biến lag này cho phép XGBoost học được các pattern dạng “hai cây nến liên tiếp thân dài cùng hướng”, hoặc “hôm qua đóng cửa sát đỉnh, hôm nay biến động mạnh”.

Nếu thiếu `open/high/low`, toàn bộ các cột price action (`pa_cols`) được set = 0.0 để duy trì cấu trúc bảng.

4.3 Volume & Money Flow Features

Khỏi thứ hai xử lý `volume` và các chỉ số dòng tiền. Nếu không có cột `volume`, toàn bộ các cột trong `vol_cols` được gán 0.0.



Hình 6: Volume features

Moving averages và `vol_ratio`

$$\text{vol_ma_5} = \text{MA}_5(\text{volume}), \quad \text{vol_ma_20} = \text{MA}_{20}(\text{volume}),$$

$$\text{vol_ratio} = \frac{\text{volume}}{|\text{vol_ma_20}| + \epsilon}.$$

`vol_ratio` » 1 cho thấy volume hiện tại lớn hơn nhiều so với 20 ngày gần nhất, tức có sự quan tâm bất thường từ thị trường. Ta còn có `vol_ratio_lag1` để mô hình biết hôm qua volume đã “nóng” hay chưa.

Money flow và `money_flow_5`

$$\text{money_flow}_t = C_t \times \text{volume}_t, \quad \text{money_flow_5} = \text{MA}_5(\text{money_flow}).$$

Đây là proxy cho lượng tiền thực sự chảy vào cổ phiếu không chỉ là số cổ phiếu được giao dịch. Các phiên tiền vào ra mạnh thường đi kèm với price action đặc biệt.

vol_z_20: volume bất thường so với trung bình

$$\text{vol_z_20} = \frac{\text{volume} - \text{vol_ma_20}}{|\text{vol_ma_20}| + \varepsilon}.$$

Giống một Z-score đơn giản cho volume, cho biết volume hôm nay cao/thấp bao nhiêu so với trung bình 20 ngày.

On-Balance Volume (OBV) và obv_diff_1d Ta tính dấu chuyển động giá:

$$\text{sign}_t = \text{sign}(C_t - C_{t-1}),$$

rồi tích luỹ:

$$\text{obv}_t = \text{obv}_{t-1} + \text{sign}_t \cdot \text{volume}_t.$$

obv_diff_1d là chênh lệch OBV ngày qua ngày. Những biến này cho phép mô hình nắm bắt xu hướng tích luỹ/phân phối của dòng tiền, một yếu tố thường báo hiệu trend đảo chiều sớm hơn giá.

4.4 STL Decomposition Features

Tiếp theo, ta áp dụng STL decomposition trên `close_log`:

```

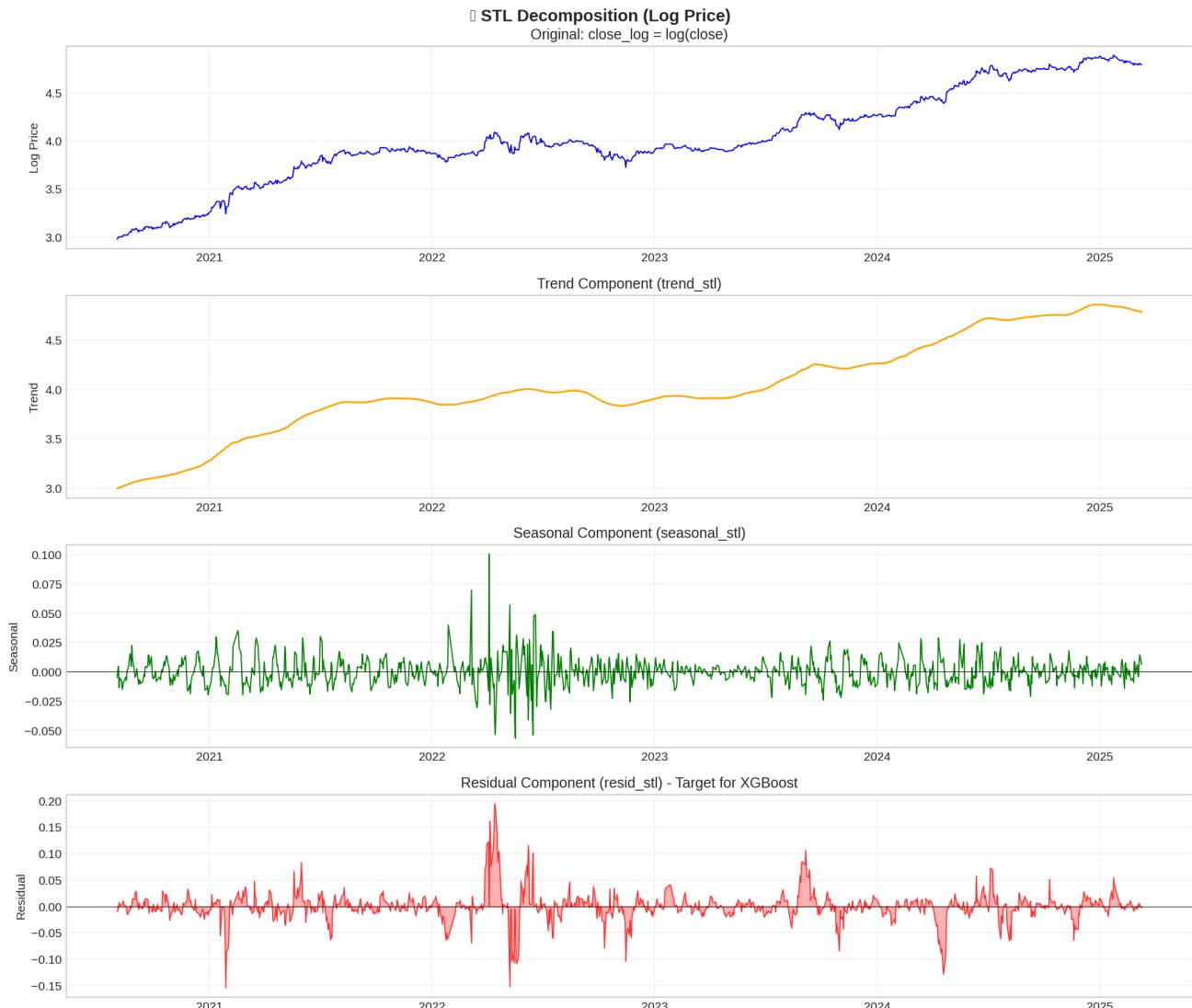
1 close_log = df[["close_log"]].copy()
2 if close_log.isna().any():
3     close_log = close_log.fillna(method="ffill").fillna(method="bfill")
4
5 res = STL(close_log.values, period=stl_period, robust=True).fit()
6 df[["trend_stl"]] = res.trend
7 df[["seasonal_stl"]] = res.seasonal
8 df[["resid_stl"]] = res.resid

```

Kết quả là:

$$\text{close_log} = \text{trend_stl} + \text{seasonal_stl} + \text{resid_stl},$$

giúp tách rõ xu hướng dài hạn, dao động có tính chu kỳ (nếu tồn tại) và nhiễu.



Hình 7: Feature Decomposition

4.5 Returns and Pattern-based Features

Log-returns 1/5/10 ngày

$$\text{ret_1d} = \Delta_1 \text{close_log}, \quad \text{ret_5d} = \Delta_5 \text{close_log}, \quad \text{ret_10d} = \Delta_{10} \text{close_log}.$$

Những biến này ghi lại quan tính giá trong ngắn hạn và trung hạn, một yếu tố quan trọng khi dự báo future_ret 100 ngày.



Hình 8: Return Features

Up/down flags và 3-streak

$$\begin{aligned} \text{up_1d} &= \mathbb{1}(\text{ret_1d} > 0), \quad \text{down_1d} = \mathbb{1}(\text{ret_1d} < 0), \\ \text{up_3streak} &= \mathbb{1}(\text{ret_1d} > 0, \text{ret_1d}^{(-1)} > 0, \text{ret_1d}^{(-2)} > 0), \\ \text{down_3streak} &= \mathbb{1}(\text{ret_1d} < 0, \text{ret_1d}^{(-1)} < 0, \text{ret_1d}^{(-2)} < 0). \end{aligned}$$

Các pattern này mã hoá những chuỗi “3 ngày tăng/giảm liên tiếp” rất thường được trader chú ý và cũng là dạng pattern phi tuyển mà XGBoost học rất tốt.

4.6 Trend Slopes, Volatility Stats and Z-scores

Cuối cùng, ta xây các biến liên quan đến động học của trend và mức độ biến động của residual:

Trend slopes và acceleration

$$\begin{aligned} \text{trend_slope_1} &= \Delta_1 \text{trend_stl}, \quad \text{trend_slope_3} = \Delta_3 \text{trend_stl}, \quad \text{trend_slope_7} = \Delta_7 \text{trend_stl}, \\ \text{trend_accel_3} &= \text{trend_slope_1} - \text{trend_slope_1}^{(-3)}. \end{aligned}$$

Chúng cho biết xu hướng không chỉ đang tăng/giảm mà còn đang tăng nhanh hơn hay chậm lại.

Rolling stats và z-score

$$\text{trend_mean_21} = \text{MA}_{21}(\text{trend_stl}), \quad \text{trend_std_21} = \text{STD}_{21}(\text{trend_stl}),$$

$$\text{resid_std_10} = \text{STD}_{10}(\text{resid_stl}), \quad \text{resid_std_20} = \text{STD}_{20}(\text{resid_stl}),$$

$$z_{\text{trend_21}} = \frac{\text{trend_stl} - \text{trend_mean_21}}{\text{trend_std_21} + \varepsilon}.$$

Những biến này đo mức độ “bất thường” hiện tại so với dải biến động 21 ngày: trend đang ở vùng cao/thấp nào, residual đang nhiễu mạnh hay yếu. Đây là nguồn thông tin quan trọng cho cả pricing-layer lẫn Monte Carlo về sau.



Hình 9: Trend Dynamics

4.7 Feature List and Modeling Targets

Danh sách feature đưa vào XGBoost Hàm `get_feature_cols` gom toàn bộ các feature kể trên vào một list `base_cols`, gồm:

- Các biến log-price, returns, patterns: `close_log`, `ret_1d/5d/10d`, `up_1d`, `down_1d`, `up_3streak`, `down_3streak`.
- Các biến STL & trend dynamics: `trend_stl`, `trend_slope_1/3/7`, `trend_accel_3`, `trend_mean_21`, `trend_std_21`, `resid_std_10/20`, `z_trend_21`.
- Price action OHLC: `body`, `range`, `upper_shadow`, `lower_shadow`, `body_rel`, `close_pos`, `gap_oc`, `gap_prev_close`, `range_pct`, `range_ma_10`, `range_expansion`, `body_lag1/2`, `close_pos_lag1`, `true_range`, `atr_14`, `park_vol`.
- Volume & money flow: `volume`, `vol_ma_5/20`, `vol_ratio`, `vol_ratio_lag1`, `money_flow`, `money_flow_5`, `vol_z_20`, `obv`, `obv_diff_1d`.

Xây modeling dataframe: future_ret, math_ret, resid_ret Hàm `build_modeling_df` là cầu nối giữa feature engineering và mô hình hoá:

1. Sắp xếp dữ liệu theo `time`, áp dụng `add_stl_ohlcv_features` để tạo `df_feat`.
2. Tạo `future_ret` với horizon h :

$$\text{future_ret}_t = \log P_{t+h} - \log P_t,$$

bằng cách shift `close_log` $-h$ ngày.

3. Xây **Math Backbone** tuyển tính trên log-price:

$$\text{trend_log}_t = \hat{\beta}_0 + \hat{\beta}_1 t, \quad \text{math_ret}_t = \text{trend_log}_{t+h} - \text{trend_log}_t.$$

4. Tạo `resid_ret`:

$$\text{resid_ret}_t = \text{future_ret}_t - \text{math_ret}_t,$$

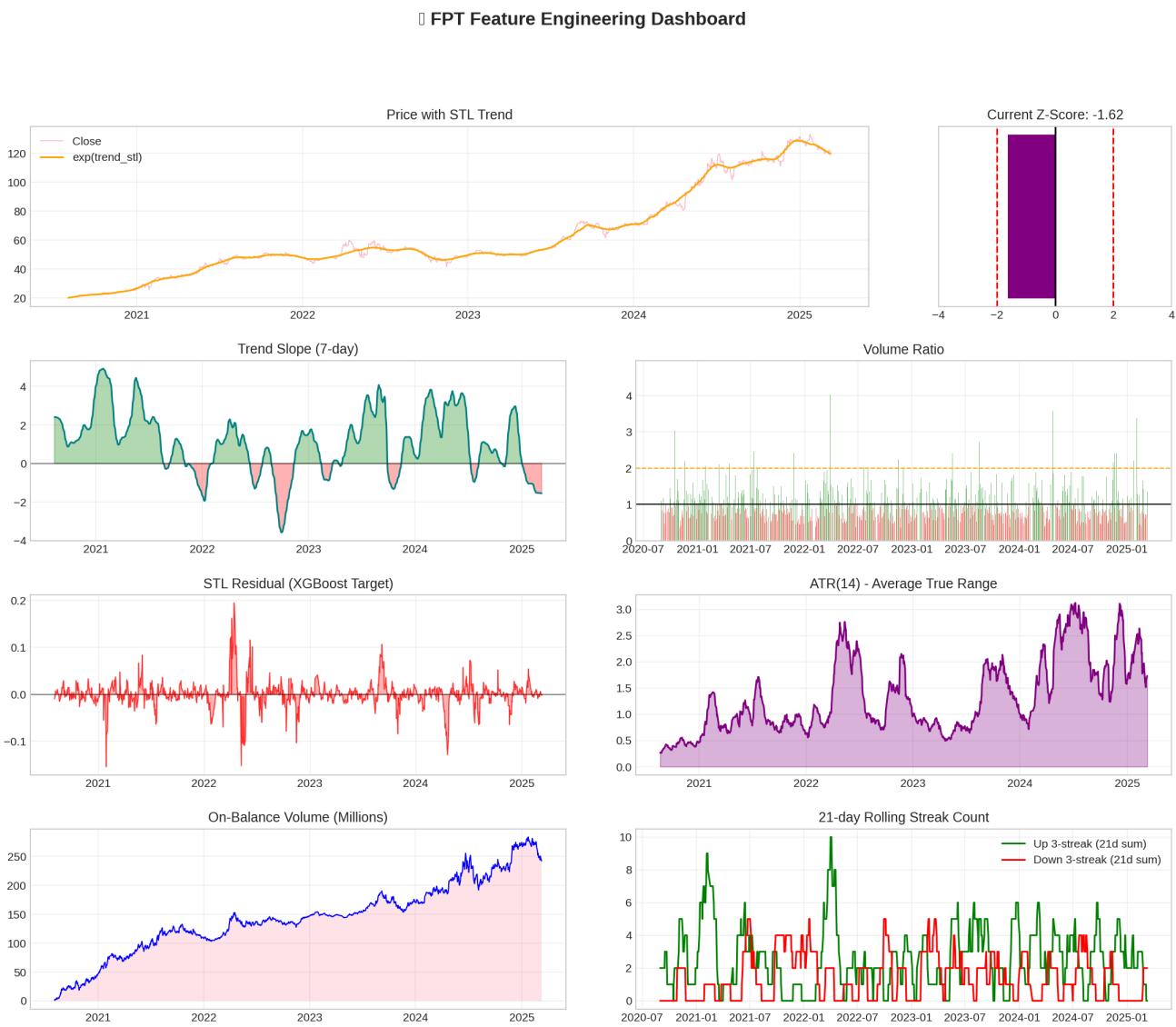
đây là target mà XGBoost residual model sẽ học.

5. Cuối cùng, `df_model` được tạo bằng cách `dropna` trên:

$$\text{feature_cols} \cup \{\text{future_ret}, \text{math_ret}, \text{resid_ret}, \text{time_future}\},$$

đảm bảo mọi hàng đưa vào mô hình đều có đầy đủ feature và target.

Như vậy, toàn bộ code feature engineering và xây modeling dataframe đã được thiết kế nhất quán với câu chuyện EDA và kiến trúc hybrid: **trend dài hạn** được tách riêng thành backbone, **dao động phi tuyến** và **nhiều ngắn hạn** được mã hóa qua các feature OHLCV, volume, STL, patterns để XGBoost chỉ tập trung vào phần residual mà nó giỏi nhất.



