1. Descripción general del módulo `py103`: Tendencia penalizada para una serie de precios

El objetivo del módulo es tratar una serie temporal de precios diarios (p. ej. NVDA) como un problema de regresión en el tiempo, aplicar varias transformaciones sobre el precio, ajustar una familia de modelos de regresión penalizada (Elastic Net polinomial en el índice de tiempo) y evaluar:

- métricas de error no ponderadas y ponderadas (más peso a los días recientes),
- tanto en el espacio transformado como en el espacio original de precios,
- incluyendo diagnósticos de residuos (normalidad) y
- una selección de modelo que balancea "buen RMSE" y "residuos lo más normales posible".

A continuación se describe con detalle qué hace cada bloque del módulo y por qué.

1.1. Notación básica

Sea

$$\{P_t\}_{t=0}^{T-1}$$

la serie de precios de cierre diario de una acción/índice (p. ej. NVDA), donde:

- t es un índice de tiempo entero (días consecutivos),
- T es el número total de observaciones, y
- $P_t > 0$ es el precio de cierre en el día t.

Definimos también:

$$t \in \{0, 1, \ldots, T-1\} \quad \Rightarrow \quad x_t := t$$

como la covariable (regresor) que representa el tiempo.

En el módulo:

- `download_price_series` descarga los precios diarios desde Yahoo! Finance usando `yfinance` y devuelve un `DataFrame` con índice de fechas y una columna única `'close'` (la serie $\{P_t\}$).

- Usamos siempre un *split temporal* 60/20/20:

$$\begin{aligned} N_{\text{train}} &= \lfloor 0{,}6\,T \rfloor, \\ N_{\text{val}} &= \lfloor 0{,}2\,T \rfloor, \\ N_{\text{test}} &= T - N_{\text{train}} - N_{\text{val}}, \end{aligned}$$

  y los índices de cada segmento se obtienen con `_segment_indices` / `temporal_split`.

2. Transformaciones de la serie de precios

2.1. Transformaciones (precio respuesta)

La idea central es no trabajar siempre directamente con P_t, sino con una versión transformada

$$Y_t = g(P_{0:T-1})_t,$$

que puede estabilizar la varianza o aproximar estacionariedad (retornos, diferencias, etc.). El módulo `py103` define:

y para las transformaciones que requieren casos escribimos por separado:

Todas estas transformaciones están registradas en el diccionario `TRANSFORM_FUNCS` y se aplican a un vector de precios `prices`.

2.2. Enmascaramiento y construcción de (X, y, dates)

`make_transformed_time_regression_data` hace:

1. Extrae el vector de precios $\mathbf{p} = (P_0, \ldots, P_{T-1})^\top$ de la columna `'close'`.
2. Aplica g para obtener $\mathbf{y}^{\text{raw}} = (Y_0, \ldots, Y_{T-1})^\top$.
3. Construye una máscara booleana

$$m_t = \mathbf{1}\{Y_t \text{ es finito}\},$$

   y la aplica tanto a Y_t como a las fechas. Esto elimina los `NaN` introducidos por transformaciones como retornos o diferencias en $t = 0$.
4. Define la covariable de regresión:

$$x_t = t, \quad t = 0, \ldots, N-1,$$

donde N es el número de puntos finitos restantes. El diseño queda como

$$X = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ N-1 \end{bmatrix} \in \mathbb{R}^{N\times 1}, \quad y = (Y_0, \ldots, Y_{N-1})^\top.$$

5. Devuelve (X, y, dates).

En resumen: para cada transformación tenemos una serie transformada y_t y un índice temporal x_t sobre el que haremos regresión.

