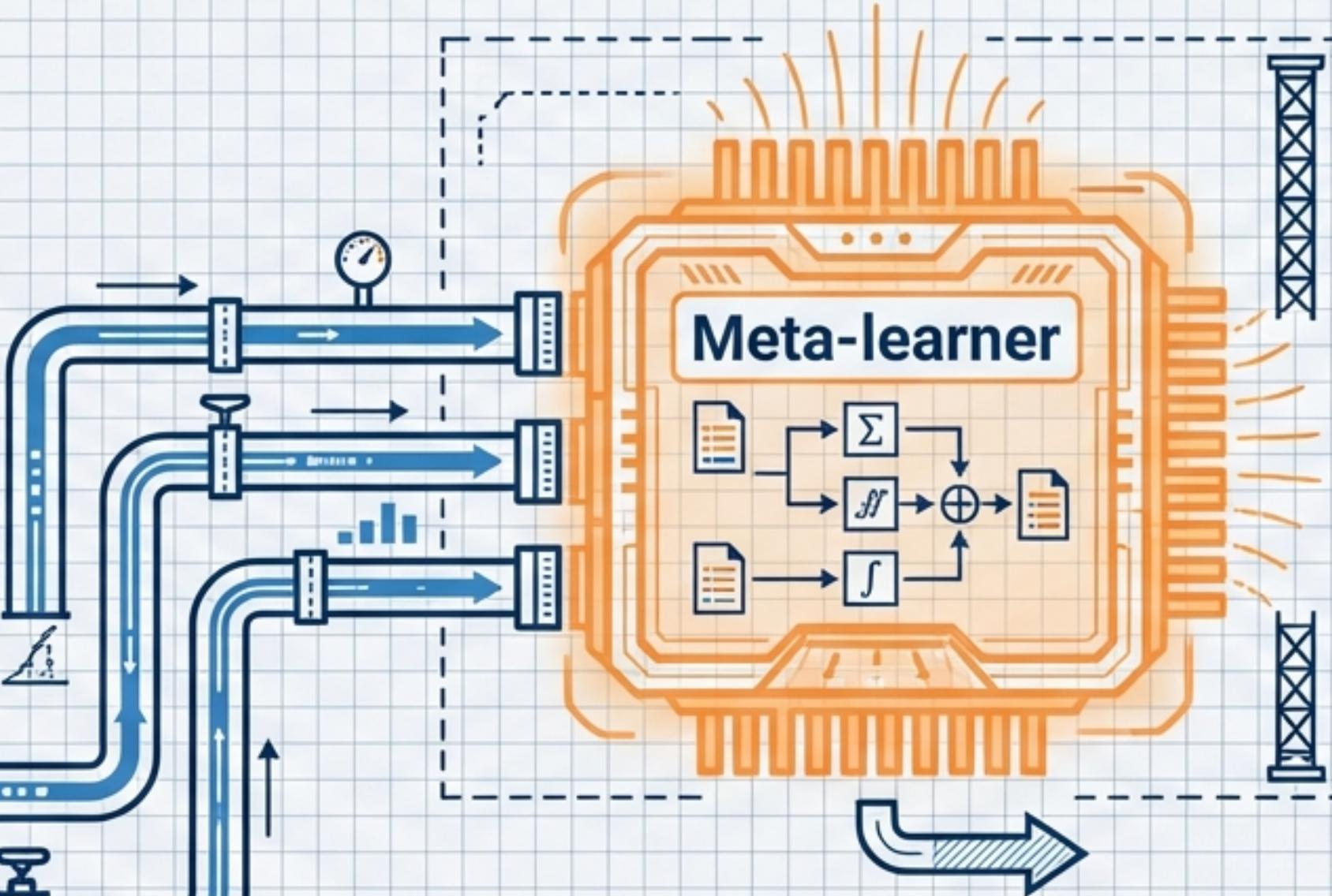
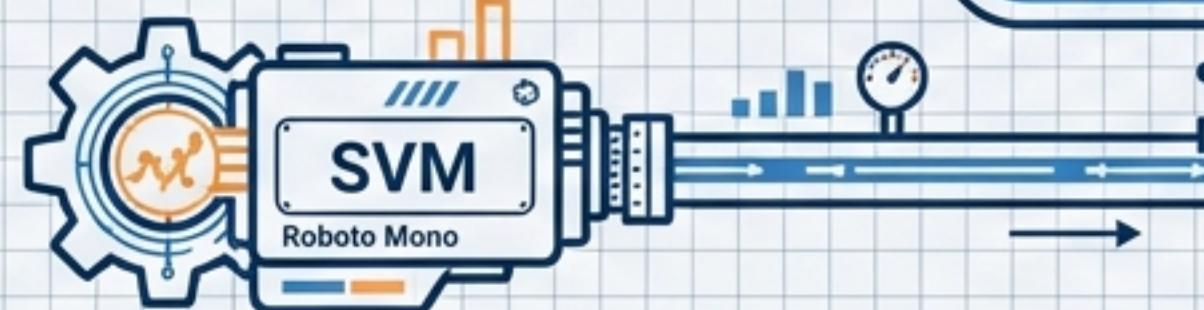
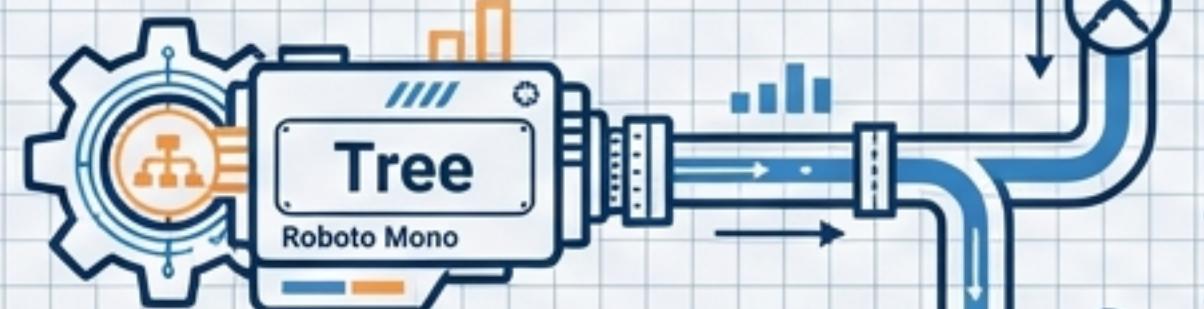


Unit 13: Stacking 堆疊法 — 進階集成學習與化工應用

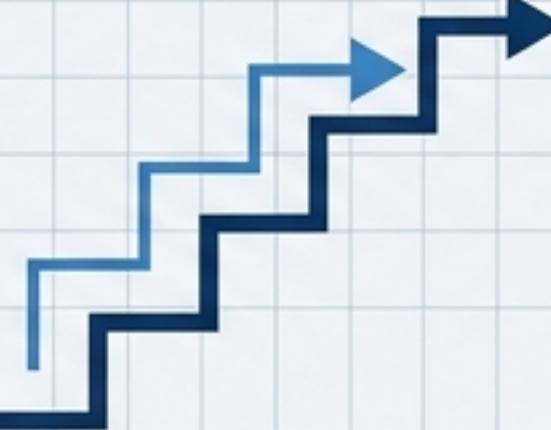
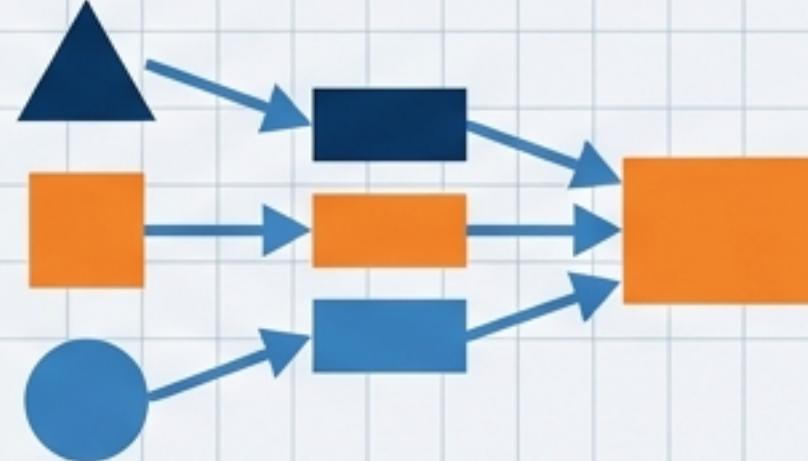
建構極致模型的系統工程學



