

Machine Learning

Author: Puchi

目錄

監督式學習(Supervised Learning) -----01

非監督式學習(Unsupervised Learning) -----12

強化式學習(Reinforcement Learning) -----17

監督式學習(Supervised Learning)

由外界提供大量資料，並提供資料對應結果，讓電腦知道資料的相關性，進而從其相關性來探討是否具有因果關係。換句話說，電腦會接收每一筆具有「標準答案」的資料，當有新資料要判斷時，就會參考先前資料產生的分布特性，來達生判別與預測。

「特徵值」是資料間的差異，每筆資料都具有各自的特徵，我們可以藉由特徵值的差異性與相似性將資料加以分類。

「分類(Classification)」是電腦藉由分析每一筆資料的特徵值，加以整理建構出一套區分物件的分類器(Classifier)，分類器能夠用來判斷新資料是屬於哪一個分類。

「最短距離分類器」是將要訓練的資料數值化後，讓每個資料皆能展現於n維座標中(n大於等於1)。每筆資料在各個類別中，將其座標取算數平均，求得類別中心點所在座標。若有一筆新座標加入進來，我們可以對新座標與類別中心座標作「歐幾里得距離」，求得與各個類別中心座標的距離。只要與類別中心座標距離越小，那就能將其歸類在此類別。

歐幾里得距離

$$d(x, y) = \sqrt{(x_1 + y_1)^2 + (x_2 + y_2)^2 + \dots + (x_n + y_n)^2}$$

已知有藍色和綠色兩種類別，其資料已數值化成二維座標。分別為：

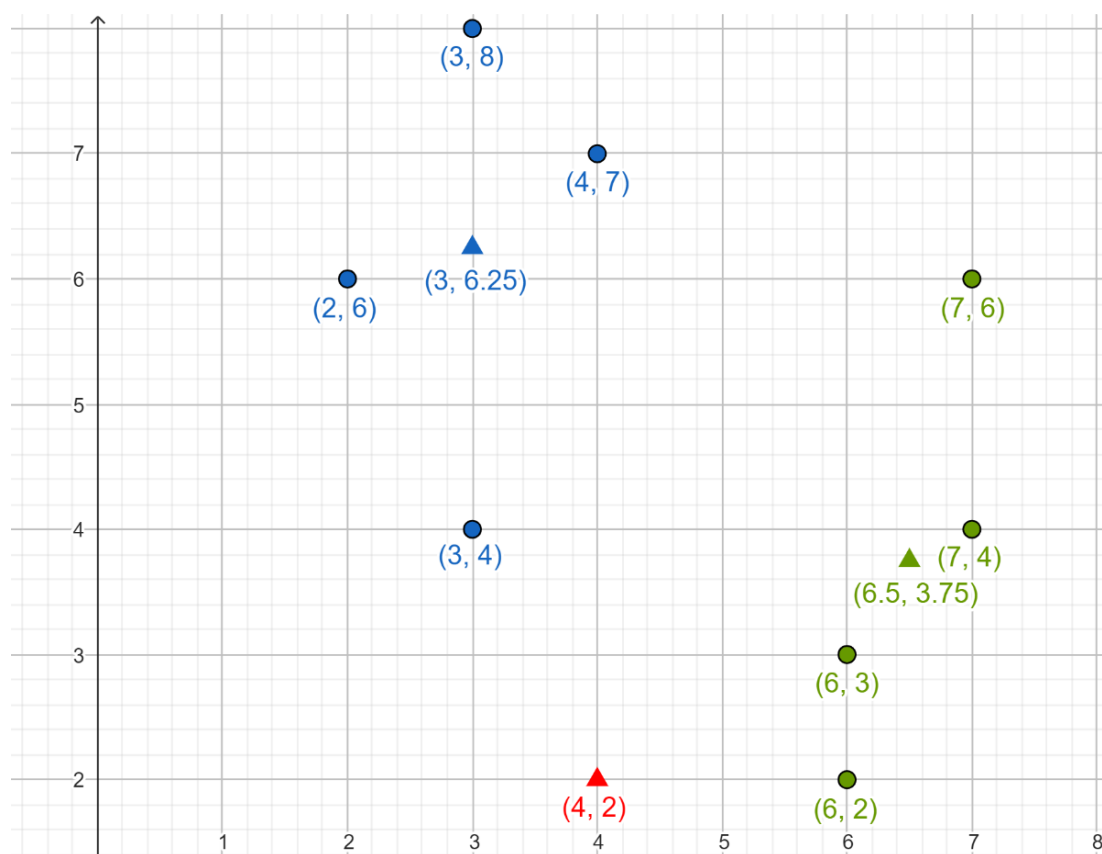
藍:(2,6)、(3,4)、(3,8)、(4,7) 藍色的類別中心座標(3,6.25)

綠:(6,2)、(6,3)、(7,4)、(7,6) 綠色的類別中心座標(6.5,3.75)

如果加入一筆新資料(4,2)點，其類別判斷要先使用歐幾里得距離計算兩者距離。與藍色的類別中心座標距離為 4.37，與綠色的類別中心座標距離為 3.05。新資料座標點與綠色的類別中心座標較近，因此將新資料分類為綠色。