3. Modelo de regresión en el tiempo: Elastic Net polinomial

3.1. Modelo polinomial en el tiempo

Para un grado polinomial $d \geq 1$, definimos el vector de características

$$\phi_d(x_t) = \left(x_t,\ x_t^2,\ \ldots,\ x_t^d\right)^\top \in \mathbb{R}^d.$$

El modelo de tendencia en el espacio transformado es:

$$y_t = \beta_0 + \beta^\top \phi_d(x_t) + \varepsilon_t, \quad t \in \mathcal{I}_{\text{train}},$$

donde $\beta_0 \in \mathbb{R}$, $\beta \in \mathbb{R}^d$ y ε_t es el residuo.

En `build_elastic_net_model` esto se implementa como:

`[ PolynomialFeatures(degree=d) ]` $\rightarrow$ `StandardScaler` $\rightarrow$ `ElasticNet`

con parámetros:

- `alpha` > 0 (intensidad de regularización),
- `l1_ratio` $\in [0, 1]$ (mezcla Lasso/Ridge),
- `poly_degree` $= d$.

3.2. Penalización Elastic Net

Sea $\tilde{X} \in \mathbb{R}^{N_{\text{train}} \times d}$ la matriz de características (posiblemente estandarizadas) y $\tilde{y} \in \mathbb{R}^{N_{\text{train}}}$ los valores transformados. La Elastic Net resuelve, en esencia,

$$\min_{\beta_0,\beta} \frac{1}{N_{\text{train}}} \sum_{t \in \mathcal{I}_{\text{train}}} \left(\tilde{y}_t - \beta_0 - \tilde{X}_t^\top \beta\right)^2 + \alpha\Big[(1-\eta)\frac{\|\beta\|_2^2}{2} + \eta\|\beta\|_1\Big],$$

donde $\eta =$ `l1_ratio` controla el balance entre penalización ℓ_2 (Ridge) y ℓ_1 (Lasso).

4. Split temporal y métricas de error ponderadas

4.1. Split 60/20/20

`temporal_split` y `_segment_indices` definen:

$$\begin{aligned} \mathcal{I}_{\text{train}} &= \{0, \ldots, N_{\text{train}} - 1\}, \\ \mathcal{I}_{\text{val}} &= \{N_{\text{train}}, \ldots, N_{\text{train}} + N_{\text{val}} - 1\}, \\ \mathcal{I}_{\text{test}} &= \{N_{\text{train}} + N_{\text{val}}, \ldots, N - 1\}. \end{aligned}$$

4.2. RMSE no ponderado

La función

$$\texttt{rmse}(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

se calcula en cada segmento (train, val, test, train_val) mediante `compute_segment_metrics`.

4.3. RMSE linealmente ponderado

Para enfatizar los últimos días, definimos pesos

$$w_i \propto i + 1, \quad i = 0, \ldots, n-1, \quad \text{con} \quad \sum_{i=0}^{n-1} w_i = 1.$$

Entonces

$$\texttt{weighted\_rmse\_linear\_time} = \sqrt{\sum_{i=0}^{n-1} w_i (y_i - \hat{y}_i)^2}.$$

Esto se usa:

- como *métrica de evaluación* para cada segmento,
- y también, más adelante, como esquema de *entrenamiento* (usando `sample_weight`) en el modo `train_weight_mode = "linear"`.

4.4. RMSE exponencialmente ponderado

Definimos pesos exponenciales en el tiempo, controlando la razón "último vs primero":

$$\frac{w_{n-1}}{w_0} = R, \quad R = \texttt{ratio\_last\_first}.$$

Esto induce

$$\gamma^{n-1} = R \quad \Rightarrow \quad \gamma = R^{1/(n-1)},$$

y

$$w_i \propto \gamma^i, \quad i = 0, \ldots, n-1, \quad \sum_i w_i = 1.$$

La métrica

$$\texttt{weighted\_rmse\_exponential\_time} = \sqrt{\sum_{i=0}^{n-1} w_i (y_i - \hat{y}_i)^2}$$

asigna mucho más peso a los días más recientes. De nuevo, se usa tanto como métrica de evaluación como, en `_fit_and_evaluate_with_train_weight_mode`, como esquema de *entrenamiento* (`train_weight_mode = .exp"`).

