Random Forest

TimeSeries Team

Ngày 22 tháng 9 năm 2025

Mục lục

I. Nhắc lại Decision Tree (Phân loại & Hồi quy)

Khái niệm nhanh

- **Decision Tree** là mô hình phân tách không gian đặc trưng bằng chuỗi điều kiện tại các *nút*; đi từ *nút gốc* tới *lá* để đưa ra dự đoán.
- **Phân loại**: dự đoán *nhãn*; tại lá có phân phối tần suất lớp ⇒ suy ra *xác suất*.
- **Hồi quy**: dự đoán *giá trị số* (thường là trung bình/median của các điểm rơi vào lá).

Cách cây học phép tách (Splitting)

Phân loại

Mục tiêu là làm "độ thuần khiết" tại nút tăng lên sau tách. Hai thước đo phổ biến:

Gini Impurity. Với nút S có phân phối lớp $\{p_k\}$,

$$G(S) \;=\; 1 - \sum_k p_k^2, \qquad p_k = \frac{\#\{y = k \text{ trong } S\}}{|S|}. \tag{1}$$

Cho đặc trưng F tạo các nhánh con S_v, *Gini gain*:

$$GG(S,F) \;=\; G(S) \;-\; \sum_v \frac{|S_v|}{|S|}\, G(S_v). \tag{2}$$

Entropy / Information Gain.

$$H(S) = -\sum_k p_k \log p_k, \qquad IG(S,F) = H(S) - \sum_v \frac{|S_v|}{|S|}\, H(S_v). \tag{3}$$

Ví dụ tính tay (rất ngắn). Nút có 10 mẫu, 6 dương/4 âm ⇒ $G(S) = 1 - (0.6^2 + 0.4^2) = 0.48$. Tách theo F thành *trái* (5/1) và *phải* (1/3):

$$G_{\text{trái}} = 1 - \left(\tfrac{5}{6}\right)^2 - \left(\tfrac{1}{6}\right)^2 = 0.278, \quad G_{\text{phải}} = 1 - \left(\tfrac{1}{4}\right)^2 - \left(\tfrac{3}{4}\right)^2 = 0.375.$$

Gini gain:

$$GG = 0.48 - \left(\tfrac{6}{10}\cdot 0.278 + \tfrac{4}{10}\cdot 0.375\right) = 0.48 - (0.1668 + 0.15) = 0.1632.$$

So sánh với các split khác; chọn split có GG (hoặc IG) lớn nhất.

Hồi quy

Mục tiêu: giảm *sai số bình phương* tại nút.

$$\text{MSE}(S) = \frac{1}{|S|}\sum_{i\in S}(y_i - \bar{y}_S)^2, \qquad \bar{y}_S = \frac{1}{|S|}\sum_{i\in S} y_i. \tag{4}$$

Độ lợi theo đặc trưng F:

$$\Delta(S,F) \;=\; \text{MSE}(S) - \sum_v \frac{|S_v|}{|S|}\,\text{MSE}(S_v), \quad \text{chọn } \Delta \text{ lớn nhất}. \tag{5}$$

Dự đoán tại lá là trung bình (hoặc median để robust hơn với ngoại lai).

Xử lý kiểu dữ liệu & ngưỡng

- **Biến số (numeric)**: tìm *ngưỡng* tối ưu bằng cách duyệt các điểm biên giữa các giá trị đã sắp xếp.
- **Biến phân loại (categorical)**:
  - Nhị phân (Có/Không): tách trực tiếp.
  - Nhiều mức: xét các nhóm con (tuỳ triển khai) hoặc mã hoá (one-hot/ordinal).
- **Thiên lệch do nhiều mức**: Information Gain có thể thiên lệch; cân nhắc Gain Ratio trong thực thi.

Dừng sớm & Cắt tỉa (chống overfit)

Cây càng sâu ⇒ *bias* giảm nhưng *variance* tăng. Điều chỉnh bằng:

Dừng sớm (pre-pruning)

- `max_depth`: giới hạn độ sâu.
- `min_samples_split`: tối thiểu số mẫu để tách nút.
- `min_samples_leaf`: tối thiểu số mẫu tại lá (giúp làm trơn xác suất & hồi quy).
- `max_leaf_nodes`, `max_features` (nếu cần).

Cắt tỉa sau huấn luyện (post-pruning)

Ý tưởng *cost–complexity pruning*:

$$\mathrm{Risk}_\alpha(T) \;=\; \mathrm{EmpiricalError}(T) + \alpha \cdot |T|, \tag{6}$$

tăng α ưu tiên cây nhỏ; chọn α bằng cross-validation.

Dự đoán & xác suất ở lá

Phân loại: $\hat{p}(k \mid L) = \frac{\#\{y=k \text{ trong } L\}}{|L|}$, nhãn là lớp có xác suất lớn nhất (có thể hiệu chỉnh xác suất bằng Platt/Isotonic nếu cần).
Hồi quy: dự đoán là trung bình (hoặc median) y của điểm trong lá.

Độ phức tạp, ưu & nhược

Độ phức tạp (CART, xấp xỉ): xây cây $O(n\,d\,\log n)$ đến $O(n\,d\,\log^2 n)$; dự đoán mỗi điểm $O(\text{depth})$.
Ưu: trực quan, giải thích được; xử lý số & phân loại; không cần chuẩn hoá; nắm tương tác phi tuyến.
Nhược: dễ overfit nếu không kiểm soát; *variance* cao (nhạy dữ liệu) ⇒ động lực dùng *ensemble* (phần 2).

Checklist thực hành

1. Chia *train/val/test* (stratify nếu phân loại).
2. Chọn tiêu chí: Gini/Entropy (phân loại), MSE (hồi quy).
3. Khống chế độ sâu & cỡ lá: `max_depth`, `min_samples_leaf` (phân loại: 1–5; hồi quy: 5–50).
4. Chọn metric: AUC/F1 (mất cân bằng) hoặc RMSE/MAE (hồi quy).

5. Kiểm tra *learning curves*:
    - Train tốt, val kém ⇒ giảm độ sâu / tăng `min_samples_leaf`.
    - Cả train & val đều kém ⇒ tăng độ sâu / thêm đặc trưng.