Hình 10: Features Dashboard

PHẦN 2: TRÁI TIM CỦA MÔ HÌNH (The Core Engine)

1 Kiến trúc Pipeline 3 lớp

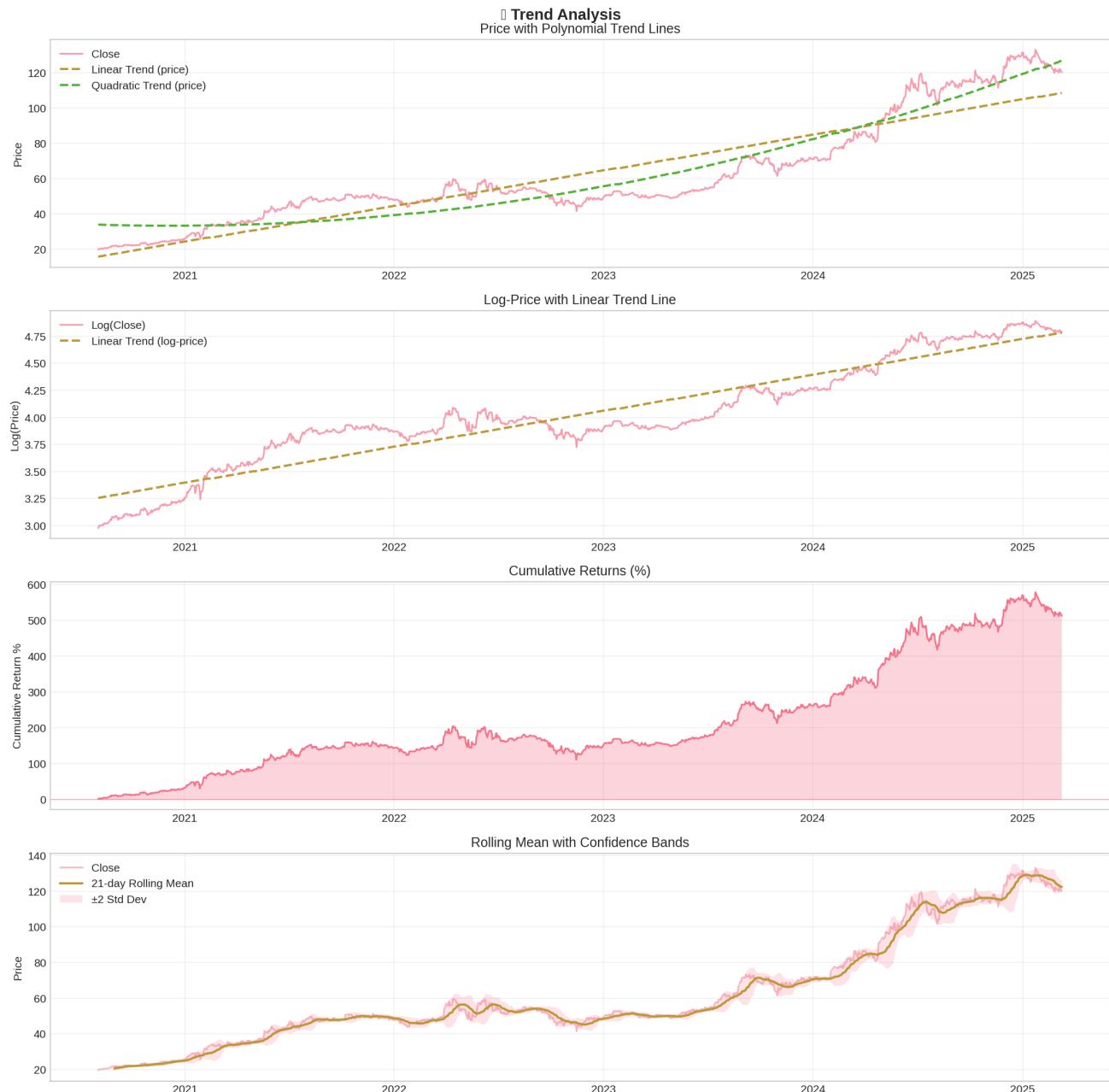
1.1 Lớp 1: Math Backbone (Trend)

Ngay từ những bước đầu xây dựng pipeline dự báo 100 ngày cho cổ phiếu FPT, một câu hỏi cứ trở đi trở lại: liệu thị trường có thật sự hỗn loạn như vẻ bề ngoài, hay đằng sau những biến động từng phiên vẫn tồn tại một quỹ đạo dài hạn mà ta có thể nắm bắt? Khi quan sát chuỗi giá nhiều năm, đặc biệt qua các biểu đồ tổng quan giá, candlestick, và đường trung bình động, điều hiện lên không phải là sự ngẫu nhiên tuyệt đối. Thay vào đó, có một dòng chảy mượt mà chạy xuyên suốt thời gian: giá FPT liên tục đi lên với nhịp độ khá đều đặn. Khi chuyển sang thang log-price [11](#), đường xu hướng ấy thậm chí còn trở nên gần như tuyến tính, việc này thể hiện một dấu hiệu mạnh mẽ rằng tăng trưởng dài hạn là cấu trúc nền của chuỗi giá.

Chính từ trực giác đó, **Math Backbone** được sinh ra: một bộ khung toán học đơn giản nhưng đóng vai trò “cột sống” giữ cho dự báo dài hạn không trôi theo nhiều động ngẫu hàn.

1.1.1 Từ log-price đến đường xu hướng dài hạn

Biểu đồ decomposition của chuỗi thời gian cho thấy phần *Trend* mượt, ổn định và tăng trưởng rõ rệt, trong khi phần seasonal lặp lại theo chu kỳ và phần residual dao động không có quy luật. Điều đó gợi ý rằng nếu ta tách riêng phần xu hướng và mô tả nó bằng một mô hình đơn giản, mô hình học máy sẽ có nhiều “không gian” hơn để học phần residual phức tạp.



Hình 11: Trend Analysis

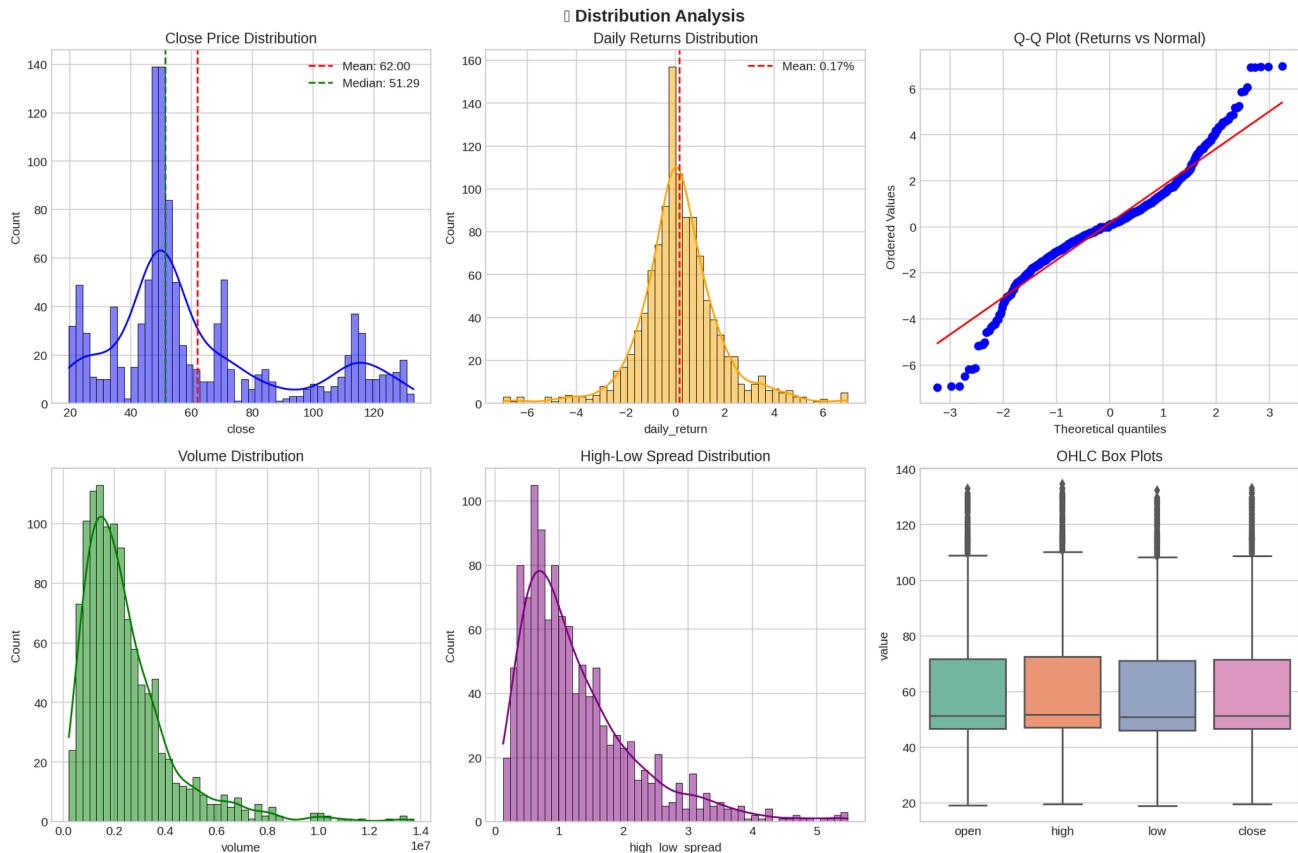
Trong code, bước đầu tiên là chuyển giá đóng cửa sang hệ log:

$$\log(P_t) = \log(\text{close}_t + \varepsilon)$$

với ε rất nhỏ để tránh lỗi số. Sau đó, ta fit một mô hình tuyến tính đơn giản:

$$\log(P_t) \approx at + b,$$

với $t = 0, 1, 2, \dots$. Điều thú vị là trong biểu đồ Trend Analysis, mô hình tuyến tính này khớp rất tốt với log-price thực tế — xác nhận rằng backbone tuyến tính là lựa chọn hợp lý.



Hình 12: Q-Q Plot

1.1.2 Mở rộng xu hướng vào tương lai: định nghĩa future_ret_math

Khi đường xu hướng quá khứ đã được học, ta kéo dài nó sang tương lai thêm $H = 100$ ngày. Từ đó, log-return theo xu hướng được định nghĩa:

$$\text{future_ret_math}(k) = \widehat{\log P}_{t_0+k+1} - \widehat{\log P}_{t_0+k},$$

với t_0 là ngày cuối của dữ liệu lịch sử.

Những giá trị này đóng vai trò như “nhịp tim dài hạn” của thị trường: đều đặn, ổn định, và hoàn toàn không bị chi phối bởi các cú sốc ngẫu nhiên. Từ biểu đồ phân phối (Distribution Histogram) và đồ thị Q-Q Plot, có thể thấy rằng chuỗi lợi nhuận ngày (daily returns) của FPT sai lệch đáng kể so với phân phối chuẩn. Histogram kết hợp KDE cho thấy phần lớn giá trị lợi nhuận tập trung quanh mức 0, nhưng hai đuôi của phân phối lại kéo dài bất thường. Hiện tượng heavy-tail này phản ánh các cú sốc thị trường, biến động do tin tức, hoặc những phiên giao dịch thanh khoản thấp tạo ra các thay đổi đột ngột. Đồ thị Q-Q giúp làm rõ hơn đặc điểm này 12: các quantile ở vùng trung tâm bám khá sát đường tham chiếu chuẩn, nhưng càng về hai đầu thì các điểm dữ liệu càng lệch mạnh, cho thấy sự xuất hiện của nhiều outlier và các sự kiện cực trị mà mô hình khó học một cách ổn định.

Trong bối cảnh đó, **Math Backbone** đóng vai trò như một “lõi toán học sạch” giúp ổn định cấu trúc dài hạn của chuỗi giá. Hoạt động trực tiếp trên chuỗi log-price backbone tách được hai thành phần riêng biệt của quá trình hình thành giá:

- **Xu hướng dài hạn mượt**, phản ánh quỹ đạo tăng trưởng mà thị trường duy trì trong nhiều năm.

- **Biến động nhiễu ngắn hạn**, chính là nguồn gốc của heavy-tail và các giá trị cực trị trong phân phối returns.

Khi trực quan hóa đường xu hướng log-price cùng biểu đồ giá và phân phối lợi nhuận, ta nhận thấy backbone loại bỏ hầu hết méo dạng do các giá trị cực trị gây ra, tạo nên một phiên bản “clean return” mô tả phần drift cấu trúc mà mô hình dự báo nên tin tưởng. Điều này đặc biệt quan trọng đối với các bài toán dự báo dài hạn dạng đệ quy (ví dụ: 100 ngày), nơi mà độ nhạy quá mức với tail có thể khiến đường dự báo trở nên giật mạnh và phi thực tế.

Nhờ việc loại bỏ cấu trúc dài hạn thông qua backbone, mô hình học máy chỉ cần học phần *residual*, vốn ít nhiễu hơn, gần với phân phối chuẩn hơn và dễ mô hình hóa hơn rất nhiều. Kết quả là pipeline dự báo trở nên ổn định hơn, dễ giải thích hơn và phù hợp hơn với bản chất thống kê của dữ liệu tài chính.

1.1.3 Khi câu chuyện toán học bước vào thực thi code

Tất cả ý tưởng trên được hiện thực hoá trong hàm `build_raw_base_path_hybrid`.

Dưới đây là đoạn code cốt lõi cho phần Math Backbone:

```

1 # 1) Fit a linear trend on log-price
2 df_state["close_log"] = np.log(df_state["close"] + 1e-8)
3 N_hist = len(df_state)
4
5 time_idx_hist = np.arange(N_hist).reshape(-1, 1)
6 y_log_hist = df_state["close_log"].values.reshape(-1, 1)
7
8 lr_trend = LinearRegression()
9 lr_trend.fit(time_idx_hist, y_log_hist)
10
11 # 2) Extend the trend into the future
12 total_len = N_hist + total_days
13 time_idx_full = np.arange(total_len).reshape(-1, 1)
14
15 trend_log_full = lr_trend.predict(time_idx_full).flatten()
16
17 # 3) Compute math backbone returns
18 math_rets_forecast = np.zeros(total_days, dtype=float)
19 for k in range(total_days):
20     base_idx = N_hist - 1 + k
21     if base_idx + 1 < len(trend_log_full):
22         math_rets_forecast[k] = (
23             trend_log_full[base_idx + 1] - trend_log_full[base_idx]
24         )
25     else:
26         # fallback: repeat last return
27         math_rets_forecast[k] = math_rets_forecast[k-1] if k > 0 else 0.0

```

Code Listing 1: Math Backbone implementation excerpt

Ba bước này tương ứng hoàn toàn với câu chuyện toán học đã trình bày:

- 1. Fit xu hướng dài hạn trên log-price
- 2. Kéo dài xu hướng vào tương lai
- 3. Tính chuỗi log-return theo xu hướng

Và chính `math_rets_forecast[k]` là hiện thân của `future_ret_math(k)` trong code.

1.1.4 Backbone kết hợp với residual: từ lý thuyết đến đường giá dự báo

Trong vòng lặp dự báo từng ngày, backbone không hoạt động độc lập. Mỗi bước, mô hình:

- 1. Tạo các đặc trưng (bao gồm STL features, volatility, technical indicators)
- 2. Chuẩn hoá đặc trưng
- 3. XGBoost dự đoán residual return
- 4. Kết hợp backbone + residual:

$$\text{final_ret}(k) = \text{math_ret}(k) + \text{resid_pred}(k)$$

Sau đó giá được cập nhật:

$$\log P_{t+1} = \log P_t + \text{final_ret}(k).$$

Phần code tương ứng:

```

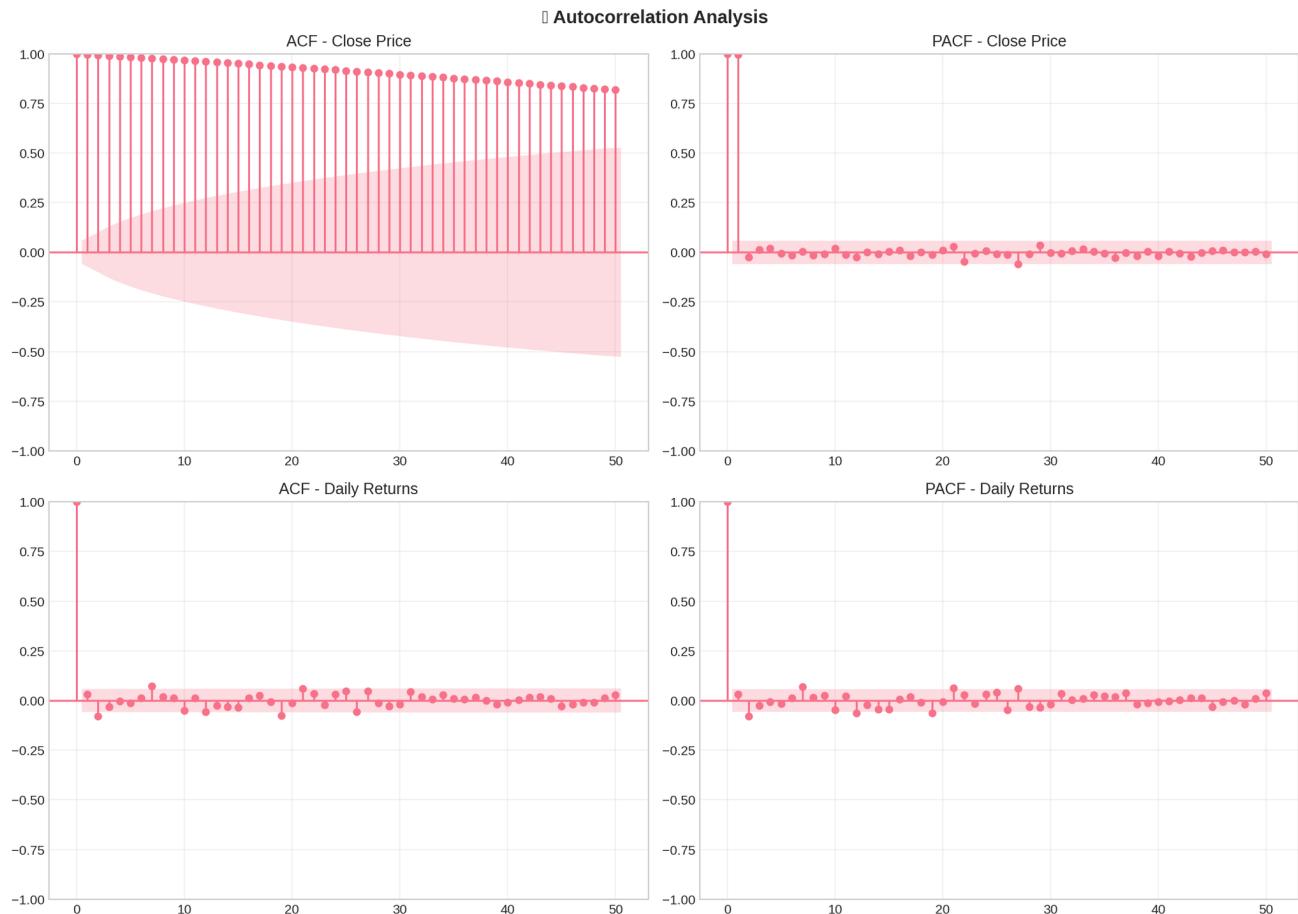
1 X_last_s = scaler_X.transform(X_last)
2 resid_pred = float(xgb.predict(X_last_s)[0])
3
4 math_ret = float(math_rets_forecast[step_idx])
5 final_ret = math_ret + resid_pred # hybrid return
6
7 last_log = df_state["close_log"].iloc[-2]
8 next_log = last_log + final_ret
9 next_price = float(np.exp(next_log))