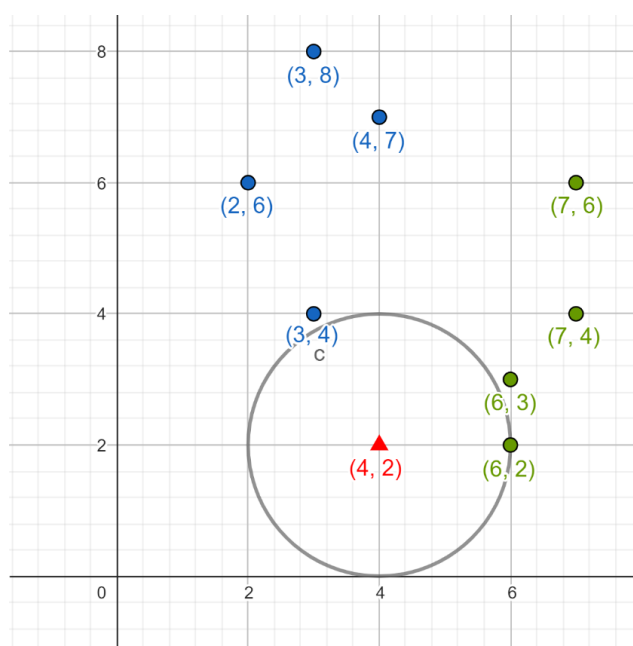
類別	X	Y	類別	X	Y
藍	2	6	綠	6	2
藍	3	4	綠	6	3
藍	3	8	綠	7	4
藍	4	7	綠	7	6
平均	3	6.25	平均	6.5	3.75

圖：藍類別與綠類別二維座標數值



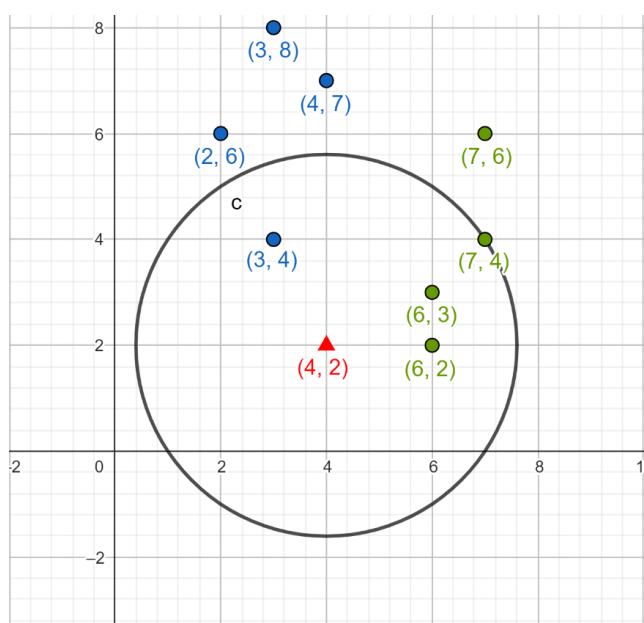
圖：所有座標的二維座標分

「K-近鄰分群法(K-Nearest Neighbor, KNN)」會找出與該筆資料最近的 k 筆資料來進行分類，當 k 個資料中某個類別數量較多時，新資料就會被分到該類別。



圖：K 值設定為 1 的座標圖

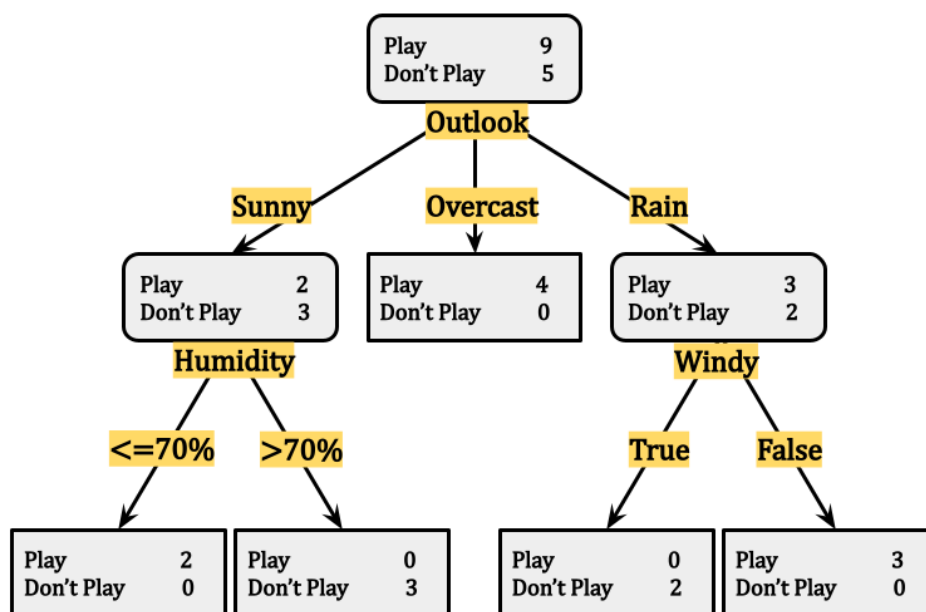
將 K 值設定為 1，也就是要找到離新資料(4,2)點最近的一個座標。由上圖可知距離最近點為綠類別，因此新資料將會被分類到綠類別。