5. Espacio transformado vs espacio de precios

5.1. Métricas en el espacio transformado

Cada vez que entrenamos un modelo, `_fit_and_evaluate` y `_fit_and_evaluate_with_train_weig` devuelven, para cada segmento $S \in \{\text{train}, \text{val}, \text{test}, \text{train\_val}\}$:

$$\text{metrics}_S^{(y)} = \{\texttt{rmse},\ \texttt{rmse\_w\_linear},\ \texttt{rmse\_w\_exp}\},$$

es decir, el error en el espacio de la transformación Y_t (log-precios, retornos, etc.).

5.2. Métricas en el espacio original de precios

Para poder comparar transformaciones entre sí en las unidades originales (precio), se introdujo `compute_price_space_metrics_for_model`:

1. Reconstruye $(X_{\text{all}}, y_{\text{all}}, \text{dates})$ para una transformación g fija.
2. Calcula $\hat{y}_t = f_{\hat{\theta}}(x_t)$ para todos los puntos.
3. Aplica una *detransformación* h para aproximar P_t:
   - `çlose"`: $h(\hat{y}_t) = \hat{y}_t$.
   - `"log_close"`: $h(\hat{y}_t) = \exp(\hat{y}_t)$.
   - `"sqrt_close"`: $h(\hat{y}_t) = (\text{máx}\{\hat{y}_t, 0\})^2$.
   - `"zscore_log_close"`: $h(\hat{y}_t) = \exp(\hat{y}_t \sigma_{\log P} + \mu_{\log P})$.
   - `"diff_close"`:
   $$\begin{aligned} \hat{P}_0 &= P_0, \\ \hat{P}_t &= \hat{P}_{t-1} + \hat{y}_t, \quad t \geq 1. \end{aligned}$$
   - `"simple_return"`:
   $$\begin{aligned} \hat{P}_0 &= P_0, \\ \hat{P}_t &= \hat{P}_{t-1}(1 + \hat{y}_t), \quad t \geq 1. \end{aligned}$$
   - `"log_return"`:
   $$\begin{aligned} \widehat{\log P}_0 &= \log P_0, \\ \widehat{\log P}_t &= \widehat{\log P}_{t-1} + \hat{y}_t, \\ \hat{P}_t &= \exp(\widehat{\log P}_t). \end{aligned}$$
4. Con $\hat{P}_t$ y P_t alineados, se define para cada segmento S el mismo trío de métricas:
   $$\text{metrics}_S^{(P)} = \{\texttt{rmse}, \texttt{rmse\_w\_linear}, \texttt{rmse\_w\_exp}\},$$
   ahora en unidades de precio.

Así, `run_transform_experiment` y `run_all_transforms_experiment` imprimen simultáneamente métricas en el espacio transformado y en el espacio de precios.

6. Experimentos de alto nivel

6.1. `run_transform_experiment`

Para un *solo* tipo de transformación (p. ej. `"log_close"`):

1. Descarga la serie `df` de Yahoo!.
2. Construye (X, y, dates).
3. Ajusta un modelo Elastic Net polinomial de grado `poly_degree`.
4. Calcula la varianza de y:

$$\text{Var}(Y) = \frac{1}{N} \sum_{t=0}^{N-1} (Y_t - \bar{Y})^2.$$

5. Calcula métricas en el espacio transformado y en el espacio de precios, para train/val/test/train_val.
6. Devuelve:
   - un diccionario `result` con:
     - `"transform"`,
     - `"variance"`,
     - `"metrics"`,
   - el modelo ajustado,
   - el `DataFrame` de precios.

6.2. `run_all_transforms_experiment`

Para una lista de transformaciones (`None` ⇒ todas las de `TRANSFORM_FUNCS`):

1. Descarga una sola vez la serie de precios.
2. Para cada transformación g:
   - ajusta un modelo Elastic Net,
   - almacena sus métricas en espacio transformado y de precios.
3. Imprime un resumen por transformación.
4. Devuelve:

- `results[name]` con varianza y métricas,
- `models[name]` con el modelo para esa transformación,
- el `DataFrame` de precios.

