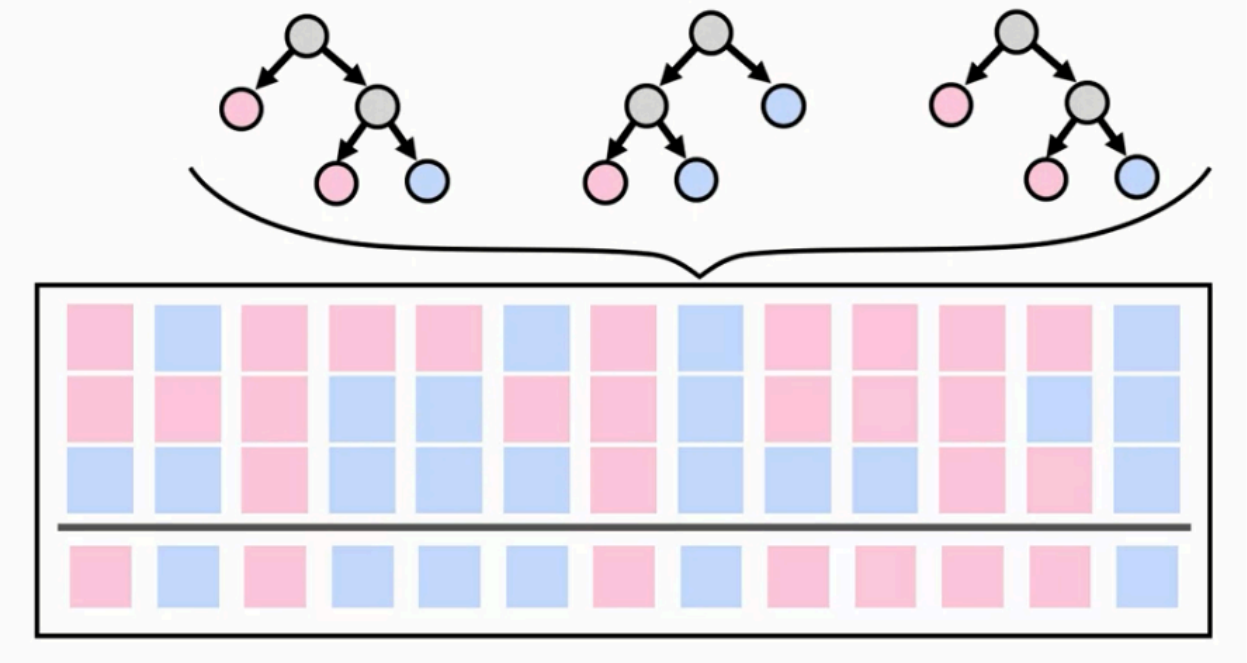
Course3_Module5

Ensemble Learning — Bagging, Random Forest, Extra Trees, Boosting & Stacking

1) Tư duy chung về Ensemble

- **Mục tiêu**: kết hợp nhiều mô hình "yếu" ➜ giảm **phương sai** (variance) và/hoặc **thiên lệch** (bias) để tổng thể **khái quát hóa tốt hơn**.
- Hai cách gộp phổ biến:
  - **Averaging** (hồi quy): trung bình dự đoán.
  - **Voting** (phân loại): đa số phiếu.



---

2) Bagging (Bootstrap Aggregating)

2.1 Ý tưởng cốt lõi

- Tạo **B** bản sao dữ liệu bằng **bootstrap** (lấy mẫu có hoàn lại, cùng kích thước với tập gốc).
- Huấn luyện **B** mô hình độc lập (thường là cây quyết định không tỉa).
- **Gộp**: trung bình (regression) hoặc bỏ phiếu (classification).
- Tác dụng chính: **giảm phương sai** nhờ trung bình nhiều dự đoán độc lập.

2.2 Thuật toán (step-by-step)

1. Với $b = 1 \dots B$: sinh mẫu bootstrap $S^{(b)}$ từ tập huấn luyện S.
2. Huấn luyện mô hình $f^{(b)} = \text{base\_learner}(S^{(b)})$.
3. Dự đoán:

$$\text{D}$$

2.3 Out-of-Bag (OOB)

- Mỗi mô hình không “thấy” ~36.8% điểm (không được chọn vào bootstrap).
- Dùng những điểm **OOB** đó để đánh giá lỗi/tỉ lệ đúng **mà không cần** tập validation riêng.

