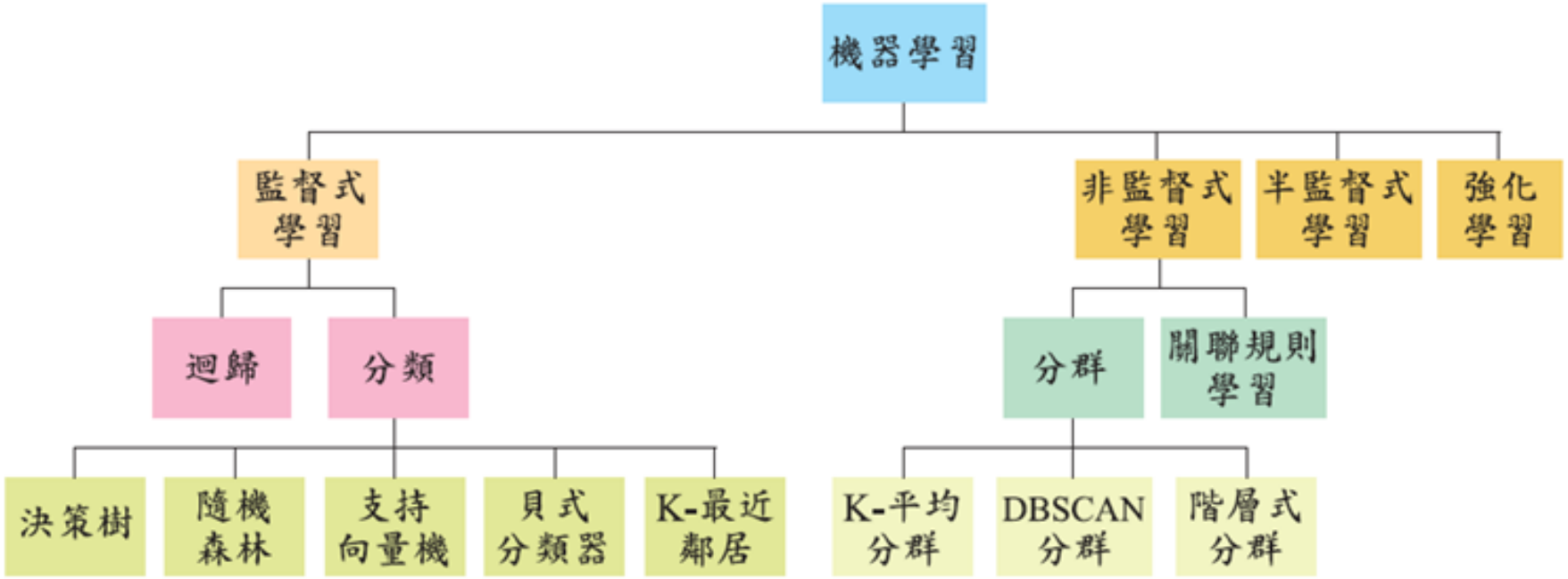
機器學習簡介

- 機器學習(Machine Learning)是一種數據分析技術，它教導計算機模仿人類從經驗中學習。

- Clustering -聚類
- Classification -分類
- Regression -回歸

機器學習簡介

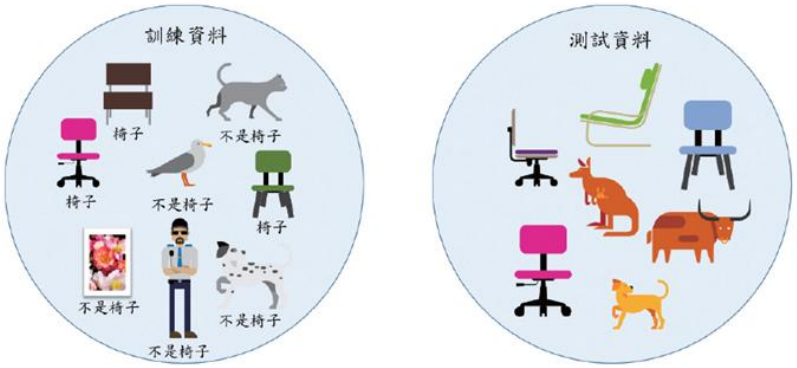
- 機器學習的分類



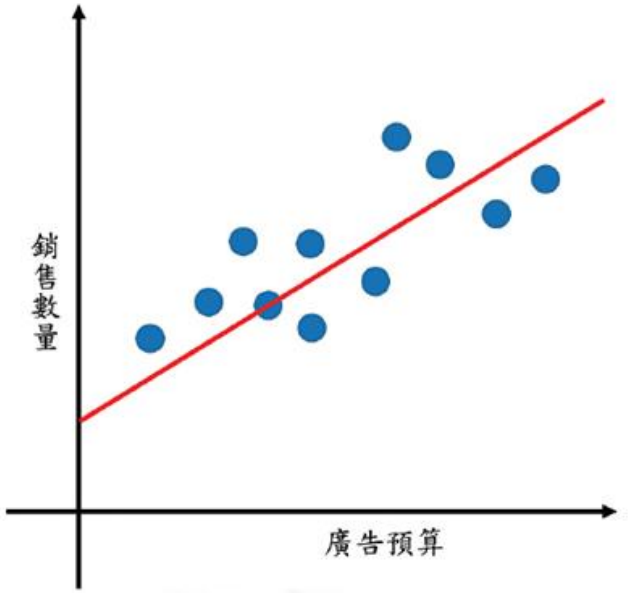
機器學習的分類

監督式學習

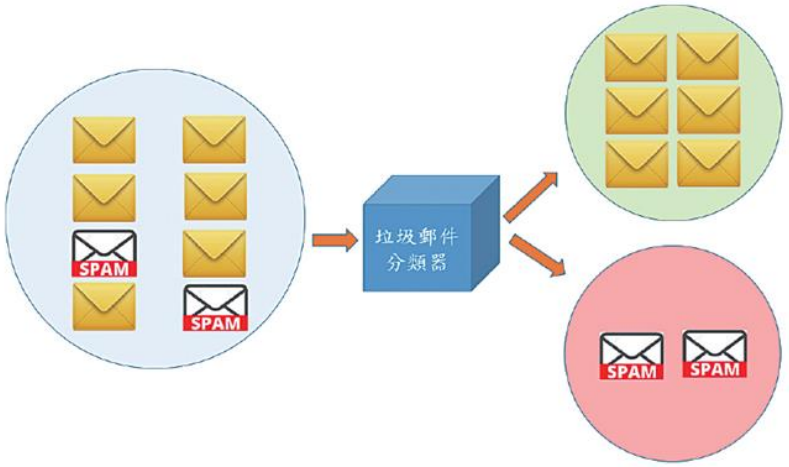
- 監督式學習（Supervised learning）



用於分類的標記訓練集



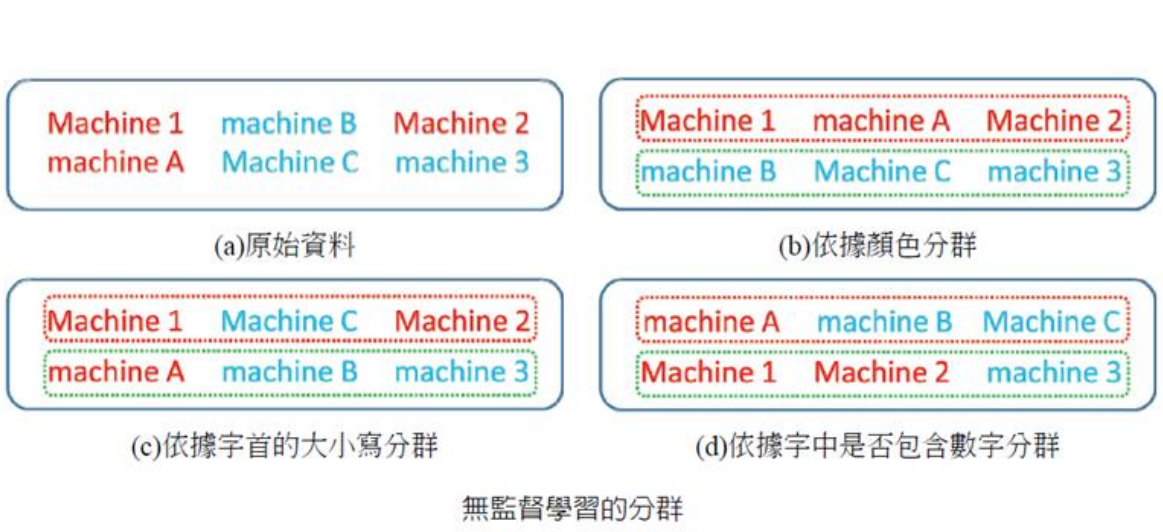
使用廣告預算來預估汽車的銷售量

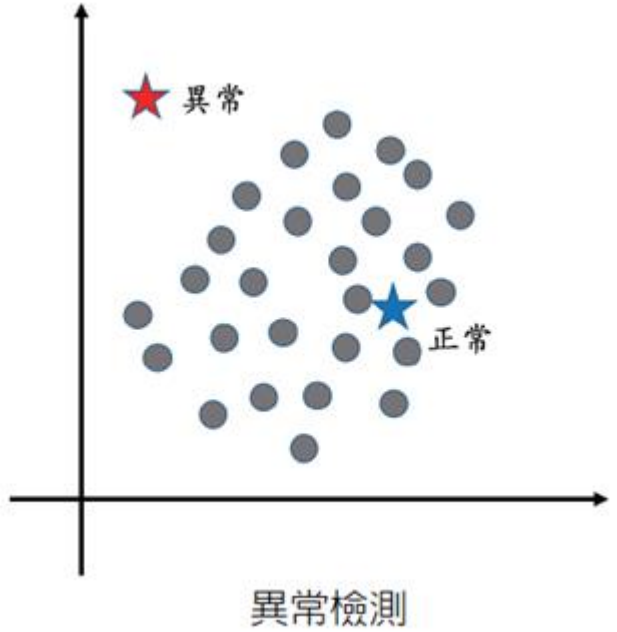


用於監督學習的垃圾郵件分類

非監督式學習

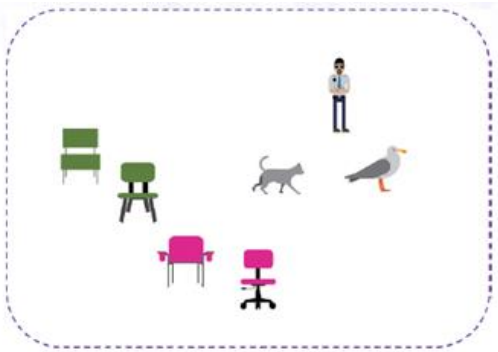
- 非監督式學習（Un-Supervised learning）



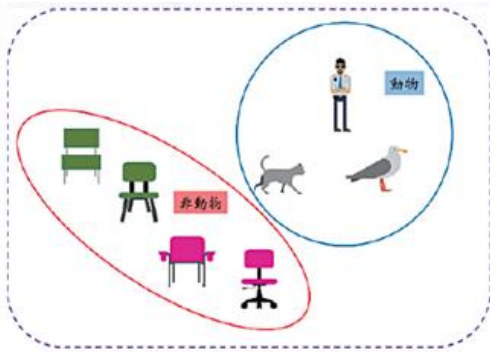


非監督式學習

- 非監督式學習（Un-Supervised learning）

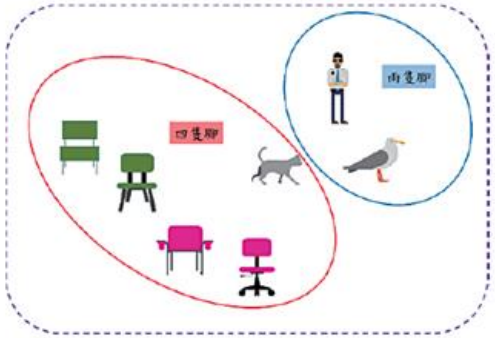


(a)原始資料集

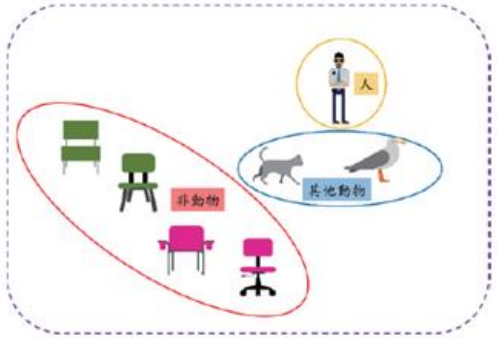


(b)依據是否爲動物來分成二群

不同的分群結果

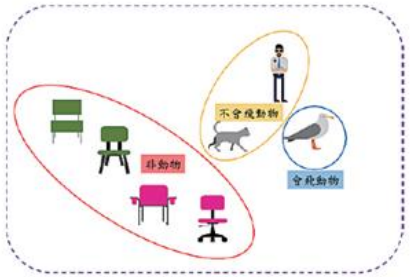


(c)依據腳的數量來分成兩隻腳和四隻腳兩群



(d)依據人、其他動物和非動物分成三群

不同的分群結果 (續)



(e)依據會飛動物、不會飛動物和非動物分成三群

(f)依據會飛動物、不會飛動物、綠色椅子和紅色椅子分成四群

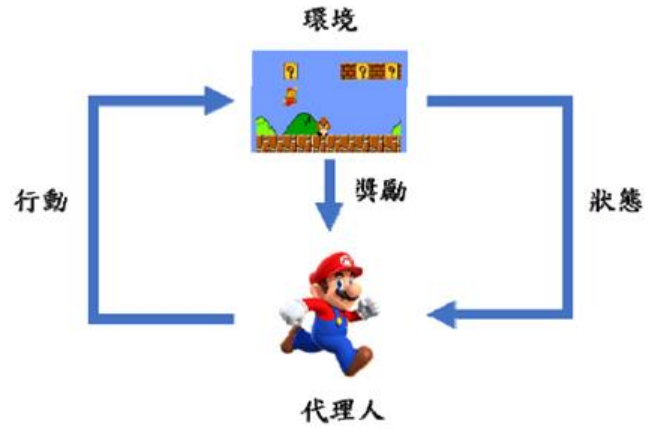
不同的分群結果

半監督式學習

- 半監督式學習（Semi-supervised learning）
  - 半監督式學習顧名思義就是結合監督式學習與非監督式學習的一種方法。
  - 在實際的應用中，有些領域人工標示的樣本成本很高，但是沒有標籤的樣本數量卻非常龐大。
  - 所以半監督式學習就是利用大量的無標籤樣本和少量的標籤樣本來解決標籤樣本不足的問題。