6.3. `run_logprice_experiment`

Es un *wrapper* de retrocompatibilidad que fija `transform_name = "log_close"` y devuelve directamente `result["metrics"]`, el modelo y el `DataFrame`.

7. Entrenamiento ponderado en el tiempo

7.1. `_fit_and_evaluate_with_train_weight_mode`

Esta función extiende el entrenamiento para permitir tres esquemas de peso en los *datos de entrenamiento*:

- `"plain"`: sin pesos (`sample_weight = None`).
- `"linear"`: `sample_weight` $\propto i + 1$.
- `."exp"`: `sample_weight` exponenciales con razón `ratio_last_first`.

Formalmente, en lugar del MSE simple

$$\frac{1}{N_{\text{train}}} \sum_{t \in \mathcal{I}_{\text{train}}} (y_t - \hat{y}_t)^2,$$

minimizamos

$$\sum_{t \in \mathcal{I}_{\text{train}}} w_t (y_t - \hat{y}_t)^2,$$

con w_t definidos según el modo.

7.2. `run_weighted_training_comparison_single_transform`

Para una transformación fija (p. ej. `"log_close"`):

1. Ajusta tres modelos (uno por modo de peso en entrenamiento).
2. Para cada modo:
   - calcula métricas en espacio transformado,
   - calcula métricas en espacio de precios.
3. Imprime, en particular, las métricas de validación para comparar cuál esquema se comporta mejor.

7.3. `plot_weighted_training_comparison`

Genera una figura con:

- panel izquierdo: serie transformada Y_t y tres curvas de tendencia (plain / linear / exp) en el espacio de la transformación,
- panel derecho: precios P_t y tres tendencias destransformadas.

8. Diagnóstico de residuos y normalidad

8.1. `compute_validation_residuals_transform_and_price`

Para un modelo ajustado:

1. Reconstruye y_t y $\hat{y}_t$.
2. Extrae solo el segmento de validación.
3. Destransforma para obtener P_t y $\hat{P}_t$ en validación.
4. Devuelve:
   - `resid_y_val` $= y_t - \hat{y}_t$,
   - `resid_p_val` $= P_t - \hat{P}_t$,
   - fechas y valores ajustados correspondientes.

8.2. `compute_normality_stats`

Dado un vector de residuos $\{e_i\}$, calcula:

$$\begin{aligned}
\bar{e} &= \frac{1}{n}\sum_i e_i, \\
s^2 &= \frac{1}{n-1}\sum_i (e_i - \bar{e})^2, \\
\text{skew} &= \frac{1}{n}\sum_i \left(\frac{e_i - \bar{e}}{s}\right)^3, \\
\text{kurt_excess} &= \frac{1}{n}\sum_i \left(\frac{e_i - \bar{e}}{s}\right)^4 - 3.
\end{aligned}$$

Además, calcula la estadística de Jarque–Bera:

$$\mathrm{JB} = \frac{n}{6}\left(\mathrm{skew}^2 + \frac{\mathrm{kurt\_excess}^2}{4}\right),$$

con p–valor bajo la aproximación χ^2_2.

8.3. `plot_validation_residual_diagnostics`

Para un modelo y transformación dados, puede trabajar en:

- espacio transformado: residuos $e_t = y_t - \hat{y}_t$,
- espacio de precios: residuos $e_t = P_t - \hat{P}_t$.

Produce una figura 2×2 con:

1. Histograma + curva Normal ajustada.
2. QQ-plot frente a una Normal.
3. Residuos vs valores ajustados.
4. Residuos vs fecha.

Además, imprime $\bar{e}$, s, t–estadístico para $H_0 : \bar{e} = 0$, p–valor, skewness y kurtosis-exceso.

