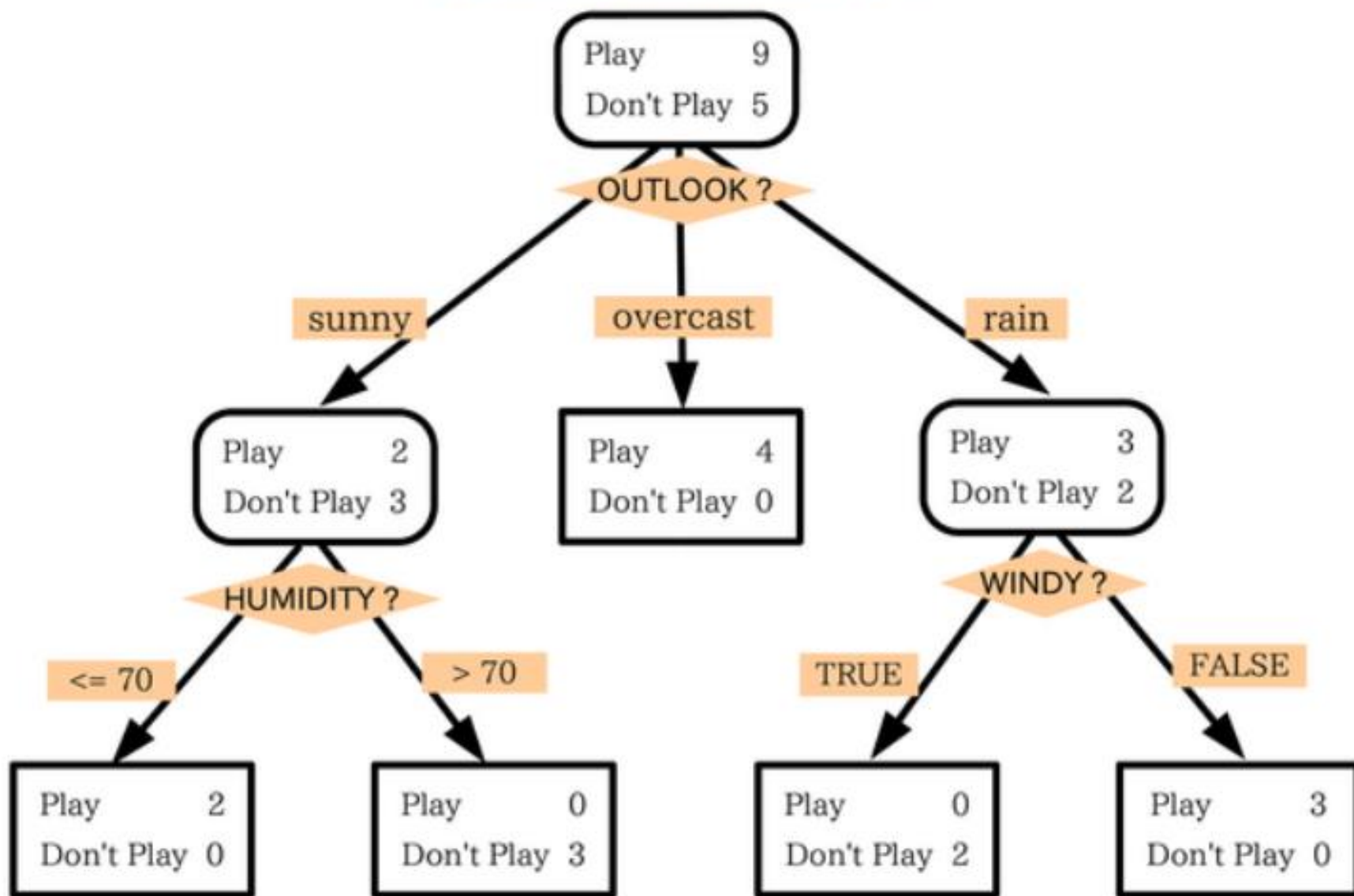


Xgboost & LightGBM

Content:

- 分類樹: 熵(信息增益)、吉尼不純度
- 迴歸樹: CART
- Boosting
- Xgboost
- LightGBM

Dependent variable: PLAY



Day	Outlook	Temperature	Humidity	Wind	PlayT
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

決策樹:三個類型

- 分類樹
- 回歸樹
- CART (Classification And Regression Trees)

- 如何分割?如何決定根結點?

熵(Entropy) & 吉尼不純度 (Gini impurity) & 均方差(MSE)

離散問題:熵(Entropy)

定義:接受的每條消息中，包含的資訊信息量的平均

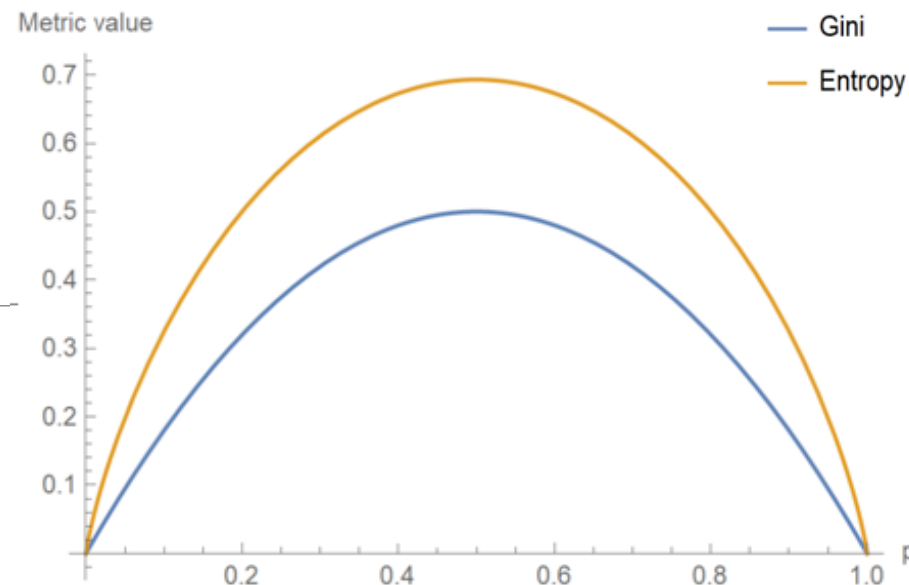
$$H(X) = E[I(X)] = E[-\ln(P(X))]$$

P 為 X 的pmf，而 $I(X)$:信息本體，為 X 的資訊量，是個隨機變數

有限樣本:
$$H(X) = \sum_i P(x_i) I(x_i) = - \sum_i P(x_i) \log_b P(x_i)$$

$p_i = 0$ 時，對於一些 i 值，對應的被加數 $0 \log_b 0$ 的值將會是0，這與極限一致。

$$\lim_{p \rightarrow 0^+} p \log p = 0。$$



英語有**26**個字母，假如每個字母在文章中出現次數平均的話，每個字母的訊息量為：

$$I_e = -\log_2 \frac{1}{26} = 4.7$$

漢字常用的有**2500**個，假如每個漢字在文章中出現次數平均的話，每個漢字的資訊量為：

$$I_e = -\log_2 \frac{1}{2500} = 11.3$$

信息增益

定義: 表示在一個條件下，信息不確定性減少的程度

公式: $\text{Information Gain} = \text{IG}(T,a) = H(T) - H(T|a)$

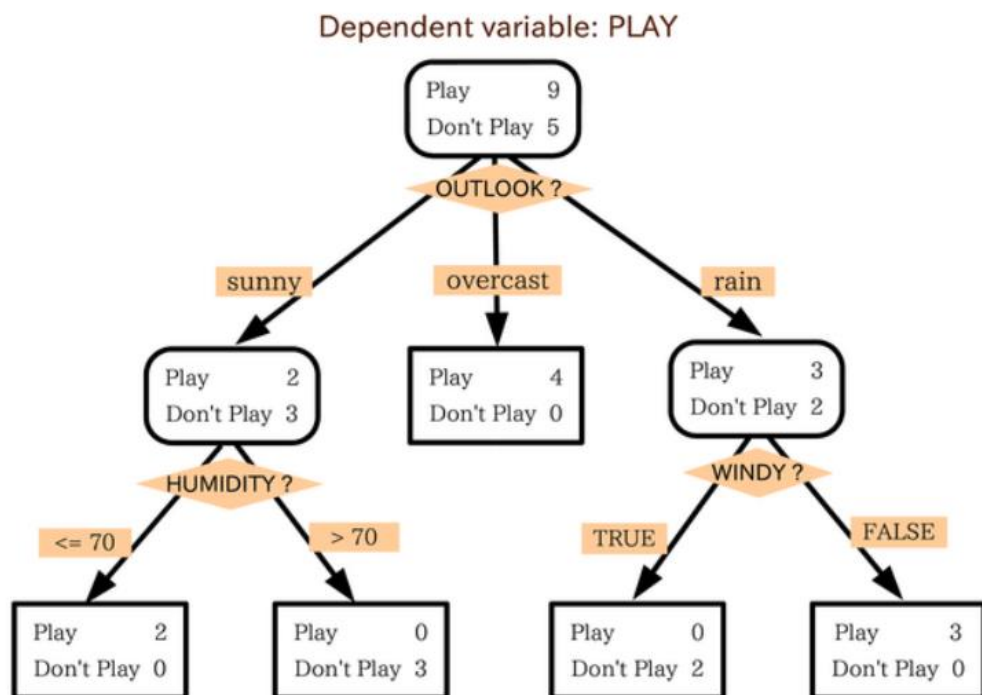
根節點流程:

1. 計算entropy = $H(T) = -(p) \cdot \log(p) - (1-p) \cdot \log(1-p)$ (二元輸出屬性)
2. 分別算根結點其信息增量
 $\text{Information Gain} = \text{IG}(T,a) = H(T) - H(T|a)$
3. 信息增量最大原則: 選 $\text{IG}(T,a) \text{ Max}$ 為根結點
4. 子節點重複上述步驟

ID3、C4.5演算法都是以信息增益下去遞迴處理

目標:是否出去玩

Day	Outlook	Temperature	Humidity	Wind	PlayT
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



1 計算熵

$$\text{Entropy}(S) = -(9/14) \cdot \log(9/14) - (5/14) \cdot \log(5/14) = 0.940$$

(以2為底)

2 計算各信息增益，以**Wind**作為根節點

Entropy(Weak)

$$= -(6/8) \cdot \log(6/8) - (2/8) \cdot \log(2/8)$$

$$= 0.811$$

Entropy(Strong)

$$= -(3/6) \cdot \log(3/6) - (3/6) \cdot \log(3/6)$$

$$= 1.0$$

Gain(Wind)

$$= \text{Entropy}(S) - (8/14) \cdot \text{Entropy}(\text{Weak}) - (6/14) \cdot \text{Entropy}(\text{Strong})$$

$$= 0.940 - (8/14) \cdot 0.811 - (6/14) \cdot 1.0 = 0.048$$

Day	Outlook	Temperature	Humidity	Wind	PlayT
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

