Decision Tree and Ensemble Learning
(第 8 章 決策樹與集成學習)

Chia-Yen Lee, Ph.D. (李家岩 博士)

Department of Information Management (資訊管理學系)
National Taiwan University (國立台灣大學)

目次

- 第一章 製造數據科學
- 第二章 製造系統分析與管理
- 第三章 數據科學基礎與模型評估
- 第四章 數據科學分析架構與系統運算決策
- 第五章 數據預處理與製造數據特性
- 第六章 線性分類器
- 第七章 無母數迴歸與分類
- **第八章 決策樹與集成學習**
- 第九章 特徵挑選與維度縮減
- 第十章 類神經網路與深度學習
- 第十一章 集群分析
- 第十二章 特徵工程、數據增強與數據平衡
- 第十三章 故障預測與健康管理
- 第十四章 可解釋人工智慧
- 第十五章 概念漂移
- 第十六章 元啟發式演算法
- 第十七章 強化學習

藍：老師課堂講授
綠：學生自學

- 附錄A 線性迴歸
- 附錄B 支持向量機
- 附錄C 統計製程管制與先進製程控制
- 附錄D 超參數最佳化

- 應用涵蓋

  產能規劃、瑕疵檢測、製程監控與診斷、機台保養、需求預測、 生產排程、電腦視覺、自動光學檢測、原料價格預測與採購等

決策樹

- 在實際數據中，特徵之間的關係可能為非線性且存在交互作用
  - 例如溫度到達某個程度後良率將直接大幅提升(非線性)
  - 線性迴歸模型的假設可能被違反或不足以詮釋特徵與目標間的非線性與交互作用關係

- 決策樹兩種思維
  - 「階層式的規則判斷」
  - 「特徵間的直線切割」
  - 描繪出數據中不同特徵空間的「非線性的階層關係」與特徵之間的「交互作用」

決策樹特徵重要性

- 說明了愈往上層(樹根)的特徵相對愈重要，如案例中溫度特徵的重要性比壓力來得重要

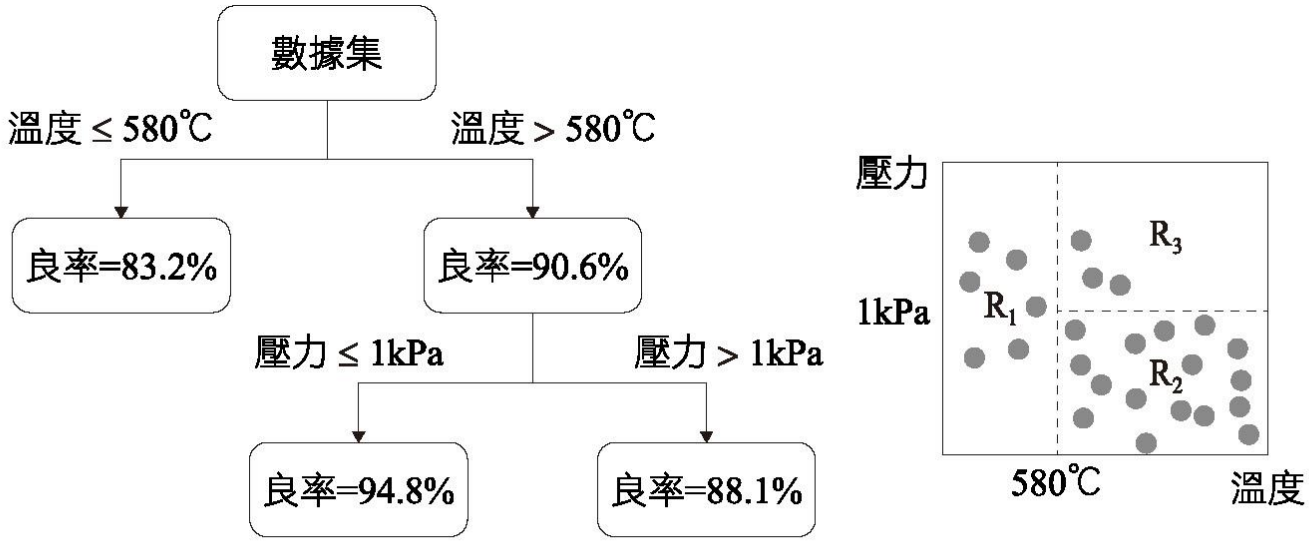


圖 8.1　決策樹的階層式結構與特徵空間切割

決策樹的兩種類型

- 迴歸樹：反應變數為連續
- 分類樹：反應變數為類別(離散)

迴歸樹的預測與解釋

- 迴歸樹將數據切割成M個區域分別為$R_1, R_2, \ldots, R_M$，則迴歸樹的預測值即是在每一區域預測一個常數$\hat{c}_m$
  - $\hat{f}(x) = \sum_{m=1}^{M} \hat{c}_m I(x \in R_m)$
- 其中$I(x \in R_m)$為指標函數，表示特徵是否落於區域R_m，而每個區域的預測值$\hat{c}_m$是由該區樣本的目標值平均所估計出的，其估計值表示為$\hat{c}_m$，而N_m代表區域m的樣本數
  - $\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$ ， $N_m = \sum_{i=1}^{n} I(x_i \in R_m)$

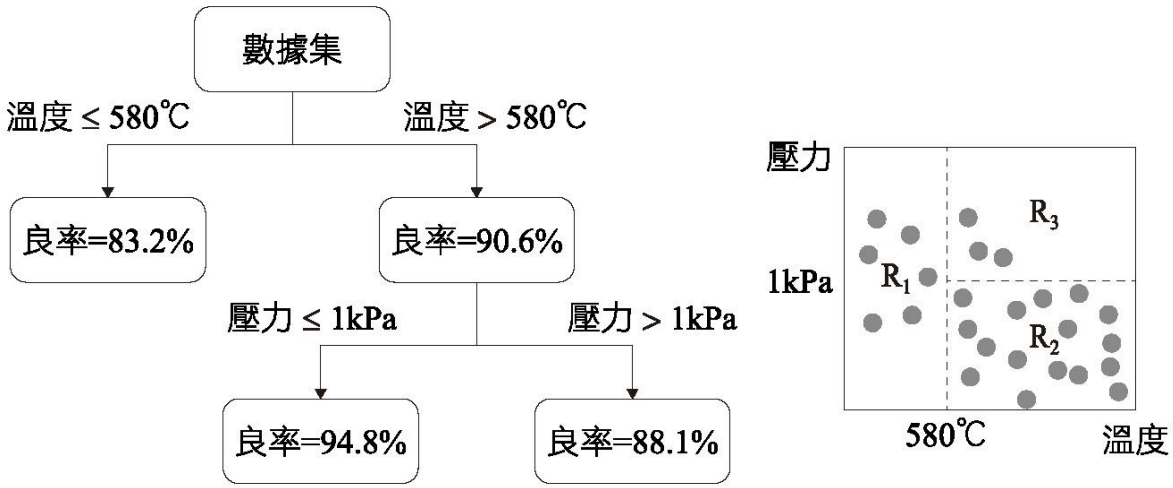


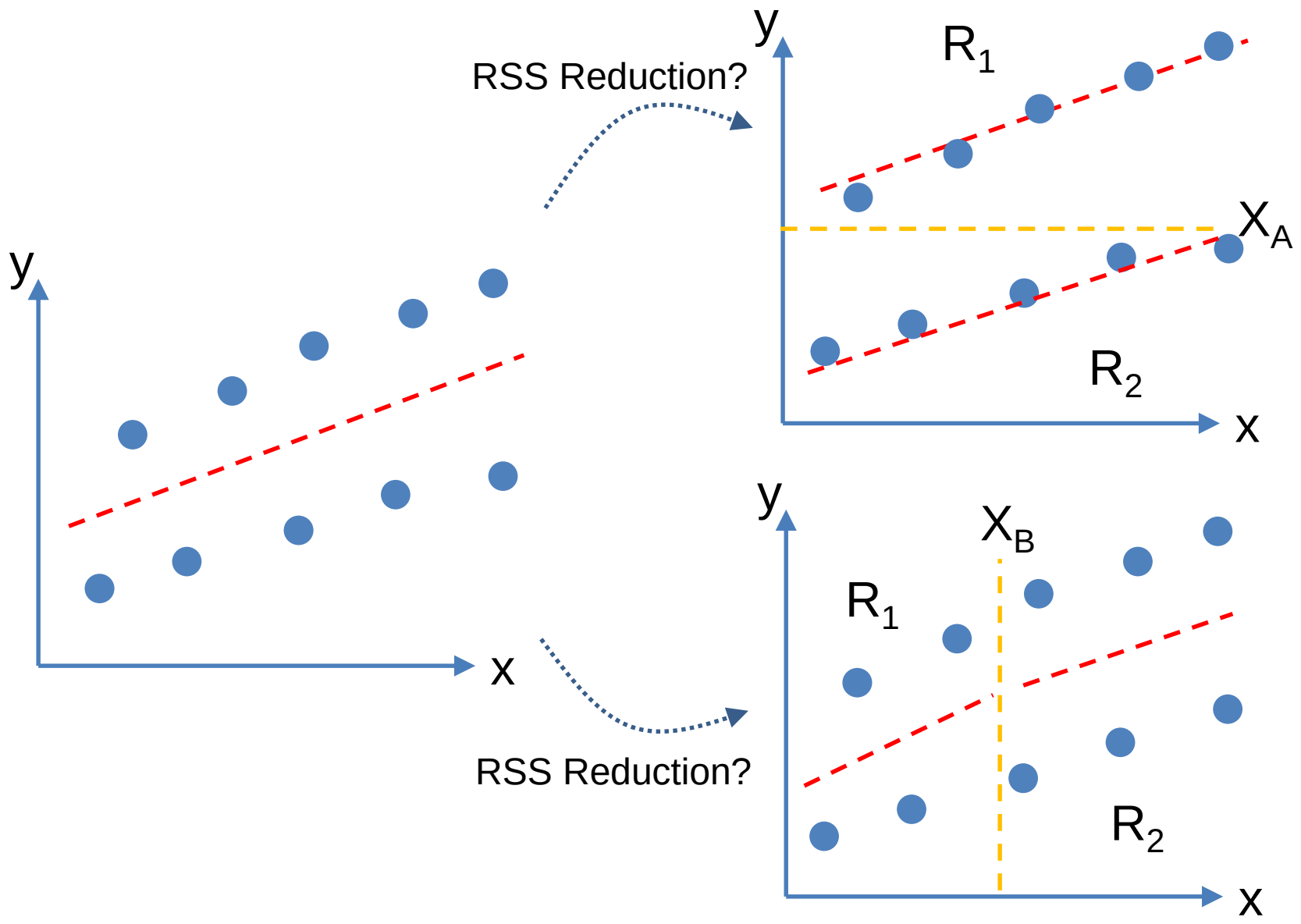
圖 8.1　決策樹的階層式結構與特徵空間切割

迴歸樹的建構

- 理想上，數據空間應該會有一個由多個特徵與分割點形成的最佳分割組合，也就是期望能最小化每個區域的「殘差平方和」(residual sum of squares, RSS)
  - $\text{RSS} = \sum_{m=1}^{M} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$

- 不論是迴歸樹或分類樹，均採用由上而下的「貪婪方法」(top-down greedy)，被稱為「遞迴二元分割」(recursive binary splitting)，分割過程如公式
  - $\min_{j,s} \left[\sum_{x_i \in R_1(j,s)} (y_i - \hat{c}_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{c}_2)^2 \right]$

RSS Reduction



- 事前修剪(修剪可預防過度配適)
  - 每次分割特徵空間生長樹的時候，評估殘差平方和的減少量是否超過某個事先設定的閥值(threshold)，若滿足該條件才產生分支，然而其隱含的缺點是閥值的設定相當困難
- 事後修剪
  - 一般情形下，建議採用事後修剪來調整模型。
  - 策略是先產生非常大的樹T，接著再將這棵樹修剪成它的子樹T^*。

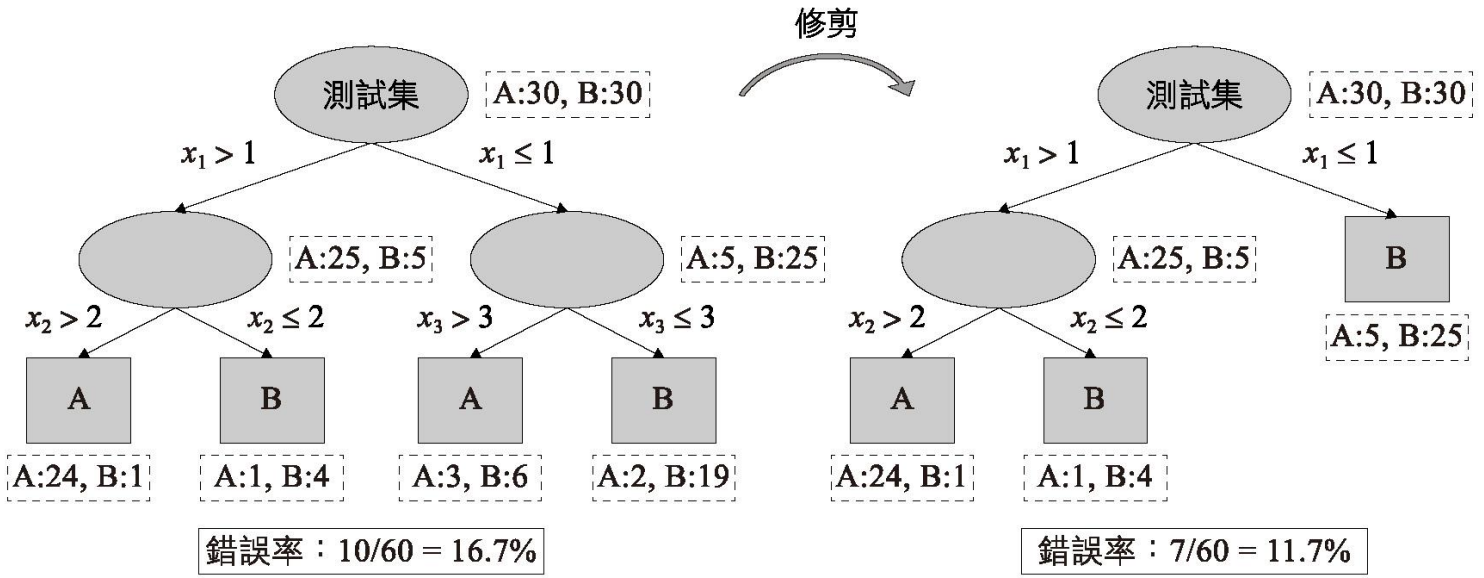


圖 8.2　決策樹的事後修剪

如何挑選修剪的子樹

- 成本複雜性修剪(cost complexity pruning或weakest link pruning)
  - $\sum_{m=1}^{M}\sum_{i\in R_m}(y_i-\hat{c}_m)^2+\alpha M$
- 當懲罰參數α為零時，則為原本最複雜的樹，當懲罰參數α越大時，則模型將為其樹節點總數付出相對應的懲罰，因此產生一棵較小的樹，而這個懲罰參數可藉由交差驗證而得出

表 8.2　迴歸樹的演算法

演算法 CART - 迴歸樹（Regression Tree）

輸入：訓練集$S=(x_1,y_1),\ldots,(x_n,y_n)$，給定超參數$\gamma_1$與$\gamma_2$
輸出：迴歸樹模型$\hat{F}_{rt}$