Mã ví dụ (scikit-learn)

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, roc_auc_score

X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

clf = DecisionTreeClassifier(
    criterion="gini",       # or "entropy"
    max_depth=12,
    min_samples_leaf=2,
    random_state=42
)
clf.fit(X_train, y_train)
proba = clf.predict_proba(X_val)[:, 1]
print("F1:", f1_score(y_val, clf.predict(X_val)))
print("AUC:", roc_auc_score(y_val, proba))
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

reg = DecisionTreeRegressor(
    criterion="squared_error",  # MSE
    max_depth=16,
    min_samples_leaf=10,
    random_state=42
)
reg.fit(X_train, y_train)
pred = reg.predict(X_val)
rmse = mean_squared_error(y_val, pred, squared=False)
print("RMSE:", rmse)
```

Bias–Variance Trade-off & Prediction Errors

Decomposition của lỗi dự đoán (hồi quy, MSE)

Giả sử $y = f(x) + \varepsilon$, với $\mathbb{E}[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$. Gọi $\hat{f}_{\mathcal{D}}$ là mô hình học từ một mẫu huấn luyện ngẫu nhiên $\mathcal{D}$. Sai số kỳ vọng tại một điểm x phân rã thành:

$$\underbrace{\mathbb{E}_{\mathcal{D}}\Big[(y - \hat{f}_{\mathcal{D}}(x))^2\Big]}_{\text{Prediction error}} = \underbrace{\Big(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)] - f(x)\Big)^2}_{\text{Bias}^2} + \underbrace{\mathbb{V}\mathrm{ar}_{\mathcal{D}}\Big[\hat{f}_{\mathcal{D}}(x)\Big]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Irreducible noise}} . \tag{7}$$

Ý nghĩa.

- **Bias**: độ lệch có hệ thống giữa trung bình dự đoán và sự thật $f(x)$ (mô hình quá đơn giản ⇒ underfit).
- **Variance**: mức dao động của dự đoán khi thay đổi tập huấn luyện (mô hình quá linh hoạt ⇒ overfit).
- **Nhiễu không thể khử** σ^2: thuộc về dữ liệu, không loại bỏ được bằng mô hình.

Trade-off: giảm Bias thường tăng Variance và ngược lại; mục tiêu là điểm cân bằng cho lỗi tổng thể thấp nhất trên dữ liệu mới.

Cây quyết định dưới lăng kính Bias–Variance

- **Cây nông** (độ sâu nhỏ, lá lớn) ⇒ *bias cao, variance thấp*: mô hình không đủ linh hoạt để bắt cấu trúc phức tạp.
- **Cây sâu** (độ sâu lớn, lá nhỏ) ⇒ *bias thấp, variance cao*: mô hình bám sát cả nhiễu của tập huấn luyện.
- **Hàm ý thực hành**:
  - Dùng `max_depth`, `min_samples_leaf` để đặt cây ở vùng cân bằng.
  - Khi chỉ dùng *một cây*, kết quả rất nhạy dữ liệu ⇒ *variance cao*; động lực dùng **ensemble** (phần 2).

Liên hệ với Bagging/Random Forest, Boosting, Stacking

- **Bagging / Random Forest** (RF): huấn luyện nhiều cây độc lập trên các bootstrap khác nhau và (với RF) chọn ngẫu nhiên một phần đặc trưng tại *mỗi nút*.
  - Trung bình dự đoán ⇒ **giảm variance** mạnh, trong khi Bias không tăng đáng kể.
  - Có *OOB error* để ước lượng chất lượng mà không cần tập validation riêng.
- **Boosting**: ghép nối các cây yếu theo chuỗi để sửa sai có hệ thống ⇒ **giảm bias**; cần điều tiết variance bằng regularization/early stopping.
- **Stacking**: kết hợp nhiều mô hình dị loại bằng một meta-learner (từ dự đoán out-of-fold) để cân bằng lỗi tổng thể.

Trực quan nhỏ (hồi quy) với nhiễu Gaussian

Giả sử $f(x) = \sin x$ trên $[0, 2\pi]$, thêm nhiễu $\varepsilon \sim \mathcal{N}(0, 0.1^2)$.

- **Cây hồi quy** với `max_depth=2`: dự đoán bậc thang thô $\Rightarrow$ *bias lớn*.
- **Cây hồi quy** không giới hạn độ sâu (lá rất nhỏ): dự đoán "lượn" theo điểm train $\Rightarrow$ *variance lớn*.
- **Bagging** trung bình B cây trên các bootstrap: đường dự đoán mượt hơn (dao động giảm), MSE kiểm tra giảm.

Sai số trong phân loại (0–1 loss)

Không có decomposition sạch như MSE, nhưng chẩn đoán thực hành tương tự:

- **Bias cao**: mô hình dự đoán "ngây thơ" có hệ thống (ví dụ ưu tiên lớp đa số).
- **Variance cao**: độ chính xác OOB/validation dao động mạnh giữa các bootstrap/splits.
- **Giải pháp**: kiểm soát độ sâu/cỡ lá; dùng **RF/Bagging** để triệt dao động; cân nhắc calibration nếu cần xác suất chuẩn.

Chẩn đoán & chọn điểm cân bằng

- **Learning curves**: vẽ metric theo kích thước train.
  - Train tốt, validation kém $\Rightarrow$ *overfit/variance cao*: tăng `min_samples_leaf`, giảm `max_depth`, dùng **RF**, hoặc tăng dữ liệu.
  - Cả train & val đều kém $\Rightarrow$ *underfit/bias cao*: tăng độ sâu, thêm đặc trưng/biến đổi, cân nhắc **Boosting**.
- **OOB/Validation**: với RF, dùng OOB như "validation miễn phí" để chọn tham số.
- **Early stopping / cắt tỉa**: dừng ở điểm tốt trước khi variance tăng mạnh.

Bảng "triệu chứng $\leftrightarrow$ xử lý" (thực hành nhanh)

Triệu chứng	Hướng xử lý
Train cao, Val thấp (dao động giữa folds)	Giảm `max_depth`, tăng `min_samples_leaf`; dùng Bagging/RF; thêm dữ liệu; bớt nhiễu/feature selection.
Cả Train & Val thấp	Tăng độ sâu; thêm đặc trưng/biến đổi; chuyển sang Boosting; rà soát dữ liệu/nhãn.
Xác suất không chuẩn (quá tự tin hoặc quá dè dặt)	Sử dụng calibration (Platt/Isotonic); tăng `min_samples_leaf`.
Thời gian dự đoán lớn	Giảm độ sâu; cắt tỉa; với RF: giảm `n_estimators`, dùng `max_features` phù hợp.

Snippet minh hoạ (tuỳ chọn) — đo "dao động" như proxy của variance

```python
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error

rng = np.random.RandomState(42)

def make_data(n=80):
    X = np.sort(2*np.pi*rng.rand(n,1), axis=0)
    y = np.sin(X).ravel() + rng.normal(0, 0.1, size=n)
    return X, y

def fit_and_eval(max_depth, runs=30):
    xs = np.linspace(0, 2*np.pi, 200).reshape(-1,1)
    preds = []
    mses = []
    for _ in range(runs):
        X, y = make_data()
        Xt, yt = make_data(200)
        m = DecisionTreeRegressor(max_depth=max_depth, random_state=rng.randint(1e6))
        m.fit(X, y)
        mses.append(mean_squared_error(yt, m.predict(Xt)))
        preds.append(m.predict(xs))
    return np.mean(mses), np.std(np.vstack(preds), axis=0).mean()