圖：K 值設定為 4 的座標圖

將 K 值設定為 4，也就是要找到離新資料(4,2)點最近的四個座標。由上圖可知距離最近點且數量最多為綠類別，因此新資料將會被分類到綠類別。

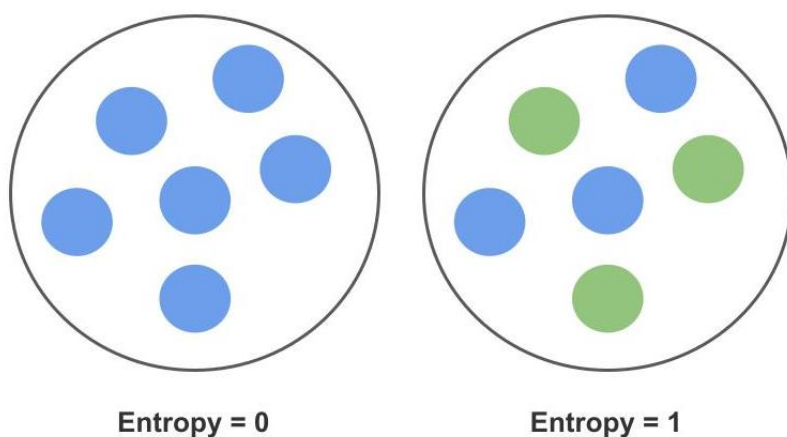
「決策樹(Decision tree)」是用來處理分類問題的樹狀結構，通常採用由上而下的方式，將整群資料從某個特徵開始展開成數個子群，直到所有子群資料都是同一個類別。



圖：決策樹概念圖

- 每個內部節點表示一個評估欄位
- 每個分枝代表一個可能的欄位輸出結果
- 每個樹葉節點代表不同分類的類別標記

為了知道哪一個特徵值可以建構最佳的決策樹，我們需要評斷特徵值的好壞，提供電腦選擇時的優先順序，以減少判斷條件的次數，而評斷特徵值的好壞是依據「熵(Entropy)」。熵可以呈現出資料的混亂程度，亂度越高代表資料越不一致。相反的，亂度越低代表資料越一致。



圖：熵概念圖

熵(Entropy)

$$H(T) = - \sum_{i=1}^n P(t_i) \log_2 P(t_i)$$

- $H(T)$: 某項特徵值的熵值
 - 數值皆為 0 到 1 之間
 - 資料編碼大多數都是二進制，所以取對數時要以 2 為底數
- n : 特徵值的總數
- p_i : 某個類別在此特徵值出現的機率

上述公式只適用於單一特徵的亂度，假如要計算所有子集合資料亂度需要使用以下公式。

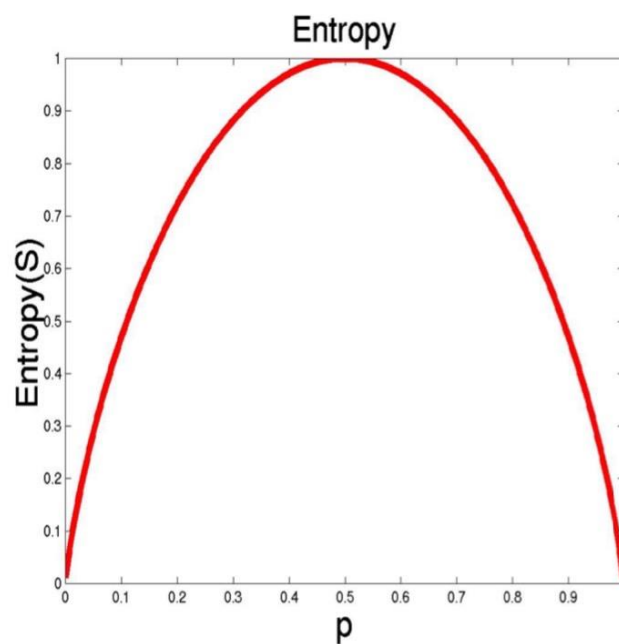
$$H(T|a) = \sum_{v \in \text{vals}(a)} \frac{|S_a(v)|}{|T|} \times H(S_a(v))$$

- $H(T|a)$: T 資料集經過 a 特徵分類之後所求的亂度
- $H(S_a(v))$: 經由 a 特徵的 v 特徵值所求出的亂度
- $\frac{|S_a(v)|}{|T|}$: 子資料集數占 T 資料集總數的比率(權重)

資訊獲利(Information Gain)

$$IG(T, a) = H(T) - H(T|a)$$

- $IG(T, a)$: 未經由 a 特徵分類的亂度減去資訊獲利
 - 原始亂度減去經由 a 特徵的 v 特徵值所求出的亂度，稱為「資訊獲利」
 - 值越大，資料越一致，比較優秀的分類特徵
 - 值越小，資料越不一致，比較差的分類特徵
- $H(T)$: 原始亂度
- $H(T|a)$: T 資料集經過 a 特徵分類之後所求的亂度



圖：熵與出現機率分布圖

Play Bottle Cap Baseball

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	Don't Play
Sunny	80	90	True	Don't Play
Sunny	72	95	False	Don't Play
Sunny	69	70	False	Play
Sunny	71	70	True	Play
Overcast	83	78	False	Play
Overcast	64	65	True	Play
Overcast	72	90	True	Play
Overcast	81	75	False	Play
Rain	70	96	False	Play
Rain	68	80	False	Play
Rain	65	70	True	Don't Play
Rain	75	80	False	Play
Rain	71	80	True	Don't Play