以**Humidity**作為根節點

$\text{Entropy}(\text{High})=0.985$; $\text{Entropy}(\text{Normal})=0.592$

$\text{Gain}(\text{Humidity})=0.940-(7/14)*\text{Entropy}(\text{High})-(7/14)*\text{Entropy}(\text{Normal})=0.151$

以**Outlook**作為根節點：

$\text{Entropy}(\text{Sunny})=0.971$; $\text{Entropy}(\text{Overcast})=0.0$; $\text{Entropy}(\text{Rain})=0.971$

$\text{Gain}(\text{Outlook})=0.940-(5/14)*\text{Entropy}(\text{Sunny})-(4/14)*\text{Entropy}(\text{Overcast})-(5/14)*\text{Entropy}(\text{Rain})=0.247$

以**Temperature**作為根節點：

$\text{Entropy}(\text{Cool})=0.811$; $\text{Entropy}(\text{Hot})=1.0$; $\text{Entropy}(\text{Mild})=0.918$

$\text{Gain}(\text{Temperature})=0.940-(4/14)*\text{Entropy}(\text{Cool})-(4/14)*\text{Entropy}(\text{Hot})-(6/14)*\text{Entropy}(\text{Mild})=0.029$

3 信息增益最大的原則

$\text{Gain}(\text{Wind})=0.048$

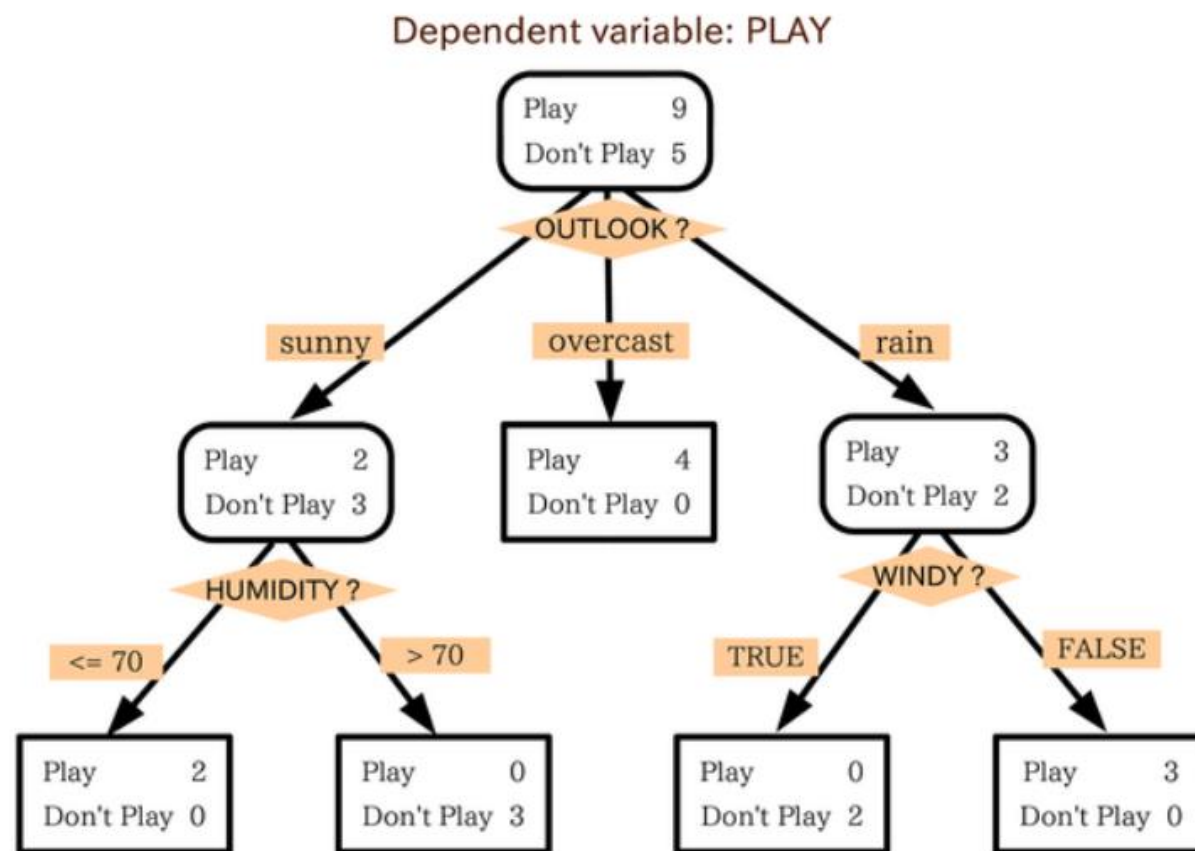
$\text{Gain}(\text{Humidity})=0.151$

$\text{Gain}(\text{Outlook})=0.247$

$\text{Gain}(\text{Temperature})=0.029$

選Outlook為根節點。

4 子節點重複上面的步驟



吉尼不純度 (Gini impurity)

定義: 反映了從數據集中隨機抽取兩個樣本，其類別標記不一致的機率

公式:

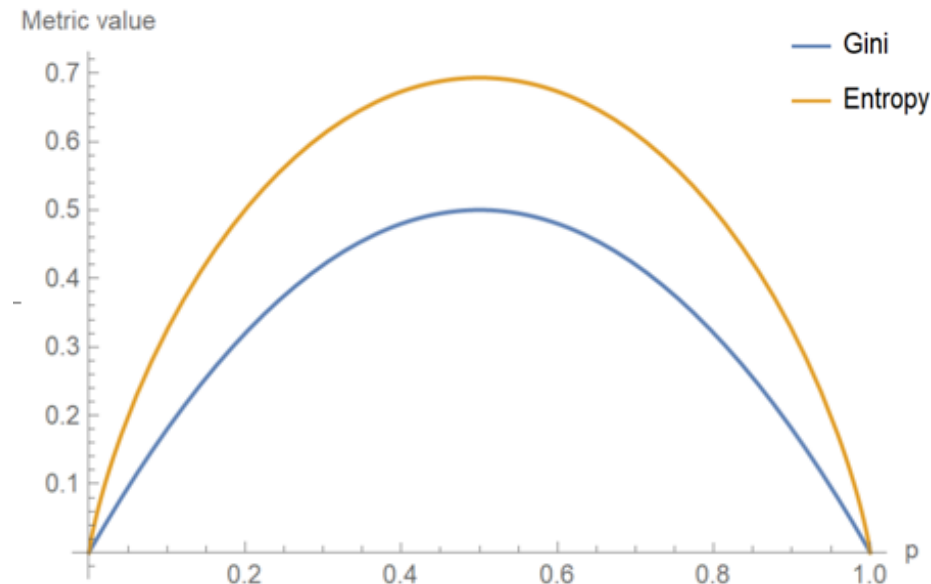
$$I_G(p) = \sum_{i=1}^J \left(p_i \sum_{k \neq i} p_k \right) = \sum_{i=1}^J p_i (1 - p_i) = \sum_{i=1}^J (p_i - p_i^2) = \sum_{i=1}^J p_i - \sum_{i=1}^J p_i^2 = 1 - \sum_{i=1}^J p_i^2$$

性質: 越低，節點的同質性越高

純節點（相同類）的基尼不純度為零

採用GINI impurity的代表是CART tree

二進位拆分，僅用於分類目標



目標:預測出喜歡打板球類型的學生

Split on Gender

Students = 30
Play Cricket = 15 (50%)



Female



Students = 10
Play Cricket = 2 (20%)

Male



Students = 20
Play Cricket = 13 (65%)

Split on Class



Class IX



Students = 14
Play Cricket = 6 (43%)

Class X



Students = 16
Play Cricket = 9 (56%)

1.分別計算每個子節點不純度再加權

用性別來分類：

Female節點：

10位女性，其中有2位打板球8位不打

Gini impurity : $1-(0.2)^2-(0.8)^2 = 0.32$

Male節點：

20位男性，其中有13位打板球7位不打

Gini impurity : $1-(0.65)^2-(0.35)^2 = 0.45$

以性別分類的Gini impurity加權後

$= (10/30)*0.32+(20/30)*0.45 = 0.41$

Split on Gender

Students = 30
Play Cricket = 15 (50%)



Female



Students = 10
Play Cricket = 2 (20%)

Male



Students = 20
Play Cricket = 13 (65%)

用班級來分類：

Class IX節點：

此班14位同學，其中6位打板球8位不打

Gini impurity: $1 - (0.43)^2 - (0.57)^2 = 0.49$

Class X節點：

此班16位同學，其中9位打板球7位不打

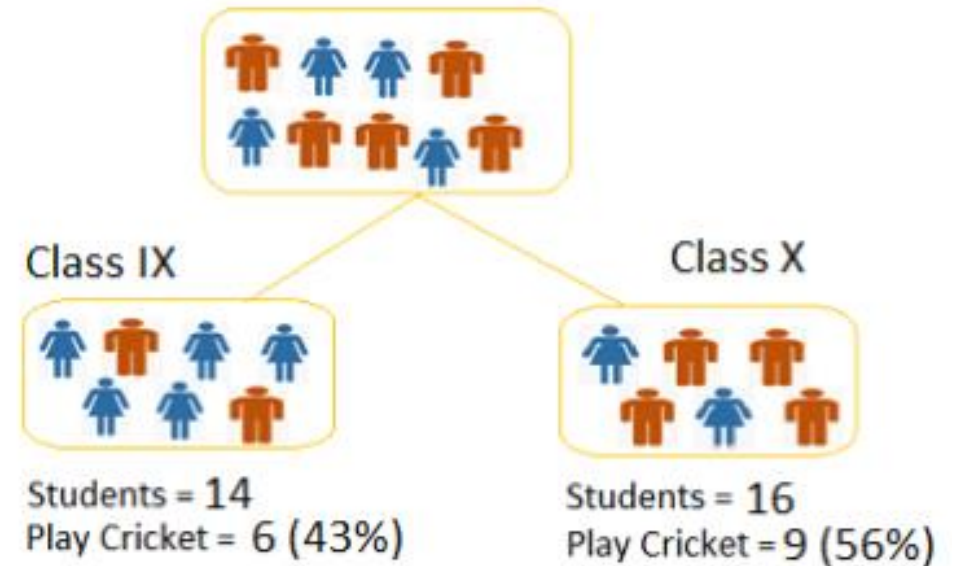
Gini impurity: $1 - (0.56)^2 - (0.44)^2 = 0.49$