for d in [2, 4, 8, None]:
    mse, avg_wiggle = fit_and_eval(d)
    print(d, "MSE=", round(mse,4), "  avg variance proxy=", round(avg_wiggle,4))
```

Kết nối sang Phần 2

- **Random Forest** (bagging + chọn ngẫu nhiên đặc trưng tại mỗi nút) nhắm trực tiếp vào *giảm variance* của cây sâu.
- **Boosting** nhắm vào *giảm bias* bằng cách học từ residual/lỗi còn lại.
- **Stacking** kết hợp các mô hình dị loại để tận dụng sự bù trừ lỗi khác nhau.

II. Random Forest & Ensemble Learning

Giới thiệu Random Forest (RF) qua Decision Tree & lợi ích

Random Forest (RF) là mô hình tổ hợp nhiều *Decision Tree (DT)* theo hai lớp ngẫu nhiên:

- **Ngẫu nhiên dữ liệu** (*bootstrap*): mỗi cây học trên một mẫu lấy *có hoàn lại* từ tập huấn luyện.
- **Ngẫu nhiên đặc trưng** (*random subspace*): tại *mỗi nút*, chỉ xét một *tập con* đặc trưng kích thước m trong tổng số p đặc trưng để chọn phép tách.

Dự đoán: phân loại dùng *bỏ phiếu đa số* (hoặc trung bình xác suất); hồi quy dùng *trung bình số học* của tất cả cây.

Vì sao RF "mặc định mạnh"?

- Cây đơn sâu ⇒ *variance* cao (nhạy tập train). RF tạo *nhiều cây đa dạng* ⇒ *tương quan* giữa cây thấp ⇒ **trung bình hoá** làm *giảm variance* mạnh mà *bias* không tăng đáng kể.
- Ít yêu cầu chuẩn hoá, *robust* với ngoại lai, có *OOB error* để ước lượng chất lượng mà không cần validation riêng.

Pipeline RF (6 bước thực dụng).

1. Chọn tham số: B cây (`n_estimators`), số đặc trưng mỗi nút m (`max_features`), `max_depth`, `min_samples_leaf`, `bootstrap=True`, tùy chọn `oob_score=True`.
2. Bootstrapping: lặp $b = 1..B$, tạo $D^{(b)}$ bằng lấy mẫu có hoàn lại từ D.
3. Xây cây với *random subspace*: ở mỗi nút rút ngẫu nhiên m đặc trưng trong p, chọn split giảm *impurity* (Gini/Entropy hoặc MSE) lớn nhất.
4. Quy tắc dừng: theo `max_depth`, `min_samples_split`, `min_samples_leaf`, hoặc *độ thuần khiết* của lá.
5. Dự đoán từng cây: phân loại trả nhãn/xác suất; hồi quy trả trung bình/median ở lá.
6. Gộp dự đoán: phân loại *vote*/trung bình xác suất; hồi quy *average*.

Góc nhìn phương sai của trung bình hoá. Giả sử phương sai một cây là σ^2, tương quan cặp giữa các cây là ρ. Phương sai của trung bình B cây xấp xỉ:

$$\mathrm{Var}\big(\overline{f}\big) \;\approx\; \rho\,\sigma^2 \;+\; \frac{(1-\rho)}{B}\,\sigma^2. \tag{8}$$

RF chủ động làm ρ nhỏ (nhờ bootstrap + random subspace), nên khi tăng B sẽ giảm phương sai đáng kể.

Tham số ảnh hưởng lớn (tư duy nhanh).

- `n_estimators` (B): tăng tới khi OOB/val bão hoà (thường 200–1000).
- `max_features` (m): phân loại $\approx \sqrt{p}$; hồi quy $\sqrt{p}$ hoặc tỉ lệ 0.3–0.6 (**rất quan trọng** để giảm tương quan giữa cây).
- `max_depth`, `min_samples_leaf`: khống chế overfit, làm mượt xác suất/hồi quy (phân loại: 1–5; hồi quy: 5–50).
- `class_weight` cho lệch nhãn; `max_samples` < 1.0 nếu muốn mỗi cây thấy $< 100\%$ dữ liệu (tăng đa dạng/giảm thời gian).

Ensemble Learning & các kỹ thuật (đặt RF vào bức tranh tổng thể)

Ensemble = kết hợp nhiều mô hình để cải thiện hiệu năng/ổn định. Ba nhánh chính:

(a) Bagging (Bootstrap Aggregating)

Huấn luyện *nhiều base learners cùng loại* trên *các bootstrap khác nhau*, sau đó *trung bình/vote*. Mục tiêu: **giảm variance** của base-learner có phương sai cao (ví dụ: cây sâu). RF = Bagging + *random subspace*, do đó giảm *tương quan giữa các cây* tốt hơn bagging thuần.

(b) Boosting (AdaBoost, Gradient Boosting, XGBoost/LightGBM/CatBoost)

Xây *chuỗi* mô hình yếu; mô hình sau học từ *residual/lỗi* còn lại của mô hình trước. Mục tiêu: **giảm bias** rất mạnh (cần điều tiết variance bằng regularization/early stopping). Trên tabular "sạch", boosting thường đạt điểm rất cao khi tuning tốt; RF là baseline ít kén chọn, nhanh ổn định.

(c) Stacking (Stacked Generalization)

Dùng nhiều *base models dị loại* (RF, GBDT, Logistic, SVM, NN, ...) $\rightarrow$ tạo *meta-features* là dự đoán *out-of-fold* $\rightarrow$ huấn luyện *meta-learner* (thường linear/GBDT) trên đó. Hữu ích khi lỗi của các base *khác nhau* (tương quan thấp).

Bảng 1: So sánh ngắn các họ ensemble

Tiêu chí	**RF (trọng tâm)**	**Bagging (thuần)**	**Boosting / Stacking**
Động lực	Giảm **variance** (đa dạng cây)	Giảm variance (không random subspace)	Giảm **bias** (Boosting); trộn **dị loại** (Stacking)
Phụ thuộc models	Độc lập (song song)	Độc lập	Tuần tự (Boosting) / 2 tầng (Stacking)
Nhạy tham số	Thấp–TB	Thấp	Cao (Boosting) / TB (Stacking cần OOF chuẩn)

Ví dụ cụ thể (tính tay) về RF

Bài toán **phân loại nhị phân** "Sunburn: Yes/No" với 3 đặc trưng phân loại: *Sunscreen* $\in$ {Yes, No}, *Hair* $\in$ {Light, Dark}, *Height* $\in$ {Short, Tall}.

Tập huấn luyện (6 mẫu)

ID	Sunscreen	Hair	Height	Sunburn
1	No	Light	Short	Yes
2	No	Dark	Tall	Yes
3	Yes	Light	Short	No
4	Yes	Dark	Tall	No
5	No	Light	Tall	Yes
6	Yes	Light	Tall	No

Tại nút gốc: Yes = 3, No = 3 $\Rightarrow$ Gini $= 1 - (0.5^2 + 0.5^2) = 0.5$.

Thiết lập RF: xây $B = 3$ cây, mỗi cây học trên một *bootstrap* size 6 (lấy có hoàn lại), và tại *mỗi nút* chỉ xét $m = 2$ *đặc trưng* ngẫu nhiên.