強化學習

- 強化學習（Reinforcement learning）

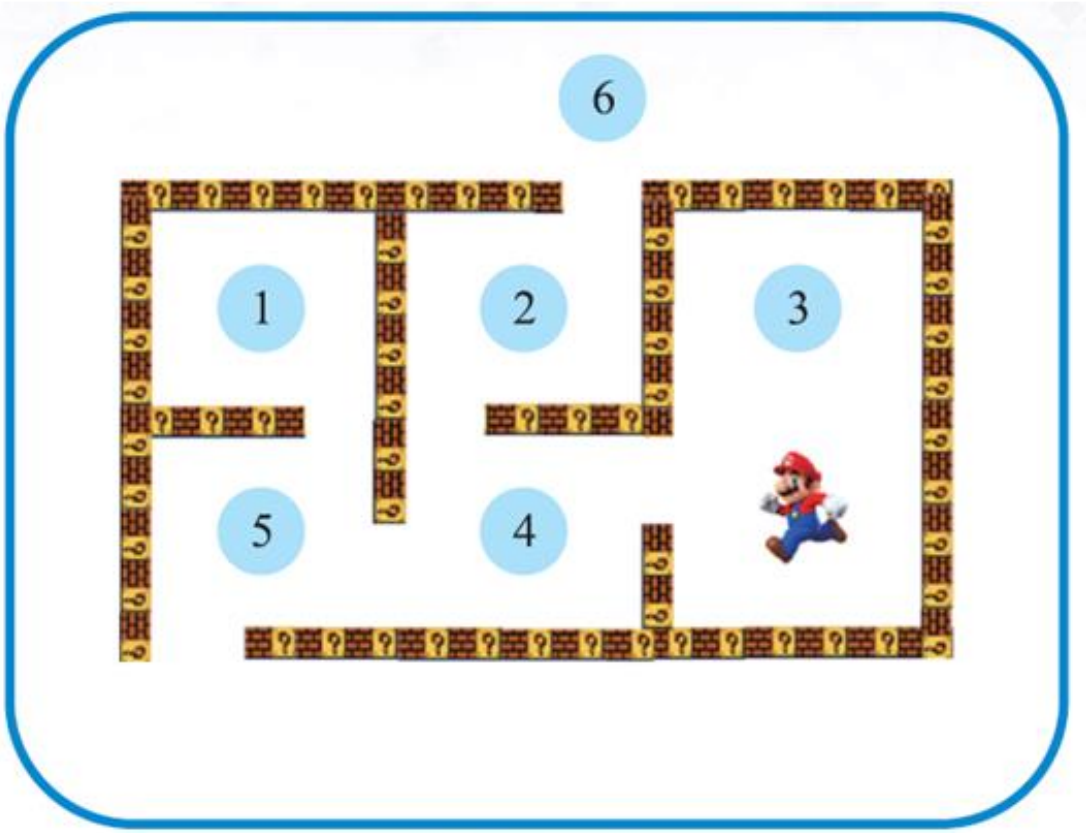


強化學習架構

AlphaGo

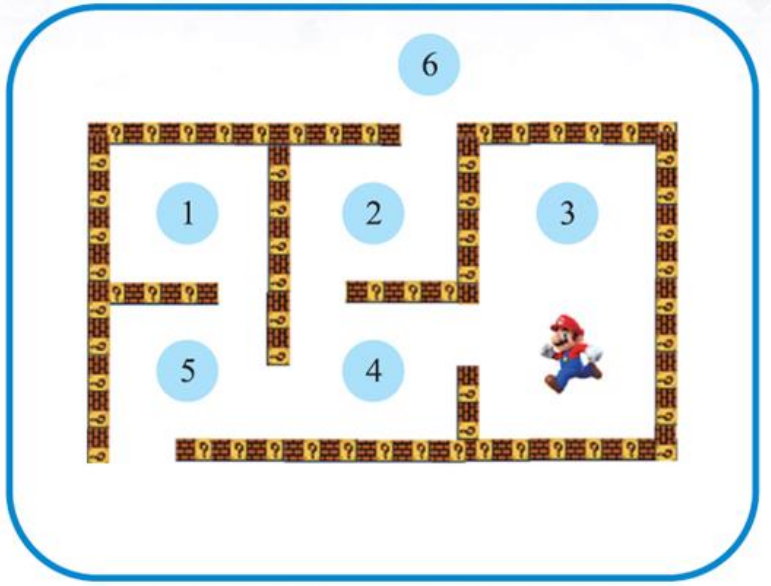
強化學習（Q-learning）

- 強化學習之 Q-learning

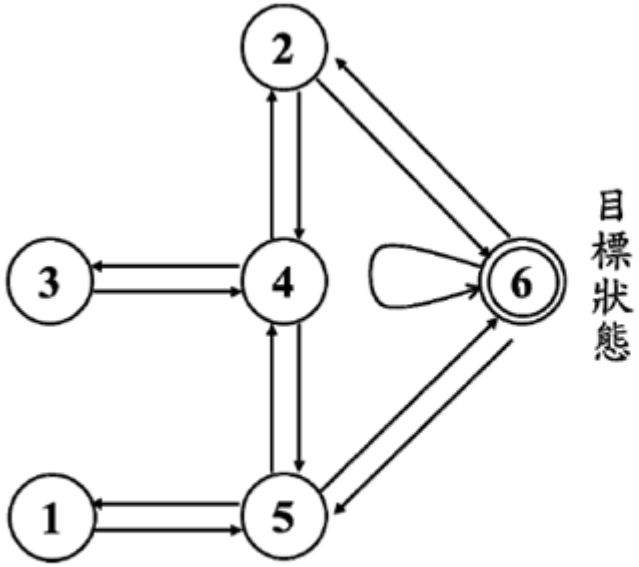


迷宮的地圖

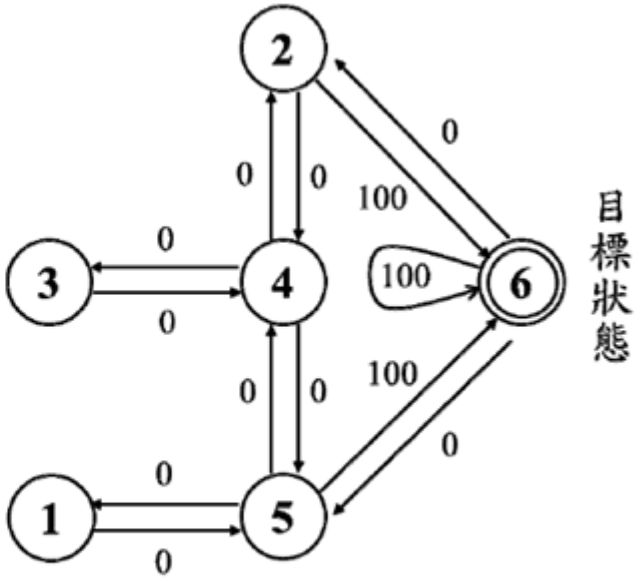
強化學習（Q-learning）



迷宮的地圖

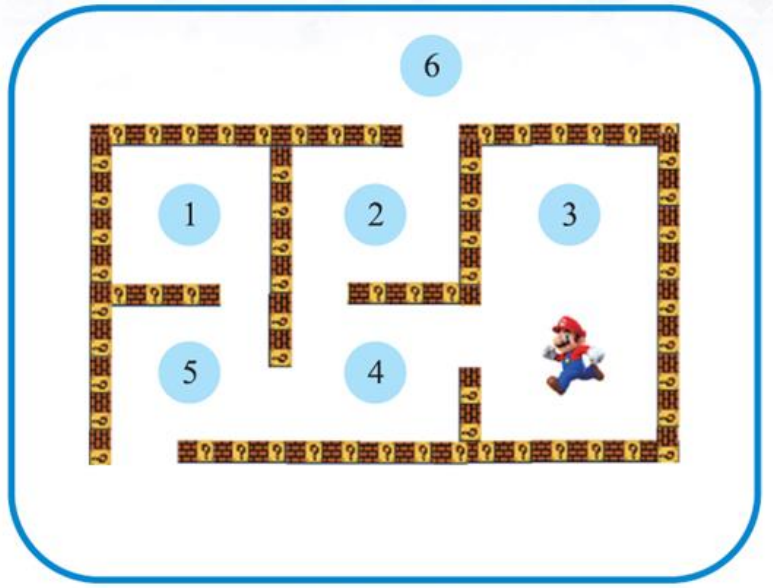


以圖來表示迷宮地圖



標上獎勵的圖

強化學習（Q-learning）



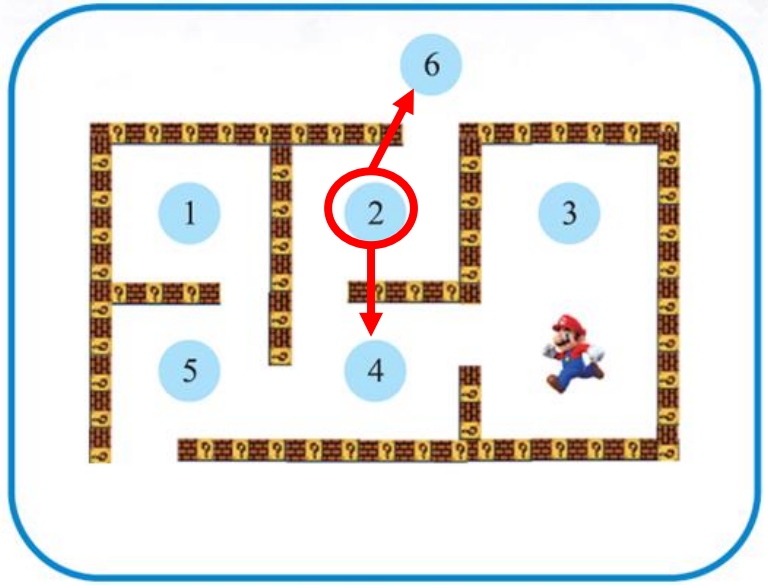
迷宮的地圖

Q(狀態，行動)=R(狀態，行動)+折扣因子*Max[Q(下個狀態，所有行動)]

行:行動
列:狀態

$$R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}$$

強化學習（Q-learning）



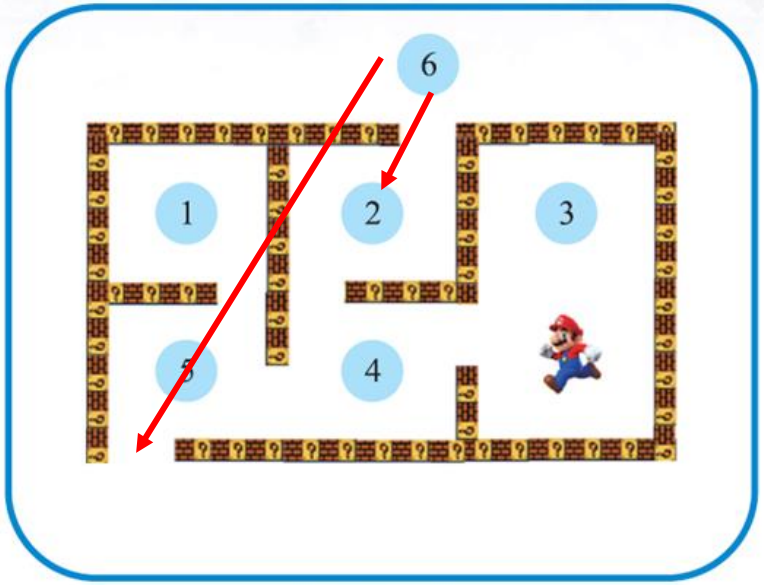
迷宮的地圖

Step1.
折扣因子=0.8
初始狀態=2
Q矩陣=0
假設選到行動=6

Q(狀態，行動)=R(狀態，行動)+折扣因子*Max[Q(下個狀態，所有行動)]