以班級分類的Gini impurity加權後
= $(14/30) * 0.49 + (16/30) * 0.49 = 0.49$

性別Gini impurity = 0.41 < 班級Gini impurity = 0.49，因此採用性別來進行節點的分類。

2.選擇最小不純度做分割直到得到同類節點

Split on Class

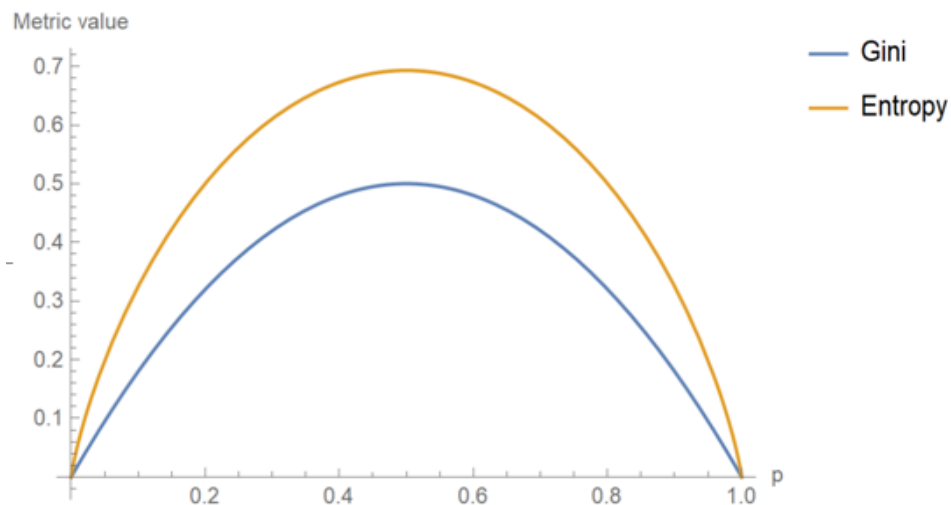


Entropy → Gini impurity

$$H(X) = \sum_i P(x_i) I(x_i) = - \sum_i P(x_i) \log_b P(x_i)$$

對 $-\log P(x)$ 做泰勒展開，忽略高階，得到

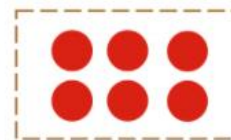
$$= \sum_{i=1}^J p_i (1 - p_i) = \sum_{i=1}^J (p_i - p_i^2) = \sum_{i=1}^J p_i - \sum_{i=1}^J p_i^2 = 1 - \sum_{i=1}^J p_i^2$$



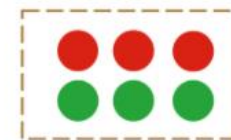
$$\text{Entropy} = -\sum p_j \log_2 p_j$$

$$\text{Information Gain} = -p * \log_2 p - q * \log_2 q$$

p : 是的機率 q : 否的機率



$$\text{Info}(6, 0) = -\frac{6}{6} \log_2 \left(\frac{6}{6}\right) - \frac{0}{6} \log_2 \left(\frac{0}{6}\right) = 0$$



$$\text{Info}(3, 3) = -\frac{3}{6} \log_2 \left(\frac{3}{6}\right) - \frac{3}{6} \log_2 \left(\frac{3}{6}\right) = 1$$

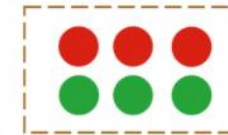
$$\text{Gini} = 1 - \sum p_j^2$$

$$\text{Gini Impurity} = 1 - (p^2 + q^2)$$

p : 是的機率 q : 否的機率



$$\text{Info}(6, 0) = 1 - (1^2 + 0^2) = 0$$



$$\text{Info}(3, 3) = 1 - (0.5^2 + 0.5^2) = 0.5$$

缺點：過擬合

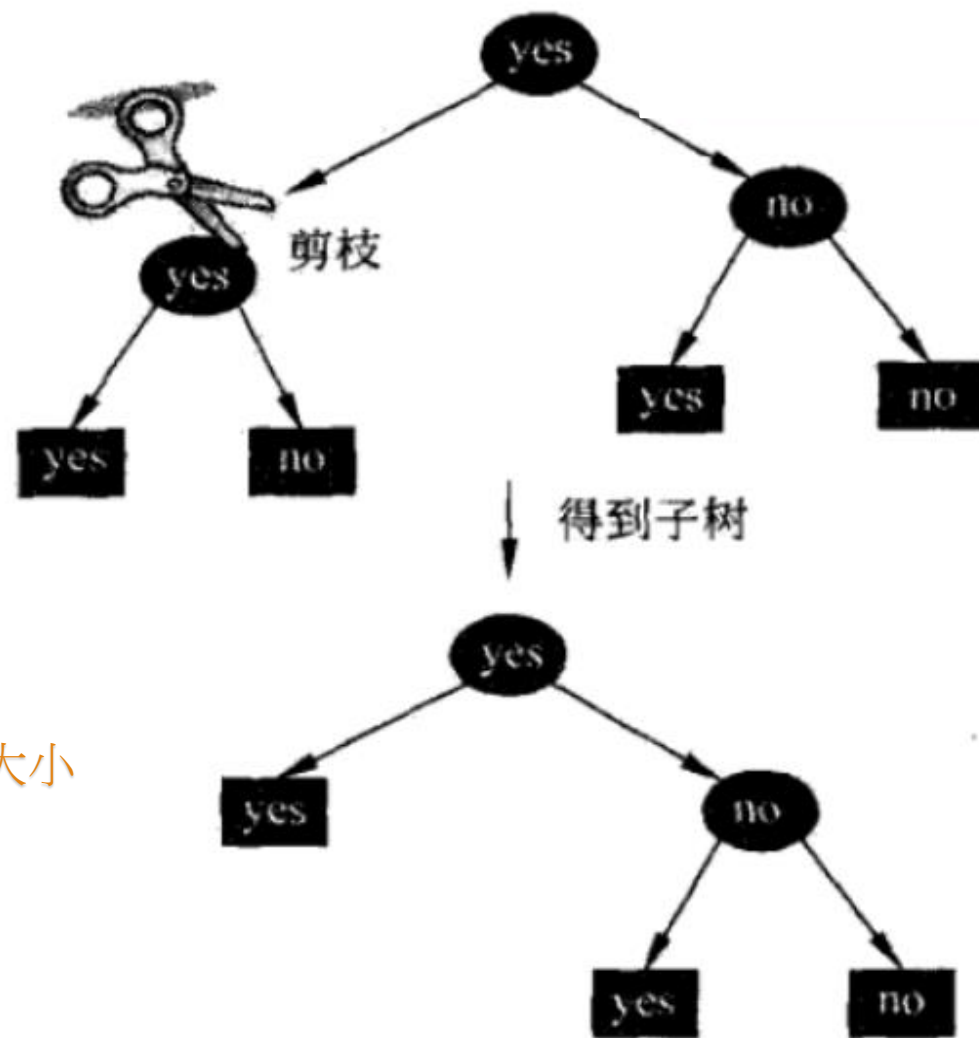
預先剪枝：

在樹的生長過程中設定一個指標，
當達到該指標時就停止生長，
容易產生「視界局限」

後剪枝：

讓樹充分生長，
然後對相鄰的葉節點，
考慮是否消去它們

樹的深度、樣本數
特徵個數、資訊增益大小



CART:迴歸樹

- Mean Square Error

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- Mean Absolute Error

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

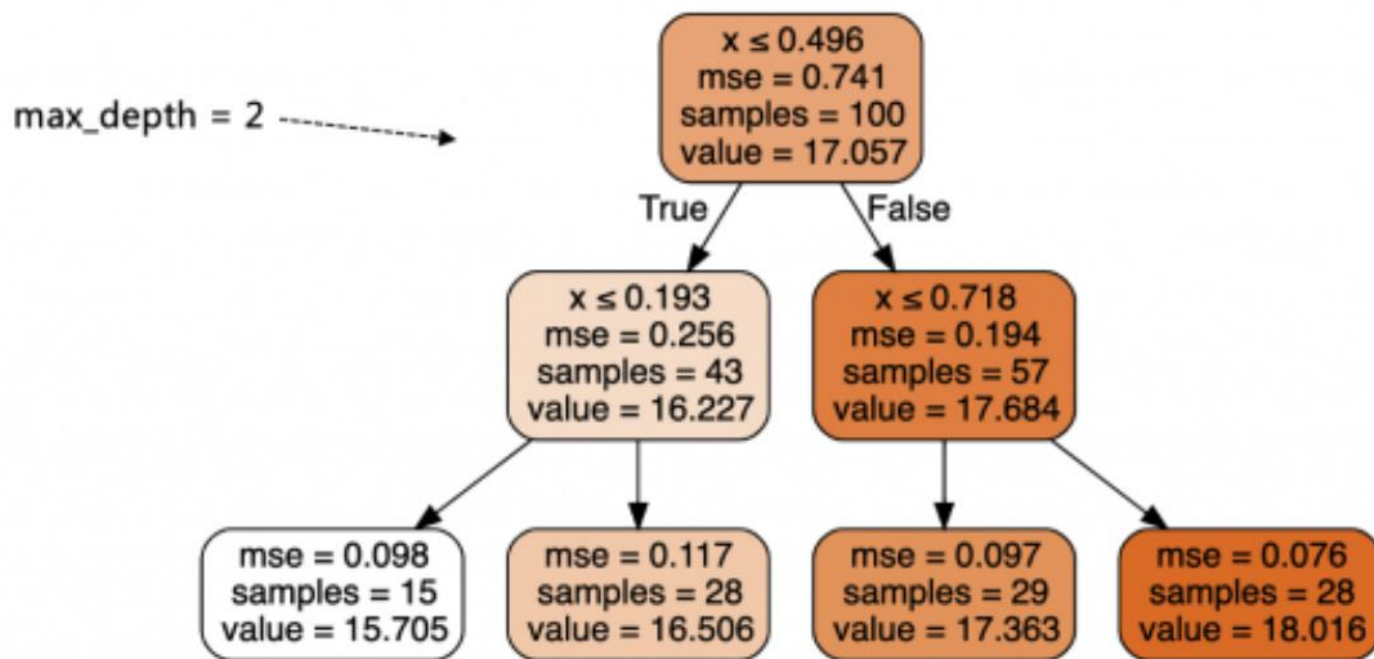
流程:

1. 分Train Test
2. Train建決策樹
3. Test 剪枝
4. 重複 2.3.

對回歸樹採用 MSE

對分類樹採 Gini impurity

結構簡潔的二叉樹



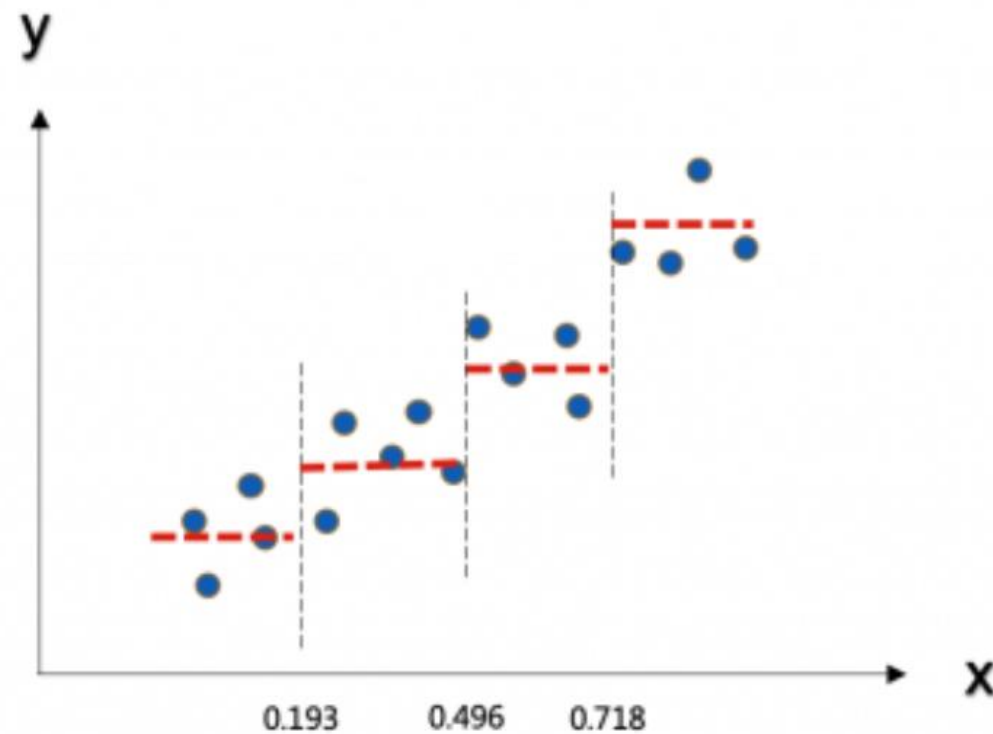
CART:迴歸樹

模型：

$$f(x) = \sum_{m=1}^M c_m I(x \in R_m)$$

輸出：

$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m)$$



算法 5.5 (最小二乘回归树生成算法)

输入: 训练数据集 D ;

输出: 回归树 $f(x)$.