圖：建立決策樹範例資訊

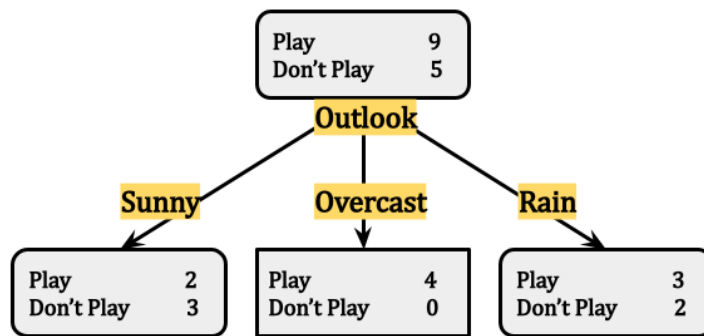
原始亂度

Play	9
Don't Play	5

$$H(T) = - \sum_{i=1}^n P(t_i) \log_2 P(t_i)$$

- $H(T) = -\frac{9}{9+5} \log \frac{9}{9+5} - \frac{5}{9+5} \log \frac{5}{9+5} = 0.940$

第一分類依據(Outlook)



$$H(T) = - \sum_{i=1}^n P(t_i) \log_2 P(t_i)$$

- $H(\text{Sunny}) = -\frac{2}{2+3} \log \frac{2}{2+3} - \frac{3}{2+3} \log \frac{3}{2+3} = 0.971$
- $H(\text{Overcast}) = -\frac{4}{4+0} \log \frac{4}{4+0} - \frac{0}{4+0} \log \frac{0}{4+0} = 0$
- $H(\text{Rain}) = -\frac{3}{3+2} \log \frac{3}{3+2} - \frac{2}{3+2} \log \frac{2}{3+2} = 0.971$

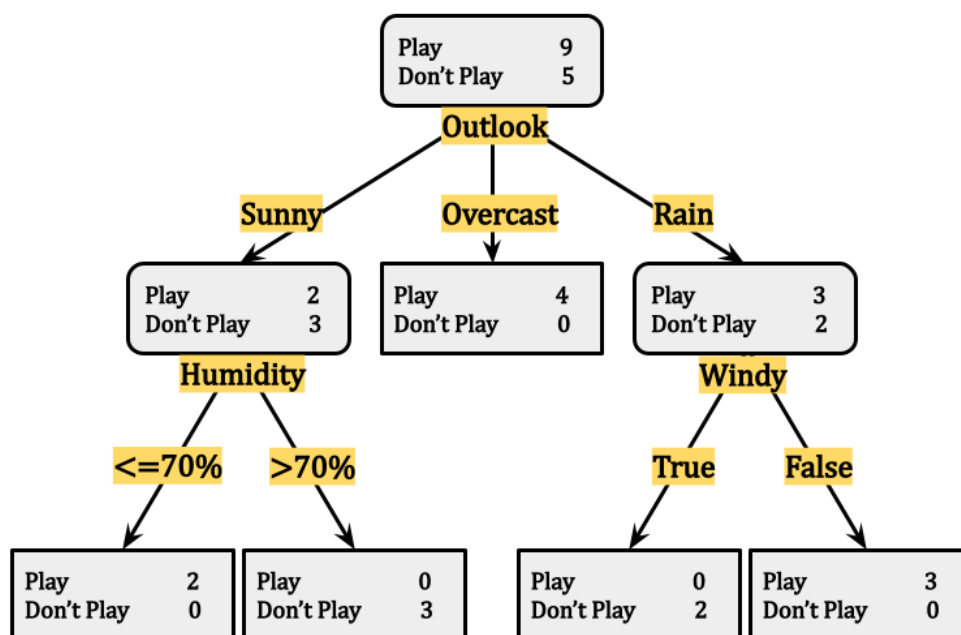
$$H(T|a) = \sum_{v \in \text{vals}(a)} \frac{|S_a(v)|}{|T|} \times H(S_a(v))$$

- $H(T|\text{Outlook}) = \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971 = 0.694$

$$IG(T, a) = H(T) - H(T|a)$$

- $IG(T|\text{Outlook}) = 0.940 - 0.694 = 0.246$

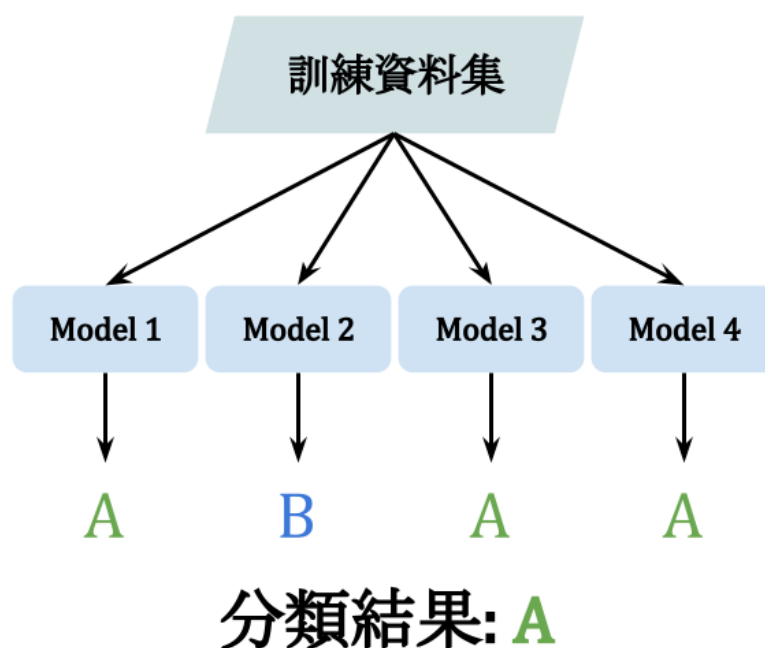
最後分類結果



根據上述計算模型，可以針對 Humidity 和 Windy 求得資訊獲利(省略計算過程)。我們能夠發現兩者的資訊獲利值會最大，代表兩個特徵會將資料分類成一致，最後求得到上圖中的最佳決策樹分類。