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}$$

強化學習（Q-learning）



迷宮的地圖

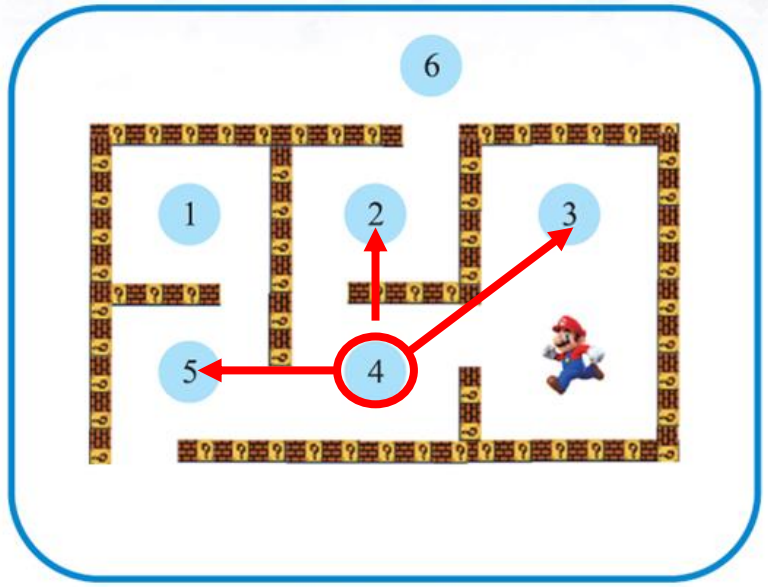
Step2.
計算Q(2，6)

$$Q(2,6)=R(2,6)+0.8*\text{Max}[Q(6,2),\ Q(6,5),\ Q(6,6)]$$
$$=100+0.8*0$$
$$=100$$

$$Q=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&100\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}$$

$$R=\begin{bmatrix}-1&-1&-1&-1&0&-1\\-1&-1&-1&0&-1&100\\-1&-1&-1&0&-1&-1\\-1&0&0&-1&0&-1\\0&-1&-1&0&-1&100\\-1&0&-1&-1&0&100\end{bmatrix}$$

強化學習（Q-learning）



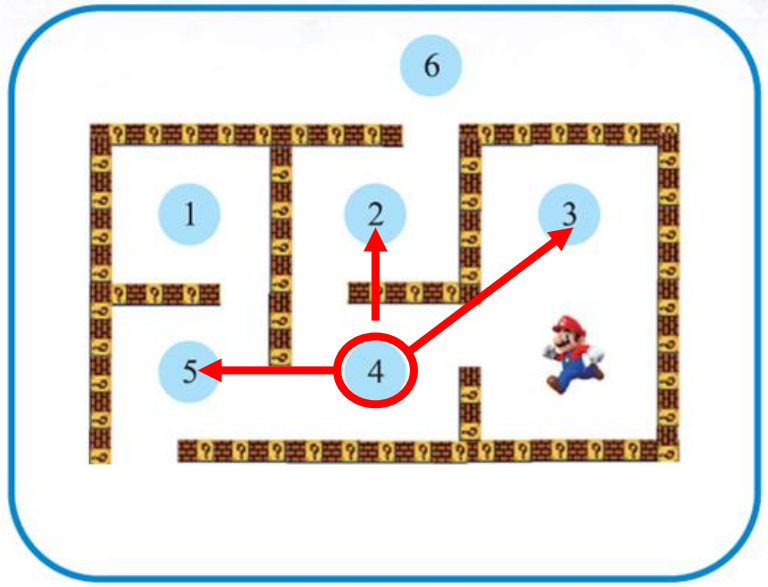
迷宮的地圖

Step1.
折扣因子=0.8
初始狀態=4
假設選到行動=2

Q(狀態，行動)=R(狀態，行動)+折扣因子*Max[Q(下個狀態，所有行動)]

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}$$

強化學習（Q-learning）



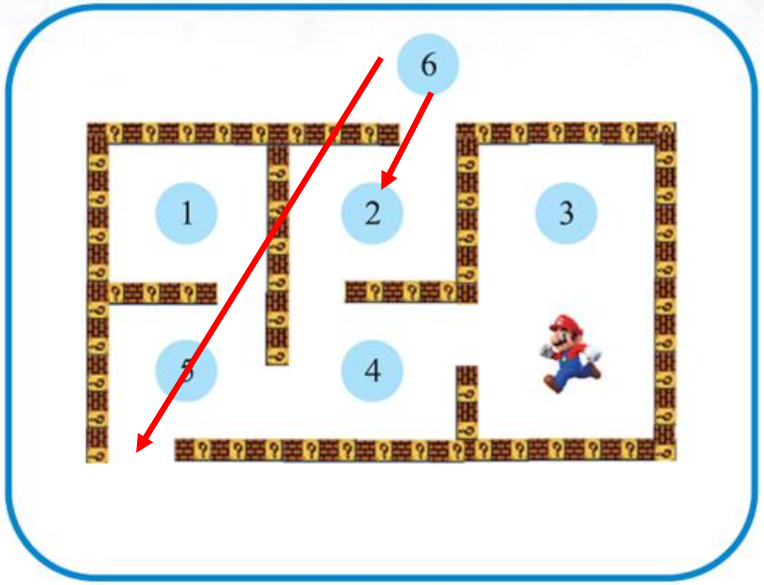
迷宮的地圖

Step2.
計算Q(4，2)

Q(4，2)=R(4，2)+0.8*Max[Q(2，4), Q(2，6)]
=0+0.8*100
=80

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}$$

強化學習（Q-learning）



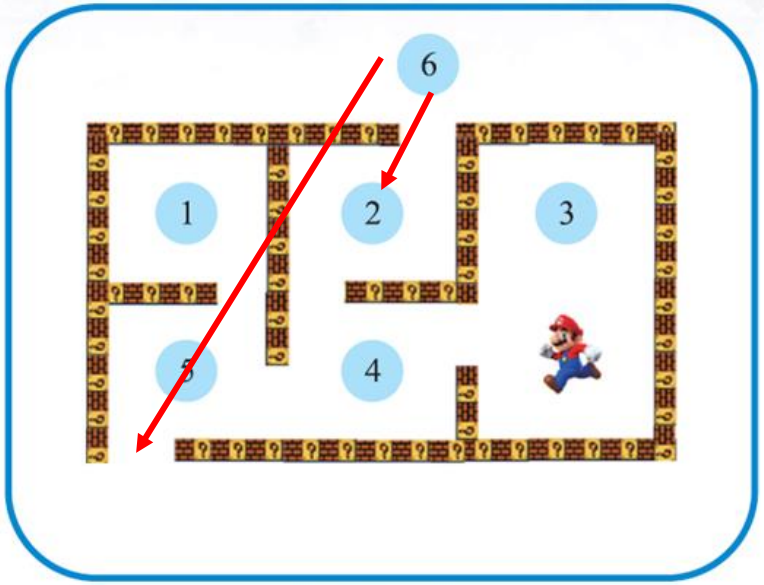
迷宮的地圖

Step3.
折扣因子=0.8
狀態=2
假設選到行動=6

Q(狀態，行動)=R(狀態，行動)+折扣因子*Max[Q(下個狀態，所有行動)]

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix}$$

強化學習（Q-learning）



迷宮的地圖

Step4.
計算Q(2，6)

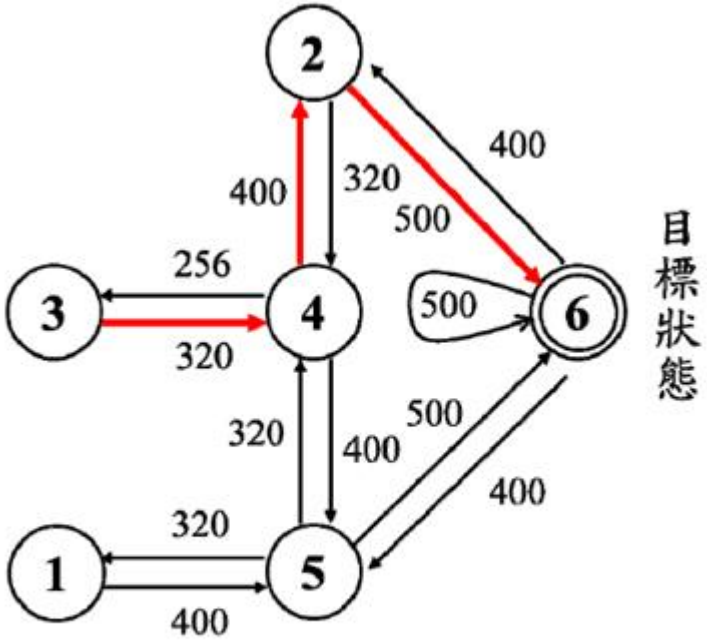
Q(2，6)=R(2，6)+0.8*Max[Q(6，2), Q(6，5) , Q(6，6)]
=100+0.8*0
=100

$$Q=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&100\\0&0&0&0&0&0\\0&80&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}$$

$$R=\begin{bmatrix}-1&-1&-1&-1&0&-1\\-1&-1&-1&0&-1&100\\-1&-1&-1&0&-1&-1\\-1&0&0&-1&0&-1\\0&-1&-1&0&-1&100\\-1&0&-1&-1&0&100\end{bmatrix}$$