在训练数据集所在的输入空间中, 递归地将每个区域划分为两个子区域并决定每个子区域上的输出值, 构建二叉决策树:

(1) 选择最优切分变量 j 与切分点 s , 求解

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \quad (5.21)$$

遍历变量 j , 对固定的切分变量 j 扫描切分点 s , 选择使式 (5.21) 达到最小值的对 (j,s) .

(2) 用选定的对 (j,s) 划分区域并决定相应的输出值:

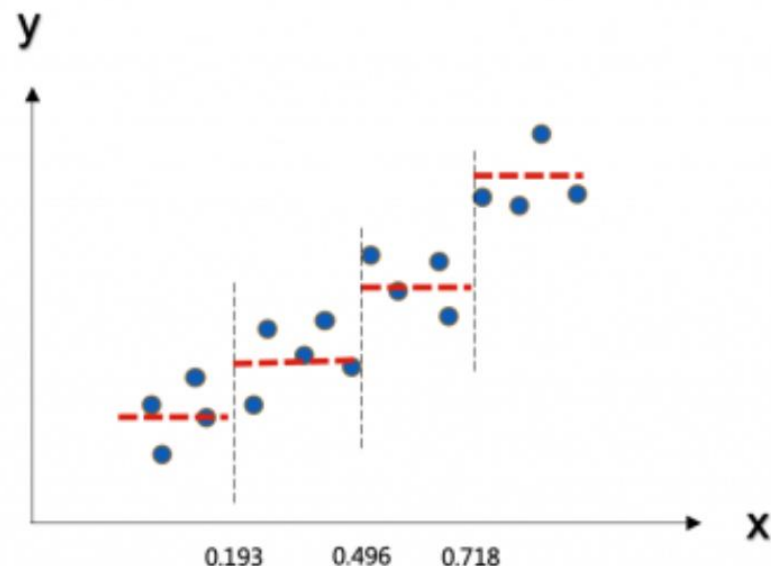
$$R_1(j,s) = \{x | x^{(j)} \leq s\}, \quad R_2(j,s) = \{x | x^{(j)} > s\}$$

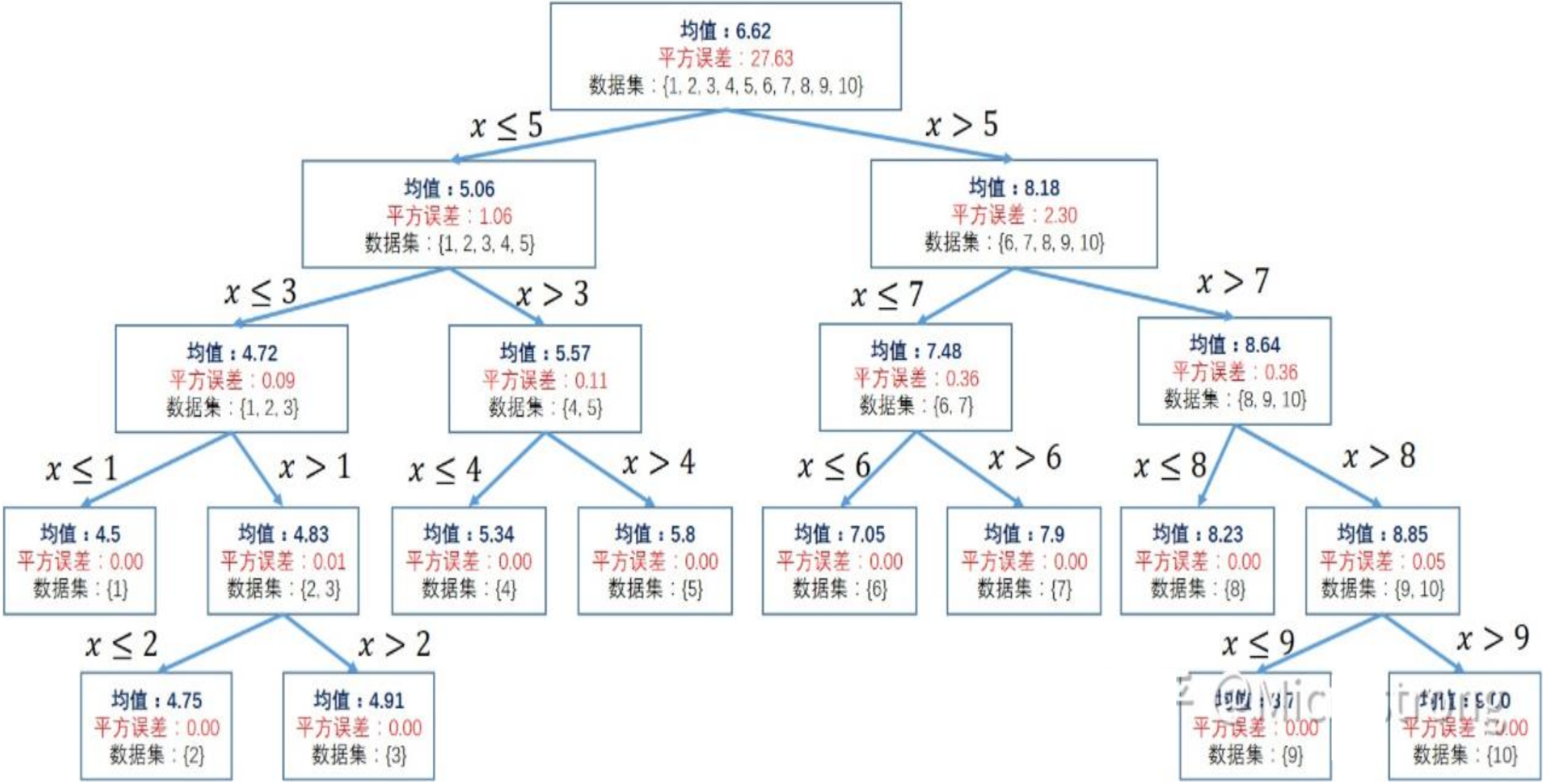
$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m(j,s)} y_i, \quad x \in R_m, \quad m=1,2$$

(3) 继续对两个子区域调用步骤 (1), (2), 直至满足停止条件.

(4) 将输入空间划分为 M 个区域 R_1, R_2, \dots, R_M , 生成决策树:

$$f(x) = \sum_{m=1}^M \hat{c}_m I(x \in R_m)$$





已知如圖所示的訓練數據，試用平方誤差損失準則生成一個二叉回歸樹

x_i	1	2	3	4	5	6	7	8	9	10
y_i	4.50	4.75	4.91	5.34	5.80	7.05	7.90	8.23	8.70	9.00

本示例來源於李航著的《統計學習方法》

x_i	1	2	3	4	5	6	7	8	9	10
y_i	4.50	4.75	4.91	5.34	5.80	7.05	7.90	8.23	8.70	9.00

1. 求解：

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

例如，取 $s=1$ 。此時 $R_1 = \{1\}$ ， $R_2 = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$ ，這兩個區域的輸出值分別為：

$$c_1 = 4.50$$

$$c_2 = \frac{1}{9}(4.75 + 4.91 + 5.34 + 5.80 + 7.05 + 7.90 + 8.23 + 8.70 + 9.00) = 6.85$$

根據上面的計算方法，可以得到下表：

s	1	2	3	4	5	6	7	8	9	10
c_1	4.50	4.63	4.72	4.88	5.06	5.39	5.75	5.18	6.35	6.62
c_2	6.85	7.12	7.43	7.78	8.18	8.46	8.64	8.85	9.00	0.00

x_i	1	2	3	4	5	6	7	8	9	10
y_i	4.50	4.75	4.91	5.34	5.80	7.05	7.90	8.23	8.70	9.00

s	1	2	3	4	5	6	7	8	9	10
c_1	4.50	4.63	4.72	4.88	5.06	5.39	5.75	5.18	6.35	6.62
c_2	6.85	7.12	7.43	7.78	8.18	8.46	8.64	8.85	9.00	0.00

把 C_1 、 C_2 的值代入到MSE中得到:

$$m(1) = 0 + \{(4.75 - 6.85)^2 + (4.91 - 6.85)^2 + (5.34 - 6.85)^2 + (5.80 - 6.85)^2 + (7.05 - 6.85)^2 + (7.90 - 6.85)^2 + (8.23 - 6.85)^2 + (8.70 - 6.85)^2 + (9.00 - 6.85)^2\} = 22.65$$

同理，可以獲得下表

s	1	2	3	4	5	6	7	8	9	10
m(s)	22.65	17.70	12.19	7.38	3.36	5.07	10.05	15.18	21.33	27.63