```

Code Listing 2: Combining math backbone and residual prediction

Điều này phản ánh chính xác bản chất thị trường ta quan sát từ các biểu đồ kỹ thuật:

- MACD cho thấy các giai đoạn momentum shift mạnh
- RSI nhiều lần chạm vùng quá mua/quá bán
- Rolling Volatility nhảy vọt tại các điểm thị trường nhạy cảm ư
- ACF/PACF của returns gần như nhiều trống. Quan sát trực tiếp trên các biểu đồ ACF/PACF trong Hình 13 cho thấy sự khác biệt rõ rệt giữa hành vi của giá đóng cửa và chuỗi lợi nhuận ngày. ACF của giá duy trì giá trị cao trong nhiều độ trễ liên tiếp, trong khi PACF chỉ có một spike mạnh tại lag 1. Mẫu hình này là đặc trưng của một chuỗi không dừng có xu hướng dài hạn, cung cấp nhận định rằng log-price mang cấu trúc drift ổn định mà Math Backbone có thể mô tả hiệu quả. Ngược lại, ACF và PACF của daily returns đều nằm hoàn toàn trong khoảng tin cậy, không xuất hiện bất kỳ spike có ý nghĩa thống kê nào. Điều này chỉ ra rằng returns gần như là một chuỗi nhiều trống, không có tương quan tuyến tính qua thời gian. Vì returns không mang thông tin xu hướng, chúng không phải đối tượng phù hợp để dự báo trực tiếp. Do đó, việc tách phần cấu trúc dài hạn thông qua backbone trở thành bước thiết yếu, giúp mô hình học máy chỉ cần học phần residual ít nhiều và dễ mô hình hóa hơn.



Hình 13: Autocorrelation Analysis

Những chuyển động này không thể được mô tả bằng backbone tuyến tính — nhưng residual learning lại có thể.

1.1.5 Kết luận: một đường thẳng giữa thế giới xao động

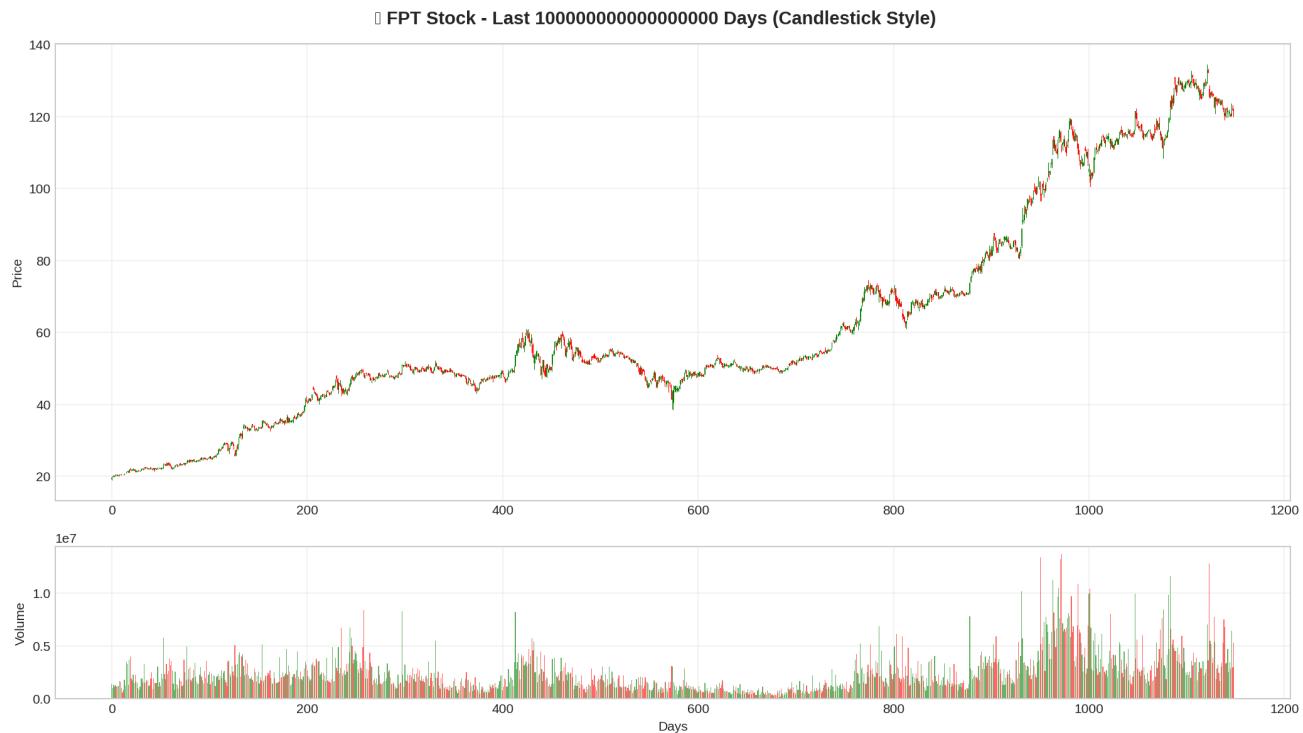
Math Backbone không phải là phần phíc tạp nhất của pipeline, nhưng nó là phần ổn định nhất. Nó đại diện cho phần chuyển động mà ta có thể tin tưởng: xu hướng dài hạn, nhất quán, được thấy rõ trong các biểu đồ trend và decomposition. Tách xu hướng dài hạn cho backbone giúp mô hình học máy tập trung vào phần còn lại — phần nhiễu động giàu thông tin mà MACD, RSI, volatility và return distribution đều cho thấy là rất phi tuyến và khó đoán. Nhờ sự phân vai rõ ràng này, mô hình hybrid trở nên:

- Ổn định hơn backbone đơn thuần
- Chính xác hơn ML đơn thuần
- Và phản ánh đúng “hơi thở” của thị trường: một thế giới đầy xao động, nhưng vẫn có một dòng chảy dài hạn dẫn dắt.

1.2 Lớp 2: ML Residual

Khi nhìn vào toàn bộ bức tranh EDA của cổ phiếu FPT, ta nhận ra thị trường vốn dĩ không vận hành chỉ theo một quy luật duy nhất. Những đường giá mượt mà trong hình Trend Analysis cho ta

thấy một quỹ đạo dài hạn khá rõ ràng: log-price gần như tuyến tính, và backbone có thể mô tả được xu hướng này một cách gọn ghẽ. Nhưng chỉ cần tiến lại gần hơn, quan sát những cây nến trong hình Candlestick Overview 14 hay biến động volume trong phần Trading Volume, ta lập tức cảm nhận một thế giới hoàn toàn khác: một nhịp điệu đầy những cú giật, cú rơi, cú bật, mà xu hướng dài hạn không thể giải thích nổi.



Hình 14: Candle Stick

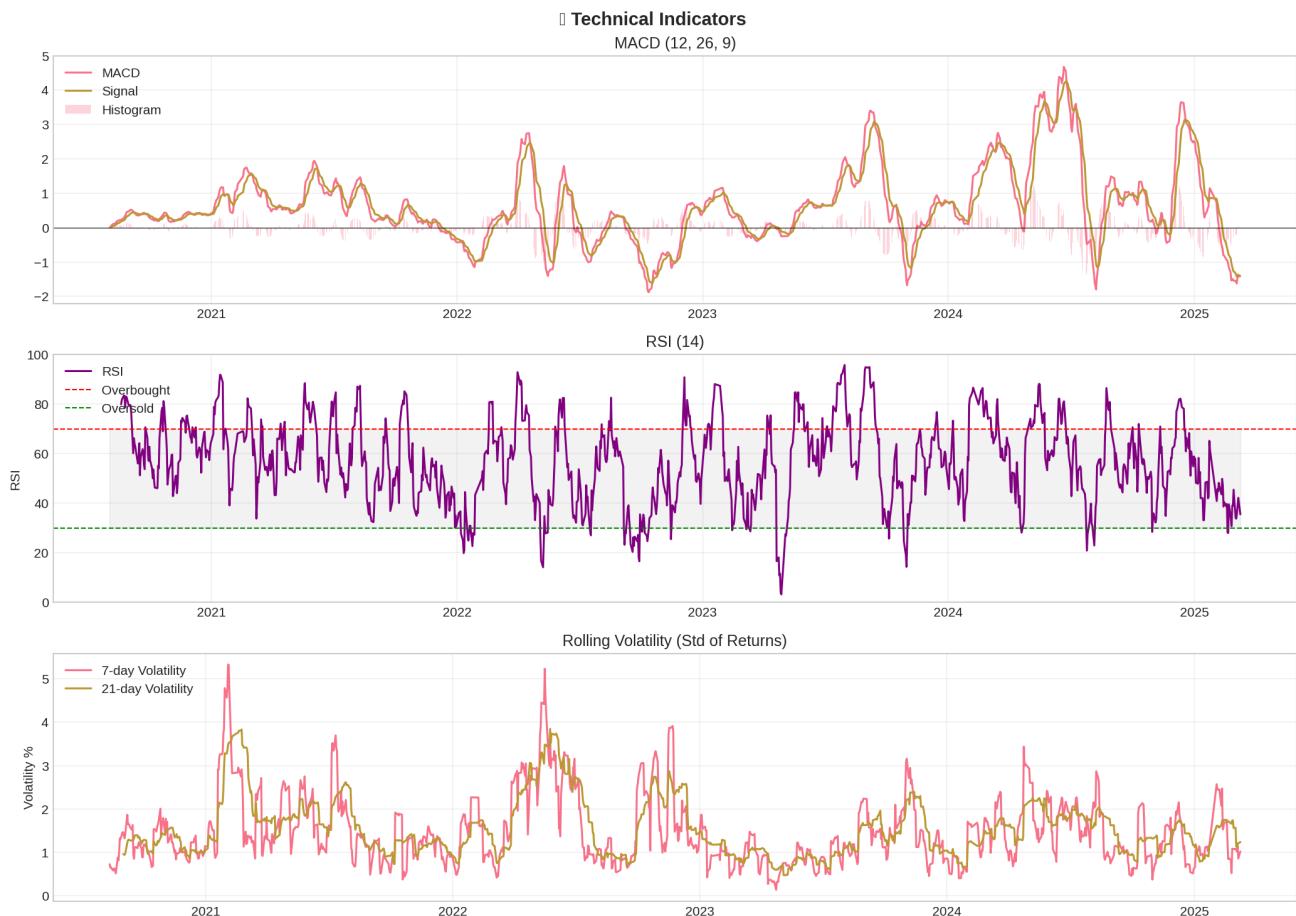
Các biểu đồ phân phối trong Hình 12 càng khắc họa rõ sự hỗn độn này: daily returns có đuôi dày, lệch mạnh và chứa vô số outliers. Q–Q Plot thì gần như “rụng rời” ở hai đầu, chứng tỏ phân phối thực tế không tuân theo luật Gaussian cổ điển. Nhưng cú đánh mạnh nhất lại đến từ Hình 13: ACF và PACF của returns phảng lặng như tờ, toàn bộ điểm dữ liệu rơi gọn vào vùng tin cậy. Nói cách khác, returns là một chuỗi nhiễu trắng với các đặc điểm không tự tương quan, không linear structure, không AR, không MA.

Điều này mang một hàm ý cực kỳ quan trọng: **không mô hình tuyến tính nào có thể dự báo trực tiếp returns**. Và đây chính là lý do ta phải tách xu hướng bằng Math Backbone trước khi làm bất kỳ điều gì khác.

Khi backbone đã bóc tách phần drift dài hạn, phần residual còn lại không còn là hỗn loạn vô nghĩa. Quan sát Hình 15, ta thấy residual chứa những nhịp điệu ngắn hạn, những cú xoay mean-reverting, và những đợt volatility clustering mà thị trường thường tạo ra theo tâm lý nhà đầu tư. Các chỉ báo kỹ thuật trong Hình 16 như RSI, MACD, volatility cũng cho thấy những dao động mạnh trong biên độ ngắn hạn và không đủ lớn để thay đổi trend, nhưng đủ quan trọng để ảnh hưởng đến dự báo 100 ngày khi thực hiện mô phỏng đề quy.



Hình 15: Decomposition



Hình 16: Technical Indicators

Đây chính là vùng đất của mô hình phi tuyến. Nhưng chọn mô hình nào?

1.2.1 Vì sao XGBoost phù hợp với bản chất dữ liệu residual hơn các mô hình khác?

Dữ liệu residual sau backbone không giống một time series truyền thống. Nó giống một tập hợp các quy tắc cục bộ (*local market behaviors*) hơn là một quá trình tạo sinh toàn cục. Và XGBoost có những đặc tính gần như sinh ra để học dạng dữ liệu này:

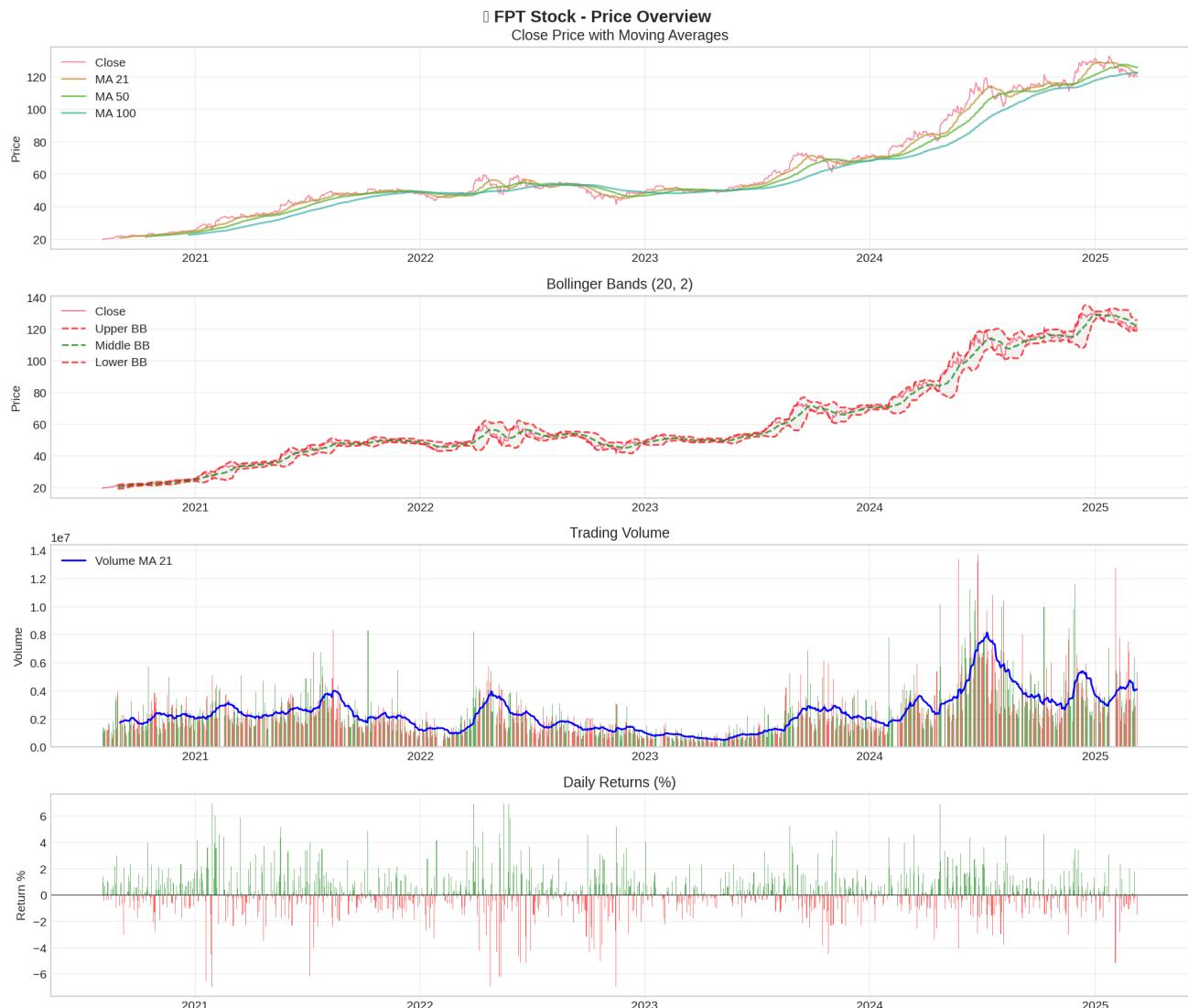
1. **XGBoost mô hình hóa mạnh mẽ các quan hệ phi tuyến.** Các cú giật giá, cú hồi, cú squeeze, volatility spike—không mô hình tuyến tính nào miêu tả nổi, nhưng XGBoost có thể phân tách không gian đặc trưng qua các cây quyết định và tự tìm ra những “luật nhỏ” kiểu:

if volatility_21d > threshold and RSI < 35 \Rightarrow residual dương trong 2–3 phiên