強化學習（Q-learning）

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix}$$



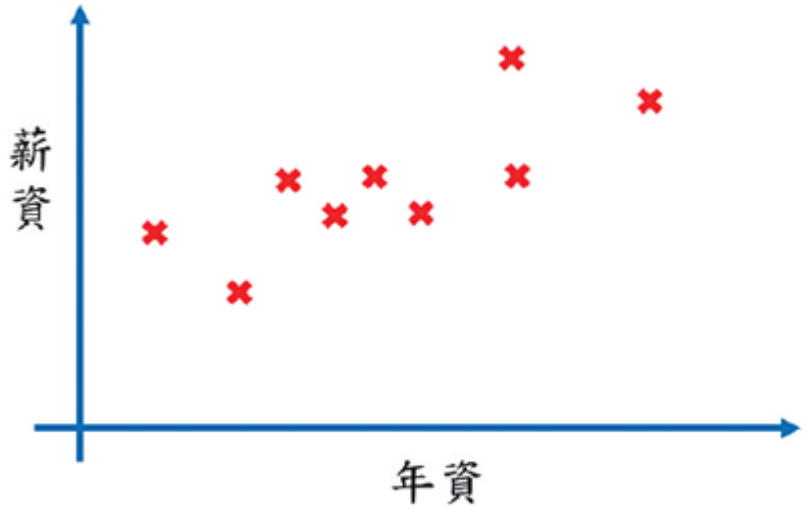
Q-learning 後的狀態行動獎勵圖

機器學習演算法

- 迴歸（Regression）

年資和薪資的資料集

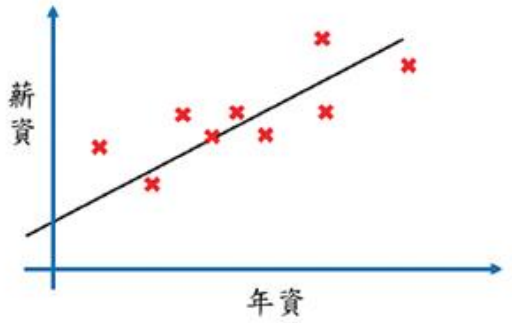
年資	薪資
1	30000
2	25000
3	35000
3.5	32000
4	35000
4.5	32000
6	35000
6	50000
7	40000