Cây 1 (Bootstrap #1)

Bootstrap chọn: $[1, 2, 3, 5, 5, 6]$. *Đặc trưng tại gốc* (ngẫu nhiên): $\{\text{Sunscreen}, \text{Hair}\}$.

[leftmargin=1.2em]

- Tách theo **Sunscreen**:
  - Sunscreen = No: {1,2,5,5} $\Rightarrow$ tất cả Yes $\Rightarrow$ Gini = 0.
  - Sunscreen = Yes: {3,6} $\Rightarrow$ tất cả No $\Rightarrow$ Gini = 0.
- Gain $= 0.5 - 0 = 0.5$ (tối đa) $\Rightarrow$ dừng ngay.

Cây 2 (Bootstrap #2)

Bootstrap chọn: $[1, 3, 3, 4, 6, 2]$. *Đặc trưng tại gốc*: $\{\text{Hair}, \text{Height}\}$ (không có Sunscreen ở gốc).

- Gốc: Yes = 2 (ID 1,2), No = 4 (ID 3,3,4,6) $\Rightarrow$ Gini $= 1 - (2/6)^2 - (4/6)^2 = 0.4444$.
- Thử tách **Hair**:
  - Hair = Light: Yes=1, No=3 $\Rightarrow G = 0.375$ (4 mẫu).
  - Hair = Dark: Yes=1, No=1 $\Rightarrow G = 0.5$ (2 mẫu).

  $\Rightarrow G_{\text{sau tách}} = \frac{4}{6} \cdot 0.375 + \frac{2}{6} \cdot 0.5 = 0.4167, \quad \text{Gain} = 0.0278.$
- Thử tách **Height**: cả hai nhánh đều $G = 0.4444 \Rightarrow$ Gain = 0.

Chọn **Hair** ở gốc (gain tốt hơn). Ở nhánh *Hair = Light*, tại tầng tiếp theo (được phép xét *Sunscreen*), ta tách:

- Sunscreen = No $\Rightarrow$ pure Yes; Sunscreen = Yes $\Rightarrow$ pure No.

Nhánh *Hair = Dark* cũng tách theo *Sunscreen* để tạo hai lá đối nghịch.

Cây 3 (Bootstrap #3)

Bootstrap chọn: $[1, 2, 3, 4, 6, 6]$. *Đặc trưng tại gốc*: {Sunscreen, Height}.

[leftmargin=1.2em]

- Tách theo **Sunscreen**:
  - Sunscreen = No: {1,2} ⇒ pure Yes.
  - Sunscreen = Yes: {3,4,6,6} ⇒ pure No.
- Gain = 0.5 (tối đa) ⇒ dừng tại gốc.

Dự đoán mẫu kiểm tra (ID 7)

Giả sử *ID 7*: Sunscreen = Yes, Hair = Light, Height = Short.

- **Cây 1**: gốc tách Sunscreen ⇒ Yes ⇒ **No**.
- **Cây 2**: gốc tách Hair; nhánh Hair=Light rồi tách Sunscreen ⇒ Yes ⇒ **No**.
- **Cây 3**: gốc tách Sunscreen ⇒ Yes ⇒ **No**.

Bỏ phiếu: 3/3 cây dự đoán **No** ⇒ **RF dự đoán: No**.

Nhìn lại lợi ích RF qua ví dụ.

- **Giảm variance**: Cây 1/3 tách sạch ở gốc; Cây 2 kém thuận lợi ở gốc (vì thiếu *Sunscreen*) nhưng sửa được ở tầng sau. Vote của rừng vẫn nhất quán.
- **Tương quan cây thấp**: bootstrap + random subspace khiến các cây ít giống nhau ⇒ trung bình hoá hiệu quả.
- **Dễ vận hành**: không cần chuẩn hoá; OOB hỗ trợ chọn tham số nhanh.

III. Xử lý dữ liệu thiếu (Missing Data) cho Random Forest

Vì sao phải xử lý thiếu?

- Nhiều triển khai Random Forest (RF) **không chấp nhận NaN** (ví dụ scikit-learn).
- Thiếu dữ liệu ảnh hưởng trực tiếp tới *tiêu chí tách* (Gini/Entropy/MSE) và *độ ổn định* của rừng.
- Trạng thái "thiếu" đôi khi **mang thông tin** (ví dụ một xét nghiệm bị bỏ qua vì đã có chẩn đoán rõ). Xoá dòng/cột có thể làm mất tín hiệu này.

Mục tiêu thực dụng: điền thiếu *nhất quán với cấu trúc dữ liệu*, tránh *rò rỉ thông tin*, và tối ưu *metric cuối* (AUC/F1/RMSE/MAE), không chỉ riêng RMSE của phần dữ liệu bị thiếu.

Bản chất thiếu dữ liệu: MCAR, MAR, MNAR

- **MCAR** (Missing Completely At Random): thiếu hoàn toàn ngẫu nhiên ⇒ các phương pháp đơn giản thường ổn.
- **MAR** (Missing At Random): thiếu phụ thuộc *các biến đã quan sát* ⇒ nên dùng phương pháp có điều kiện (Iterative/MissForest/Proximity).
- **MNAR** (Missing Not At Random): thiếu phụ thuộc *giá trị bị thiếu* hoặc yếu tố ẩn ⇒ khó nhất; cần hiểu nghiệp vụ để mô hình hoá.

Quy tắc vàng chống rò rỉ:

- Với cross-validation: *fit imputer trên train-fold*, *transform* trên val-fold/test-set.
- Với chuỗi thời gian: tuyệt đối *không nhìn tương lai* (chi tiết ở Phần 4).

Proximity Matrix của Random Forest (định nghĩa theo tài liệu)

Trực giác. Hai mẫu "gần nhau" nếu chúng *thường xuyên rơi cùng một lá* trên nhiều cây.

Cách xây dựng (chuẩn tài liệu).

[leftmargin=1.2em]

1. Chạy một RF (ban đầu có thể dùng dữ liệu đã *điền sơ bộ* bằng median/mode để mô hình chạy được).
2. Với mỗi cây, gom các mẫu ở *cùng lá*; với mỗi cặp (i, j) trong cùng lá, *cộng 1* vào đếm "đồng lá".
3. Sau khi duyệt hết các cây, **chuẩn hoá theo hàng** để tổng các lân cận của *mỗi hàng* $= 1$, và **đặt đường chéo** $= 0$ (không tự-giống).
   Ký hiệu ma trận kết quả là P với tính chất: $P_{ij} \in [0, 1]$, $P_{ii} = 0$, và $\sum_{j \neq i} P_{ij} = 1$ với mọi i.

Ý nghĩa: mỗi hàng $P_{i\cdot}$ là *trọng số lân cận* (phân phối xác suất) theo "hình học của rừng".

Dự đoán nhãn bằng Proximity (classification)

Khi dự đoán nhãn cho mẫu i:

$$W_c(i) = \sum_{j:\, y_j = c} P_{ij} \quad \text{(tổng trọng số proximity của lớp } c \text{ quanh } i\text{)}, \tag{9}$$

$$f_c = \frac{\#\{y_j = c\}}{N} \quad \text{(tần suất lớp trong tập huấn luyện)}, \tag{10}$$

$$S_c(i) = W_c(i) \times f_c \quad \text{(điểm lớp)}, \tag{11}$$

$$\hat{y}_i = \arg\max_c S_c(i). \tag{12}$$

Ví dụ ngắn: Hàng "1" có $W_{\text{Yes}} = 0.1$, $W_{\text{No}} = 0.9$; nếu $f_{\text{Yes}} = 1/3$, $f_{\text{No}} = 2/3$ thì $S_{\text{Yes}} = 0.033$, $S_{\text{No}} = 0.6$ $\Rightarrow$ dự đoán No.

Điền thiếu biến số bằng Proximity (regression imputation)

Với cột số m, nếu $x_{i,m}$ bị thiếu:

$$\widehat{x}_{i,m} = \sum_{j \neq i,\, x_{j,m} \text{ đầy đủ}} P_{ij}\, x_{j,m}. \tag{13}$$

Ví dụ "198.5" (đúng tinh thần tài liệu). Giả sử hàng "3" thiếu một đại lượng s; hàng này có lân cận "1, 2, 4" với *proximity hàng* $P_{3,\cdot} = [0.1, 0.1, 0.8]$ (đã chuẩn hoá, tổng = 1). Các giá trị quan sát ở lân cận: $s_1 = 125$, $s_2 = 180$, $s_4 = 210$.

$$\widehat{s}_3 = 125 \times 0.1 + 180 \times 0.1 + 210 \times 0.8 = 12.5 + 18.0 + 168.0 = \mathbf{198.5}.$$

Ghi chú: nếu một vài lân cận cũng thiếu ở cột m, loại chúng và *chuẩn hoá lại* hàng trên các lân cận còn quan sát (tổng vẫn = 1). Có thể "cắt ngưỡng" các P_{ij} nhỏ (ví dụ < 0.02) về 0 rồi chuẩn hoá lại, hoặc dùng *K-prox-NN* (chỉ K lân cận lớn nhất).

Quy trình RF+Proximity (lặp đến hội tụ)

1. **Khởi tạo**: impute đơn giản (median/mode) + tạo `is_missing` cho mỗi cột.
2. **Huấn luyện RF tạm** (nhẹ: `max_depth` vừa phải, `n_estimators` nhỏ).
3. **Tính Proximity** P (chuẩn hoá theo hàng, $P_{ii} = 0$).
4. **Cập nhật giá trị thiếu**:
   - Cột số: trung bình có trọng số proximity (đã chuẩn hoá theo hàng).
   - Cột phân loại: *mode* có trọng số proximity (cộng trọng số theo từng lớp, lấy lớn nhất).
5. **Lặp** các bước trên tới khi chênh lệch giữa hai vòng (NRMSE cho số, tỉ lệ sai phần bị che cho phân loại) *rất nhỏ*.
6. **Huấn luyện RF cuối** trên dữ liệu sau hội tụ.

Kinh nghiệm: thường 2–5 vòng là đủ; giữ RF tạm *nhẹ* để vòng lặp nhanh.

MissForest (RF-based imputation không dùng Proximity)

Tinh thần. Xem mỗi cột bị thiếu là *biến đích*; dùng RF (regressor/classifier) dự đoán từ các cột còn lại và *cập nhật dần*.

1. Impute khởi tạo (median/mode).
2. Sắp cột theo *tỉ lệ thiếu tăng dần*.
3. Với từng cột m:
   - **Số**: train RF Regressor dự đoán $x_{\cdot,m}$ từ $\{x_{\cdot,k}\}_{k\neq m}$; điền các ô thiếu (có thể dùng trung bình OOB).
   - **Phân loại**: RF Classifier, điền bằng lớp/ phân phối dự đoán.
4. Lặp nhiều vòng tới khi NRMSE/PFC ổn định.

Lai ghép: sau mỗi vòng MissForest, có thể *tinh chỉnh* các cột số bằng một bước *proximity-weighted* để mượt hơn.

Ví dụ mở rộng (numeric + categorical, hai cột thiếu)

Bối cảnh. Bảng 6 dòng (ID 1–6). Thiếu **BMI** ở ID 2, 5 (numeric) và thiếu **SkinType** ở ID 3 (categorical). Trước tiên dùng *median/mode* để điền sơ bộ → train RF tạm (giới hạn nhẹ) → tính Proximity *chuẩn tài liệu* (hàng tổng = 1, diag = 0).

(A) Proximity hàng cho ID 2 (tổng = 1). $P_{2,1} = 0.60$, $P_{2,3} = 0.35$, $P_{2,4} = 0.00$, $P_{2,6} = 0.05$.
BMI quan sát: $x_1 = 18.0$, $x_3 = 18.5$, $x_4 = 26.0$, $x_6 = 25.0$.
$\Rightarrow \widehat{\text{BMI}}_2 = 0.60\times18.0 + 0.35\times18.5 + 0.05\times25.0 = 10.8 + 6.475 + 1.25 = \mathbf{18.525}$.

(B) Proximity hàng cho ID 5 (tổng = 1). $P_{5,1} = 0.40$, $P_{5,3} = 0.15$, $P_{5,4} = 0.20$, $P_{5,6} = 0.25$.
$\Rightarrow \widehat{\text{BMI}}_5 = 0.40\times18.0 + 0.15\times18.5 + 0.20\times26.0 + 0.25\times25.0 = 7.2 + 2.775 + 5.2 + 6.25 = \mathbf{21.425}$.

(C) Proximity hàng cho ID 3 (điền *SkinType* — Fair/Medium/Dark). $P_{3,1} = 0.30$ (Fair), $P_{3,2} = 0.20$ (Fair), $P_{3,4} = 0.10$ (Medium), $P_{3,5} = 0.05$ (Medium), $P_{3,6} = 0.05$ (Dark).
Trọng số theo lớp: Fair = 0.50, Medium = 0.15, Dark = 0.05.
Tần suất lớp (giả sử): Fair = 0.4, Medium = 0.4, Dark = 0.2.
Điểm lớp: $S_{\text{Fair}} = 0.50\times0.4 = 0.20$, $S_{\text{Medium}} = 0.15\times0.4 = 0.06$, $S_{\text{Dark}} = 0.05\times0.2 = 0.01$
$\Rightarrow$ điền **SkinType(3) = Fair**. Sau đó train lại RF và lặp 1–2 vòng; nếu thay đổi nhỏ $\Rightarrow$ hội tụ.

Khi nào dùng phương án nào? (lộ trình nâng dần)

1. **Baseline nhanh + ổn**: số → median (+ cờ thiếu), phân loại → most_frequent (+ cờ thiếu); train RF, xem OOB/validation.
2. **Còn dư địa cải thiện**: thử Iterative (MICE) hoặc MissForest.
3. **Phi tuyến/ nhiều tương tác**: cân nhắc Proximity-weighted (chuẩn hoá theo hàng, diag = 0).
4. **Imbalanced/High-cardinality**: dùng `class_weight` hoặc resampling; với categorical nhiều mức, cân nhắc target encoding (làm *out-of-fold*).

Đánh giá "đúng chuẩn"

- **Hold-out masking**: che một phần giá trị đã biết, impute, đo RMSE/MAE (số) hoặc Accuracy/F1 (phân loại) *chỉ trên phần che.*
- **Ưu tiên metric mô hình**: AUC/F1/RMSE/MAE và *độ ổn định* qua CV/OOB (không chỉ mỗi RMSE imputation).
- **Phân tích nhạy cảm**: so sánh median/mode ↔ Iterative ↔ MissForest ↔ Proximity; chọn phương án *ổn định* giữa các lần chạy.

Lưu ý kỹ thuật & tối ưu chi phí

- RF không cần scale; nhưng KNN Imputer cần scale để khoảng cách có ý nghĩa.
- MDI importance có thể thiên lệch với one-hot dày/ nhiều mức ⇒ dùng Permutation importance hoặc SHAP để kiểm định.
- Dữ liệu lớn: giới hạn `max_depth`/`n_estimators`, dùng `max_samples<1.0` trong vòng impute để giảm thời gian; tận dụng song song (`n_jobs=-1`).
- Với time series, mọi impute/biến đổi phải tôn trọng *trật tự thời gian* (không nhìn tương lai) — chi tiết ở Phần 4.

0.1 Pseudocode nhanh cho Proximity-weighted (chuẩn hoá theo hàng)