2. **XGBoost khai thác tương tác giữa nhiều feature một cách tự nhiên.** Correlation heatmap ở Hình 18 cho ta một thông điệp rất rõ: các biến trong bộ dữ liệu FPT gần như **không có quan hệ tuyến tính mạnh**. Điều này thể hiện qua việc hầu hết hệ số tương quan đều nằm trong khoảng từ -0.1 đến 0.6 , không có cặp biến nào “đi chung” theo một đường thẳng rõ rệt. Nhưng chính vì heatmap im lặng như vậy nên ta càng phải cẩn thận. Nếu nhìn kĩ volume không tương quan trực tiếp với daily return, nhưng khi nhìn vào chart volume trong Hình 17, ta thấy volume tăng mạnh thường đi kèm với tăng biến động và các mẫu nến mở rộng—tức là **volume ảnh hưởng**

đến momentum theo cách phi tuyến. Tương tự, high-low spread gần như không tương quan với giá đóng cửa nhưng histogram spread ở Hình 12 lại cho thấy các phiên có spread lớn thường là lúc thị trường hoảng loạn hoặc hưng phấn và chính những phiên như vậy lại tạo ra residual lớn trong mô hình xu hướng. Đây là dạng quan hệ:

if spread $> \tau \Rightarrow$ residual biến động mạnh,



Hình 17: Price Overview

mà không mô hình tuyến tính nào có thể mô tả bằng một hệ số tương quan duy nhất. Hay ngay cả RSI—nhìn heatmap thì gần như không liên hệ với giá hay volume. Nhưng biểu đồ RSI trong Hình 16 lại kể một câu chuyện khác: RSI phản ứng rất nhạy với các giai đoạn thị trường tăng tốc hoặc chững lại. Điều này tạo nên các ngữ cảnh “bán quá mức” và “mua quá mức” mà các mô hình tuyến tính hoàn toàn không thể nhận diện:

if RSI $< 30 \Rightarrow$ mean-reversion signal mạnh.

Những mối quan hệ kiểu điều kiện—"nếu A cao và B thấp thì C thay đổi mạnh"—không khi nào xuất hiện trong heatmap, nhưng lại vô cùng quan trọng đối với thị trường tài chính. Và đây chính là nơi XGBoost vượt trội: mỗi cây quyết định của nó chính là một "bộ tách phi tuyến" dễ dàng tạo ra các ngưỡng:

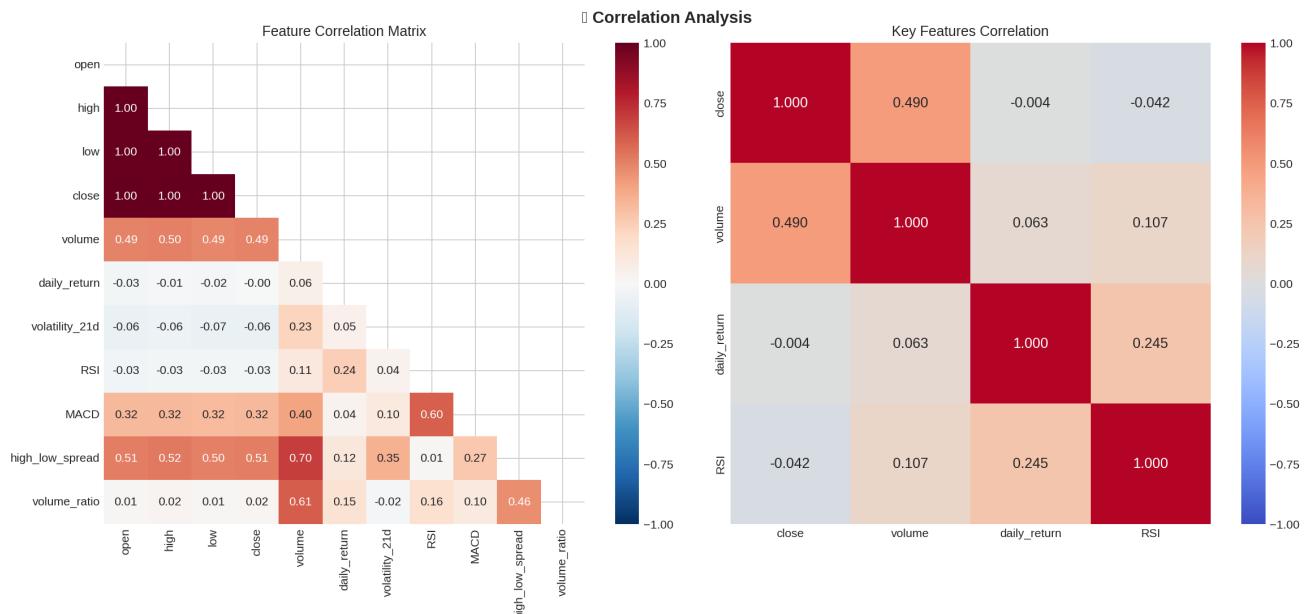
if volume spike and volatility rising \Rightarrow residual dương,

hoặc:

if momentum giảm nhưng giá vẫn trên MA21 \Rightarrow residual âm nhẹ.

XGBoost không cố ép dữ liệu tài chính vào một khung gian tuyến tính như hồi quy. Thay vào đó, nó tự tìm ra các tương tác phi tuyến mà heatmap không hề thể hiện và biến chúng thành những luật cục bộ có tính dự báo. Chính nhờ khả năng xử lý các interaction splits này mà XGBoost phù hợp tự nhiên với cấu trúc dữ liệu residual—một tập hợp nhỏ tinh tế của hàng loạt "tín hiệu phi tuyến" mà thị trường tạo ra mỗi ngày.

3. **Hoàn hảo cho dữ liệu short-term, noisy, local.** Candlestick chart Hình 14 cho thấy từng phiên có hành vi rất "địa phương": gap nhỏ, breakout, false break, doji, long wick. Đây không phải loại dữ liệu bạn đưa vào LSTM hay ARIMA rồi mong nó tự hiểu. Nhưng XGBoost lại cực kỳ giỏi trong việc nắm bắt local rules.
4. **Không yêu cầu stationarity.** Backbone đã làm phẳng phần trend, residual đôi khi vẫn hơi lệch mean hoặc biến thiên theo thời kỳ. ARIMA hoặc VAR sẽ thất bại ngay tại cửa kiểm tra stationarity. XGBoost thì không bận tâm.
5. **Chịu nhiễu tốt và ít overfit hơn neural networks trên dataset vừa và nhỏ.** Dữ liệu FPT dù kéo dài nhiều năm nhưng ở granular daily không đủ lớn cho deep learning. XGBoost vừa đủ mạnh, vừa đủ "thận trọng".



Hình 18: Correlation Heatmap

1.2.2 Câu chuyện cuối cùng: sự kết hợp giữa cái lớn và cái nhỏ.

Trend của log-price—như ta thấy trong hình Trend Analysis—chạy rất mượt. Residual thì lại nhảy múa trong biên độ nhỏ mỗi ngày. Một mô hình tốt cần mô tả được **cả hai thế giới**. Math Backbone giữ phần “lõi” lớn: quỹ đạo tăng trưởng dài hạn. XGBoost giữ phần “hồn” nhỏ: dao động tinh tế mà trader nào nhìn chart cũng thấy. Và chính sự kết hợp này tạo ra một pipeline dự báo vừa ổn định và lại vừa nhạy bén, điều mà bất kỳ mô hình đơn lẻ nào cũng không làm được.

1.2.3 Thực thi mô hình XGBoost trên residual: từ ý tưởng đến code

Sau khi xây dựng Math Backbone và tạo được chuỗi residual

$$r_t^{\text{resid}} = r_t^{\text{future}} - r_t^{\text{math}},$$

bước tiếp theo là huấn luyện một mô hình phi tuyến để học phần nhiễu có cấu trúc còn sót lại. Dựa trên những quan sát từ EDA (phân phối heavy-tail, ACF/PACF gần nhiều trảng, các tương tác phi tuyến giữa volatility, volume, RSI, spread trong heatmap tương quan), Ta triển khai XGBoost để mô hình hóa trực tiếp residual:

```

1 def train_xgb_on_dfmodel(df_model: pd.DataFrame, feature_cols: List[str]) -> Tuple[XGBRegressor, StandardScaler, float, float]:
2     """
3     Train XGBoost on residual returns: resid_ret = future_ret - math_ret.
4     """
5     if len(df_model) < 200:
6         raise ValueError(f"[XGB] Too few samples: {len(df_model)}")
7
8     # 1) Extract features and residual target
9     X_all = df_model[feature_cols].values.astype(np.float32)
10    y_resid = df_model[["resid_ret"]].values.astype(np.float32)
11
12    # 2) Time-ordered split: 80% train, 10% val, 10% test
13    N = len(df_model)
14    train_ratio = 0.8
15    val_ratio = 0.1
16
17    train_end = int(N * train_ratio)
18    val_end = int(N * (train_ratio + val_ratio))
19
20    X_train = X_all[:train_end]
21    y_train = y_resid[:train_end]
22    X_val = X_all[train_end:val_end]
23    y_val = y_resid[train_end:val_end]
24    X_test = X_all[val_end:]
25    y_test = y_resid[val_end:]
26
27    # 3) Standardize inputs (fit only on train)
28    scaler_X = StandardScaler().fit(X_train)
29    X_train_s = scaler_X.transform(X_train)
30    X_val_s = scaler_X.transform(X_val)
31    X_test_s = scaler_X.transform(X_test)
32
33    # 4) Configure and train XGBoost on residuals
34    xgb = XGBRegressor(
35        n_estimators=450,
36        max_depth=4,
37        learning_rate=0.03,
38        subsample=0.9,
39        colsample_bytree=0.9,
```

```

40     reg_lambda=2.0,
41     min_child_weight=3,
42     objective="reg:squarederror",
43     random_state=SEED,
44 )
45 xgb.fit(X_train_s, y_train)
46
47 # 5) Evaluate on train / val / test using MSE and MAE
48 def eval_block(name, X_s, y_true):
49     if len(y_true) == 0:
50         print(f"{name}: empty")
51         return
52     y_pred = xgb.predict(X_s)
53     mse = mean_squared_error(y_true, y_pred)
54     mae = mean_absolute_error(y_true, y_pred)
55     print(f"{name} (resid) -> MSE: {mse:.6e} | MAE: {mae:.6e}")
56
57 print(f"==== XGB RESIDUAL MODEL ({len(df_model)} samples) ===")
58 eval_block("Train", X_train_s, y_train)
59 eval_block("Val", X_val_s, y_val)
60 eval_block("Test", X_test_s, y_test)
61
62 # 6) Compute residuals after XGB to measure remaining noise
63 y_pred_all = xgb.predict(scaler_X.transform(X_all))
64 residuals = y_resid - y_pred_all
65 resid_std = float(np.std(residuals, ddof=1))
66 resid_mean = float(np.mean(residuals))
67
68 print(f"[RESID] std={resid_std:.6e}, mean={resid_mean:.6e}")
69 return xgb, scaler_X, resid_std, resid_mean

```

Code Listing 3: Training XGBoost on residual returns

Đoạn code trong Listing 3 hiện thực hoá chính xác ý tưởng “backbone giữ xu hướng, XGBoost bắt nhiễu phi tuyến”:

- **Bước 1–2:** Chuỗi residual `resid_ret` được dùng làm mục tiêu huấn luyện, trong khi các feature đầu vào được trích từ những đặc trưng giàu ý nghĩa mà EDA đã gợi ý (volatility, volume, RSI, spread, ...). Việc chia theo thứ tự thời gian (80%–10%–10%) phản ánh đúng bản chất dự báo trong thực tế: mô hình chỉ được phép học từ quá khứ.
- **Bước 3:** Chuẩn hoá feature bằng `StandardScaler` giúp XGBoost ổn định hơn khi các biến có đơn vị và thang đo rất khác nhau (giá, phần trăm, khối lượng).
- **Bước 4:** Cấu hình XGBoost với nhiều cây nhỏ (`max_depth=4`) cho phép mô hình nắm bắt các tương tác phi tuyến cục bộ giữa RSI, volatility, volume, high-low spread như những gì quan sát được trong các biểu đồ kỹ thuật và heatmap tương quan.
- **Bước 5:** Việc in ra MSE/MAE cho train/validation/test không chỉ dùng để chọn hyperparameter mà còn là cách kiểm tra xem mô hình có thực sự khai thác được cấu trúc còn lại trong residual hay không. Nếu XGBoost chỉ gặp nhiễu trắng thuần túy, sai số trên tập test sẽ gần tương đương với một mô hình “đoán ngẫu nhiên”.
- **Bước 6:** Sau khi trừ đi dự báo của XGBoost, ta tính lại độ lệch chuẩn và trung bình của residual mới. Nếu std giảm đáng kể và mean ≈ 0 điều đó cho thấy mô hình đã loại bỏ phần nhiễu có cấu trúc (volatility clustering, hiệu ứng RSI, volume spike, ...), để lại một chuỗi residual gần hơn với nhiễu trắng thực sự. Chuỗi này sau đó sẽ được sử dụng trong lớp Pricing Layer như một proxy cho “rủi ro còn lại”.

1.3 Lớp 3: Pricing Layer

Sau khi đi qua hai lớp đầu tiên, mô hình cho ta:

- t : xu hướng dài hạn (Math Backbone),
- t : dự báo “thô” sau khi đã cộng residual do mô hình ML học được.

Tuy nhiên, đường t đôi khi có thể quá “hung hăng”: tăng/giảm quá mạnh chỉ trong vài phiên, hoặc trôi xa khỏi xu hướng dài hạn. Vì vậy, thay vì dùng trực tiếp t , chúng tôi xây dựng một lớp kiểm soát thứ ba – *Pricing Layer* – đóng vai trò như một “bộ điều khiển vật lý” đặt lên đường giá.

2 Deep Dive vào Pricing Layer

2.1 Cơ chế vật lý

2.1.1 Clipping

Clipping là cơ chế đầu tiên trong Pricing Layer, đóng vai trò giống như **giới hạn tốc độ** trong một hệ động lực. Ý tưởng rất đơn giản: dù mô hình ML (Hybrid Backbone + XGBoost) có dự đoán một cú nhảy giá rất lớn, ta vẫn đặt câu hỏi:

“Trên thực tế, FPT có thể di chuyển nhanh đến mức đó chỉ trong một phiên không?”

Trên thị trường chứng khoán Việt Nam, đặc biệt với cổ phiếu vốn hóa lớn như FPT, biến động ngày thường nằm trong vùng 1–3%, các cú sốc 5–7% đã là cực đoan và thường gắn với tin tức rất lớn. Do đó, nếu để mô hình ML tự do trả ra các log-return tương đương với +15% hoặc -20% một ngày, dự báo sẽ trở nên *phi thực tế*.