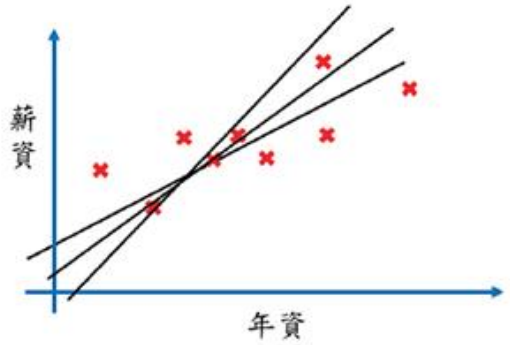
年資和薪資的二維分佈圖

迴歸（Regression）

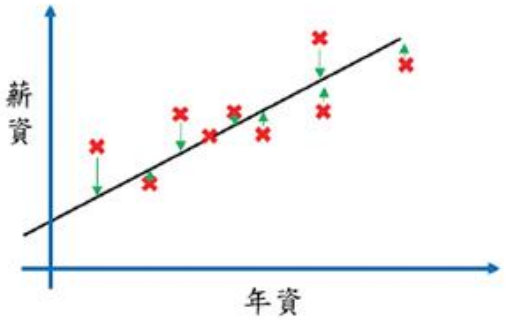
- 迴歸預測



(a)迴歸直線方程式



(b)無數條的直線方程式

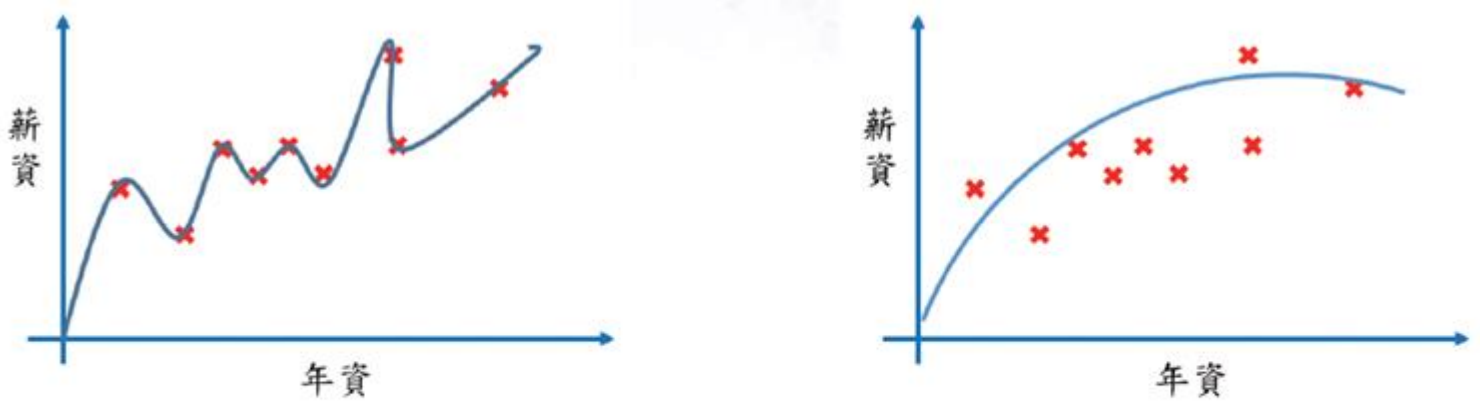


(c)誤差平方和爲最小的直線

迴歸預測

迴歸（Regression）

- 非線性函數

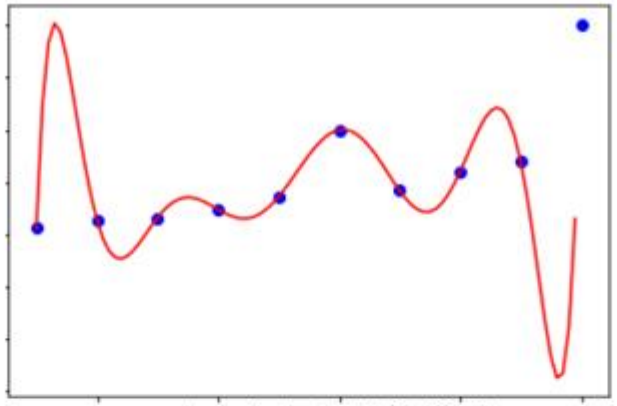


(a)二次多項式的函數線　(b)高次多項式的函數線

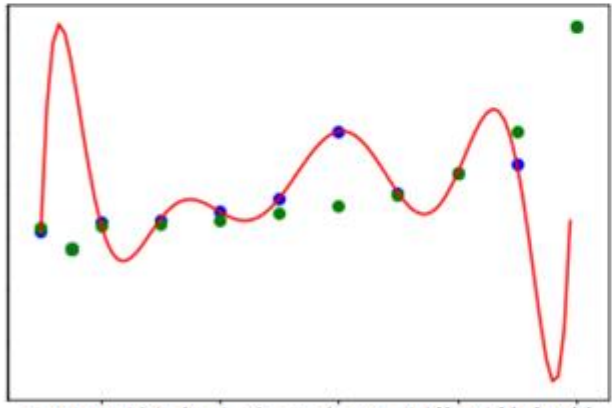
非線性函數

迴歸（Regression）

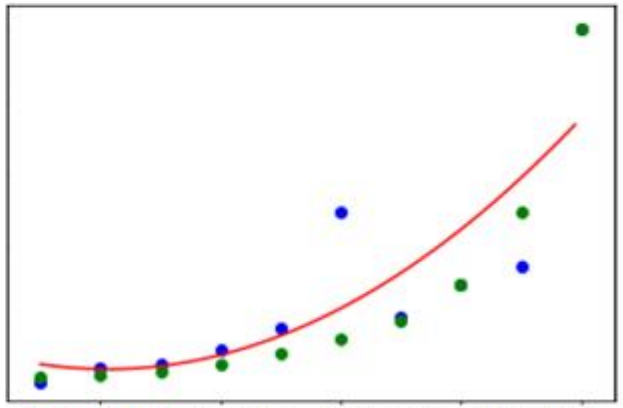
- 預測模型



(a)十次方多項式的模型



(b)預測值在十次方多項式模型的趨勢



(c)預測值在二次方多項式模型的趨勢

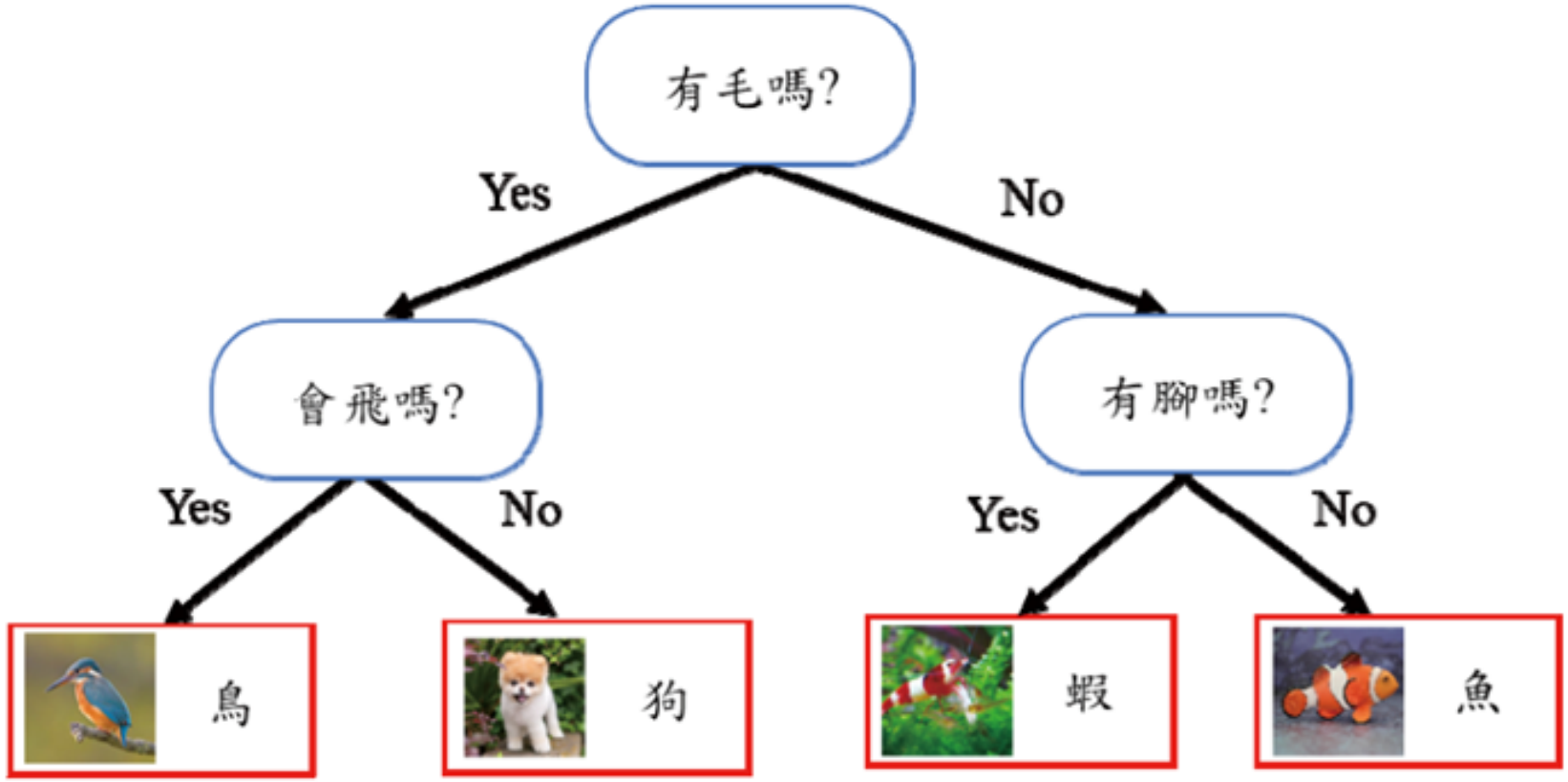
預測模型

迴歸（Regression）

以下的範例是Boston的房價資料集。