2.4 Khi tăng số cây B

- Sai số thường **giảm dần và hội tụ**; thêm cây vượt điểm bão hòa **không làm xấu đi** (nhưng tốn thời gian).

---

3) Random Forest (RF)

3.1 Khác gì so với Bagging?

- Ngoài bootstrap trên **mẫu**, RF còn **ngẫu nhiên hoá đặc trưng tại mỗi nút**: chỉ xét m_{try} đặc trưng ngẫu nhiên khi tìm split tốt nhất.
- Mục tiêu: **giảm tương quan** giữa các cây ➜ giảm phương sai sau khi gộp.

3.2 Cấu hình thực hành (sklearn)

- `bootstrap=True` (mặc định); cỡ mẫu phụ do `max_samples`.
- Mỗi split chọn trong **tập con đặc trưng** (vd. $\sqrt{p}$ cho phân loại).
- `n_estimators` đủ lớn đến khi lỗi ổn định.

---

4) Extra Trees (Extremely Randomized Trees)

4.1 Ý tưởng

- **Không** nhất thiết dùng bootstrap (mặc định dùng **toàn bộ** mẫu).
- **Tăng ngẫu nhiên** thêm 1 bước: với mỗi đặc trưng được chọn, **lấy ngưỡng split ngẫu nhiên**, rồi chọn split tốt nhất trong các ngưỡng ngẫu nhiên đó.
- Hệ quả: **nhanh hơn**, giảm tương quan giữa cây, có thể đổi trade-off bias/variance.

4.2 Thực hành (sklearn)

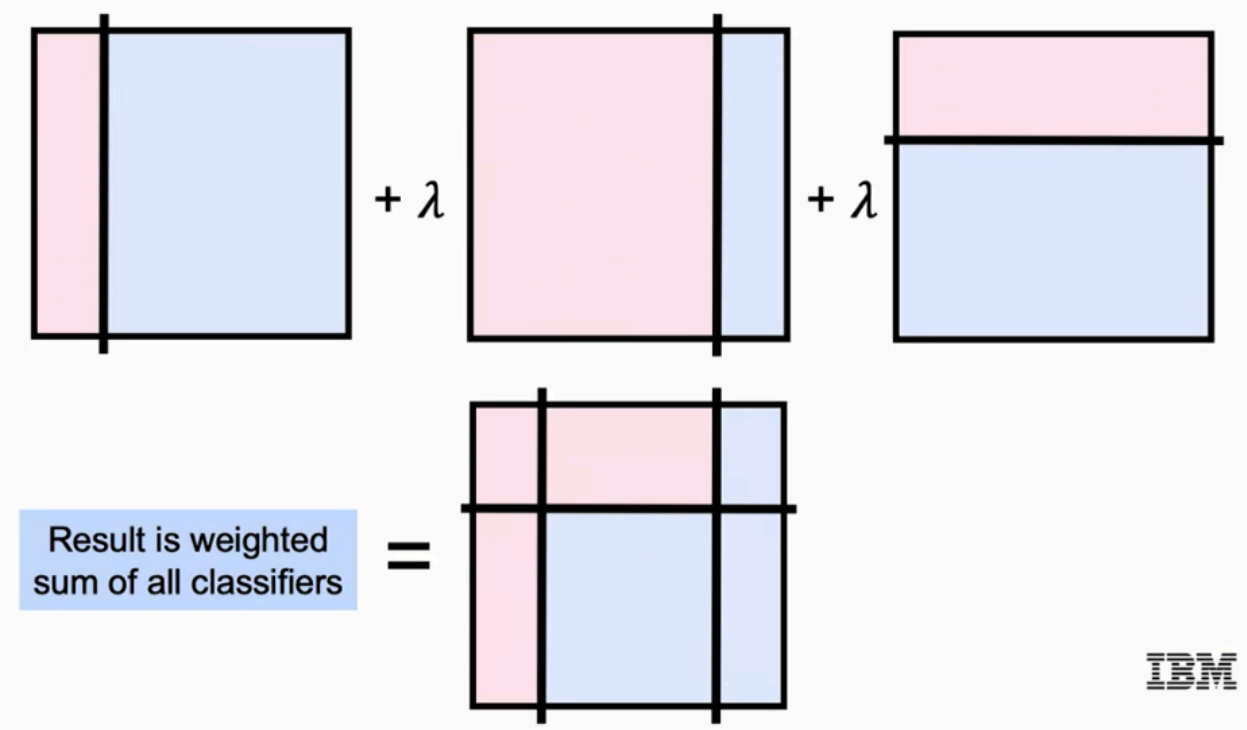
- `ExtraTreesClassifier` tương tự RF nhưng `splitter` ngẫu nhiên; có tuỳ chọn `bootstrap`.