利用模型的**多樣性**，打造超越
單一專家的**超級決策系統**。

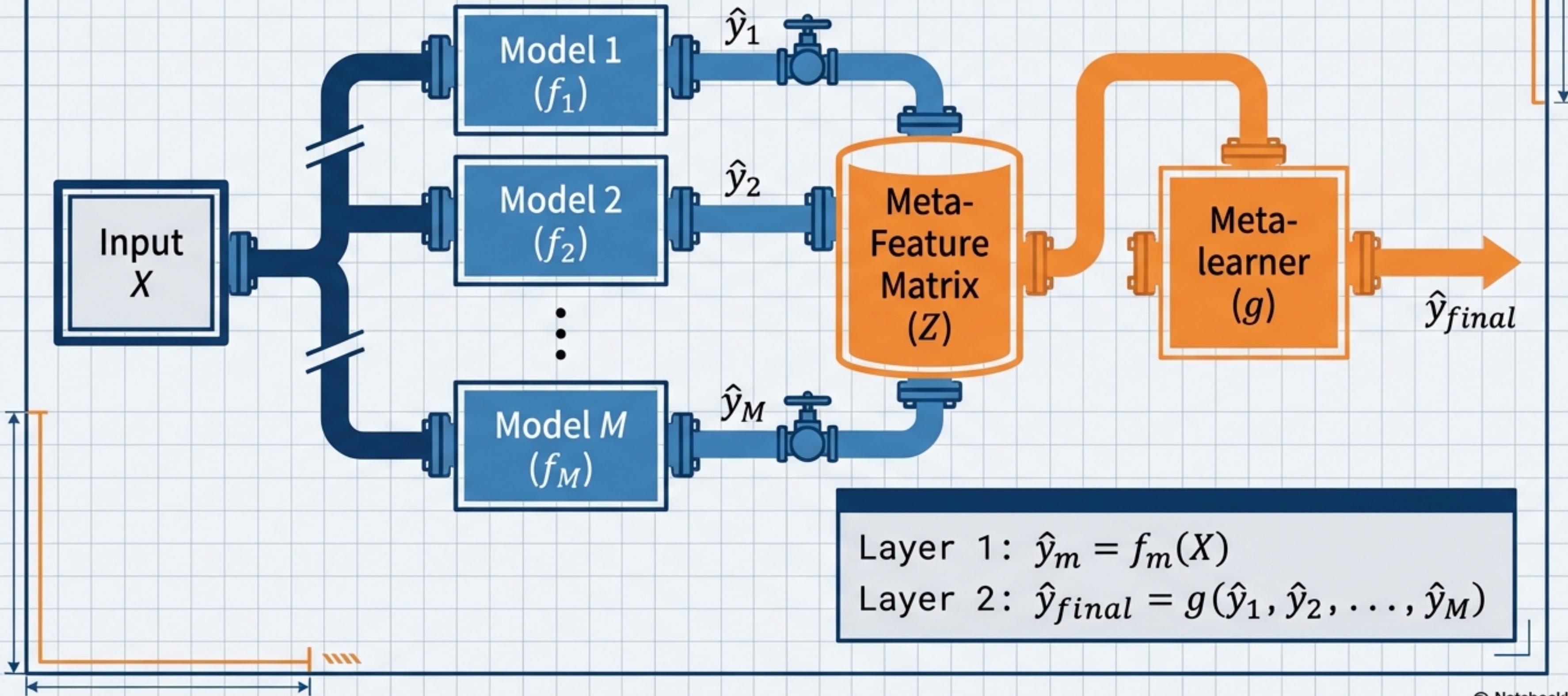
什麼是 Stacking？

一種異質性集成學習（Heterogeneous Ensemble Learning）。不同於 Bagging（並行/同質）或 Boosting（序列/同質），Stacking 結合了「不同類型」的演算法優勢。

Bagging (Random Forest)	Boosting (XGBoost)	Stacking
		
降低方差 (Variance)	降低偏差 (Bias)	提升整體性能 (Performance)
基礎模型：同質 (Homogeneous)	基礎模型：同質 (Homogeneous)	基礎模型：異質 (Heterogeneous)
組合：平均/投票	組合：加權迭代	組合：元學習器 (Meta-learner)