9. Selección de modelo regularizada por normalidad

9.1. `run_normality_regularized_model_single_transform`

Para una transformación fija (p. ej. `"log_close"`) y una grilla de hiperparámetros:

$$\alpha \in \mathcal{A}, \quad \eta \in \mathcal{L}, \quad d \in \mathcal{D}, \quad \text{modo} \in \{\text{plain, linear, exp}\},$$

se exploran todas las combinaciones.

Para cada combinación se hace:

1. Entrenamiento con el modo de peso escogido (`_fit_and_evaluate_with_train_weight_mode`).

2. Cálculo de métricas en el espacio de precios (en particular RMSE de validación $\mathrm{RMSE}_{\mathrm{val}}^{(P)}$).

3. Cálculo de residuos de validación en el espacio escogido (`normality_space = "price"` o `"transform"`), y a partir de ellos la estadística JB.

4. Definición de una función objetivo

$$\mathrm{obj} = w_{\mathrm{rmse}} \cdot \mathrm{RMSE}_{\mathrm{val}}^{(P)} + w_{\mathrm{jb}} \cdot \mathrm{JB},$$

   y se almacena todo en un diccionario de candidatos.

El "mejor" modelo es el que minimiza obj. Se devuelve:

- `best_summary`: hiperparámetros óptimos, métricas y estadísticas de normalidad,
- `best_model`: el `Pipeline` entrenado,
- `all_candidates`: detalles de todas las combinaciones,
- el `DataFrame` de precios.

10. Visualizaciones de alto nivel

10.1. `plot_transform_trend` y `plot_log_trend`

- `plot_transform_trend` muestra la serie transformada Y_t separada por colores (train/val/test) y la curva de tendencia ajustada (Elastic Net) sobre todo el rango temporal.
- `plot_log_trend` es un wrapper específico para `"log_close"`.

10.2. `plot_price_with_exp_trend`

Es un caso particular para log-precios: toma un modelo entrenado en `"log_close"`, destransforma por exponencial y dibuja la serie P_t junto con la curva $\exp(\hat{y}_t)$ (separando colores por segmento).

10.3. `plot_all_transform_trends` y `plot_all_destransformed_price_trends`

Para cada transformación y su modelo correspondiente:

- `plot_all_transform_trends` recorre `transform_names` y llama a `plot_transform_trend` para cada una.
- `plot_all_destransformed_price_trends` hace lo mismo pero en el espacio de precios, usando `plot_destransformed_price_trend`.

10.4. `subplot_all_transforms_and_destransforms`

Construye una figura compacta en forma de rejilla:

- Cada fila corresponde a una transformación.
- Columna izquierda: serie transformada Y_t con su tendencia.
- Columna derecha: P_t con la tendencia destransformada.

Esto permite ver de un vistazo cómo cambian la forma de la serie y la suavidad de la tendencia según la transformación elegida.

11. Resumen conceptual

En síntesis, el módulo:

1. Descarga una serie de precios y la convierte en un problema de regresión en el tiempo.
2. Aplica múltiples transformaciones sobre el precio (niveles, log, raíz, diferencias, retornos, z-score).
3. Ajusta modelos Elastic Net polinomiales en el índice temporal, con posibilidad de entrenar enfatizando más los datos recientes.
4. Evalúa el ajuste tanto en el espacio transformado como en el espacio original de precios, usando RMSE no ponderado y dos variantes de RMSE ponderado.
5. Genera figuras que muestran:
    - la tendencia en el espacio transformado,

- la tendencia destransformada en el espacio de precios,
- la comparación de distintos esquemas de peso.

6. Analiza los residuos de validación (histograma, QQ-plot, residuo vs ajustado vs tiempo) y cuantifica su normalidad.

7. Implementa una selección de modelo que combina "buen desempeño predictivo" (bajo RMSE de validación) con "residuos lo más normales posible" (bajo Jarque–Bera), explorando una grilla de hiperparámetros y modos de peso.