---

5) Boosting (AdaBoost, Gradient Boosting)

5.1 Khác biệt với Bagging

- **Bagging**: huấn luyện **song song**, mô hình **độc lập**.
- **Boosting**: huấn luyện **tuần tự**, mỗi mô hình **tập trung** vào điểm **khó** của mô hình trước (trọng số/“residual” lớn).



5.2 AdaBoost (với base learner là decision stump)

Thuật toán (binary) — step-by-step

1. Khởi tạo trọng số mẫu đều nhau: $w_i^{(1)} = \frac{1}{N}$.
2. Lặp $t = 1 \dots T$:
   - Huấn luyện hth_tht để **min** lỗi có trọng số $\varepsilon_t = \sum_i w_i^{(t)} \, [y_i \neq h_t(x_i)]$.
   - Tính **hệ số** $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$.
   - Cập nhật **trọng số**:

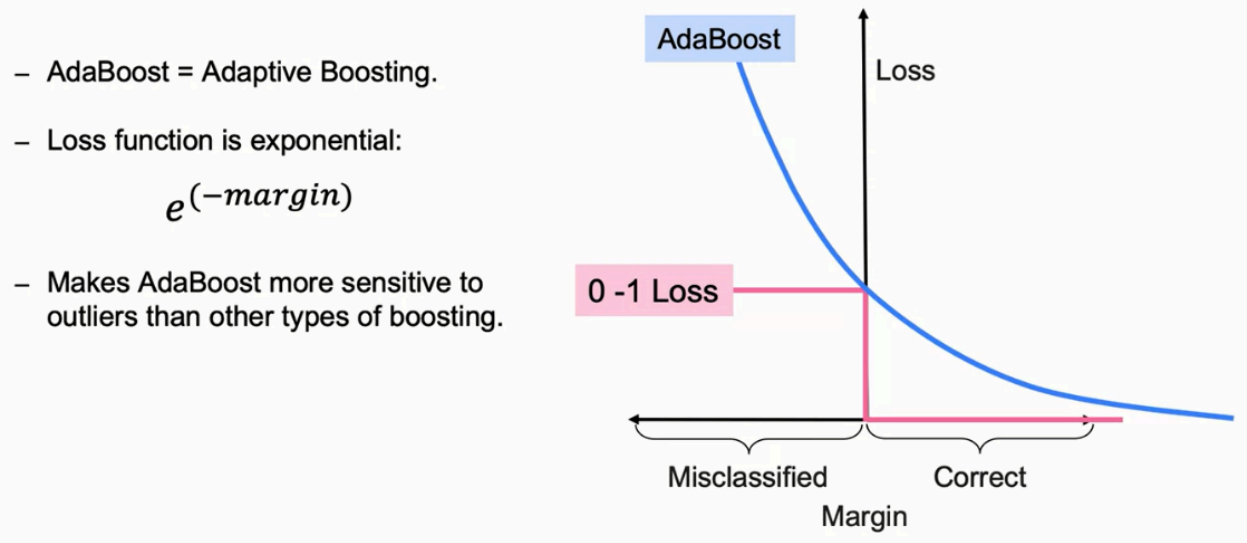
$$w_i^{(t+1)} \propto w_i^{(t)} \exp\big(-\alpha_t \, y_i \, h_t(x_i)\big)$$

   - Chuẩn hoá $w^{(t+1)}$ để tổng bằng 1.

1. Mô hình cuối:

$$F(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha_t \, h_t(x)\Big)$$

- AdaBoost gián tiếp **tối ưu hoá hàm mất mát mũ (exponential loss)** và liên hệ chặt với khái niệm **margin**.



5.3 Gradient Boosting (GBM)

Ý tưởng: xây mô hình **cộng dồn** $F_M(x) = \sum_{m=1}^{M} \nu \, \gamma_m \, h_m(x)$, trong đó mỗi hmh_mhm khớp **gradient âm của loss** trên phần dư.