1. 使用 K-fold 交叉驗證選擇一個最佳的懲罰權重α^*
2. For $i=1$ to K:
3. 將訓練集S使用遞迴二元分支訓練一棵很大的樹，但訓練時不考慮第K折訓練集（驗證集），而樹生長的停止準則為每個葉節點的樣本數均小於γ_1或分支後的葉節點樣本數小於γ_2
4. 使用成本複雜性修剪將原本很大的樹修剪成一棵較小的子樹
5. 計算第K折訓練集（驗證集）的殘差平方差
6. End;
7. 將每個不同懲罰權重α於交叉驗證得到的誤差平均，並選擇平均誤差最小的懲罰權重α^*，並以該超參數對整個訓練集生長一棵迴歸樹，得到最終的迴歸樹模型$\hat{F}^{RT}$
8. Return 迴歸樹模型$\hat{F}^{RT}$

分類樹

- 分類樹同樣使用遞迴二元分支來生長分類樹，以分類的視角計算分類誤差，最常用的分類誤差為「分類錯誤率」
  - $\hat{p}_{mk} = \frac{1}{N_m}\sum_{x_i \in R_m} I(y_i = k)$
  - $g(m) = \arg\max_k \hat{p}_{mk}$
  - 分類錯誤率 $= \frac{1}{N_m}\sum_{x_i \in R_m} I(y_i \neq g(m)) = 1 - \hat{p}_{mg(m)}$
  - 然而分類錯誤率對於樹的生長是相對不敏感的

- 因此分類樹通常使用「吉尼不純度」或「熵」，並計算其「資訊增益」(information gain)作為分支準則
  - 在分類問題上，若隨機猜測，則預測結果誤判情形相對較高，這種不確定性(變異)所代表的即是資訊量。若在建模後分類相當精準，不確定下降，代表數據資訊量被模型解釋了，因此資訊量會變少，而我們稱不確定的資訊量為「不純度」

吉尼不純度(Gini impurity)

- 計算不同類別之間「比例變異」的總和，以代表K個類別不純度的加總
  - 當$\hat{p}_{mk}$均接近0.5時，則吉尼不純度的值會變得很大(最大值0.5)
  - 當$\hat{p}_{mk}$均接近0或1時，則吉尼不純度的值會變得很小(最小值0)
- $\text{Gini} = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2$

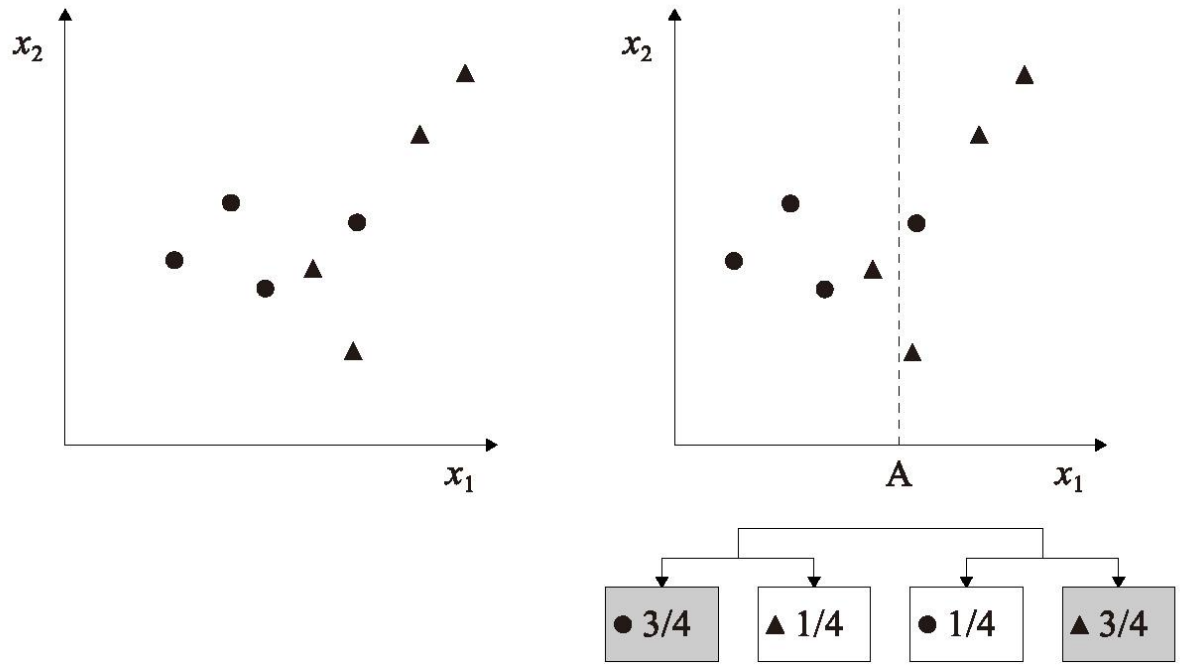


圖 8.3　不純度的說明

- 熵(Entropy)
  - 它計算不同類別之間「比例乘上比例對數」的總和〔具有對數概似函數(log-likelihood)的思維〕，以代表K個類別不純度的加總
    - 當$\hat{p}_{mk}$均接近0.5時，則熵的值會變得很大(最大值1)
    - 當$\hat{p}_{mk}$均接近0或1時，則熵的值會變得很小(最小值0)
  - $\text{Entropy} = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

- 從微分特性的角度而言，「吉尼不純度」與「熵」是可微且變化明顯的，因此相較於分類錯誤率更適合用於數值最佳化

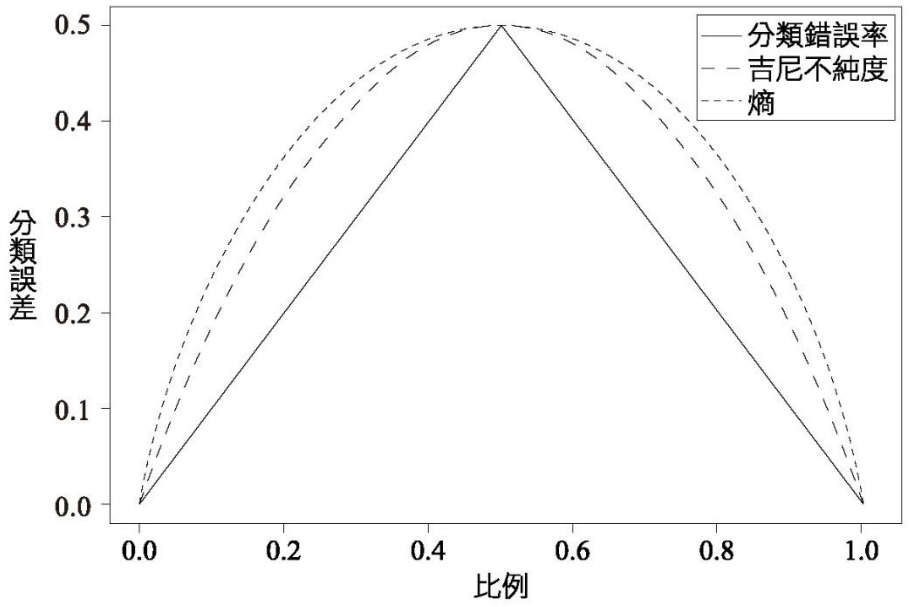


圖 8.4　吉尼不純度、熵與分類錯誤率

分類樹

- 計算分支後不純度所降低的量，才能找出最適的特徵分割點
  - 代表不純度降低的量稱作「資訊增益」(information gain, IG)
  - $IG(D,A) = \underbrace{I(D,A)}_{\text{原始不純度}} - \underbrace{\frac{N_{left}}{N} I\left(D_{left}, A\right)}_{\text{左邊不純度}} - \underbrace{\frac{N_{right}}{N} I\left(D_{right}, A\right)}_{\text{右邊不純度}}$
  - 函數I為對數據集D與特徵A的任意不純度函數， 因此套用「吉尼不純度」與「熵」即可得到「吉尼增益」(Gini gain)以及「熵增益」(entropy gain)

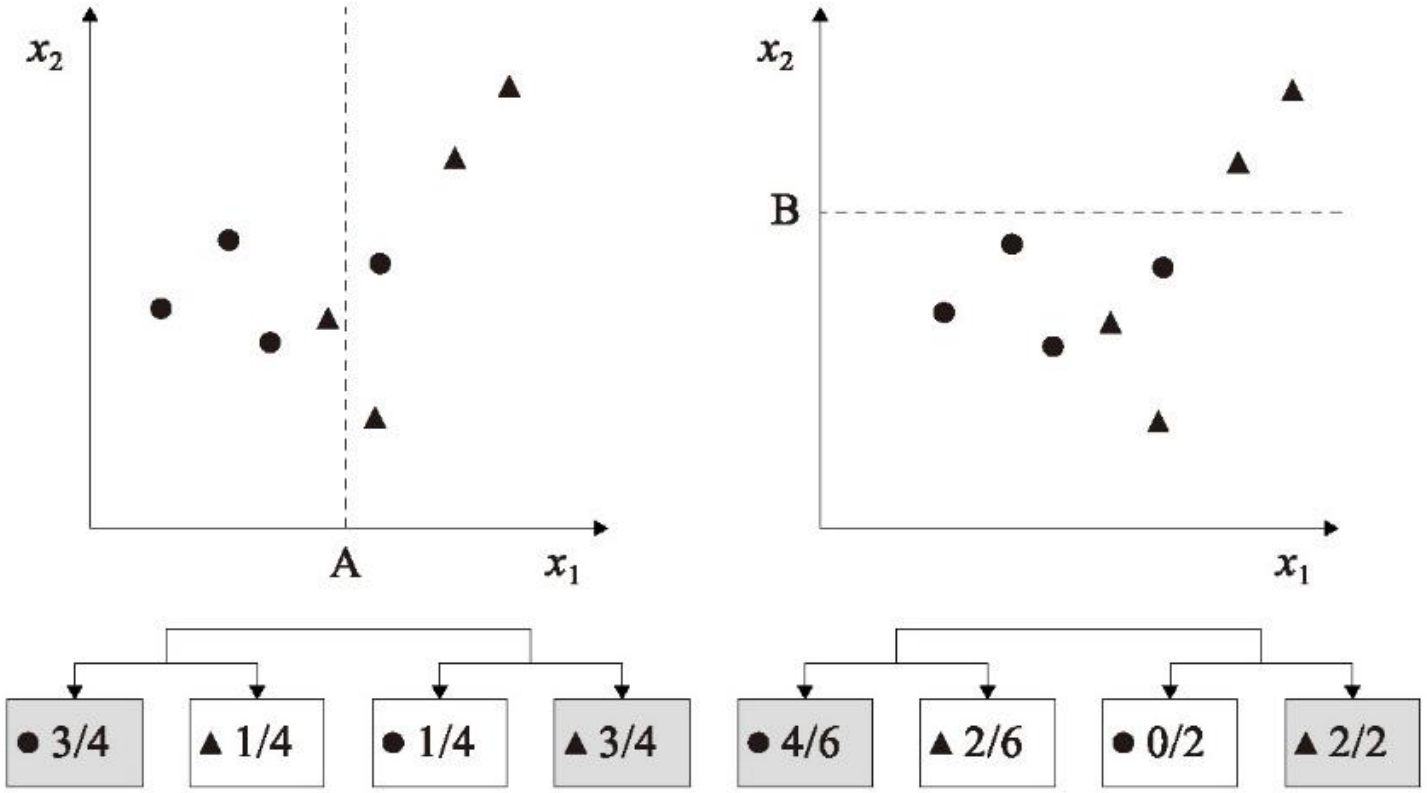


圖 8.5　資訊增益在不同切割下的計算

分類錯誤率的資訊增益

- 模型A的分類錯誤率I_M^A與其增益IG_M^A
  - $I_M(D) = 1 - \frac{4}{8} = 0.5$
  - $I_M^A(D_{left}) = 1 - \frac{3}{4} = 0.25$
  - $I_M^A(D_{right}) = 1 - \frac{3}{4} = 0.2$
  - $IG_M^A = I_M(D) - \frac{N_{left}}{N} I_M^A(D_{left}) - \frac{N_{right}}{N} I_M^A(D_{right}) = 0.5 - \frac{4}{8} \cdot 0.25 - \frac{4}{8} \cdot 0.25 = 0.25$
- 模型B的分類錯誤率I_M^B與其增益IG_M^B
  - $I_M^B(D_{left}) = 1 - \frac{4}{6} = 0.33$
  - $I_M^B(D_{right}) = 1 - \frac{2}{2} = 0$
  - $IG_M^B = I_M(D) - \frac{N_{left}}{N} I_M^B(D_{left}) - \frac{N_{right}}{N} I_M^B(D_{right}) = 0.5 - \frac{6}{8} \cdot 0.33 - \frac{2}{8} \cdot 0 = 0.25$
- 經由A、B兩種模型所得到「分類錯誤率」的增益是相同的($IG_M^A = IG_M^B = 0.25$)

吉尼不純度的資訊增益

- 模型A的吉尼不純度I_G^A與其增益IG_G^A
  - $I_G(D) = \left(\frac{4}{8}\cdot\frac{4}{8}\right) + \left(\frac{4}{8}\cdot\frac{4}{8}\right) = 0.5$
  - $I_G^A(D_{left}) = \left(\frac{3}{4}\cdot\frac{1}{4}\right) + \left(\frac{1}{4}\cdot\frac{3}{4}\right) = 0.375$
  - $I_G^A(D_{right}) = \left(\frac{1}{4}\cdot\frac{3}{4}\right) + \left(\frac{3}{4}\cdot\frac{1}{4}\right) = 0.375$
  - $IG_G^A = I_G(D) - \frac{N_{left}}{N} I_G^A(D_{left}) - \frac{N_{right}}{N} I_G^A(D_{right}) = 0.5 - \frac{4}{8}\cdot 0.375 - \frac{4}{8}\cdot 0.375 = 0.125$
- 模型B的吉尼不純度I_G^B與其增益IG_G^B
  - $I_G^B(D_{left}) = \left(\frac{4}{6}\cdot\frac{2}{6}\right) + \left(\frac{2}{6}\cdot\frac{4}{6}\right) = 0.44$
  - $I_G^B(D_{right}) = \left(\frac{0}{2}\cdot\frac{2}{2}\right) + \left(\frac{2}{2}\cdot\frac{0}{2}\right) = 0$
  - $IG_G^B = I_G(D) - \frac{N_{left}}{N} I_G^B(D_{left}) - \frac{N_{right}}{N} I_G^B(D_{right}) = 0.5 - \frac{6}{8}\cdot 0.44 - \frac{2}{8}\cdot 0 = 0.17$
- 比較A、B兩種模型所得到「吉尼不純度」的增益，模型B所得到的較大$(IG_G^B = 0.17 > IG_G^A = 0.125)$。→因此建議先用模型B(特徵$x_2$)作為分支

熵的資訊增益

- 模型A的熵I_E^A與其增益IG_E^A
  - $I_E(D) = -\left(\frac{4}{8}\log\frac{4}{8} + \frac{4}{8}\log\frac{4}{8}\right) = 1$
  - $I_E^A(D_{left}) = -\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right) = 0.81$
  - $I_E^A(D_{right}) = -\left(\frac{1}{4}\log\frac{1}{4} + \frac{3}{4}\log\frac{3}{4}\right) = 0.81$
  - $IG_E^A = I_E(D) - \frac{N_{left}}{N} I_E^A(D_{left}) - \frac{N_{right}}{N} I_E^A(D_{right}) = 1 - \frac{4}{8}\cdot 0.81 - \frac{4}{8}\cdot 0.81 = 0.19$
- 模型B的熵I_E^B與其增益IG_E^B
  - $I_E^B(D_{left}) = -\left(\frac{4}{6}\log\frac{4}{6} + \frac{2}{6}\log\frac{2}{6}\right) = 0.92$
  - $I_E^B(D_{right}) = -\left(\frac{0}{2}\log\frac{0}{2} + \frac{2}{2}\log\frac{2}{2}\right) = 0$
  - $IG_E^B = I_E(D) - \frac{N_{left}}{N} I_E^B(D_{left}) - \frac{N_{right}}{N} I_E^B(D_{right}) = 1 - \frac{6}{8}\cdot 0.92 - \frac{2}{8}\cdot 0 = 0.31$
- 比較A、B兩種模型所得到「熵」的增益，模型B所得到的較大 ($IG_E^B = 0.31 > IG_E^A = 0.19$)。 →因此建議先用模型B(特徵$x_2$)作為分支