核心概念：第一層 (Base Layer) 的多個模型獨立預測，第二層 (Meta Layer) 學習如何最佳化組合這些預測結果。

系統架構與數學原理

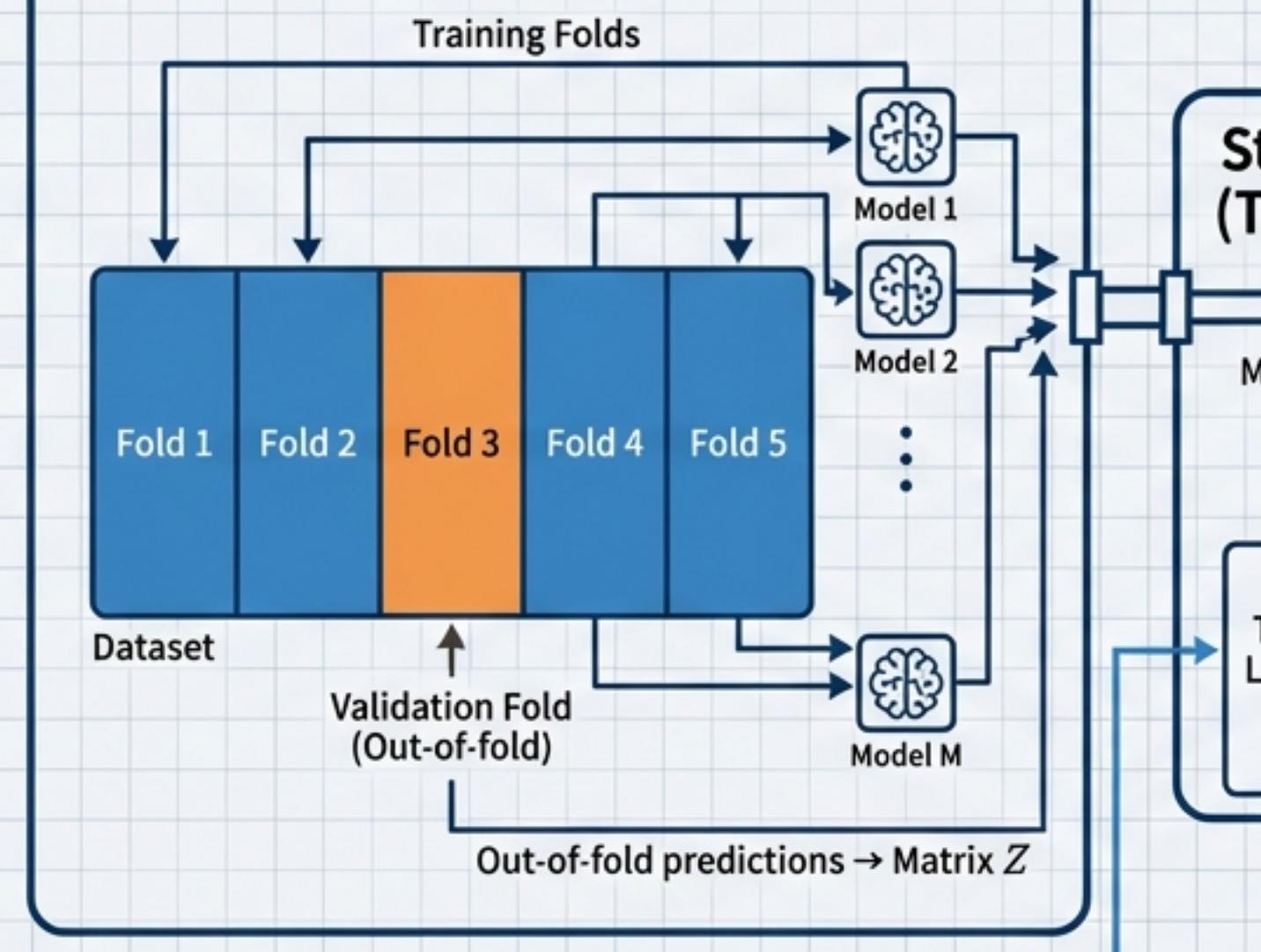


訓練流程與防漏機制

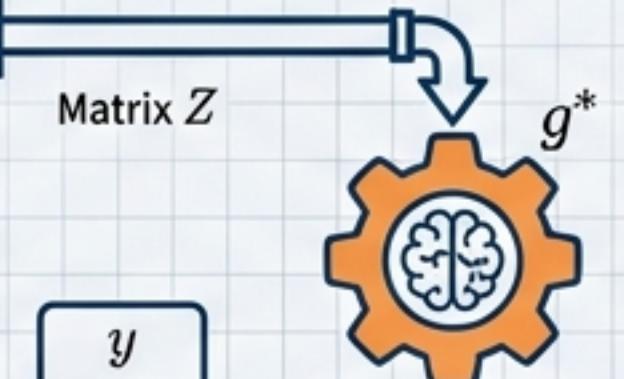


Data Leakage Warning (數據洩露警告):
直接使用訓練集預測會導致元學習器過擬合，必須使用 Cross-Validation。

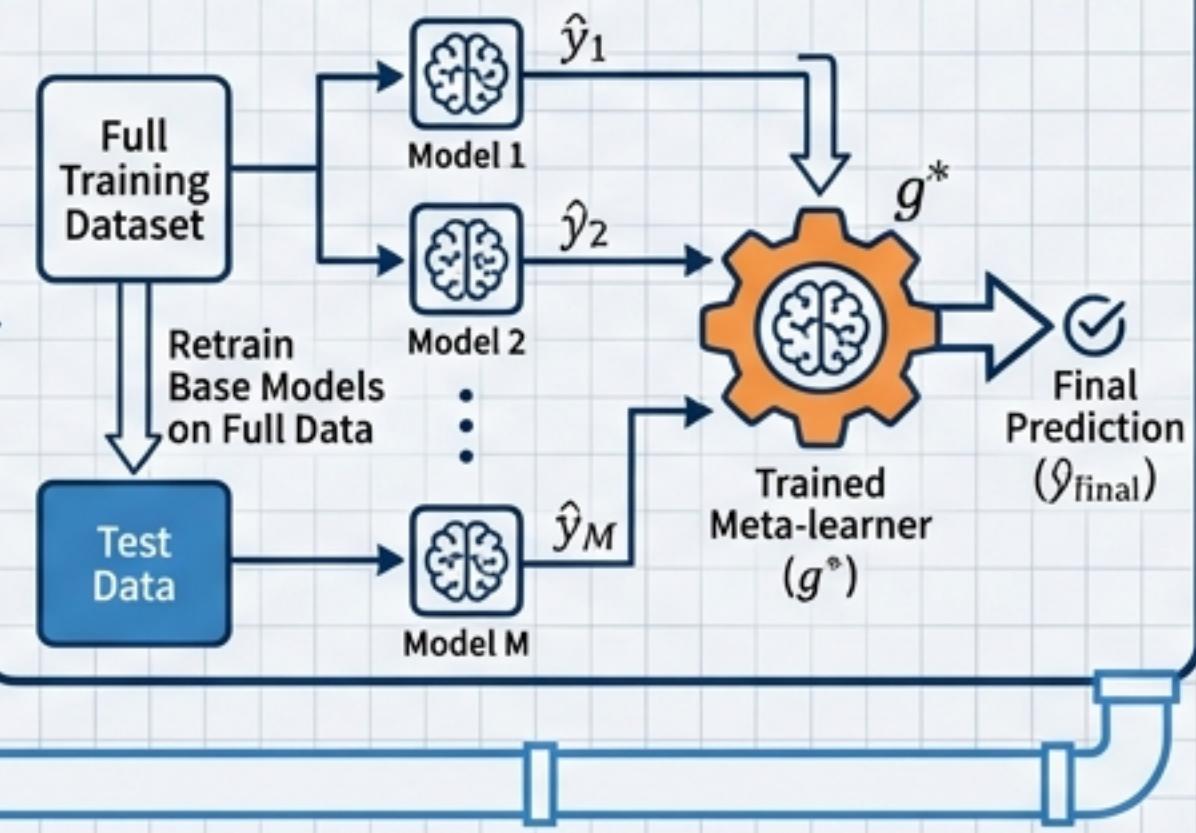
Step 1: 生成元特徵 (Meta-features)



Step 2: 訓練元學習器 (Train Meta-learner)



Step 3: 最終預測 (Final Prediction)



關鍵：確保元學習器的輸入數據是基礎模型「未見過」的，以保證泛化能力。

Python 實作：建立自動化控制單元

```
1 from sklearn.ensemble import StackingRegressor  
2 from sklearn.linear_model import Ridge  
3  
4 # 定義基礎模型 (Unit Operations)  
5 estimators = [ ('rf', RandomForestRegressor()),  
6                 ('svr', SVR()),  
7                 ('ridge', Ridge())]  
8  
9 # 定義堆疊模型 (System Architecture)  
10 reg = StackingRegressor(  
11     estimators=estimators,  
12     final_estimator=Ridge(alpha=0.5), # Meta-learner  
13     cv=5, # Cross-Validation  
14     passthrough=False # 是否傳遞原始特徵  
15 )
```

建議使用簡單模型 (如 Ridge) 避免過擬合

若設為 True，則將原始特徵 X 直接旁路 (Bypass) 傳給元學習器

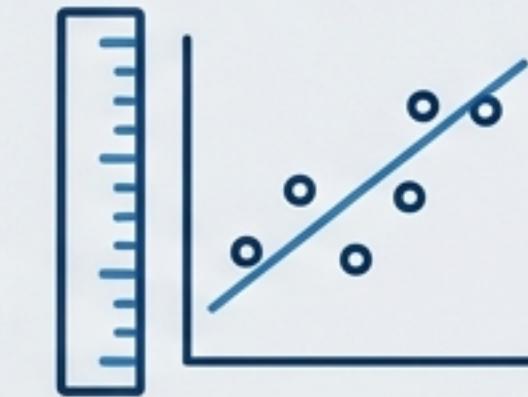
模型選擇策略：多樣性是關鍵

Parts Catalog (組件目錄)