```
# Input: X (co o thieu), y (tuy bai toan)
# 0) X0 = simple_impute(X); add missing_indicators
# Loop until convergence:
#  1) fit RF_temp on (Xk, y)   # RF nhe: it cay, max_depth vua phai
#  2) P = proximity_from_forest(RF_temp, Xk)
#     - set P[ii] = 0
#     - row-normalize: for each i, sum_{j!=i} P[i,j] = 1
#  3) for each column m:
#       if numeric:
#         for i missing at m:
#           Xk+1[i,m] = sum_{j observed m} P[i,j] * Xk[j,m]
#       if categorical:
#         for i missing at m:
#           choose class c maximizing sum_{j observed m & y_j=c} P[i,j]
#  4) check convergence (NRMSE / masked error). If small -> stop
# Return X_imputed = Xk*, then fit final RF on X_imputed
```

Các ví dụ bổ sung

Ví dụ 1 — Tự xây Proximity (B=2) rồi impute & dự đoán nhãn

Dữ liệu (5 mẫu): cột số `Chol` thiếu ở ID5, nhãn nhị phân `Outcome`.

ID	Age	BMI	Sunscreen	Outcome	Chol
1	25	18.0	No	Yes	160
2	27	18.8	No	Yes	165
3	40	26.0	Yes	No	210
4	38	25.2	Yes	No	200
5	28	21.0	No	?	NaN

Xây 2 cây minh hoạ.

- Tree 1: tách theo `Sunscreen` ⇒ Leaf A: {1,2,5}, Leaf B: {3,4}.
- Tree 2: tách theo `BMI` (ngưỡng 22) ⇒ Leaf C: {1,2,5}, Leaf D: {3,4}.

Đếm "đồng lá" trên 2 cây: (1,2)=2, (1,5)=2, (2,5)=2; (3,4)=2; cặp khác = 0.

Chuẩn hoá theo hàng (diag=0).

- Hàng 1: tổng thô = 4 ⇒ $P_{1,2} = 0.5$, $P_{1,5} = 0.5$.
- Hàng 2: $P_{2,1} = 0.5$, $P_{2,5} = 0.5$.
- Hàng 3: $P_{3,4} = 1.0$; Hàng 4: $P_{4,3} = 1.0$.
- Hàng 5: $P_{5,1} = 0.5$, $P_{5,2} = 0.5$.

Impute `Chol` cho ID5.

$$\widehat{\text{Chol}}_5 = 0.5 \cdot 160 + 0.5 \cdot 165 = \mathbf{162.5}.$$

Dự đoán `Outcome` cho ID5 (có điều chỉnh tần suất lớp).

- Proximity theo lớp: hàng xóm của 5 là 1 (Yes, 0.5), 2 (Yes, 0.5) ⇒ $W_{\text{Yes}} = 1.0$, $W_{\text{No}} = 0$.
- Tần suất lớp (4 mẫu có nhãn): $f_{\text{Yes}} = 0.5$, $f_{\text{No}} = 0.5$.
- Điểm lớp: $S_{\text{Yes}} = 1.0 \cdot 0.5 = 0.5$, $S_{\text{No}} = 0 \cdot 0.5 = 0$.

⇒ Dự đoán **Outcome(5) = Yes**.

Ví dụ 2 — Vòng lặp hội tụ (2 vòng) với Proximity-weighted

Thiết lập. Dữ liệu 6 mẫu, `BMI` thiếu ở ID2 & ID5; `SkinType` thiếu ở ID3. Impute sơ bộ: *median* cho BMI (21.75), *most_frequent* cho SkinType (Medium).

Vòng 1. Train RF tạm (nhẹ) → Proximity (chỉ hiển thị phần tử lớn):

[leftmargin=1.2em]

- Hàng 2: $P_{2,1} = 0.60$, $P_{2,3} = 0.35$, $P_{2,6} = 0.05$.
- Hàng 5: $P_{5,1} = 0.40$, $P_{5,3} = 0.15$, $P_{5,4} = 0.20$, $P_{5,6} = 0.25$.
- Hàng 3: $P_{3,1} = 0.30$, $P_{3,2} = 0.20$, $P_{3,4} = 0.10$, $P_{3,5} = 0.05$, $P_{3,6} = 0.05$.

Cập nhật:

$\widehat{\text{BMI}}_2^{(1)} = 18.525$, $\widehat{\text{BMI}}_5^{(1)} = 21.425$, SkinType(3) = Fair (với tần suất lớp Fair/Medium/Dark = 0.4/0.4/0.2).

Vòng 2. Thay giá trị mới → train lại RF tạm → Proximity thay đổi nhẹ:

[leftmargin=1.2em]

- Hàng 2: $P_{2,1} = 0.55$, $P_{2,3} = 0.40$, $P_{2,6} = 0.05$.
- Hàng 5: $P_{5,1} = 0.38$, $P_{5,3} = 0.12$, $P_{5,4} = 0.22$, $P_{5,6} = 0.28$.

Cập nhật:

$$\widehat{\text{BMI}}_2^{(2)} = 18.575 \quad (\Delta = 0.05), \qquad \widehat{\text{BMI}}_5^{(2)} = 21.41 \quad (\Delta = 0.015).$$

Kết luận: thay đổi rất nhỏ ⇒ hội tụ sau 2 vòng; train RF cuối trên dữ liệu đã điền.

Ví dụ 3 — So sánh MissForest vs Proximity cho cùng một cột số

Thiết lập. Cùng dữ liệu như Ví dụ 2 (thiếu BMI ở ID2 & ID5).

[leftmargin=1.2em]

- **MissForest (lượt 1)**: RF Regressor dự đoán BMI từ các biến khác ⇒ giả sử $\widehat{\text{BMI}}_2 = 18.6$, $\widehat{\text{BMI}}_5 = 21.3$.
- **Proximity (lượt 1)**: như Ví dụ 2 ⇒ $\widehat{\text{BMI}}_2 = 18.525$, $\widehat{\text{BMI}}_5 = 21.425$.

Nhận xét: nếu quan hệ mang tính *cục bộ theo cấu trúc rừng* ⇒ Proximity thường mượt/ăn khớp; nếu phụ thuộc *toàn cục* rõ rệt ⇒ MissForest có thể sát hơn. Kết hợp: chạy MissForest, rồi *tinh chỉnh* thêm 1 vòng Proximity-weighted để bắt cấu trúc cục bộ.

0.2 Tóm tắt "mang đi thi"

[leftmargin=1.2em]

- **Proximity Matrix** dùng ở đây: *chuẩn hoá theo hàng, đường chéo* $= 0$; mỗi hàng là *trọng số lân cận* tổng bằng 1.
- **Dự đoán nhãn**: proximity theo lớp × *tần suất lớp* ⇒ lớp có điểm cao nhất.
- **Điền số**: *trung bình có trọng số proximity* (đã chuẩn hoá).
- **MissForest**: RF-based imputation theo cột, lặp đến hội tụ; có thể *tinh chỉnh* bằng proximity sau mỗi vòng.
- Bắt đầu *đơn giản + an toàn*, rồi *nâng cấp* nếu còn dư địa; đánh giá bằng *metric mô hình + độ ổn định* (CV/OOB) và luôn giữ nguyên tắc *không rò rỉ*.

IV. Bài toán Time Series với Random Forest & Ensemble

Tổng quan

Time series (TS) có *tự tương quan*, *xu hướng* và *mùa vụ*, rất dễ gặp *rò rỉ tương lai* nếu xử lý sai. Random Forest (RF) không "biết thời gian" sẵn, do đó cần *chuyển chuỗi → bảng đặc trưng* (supervised framing), rồi áp dụng RF hồi quy/phân loại. RF hữu ích nhờ học *phi tuyến & tương tác* (giữa lags, lịch, ngoại sinh), *robust* với nhiễu và ít yêu cầu chuẩn hoá.

Quy tắc vàng "không rò rỉ"

1. **Chia theo thời gian**: Train < Validation < Test theo mốc tuyệt đối; *không shuffle*.
2. **Đặc trưng quá khứ**: mọi feature tại thời t chỉ dùng thông tin $\leq t$. Tất cả rolling phải `.shift(1)`.
3. **Tiền xử lý**: impute/scale/encoding phải *fit* trên quá khứ, *transform* cho hiện tại (mỗi fold walk-forward).
4. **Đánh giá**: dùng *rolling-origin (walk-forward)* hoặc *Blocked TimeSeriesSplit*. **Không dùng OOB** của RF cho TS.

Chuyển chuỗi → bảng (supervised framing)

Giả sử chuỗi mục tiêu y_t và ngoại sinh x_t.

- **Lags**: $y_{t-1}, \ldots, y_{t-k}$; lags theo mùa: $y_{t-7}, y_{t-14}, y_{t-28}$ (dữ liệu ngày).
- **Rolling/Expanding**: $\overline{y}_{t-w:t-1}$, $\mathrm{sd}(y)_{t-w:t-1}$, min/max, quantiles (`shift(1)` trước khi tính).
- **Calendar**: day-of-week, day-of-month, week-of-year, end-of-month, flags lễ/tết, khoảng cách đến/từ ngày lễ.
- **Fourier (mùa vụ)**: $\sin\frac{2\pi kt}{P}$, $\cos\frac{2\pi kt}{P}$ cho chu kỳ P (tuần=7, năm=365), $k = 1..K$.
- **Ngoại sinh trễ**: nếu x_t *được biết trước* thì dùng trực tiếp; nếu không, dùng $x_{t-1}, x_{t-2}, \ldots$ để tránh leak.

Ví dụ: $\mathrm{roll\_mean7}(t) = \frac{1}{7}\sum_{i=1}^{7} y_{t-i}$; $\mathrm{lag7}(t) = y_{t-7}$.

Ví dụ 1 (tính tay) — Xây đặc trưng & dự báo 1 bước

Dữ liệu *doanh số ngày* (14 ngày); mục tiêu dự báo *ngày 15*. Dùng lags $[1, 7]$ và rolling mean 7 ngày.

Bảng 2: Lịch sử ngắn và đặc trưng cho dự báo +1

Ngày	y_t	y_{t-1}	y_{t-7}	$\mathrm{mean}_{t-7:t-1}$
8	22	21	16	18.9
9	23	22	18	19.6
10	20	23	19	20.0
11	24	20	21	20.7
12	25	24	22	21.7
13	23	25	23	22.6
14	26	23	24	22.7

Tại thời điểm **dự báo ngày 15**, đặc trưng hợp lệ: $y_{14} = 26$, $y_8 = 22$, $\mathrm{mean7}_{8:14} = 23.3$. Đưa vector này vào RF (đã huấn luyện trên lịch sử dài hơn) để thu $\hat{y}_{15}$.

Hồi quy nhiều bước (multi-step forecasting)

Gọi H là horizon (ví dụ $H = 7$ ngày).

- **Recursive**: 1 model cho +1; để dự báo xa hơn dùng chính dự báo trước làm lag. Ưu: ít model; Nhược: lỗi tích luỹ.
- **Direct**: 1 model cho mỗi $+h$ ($h = 1..H$). Ưu: ổn định; Nhược: tốn tài nguyên.
- **DirRec**: kết hợp Direct + Recursive (mỗi horizon dùng thêm các dự báo gần làm đặc trưng).
- **Multi-output**: 1 model dự báo vector $(y_{t+1}, \ldots, y_{t+H})$.

Khuyến nghị: bắt đầu với Direct (ổn định) hoặc Multi-output nếu thư viện hỗ trợ và dữ liệu đủ.

Walk-forward backtesting (rolling-origin)

Ví dụ *expanding window* theo tuần:

Fold 1	Train tuần 1–4 → Val tuần 5
Fold 2	Train tuần 1–5 → Val tuần 6
Fold 3	Train tuần 1–6 → Val tuần 7
Fold 4	Train tuần 1–7 → Val tuần 8
Fold 5	Train tuần 1–8 → Val tuần 9

Tổng hợp MAE/RMSE theo fold, kiểm tra ổn định. Với *concept drift*, dùng *sliding window* (giữ cửa sổ gần nhất độ dài W).

Ví dụ 2 — Walk-forward cho daily & Direct $H = 3$

Mục tiêu: dự báo $(y_{t+1}, y_{t+2}, y_{t+3})$ theo **Direct** (3 model). Đặc trưng: y_{t-1}, y_{t-7}, mean7, day-of-week, $\texttt{promo}_t$ (biết trước).

- **Fold A**: Train ngày 1–28 → Val ngày 29–31.
- **Fold B**: Train ngày 1–31 → Val ngày 32–34.
- **Fold C**: Train ngày 1–34 → Val ngày 35–37.

Mỗi fold: sinh đặc trưng hợp lệ, train 3 RF (M_{+1}, M_{+2}, M_{+3}), dự báo cho cửa sổ val, tính $\text{RMSE}_{+1}, \text{RMSE}_{+2}, \text{RMSE}$ và lấy trung bình theo fold.

Missing data trong TS (gắn Phần 3)

- **Forward-fill có kiểm soát**: giới hạn tối đa k bước; nếu vượt k chuyển *seasonal carry* (y_{t-7}) hoặc *linear interp* trong đoạn.
- **Rolling impute**: dùng mean/median của quá khứ gần nhất (`shift(1)`).
- **Proximity/MissForest cho TS**: chỉ xây trên *cửa sổ quá khứ* (không dùng quan sát tương lai). Với chuỗi đơn biến, rolling/seasonal impute thường hiệu quả và đơn giản hơn; Proximity/MissForest phát huy khi có nhiều ngoại sinh phi tuyến.

Ví dụ nhỏ (thiếu liên tiếp).