Trong Pricing Layer, cơ chế Clipping được hiện thực hoá bằng đoạn mã:

```
1 # raw_ret: log-return do hybrid model sinh ra
2 pred_ret = np.clip(raw_ret, -ret_clip, ret_clip)
```

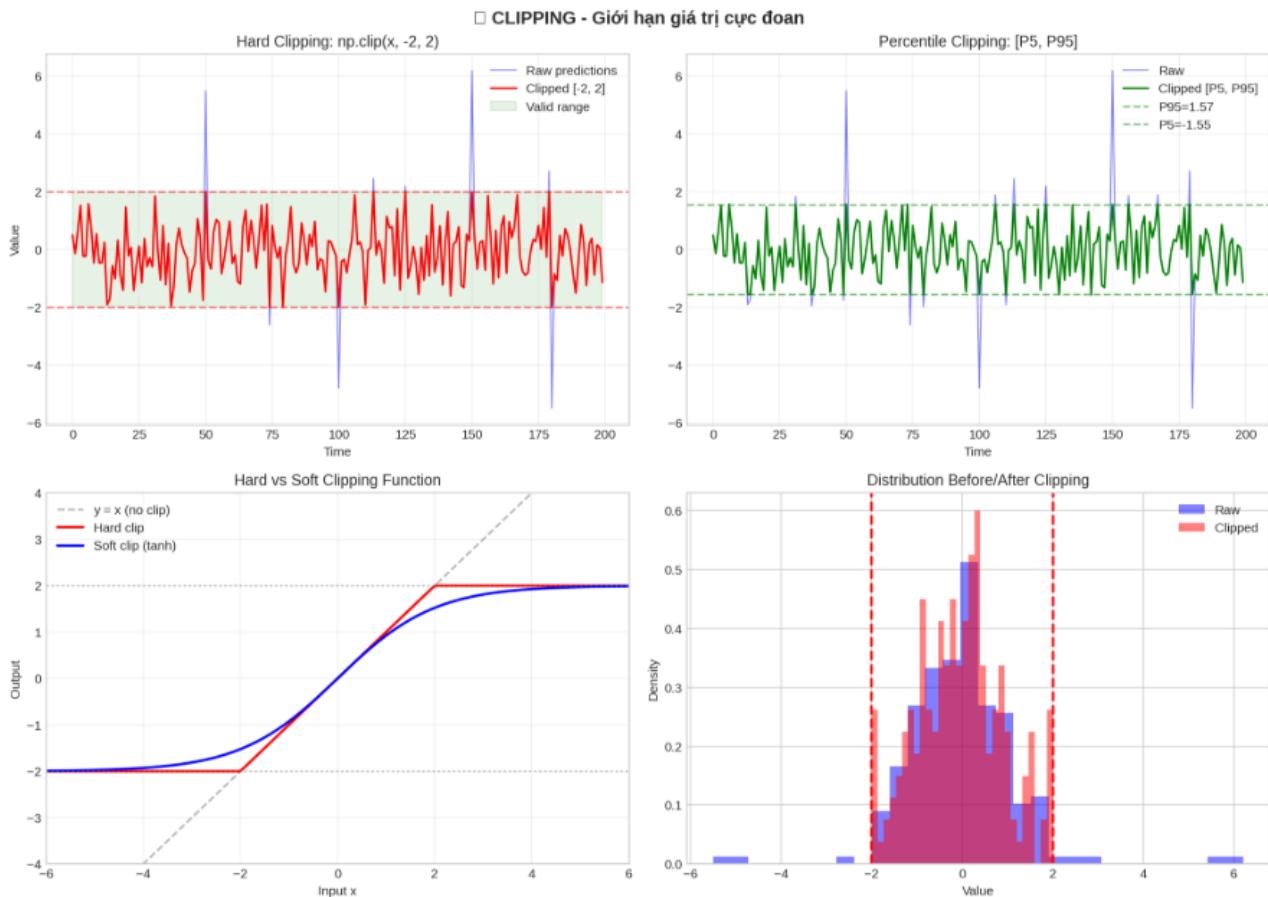
Trong đó, `ret_clip` không cố định mà phụ thuộc vào:

- **Regime thị trường (BULL / BEAR / SIDEWAYS):** Trong BULL, thị trường thường chấp nhận những cú tăng mạnh hơn, nên biên dương có thể được nới rộng nhẹ. Trong BEAR, các cú sụt giảm mạnh lại phổ biến hơn, nhưng ta vẫn hạn chế để tránh mô hình dự đoán các pha sập giá phi lý.
- **Mức độ biến động hiện tại (vol_ratio):** Nếu volatility 20 ngày gần đây thấp hơn nhiều so với lịch sử, không có lý do gì để cho phép mô hình “nhảy” với biên độ quá lớn.

Ta có thể viết dưới dạng công thức:

$$\text{ret_clip} = \text{base_ret_clip} \times \text{clip_scale_regime} \times \text{clip_scale_vol}.$$

Hình 19 minh họa ý nghĩa của Clipping: đường màu xám thể hiện các log-return mà mô hình ML muốn dự đoán, trong khi đường màu xanh dương là log-return *sau khi đã được clip*, với đỉnh/dáy bị cắt bớt.



Hình 19: Minh họa cơ chế Clipping: log-return cực đoan của mô hình được “cắt” về một ngưỡng biên độ hợp lý hơn.

Về mặt trực giác, Clipping giống như việc *gắn bô giới hạn tốc độ* lên dự báo:

- Mô hình vẫn được phép phản ứng với thông tin mới (giá có thể tăng/giảm nhanh hơn trong BULL hoặc giai đoạn biến động cao).
- Nhưng mọi cú nhảy đều phải nằm trong một *khung vật lý hợp lý* so với lịch sử biến động của FPT và mặt bằng thị trường Việt Nam.

Nhờ đó, Clipping là lớp phòng tuyến đầu tiên, ngăn cho Pricing Layer không “quá tin” vào những dự báo cực đoan từ mô hình ML, đặc biệt là trong những giai đoạn dữ liệu nhiễu hoặc có outlier mạnh.

2.1.2 Damping

Nếu Clipping là *giới hạn tốc độ*, thì Damping là **lực ma sát** khi mô hình tiến xa về tương lai. Trong vật lý, một con lắc dao động trong môi trường có ma sát sẽ dần giảm biên độ và tiến về trạng thái tĩnh. Trong Pricing Layer, ta áp dụng một nguyên lý tương tự lên log-return dự báo.

Giả sử mô hình ML sinh ra một chuỗi log-return $\{r_t\}_{t=1}^H$ cho 100 ngày tới. Về lý thuyết, nếu không có Damping, biên độ của các r_t có thể giữ nguyên từ ngày 1 đến ngày 100, tạo ra những đường giá tương lai quá “mạnh tay” so với thực tế. Nhưng trong hành vi thị trường, đặc biệt với cổ phiếu FPT:

- Tác động của một tin tức hoặc một pha FOMO thường giảm dần sau vài tuần.

- Một tín hiệu kỹ thuật ở ngày hôm nay không nên chi phối quá mạnh kết quả sau 3 tháng.

Trong Pricing Layer, Damping được hiện thực bằng cách nhân log-return với một hệ số giảm dần theo thời gian:

```

1 lambda_damp = np.log(2.0) / float(max(int(
2     pricing.half_life_days * damp_scale_regime), 1))
3
4 for step_idx in range(total_days):
5     raw_ret = float(raw_rets[step_idx])
6     pred_ret = np.clip(raw_ret, -ret_clip, ret_clip)
7
8     # Damping theo half-life
9     scale = np.exp(-lambda_damp * step_idx)
10    pred_ret *= scale

```

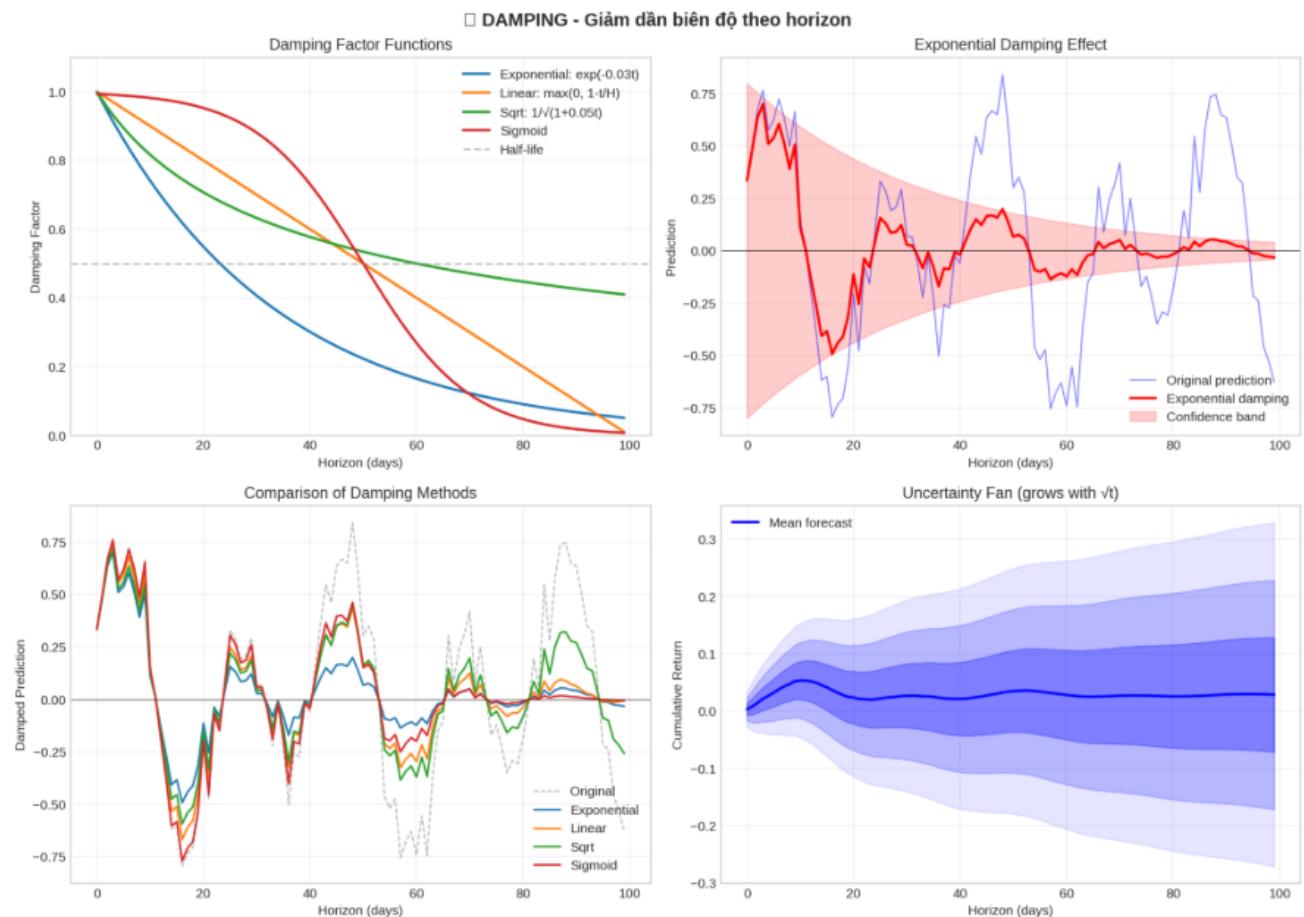
Ở đây:

$$\lambda_{\text{damp}} = \frac{\ln 2}{\text{half_life_days} \times \text{damp_scale_regime}},$$

và half_life_days có thể hiểu như “**thời gian để biên độ tín hiệu giảm đi một nửa**”.

- Nếu half_life_days = 40, thì sau khoảng 40 phiên, một cú xung rất mạnh ở đầu forecast sẽ chỉ còn ~ 50% tác động.
- Sau 80 phiên, nó chỉ còn ~ 25%, và cứ thế bị “tắt dần”.

Hình 20 minh họa một chuỗi log-return dự báo và phiên bản đã được Damping: tín hiệu ban đầu vẫn được giữ lại, nhưng càng xa về tương lai, biên độ càng co lại.



Hình 20: Minh họa Damping: log-return dự báo được nhân với hệ số giảm dần, giúp tín hiệu ngắn hạn không chi phối quá mạnh phần cuối của đường forecast.

Trong bối cảnh thị trường Việt Nam:

- Các pha tin tức / dòng tiền thường tạo ra sóng ngắn 20–40 phiên.
- Các dự án dài hạn của doanh nghiệp (như FPT) lại tạo ra nền tăng trưởng dần dần trong nhiều năm.

Damping giúp Pricing Layer *tôn trọng* cả hai: tín hiệu ngắn hạn vẫn được phản ánh ở đoạn đầu forecast, nhưng vai trò của nó giảm bớt khi mô hình tiến dần đến cuối 100 ngày, nhường chỗ cho xu hướng dài hạn (được thể hiện qua TREND và Math Backbone).

2.1.3 Mean Reversion

Mean Reversion là cơ chế “lực đàn hồi” của Pricing Layer, kéo giá dự báo quay trở lại một **vùng cân bằng** khi nó đi quá xa. Trong vật lý, đây chính là lực đàn hồi Hooke:

$$F = -k(x - x_0),$$

trong đó x_0 là vị trí cân bằng và k là độ cứng lò xo.

Trong mô hình của chúng tôi, “vị trí cân bằng” không phải là một điểm đơn lẻ, mà là một dải *fair value band* quanh một mức tham chiếu *fair_level*:

$$\text{upper} = \text{fair_level} \cdot \text{fair_up_mult}, \quad \text{lower} = \text{fair_level} \cdot \text{fair_down_mult}.$$

Ta có thể hiểu:

- Nếu giá dự báo vượt quá *upper*, mô hình coi đó là trạng thái “quá nóng” so với giá trị hợp lý.
- Nếu giá dự báo rơi dưới *lower*, mô hình coi đó là trạng thái “quá rẻ”.

Cơ chế Mean Reversion được hiện thực bằng đoạn mã:

```

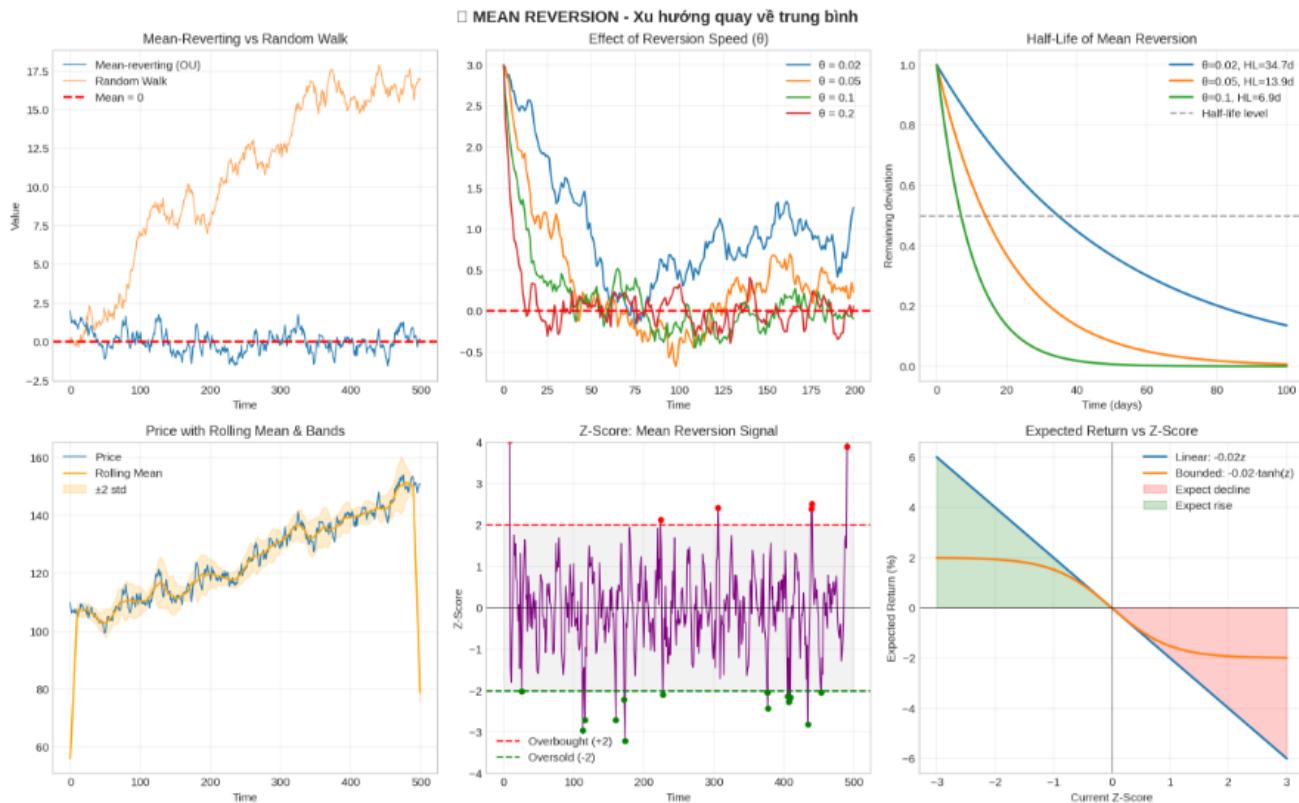
1 if step_idx >= pricing.mean_revert_start:
2     upper = fair_level * fair_up_mult
3     lower = fair_level * fair_down_mult
4
5     if (next_price > upper) and (not is_strong_uptrend):
6         alpha_up = alpha_base * (0.7 if regime == "BULL" else 1.0)
7         next_price = (1 - alpha_up) * next_price + alpha_up * upper
8
9     elif next_price < lower:
10        if regime == "BEAR" and is_strong_downtrend:
11            alpha_down = alpha_base * 0.7
12        else:
13            alpha_down = alpha_base
14            next_price = (1 - alpha_down) * next_price + alpha_down * lower

```

Trong đó:

- *mean_revert_start* xác định từ ngày thứ bao nhiêu trong forecast thì lực Mean Reversion bắt đầu kích hoạt (ví dụ sau 28 ngày).
- *alpha_base* điều khiển độ mạnh của lực kéo: càng lớn thì giá bị “giật” về vùng cân bằng càng nhanh.
- Các hệ số điều chỉnh theo regime (BULL/BEAR) đảm bảo rằng:
 - Trong BULL, nếu giá hơi cao hơn fair value, mô hình có thể cho phép “overpricing” kéo dài hơn một chút (giảm α).
 - Trong BEAR, nếu giá rơi dưới fair value trong khi downtrend còn rất mạnh, mô hình bớt vội vàng kéo giá lên (giảm α_{down}).