$s = 5$ 時， $m(s)$ 最小，因此第一個最優切分變量為 $j = x$ 、最優切分點為 $s = 5$

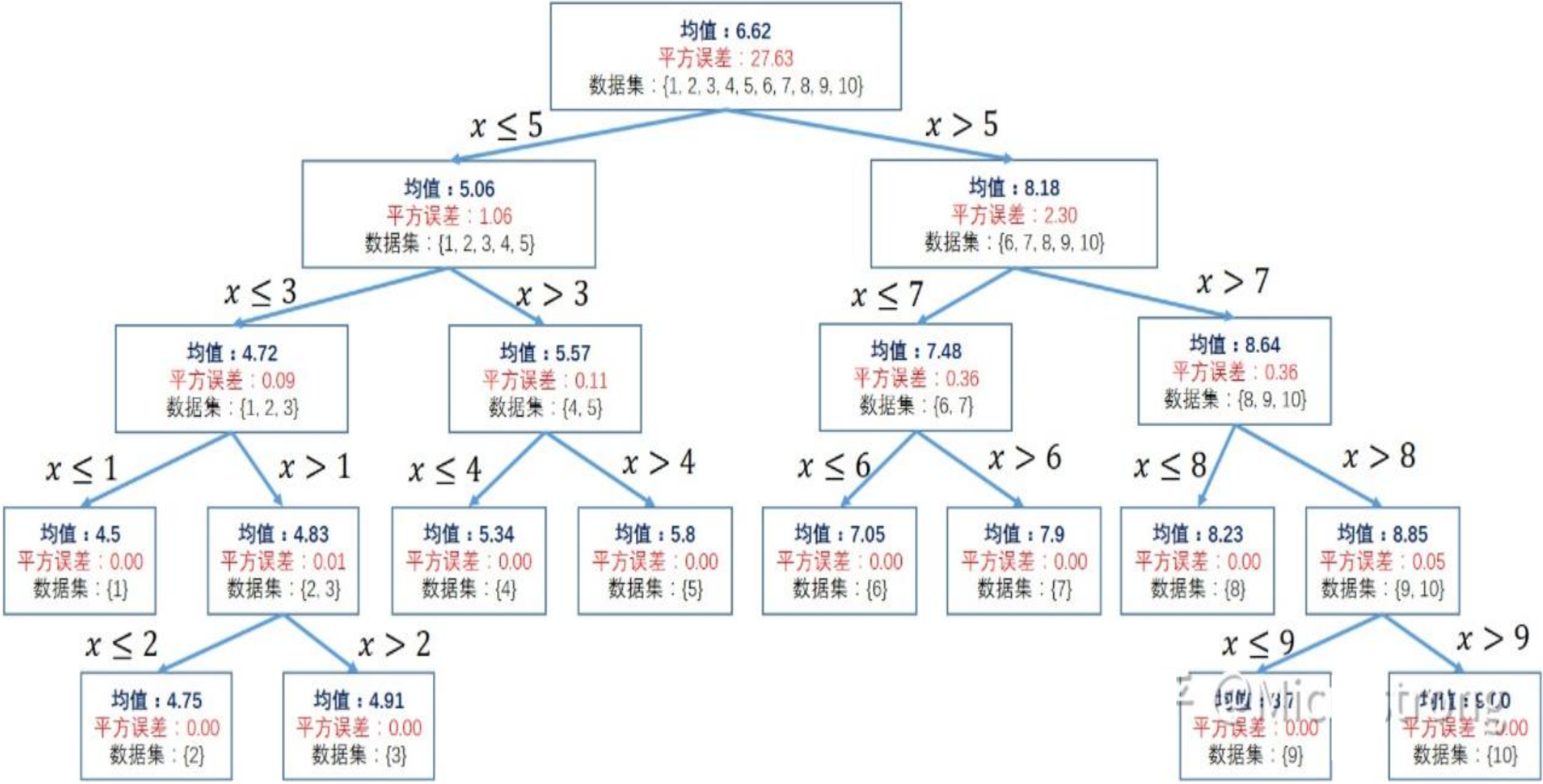
2. 用選定的 (j, s) 劃分區域，並決定輸出值

兩個劃分的區域分別是： $R_1 = \{1, 2, 3, 4, 5\}$, $R_2 = \{6, 7, 8, 9, 10\}$ 。輸出值用公式：

$$\hat{c}_1 = ave(y_i | x_i \in R_1(j, s)) \text{ 和 } \hat{c}_2 = ave(y_i | x_i \in R_2(j, s))$$

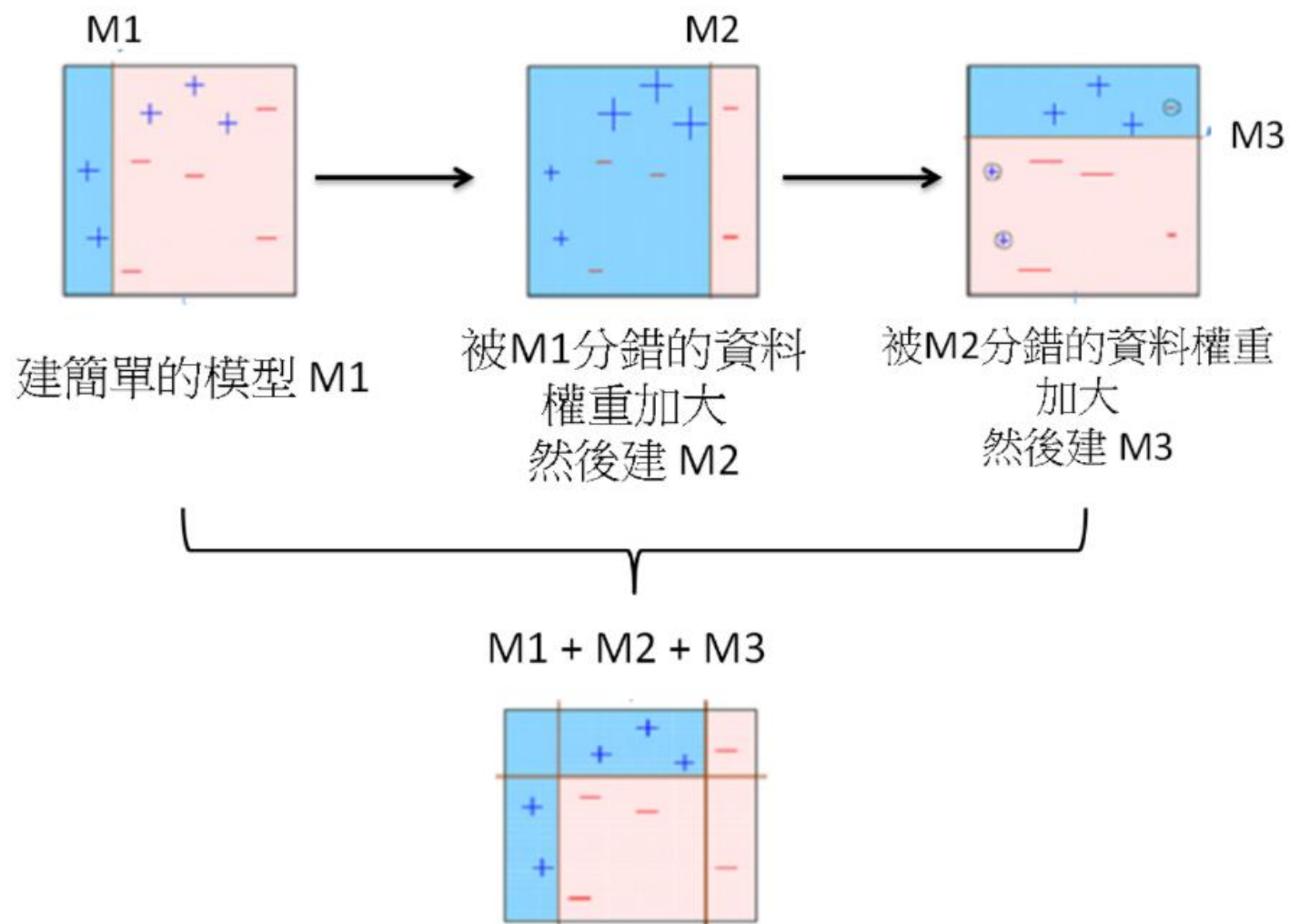
得到 $c_1 = 5.06$, $c_2 = 8.18$ 。

s	1	2	3	4	5	6	7	8	9	10
c_1	4.50	4.63	4.72	4.88	5.06	5.39	5.75	5.18	6.35	6.62
c_2	6.85	7.12	7.43	7.78	8.18	8.46	8.64	8.85	9.00	0.00



算法	支持模型		树结构	特征选择		连续值处理	缺失值处理	剪枝
ID3	分类		多叉树	信息增益		不支持	不支持	不支持
C4.5	分类		多叉数	信息增益比		支持	支持	支持
CART	分类	回归	二叉树	基尼指数	均方差	支持	支持	支持

分類樹：利用投票決定最後預測
迴歸樹：利用均值作為預測值



Boosting

$$\begin{aligned} E(F) &= E\left(\sum_i^m r_i f_i\right) \\ \text{模型總體期望} &= \sum_i^m r_i E(f_i) \end{aligned}$$

模型總體方差（公式推導參考協方差的性質，協方差與方差的關係）：

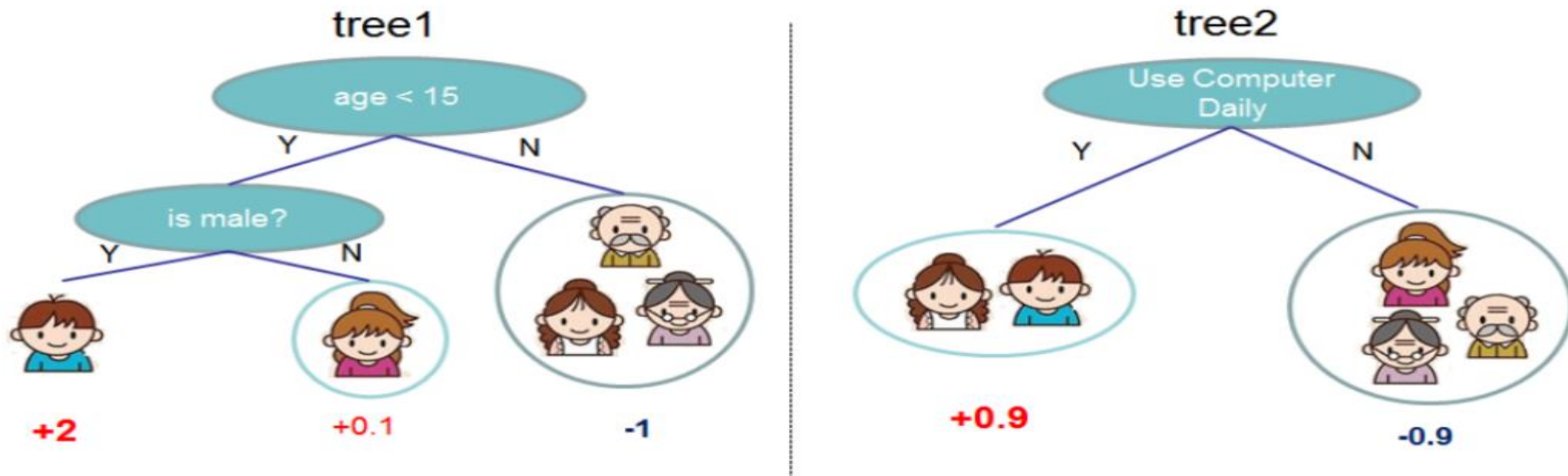
$$\begin{aligned} Var(F) &= Var\left(\sum_i^m r_i f_i\right) \\ &= \sum_i^m Var(r_i f_i) + \sum_{i \neq j}^m Cov(r_i f_i, r_j f_j) \\ &= \sum_i^m r_i^2 Var(f_i) + \sum_{i \neq j}^m \rho r_i r_j \sqrt{Var(f_i)} \sqrt{Var(f_j)} \\ &= mr^2 \sigma^2 + m(m-1) \rho r^2 \sigma^2 \\ &= mr^2 \sigma^2 (1 - \rho) + m^2 r^2 \sigma^2 \rho \end{aligned}$$