Linear Models

Ridge, Lasso

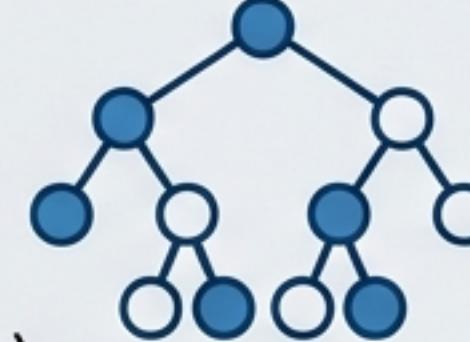
擅長線性關係，提供穩定基準 (Stable Baseline)。



Tree Models

Random Forest, GBDT

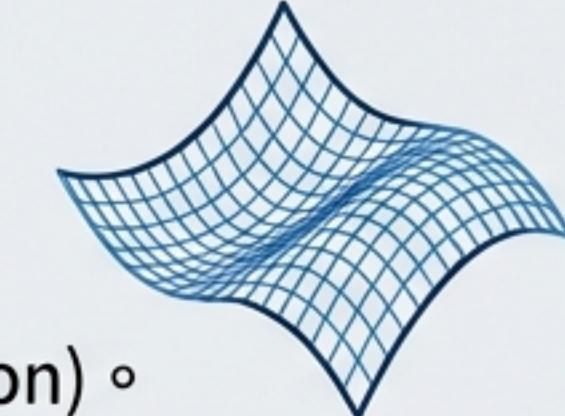
捕捉非線性與特徵交互 (Non-linear & Interactions)。



Kernel Methods

SVR

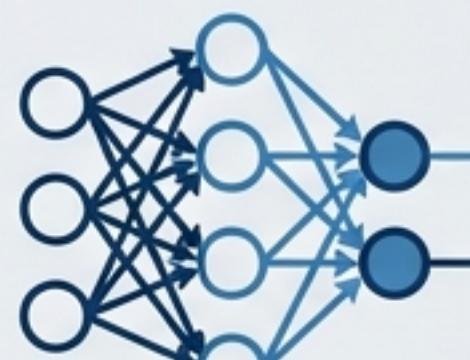
高維空間優化 (High-dimensional Optimization)。



Neural Networks

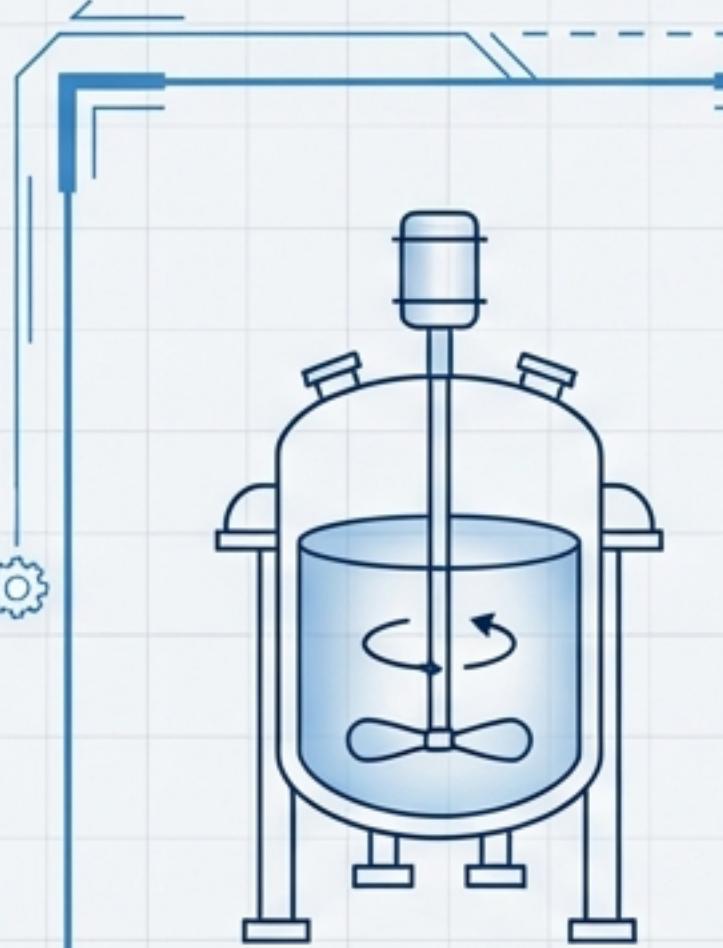
MLP

複雜模式擬合。



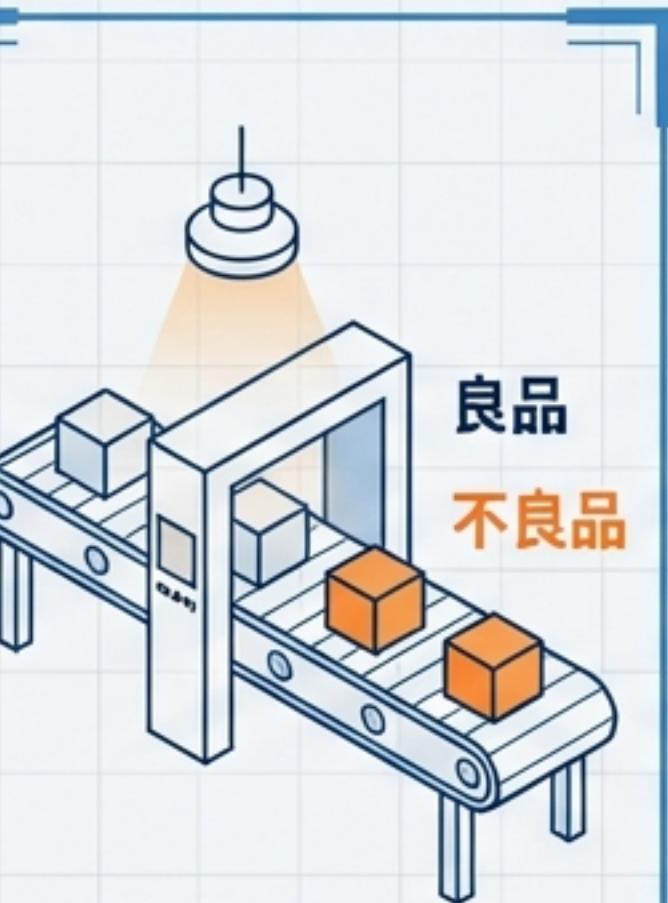
Selection Rule: 基礎模型應具備「異質性」(Heterogeneity)。
如果所有模型犯錯的地方都一樣，Stacking 就無效。