Hình 21 minh họa một đường giá dự báo *trước* và *sau* khi áp lực Mean Reversion. Ta thấy các đoạn vọt lên quá xa bị kéo nhẹ xuống, các đoạn rơi quá sâu được kéo lên, tạo thành một đường giá “điềm tĩnh” hơn và hợp lý hơn với bối cảnh FPT là cổ phiếu tăng trưởng nhưng không phải dạng đầu cơ.



Hình 21: Minh họa Mean Reversion: khi giá dự báo đi quá xa khỏi dải fair value, mô hình áp lực dần hồi kéo giá quay về vùng cân bằng.

Trong thị trường chứng khoán Việt Nam, hành vi Mean Reversion thể hiện rất rõ:

- Sau các pha hưng phấn, giá thường “nguội” dần và quay về vùng P/E hợp lý.
- Khi panic sell xảy ra, nhiều cổ phiếu tốt (như FPT) nhanh chóng được dòng tiền dài hạn mua vào, kéo giá quay về vùng hợp lý.

Bằng cách mã hoá hành vi đó vào Pricing Layer, Mean Reversion giúp mô hình:

- Không trôi dạt khỏi thực tế khi forecast dài 100 ngày.
- Tự động “tự sửa sai nhẹ” nếu ML mức độ nào đó dự đoán quá tay.
- Phản ánh đúng tính chất “blue-chip tăng trưởng” của FPT: có thể cao hơn fair value trong một thời gian, nhưng khó giữ trạng thái cực đoan mãi.

2.2 Regime-aware Pricing

Trong pipeline dự báo 100 ngày cho cổ phiếu FPT, Pricing Layer không chỉ đơn thuần là một bước hậu xử lý kỹ thuật. Nó được thiết kế để phản ánh một thực tế rất quan trọng của thị trường chứng khoán Việt Nam nói chung và FPT nói riêng: **thị trường luôn vận hành theo chế độ (regime)**, và **cùng một mức biến động** có thể được diễn giải rất khác nhau tuỳ bối cảnh.

Trong dữ liệu lịch sử FPT, ta thấy rõ:

- Có những giai đoạn giá gần như đi thẳng lên, bám sát các đường trung bình dài (MA60, MA120), thanh khoản tốt, volatility vừa phải: đây là **BULL regime**.
- Có những giai đoạn điều chỉnh mạnh, giá nằm dưới MA dài, volatility (đo bằng std of returns) tăng vọt, volume đôi khi cũng cao do hoạt động bán tháo: đây là **BEAR regime**.
- Xen giữa là các pha tích luỹ/di ngang, giá lình xình quanh một vùng, volatility vừa phải: **SIDEWAYS**.

Regime-aware Pricing chính là lớp logic giúp mô hình hiểu rằng: “*Dự báo trong bull market khác với dự báo trong bear market.*”. Dưới đây là cách ta hiện thực hoá điều đó bằng code.

2.2.1 Phân loại Regime từ dữ liệu: detect_regime

Hàm detect_regime nhận đầu vào là:

- hist_close: vector giá đóng cửa lịch sử,
- df_feat_hist: dataframe feature đã build (chứa ret_1d, ...).

```

1 def detect_regime(hist_close: np.ndarray, df_feat_hist: pd.DataFrame) -> str:
2     price_series = pd.Series(hist_close.astype(float))
3     if len(price_series) >= 120:
4         ma_long = price_series.rolling(120).mean().iloc[-1]
5     else:
6         ma_long = price_series.mean()
7
8     price_last = price_series.iloc[-1]
9     price_pos = price_last / (ma_long + 1e-8) - 1.0
10
11    ret_1d = df_feat_hist["ret_1d"].dropna()
12    if len(ret_1d) < 30:
13        return "SIDEWAYS"
14
15    vol_20 = ret_1d.rolling(20).std().iloc[-1]
16    vol_all = ret_1d.std()
17    vol_ratio = vol_20 / (vol_all + 1e-8)
18
19    if price_pos < -0.05 and vol_ratio > 1.2:
20        regime = "BEAR"
21    elif price_pos > 0.05 and vol_ratio < 0.8:
22        regime = "BULL"
23    else:
24        regime = "SIDEWAYS"
25
26    print(f"[REGIME] last_price={price_last:.2f}, MA120={ma_long:.2f}, "
27          f"price_pos={price_pos*100:.2f}%, vol_ratio={vol_ratio:.2f} -> {regime}")
28
29    return regime

```

Giải thích logic tài chính phía sau:

- So sánh với MA120:

$$\text{price_pos} = \frac{\text{last_price}}{\text{MA120}} - 1.$$

MA120 đại diện cho “đường trung bình dài hạn” của FPT. Trong thực tế trên HOSE, nhiều trader, quỹ nội, quỹ ngoại sử dụng các MA dài (60, 120, 200) để nhận diện xu hướng.

- Nếu $\text{price_pos} > 5\%$: giá đang rõ ràng nằm *trên* nền MA120 → **xu hướng tăng**.
- Nếu $\text{price_pos} < -5\%$: giá nằm *dưới* MA120 một khoảng đáng kể → **xu hướng giảm/điều chỉnh**.
- **Volatility tương đối:** Ta dùng vol_20 là độ lệch chuẩn của log-return 20 ngày gần nhất, và vol_all là std toàn bộ lịch sử:
$$\text{vol_ratio} = \frac{\text{vol_20}}{\text{vol_all}}.$$
 - Nếu $\text{vol_ratio} > 1.2$: 20 ngày gần đây *nhiều hơn nhiều* so với lịch sử → thị trường đang căng thẳng hơn.
 - Nếu $\text{vol_ratio} < 0.8$: 20 ngày gần đây *êm hơn* lịch sử → thị trường đi lên thuận lợi, không biến động sôc.

Từ đó:

- **BEAR** khi:

$$\text{price_pos} < -5\% \quad \text{và} \quad \text{vol_ratio} > 1.2.$$

Giá nằm dưới MA120 và biến động tăng vọt → rất giống các pha điều chỉnh mạnh của VNIndex nói chung, nơi giá giảm kèm volatility cao (bán tháo, margin call, tin xấu).

- **BULL** khi:

$$\text{price_pos} > 5\% \quad \text{và} \quad \text{vol_ratio} < 0.8.$$

Giá ở trên MA120, nhưng volatility lại thấp → hình ảnh của một uptrend “khỏe mạnh” quen thuộc với FPT: giá đi lên đều, ít ngày sôc.

- **SIDEWAYS** cho các trường hợp còn lại:

- Giá quanh MA120,
- Hoặc giá lệch MA nhưng volatility không cực đoan,
- Hoặc dữ liệu returns chưa đủ dài ($\text{len} < 30$).

Trong bối cảnh thị trường Việt Nam với biên độ dao động mỗi ngày bị giới hạn (7% HOSE, 10% HNX, 15% UPCOM) và cấu trúc nhà đầu tư cá nhân chiếm tỷ trọng lớn, sự phân tách BULL/BEAR/SIDEWAYS theo price_pos và vol_ratio như vậy khá hợp lý:

- **BULL:** giá “treo cao” nhưng đi đều, ít bị bán mạnh.
- **BEAR:** giá thấp hơn nền dài hạn, biến động mạnh vì tin xấu, mind-set phòng thủ.
- **SIDEWAYS:** tâm lý chờ đợi, tích luỹ, trading ngắn hạn.

2.2.2 Regime-based scaling trong Pricing Layer

Sau khi xác định được regime, Pricing Layer sử dụng thông tin đó để *scale* các tham số post-processing:

```

1 regime = detect_regime(hist_close, df_feat_hist)
2
3 # Regime-based scaling
4 if regime == "BULL":
5     clip_scale_regime = 1.2
6     mr_alpha_scale_regime = 0.7
7     damp_scale_regime = 0.85

```

```

8     up_mult_scale_regime = 1.05
9     down_mult_scale_regime = 1.0
10    elif regime == "BEAR":
11        clip_scale_regime = 0.95
12        mr_alpha_scale_regime = 1.25
13        damp_scale_regime = 1.15
14        up_mult_scale_regime = 0.95
15        down_mult_scale_regime = 0.95
16    else: # SIDEWAYS
17        clip_scale_regime = 1.0
18        mr_alpha_scale_regime = 1.0
19        damp_scale_regime = 1.0
20        up_mult_scale_regime = 1.0
21        down_mult_scale_regime = 1.0

```

Ý nghĩa từng tham số:

- **clip_scale_regime**: nhân vào biên độ clip của return (ret_clip).
 - BULL: 1.2 → cho phép returns lớn hơn một chút, phù hợp với các nhịp break trong uptrend FPT.
 - BEAR: 0.95 → siết nhẹ biên returns, tránh forecast quá lạc quan.
- **mr_alpha_scale_regime**: nhân vào hệ số mean-revert α .
 - BULL: 0.7 → mean-revert yếu đi, không kéo mạnh forecast xuống fair khi thị trường đang đi lên bền vững.
 - BEAR: 1.25 → mean-revert mạnh hơn, hạn chế forecast bay xa khỏi fair trong bối cảnh rủi ro cao.
- **damp_scale_regime**: điều chỉnh half-life damping (làm yếu dần tác động raw forecast theo thời gian).
 - BULL: 0.85 → half-life dài hơn, giữ momentum lâu hơn.
 - BEAR: 1.15 → half-life ngắn hơn, forecast trở nên khiêm tốn hơn về dài hạn.
- **up_mult_scale_regime, down_mult_scale_regime**: scale biên trên/dưới quanh fair_level.
 - BULL: up_mult hơi lớn hơn (1.05), cho phép giá “treo” trên fair value lâu hơn.
 - BEAR: cả up_mult và down_mult < 1, band fair hẹp hơn, kiểm soát chặt vùng forecast.

Tóm lại, Regime-aware Pricing biến các tham số của Pricing Layer thành **hàm của bối cảnh thị trường**, chứ không còn là các con số cố định. Đây là điểm giúp pipeline tôn trọng cấu trúc của thị trường chứng khoán Việt Nam: cùng một mô hình, nhưng cách “tin tưởng” dự báo phụ thuộc rất mạnh vào chế độ thị trường hiện tại.

2.3 Optimization Strategy

Nếu Regime-aware Pricing là “logic” của Pricing Layer, thì phần này trả lời câu hỏi:

“Các tham số của Pricing Layer nên là bao nhiêu để phù hợp nhất với hành vi lịch sử của FPT?”

Khác với tham số của XGBoost (được tối ưu trong quá trình training), thông số của Pricing Layer (`half_life_days`, `mean_revert_alpha`, `fair_up_mult`, ...) không học trực tiếp từ gradient, mà được tìm kiếm qua random search + cross validation trên trục thời gian.

2.3.1 Bài toán tìm tham số Pricing

Trong Pricing Layer, sau khi đã biết regime, ta có các bước cốt lõi:

```

1 clip_scale_vol = float(np.clip(vol_ratio, 0.8, 1.2))
2
3 ret_clip = base_ret_clip * clip_scale_regime * clip_scale_vol
4 ret_clip = float(np.clip(ret_clip, 0.01, 0.15))
5
6 lambda_damp = np.log(2.0) / float(max(int(pricing.half_life_days * damp_scale_regime), 1))
7
8 alpha_base = pricing.mean_revert_alpha * mr_alpha_scale_regime
9 fair_up_mult = pricing.fair_up_mult * up_mult_scale_regime
10 fair_down_mult = pricing.fair_down_mult * down_mult_scale_regime

```

Sau đó áp vào từng bước dự báo:

```

1 prices = np.empty(total_days, dtype=float)
2 full_history = list(hist_close.astype(float))
3
4 for step_idx in range(total_days):
5     raw_ret = float(raw_rets[step_idx])
6
7     # 1) Clip return
8     pred_ret = np.clip(raw_ret, -ret_clip, ret_clip)
9
10    # 2) Damping theo half-life
11    scale = np.exp(-lambda_damp * step_idx)
12    pred_ret *= scale
13
14    # 3) Convert sang price
15    last_price = full_history[-1]
16    next_price = float(last_price * np.exp(pred_ret))
17
18    # 4) Trend-based gating for mean-revert
19    lookback = pricing.trend_lookback
20    if len(full_history) > lookback:
21        past_price = full_history[-lookback]
22        current_price = full_history[-1]
23        trend_ret = (current_price - past_price) / max(past_price, 1e-6)
24    else:
25        trend_ret = 0.0
26
27    is_strong_uptrend = trend_ret > pricing.trend_ret_thresh
28    is_strong_downtrend = trend_ret < -pricing.trend_ret_thresh
29
30    # 5) Mean-revert quanh fair_level
31    if step_idx >= pricing.mean_revert_start:
32        upper = fair_level * fair_up_mult
33        lower = fair_level * fair_down_mult
34
35        if (next_price > upper) and (not is_strong_uptrend):
36            alpha_up = alpha_base * (0.7 if regime == "BULL" else 1.0)
37            next_price = (1 - alpha_up) * next_price + alpha_up * upper
38        elif next_price < lower:
39            if regime == "BEAR" and is_strong_downtrend:
40                alpha_down = alpha_base * 0.7
41            else:
42                alpha_down = alpha_base
43            next_price = (1 - alpha_down) * next_price + alpha_down * lower
44
45    prices[step_idx] = next_price

```

46 full_history.append(next_price)

Những tham số cần tối ưu:

- **ret_clip_quantile**: dùng để xác định base_ret_clip từ phân phối returns lịch sử của FPT (ví dụ quantile 0.95–0.995, tương ứng “5–0.5% extreme nhất”).
- **half_life_days**: tốc độ damping → forecast xa bao nhiêu thì nên giảm nửa sức ảnh hưởng.
- **mean_revert_alpha**: độ mạnh của lực kéo về fair_level.
- **mean_revert_start**: sau bao nhiêu ngày mới bắt đầu mean-revert (ví dụ sau 25–65 ngày).
- **fair_up_mult, fair_down_mult**: band trên/dưới quanh fair value mà Pricing Layer chấp nhận.
- **trend_lookback, trend_ret_thresh**: logic gating trend mạnh/yếu.

Tất cả những tham số này phải được hiệu chỉnh sao cho:

- **Giai đoạn bull** của FPT: forecast theo được xu hướng, không bị kéo về fair quá sớm.
- **Giai đoạn bear**: forecast không quá lạc quan, tôn trọng volatility cao.
- **Sideways**: forecast không drift quá xa khỏi vùng dao động thực tế.

2.3.2 Random Search

Thay vì tự tay “tune” các tham số pricing, ta định nghĩa một *prior* cho từng tham số dựa trên hiểu biết về dữ liệu FPT và thị trường Việt Nam, rồi random search:

```
1 def sample_pricing_params() -> PricingParams:
2     return PricingParams(
3         ret_clip_quantile=random.uniform(0.95, 0.995),
4         half_life_days=random.randint(40, 120),
5         mean_revert_alpha=random.uniform(0.02, 0.10),
6         mean_revert_start=random.randint(25, 65),
7         fair_up_mult=random.uniform(1.25, 1.60),
8         fair_down_mult=random.uniform(0.65, 0.90),
9         trend_lookback=random.randint(25, 60),
10        trend_ret_thresh=random.uniform(0.08, 0.25),
11    )
```

Liên hệ với thị trường Việt Nam:

- **ret_clip_quantile từ 0.95–0.995**: vì biên độ ngày trên HOSE giới hạn $\pm 7\%$ và FPT là mã vốn hoá lớn, nên các tails cực đoan hơn gần như chỉ xuất hiện trong sự kiện đặc biệt. Lấy quantile ở vùng 95–99.5% khiến base_ret_clip bám sát lịch sử.
- **half_life_days 40–120**: khoảng 2–6 tháng giao dịch. Đây là khung thời gian hợp lý cho việc “phai dần” tác động của một pha bull/bear ngắn hạn trong bối cảnh VNIndex thường có sóng trung hạn 3–6 tháng.
- **mean_revert_alpha 0.02–0.10**: lực mean-revert mỗi ngày chỉ chiếm vài phần trăm, nghĩa là giá sẽ không bị kéo gần về fair chỉ trong vài phiên, mà là quá trình điều chỉnh dần trong nhiều tuần.