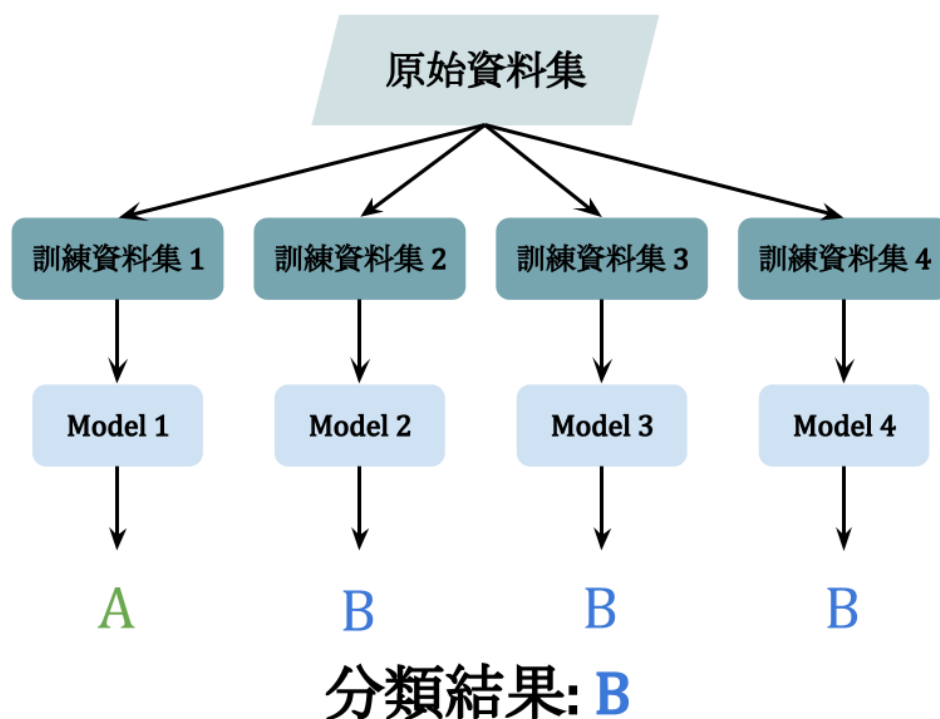
隨機森林(Random Forest)是非監督式學習的一種分類方式，是由多個分類器產生出多個分類結果，並整合所有結果求出「眾數」，此眾數就是分類的最後結果。「隨機」是由原始的資料中，隨機挑選某些資料作為訓練用資料集；「森林」是利用隨機挑選的訓練資料集所建構出的所有分類器總稱。

集成學習(Ensemble Learning)是透過多個獨立學習的分類器模型來解決單一預測問題。當輸入新資料集時，可以藉由多個分類器模型產生的預測結果，整合成單一結果，其結果優於僅靠單一分類器的訓練方式。然而挑選重複的資料集來訓練模型時，會導致分類結果趨於一致性，反而失去了集成學習的優點。因此要採取「重採樣估計」的方式，才能確保每個分類器模型之間具有差異性。



圖：集成學習概念圖

自助採樣(Bootstrap Sampling)是一種隨機取樣的概念，由原始的資料集中，挑選固定個數的資料來作為訓練用資料集，但是在每次挑選完過後，必須將被挑選的資料集放回原始資料集中。自助採樣能夠讓部分資料被重複選取，而讓部分資料從未被選取。如此一來，就不會因為每個資料集無任何交集，導致最後分類結果差異性極大。



圖：自助採樣概念圖

非監督式學習(Unsupervised Learning)

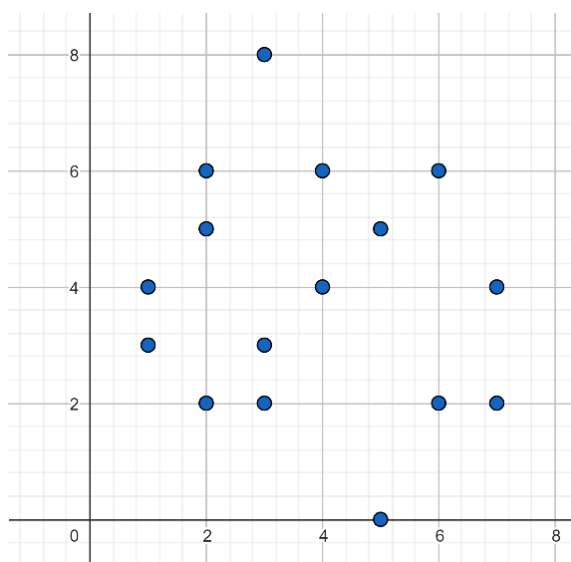
非監督式學習的訓練資料沒有標準答案，機器只能自行摸索、找出潛在的規則進行分類，因此這種學習方式通常用來處理「分群(Clustering)」問題。利用現有的資料特徵分成不同群體，每個群體之間的特徵相似。為了讓資料間的關聯性更加接近，會將特徵值數值化計算資料間的相近程度。