化工領域的應用場景



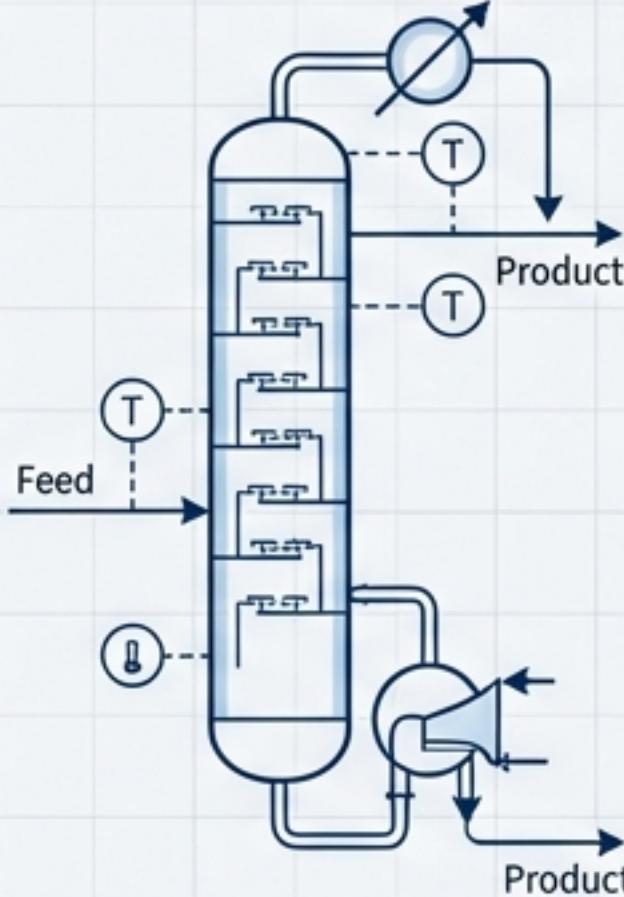
反應器溫度控制

處理非線性催化劑衰減
與線性流體力學。



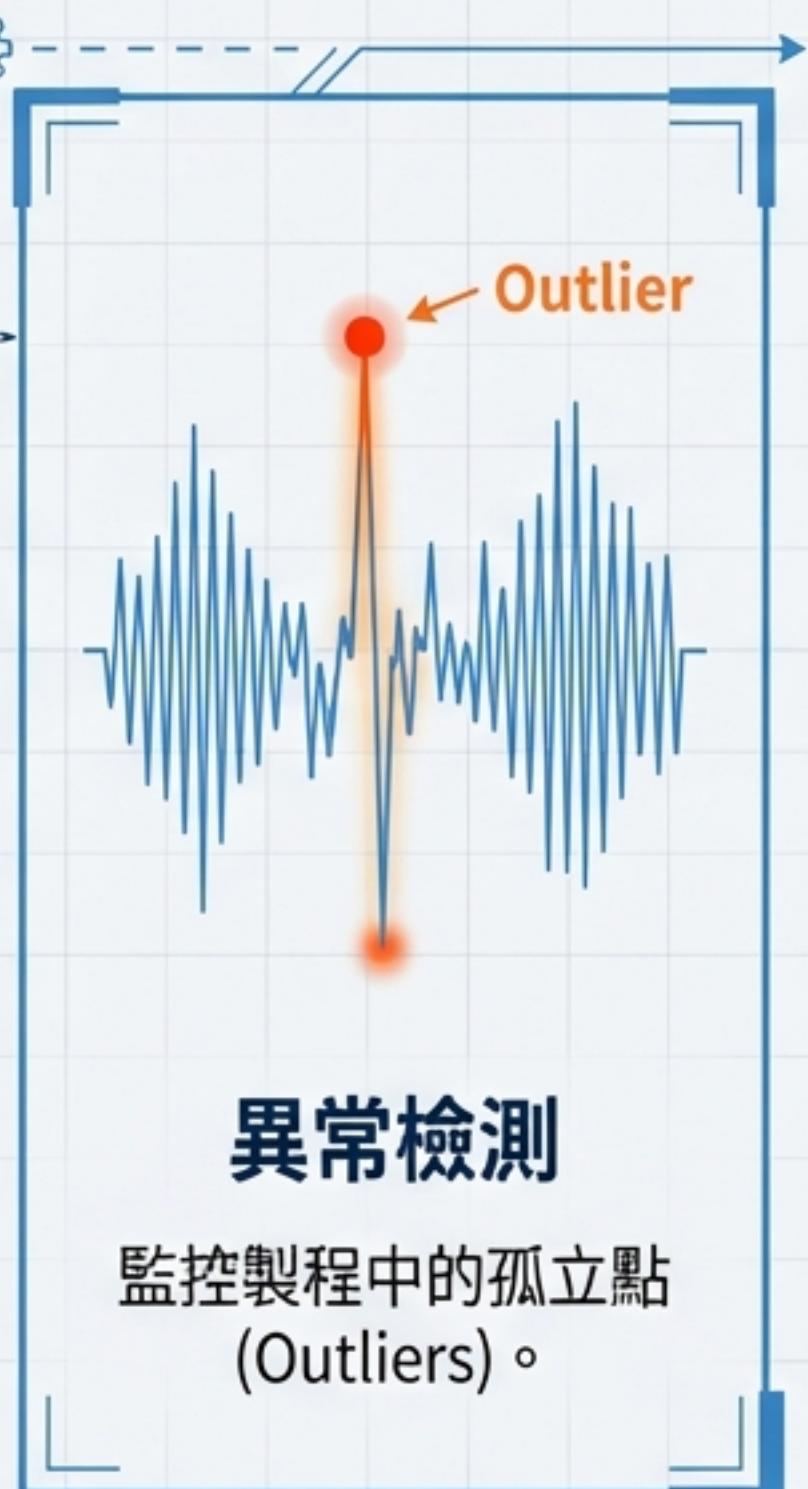
產品品質分類

良品/不良品判定，處
理不平衡數據。



蒸餾塔優化

同時預測塔頂/塔底溫度
與純度 (Multi-output)。

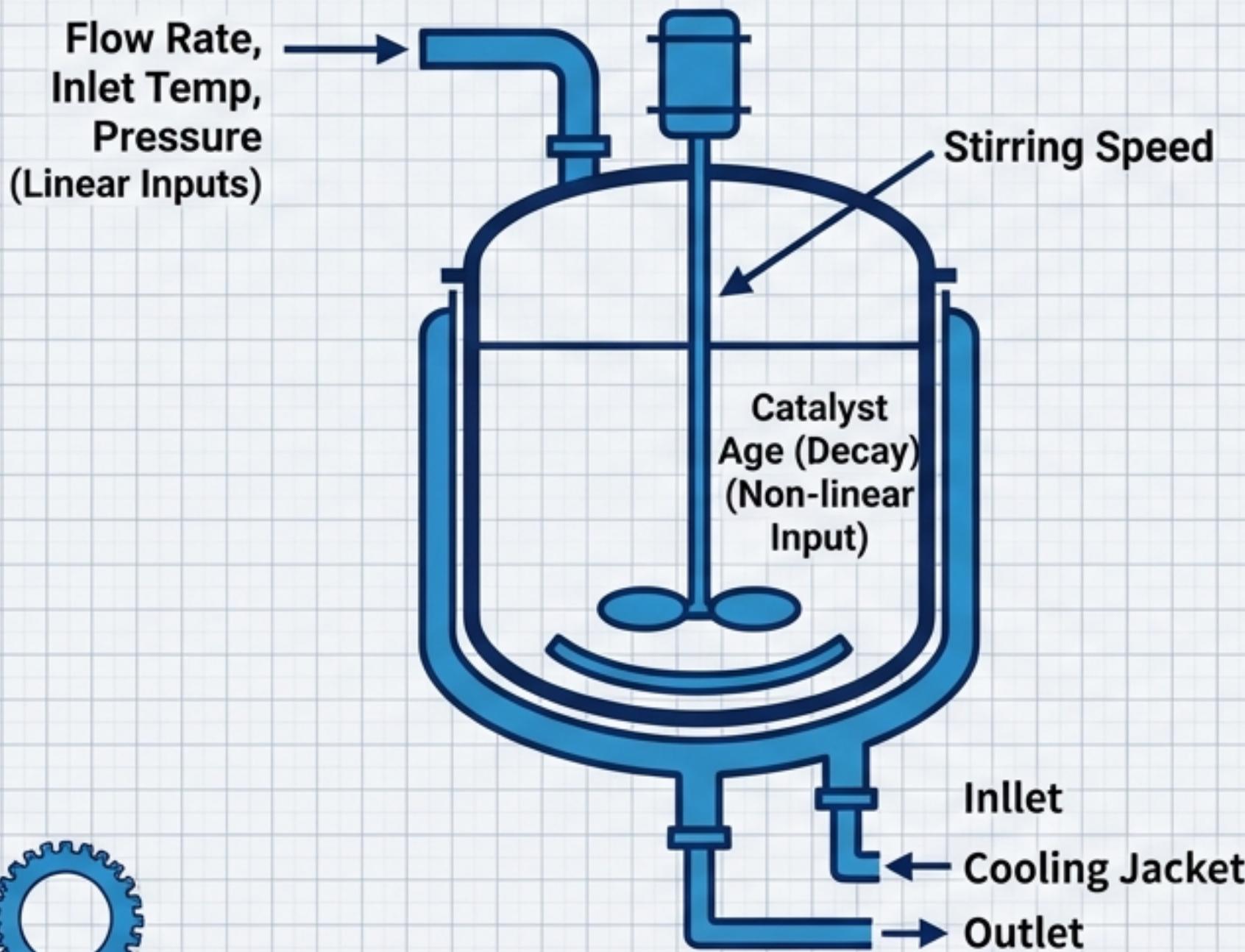


異常檢測

監控製程中的孤立點
(Outliers)。

案例 1：反應器溫度預測

目標：預測出口溫度 T_{outlet}



挑戰 (Challenge)：

單一模型難以同時完美捕捉「線性流