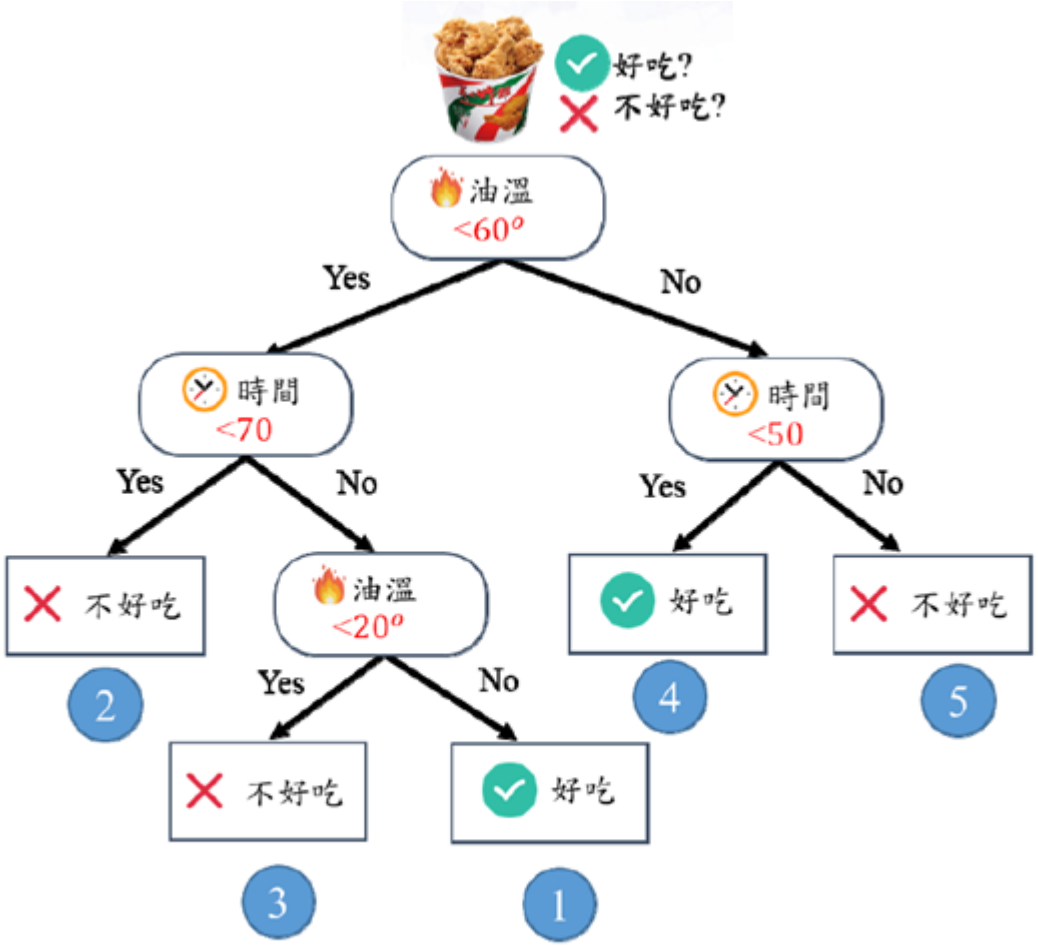
- 共有14個特徵欄位，506筆資料。
- CRIM：人均犯罪率。
- ZN：住宅用地超過25,000平方英呎的比例。
- INDUS：城鎮非零售商用土地的比例。
- CHAS：是否鄰近查爾斯河(1：是、0：否)。
- NOX：一氧化氮濃度。
- RM：住宅的平均房間數。
- AGE：1940年之前建造的自用房屋比例。
- DIS：到波士頓五個中心區域的加權距離。
- RAD：到達高速公路的方便性指標。
- TAX：每萬元的全價值房屋稅。
- PTRATIO：城鎮的師生比例。
- B：1000*(Bk-0.63)**2，(Bk：城鎮中黑人的比例)。
- LSTAT：低收入人口的比例。
- MEDV：自住房屋的平均房價(單位：千元美金)。

決策樹（Decision Tree）



分辨何種動物的決策樹

決策樹（Decision Tree）

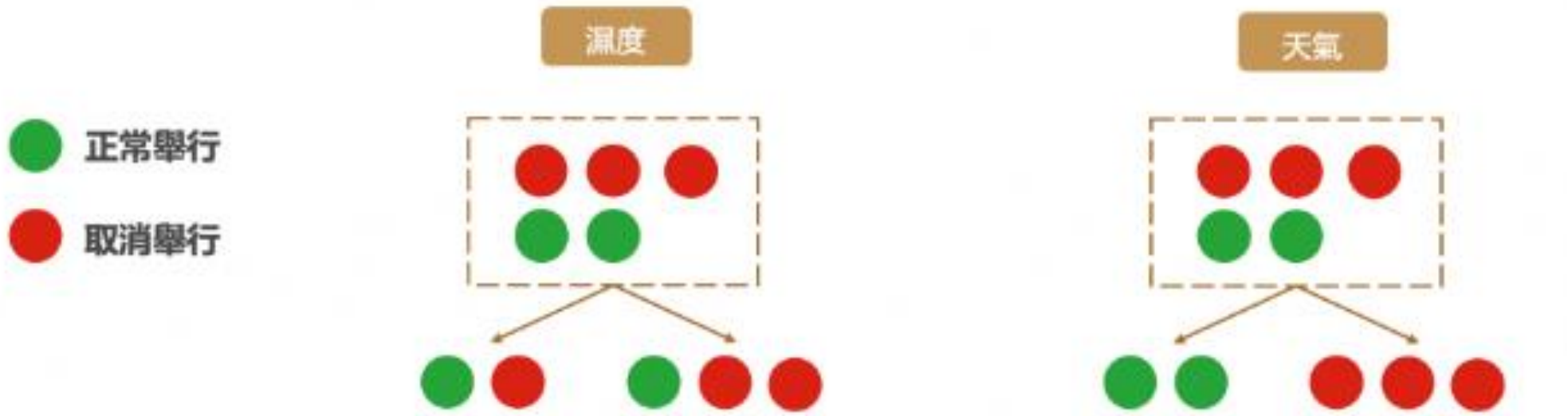


判斷炸雞是否好吃的決策樹

炸雞的數據

編號	油溫	油炸時間	好不好吃
1	50	80	好
2	45	60	不好
3	19	100	不好
4	100	30	好
5	100	70	不好
6	70	30	?