K-平均演算法(K-means Clustering)是先設定要分群的數量，將相近資料彼此分在同一群體，其概念就像是國中數學裡的「重心」，透過公式求得群體間的距離關係，再將資料逐漸分群。

K-means 執行步驟

- 步驟一：設定 K 值，代表接下來要將資料分 K 群
- 步驟二：任意指定座標平面上的 K 個點，作為初始分群中心點
- 步驟三：用「歐幾里得距離」計算座標上各點與初始分群中心點距離
- 步驟四：經由距離關係決定座標點歸屬於哪一個群體中
- 步驟五：根據分群結果以「算術平均」來求得新的分群中心點
- 步驟六：重複步驟三到步驟五，直到分群結果與分群中心點不再變動

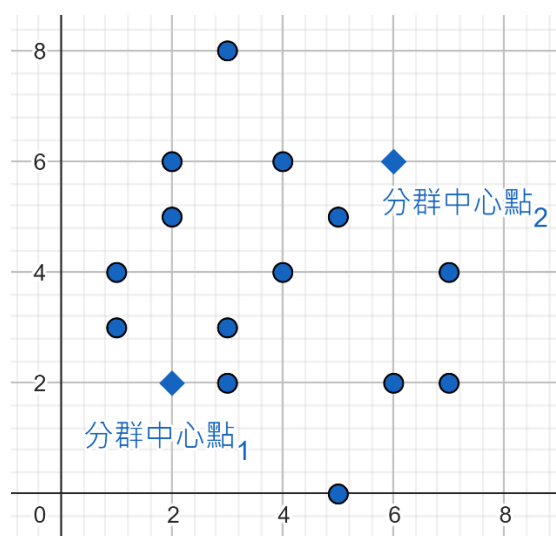
座標上 16 點分別(1, 3)、(1, 4)、(2, 2)、(2, 5)、(2, 6)、(3, 2)、(3, 3)、(3, 8)、(4, 4)、(4, 6)、(5, 0)、(5, 5)、(6, 2)、(6, 6)、(7, 2)、(7, 4)。



圖：座標點分布圖

步驟一與步驟二：

設定 K 值為 2，初始分群中心點為 $(2, 2)$ 和 $(6, 6)$ 。



圖：分群中心點分布圖

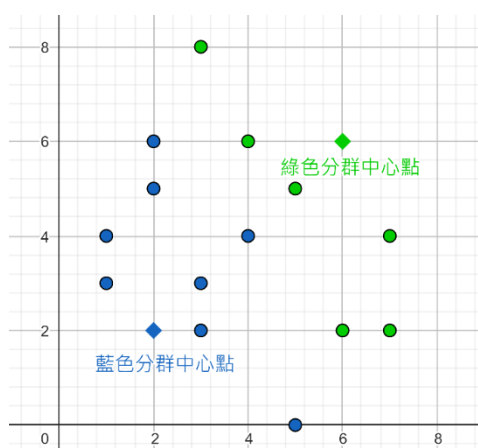
步驟三：

計算座標上各點與初始分群中心點 $(2, 2)$ 和 $(6, 6)$ 距離。

座標(x, y)	與(2, 2)距離	與(6, 6)距離	座標所在歸屬
(1, 3)	1.14	5.83	(2, 2)
(1, 4)	2.24	5.39	(2, 2)
(2, 5)	3	4.12	(2, 2)
(2, 6)	4	4	(2, 2) or (6, 6)
(3, 2)	1	5	(2, 2)
(3, 3)	1.41	4.24	(2, 2)
(3, 8)	6.08	3.61	(6, 6)
(4, 4)	2.83	2.83	(2, 2) or (6, 6)
(4, 6)	4.47	2	(6, 6)
(5, 0)	3.61	6.08	(2, 2)
(5, 5)	4.24	1.41	(6, 6)
(6, 2)	4	4	(2, 2) or (6, 6)
(7, 2)	5	4.12	(6, 6)
(7, 4)	5.39	2.24	(6, 6)

步驟四：

經由距離關係決定座標點歸屬於哪一個群體中，分為藍綠兩群。



圖：步驟四座標點分布圖

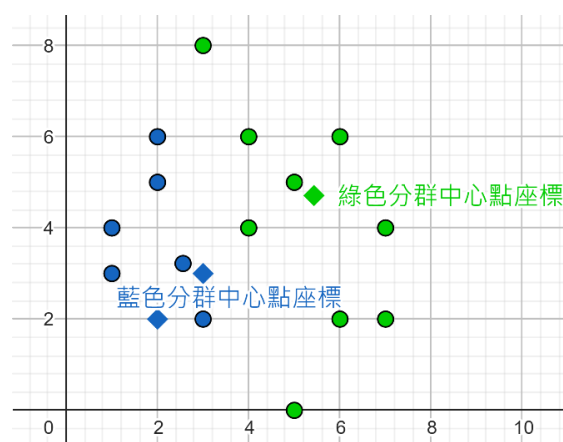
步驟五：

根據分群結果以「算術平均」來求得新的分群中心點(2.56, 3.22)和(5.43, 4.71)。

	(1, 3)、(1, 4)、(2, 2)、 (2, 5)、(2, 6)、(3, 2)、 (3, 3)、(4, 4)、(5, 0)	(3, 8)、(4, 6)、(5, 5)、 (6, 2)、(6, 6)、(7, 2)、 (7, 4)
新分群中心 X	2.56	5.43
新分群中心 Y	3.22	4.71