流體關係」與「非線性催化劑老化」

$$\text{Catalyst Decay: } -0.0001 \times t_{catalyst}^2$$

性能分析：誤差降低 10.5%

Metrics Comparison

Best Single Model (GBDT)

RMSE = 7.42°C

Stacking Model

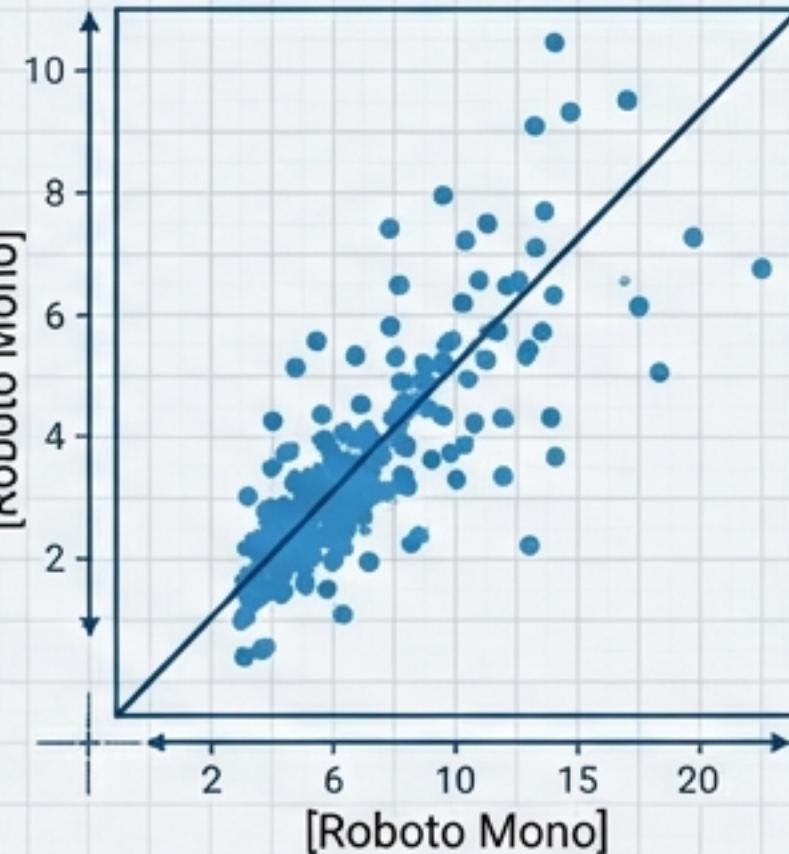
RMSE = 6.64°C

Improvement

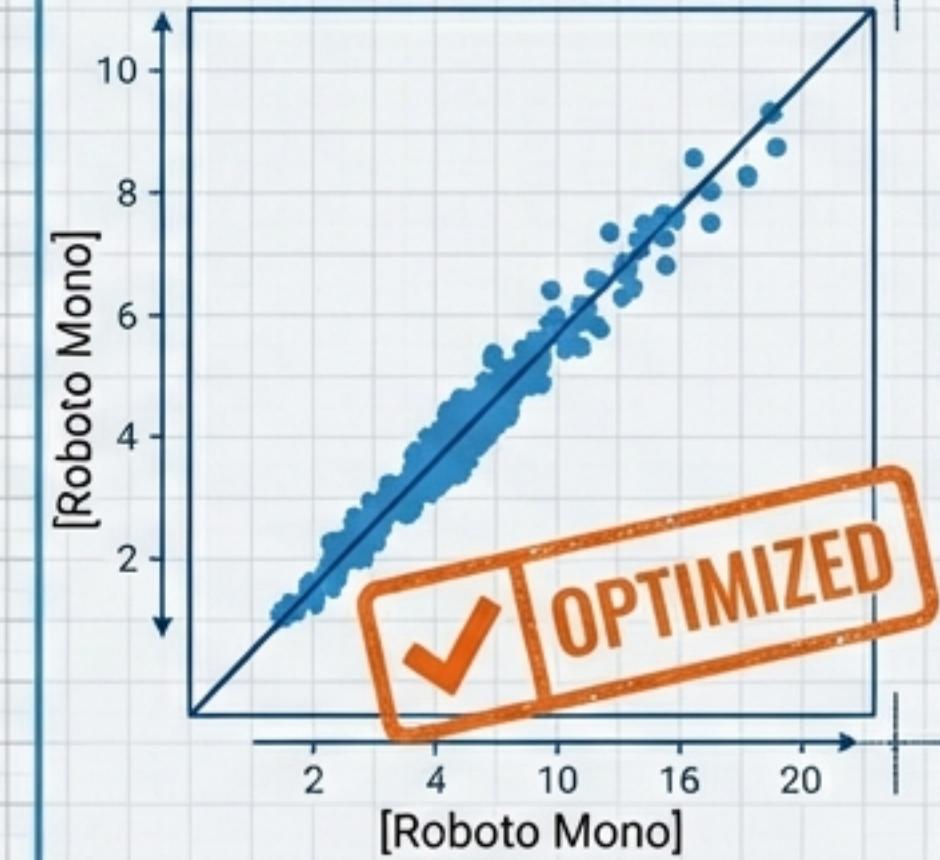
-10.5%

預測誤差降低 0.78°C ，在年產量千噸級工廠可節省巨額能耗成本。

A Single Model (GBDT)