En notación compacta, el corazón del enfoque puede resumirse como:

$$\boxed{\hat{\theta} = \arg\min_{\theta} \Big[\underbrace{\mathrm{RMSE}_{\mathrm{val}}^{(P)}(\theta)}_{\text{ajuste en precio}} + \lambda \quad \underbrace{\mathrm{JB}\big(e_{\mathrm{val}}^{(*)}(\theta)\big)}_{\text{normalidad de residuos}} \Big]}$$

donde θ codifica $(\alpha, \eta, d, \text{modo de peso})$, $e_{\mathrm{val}}^{(*)}$ son los residuos (en espacio de precios o transformado) y λ captura el peso relativo que damos a la normalidad frente al RMSE.

Transformaciones, regresión y "destransformación" de la serie

En esta sección explicamos con todo detalle qué está pasando en el módulo `py103.py`, especialmente en el caso de la transformación por diferencias (`diff_close`), y aclaramos exactamente qué se está "destransformando" cuando dibujamos la serie en el espacio original de precios.

1. Notación básica

- Sea $t \in \{0, 1, \ldots, n-1\}$ el índice de tiempo discreto (días).
- Sea P_t el precio de cierre (`close`) de NVDA en el día t.
- Sea $\mathcal{D} = \{(t, P_t)\}_{t=0}^{n-1}$ nuestra serie temporal.
- Sea $g(\cdot)$ una transformación escalar aplicada a P_t.
- Sea $y_t = g(P_\bullet)$ el target transformado que realmente usamos en la regresión.

En el código, usamos varios g distintos:

$$\begin{aligned}
\texttt{close}: &\quad y_t = P_t, \\
\texttt{log_close}: &\quad y_t = \log P_t, \\
\texttt{diff_close}: &\quad y_t = P_t - P_{t-1}, \\
\texttt{simple_return}: &\quad y_t = \frac{P_t - P_{t-1}}{P_{t-1}}, \\
\texttt{log_return}: &\quad y_t = \log P_t - \log P_{t-1},
\end{aligned}$$

etc. (para algunos t iniciales, la serie se recorta o se rellena de forma conveniente; en el código lo manejamos con máscaras y `dropna`).

2. Construcción del problema de regresión

El modelo de regresión que usamos es un polinomio en el tiempo t, regularizado con Elastic Net.

2.1. Variable explicativa: índice de tiempo

Definimos el índice de tiempo

$$x_t = t \in \mathbb{R}, \qquad t = 0, 1, \ldots, n-1.$$

En notación matricial,

$$X = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ n-1 \end{bmatrix} \in \mathbb{R}^{n \times 1}.$$

Si usamos un polinomio de grado d en t, internamente aplicamos `PolynomialFeatures`:

$$\phi_d(t) = \big(t, t^2, \ldots, t^d\big),$$

y el diseño completo es

$$Z = \big[\phi_d(t_0)^\top;\ \phi_d(t_1)^\top;\ \ldots;\ \phi_d(t_{n-1})^\top\big] \in \mathbb{R}^{n \times d}.$$

2.2. Variable respuesta: transformación de la serie

Dado un esquema de transformación g, construimos

$$y_t = g(P_\bullet), \qquad t \in \mathcal{I}_g,$$

donde $\mathcal{I}_g \subseteq \{0, \ldots, n-1\}$ es el conjunto de instantes para los que la transformación está bien definida (por ejemplo, para `diff_close` empezamos en $t = 1$ y ajustamos índices).

En forma vectorial escribimos

$$y = (y_t)_{t \in \mathcal{I}_g} \in \mathbb{R}^m,$$

donde $m = |\mathcal{I}_g|$.