```
Ngày:     1  2  3  4  5  6  7  8
y:       12 14 -- -- 16 18 15 17
```

FF giới hạn k=1 ⇒ ngày 3 = 14; ngày 4 (vượt k) dùng nội suy tuyến tính giữa 14 và 16 ⇒ 15. Sau điền: 12,14,14,15,16,18,15,17 rồi mới sinh lags/rolling.

Phân loại sự kiện theo thời gian (event classification)

Bài toán: dự đoán khả năng "cháy hàng" ngày mai. Nhãn: $z_{t+1} = \mathbf{1}\{y_{t+1} = 0\}$.

- **Đặc trưng**: y_{t-1}, y_{t-7}; rolling-min 7 ngày; % ngày bán 0 trong 14 ngày; tồn kho inv_t; dow, gần lễ; $\texttt{promo}_t$, $\texttt{price}_t$.
- **Đánh giá**: walk-forward; xử lý mất cân bằng bằng `class_weight="balanced"`; ưu tiên PR-AUC/F1.
- **Diễn giải**: permutation importance; calibrate xác suất (Isotonic/Platt) nếu cần.

Panel time series (đa chuỗi) — Global RF

Nhiều chuỗi (cửa hàng A/B/C...).

- **Local**: mỗi chuỗi 1 model → dễ thiếu dữ liệu & overfit.
- **Global (khuyến nghị)**: 1 model chung, thêm *ID/nhóm* (one-hot hoặc target stats) để chia sẻ mẫu hình.

Gợi ý feature: lags/rolling của y, calendar, ngoại sinh địa phương (giá/tồn kho), ID one-hot, *thống kê dài hạn theo chuỗi* (mean/volatility). RF học tương tác giữa ID và lags/ngoại sinh, dự báo riêng cho từng chuỗi trong một mô hình chung.

Mùa vụ & xu hướng (trend/seasonality)

[leftmargin=1.2em]

- **Trend**: thêm time-index, polynomial/spline; hoặc tách trend bằng mô hình tuyến tính & để RF học residual phi tuyến.
- **Seasonality**: Fourier (7/365), hoặc rolling "đúng pha" (mean của $t-7, t-14, t-21, \ldots$).
- **Sự kiện**: flags holiday $\pm n$ ngày, end-of-month/quý, khoảng cách tới sự kiện.

Ví dụ 3 — Phân loại sự kiện (stockout)

Thiết lập. Nhãn z_{t+1}; đặc trưng tại t gồm lags của y, rolling-min7, tần suất 0 trong 14 ngày, inv_t, dow, holiday, promo_t, price_t.

Walk-forward. 4 folds theo tuần; metric PR-AUC/F1.

Kết quả kỳ vọng. Tầm quan trọng thường: $\text{inv}_t \succ$ rolling-min7 $\succ \text{promo}_t \succ$ dow. Calibration giúp xác suất trung thực hơn.

Ví dụ 4 (workflow đầy đủ) — Dự báo 7 ngày tới

Bối cảnh. Dự báo doanh số ngày 7 ngày tới; có `price`, `promo`, `temp`.

1. **Làm sạch & điền thiếu**: bổ sung mốc thiếu; FF `promo` (nếu biết lịch), nội suy `temp` (không nhìn tương lai).
2. **Sinh đặc trưng tại** t: $y_{t-1}, y_{t-7}, y_{t-14}, y_{t-28}$; mean/std 7/14/28 (đều `shift(1)`); dow, dom, weekofyear, eom, holiday; Fourier (7: $k = 1..2$, 365: $k = 1$); $price_t, promo_t, temp_t$ (nếu biết trước).
3. **Nhãn multi-horizon (Direct)**: tạo $y_{t+1}, ..., y_{t+7}$ (`shift(-h)`), rồi *drop* các dòng thiếu nhãn.

4. **Walk-forward**: 5 folds (mỗi fold 28 ngày val). Train 7 RF (mỗi horizon 1 model) *hoặc* 1 RF multi-output (vector 7 chiều).

5. **Chọn tham số & đánh giá**: `n_estimators` 600–1000, `max_depth` 14–20, `max_features` 0.4–0.7, `min_samples_leaf` 10–50. Báo RMSE/sMAPE theo từng horizon; kiểm tra *độ ổn định* theo fold.

6. **Triển khai**: hằng ngày sinh feature tại $t \rightarrow$ dự báo $\hat{y}_{t+1..t+7}$. Đặt *ngưỡng drift* (ví dụ sMAPE trên cửa sổ gần nhất) để kích hoạt retrain với cửa sổ 180–365 ngày.

Snippet tham khảo (Python/scikit-learn)