步驟六：

重複步驟三到步驟五，直到分群結果與分群中心點不再變動。



圖：步驟六座標點分布圖

階層分群法(Hierarchical Clustering)可以動態決定要分群的數量，這裡的「階層」代表分群數量階層，其又有分為兩種方法「聚合法(Bottom-up Clustering)」和「分裂法(Top-down Clustering)」。

「聚合法」概念如同「異中求同」，將所有資料先視作為不同的群，再找最相似的兩群，將其結合為一個新群。重複上述行為，直到聚合後的群數為目標群數。

「分裂法」概念如同「同中求異」，將所有資料先視作為相同的群，再依據群內的相異性，將其拆分為兩個群。重複上述行為，直到分裂後的群數為目標群數。

由於資料數值化的關係，資料會在座標上呈現不均勻分布。如果想要得知點和點、點和群、群和群間的距離關係，就要用到不同的計算距離方式。

中心連結(Centroid-linkage)

$$d(G_1, G_2) = d(\bar{a}, \bar{b})$$

$$\bullet G_1 \in a, G_2 \in b$$

在不同的兩群中，選擇各群的中心點，即為兩群距離

單一連結(Single-linkage)

$$d(G_1, G_2) = \min(a, b)$$

$$\bullet G_1 \in a, G_2 \in b$$

在不同的兩群中，選擇最短距離兩點，即為兩群距離

平均連結(Average-linkage)

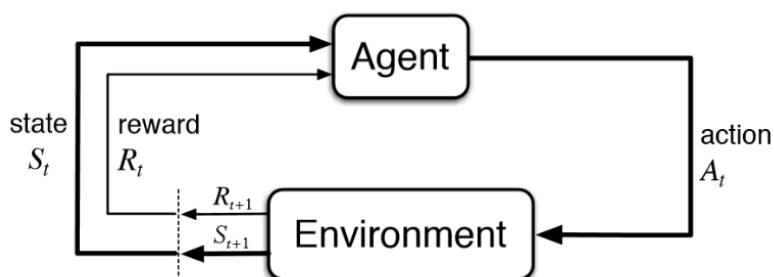
$$d(G_1, G_2) = \frac{\sum d(a, b)}{|G_1||G_2|}$$

$$\bullet G_1 \in a, G_2 \in b$$

在不同的兩群中，各點之距離的平均，即為兩群距離

強化式學習(Reinforcement Learning)

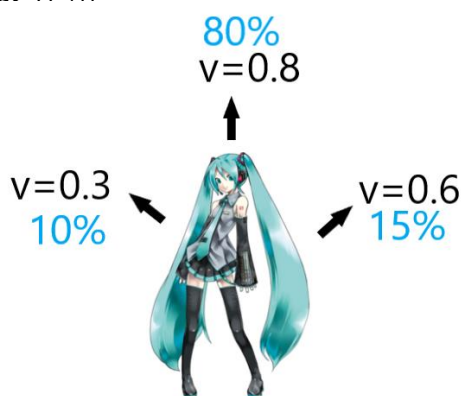
強化式學習是指機器與環境互動過程中，人類不必直接提供解決方案，而是透過電腦不斷嘗試中學習。在訓練中以獎勵機制來強化機器行為，找到最佳的行動策略。




圖：馬可夫決策過程示意圖

- Environment(環境): 根據 Action 給予對應的 Reward
- Agent(代理人): 透過 Action 與 Environment 互動的角色
- Reward(獎勵): Environment 給 Agent 所做 Action 的獎勵懲罰
- State(狀態): Agent 身處的狀態
- Action(行動): Agent 依 Environment 給予的 Reward 做出的決策

不同於演算法中的「貪婪法(Greedy method)」，在每個決策點時只會選擇當下情況的最佳決策。「馬可夫決策過程(Markov decision process)」是決策者在決策過程時，除了選擇當下最佳決策外，也會隨機選擇其他決策。在最佳化問題中會有一個「最佳解(Optimum)」與其他「次佳解(Suboptimal)」的結果，而貪婪法往往只能找到次佳解，但馬可夫決策過程卻可以找到最佳解。



圖：馬可夫決策過程隨機決策概念

			寶藏 (+1)
	懲罰 (-1)		
			


圖：強化式學習中代理人所處環境

假設有一個 初音(Agent) 在 迷宮(Environment) 中，初音依照當下路徑(State) 來選擇走法。初音的目標是要 得到(Action) 寶藏(Reward)並且避開(Action) 懲罰(Reward)，而初音會挑選最短且獎勵最多的路徑來完成目標。


對代理人來說，一個優秀的行動，需要依靠以下兩個因素：

1. 做一個行動，要能夠立刻獲得獎勵。
2. 做一個行動後，轉到下一個狀態時，可能在未來獲得獎勵

環境給予代理人的影響稱為「獎勵(Reward)」，獎勵可能是正值或負值，依照正負大小值來影響代理人的決策。代理人從某個特定狀態開始，預期未來可以取得的獎勵是多少，選擇最大的「值(Value)」作為決策。

	Value = +0.9	
Value = +0.3		Value = +0.6
	Value = -0.3	

圖：值(Value)概念圖


			寶藏 (+1)
	懲罰 (-1)		
			