模型的準確度可由偏差和方差共同決定：

$$Error = bias^2 + var + \xi$$


XGBOOST


$$Obj = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$



已訓練k棵樹，對第i個樣本的預測值為

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), f_k \in F$$

f() = 2 + 0.9 = 2.9

f() = -1 - 0.9 = -1.9

疊加式訓練

$$\hat{y}_i^t = \hat{y}_i^{(t-1)} + f_t(x_i)$$

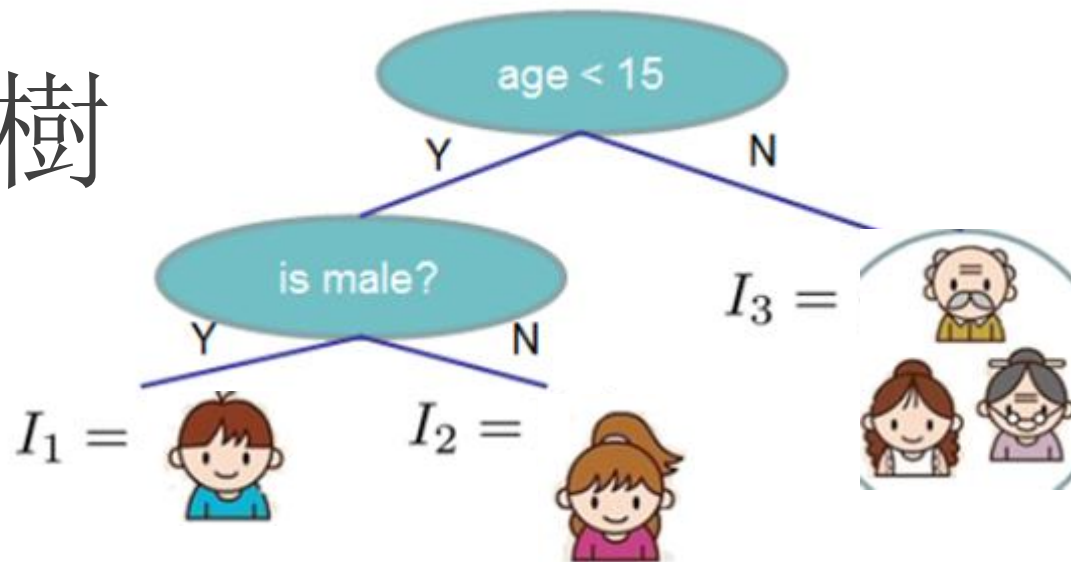
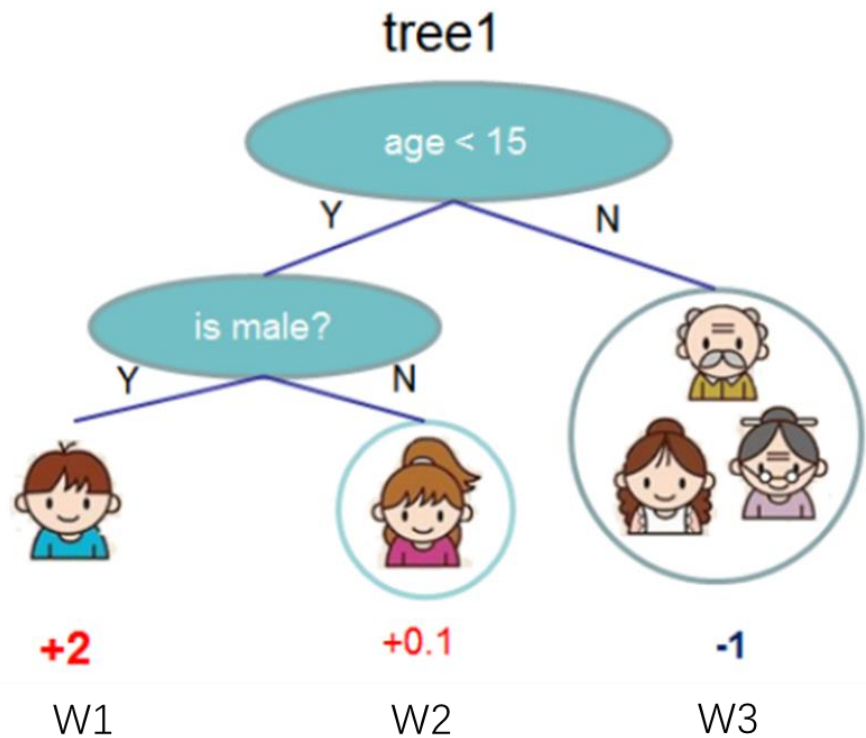
目標函數改寫為：

$$Obj = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{i=1}^{t-1} \Omega(f_i) + \Omega(f_t)$$

泰勒展開近似目標函數

$$Obj^{(t)} = \sum_{i=1}^n \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

定義一棵樹



$q(\mathbf{x}_i)$ 樣本 \mathbf{x} 的位置

$$f_t(x_i) = w_{q(x_i)} f(\text{young man icon}) = 2$$

$$I_j = \{i | q(\mathbf{x}_i) = j\}$$

$$\Omega(f_i) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

$$Obj^{(t)} = \sum_{i=1}^n \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$






$$Obj^{(t)} = \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T$$

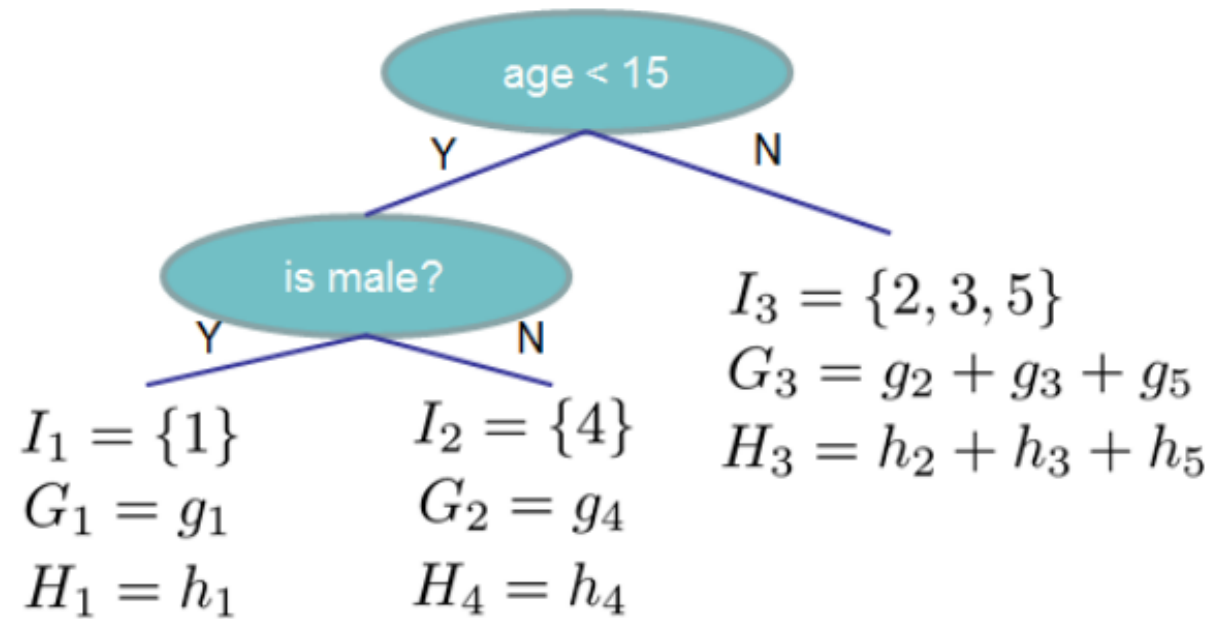
$$Obj^{(t)} = \sum_{j=1}^T \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T$$

$$w_j^* = - \frac{G_j}{H_j + \lambda}$$

$$Obj^{(t)} = - \frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$$

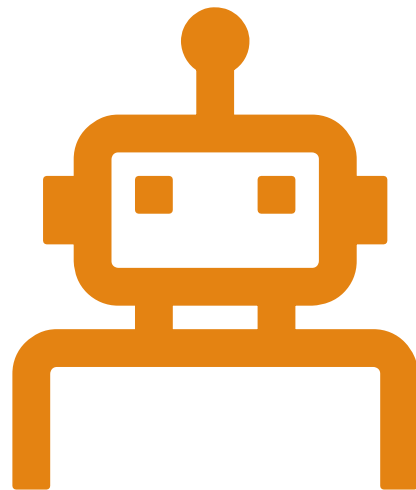
Instance index gradient statistics

1		g_1, h_1
2		g_2, h_2
3		g_3, h_3
4		g_4, h_4
5		g_5, h_5



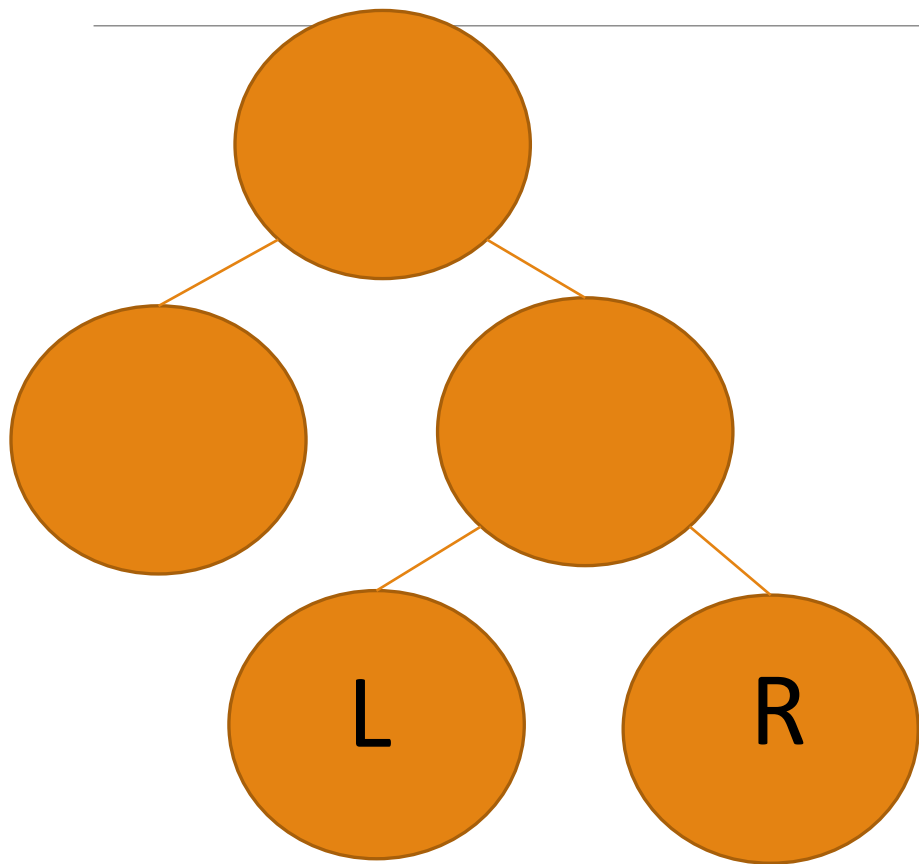
$$Obj = - \sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is



如何尋找樹的形狀

貪心算法



$$obj_1 - obj_2 = \frac{1}{2} \left[\frac{G_L}{H_L + \lambda} + \frac{G_R}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

近似算法

Algorithm 2: Approximate Algorithm for Split Finding

```
for  $k = 1$  to  $m$  do  
    | Propose  $S_k = \{s_{k1}, s_{k2}, \dots, s_{kl}\}$  by percentiles on feature  $k$ .  
    | Proposal can be done per tree (global), or per split(local).  
end  
for  $k = 1$  to  $m$  do  
    |  $G_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$   
    |  $H_{kv} \leftarrow \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$   
end
```

Follow same step as in previous section to find max score only among proposed splits.