Thuật toán (khung tổng quát, phân loại nhị phân với log-loss)

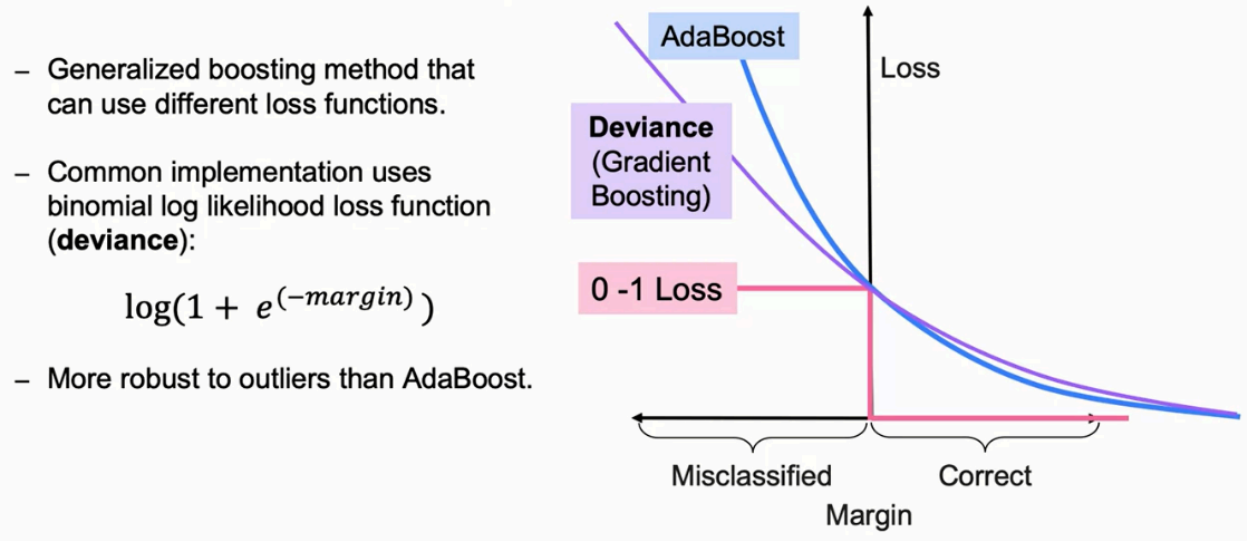
1. Khởi tạo $F_0(x) = \arg\min_c \sum_i \ell(y_i, c)$ (với log-loss là logit^{-1} hằng).
2. Với $m = 1 \dots M$:
   - Tính **pseudo-residual**: $r_i^{(m)} = -\big[\partial \ell(y_i, F(x_i))/\partial F\big]_{F=F_{m-1}}$
   - Khớp cây bé $h_m(x)$ vào $\{(x_i, r_i^{(m)})\}$.
   - Tìm γ_m (line search) rồi **cộng dồn**: $F_m(x) = F_{m-1}(x) + \nu \, \gamma_m \, h_m(x) (\nu = learning\ rate)$.

Loss thường dùng

- `log_loss` **(binomial deviance)** cho phân loại xác suất, **bền vững hơn** trước outliers so với exponential loss.
- Đặt `loss="exponential"` trong GBM sẽ **khôi phục AdaBoost**.