圖：獎勵(Reward)概念圖

貝爾曼方程(Bellman Equation)

$$V(S) = \max(R(S, a) + \gamma V(S'))$$

- $V(S)$: 初始狀態
- $R(S, a)$: 在當前狀態中，選擇行動的獎勵
- $V(S')$: 根據行動，轉移至下一個新狀態
- γ : 「衰退常數」是控制下個狀態的值對當前狀態值的影響力，通常是用來表示 Reward 對狀態的影響力
 - γ 數值介於 0 到 1 之間
 - γ 數值越大時，代理人更加重視未來獲得的獎勵
 - γ 數值越小時，代理人只在乎目前可獲得的獎勵

$V=0.81_{(3)}$ ↗	$V=0.9_{(2)}$ →	$V=1_{(1)}$ →	寶藏 (+1)
$V=0.73_{(4)}$ ↑	懲罰 (-1)	$V=0.9_{(2)}$	$V=1_{(1)}$
$V=0.66_{(5)}$ ↑	$V=0.73_{(4)}$	$V=0.81_{(3)}$	$V=0.9_{(2)}$
	$V=0.66_{(5)}$	$V=0.73_{(4)}$	$V=0.81_{(3)}$

圖：路徑上各狀態的值($\gamma = 0.9$)

1. 編號(1):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 1 + 0.9 \times 0 = 1$$

寶藏位置前一格值定為 1，因為下一步就已經取得寶藏，所以沒有下一個狀態，故由公式可知此位置的值為 1。

2. 編號(2):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.9 \times 1 = 0.9$$

當前狀態無法獲得寶藏 $R(S, a)$ 為 0，將下一個狀態的值 1 乘上 γ ，再相加得出此位置的值為 0.9。

3. 編號(3):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.9 \times 0.9 = 0.81$$

當前狀態無法獲得寶藏 $R(S, a)$ 為 0，將下一個狀態的值 0.9 乘上 γ ，再相加得出此位置的值為 0.81。

4. 編號(4):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.81 \times 0.9 = 0.73$$

當前狀態無法獲得寶藏 $R(S, a)$ 為 0，將下一個狀態的值 0.81 乘上 γ ，再相加得出此位置的值為 0.73。

5. 編號(5):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.73 \times 0.9 = 0.66$$

當前狀態無法獲得寶藏 $R(S, a)$ 為 0，將下一個狀態的值 0.73 乘上 γ ，再相加得出此位置的值為 0.66。

之前的方法是將值(Value)儲存在各個狀態(State)中，但「Q-Learning」是將值(Value)儲存在各個狀態(State)所對應行動(Action)當中。

Q-Learning 記錄下每個狀態所對應行動的值稱為 Q-value，而許多 Q-Value 所形成的表稱為 Q-table。

Q-Function

$$Q(S, a) = R(S, a) + \gamma \max_{a'} (Q(S', a'))$$

- $Q(S, a)$: 在當前狀態下執行行動的 Q-value
- $R(S, a)$: 在當前狀態中，選擇行動的獎勵
- $Q(S', a')$: 根據行動，轉移至下一個新狀態的 Q-Value
- γ : 「衰退常數」是控制下個狀態的值對當前狀態值的影響力，通常是用來表示 Reward 對狀態的影響力

$$V(S) = \max_a (R(S, a) + \gamma V(S'))$$

將 $R(S, a) + \gamma V(S')$ 提出



$$Q(S, a) = R(S, a) + \gamma V(S')$$

修改 $V(S')$



$$Q(S, a) = R(S, a) + \gamma \max_{a'} (Q(S', a'))$$

圖：Q-Function 導出圖

Q-Table

	a_1	a_2	a_3
S_1	$Q(S_1, a_1)$	$Q(S_1, a_2)$	$Q(S_1, a_3)$
S_2	$Q(S_2, a_1)$	$Q(S_2, a_2)$	$Q(S_2, a_3)$
.....

圖：Q-Table 示意圖

要將 Q-Table 中的 Q-Value 更新優化，就要使用「時值差異(Temporal difference)」這個方法。

為什麼不直接用 Q-Function 求出新的 Q-value? 原因是直接求出來的 Q-Value 可能不是最好的，會有高估或低估的情況。最後造成 Q-Value 量值忽大忽小來回跳動，這不是希望看到的情況。

時值差異(Temporal difference)

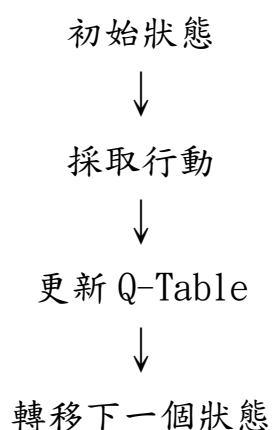
$$\begin{aligned} T(S, a) &= R(S, a) + \gamma \max(Q(S', a')) - Q(S, a) \\ &= Q'(S, a) - Q(S, a) \end{aligned}$$

將已經更新過的 Q-Value 減去原本的 Q-Value，可求得兩者之間的差值。得到的時值差異，可以用來更新 Q-Table 中的 Q-Value。

$$Q_t(S, a) = Q_{t-1}(S, a) + \alpha TD(S, a)$$

- α : 「學習率」決定了新獲得的資訊對現有 Q-Value 的影響程度。
 - α 數值介於 0 到 1 之間。
 - α 數值越大時，時值差異學習的幅度越大，下一步的 Q-Value 影響也越大。

在某個狀態下執行行動時，其 Q-Value 變化在原有的 Q-Value 中，又會參照時值差異來推估新的 Q-Value。



圖：Q-Learning 流程圖