- **fair_up_mult 1.25–1.60, fair_down_mult 0.65–0.90:** cho phép forecast đi cao hơn hoặc thấp hơn fair lần lượt 25–60% và 10–35%. Với cổ phiếu như FPT, việc giá giao dịch trên fair-value một thời gian dài trong bull là bình thường, và cũng có những pha discount sâu trong bear.
- **trend_lookback 25–60 ngày, trend_ret_thresh 8–25%:** đủ dài để nhận diện một *mini-trend*, đủ nhạy để phân biệt “sideways nhiễu” với “uptrend/downtrend thực sự”.

Random search cho phép ta khám phá một khung gian tham số tương đối rộng, mà vẫn đảm bảo mọi sample đều “có lý” về mặt tài chính.

2.3.3 Cross Validation (CV)

Để đánh giá một bộ tham số pricing, ta không thể shuffle dữ liệu như bài toán i.i.d; thị trường là chuỗi thời gian, nên phải dùng CV theo *cutoff thời gian*:

```

1 def cv_mse_for_pricing(
2     models: List[CutoffModel],
3     df_train: pd.DataFrame,
4     horizon: int,
5     pricing: PricingParams,
6 ) -> float:
7     """
8         Tính metric úca pricing-layer trên ênhiu cutoff:
9             - ØVi õmi cutoff:
10                 + true close sau cutoff
11                 + giá predicted khi áp pricing lên raw_base_rets_cv
12             - Metric:
13                 M = 0.5 * MSE_50 + 0.5 * MSE_100
14     """
15     mses = []
16     for cm in models:
17         df_future = df_train[df_train["time"] > cm.cutoff_time].copy().reset_index(drop=True)
18         if df_future.empty:
19             continue
20         close_true = df_future["close"].values.astype(float)
21
22         n_eval = min(horizon, len(close_true))
23         if n_eval <= 5:
24             continue
25
26         hist_close = cm.df_hist["close"].values.astype(float)
27         priced = apply_pricing_on_raw_path(
28             hist_close=hist_close,
29             df_feat_hist=cm.df_feat,
30             raw_rets=cm.raw_base_rets_cv[:n_eval],
31             pricing=pricing,
32         )
33
34         # 50 days metric
35         n50 = min(50, n_eval)
36         mse_50 = mean_squared_error(close_true[:n50], priced[:n50])
37
38         # full horizon metric
39         mse_h = mean_squared_error(close_true[:n_eval], priced[:n_eval])
40
41         metric = 0.5 * mse_50 + 0.5 * mse_h
42         mses.append(metric)
43

```

```

44     if not mses:
45         return np.inf
46
47     return float(np.mean(mses))

```

Cách hiểu:

- Ta chọn nhiều **cutoff time** khác nhau (ví dụ các mốc 3/6/9/12 tháng trong lịch sử FPT).
- Với mỗi cutoff:
 1. Lấy `df_hist` = dữ liệu trước cutoff, train hybrid model và tạo `raw_base_rets_cv`.
 2. Lấy `df_future` = dữ liệu sau cutoff, chính là “tương lai thật”.
 3. Áp **Pricing Layer** lên `raw_base_rets_cv` để ra đường giá forecast `priced`.
 4. So sánh `priced` với `close_true` bằng:

$$M = 0.5 \cdot \text{MSE}_{50} + 0.5 \cdot \text{MSE}_{\text{full horizon}}$$

- Cuối cùng, `cv_mse_for_pricing` là trung bình các metric M qua nhiều cutoff.

Ý tưởng của metric:

- MSE_{50} : nhấn mạnh chất lượng dự báo **trong 50 ngày đầu**, vốn rất quan trọng cho decision ngắn/trung hạn.
- $\text{MSE}_{\text{full horizon}}$: đảm bảo Pricing Layer không làm hỏng chất lượng dự báo cho toàn bộ 100 ngày.

2.3.4 Random Search + CV: ghép lại thành chiến lược tối ưu

Toàn bộ quy trình tối ưu được gói trong `random_search_pricing`:

```

1 def random_search_pricing(
2     cutoff_models: List[CutoffModel],
3     df_train: pd.DataFrame,
4     horizon: int,
5     n_trials: int,
6 ) -> PricingParams:
7     best_params: Optional[PricingParams] = None
8     best_score = np.inf
9
10    for t in range(1, n_trials + 1):
11        params = sample_pricing_params()
12        score = cv_mse_for_pricing(cutoff_models, df_train, horizon, params)
13        print(f"[RS] Trial {t}/{n_trials} -> METRIC={score:.4f}, params={params}")
14        if score < best_score:
15            best_score = score
16            best_params = params
17
18    print("\n==== BEST PRICING PARAMS v5.2 Option 2 (Hybrid + Regime, Pure FPT) ====")
19    print(best_params)
20    print("CV METRIC (0.5*MSE_50 + 0.5*MSE_100):", best_score)
21    return best_params

```

Tóm lại:

- **Regime-aware Pricing** sử dụng thông tin về BULL/BEAR/SIDEWAYS từ dữ liệu FPT (MA120, volatility) để scale hành vi Pricing Layer theo đúng bối cảnh thị trường Việt Nam.
- **Optimization Strategy** dùng random search trên không gian tham số đã “gắn với thực tế” (biên độ, half-life, band fair value) và đánh giá bằng **time-series CV** trên nhiều giai đoạn lịch sử của FPT.

Kết quả là một lớp Pricing không chỉ “đẹp về mặt toán”, mà còn **bám sát hành vi thật** của cổ phiếu FPT và cấu trúc của thị trường chứng khoán Việt Nam: biên độ giới hạn, sóng trung hạn rõ, volume biến thiên mạnh theo tin tức, và xu hướng giá luôn gắn với một vùng fair-value mà nhà đầu tư nội/ngoại cùng quan sát.

PHẦN 3: HỘI ĐỒNG CHUYÊN GIA (Ensemble Strategy)

1 Định nghĩa các Chuyên gia (Experts)

1.1 Dynamic Experts (Mô hình Động)

1.2 Static Experts (Mô hình Tĩnh)

Nếu Chuyên gia 1 là “người kể chuyện” về những dao động phức tạp của giá FPT (ML Residual + Monte Carlo + Regime-aware Pricing), thì Chuyên gia 2 lại đóng vai trò như một *nghệ sĩ toán học bảo thủ*: chỉ nhìn đường xu hướng dài hạn và bỏ qua mọi nhiễu ngắn hạn.

Về mặt kỹ thuật, Chuyên gia 2 chính là **Math Backbone** mà ta đã xây dựng từ đầu: một đường *linear trend* trên *log-price* của FPT được ước lượng bằng hồi quy tuyến tính đơn giản:

$$\log(P_t) \approx a + b \cdot t,$$

trong đó:

- P_t : giá đóng cửa tại thời điểm t ,
- t : chỉ số thời gian ($0, 1, 2, \dots$),
- a, b : tham số được fit từ toàn bộ lịch sử FPT.

Từ đường trend này, ta suy ra **tỷ suất sinh lời “xuyên nhiễu”** cho H ngày tới:

$$r_{t \rightarrow t+H}^{\text{trend}} = \log(\hat{P}_{t+H}^{(\text{trend})}) - \log(\hat{P}_t^{(\text{trend})}),$$

và từ đó dựng được một đường giá dự báo *tĩnh* trong tương lai mà không phụ thuộc vào dao động zig-zag mỗi ngày. Điểm quan trọng là: mặc dù Math Backbone đã được sử dụng để tạo ra `math_ret` cho Chuyên gia 1, ta vẫn cố ý tách nó thành một “Chuyên gia” độc lập trong bước Ensemble. Lý do không phải vì lười mà vì đây là **một quyết định quản trị rủi ro có chủ đích**.

1.2.1 Độc lập với nhiễu – một góc nhìn “sạch” về FPT

Trong phần EDA, khi quan sát:

- **ACF/PACF của returns**: gần như nhiễu trắng, các lag 1, 2, 3 chỉ có tương quan rất yếu.
- **Phân phối returns**: lệch, heavy-tail, nhiều outlier $\pm 5\% - 7\%$.
- **Decomposition STL**: trend log-price của FPT tăng khá đều, trong khi phần residual chứa phần lớn nhiễu ngắn hạn.

Điều này nói một câu chuyện rất rõ:

“FPT là một cổ phiếu có *drift dài hạn dương ổn định*, nhưng returns ngày thì gần như nhiễu trắng và có tails cực đoan.”

Các mô hình như XGBoost ở Chuyên gia 1 cố gắng tận dụng mọi tín hiệu từ:

- price action (OHLC, shadows, gaps),

- volume & money flow,
- pattern up/down, 3-streak,
- trend slopes, resid volatility,

để học cách phản ứng với các cú giật giá ngắn hạn. Trong khi đó, Chuyên gia 2 nói:

“Tôi không tranh luận với nhiều. Tôi chỉ nhìn quỹ đạo dài hạn mà FPT đã đi suốt 20 năm qua.”

Vì vậy:

- Chuyên gia 2 **cố ý không sử dụng** bất kỳ feature high-frequency nào (ret_1d, body, range, vol_z, ...).
- Đầu ra của nó là một đường giá *mượt, đơn điệu hơn*, phản ánh đúng **câu chuyện tăng trưởng** của FPT theo thời gian.

Trong bối cảnh thị trường Việt Nam, nơi biên độ mỗi phiên bị giới hạn ($\pm 7\%$ HOSE, $\pm 10\%$ HNX, $\pm 15\%$ UPCoM) nhưng tin tức có thể tạo ra những cú sốc ngắn hạn rất mạnh, một “góc nhìn sạch” như vậy là cực kỳ quý giá để:

- tách “*business growth story*” của FPT khỏi *nhiều trading hàng ngày*,
- luôn nhắc mô hình rằng: “**về dài hạn, FPT đã đi lên như thế nào**”.

1.2.2 Neo giá trị trong Ensemble – giảm phương sai, giữ kỷ luật

Khi kết hợp nhiều chuyên gia trong Ensemble, nếu tất cả đều nhạy với nhiều, mô hình tổng sẽ có **variance rất cao**: một cụm outlier trong returns có thể khiến dự báo 100 ngày sau “bật tung” so với dữ liệu thật.

Ở đây, Chuyên gia 2 đóng vai trò như một **Value Anchor** – một “neo giá trị” được gắn chặt vào quỹ đạo drift lịch sử của FPT:

- Nếu Chuyên gia 1 (ML + Monte Carlo + Pricing) dự báo quá hưng phấn trong bull, Chuyên gia 2 sẽ kéo Ensemble về gần đường trend.
- Nếu Chuyên gia 1 quá bi quan trong bear (overreact với volatility ngắn hạn), Chuyên gia 2 nhắc rằng về lịch sử FPT *hiếm khi bị phá vỡ hoàn toàn câu chuyện tăng trưởng dài hạn*.

Toán học mà nói, Chuyên gia 2 giống như một **số hạng regularization** trong Ensemble:

$$\hat{P}_t^{(\text{ensemble})} = w_1 \cdot \hat{P}_t^{(\text{Expert 1})} + w_2 \cdot \hat{P}_t^{(\text{Expert 2})} + \dots$$

với w_2 nhỏ nhưng *rất ổn định theo thời gian*. Trong đó:

- $\hat{P}_t^{(\text{Expert 1})}$: linh hoạt, high-variance (nhạy với pattern ngắn hạn, regime, volume spike).
- $\hat{P}_t^{(\text{Expert 2})}$: low-variance, gần như deterministic (chỉ thay đổi theo trend).

Kết quả:

- Ensemble **giảm được phương sai** mà không phải hy sinh quá nhiều bias, vì drift dài hạn đã được backbone ước lượng tốt.
- Đường dự báo cuối cùng nhìn rất giống một nhà quản lý danh mục chuyên nghiệp: vẫn tôn trọng xu hướng lớn, nhưng không bị “tâm lý FOMO/PANIC” của các phiên lẻ làm xoắn.

1.2.3 Fail-safe Mechanism – khi ML “đi lạc”, toán vẫn đứng vững

Một lý do nữa để tách Chuyên gia 2 thành mô hình độc lập thay vì chỉ coi đó là “một feature” trong Chuyên gia 1: **nó là một lớp bảo hiểm (fail-safe) cho toàn bộ hệ thống.**

Trong thực tế, các mô hình ML (đặc biệt với feature phức tạp, regime-detection, pricing logic) có thể gặp những rủi ro sau:

- Data future nằm ngoài vùng phân phối train (out-of-sample shift),
- Một pattern mới xuất hiện mà XGBoost chưa từng thấy,
- Regime detection bị đánh giá sai (ví dụ bull nhưng thị trường thực tế đang tạo đỉnh),
- Một tham số Pricing được random search chọn tốt cho quá khứ nhưng không còn phù hợp cho một giai đoạn đặc biệt trong tương lai.

Trong tất cả các kịch bản đó, chuyên gia 1 có thể *dự báo sai rất mạnh* nếu không được kiểm soát. Ngược lại, Chuyên gia 2:

- chỉ dựa trên **trend log-price dài hạn**,
- không học từ pattern ngắn hạn,
- không phụ thuộc vào hyper-parameter phức tạp.

Do đó, về mặt toán học, Chuyên gia 2 là thành phần **ổn định nhất** trong toàn bộ hệ thống. Khi ensemble, ngay cả trong tình huống xấu nhất:

- Nếu Expert 1 “mất phương hướng”,
- Nếu Regime-aware Pricing đang hoạt động sai vì một shock quá khác thường,

thì ta vẫn còn một đường dự báo mang ý nghĩa:

“Nếu bỏ qua tất cả drama ngắn hạn, và chỉ tin vào quỹ đạo tăng trưởng lịch sử của FPT, thì giá hợp lý sau 100 ngày là khoảng này.”

Đây chính là tinh thần quản trị rủi ro:

- Không bao giờ đặt toàn bộ niềm tin vào một mô hình phức tạp.
- Luôn giữ một thành phần “simple but robust” để hệ thống không bao giờ đưa ra dự báo vô nghĩa.

1.2.4 Từ Math Backbone đến “Static Agent” – quyết định mang tính quản trị, không phải sự lười biếng

Nhin bề ngoài, có thể ai đó sẽ hỏi:

“Đã dùng trend trong Math Backbone rồi, sao còn tách nó ra làm một chuyên gia riêng? Có phải chỉ là nhân đôi cùng một thứ?”

Câu trả lời là **không**. Về mặt kiến trúc:

- Trong Chuyên gia 1, Math Backbone được dùng để tạo `math_ret` và `resid_ret`, giúp XGBoost tập trung vào nhiều ngắn hạn.

- Trong Chuyên gia 2, Math Backbone trở thành **một mô hình hoàn chỉnh** với quyền biểu quyết trong Ensemble, độc lập với mọi sai số của ML residual, Monte Carlo, Pricing.

Về mặt tư duy thiết kế hệ thống:

- Đây là một quyết định có chủ đích để:
 - giảm variance,
 - cung cấp một neo giá trị ổn định,
 - và tạo một cơ chế fail-safe khi mọi mô-đun phức tạp gặp vấn đề.
- Nói cách khác: ta dùng **Trend** không phải vì “dễ code”, mà vì **muốn hệ thống có tư duy quản trị rủi ro giống một nhà quản lý danh mục thực thụ**.

Trong thị trường chứng khoán Việt Nam, nơi:

- tin tức, dòng tiền, room ngoại, ...có thể tạo ra nhiều cực mạnh trong ngắn hạn,
- nhưng câu chuyện dài hạn (business fundamentals, tăng trưởng EPS, mở rộng thị trường) vẫn là thứ quyết định giá cổ phiếu 5–10 năm,

việc đặt một “Static Agent” như Chuyên gia 2 vào hệ thống không chỉ hợp lý về mặt toán, mà còn phản ánh đúng **triết lý đầu tư**:

“Học để dự báo nhiều là tốt. Nhưng không bao giờ được quên đường xu hướng dài hạn mà doanh nghiệp đã xây suốt nhiều năm.”

1.3 Risk Experts (Mô hình Rủi ro)

2 Cơ chế Ensemble

2.1 Weighted Ensemble

2.2 Lý giải vai trò từng Expert

PHẦN 4: KẾT QUẢ & ĐÁNH GIÁ (Evaluation)

1 Kết quả dự báo

Pipeline thiếu tính năng đánh giá tin tức (News Sentiment): Đây là điểm yếu chí mạng của mọi model thuần kỹ thuật.