分類樹

- 因此比較「分類錯誤率」、「吉尼不純度」或「熵」各別於兩模型的增益後，結果可發現後兩者較能找出使得增益較大的分支，且兩者特性相似。結論建議使用模型B(也就是特徵$x_2 = B$)來做分支，將使得分類效果較佳

- 在常見的分類樹演算法中
  - CART演算法以「吉尼不純度」作為不純度指標
  - C4.5或C5.0演算法則以「熵」作為不純度指標

- 「資訊增益比」(information gain ratio, IGR)
  - Problem
    - Eg.: 10 x 0.1 x ln(0.1) = 2.3；3 x 0.33 x ln(0.33) = 1.1
  - 用於對資訊增益進行正規化(normalization)以利增益的比較，避免資訊增益在計算上偏袒水準越多的類別特徵
  - C4.5 (a successor of ID3) uses gain ratio to overcome the problem
    - $IGR(D,A) = \frac{IG(D,A)}{S(D,A)}$
    - $S(D,A) = -\sum_{k=1}^{K} \frac{N_k}{N} \log \frac{N_k}{N}$
    - E.g.

$$SplitInfo(age) = -[\frac{5}{14}\log\frac{5}{14} + \frac{4}{14}\log\frac{4}{14} + \frac{5}{14}\log\frac{5}{14}] = 1.5774$$

$$Gainratio = \frac{Gain(age)}{SplitInfo(age)} = \frac{0.94 - 0.6935}{1.5774} = \frac{0.2465}{1.5774} = 0.1563$$

  - The attribute with the maximum gain ratio is selected as the splitting attribute

決策樹

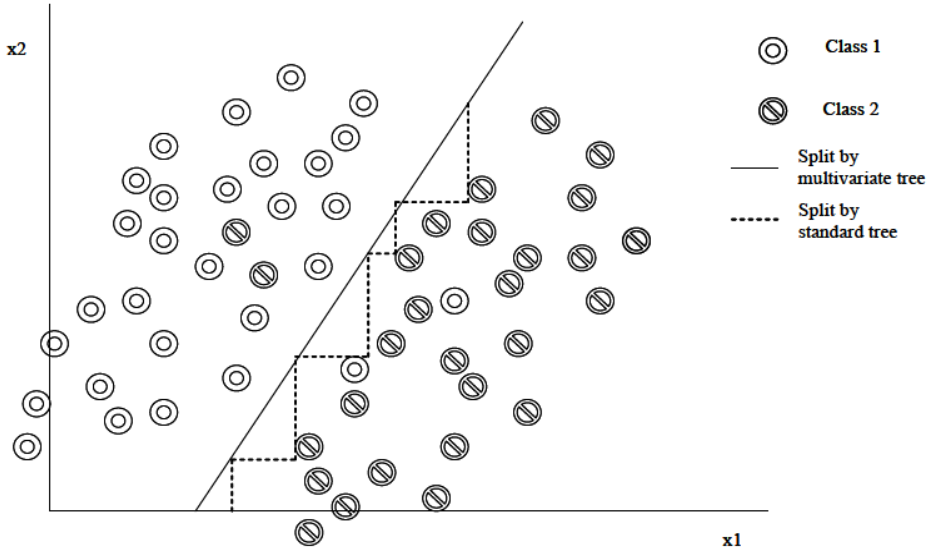
決策樹的優點

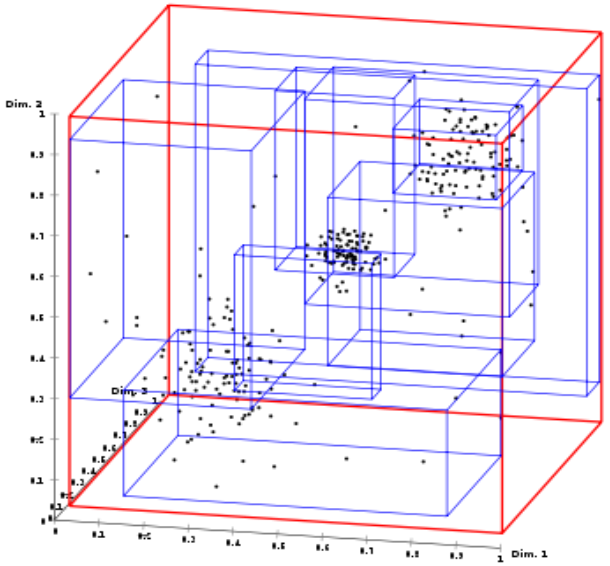
- 呈現出數據中的非線性與交互作用關係
- 可以對特徵重要性排序並建構出一個以決策為導向的模型與規則
- Nonparametric (no probabilistic assumptions)
- Automatically performs variable selection
- Uses any combination of continuous/discrete variables
  - Very nice feature: ability to automatically bin massively categorical variables into a few categories.
    - zip code, business class, make/model…
- Discovers “interactions” among variables
  - Good for “rules” search
  - Hybrid GLM-CART models (GLM: generalized linear model)
- CART handles missing values automatically
    - Using “surrogate splits”
- Invariant to monotonic transformations of predictive variable
- Not sensitive to outliers in *predictive* variables
  - Unlike regression
- Great way to explore, visualize data

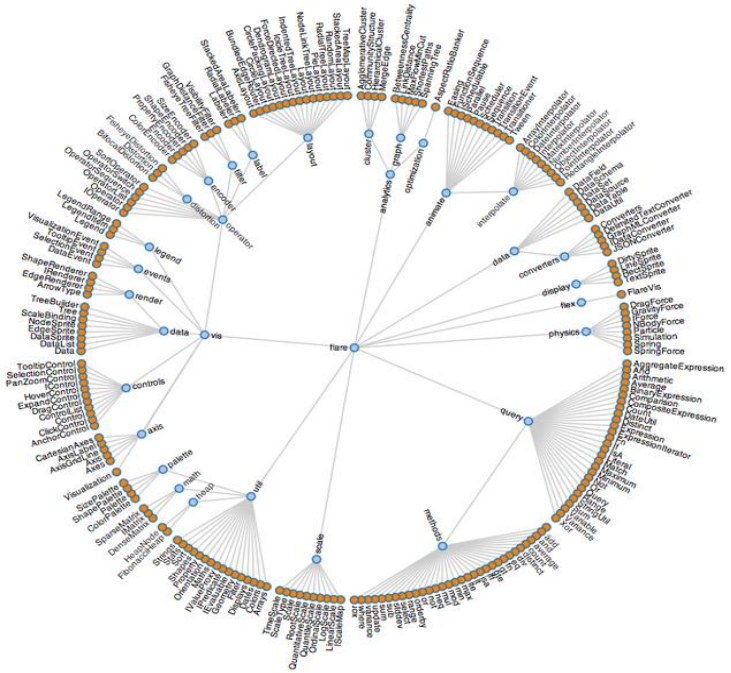
決策樹的缺點

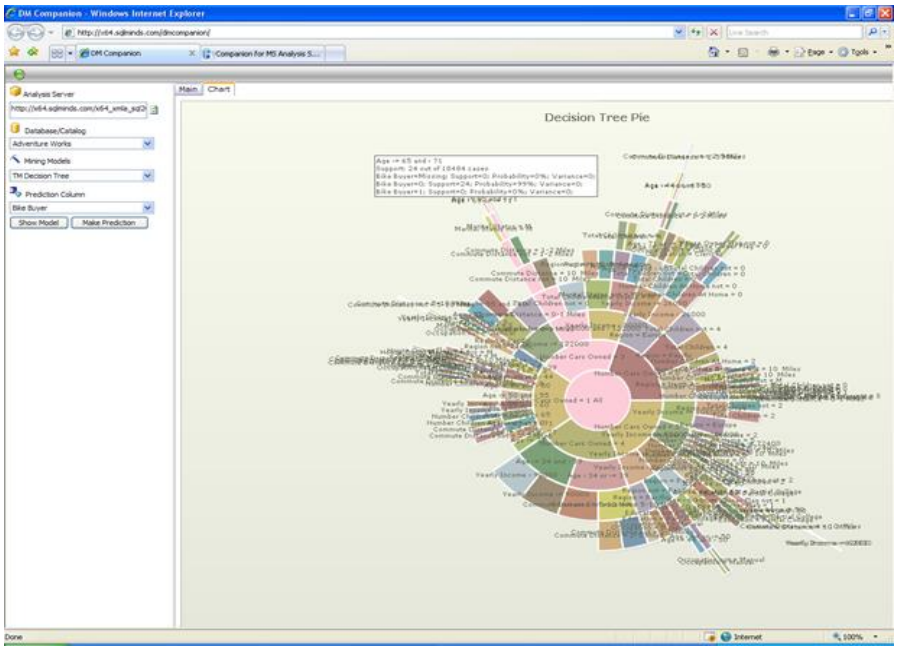
- 相較於集成模型、深度學習，預測效果仍然有限
- 平滑程度相當低，因此通常在迴歸問題上，難以建構出平滑的迴歸曲線
- 對於數據微幅的變化相當敏感，模型的建構並非穩健 且一致
- The model is a *step function*, not a continuous score
  - So if a tree has 10 nodes, $\hat{y}$ can only take on 10 possible values.
  - MARS improves this.
- Might take a large tree to get good lift
  - But then hard to interpret
  - Data gets chopped thinner at each split
- Instability of model structure
  - Correlated variables ➔ random data fluctuations could result in entirely different trees.
- CART does a poor job of modeling *linear structure*

決策樹的視覺化









http://blog.bigml.com/2013/04/19/a-new-way-to-visualize-decision-trees/

Algorithm Comparison

Decision Tree Algorithm Comparison

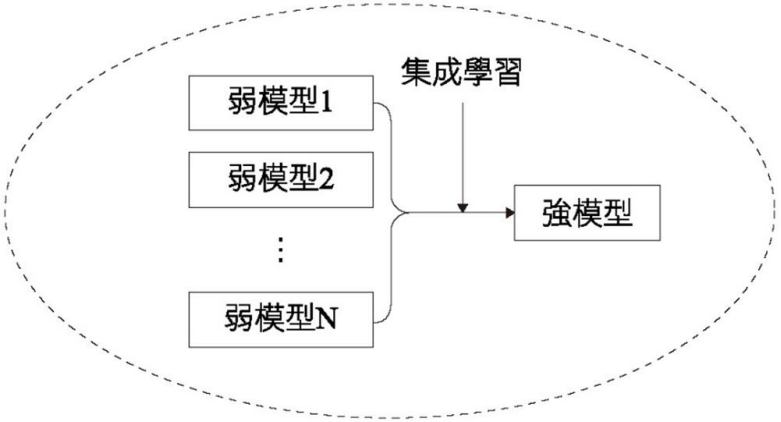
Algorithm	Proposed Year	Tree Type	Split Rule	Pruning Rule	Data Type	Theory Basis
ID3	Quinlan, 1986	Ternary Tree	Information Gain	No pruning	Category	Information theory
C4.5	Quinlan, 1993	Ternary Tree	Gain Ratio	Node predicted error rate	Category	Information theory
CART	Breiman et al., 1984	Binary Tree	Gini index	Entire error rate	Category and continuous	Information theory
CHAID	Kass, 1980	Ternary Tree	Chi-square test	No pruning	Continuous	Statistical test

Evaluating Classification Methods

- Predictive accuracy
  - Ability to predict the label of new observation
- Speed
  - Computational cost of model construction and prediction
- Robustness
  - Ability for accurate prediction even with noise and missing data
- Scalability
  - Ability to build the effective model for large dataset (BIG DATA)
- Interpretability
  - Model explanation and ability to provide the story and insights
- Goodness of rules
  - decision tree size
  - compactness of classification rules

集成學習的核心概念

- 透過建立多個基本的模型，稱之為弱模型或基底模型。藉由不同的集成策略，例如：裝袋法、提升法、堆疊法與混合法等。將這些基底模型集結成一個強大且穩健的模型，稱之為強模型

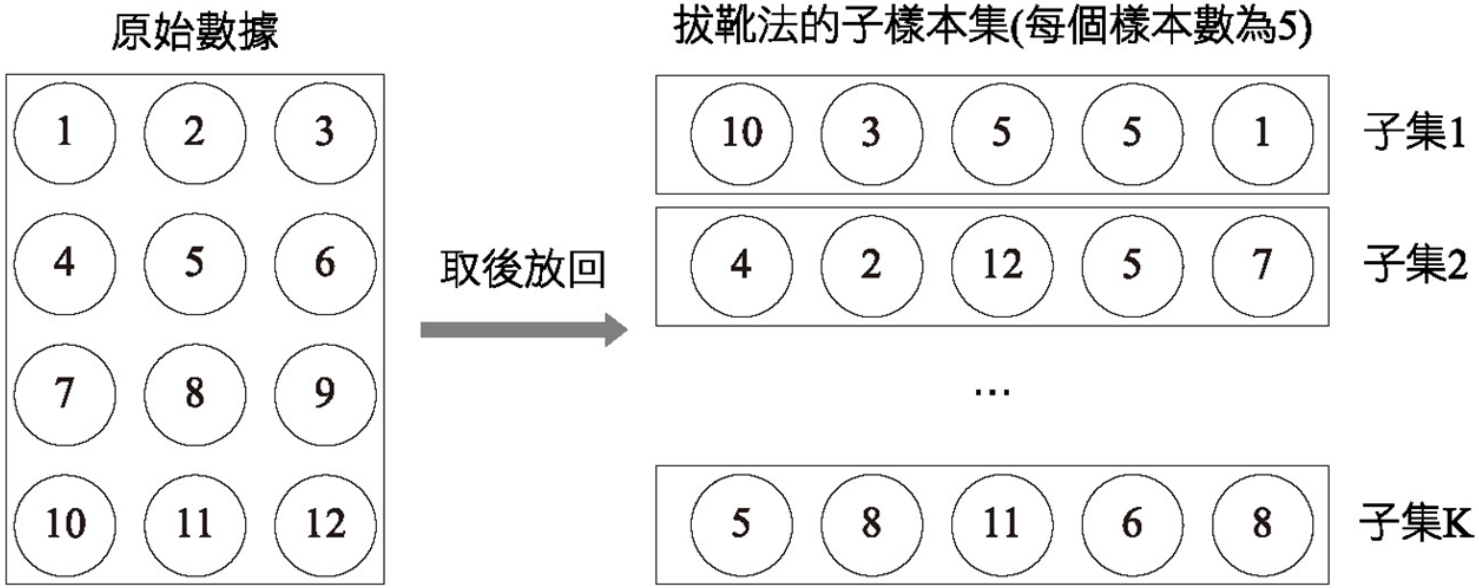


- 集成學習能提升預測結果的關鍵在於模型的異質性，因此會利用隨機抽樣、隨機變數挑選、樣本權重改變、或是不同模型的選擇，以增加基底模型(base learner)的異質性
- 在探討期望預測誤差時，我們期望能在偏誤和變異之間取得權衡。
  - 通過組合多個模塊來結合成一個複雜的模型，這些基底模型可以是具有高變異與低偏誤的複雜模型(自由度高)，或是具有高偏誤與低變異的簡單模型(自由度低)，而集成學習的思維是試通過將這些「基底模型」結合試圖降低整體的「偏誤與變異」