2.3. Separación en train / val / test

Ordenando por tiempo, dividimos y y X en tres segmentos contiguos:

$$\begin{aligned} \text{train: } & t = 0, \ldots, n_{\text{train}} - 1, \\ \text{val: } & t = n_{\text{train}}, \ldots, n_{\text{train}} + n_{\text{val}} - 1, \\ \text{test: } & t = n_{\text{train}} + n_{\text{val}}, \ldots, n-1. \end{aligned}$$

En el código usamos proporciones

$$\texttt{train\_frac} = 0{,}6, \quad \texttt{val\_frac} = 0{,}2, \quad \text{resto} = 0{,}2 \text{ para test.}$$

3. Modelo de regresión en el espacio transformado

El modelo es siempre una regresión lineal sobre las features polinomiales, regularizada con Elastic Net. En notación estadística:

$$y_t = f_\theta(t) + \varepsilon_t, \qquad t \in \mathcal{I}_g,$$

donde

$$f_\theta(t) = \beta_0 + \beta_1 t + \beta_2 t^2 + \cdots + \beta_d t^d$$

(es decir, el modelo es lineal en los parámetros β, pero no en t), y ε_t es el error.

El ajuste se hace resolviendo

$$\hat{\theta} = \arg\min_\theta \left\{ \sum_{t \in \text{train}} \big(y_t - f_\theta(t)\big)^2 + \lambda \left[(1-\alpha)\|\theta\|_2^2 + \alpha \|\theta\|_1 \right] \right\},$$

donde:

- $\lambda =$ `alpha` en el código.
- $\alpha =$ `l1_ratio` controla la mezcla Lasso/Ridge.

El modelo ajustado produce, para cada t,

$$\hat{y}_t = f_{\hat{\theta}}(t).$$

4. Caso especial: transformación de diferencias

Para `diff_close`, definimos

$$D_t = P_t - P_{t-1}, \qquad t = 1, \ldots, n-1.$$

En la práctica:

- Acomodamos los índices para que $y_t = D_t$ esté alineado con el conjunto de fechas correspondiente.
- Ajustamos el modelo
$$D_t = f_\theta(t) + \varepsilon_t.$$

Tras ajustar, tenemos un estimador suave de las diferencias diarias:

$$\hat{D}_t = f_{\hat{\theta}}(t).$$

5. "Destransformación": reconstruir un *trend* en el espacio de precios

El punto clave: **sólo destransformamos la salida del modelo $\hat{y}_t$, no la serie observada y_t.**

5.1. Transformaciones directas (niveles, logaritmos, etc.)

Cuando g es biyectiva punto a punto $P_t \mapsto y_t$, por ejemplo:

- `close`: $y_t = P_t$, entonces $h(y_t) = y_t$.
- `log_close`: $y_t = \log P_t$, entonces $h(y_t) = e^{y_t}$.

Definimos una inversa h tal que

$$h(g(P_t)) = P_t.$$

En este caso, el *trend* de precios que dibujamos es

$$\hat{P}_t = h(\hat{y}_t),$$

y la banda de confianza en el espacio de precios es

$$\hat{P}_t^{\text{low}} = h\big(\hat{y}_t + q_{\text{low}}\big), \qquad \hat{P}_t^{\text{high}} = h\big(\hat{y}_t + q_{\text{high}}\big),$$

donde $q_{\text{low}}, q_{\text{high}}$ son cuantil(es) empíricos de los residuos en el *espacio transformado*.

5.2. Transformaciones acumulativas (diferencias, rendimientos)

Para `diff_close`, el transform es

$$y_t = D_t = P_t - P_{t-1}.$$

Ya no existe una inversa punto a punto h que dependa sólo de y_t; la información de P_t está codificada *acumulativamente* en todos los incrementos previos. La relación verdadera es

$$P_t = P_0 + \sum_{s=1}^{t} D_s.$$

En el modelo, sustituimos D_s por el valor ajustado $\hat{D}_s$:

$$\hat{P}_t = P_0 + \sum_{s=1}^{t} \hat{D}_s = P_0 + \sum_{s=1}^{t} f_{\hat{\theta}}(s).$$

En código se implementa como recursión:

$$\begin{aligned} \hat{P}_0 &= P_0, \\ \hat{P}_t &= \hat{P}_{t-1} + \hat{D}_t, \qquad t = 1, \ldots, n-1. \end{aligned}$$

Esto produce un **trend suave** en el espacio de precios:

$$\hat{P}_t = \text{“precio suavizado”}$$

porque $\hat{D}_t$ es una función suave de t (polinomio regularizado).