B Stacking Model

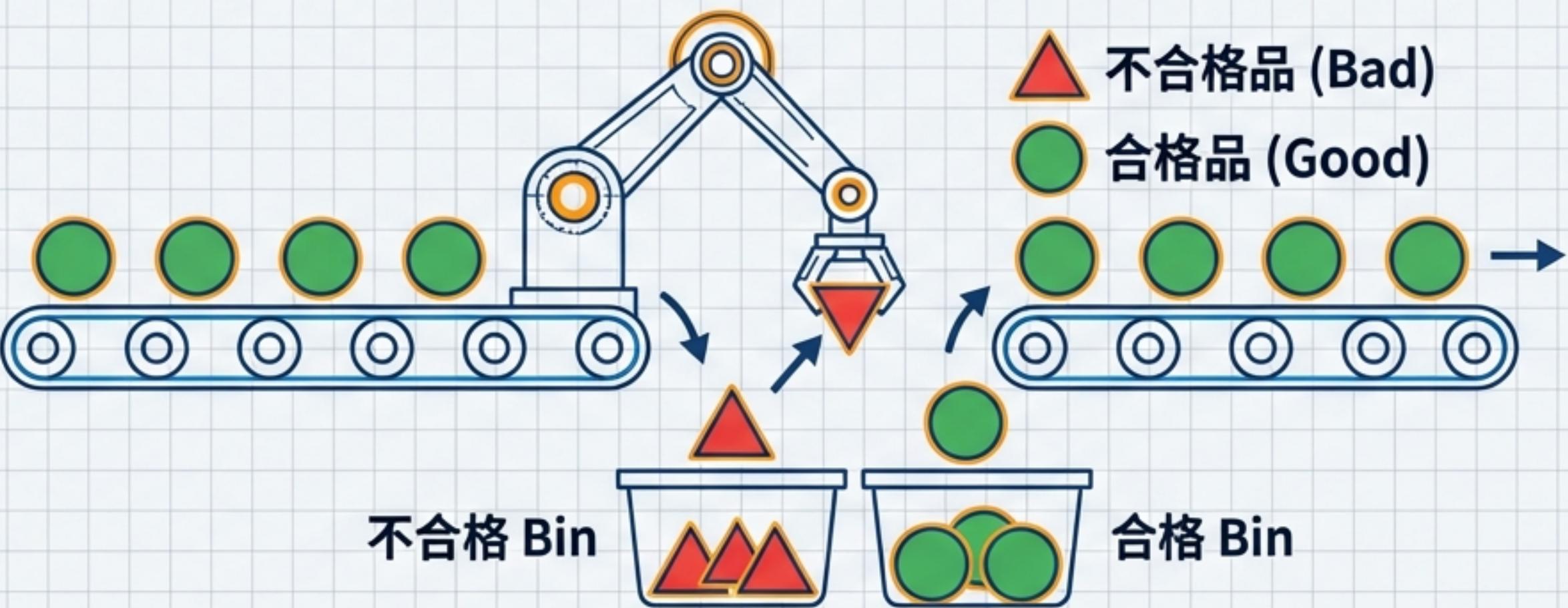


案例 2：產品品質判定

二元分類 (合格 vs 不合格) | 不平衡數據 (1:0.8)

特徵 (Features)
Purity (純度)
Viscosity (黏度)
Temperature

Input Ratio:
80% Good / 20% Bad



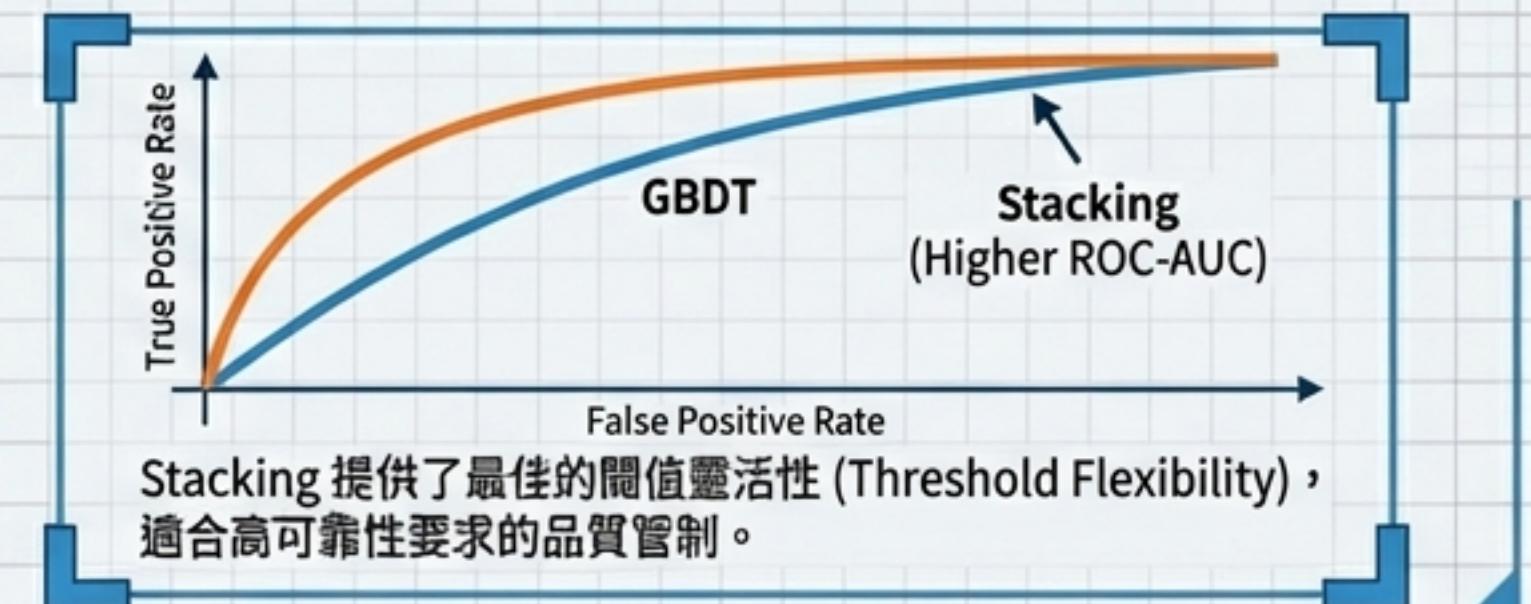
The Trade-off: 工業場景中，我們更關注 Precision (精確率) —— 「寧可漏
檢，不可誤判」。(Better to reject a good product than ship a bad one).

決策優化：追求最高的精確率

Comparison Data

- GBDT: Recall 較高(**0.9438**)，但 Precision 較低 (**0.9042**)
- Stacking: Precision 最高 (**0.9444**)，ROC-AUC 最高 (**0.9161**)

Confusion Matrix - Digital Production Report		
	Predicted Good	Predicted Bad
Actual Good	756 True Positive	44 False Negative (Missed)
Actual Bad	5 False Positive ▲ Low Risk (關鍵)	195 True Negative