ϵ -approximate ϕ -quantiles

input: 14, 19, 3, 15, 4, 6, 1, 13, 13, 7, 11, 8, 4, 5, 15, 2

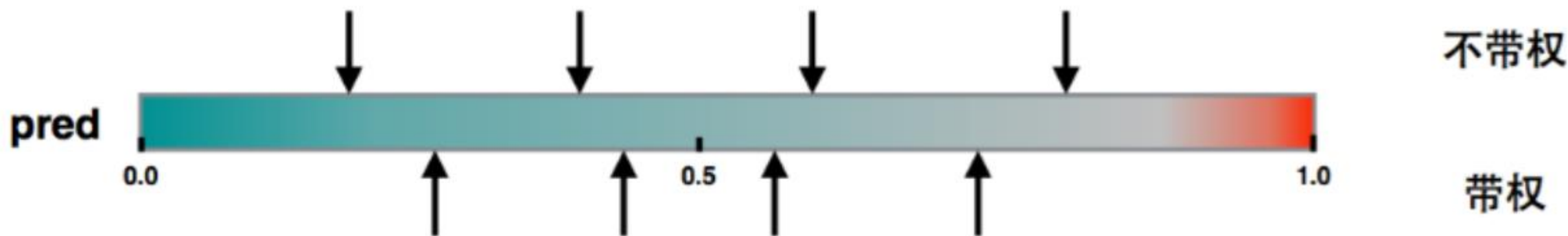
sort: 1, 2, 3, 4, 4, 5, 6, 7, 8, 11, 13, 13, 14, 15, 15, 19

rank: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

$$\sum_i \left[L(y_i, \hat{y}_i^{K-1}) + g_i f_K(x_i) + \frac{1}{2} h_i f_K^2(x_i) \right] + \Omega(f_K) + \text{constant}$$

$$\sum_i \left[\frac{1}{2} h_i (f_K(x_i) - (-g_i/h_i))^2 \right] + \Omega(f_K) + \text{constant}$$

预测结果越是不确定，样本的权重越高，切分的粒度越密集！



$$D_k = \{(x_{1k}, h_1), (x_{2k}, h_2), \dots (x_{nk}, h_n)\}$$

$$r_k : \mathbb{R} \rightarrow [0, +\infty) \text{ as}$$

$$r_k(z) = \frac{1}{\sum_{(x,h) \in \mathcal{D}_k} h} \sum_{(x,h) \in \mathcal{D}_k, x < z} h,$$

$$|r_k(s_{k,j}) - r_k(s_{k,j+1})| < \epsilon, \quad s_{k1} = \min_i \mathbf{x}_{ik}, s_{kl} = \max_i \mathbf{x}_{ik}$$


Lightgbm

LIGHT(輕量) GBM (GRADIENT BOOSTING MACHINE)

由為軟體出，用於解決GBDT中遇到的問題，以便其可以在工業中更容易地被實現。

與XGBoost的比較

Data	xgboost	xgboost_hist	LightGBM	Data	xgboost	xgboost_hist	LightGBM
Higgs	4.853GB	3.784GB	0.868GB	Higgs	3794.34 s	551.898 s	238.505513 s
Yahoo LTR	1.907GB	1.468GB	0.831GB	Yahoo LTR	674.322 s	265.302 s	150.18644 s
MS LTR	5.469GB	3.654GB	0.886GB	MS LTR	1251.27 s	385.201 s	215.320316 s
Expo	1.553GB	1.393GB	0.543GB	Expo	1607.35 s	588.253 s	138.504179 s



Lightgbm無論是速度或是內存都相較於xgboost和xgboost_hist更好

Lightgbm優點

- 1.更快的訓練速度
- 2.更低的內存
- 3.準確度更高
- 4.能處理大量運算

改良xgboost缺點

- 1.單邊梯度抽樣算法(`gradient-based one side sampling` , `goss`)
- 2.直方圖算法
- 3.互斥特徵捆绑算法(`greedy bundling`, `merge exclusive features`)
- 4.`leaf-wise`
- 5.類別特徵最優分割
- 6.特徵並行與數據並行
- 7.緩存優化

單邊梯度抽樣算法(goss)

GOSS演算法主要為了解決權重問題。

方法:

- 1.對樣本進行抽樣，選擇梯度較小部分的樣本
- 2.讓模型關注梯度高的樣本，以達到減少計算量

Q:為何選擇梯度較小的樣本抽樣?

GOSS演算法

Algorithm 2: Gradient-based One-Side Sampling

Input: I : training data, d : iterations

Input: a : sampling ratio of large gradient data

Input: b : sampling ratio of small gradient data

Input: $loss$: loss function, L : weak learner

$models \leftarrow \{\}$, $fact \leftarrow \frac{1-a}{b}$

$topN \leftarrow a \times \text{len}(I)$, $randN \leftarrow b \times \text{len}(I)$

for $i = 1$ **to** d **do**

$preds \leftarrow models.predict(I)$

$g \leftarrow loss(I, preds)$, $w \leftarrow \{1, 1, \dots\}$

$sorted \leftarrow \text{GetSortedIndices}(\text{abs}(g))$

$topSet \leftarrow sorted[1:topN]$

$randSet \leftarrow \text{RandomPick}(sorted[topN:\text{len}(I)],$
 $randN)$

$usedSet \leftarrow topSet + randSet$

$w[randSet] \times = fact$ ▷ Assign weight $fact$ to the
 small gradient data.

$newModel \leftarrow L(I[usedSet], -g[usedSet],$

$w[usedSet])$

$models.append(newModel)$

$\frac{1-a}{b}$: 權重係數(保持與原樣
本同分佈)

類似adaboost，但adaboost一開始就會因為分錯而改變權值，使得學習機器優先關注分錯的樣本。

goss不一樣，goss沒有權值，依靠著天生的梯度提供了有用的資料抽樣

例子： 100個樣本，梯度大的20個(即 $a=20\%$)，假設我們選梯度小的30%即 $b=30\%$

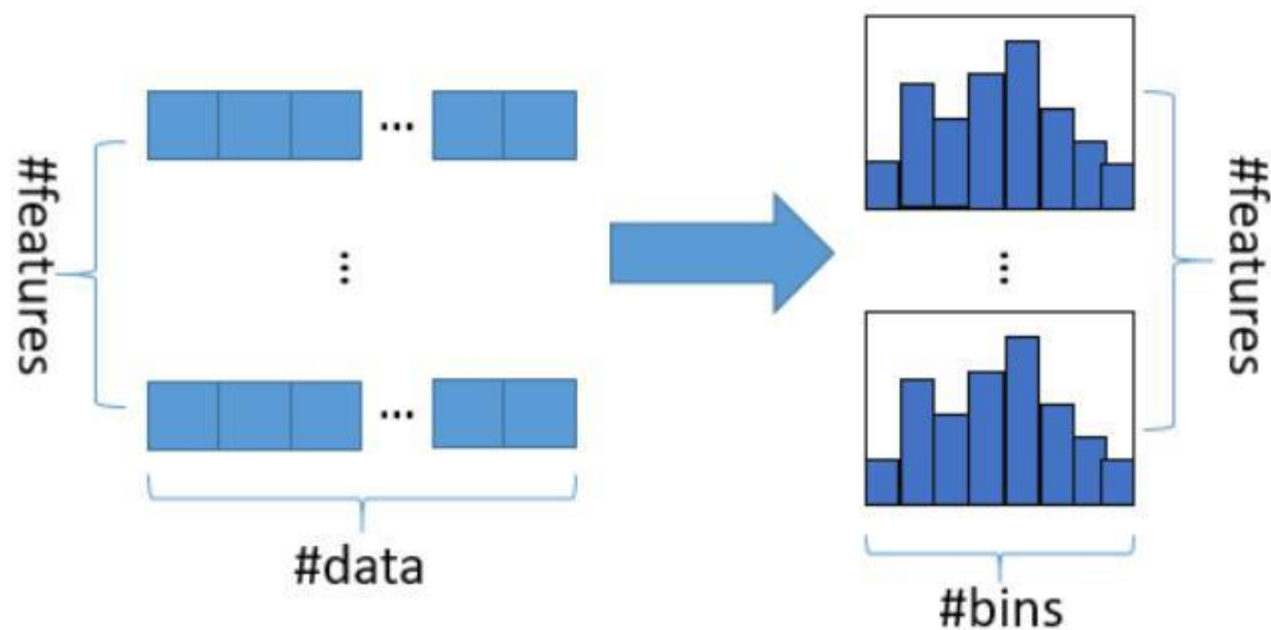
$a=100\%$ ，也就是說我全部都選大樣本為抽樣(權重係數=0)