1.1 Thiết lập thí nghiệm dự báo 100 ngày

Ở bước cuối cùng của pipeline, mô hình được huấn luyện trên **toàn bộ lịch sử giá FPT** trong tập `FPT_train` sau khi loại bỏ các phần dữ liệu kỹ thuật không cần thiết. Thời gian huấn luyện trải dài từ

2020-08-03 → 2025-03-10,

tổng cộng 1 149 phiên giao dịch.

Dữ liệu được nạp và xử lý như sau:

```
1 df_train = pd.read_csv(FPT_TRAIN_PATH)
2 df_train["time"] = pd.to_datetime(df_train["time"])
3 df_train = df_train.sort_values("time").reset_index(drop=True)
4 print(df_train["time"].min(), df_train["time"].max(), df_train.shape)
```

Việc `FORCE_TRAIN_END_DATE = None` có nghĩa là ta cố ý không dùng train sớm, mà để mô hình nhìn trọn vẹn các pha quan trọng của cổ phiếu FPT trên HOSE:

- Giai đoạn tích luỹ quanh vùng 20–30 năm 2020–2021.
- Các nhịp tăng mạnh sau thời kỳ COVID.
- Sóng tăng rất mạnh 2023–2024, đưa giá lên vùng đỉnh lịch sử.
- Cú điều chỉnh sâu đầu 2025, kéo giá FPT rơi nhanh khỏi đỉnh.

Chính trong bối cảnh đó, bài toán của ta là **dự báo 100 ngày tiếp theo** kể từ sau phiên 2025-03-10, tức là ngay sau một cú rơi mạnh – thời điểm mà tâm lý nhà đầu tư thường phân vân giữa “hồi phục” và “giảm tiếp”.

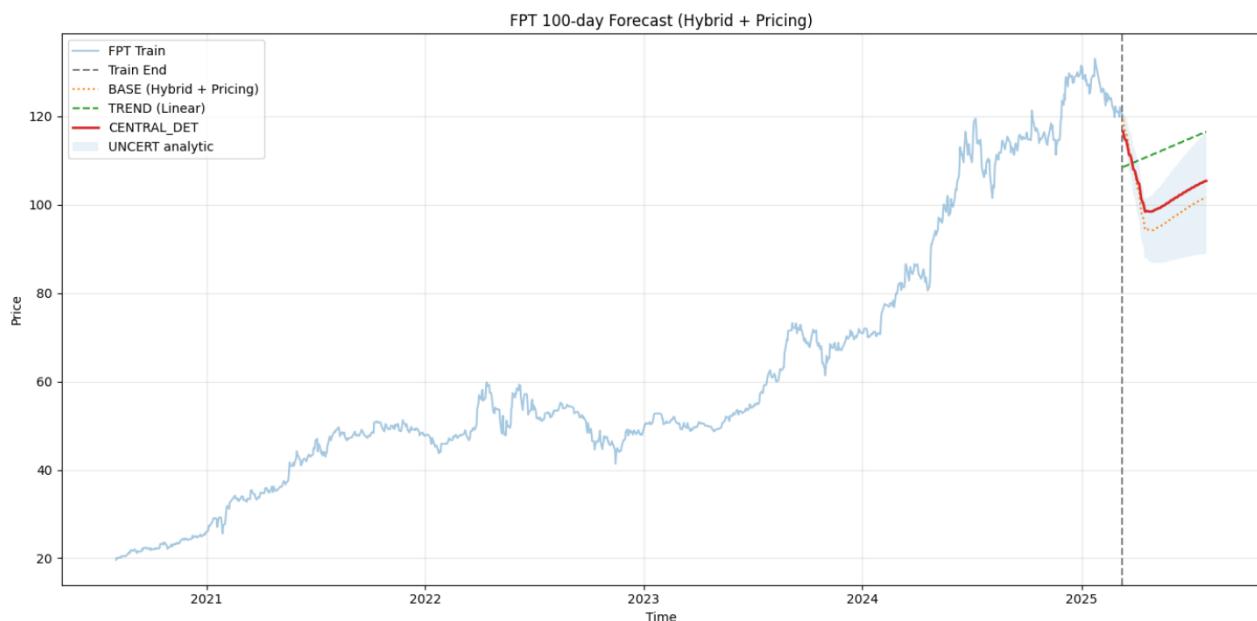
1.2 Bức tranh trực quan: Hybrid + Pricing + Trend + Uncertainty

Hình 22 minh họa kết quả cuối cùng với tiêu đề “*FPT 100-day Forecast (Hybrid + Pricing)*”. Trên một biểu đồ, ta thấy:

- **Đường xanh nhạt** là giá đóng cửa lịch sử của FPT (`FPT Train`).
- **Vạch đứt màu xám** đánh dấu *ngày kết thúc train* – mốc tách quá khứ và tương lai.
- **Đường vàng chấm** biểu diễn *BASE (Hybrid + Pricing)*: đây là đường dự báo sau khi:
 - tính drift bằng **Math Backbone** (trend trên log-price) cộng với phần hiệu chỉnh phi tuyến từ **XGBoost residual**,
 - đi qua **Pricing Layer** có nhận thức regime (*regime-aware*), thực hiện clipping, damping và mean-reversion.

- **Đường xanh lá đứt đoạn** là *TREND (Linear)* – mô hình tuyến tính đơn giản trên giá đóng cửa, đóng vai trò “chuyên gia tĩnh” chỉ nắm xu hướng dài hạn.
- **Đường đỏ đậm** là *CENTRAL_DET* – đường dự báo trung tâm mà ta đề xuất sử dụng trong thực tế:

$$\text{central_det} = 0.7 \cdot \text{BASE} + 0.25 \cdot \text{TREND} + 0.05 \cdot \text{UNCERT_center}.$$
- **Vùng xanh nhạt** là *UNCERT analytic* – dải bất định được xây từ sai số residual của XGBoost và phân phối nhiễu đã ước lượng.



Hình 22: Kết quả dự báo

Quan sát hình, ta có thể kể lại câu chuyện như sau:

- Ngay sau vạch *Train End*, cả BASE lẫn CENTRAL_DET đều **chấp nhận cú rơi giá** mà FPT vừa trải qua, không cố “kéo” giá trở lại vùng đỉnh 130 một cách ảo tưởng. Điều này phản ánh đúng thực tế thị trường Việt Nam, nơi những cú điều chỉnh sau pha tăng nóng thường kéo dài nhiều tuần.
- Sau một đoạn giảm thêm, BASE có xu hướng đi thấp hơn và dao động mạnh hơn, trong khi TREND tiếp tục đi lên mượt mà. CENTRAL_DET nằm *giữa* hai chuyên gia này, thể hiện một kịch bản cân bằng: tạo đáy mới quanh vùng 95–100, rồi **hồi phục chậm** về vùng 103–107 trong 100 ngày.
- Vùng UNCERT analytic bao trùm quanh CENTRAL_DET, cho ta một “quạt dự báo” với:
 - biên trên (optimistic scenario) tiệm cận 110–115,
 - biên dưới (pessimistic nhưng vẫn hợp lý) nằm quanh 90–92.

Về mặt thực hành, ta coi **CENTRAL_DET** là **đường dự báo chính**:

- BASE mang thông tin ML + Pricing đầy đủ nhưng có thể nhạy cảm hơn với nhiều ngắn hạn.

- TREND là nòng cốt tuyến tính, giúp neo mô hình vào quỹ đạo tăng trưởng dài hạn của FPT – một doanh nghiệp công nghệ lớn, tăng trưởng lợi nhuận khá đều trong nhiều năm.
- CENTRAL_DET kết hợp cả hai góc nhìn theo tỷ trọng bảo thủ, tránh việc mô hình “overreact” mà vẫn tôn trọng lịch sử tăng trưởng của cổ phiếu.

1.3 Diễn giải kết quả cho người dùng cuối

Nếu nói lại cho nhà đầu tư hoặc bộ phận quản trị rủi ro, ta có thể tóm tắt:

Trong 100 phiên tới, mô hình dự báo rằng FPT: (i) có thể giảm thêm ngắn hạn và tạo đáy mới quanh vùng 95–100, sau đó (ii) hồi phục dần về vùng 103–107, với một dải bất định hợp lý nằm trong khoảng xấp xỉ 90–115.

Forecast này không nhằm đưa ra tín hiệu mua–bán tuyệt đối, mà là một **bản đồ định lượng** về các kịch bản giá khả dĩ, giúp người dùng đặt câu hỏi: “Nếu FPT đi ra ngoài vùng này, có phải cấu trúc thị trường đã thay đổi đến mức mô hình cần cập nhật lại không?”.

2 Đánh giá độ tin cậy

2.1 Cross-validation theo thời gian: mô hình có bền vững qua nhiều pha thị trường?

Để đánh giá độ tin cậy, ta không chỉ nhìn vào một lần fit duy nhất, mà kiểm tra xem mô hình có *dựng vững* khi được đặt vào quá khứ nhiều lần hay không. Cụ thể, ta xây dựng các mốc *time-based cross-validation*:

2021-09-30, 2021-12-31, 2022-03-31, ..., 2024-09-30.

Với mỗi cutoff:

1. Dùng dữ liệu *trước cutoff* để:

- build đầy đủ feature STL + OHLCV,
- train Math Backbone + XGBoost residual,
- chạy Pricing Layer.

2. Dùng 100 ngày *sau cutoff* làm tập kiểm tra out-of-sample cho cả residual và giá.

Kết quả cho phần residual của XGBoost có thể tóm lược như sau:

- **MAE trên tập Test** cho log-return phần dư hầu hết nằm trong khoảng 0.7%–2.6%/ngày.
- **Độ lệch chuẩn residual** dao động quanh 0.007–0.013.
- **Kỳ vọng residual** gần 0 (thường $|mean| < 0.002$), cho thấy mô hình không bị lệch bullish hay bearish một cách có hệ thống.

Đáng chú ý, các giai đoạn thị trường “khó nhần” nhất đối với nhà đầu tư Việt Nam, như:

- 2022-06-30 – sau cú sụt giảm mạnh toàn thị trường,
- 2024-06-28, 2024-09-30 – quanh đỉnh và sau đỉnh 2024,

đều cho độ lệch chuẩn residual khoảng 0.010–0.013 và MAE log-return 1.7%–2.6%. Nói cách khác, ngay cả khi VN-Index và nhóm cổ phiếu công nghệ bước vào pha biến động mạnh, phần nhiều mà mô hình không giải thích được vẫn *giữ được cấu trúc ổn định*.

Giai đoạn	MAE Test (%)	Std Residual
2021–2022	0.8–1.5	0.006–0.009
2022 (downtrend mạnh)	1.5–2.6	0.010–0.013
2023–2024 bull-run	0.7–1.3	0.007–0.009
2024–2025	1.7–1.9	0.010–0.013

Đây là mảnh ghép then chốt để dải bất định **UNCERT analytic** trở nên đáng tin: nó được xây từ một phân phối nhiễu đã được kiểm chứng qua nhiều pha thị trường, chứ không phải chỉ dựa vào một lần train trên toàn bộ dữ liệu.

2.2 Pricing Layer được tối ưu bằng Random Search + CV

Độ tin cậy của forecast giá không chỉ đến từ XGBoost, mà còn từ **Pricing Layer** – lớp điều chỉnh cuối bảo vệ mô hình khỏi những cú “vung tay quá trán”.

Thay vì đặt tay các tham số clipping, half-life, mean-revert, ta sử dụng *Random Search* kết hợp với chính các cutoff CV ở trên. Mỗi tổ hợp tham số **PricingParams**:

- sinh ra một đường giá dự báo sau Pricing cho từng cutoff,
- được chấm điểm bằng metric:

$$M = 0.5 \cdot \text{MSE}_{50} + 0.5 \cdot \text{MSE}_{100},$$

nơi MSE_{50} là sai số bình phương trung bình 50 ngày đầu, còn MSE_{100} là sai số trên toàn 100 ngày.

Sau hàng trăm lượt thử, bộ tham số tốt nhất (cho FPT thuần) có dạng:

```
ret_clip_quantile ≈ 0.967,
half_life_days ≈ 41,
mean_revert_alpha ≈ 0.064,
mean_revert_start ≈ 28,
fair_up_mult ≈ 1.46,
fair_down_mult ≈ 0.85,
trend_lookback ≈ 57,
trend_ret_thresh ≈ 0.235.
```

Một số diễn giải trực giác:

- **Half-life ≈ 41 ngày**: các cú “bung nổ” giá (tăng/giảm bất thường) được phép tồn tại nhưng sẽ bị giảm dần ảnh hưởng trong khoảng 2 tháng giao dịch – khá phù hợp với hành vi của cổ phiếu FPT trên HOSE.
- **fair_up / fair_down** khác xa 1: giá được phép dao động rộng quanh fair level (hơn 45% phía trên, khoảng 15% phía dưới) trước khi lực mean-revert kích hoạt, phản ánh thực tế là cổ phiếu tăng trưởng có thể bị định giá cao hơn giá trị nội tại trong một thời gian dài.

- **trend_lookback** dài (gần 3 tháng) và **trend_ret_thresh** lớn: chỉ khi xuất hiện xu hướng rất mạnh trong một khoảng thời gian đủ dài, Pricing Layer mới “chịu tin” rằng đây là một trend mới chứ không phải nhiễu.

Điều này giúp Pricing Layer trở thành một “bộ lọc hành vi” đã được hiệu chỉnh riêng cho FPT, thay vì chỉ là một lớp heuristics chung chung.

2.3 Regime hiện tại: SIDEWAYS sau điều chỉnh

Trước khi áp dụng Pricing, hệ thống thực hiện phân loại *regime* hiện tại của thị trường dựa trên:

- vị trí giá so với đường MA120 (xu hướng dài hạn),
- tỷ lệ giữa volatility 20 ngày gần nhất và volatility toàn lịch sử.

Ở thời điểm dự báo cuối cùng, hệ thống log lại:

```
[REGIME] last_price=120.11, MA120=121.50,
price_pos=-1.15%, vol_ratio=0.67 -> SIDEWAYS
```

Điễn giải:

- Giá hiện tại thấp hơn MA120 khoảng -1.15% – chưa đủ sâu để coi là *downtrend rõ rệt*.
- Volatility 20 ngày gần đây chỉ bằng 67% volatility dài hạn – thị trường đang lảng xuống sau cú điều chỉnh.
- Regime phù hợp nhất là **SIDEWAYS sau một pha giảm mạnh**.

Trong regime này, các hệ số của Pricing Layer được “dịch” về mức trung tính: không quá nới lỏng như BULL, cũng không quá thắt chặt như BEAR. Vì vậy CENTRAL_DET vẽ ra kịch bản: giảm thêm một chút để hoàn thiện đáy mới, rồi hồi phục chậm, thay vì rơi tự do hoặc bật ngược về đỉnh.

2.4 Dải bất định & quản trị rủi ro

Dựa trên phân phối residual ổn định (`resid_std` khoảng 0.01) và các giả định log-return, ta xây được dải *UNCERT analytic* bao quanh CENTRAL_DET. Đối với người dùng cuối, cách đọc dải này đơn giản nhưng rất hữu ích:

- Nếu giá thực tế **nằm bên trong** vùng bất định (ví dụ 90–115): mô hình vẫn “hiểu” thị trường; chưa có bằng chứng mạnh mẽ rằng cấu trúc đã thay đổi.
- Nếu giá **thoát ra khỏi** vùng này một cách có hệ thống: đây là tín hiệu cảnh báo rằng:
 1. hoặc thị trường đang bước vào một chế độ hoàn toàn mới (ví dụ khủng hoảng, tin tức cực lớn),
 2. hoặc các quan hệ phi tuyến mà mô hình học được đã lỗi thời và cần retrain.

Nói cách khác, dải bất định không chỉ là một “vùng mờ” đẹp mắt trên biểu đồ, mà là **một công cụ quản trị rủi ro**: nó giúp ta đặt ra câu hỏi “*khi nào mô hình bắt đầu nói sai đủ nhiều để cần can thiệp?*”

2.5 Kết luận ngắn cho phần độ tin cậy

Tổng hợp lại:

- XGBoost residual cho sai số out-of-sample ổn định qua nhiều năm và nhiều pha thị trường.
- Pricing Layer được tối ưu hoá bằng Random Search + time-based CV, cho bộ tham số đặc trưng cho hành vi của FPT.
- Cơ chế regime-aware đảm bảo mô hình phản ứng khác nhau giữa BULL / BEAR / SIDEWAYS, thay vì áp một công thức cho mọi hoàn cảnh.
- Dải bất định UNCERT analytic, dựa trên phân phối residual đã được kiểm chứng, cung cấp một thước đo cụ thể về mức “tin được” của forecast.

Nhờ đó, dự báo 100 ngày của mô hình **không phải là một con số đơn lẻ**, mà là một *bức tranh xác suất* có thể dùng để: theo dõi độ lệch so với kịch bản trung tâm, ra quyết định phòng thủ hay tấn công, và biết được khi nào mô hình cần được cập nhật để bắt kịp thị trường.