集成學習：裝袋法

裝袋法(bagging)

- 通常選擇相同核心算法的基底模型，同時且相互獨立地學習它們，並在分類的問題中進行投票，而在迴歸問題中進行平均，將各別模型的預測結果彙總在一起，以獲得具有較低「變異」的模型。
- 但實務上難以配適多個獨立的模型，因為這需要取得多個獨立的資料集，因此我們常運用「拔靴法」(bootstrap)樣本良好的「近似統計性質」(代表性和獨立性)來配適近乎獨立的模型。
- 「拔靴法」是蒙地卡羅法的一種應用，它假設已知樣本為母體，利用自身進行多次「取後放回的重複抽樣」，抽出多個樣本子集以估計母體參數



集成學習：裝袋法

- 抽樣的樣本要如何具備良好的統計性質
  - 這些樣本是來自真正潛在的母體分配(通常是未知的)
  - 它們可以被視為來自真實資料分布具有代表性且獨立的樣本，統計上常以相同且獨立分配(independent and identically distributed, i.i.d.)來表示

- 「拔靴法」方式得到的樣本要如何滿足或近似上述兩大假設
  - 原始資料的樣本數大小N要足夠大使其分布狀況近似真實分布的複雜度，滿足樣本的代表性
  - 子樣本集樣本數大小B相比，原始資料的樣本數大小N要相對大，使得「拔靴法」樣本之間沒有太多相關性，滿足樣本的獨立性

集成學習：裝袋法

- 「隨機森林」
  - 是「裝袋法」的一種延伸應用方法，利用拔靴法的子樣本集獨立地配適多個深層的決策樹，以降低預測結果的變異
  - 隨機森林中每棵樹在生長時，會對數據集的「樣本」進行隨機抽樣，同時會對「特徵」作隨機抽樣。一方面保證基底模型的「異質性」，另一方面也能處理有「遺漏值」的資料，其概念是僅挑選沒有 / 較少遺漏的變數建構決策樹（事實上決策樹可部分處理遺漏值狀況）
  - 若某一樣本在多棵決策樹中有較高比例的預測為不良品，則我們最終傾向於預測為不良品

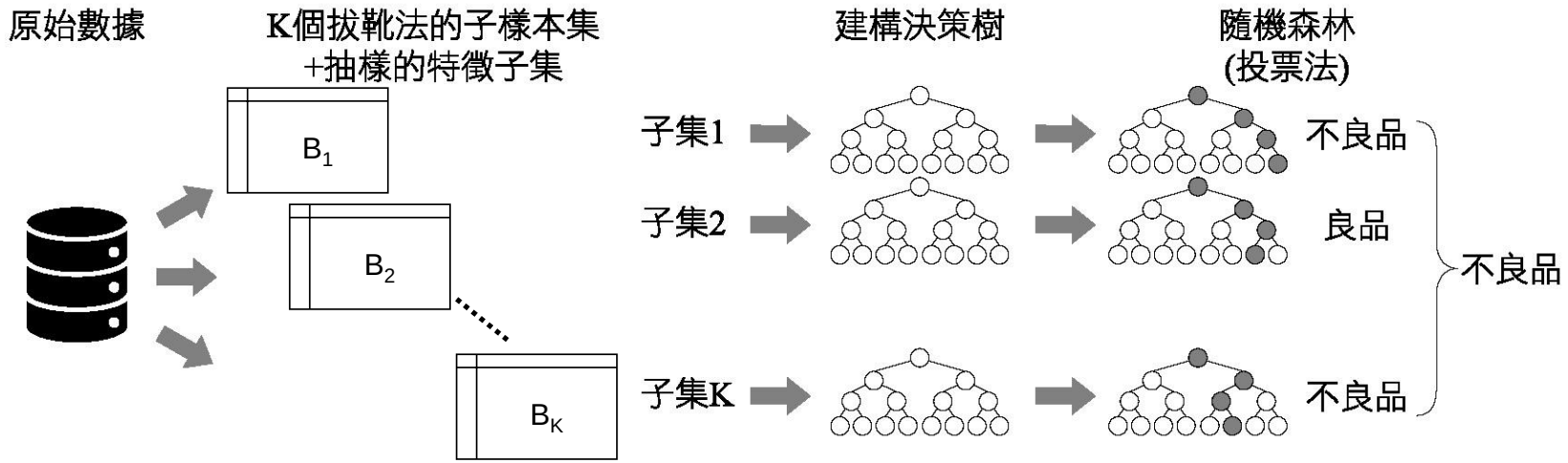
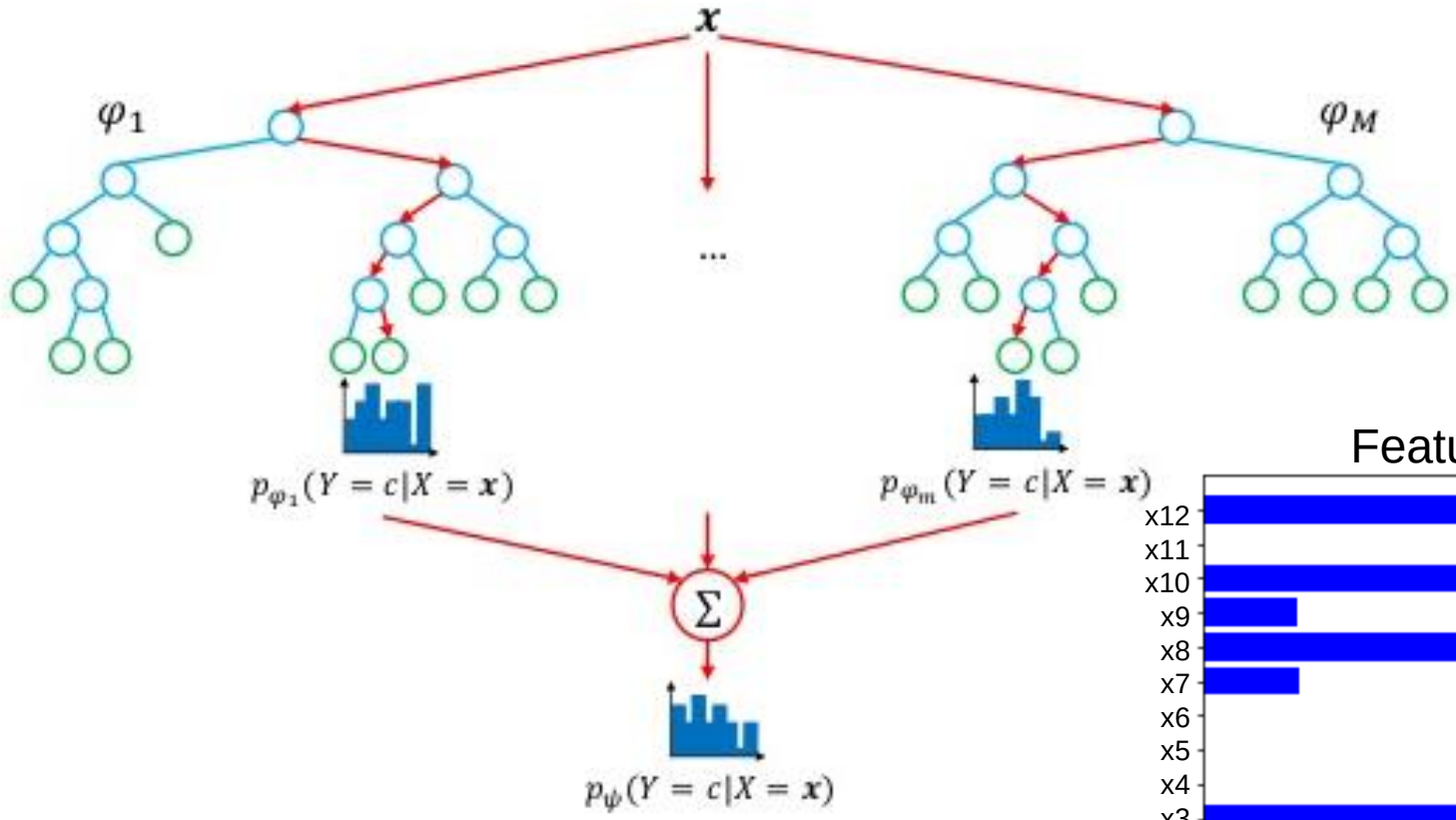
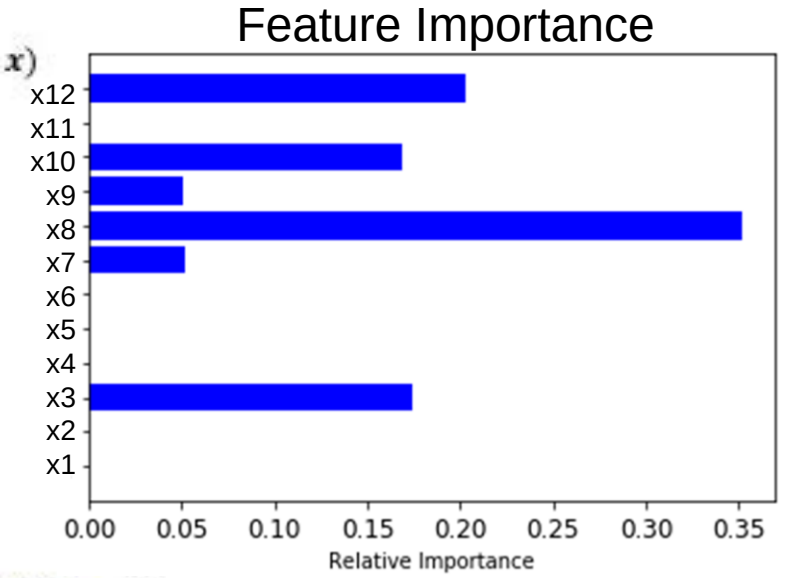


圖 8.9　隨機森林

隨機森林

Random forests





Randomization

- Bootstrap samples
- Random selection of $K \leqslant p$ split variables } Random Forests
- Random selection of the threshold } Extra-Trees

https://www.kdnuggets.com/2017/10/random-forests-explained.html

集成學習：提升法

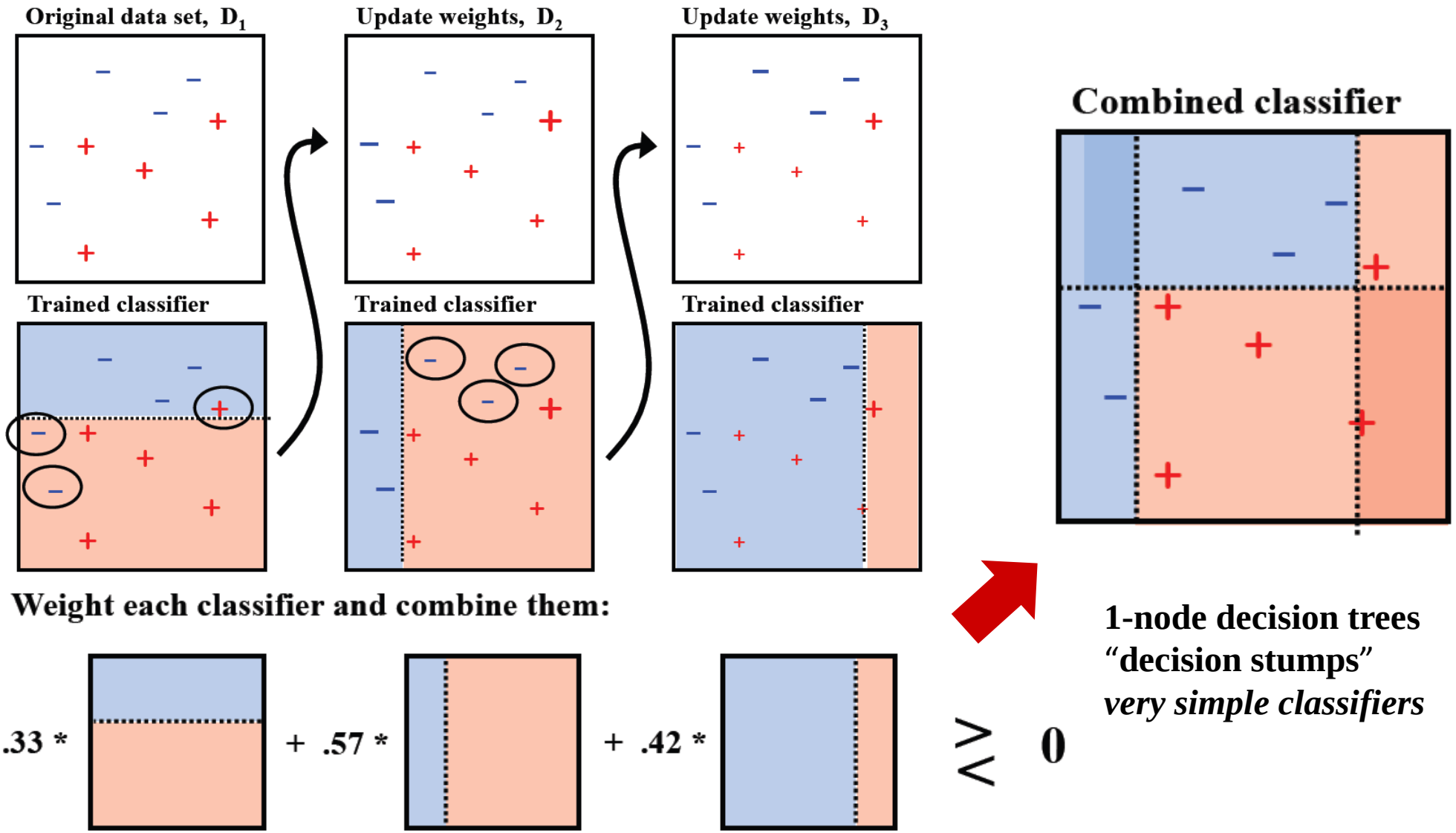
提升法(boosting)

- 通常選擇相同的「基底模型」，以順序性的方式進行學習，並針對每次預測錯誤的樣本增加其權重，之後再在訓練新的模型，最後再將每一回合訓練的模型組合成一個強而穩健的模型
- **裝袋法的思維在於降低預測的「變異」，一般選擇高變異低偏誤的基底模型，而提升法的思維在於降低預測的「偏誤」，一般選擇高偏誤低變異的基底模型**

提升法如何降低偏誤

- 順序性建模
  - 以分類問題為例，提升法以順序性的方式配適多個弱模型，在每次訓練弱模型時，會將先前分類錯誤的樣本加大權重，把更多的注意力集中在錯分的情形。在這樣的迭代下，每個弱模型專注於區別上困難的不同樣本，調整「樣本權重」，逐步地降低整體的「偏誤」

Boosting (eg. Gradient Boosting Machine, GBM)



Alexander Ihler (2012). http://sli.ics.uci.edu/Classes/2012F-273a? action=download&upname=10-ensembles.pdf