$a=0\%$ ，也就是說goss採取著隨機採樣

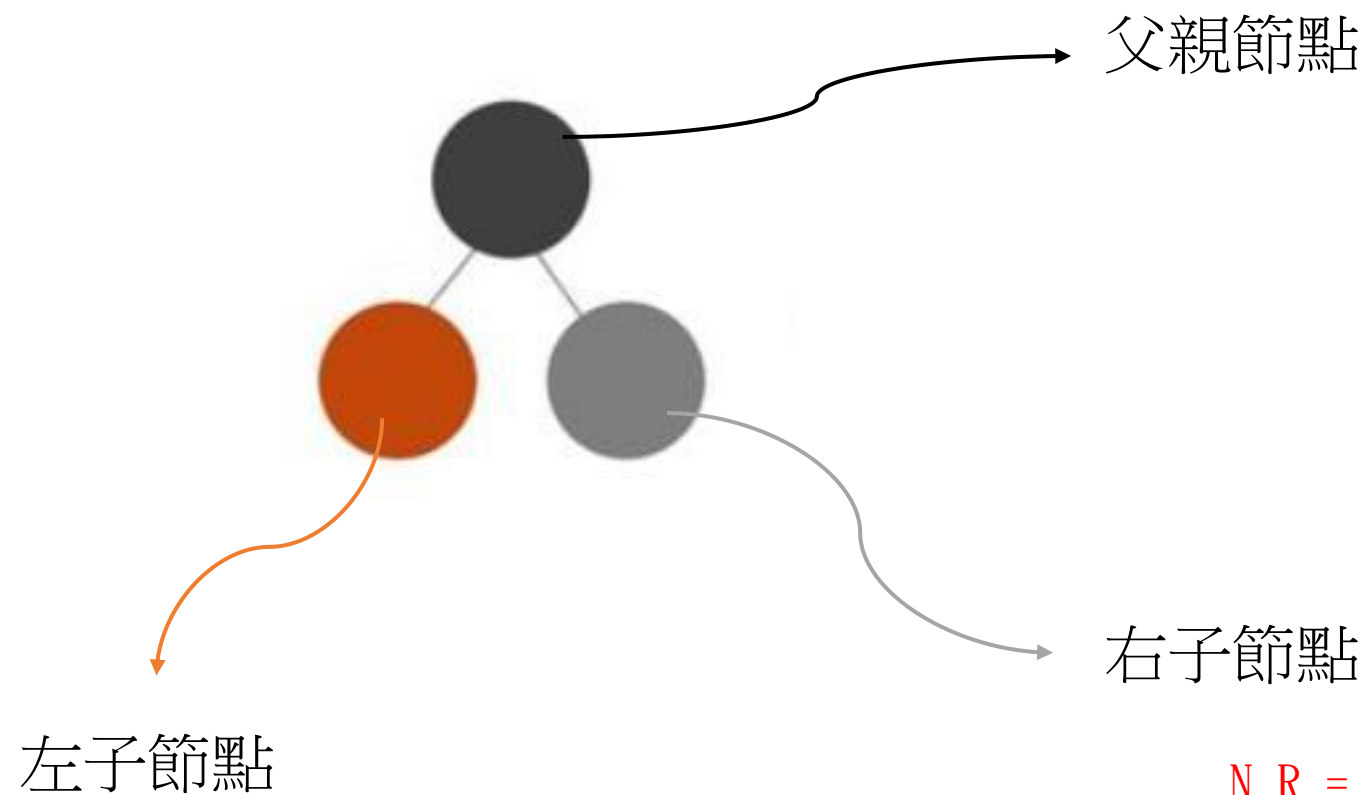
Goss可以在不改變數據分佈和不損精準度的情況下，減少模型學習的速率

直方圖優化演算法

將我們連續型的特徵(feature)資料，轉化為離散的資料，相當於裝進盒子中(bin)處理。



父親節點、左右節點



$$N_R = N_P - N_L$$

右子節點樣本數=父親節點樣本數-左子節點樣本數

直方圖優化演算法

Algorithm: **FindBestSplitByHistogram**

Input: Training data X , Current Model $T_{c-1}(X)$

First order gradient G , second order gradient H

For all Leaf p in $T_{c-1}(X)$:

For all f in X .Features:

▷ construct histogram

$H = \text{new Histogram}()$

For i in $(0, \text{num_of_row})$ //go through all the data row

$H[f.\text{bins}[i]].g += g_i; H[f.\text{bins}[i]].n += 1$

▷ find best split from histogram

For i in $(0, \text{len}(H))$: //go through all the bins

$S_L += H[i].g; n_L += H[i].n$

$S_R = S_P - S_L; n_R = n_P - n_L$

$$\Delta loss = \frac{s_L^2}{n_L} + \frac{s_R^2}{n_R} - \frac{s_P^2}{n_P}$$

if $\Delta loss > \Delta loss(p_m, f_m, v_m)$:

$(p_m, f_m, v_m) = (p, f, H[i].value)$

此演算法之前，
需先將我們的特徵
(feature)->箱子(bin)

與goss不同，直方圖優
化演算法不需要排序

S_L :累積其左邊bin至目前bin的梯度和

N_L :累積其左邊樣本和

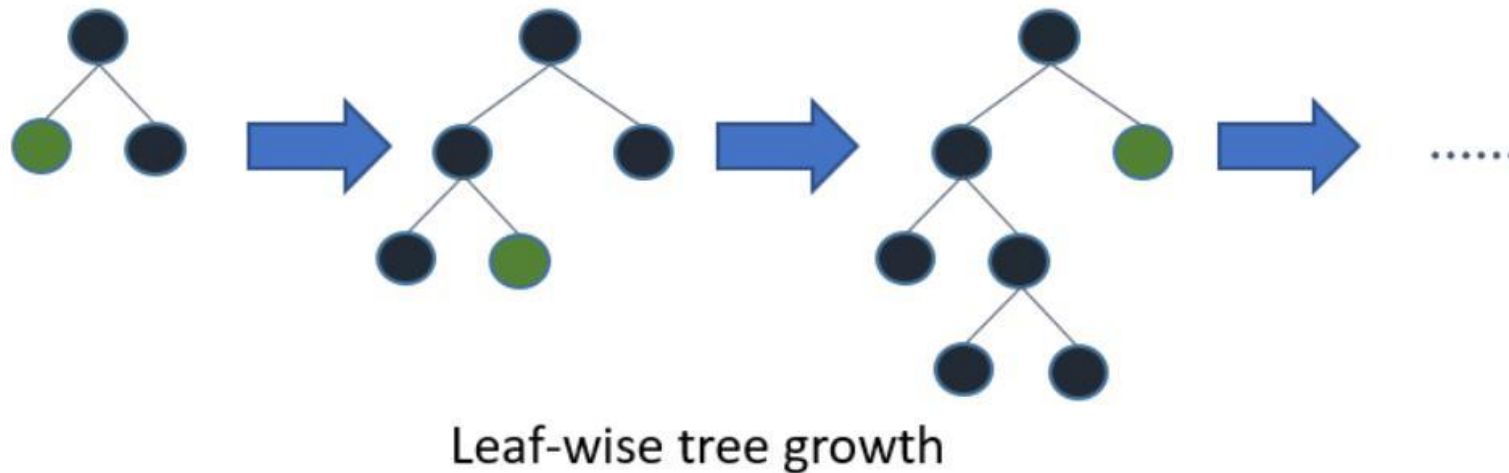
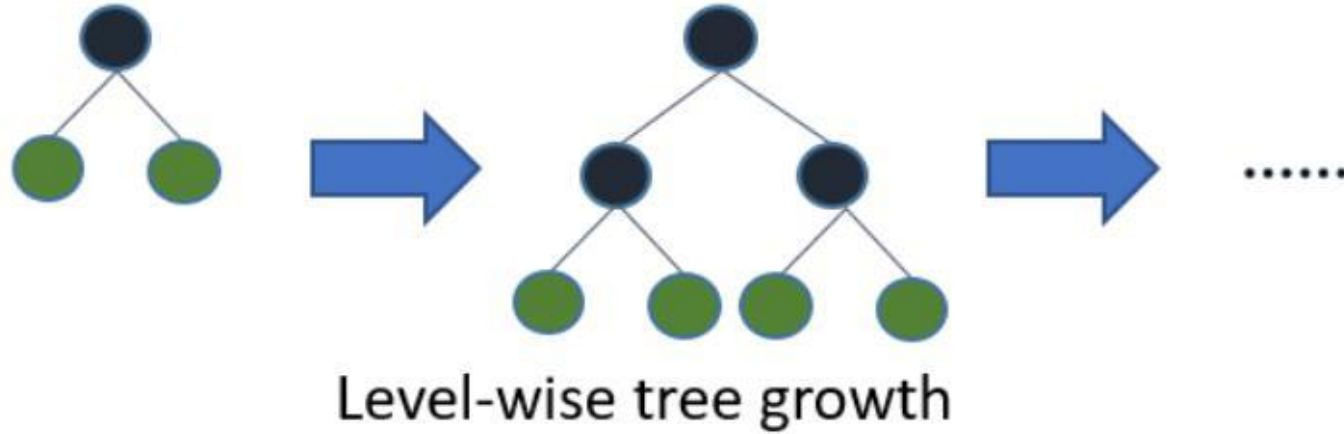
例子

父親節點**10**個樣本，**2**個特徵。左節點具有**4**個樣本，右節點具有**6**個樣本。

樹的優化

上面的兩個方法都是優化分裂點，其實lightgbm不只優化了分裂點，甚至還優化了樹的生長過程，他拋棄xgboost的按層生長(level wise)，而是選擇葉子生長(leaf wise)。

Leaf-wise(Best-first) Tree growth



缺點與改良

缺點:擔心過度擬合是最大的缺點

改良:**leaf wise** 追求的更高的速度，更好的精度，會選擇最好的點做分裂，害怕過度擬合，因此使用了**max_depth**來控制整個樹的深度。

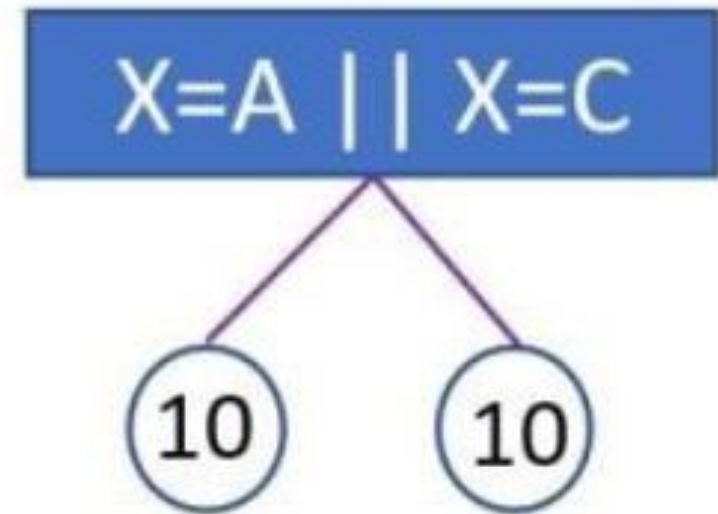
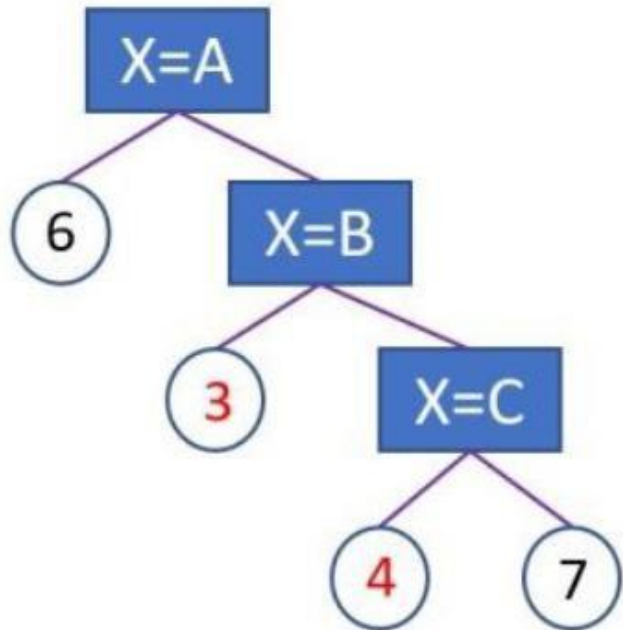
類別特徵最優分割

大部分的機器演算法都不能直接支持類別特徵，一般都會對類別特徵編碼，再輸入到模型，常見的如one-hot編碼，但因為one-hot編碼對於樣本的切分不平衡，會讓切分效益小。

Lightgbm使用的是many-vs-many進行分裂。

Many-vs-many

將數據分成兩大集合，主要是把很多樣本只有0或1小樣本的決策點減少，達到學習效率更好。



結論

xgboost vs lightgbm

	内存大小	訓練速度	精準度	支援大規模運算
xgboost	大	慢	較低	否
lightgbm	小	快	較高	是