決策樹（Decision Tree）



決策樹（Decision Tree）

- 分割原則-資訊增益(Information gain, 簡稱IG)
  - 熵(Entropy)
  - Gini不純度(Gini Impurity)

資訊增益

獲得的資訊量　原本的資訊量　經由分割後的資訊量

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

二元分類資訊增益

獲得的資訊量　原本的資訊量　分割後左邊資訊量　分割後右邊資訊量

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

決策樹（Decision Tree）

- 分割原則-資訊增益(Information gain, 簡稱IG)
- 評估分割資訊量

$$Entropy = -\sum_j p_j \log_2 p_j$$

$$Gini = 1 - \sum_j p_j^2$$

決策樹（Decision Tree）

- 分割原則-資訊增益(Information gain, 簡稱IG)
- 熵 (Entropy)
  - 計算Information Gain 的一種方法

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

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

p：是的機率　q：否的機率

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

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

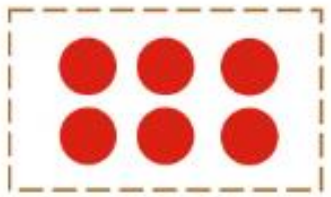
決策樹（Decision Tree）

- 分割原則-資訊增益(Information gain, 簡稱IG)
- Gini 不純度 (Gini Impurity)
  - 另外一種計算Information Gain 的方法

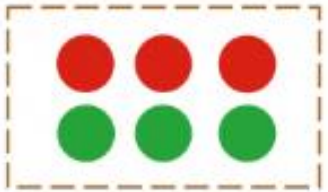
$$\text{Gini} = 1 - \sum p_j^2$$

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

p：是的機率　q：否的機率



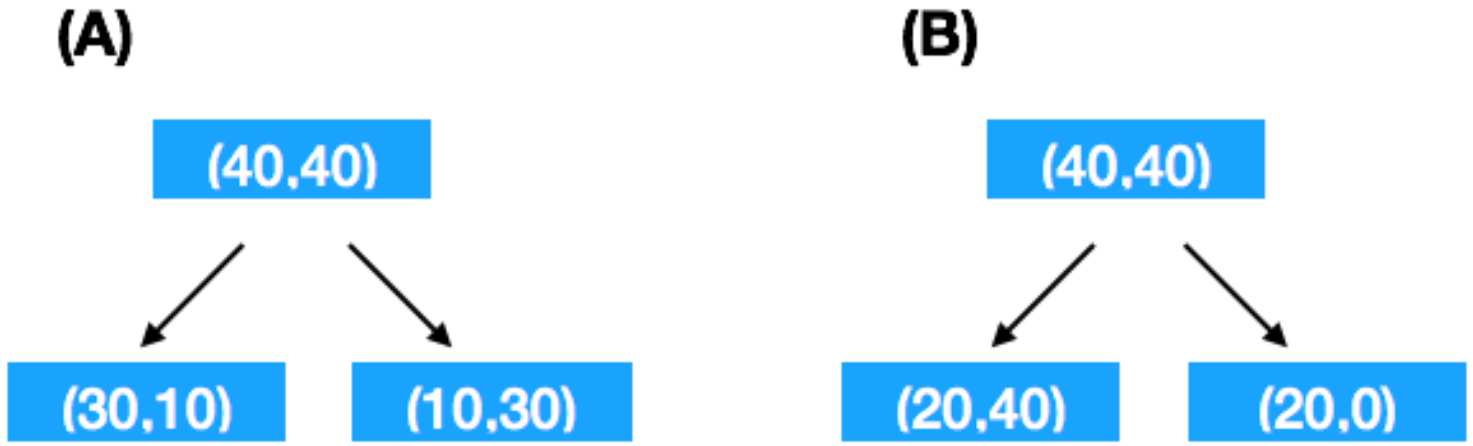
$\text{Info}(6, 0) = 1 - (1^2 + 0^2) = 0$



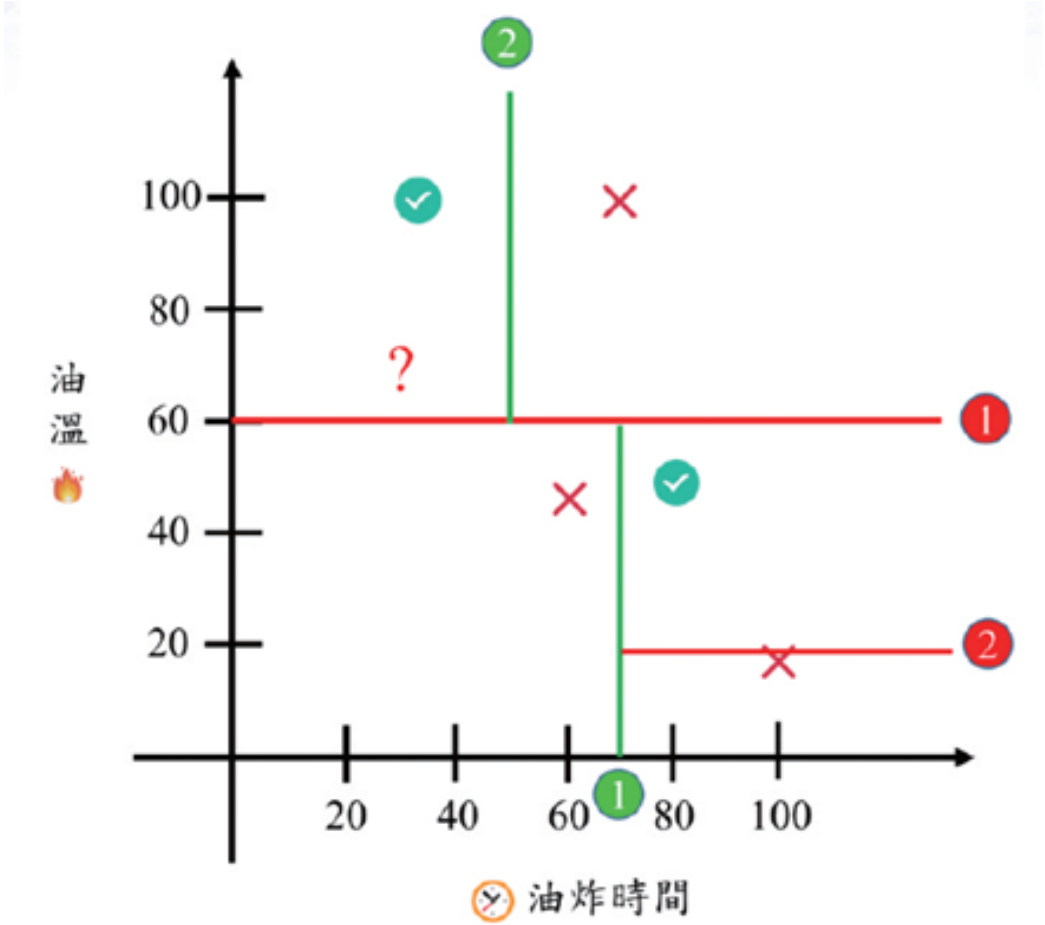
$\text{Info}(3, 3) = = 1 - (0.5^2 + 0.5^2) = 0.5$

決策樹（Decision Tree）

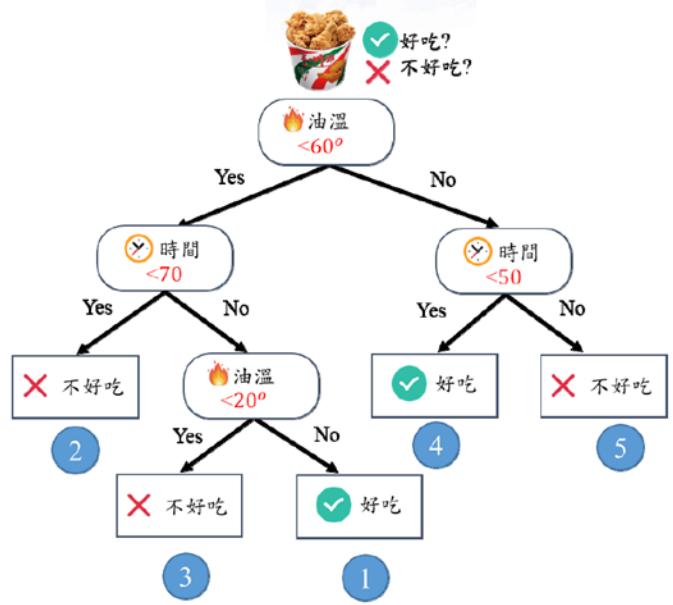
- 分割原則-資訊增益(Information gain, 簡稱IG)
- 有80筆資料，有40是1類別、40筆是2類別。使用兩種不同的切割方法A與B