集成學習：提升法

提升法如何降低偏誤

- 弱模型各自的權重
  - 除了順序性建模以及對樣本處理方式不同外，其發展的方法中更加入了不同「弱模型各自的權重」，因此可以更廣義地把強模型F以L個弱模型T_l與權重α_l來表示的「加性模型」
  - 加性模型(additive models)：$F(x) = \sum_{l=1}^{L} \alpha_l \cdot T_l(x)$
  - 同時找到所有的「弱模型$T_l(x)$」與其權重α_l是一個困難的最佳化問題，因此提升法將問題轉成以迭代方式逐一找到每次迭代中的最佳解，如公式
  - $F_l(x) = F_{l-1}(x) + \alpha_l \cdot T_l(x)$
  - 並可將損失函數e的最佳化問題表示如公式
  - $\{\alpha_l \cdot T_l\} = \arg\min_{\alpha,T} \sum_{i=1}^{n} e(y_i, F_{l-1}(x) + \alpha \cdot T(x))$

集成學習：提升法

- 「樣本權重的調整」與「弱模型各別權重的調整」為提升法逐步降低偏誤的關鍵
  - 「自適應提升法」(adaboost)以「弱模型的分類誤差」作為自適應(adaptive)調整的依據

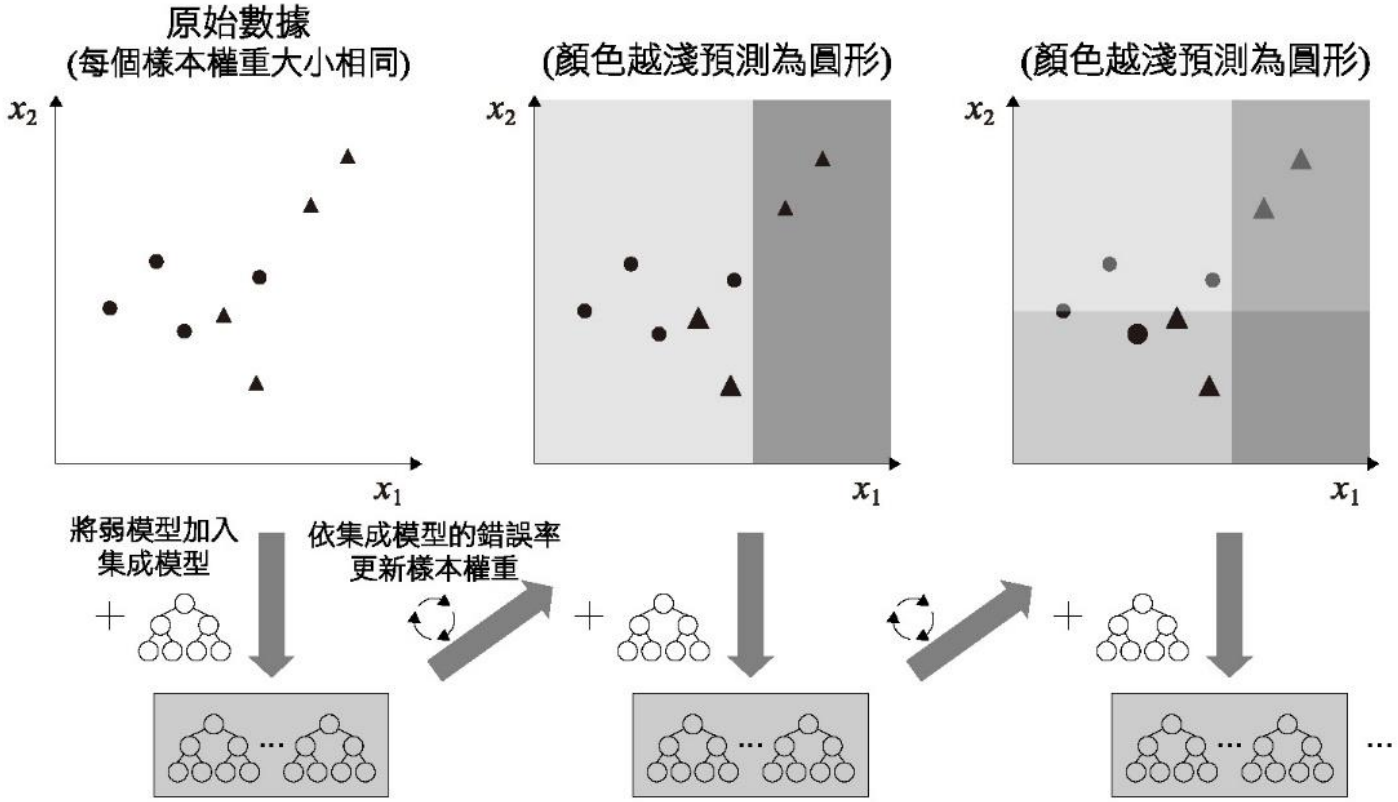


圖 8.10　自適應提升法

自適應提升演算法

表 8.4　自適應提升演算法

演算法　自適應提升法（AdaBoost）

輸入：訓練集$S = (x_1, y_1), \ldots, (x_n, y_n)$ and $y_i \in \{1, -1\}$，迭代次數L

輸出：自適應提升模型$\hat{F}_{ab}$

1. 初始化訓練集S的資料權重：

$$D_1 = (w_{11}, w_{12}, \ldots, w_{1n}),\ w_{1i} = \frac{1}{n},\ i = 1, \ldots, n;$$

2. For $l = 1$ to L
3. 決策樹T_l以樣本權重D_1進行建模，並將其儲存；
4. 計算決策樹T_l的分類誤差：

$$\varepsilon_l = \sum_{i=1}^{n} w_{li} I(T_l(x_i) \neq y_i);$$

5. 計算決策樹T_l的模型權重：

$$\alpha_l = \frac{1}{2}\ln\left(\frac{1-\varepsilon_l}{\varepsilon_l}\right);$$

6. 更新樣本權重：

$$w_{l+1,i} = \frac{w_{li}\exp(-\alpha_l y_i T_l(x_i))}{\sum_{i=1}^{n} w_{li}\exp(-\alpha_l y_i T_l(x_i))},\ i = 1, \ldots, m;$$

$$D_{l+1} = (w_{l+1,1}, w_{l+1,2}, \ldots, w_{l+1,n});$$

7. End
8. 將所有決策樹$\{T_l\}_1^L$以$\{\alpha_l\}_1^L$進行加權平均得到集成模型：

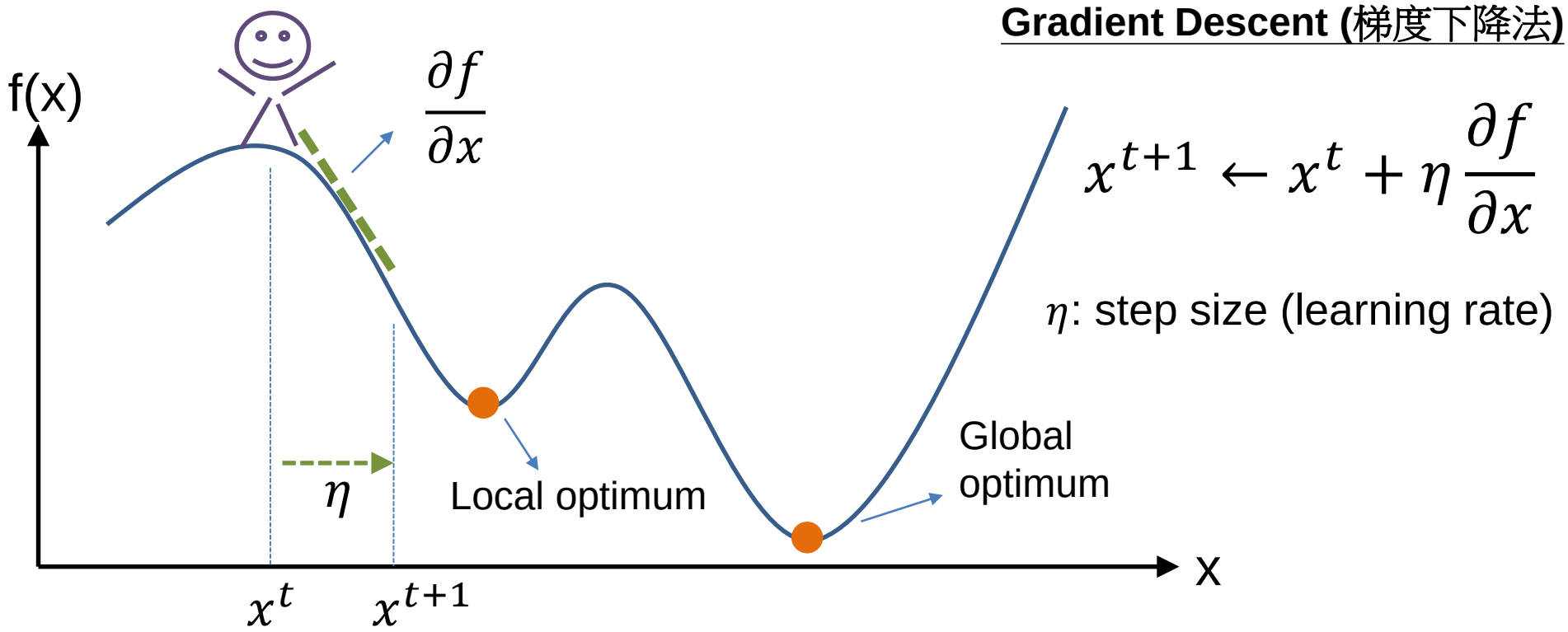
$$\hat{F}_{ab}(x) = sign\left(\sum_{l=1}^{L} \alpha_l T_l(x)\right)$$

9. Return 自適應模型$\hat{F}_{ab}$

集成學習：提升法

梯度提升機(gradient boosting machine, GBM)

- 「梯度提升機」相較於「自適應提升法」的差異就在於逐步最佳化的方式，將前述公式$F_l(x) = F_{l-1}(x) + \alpha_l \cdot T_l(x)$所表達的弱模型改寫成梯度下降(gradient descent)的形式，如下公式，其中權重項α_l可視為一種學習率，而對損失函數微分項的梯度則是預測誤差。
- $F_l(x) = F_{l-1}(x) - \alpha_l \cdot \nabla_{F_{l-1}} e(y, F_{l-1}(x))$



Gradient Descent (梯度下降法)

$$x^{t+1} \leftarrow x^t + \eta \frac{\partial f}{\partial x}$$

η: step size (learning rate)

- 梯度提升機(gradient boosting machine, GBM)
  - 若損失函數以「最小平均誤差」(mean squared error, MSE)，則如下公式計算
    - $-\nabla_{F_{l-1}} e\left(y, F_{l-1}(x)\right) = \frac{d\left(y-F_{l-1}(x)\right)^2}{dF_{l-1}(x)} = 2\left(y - F_{l-1}(x)\right)$
  - 此梯度的微分結果為兩倍的殘差，也就是我們所建立的下一個弱模型其實是基於上一個模型預測的殘差去進行配適的，因此損失函數的梯度也被稱之為「偽殘差」(pseudo-residuals)

集成學習：提升法

- 梯度提升機(gradient boosting machine, GBM)

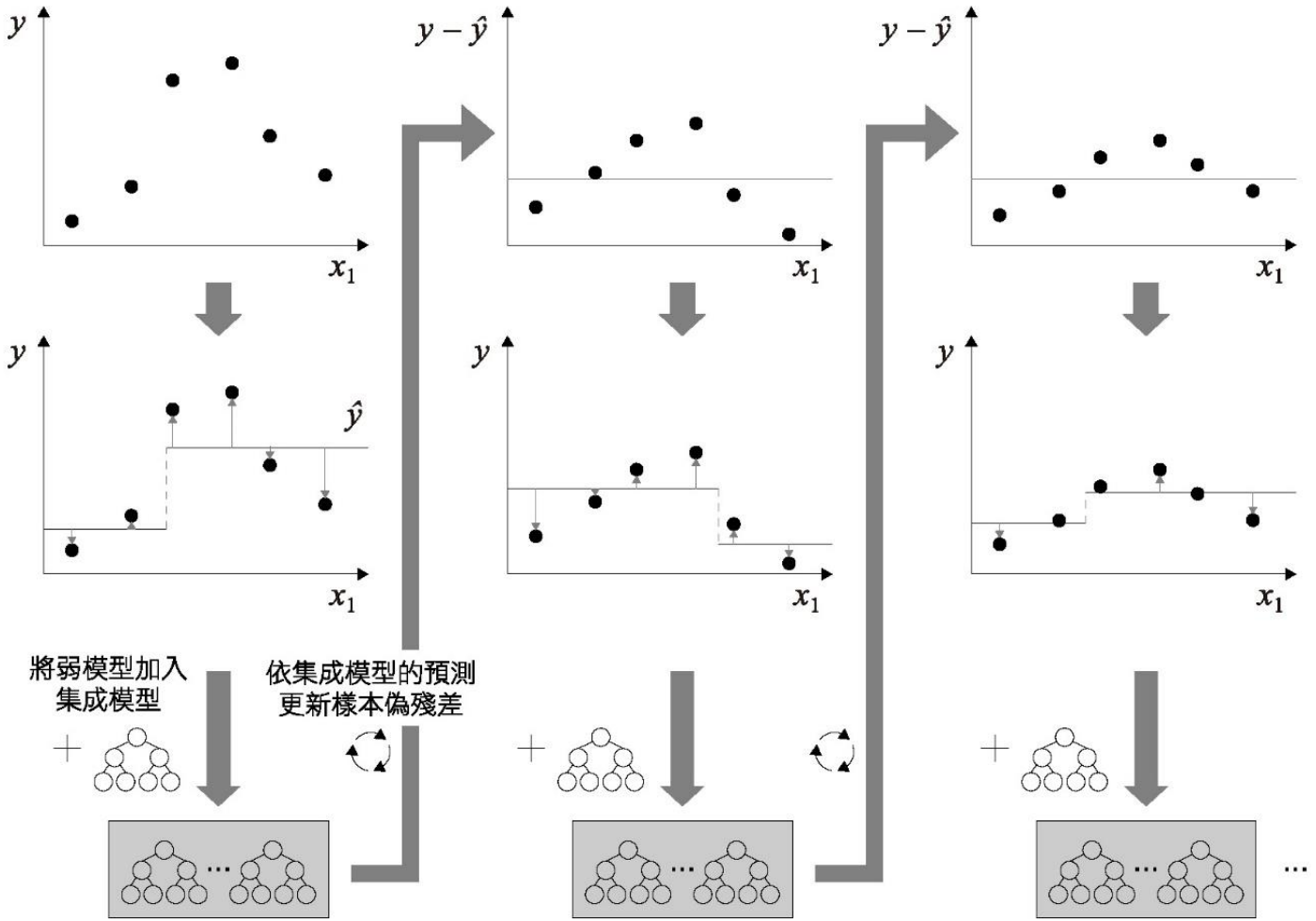


圖 8.11　梯度提升機

梯度提升機演算法(gradient boosting machine, GBM)

表 8.5　梯度提升機演算法

演算法 梯度提升機（GBM）

輸入：訓練集$S=(x_1,y_1),\ldots,(x_n,y_n)$，可微的損失函數$e(y,F(x))$，迭代次數$L$

輸出：梯度提升機模型$\hat{F}_{gbm}$

1. 找出初始化模型的權重常數：

$$F_0(x)=\arg\min_{\alpha}\sum_{i=1}^{n}e(y_i,\alpha)\ ;$$

2. For $l=1$ to L
3. 計算偽殘差（pseudo-residuals）：

$$r_{il}=-\left[\frac{de(y_i,F(x_i))}{dF(x_i)}\right]_{F(x)-F_{l-1}(x)},\ \text{for } i=1,\ldots,n\ ;$$