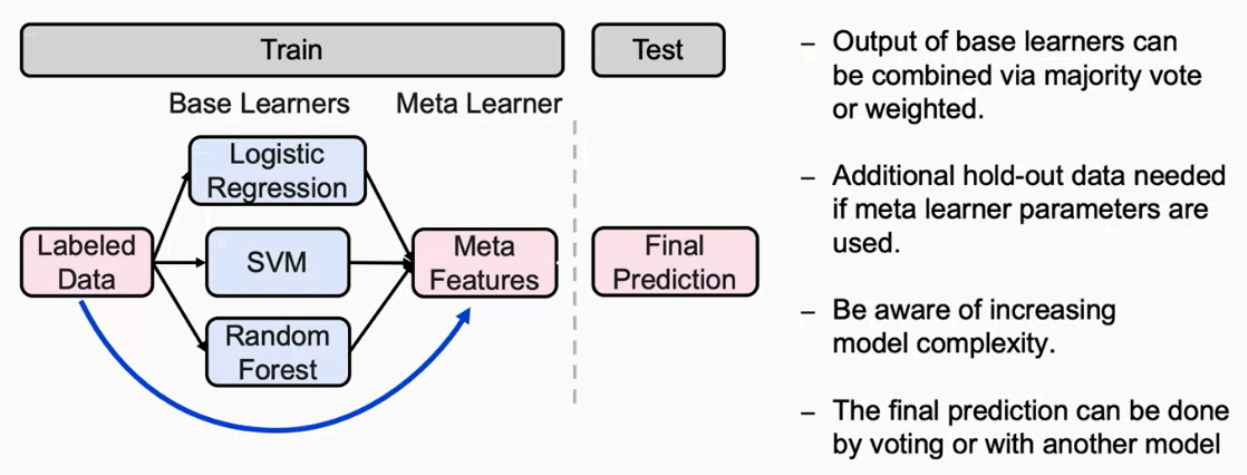
Mẹo tuning nhanh

- Giảm **learning_rate**, tăng **n_estimators** (đổi lại thời gian).
- Dùng **subsample < 1** (stochastic GBM) để giảm phương sai.



6) Stacking

- Huấn luyện **nhiều mô hình nền** (RF, SVM, LR, v.v.), lấy **predictions/probabilities** của chúng làm **đặc trưng đầu vào** cho một **meta-learner** (thường là logistic/linear).
- Kết hợp có thể là **majority vote** hoặc **trọng số** theo năng lực mô hình.



7) So sánh nhanh & khi nào dùng?

Phương pháp	Cách xây	Mục tiêu chính	Khi nên dùng
Bagging	Bootstrap + gộp song song	↓ Variance	Base learner "ồn" (như cây sâu), muốn OOB để ước lượng nhanh.
Random Forest	Bagging + chọn đặc trưng ngẫu nhiên ở mỗi split	↓ Variance (↓ tương quan)	Cân bằng giữa hiệu quả & dễ dùng; baseline mạnh.
Extra Trees	(Thường) không bootstrap + ngưỡng split ngẫu nhiên	↓ Tương quan, ↑ tốc độ	Dữ liệu lớn, cần cây rất ngẫu nhiên/nhanh.
AdaBoost	Tuần tự, tăng trọng số điểm khó	↓ Bias; nhạy outliers (exponential loss)	Dữ liệu sạch/vừa, muốn mô hình mảnh mai (stumps).
Gradient Boosting	Tuần tự, tối ưu theo gradient của loss	Bias-variance trade-off linh hoạt	Cần mô hình mạnh, có thời gian tuning; xem thêm HistGB.

8) Công thức/điểm nhấn "nhớ lâu"

- **Bagging giảm phương sai**: trung bình nhiều mô hình ít tương quan ➜ phương sai trung bình **giảm**; tương quan càng thấp hiệu quả càng cao (động lực của RF/ExtraTrees).

- **OOB error** ≈ lỗi hold-out nội bộ: tiện để chọn BBB, độ sâu cây con.
- **AdaBoost** tối ưu **exponential loss** và cải thiện **margin**; trọng số mẫu cập nhật theo$\exp(-\alpha_t y_i h_t(x_i))$.
- **GBM** với `loss="log_loss"` ~ deviance (giống logistic regression), **bền** hơn với ngoại lai; `loss="exponential"` ~ AdaBoost.

---

9) Mẫu cấu hình sklearn

```
# Bagging
from sklearn.ensemble import BaggingClassifier
model = BaggingClassifier(
    n_estimators=200, bootstrap=True, oob_score=True, n_jobs=-1, random_sta
te=42
)
```

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(
    n_estimators=400, max_features="sqrt", bootstrap=True, n_jobs=-1, rando
m_state=42
)
```

```
# Extra Trees
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier(
    n_estimators=400, max_features="sqrt", bootstrap=False, n_jobs=-1, rando
m_state=42
)
```

```
# Gradient Boosting (trees nhỏ, learning_rate thấp)
from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier(
```

```
    n_estimators=300, learning_rate=0.05, max_depth=3, subsample=0.8, rand
om_state=42
)
```

```
# AdaBoost (stumps)
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier(
    n_estimators=300, learning_rate=0.1, random_state=42
)
```