決策樹（Decision Tree）



以油溫和油炸時間來分類



判斷炸雞是否好吃的決策樹

決策樹（Decision Tree）

優點

- 簡單且高度可解釋性
- 低計算時間複雜度
- 每個決策階段都相當的明確清楚
- 幾乎沒有要調整的超參數

缺點

- 模型容易過度擬合
- 當標籤類別種類多時樹會很複雜

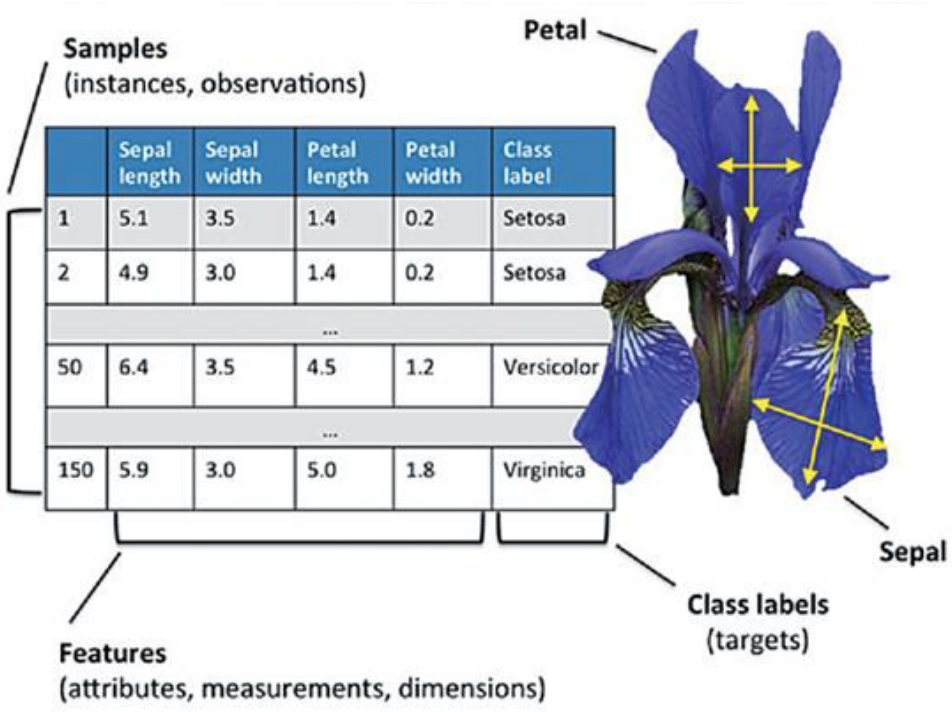
決策樹（Decision Tree）

範例練習

- 以下的範例是鳶(ㄩㄢ)尾花的分類資料集。
- 共有4個欄位，150筆資料。
  - sepal length (cm)：花萼的長度。
  - sepal width (cm)：花萼的寬度。
  - petal length (cm)：花瓣的長度。
  - petal width (cm)：花瓣的寬度。

決策樹（Decision Tree）

- Setosa
- Versicolor
- Virginica



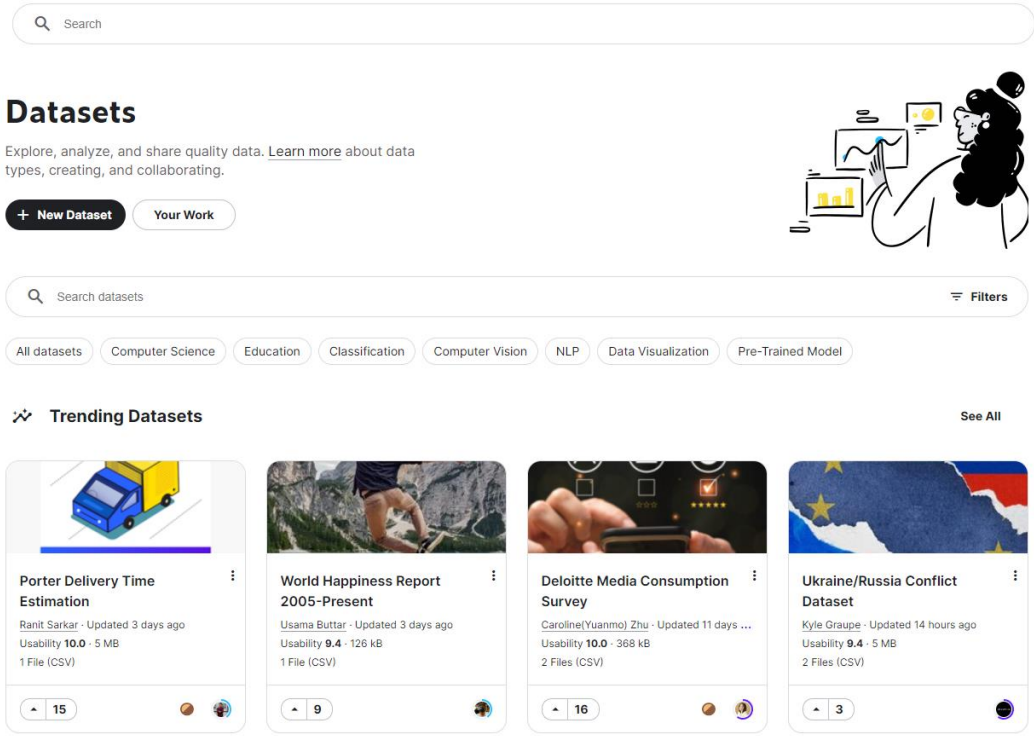
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

鳶尾花的花瓣長寬和花萼長寬

- Entropy:使用每種特徵分類後的資訊量，資訊量越多就越優先作為決策條件
- Gini Impurity:指的是分類器的分錯機率，越小代表錯誤機率越小故選為決策的條件。

資料集怎麼來?

Kaggle Dataset
scikit-learn



sklearn.datasets : Datasets ¶

The `sklearn.datasets` module includes utilities to load datasets, including methods to load and fetch popular reference datasets. It also features some artificial data generators.

User guide: See the Dataset loading utilities section for further details.

Loaders

`datasets.clear_data_home` ([data_home])	Delete all the content of the data home cache.
`datasets.dump_svmlight_file` (X, y, f[, ...])	Dump the dataset in svmlight / libsvm file format.
`datasets.fetch_20newsgroups` ([data_home, ...])	Load the filenames and data from the 20 newsgroups dataset.
`datasets.fetch_20newsgroups_vectorized` ([...])	Load the 20 newsgroups dataset and transform it into tf-idf vectors.
`datasets.fetch_california_housing` ([...])	Loader for the California housing dataset from StatLib.
`datasets.fetch_covtype` ([data_home, ...])	Load the covertype dataset, downloading it if necessary.
`datasets.fetch_kddcup99` ([subset, data_home, ...])	Load and return the kddcup 99 dataset (classification).
`datasets.fetch_lfw_pairs` ([subset, ...])	Loader for the Labeled Faces in the Wild (LFW) pairs dataset
`datasets.fetch_lfw_people` ([data_home, ...])	Loader for the Labeled Faces in the Wild (LFW) people dataset
`datasets.fetch_mldata` (dataname[, ...])	Fetch an mldata.org data set
`datasets.fetch_olivetti_faces` ([data_home, ...])	Loader for the Olivetti faces data-set from AT&T.
`datasets.fetch_rcv1` ([data_home, subset, ...])	Load the RCV1 multilabel dataset, downloading it if necessary.
`datasets.fetch_species_distributions` ([...])	Loader for species distribution dataset from Phillips et.
`datasets.get_data_home` ([data_home])	Return the path of the scikit-learn data dir.
`datasets.load_boston` ([return_X_y])	Load and return the boston house-prices dataset (regression).
`datasets.load_breast_cancer` ([return_X_y])	Lo sklearn.datasets e breast cancer wisconsin dataset (classification).
`datasets.load_diabetes` ([return_X_y])	Load and return the diabetes dataset (regression).
`datasets.load_digits` ([n_class, return_X_y])	Load and return the digits dataset (classification).
`datasets.load_files` (container_path[, ...])	Load text files with categories as subfolder names.
`datasets.load_iris` ([return_X_y])	Load and return the iris dataset (classification).
`datasets.load_linnerud` ([return_X_y])	Load and return the linnerud dataset (multivariate regression).
`datasets.load_mlcomp` (*args, **kwargs)	DEPRECATED: since the http://mlcomp.org/ website will shut down in March 2017, the load_mlcomp function was deprecated in version 0.19 and will be removed in 0.21.
`datasets.load_sample_image` (image_name)	Load the numpy array of a single sample image
`datasets.load_sample_images` ()	Load sample images for image manipulation.

Iris dataset

鳶尾花資料集是非常著名的生物資訊資料集之一，取自美國加州大學歐文分校的機器學習資料庫。
http://archive.ics.uci.edu/ml/datasets/Iris