4. 決策樹T_l對偽殘差r_l進行建模，新訓練集為$S_l=(x_1,r_{1l}),\ldots,(x_n,r_{nl})$）；
5. 找出決策樹T_l的模型權重（一維度的最佳化問題）：

$$\alpha_l=\arg\min_{\alpha}\sum_{i=1}^{n}e(y_i,F_{l-1}(x_i)+\alpha T_l(x_i))\ ;$$

6. 更新集成模型：

$$F_l(x)=F_{l-1}(x)+\alpha_l T_l(x)$$

7. End
8. 最終的集成模型：

$$\hat{F}_{gbm}(x)=F_L(x)\ ;$$

9. Return 梯度提升機模型$\hat{F}_{gbm}$

集成學習：堆疊法與混合法

- 堆疊法(stacking)與混合法(blending)
  - 概念是學習多個異質的基底模型(base learner)，並將這些基底模型的多組預測結果視為新特徵，接著使用一個元模型(meta model)建模訓練來組合它們。舉例來說，我們可以選擇決策樹、羅吉斯迴歸和支持向量機作為基底模型，並使用類神經網路作為元模型
  - 實際上，堆疊法與混合法十分相似，差異在於交叉驗證的步驟，也是此二方法中重要的一個步驟

堆疊法

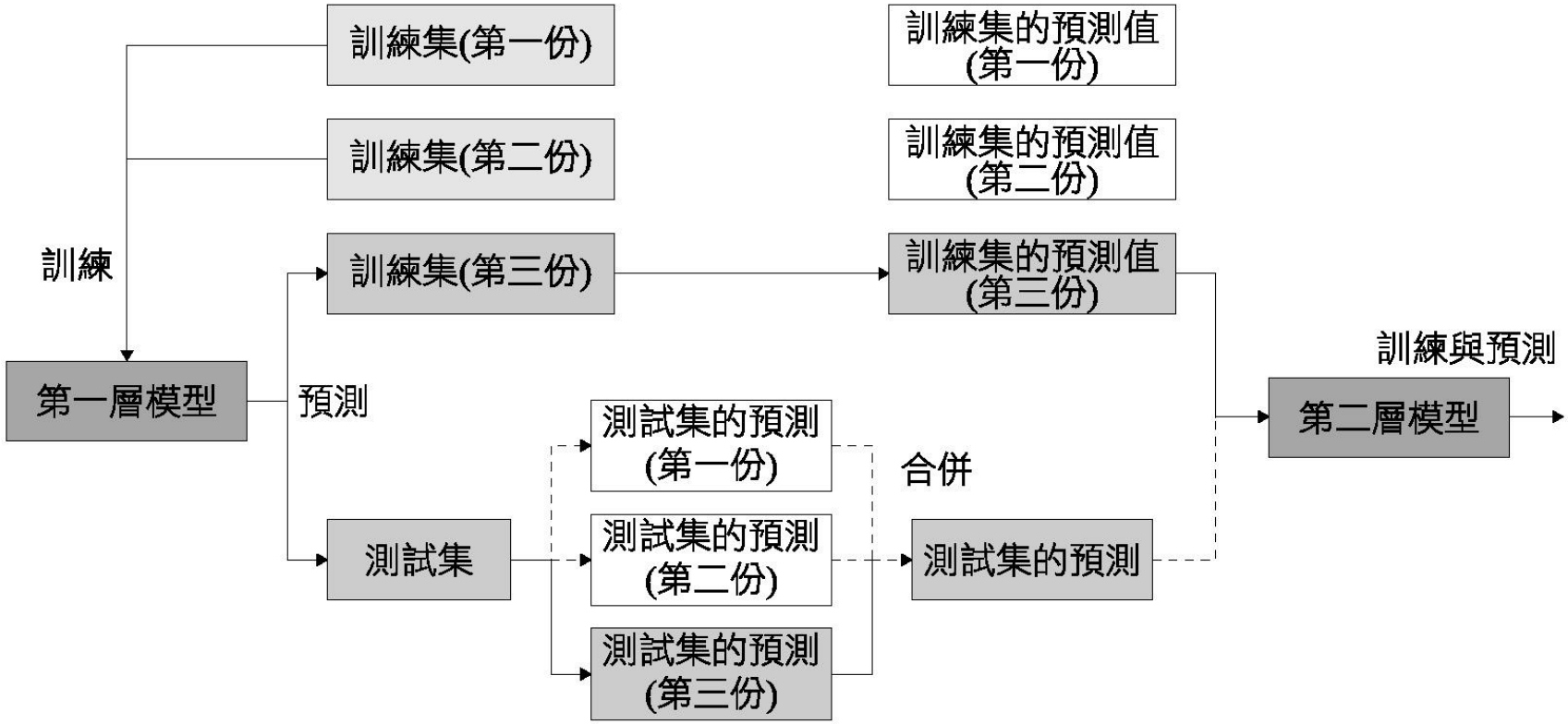

訓練集(第一份)
訓練集(第二份)
訓練集(第三份)
訓練
第一層模型
預測
測試集
測試集的預測(第一份)
測試集的預測(第二份)
測試集的預測(第三份)
合併
測試集的預測
訓練集的預測值(第一份)
訓練集的預測值(第二份)
訓練集的預測值(第三份)
訓練與預測
第二層模型

圖 8.12　堆疊法

堆疊法演算法

表 8.6　堆疊法演算法

演算法　堆疊法（Stacking）

輸入：訓練集$S = (x_1, y_1), \ldots, (x_n, y_n)$，迭代次數$L$

輸出：堆疊模型$\hat{F}_{stack}$

1. For $l = 1$ to L
2. 　從S建構L個不同算法的基底模型$\{h_l\}_1^L$
3. End
4. For $i = 1$ to n
5. 　將基底模型產生的預測值作為新的特徵：

$$S_h = (x_i', y_i)，\text{其中 } x_i' = \{h_1(x_i), \ldots, h_L(x_i)\}；$$

6. End
7. 從S_h建構一個元模型（meta-model），得到最終的集成模型:

$$\hat{F}_{stack}(x) = F(\{h_l\}_1^L)$$

8. Return 堆疊模型$\hat{F}_{stack}$

混合法

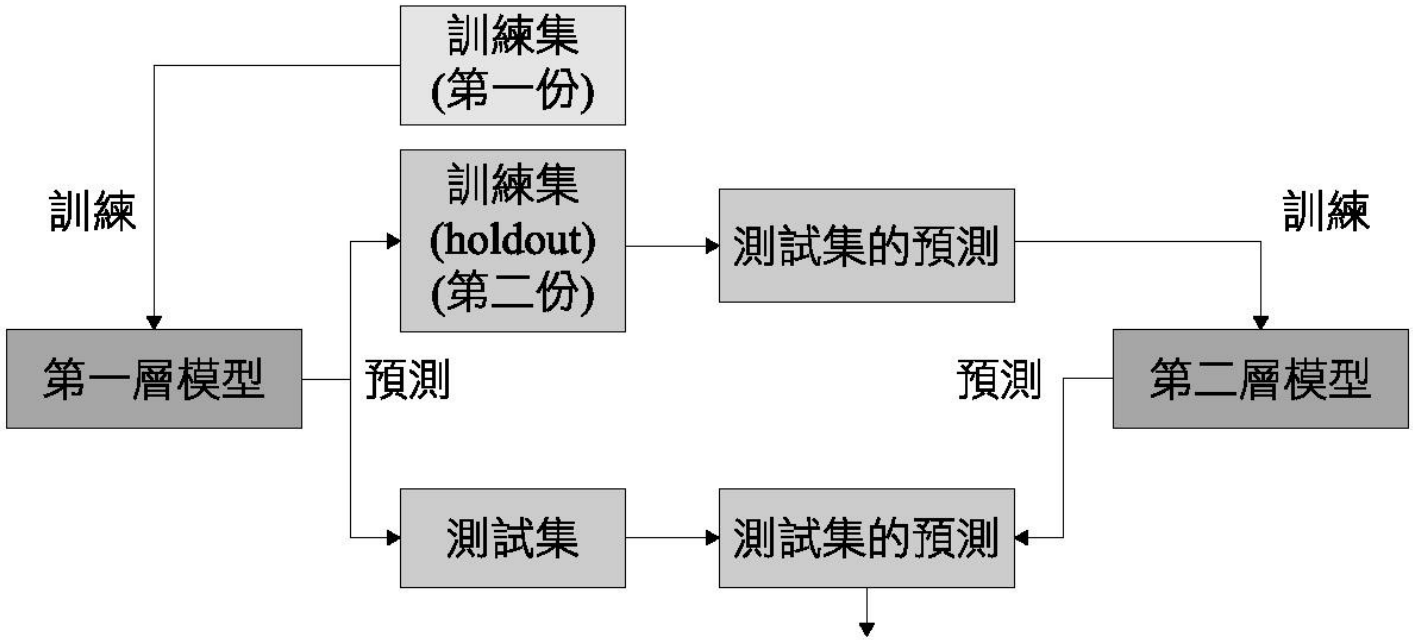

訓練集
(第一份)
訓練集
(holdout)
(第二份)
測試集的預測
訓練
訓練
第一層模型
預測
預測
第二層模型
測試集
測試集的預測

圖 8.13　混合法

集成學習：堆疊法與混合法

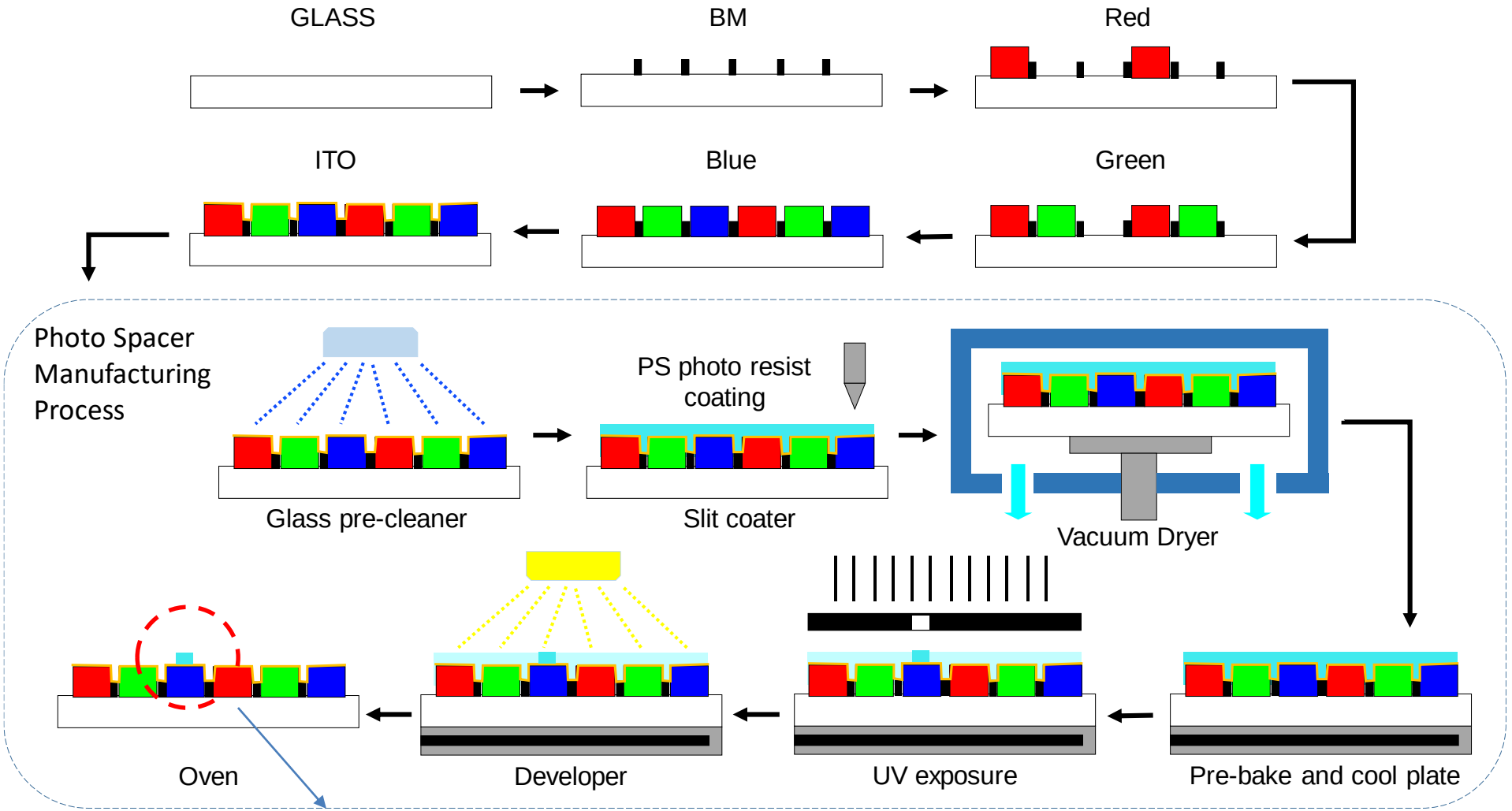
- 堆疊法、混合法與裝袋法、提升法不同之處
  - 前二者將異質性放在不同演算法的組合上，而後二者則是將異質性放在樣本權重上
  - 前二者是建構元模型對基底模型進行彙總，其由第二層模型中挑選第一層的數個模型中較有價值的資訊，而後二者則使用確定性算法(投票、加權平均等)結合基底模型的預測結果

實證案例：彩色濾光片工程參數篩選與膜厚預測

- TFT-LCD Manufacturing (https://www.youtube.com/watch?v=JzD-CVrtVyg)

彩色濾光片工程參數篩選與膜厚預測

Color Filter



Thickness of PS prediction

Lee and Tsai (2019)