Análogamente, si usamos una banda empírica en el espacio de diferencias:

$$\hat{D}_t^{\text{low}} = \hat{D}_t + q_{\text{low}}, \qquad \hat{D}_t^{\text{high}} = \hat{D}_t + q_{\text{high}},$$

reconstruimos las bandas de precio por

$$\begin{aligned} \hat{P}_0^{\text{low}} &= P_0, \qquad \hat{P}_t^{\text{low}} = \hat{P}_{t-1}^{\text{low}} + \hat{D}_t^{\text{low}}, \\ \hat{P}_0^{\text{high}} &= P_0, \qquad \hat{P}_t^{\text{high}} = \hat{P}_{t-1}^{\text{high}} + \hat{D}_t^{\text{high}}. \end{aligned}$$

Conclusión importante. En el caso `diff_close`:

- Nunca reconstruimos P_t usando los verdaderos incrementos D_t en la gráfica de *trend*.
- Siempre usamos los *incrementos ajustados* $\hat{D}_t$ para obtener una curva suave $\hat{P}_t$.
- La serie observada P_t se dibuja tal cual, con sus brincos; la curva $\hat{P}_t$ es un suavizado.

Por eso, visualmente, la línea negra (trend) es continua y suave, mientras que la serie de puntos/segmentos de la serie cruda puede ser más “salto a salto”.

6. Qué se destransforma y qué no

Podemos resumir así:

1. **Espacio transformado.**
   - Datos observados: $y_t = g(P_\bullet)$.
   - Modelo: $\hat{y}_t = f_{\hat{\theta}}(t)$.
   - Residuos: $r_t = y_t - \hat{y}_t$.
2. **Espacio de precios (destransformado).**
   - Serie observada: P_t se dibuja tal cual (`df[çlose"]`).
   - **Sólo destransformamos:**
     - La curva de trend $\hat{y}_t \mapsto \hat{P}_t$.
     - Las bandas de confianza $\hat{y}_t + q \mapsto \hat{P}_t^{(\cdot)}$.

En particular, nunca hacemos

$$y_t \xrightarrow{h} P_t^\star$$

para luego sustituir P_t por $P_t^\star$; es decir, no "destransformamos" la serie observada, sólo la salida del modelo.

7. Ejemplo compacto: `diff_close`

Para fijar ideas, el pipeline de `diff_close` es:

$$\begin{aligned} D_t &= P_t - P_{t-1}, \\ D_t &= f_\theta(t) + \varepsilon_t, \\ \hat{D}_t &= f_{\hat{\theta}}(t), \\ \hat{P}_0 &= P_0, \\ \hat{P}_t &= \hat{P}_{t-1} + \hat{D}_t. \end{aligned}$$

Visualmente:

- En la gráfica de **transformación** (`diff_close`):

D_t observado (salta) $\quad vs \quad$ $\hat{D}_t$ suave.

- En la gráfica de **precio**:

$$\boxed{P_t \text{ observado (salta)} \quad vs \quad \hat{P}_t = P_0 + \sum_{s=1}^{t} \hat{D}_s \text{ suave.}}$$

8. Resumen simbólico

La idea central que explica tu observación sobre la línea continua es:

(1) Transformamos la serie: $y_t = g(P_\bullet)$.
(2) Ajustamos $y_t \approx f_\theta(t) \;\Rightarrow\; \hat{y}_t$.
(3) Construimos un *trend* en precios: $\hat{P}_t = H(\hat{y}_{0:t})$,
donde H integra o invierte g usando sólo la salida del modelo.

En el caso de diferencias, H es la suma acumulada de incrementos ajustados; como éstos son suaves, la curva $\hat{P}_t$ es también suave y no reproduce los saltos originales de P_t, sino su tendencia estimada.