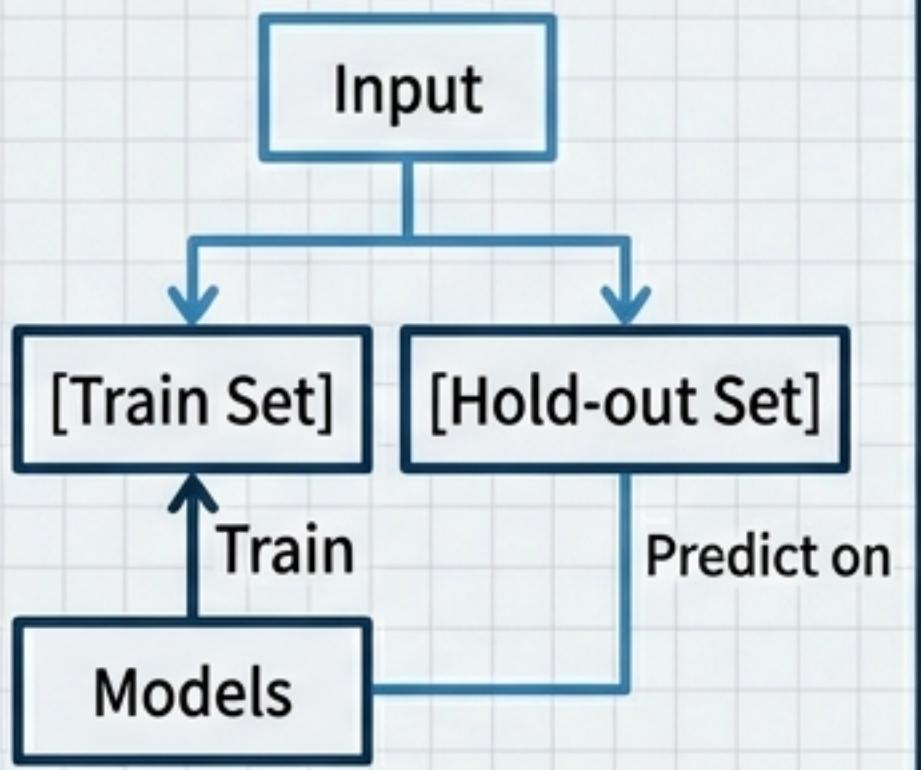
實務建議與最佳實踐

CHECKLIST

- 多樣性 (Diversity):**
混用 Linear, Tree, Kernel 模型。
- 簡單化 (Simplicity):**
Meta-learner 應保持簡單 (如 Ridge, Logistic) 以避免過擬合。
- 交叉驗證 (Cross-Validation):**
絕對禁止使用訓練集直接生成元特徵 (No Leakage)。
- 基準測試 (Baseline):**
先優化單一模型，若提升 < 1% 則考慮成本效益。

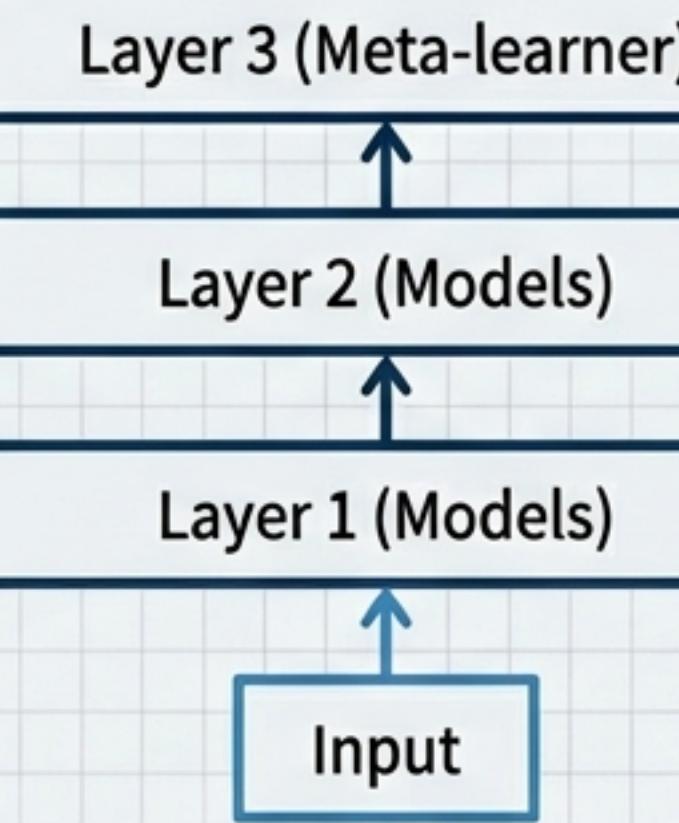
進階架構延伸

Blending



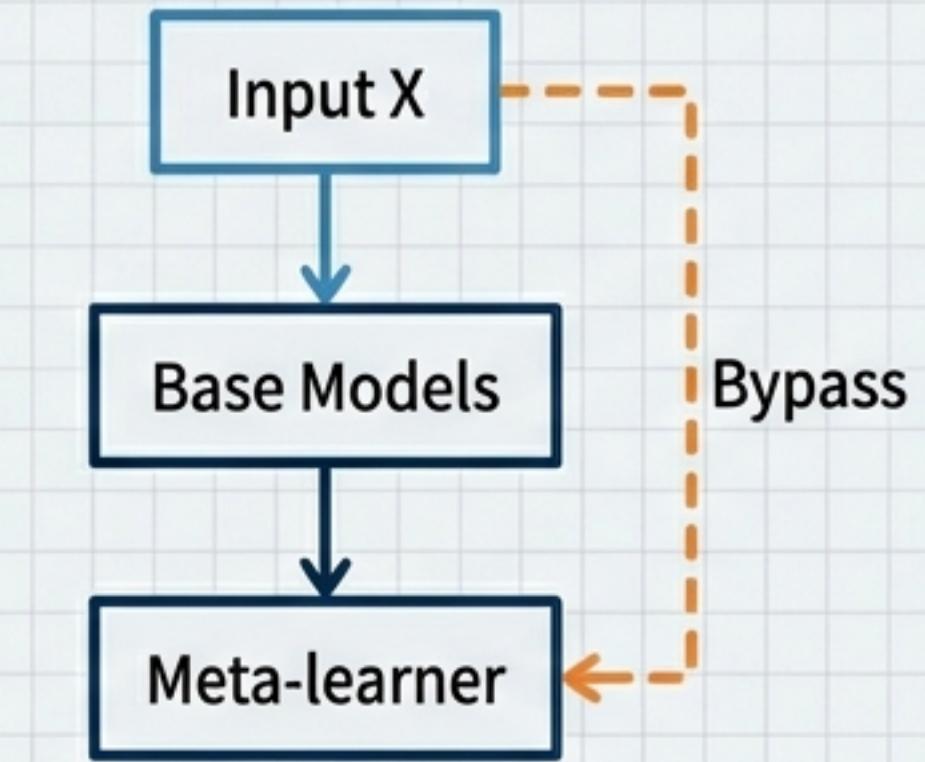
使用留出集 (Hold-out set) 而非 CV，速度快但浪費數據。

Multi-Level Stacking



增加第三層、第四層 (Kaggle常見，工業界少用)。

FWLS (Feature Weighted)



元學習器同時接收預測值與原始特徵 (`passthrough=True`)。

總結：高成本帶來的高回報

極致性能 (Performance)



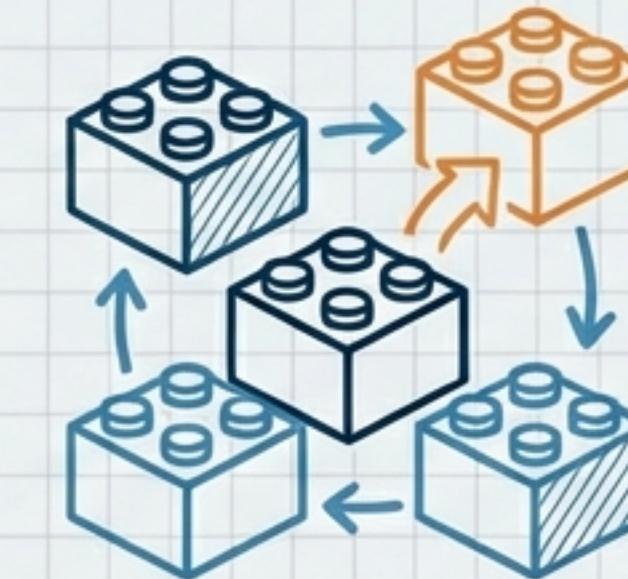
突破單一模型的上限
(RMSE -10.5%)。

系統穩健性 (Robustness)



對抗異常值與噪聲，由
多多個專家共同決策。

靈活架構 (Flexibility)



可模組化替換基礎模型
，適應不同化工場景。

注意：計算成本高 (High Computational Cost)，不適合毫秒級 (Real-time) 應用。⚠

下一步：啟動您的數位工廠

- > 開啟 Unit13_Stacking_Regression.ipynb
- > 開啟 Unit13_Stacking_Classification.ipynb
- > 實驗不同的 Base Models 組合

Theory is nothing without practice. Start coding.
(理論是不夠的，您需要親自寫 Code。)