```
#  G i    s    df  c  : date, y, price, promo, temp (daily)
df = df.sort_values("date").reset_index(drop=True)

# 1) Lags & rolling an to n (kh ng nh n  t   ng lai)
for lag in [1,7,14,28]:
    df[f"y_lag{lag}"] = df["y"].shift(lag)
for w in [7,14,28]:
    df[f"y_mean_{w}"] = df["y"].shift(1).rolling(w).mean()
    df[f"y_std_{w}"]  = df["y"].shift(1).rolling(w).std()

# Calendar
d = pd.to_datetime(df["date"])
df["dow"] = d.dt.weekday
df["dom"] = d.dt.day
df["eom"] = d.dt.is_month_end.astype(int)

# Fourier weekly
t = np.arange(len(df)); w = 7
for k in [1,2]:
    df[f"sin{k}_w"] = np.sin(2*np.pi*k*t/w)
    df[f"cos{k}_w"] = np.cos(2*np.pi*k*t/w)

# 2) Multi-horizon labels (Direct)
H = 7
for h in range(1, H+1):
    df[f"y_tplus{h}"] = df["y"].shift(-h)

# 3) Drop rows with NaN from initial lags & labels
dfm = df.dropna().copy()

# 4) Walk-forward (TimeSeriesSplit) -> train 7 RF (one per horizon)
#  h o c  1 RF multi-output (predicts 7 targets at once)
```

Hạn chế & kết hợp mô hình

RF không ngoại suy trend dài nếu thiếu feature phù hợp. Giải pháp:

- **Thêm trend features** (time-index, poly/spline), Fourier mùa vụ.
- **Hybrid**: mô hình tuyến tính/ARIMA bắt trend/mùa vụ "mượt", RF học residual phi tuyến (giá, khuyến mãi, thời tiết, tương tác).
- Với horizon dài hoặc nhiều ngoại sinh thứ tự, **GBDT** (LightGBM/CatBoost) thường nhỉnh hơn khi tuning kỹ; RF vẫn là baseline mạnh/ổn định.

Checklist chẩn đoán nhanh

- **Dự báo phẳng** ⇒ thêm trend features; cân nhắc hybrid Linear/ARIMA + RF residual.
- **Sai số tăng theo horizon** ⇒ Direct/Multi-output; thêm Fourier/rolling dài; tăng dữ liệu/ngoại sinh.
- **Nghi rò rỉ** ⇒ kiểm mọi `.shift(1)`, chia thời gian, impute/scale theo quá khứ.
- **Concept drift** ⇒ sliding window, retrain thường xuyên, thêm regime flags.
- **Quá nhiều feature** ⇒ tăng `min_samples_leaf`, giảm `max_features`, sàng lọc bằng permutation importance.