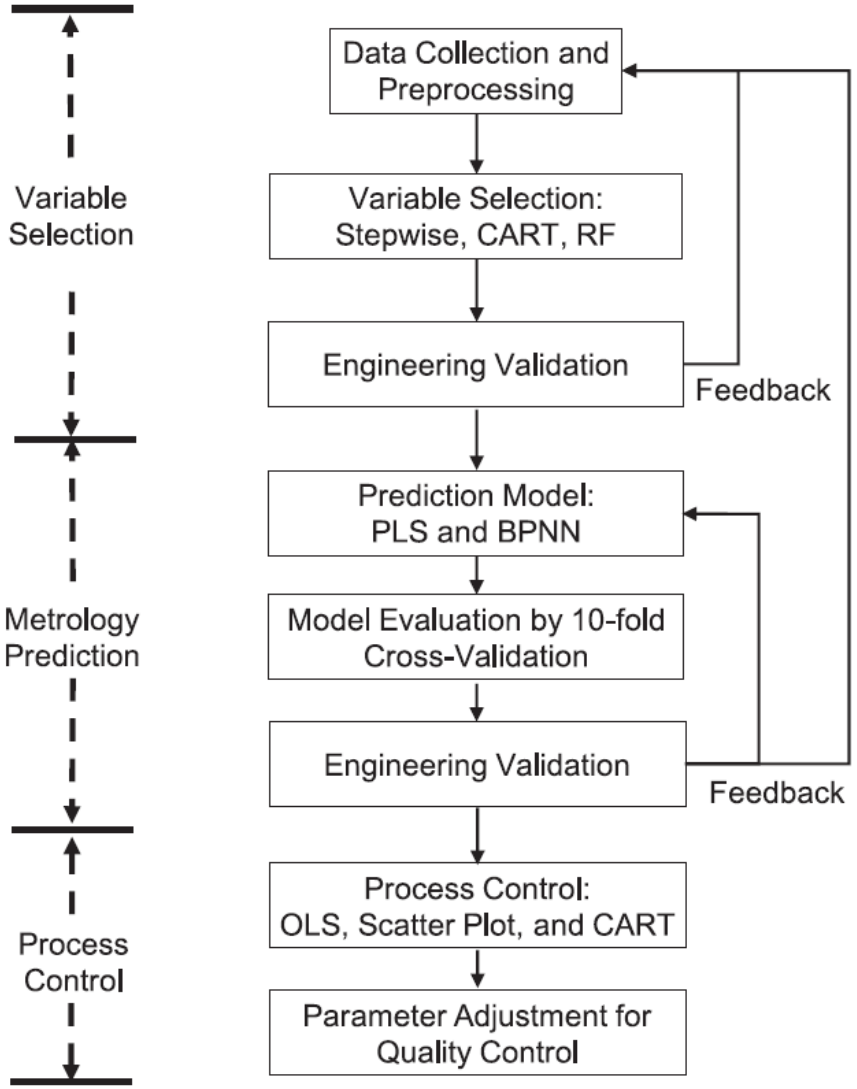
彩色濾光片工程參數篩選與膜厚預測

Data Source

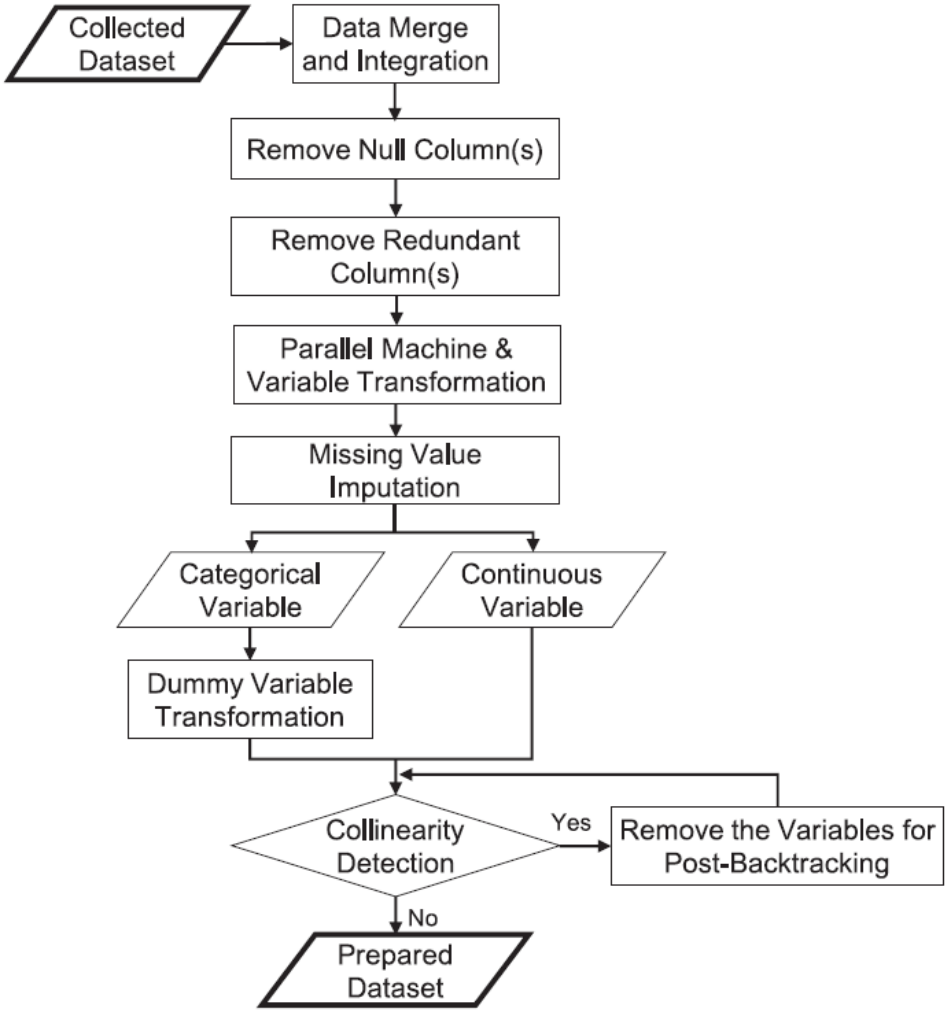
廠商 (Vendor)	Fab A	備註
收集對象 (Target)	Color filter process BM →R →G →B →UV Asher →ITO→MVA →PS	
資料來源 (Data Source)	MES and engineering data center (EDC) database	
資料收集時間(Time)	2020/10	
樣本數 (Sample size)	1000	Training:Testing = 8:2 (randomly sampling)
獨立變數(Independent Variable)	20000 factors Continuous: Categorical:	Null column: Identical column: Category w/ 1 sample: Missing value:
依變數 (Dependent Variable)	Category: Pass (-1) or Fail (1)	Pass:Fail: 700:300

彩色濾光片工程參數篩選與膜厚預測

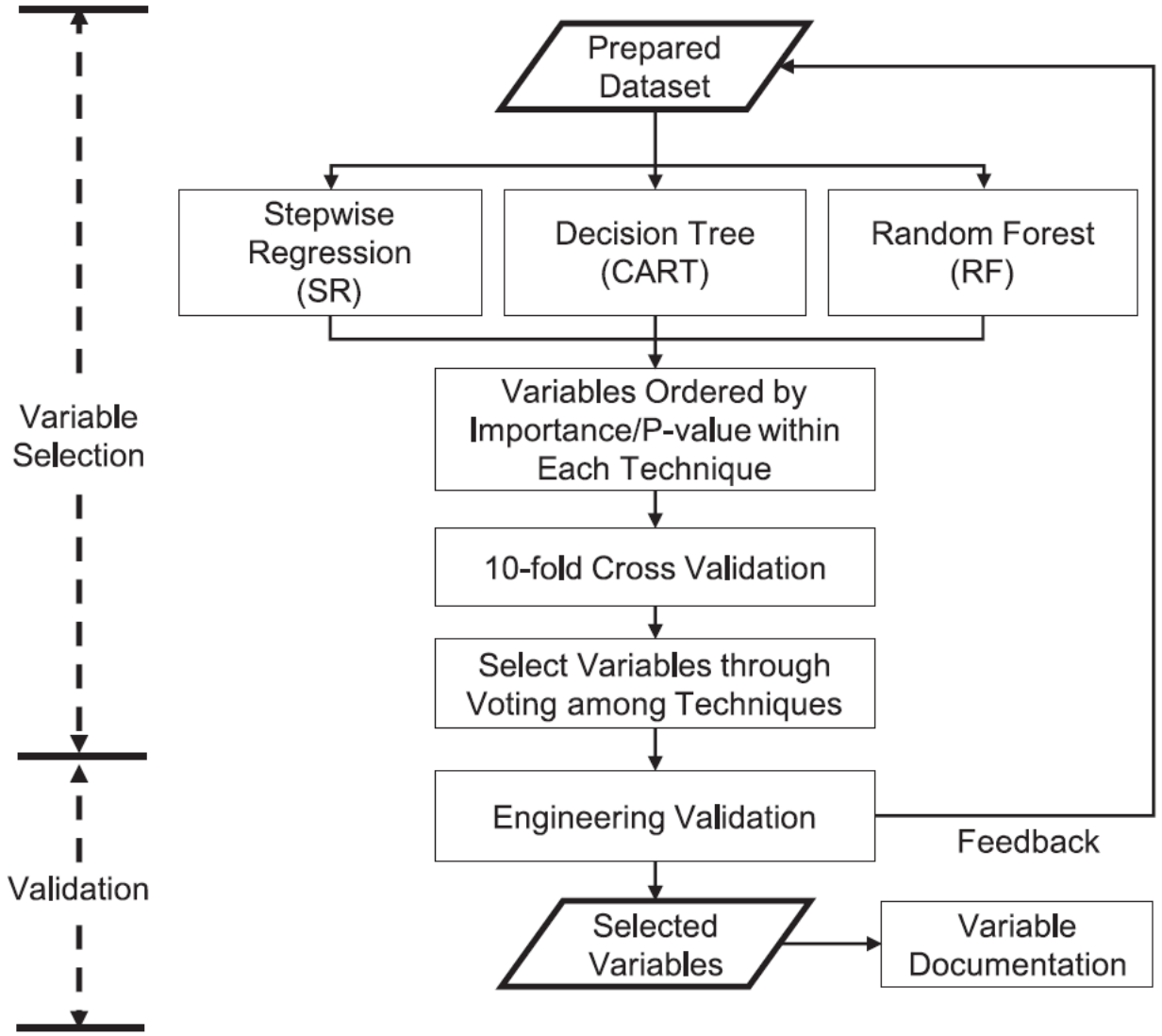
Analytics Framework



Data Preprocessing



Feature Selection



彩色濾光片工程參數篩選與膜厚預測

Feature Selection

- 10 out of 20,000 variables

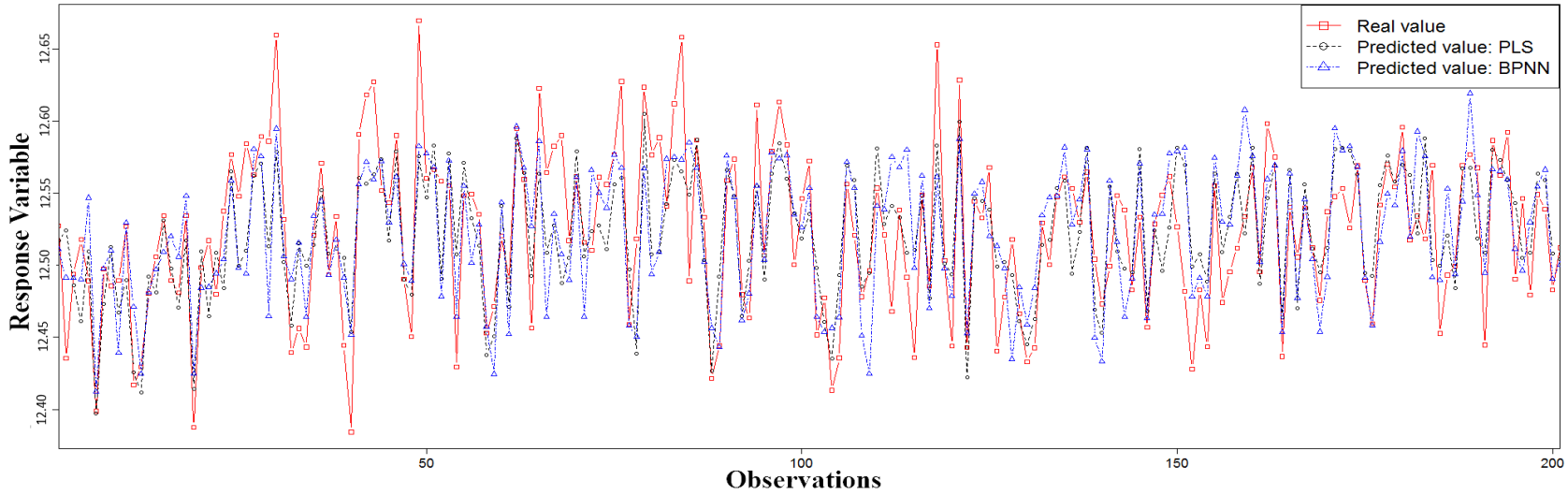
Variables	SR	CART	RF	Votes
X_097	◯	◯	◯	3
X_034	◯		◯	2
X_126		◯	◯	2
X_019		◯	◯	2
X_005			◯	1
…	…	…	…	…

Prediction Performance

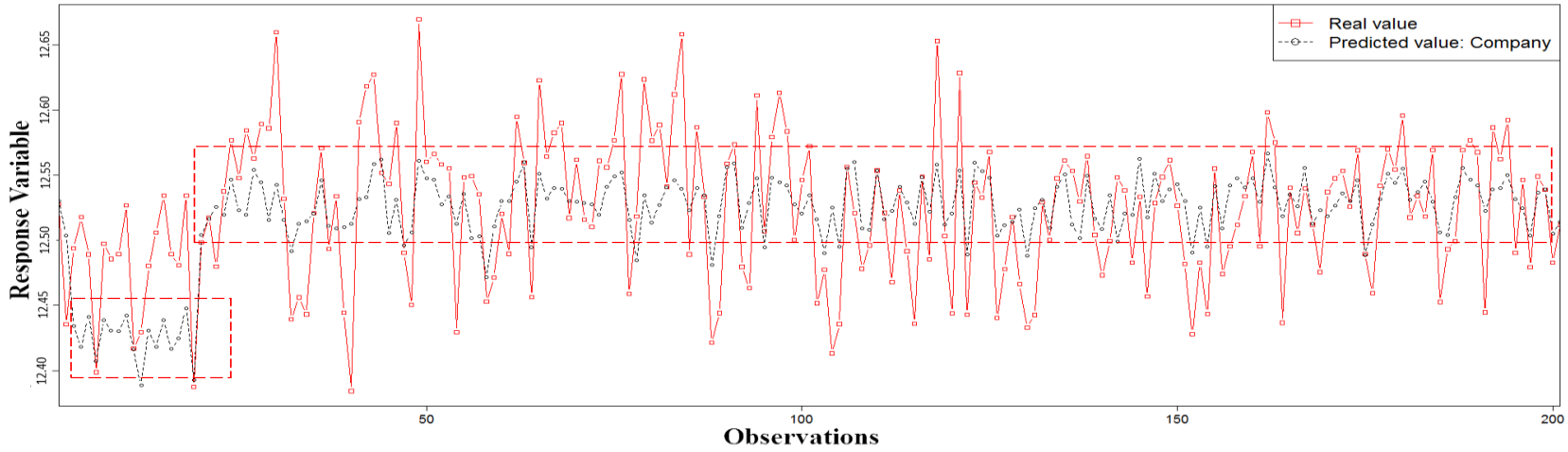
	R-squared	MSE	# of selected var.
As-Is (PLS)	0.392	8.15E-05	373
PLS	0.561	5.57E-05	10
BPNN	0.760	7.02E-05	10

彩色濾光片工程參數篩選與膜厚預測

- Proposed BPNN & PLS

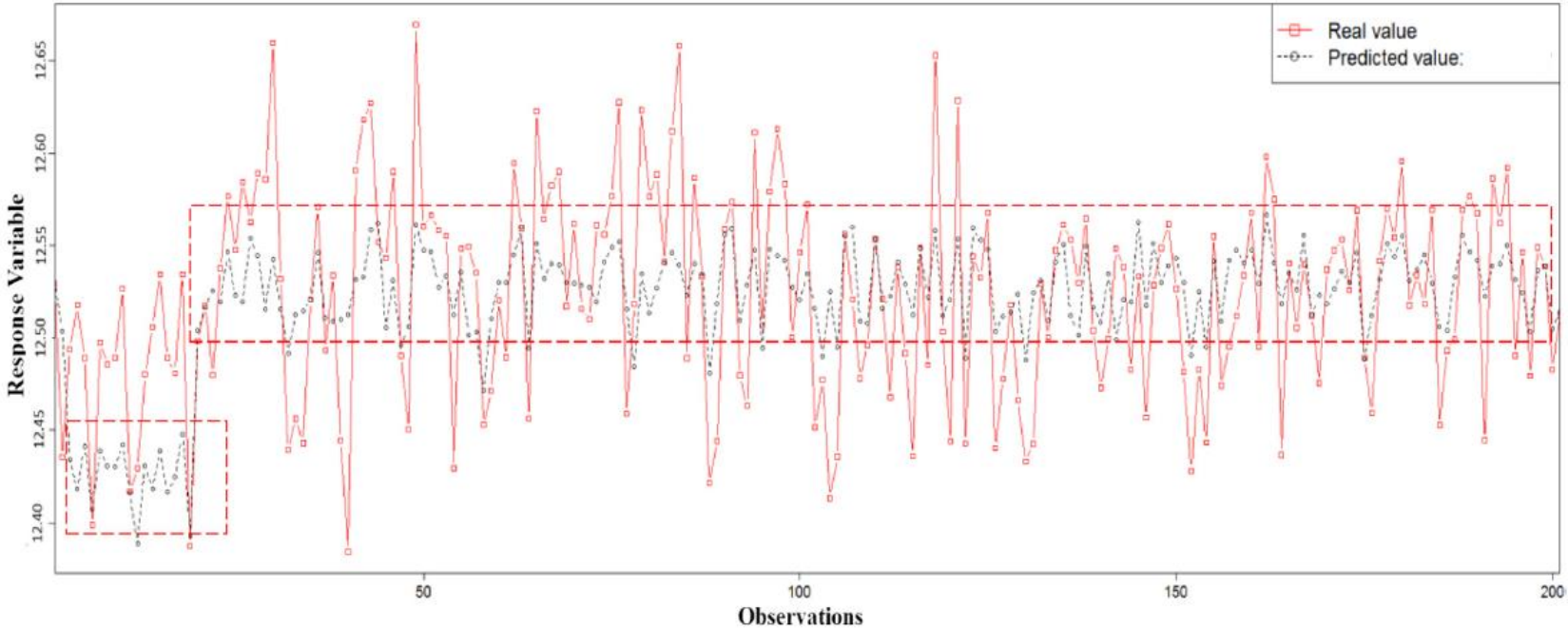


- As-Is method



Myth

- Prediction results with small MSE, but…



- Too many variables in the DS/ML model/ No identify the important variable
- Outlier removal (outlier could give the variance to the model too) → generate outlier for addressing class imbalance
- Loss function: MSE → Cross Entropy or MAE
- Log/exp transform
- Weak prediction model
- Small data (change LSTM to GRU)
- Time series decomposition

實證案例：彩色濾光片工程參數篩選與膜厚預測

- 收集製程數據時特徵數量往往非常多，透過數據科學可以來挑選重要參數，另一方面找出有物理意義且具解釋性的規則來協助現場改善良率
- 使用分類與迴歸樹可以簡單地釐清重要因子與特徵排序，也能考慮特徵之間的交互作用

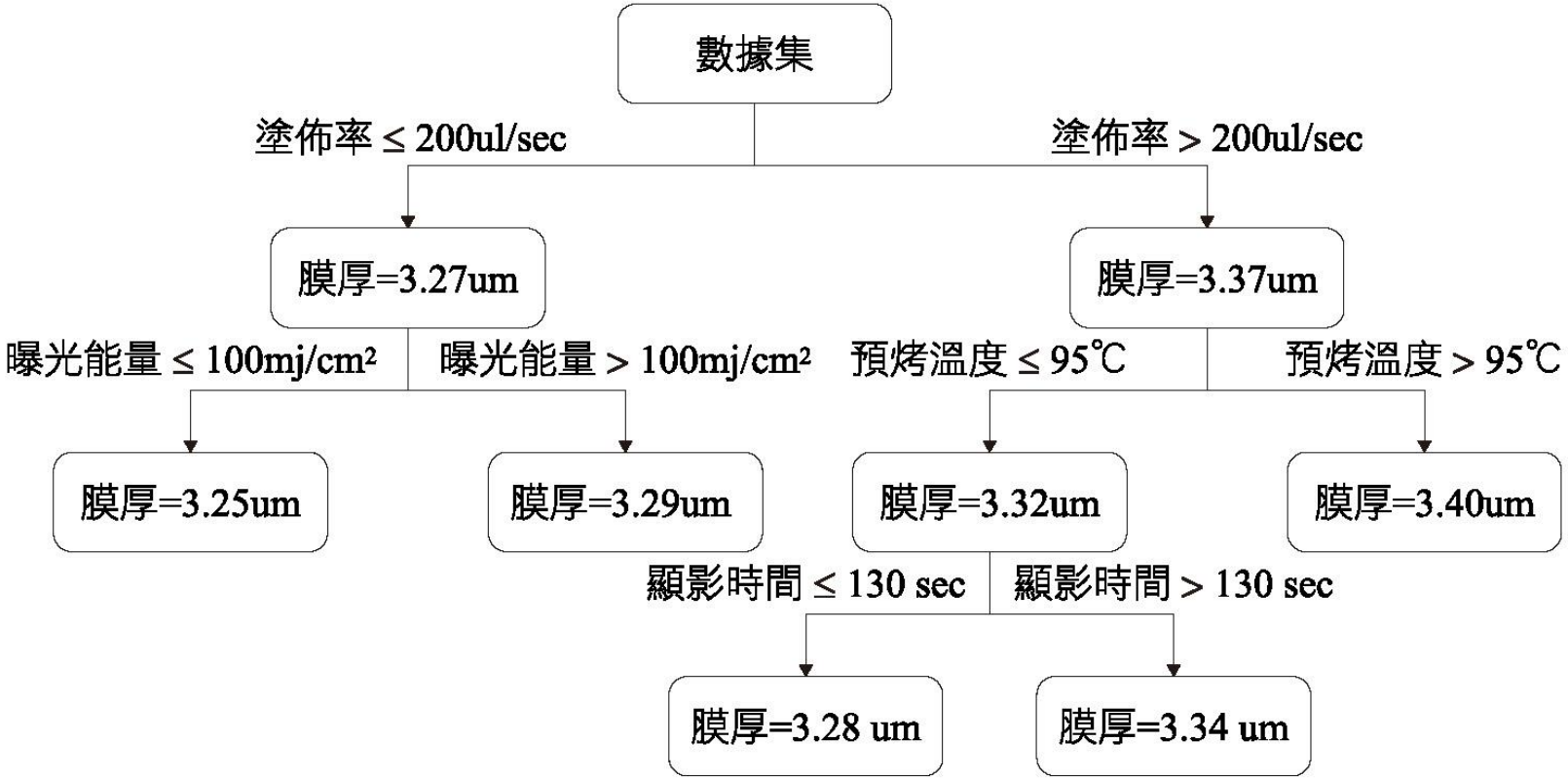


圖 8.15　分類與迴歸樹對於感光間隔物的分析

製程調控

- 決策樹可以萃取出重要工程規則協助製程改善或故障排除
  - 塗佈率大於200 時，膜厚預測約3.37μm；
  - 塗佈率大於200 且預烤溫度大於95 時，膜厚預測約3.40μm；
  - 塗佈率大於200 且預烤溫度小於95 且顯影時間大於130 時，膜厚預測約3.34μm。

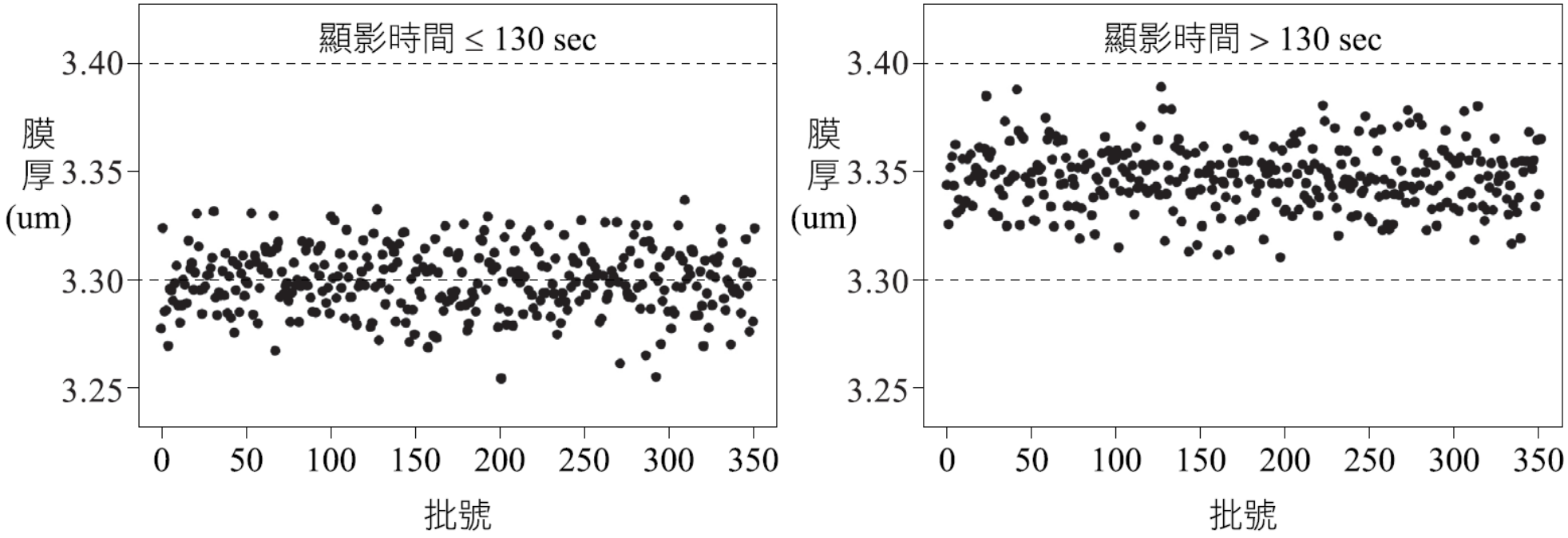
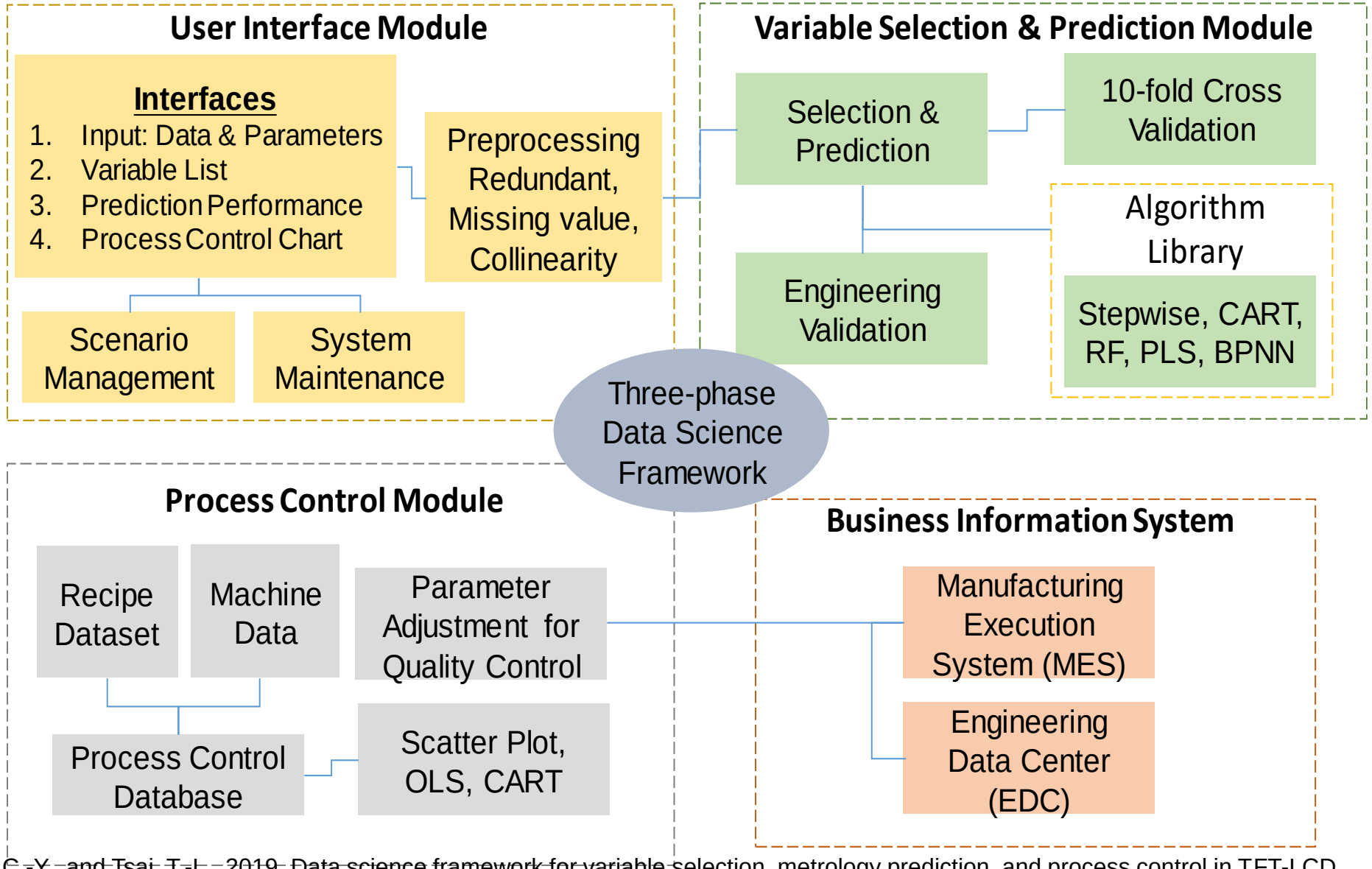


圖 8.16　製程調控與改善

彩色濾光片工程參數篩選與膜厚預測



Lee, C.-Y., and Tsai, T.-L., 2019. Data science framework for variable selection, metrology prediction, and process control in TFT-LCD manufacturing. Robotics and Computer-Integrated Manufacturing, 55, 76-87.

決策樹與集成學習

- 從本質上剖析集成學習是如何藉由「異質性」與「多個模型的彙總」有效地降低預測的「偏誤與變異」
- 「裝袋法」、「提升法」與「堆疊法」的統計思維以及演算法架構
  - 「裝袋法」以拔靴法抽樣增加異質性，並用投票或平均彙總以降低模型變異
  - 「提升法」則藉由改變錯分樣本的權重，透過順序性建模降低模型偏誤
  - 「堆疊法與混合法」則藉由元模型對多種基底模型進行彙總
  - 另一方面，運用集成學習同時也需付出對應的代價：多個超參數的調整、喪失部分模型解釋性、資料洩漏等議題，這是為了提高預測準確度（複雜度、彈性）所可能遭遇的挑戰。

表 8.7　集成方法的比較

方法＼特性	數據處理	目標	優點	缺點
裝袋法	拔靴法抽樣	降低變異	1. 可平行運算 2. 可處理有遺漏值的資料	當特徵維度過高時，重要特徵可能不易被挑選
提升法	加大錯分樣本權重	降低偏誤	1. GBM 運算效率高 2. GBM 損失函數的選擇彈性大	1. 不可平行運算 2. 對離群值較為敏感
堆疊法與混合法	拔靴法或取後不放回抽樣	降低變異	可結合多個不同的基底模型	模型選擇與超參數的複雜度高，不易訓練

Thanks for your attention

NTU Dept. of Information Management
name: 李家岩 (FB: Chia-Yen Lee)
phone: 886-2-33661206
email: chiayenlee@ntu.edu.tw
web: https://polab.im.ntu.edu.tw/

Appendix

- LightGBM
  - https://medium.com/@jimmywu0621/%E6%B1%BA%E7%AD%96%E6%A8%B9%E7%B3%BB%E5%88%97-lightgbm%E6%A8%A1%E5%9E%8B%E7%90%86%E8%AB%96-96ce38ea8940
- Basic Aggregation Models
  - https://www.ycc.idv.tw/ml-course-techniques_4.html