



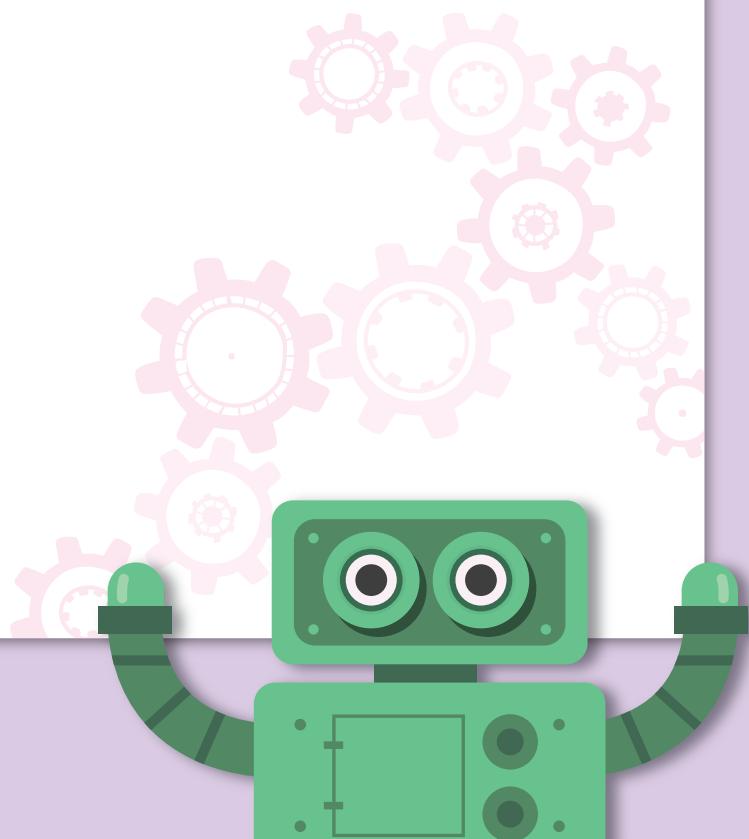
CH

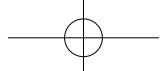
3



動手實作監督式學習

- 3-1 迴歸與分類
- 3-2 房價秒預測—線性迴歸介紹與應用
- 3-3 乳癌機率有多高—KNN 分類器介紹與應用
- 3-4 傑克與蘿絲誰的生存機率高—決策樹分類器介紹與應用
- 3-5 總結





3-1 迴歸與分類

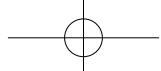
在經過前兩章的介紹後，相信讀者對於人工智慧已經有了初步的認識。承第一章 1-4 節，我們知道人工智慧的核心技術之一就是**機器學習（Machine Learning）**，從圖 1-13 中也可以了解到機器學習在人工智慧的發展過程中，具有承先啓後的重要地位。而本章就要帶領大家一起透過動手實作的歷程來體驗機器學習，了解電腦是如何透過資料分析協助人們做出決策，甚至達到預測的目的。在接下來的章節中，將揭開機器學習的神秘面紗。

機器學習，是指人類期待機器可以像人一般，透過自主性的學習產生相對應的決策能力。電腦科學家參考了許多人類的行為，以及認知神經科學、心理學、生物學等專家們的建議，他們反向思考著：『既然要讓機器像人類一樣學習，那人類究竟是怎麼學習的呢？』而這個問題，讀者也可以試著回想自我的成長與學習經驗。根據俄國的心理學家維高斯基（Vygotsky）所提出的**近側發展區理論（Zone of Proximal Development）**指出：『兒童在成長過程中，若有一個導師（如：父母、教師、長輩）可以協助他嘗試能力以外的事物，透過不斷修正錯誤的歷程與反饋就能加快其成長的速度』。而這就是我們要在本章介紹給讀者的**監督式學習法**。

在監督式學習領域中，常會看到**迴歸（Regression）**與**分類（Classification）**這兩個名詞。專家們在利用人工智慧技術解決問題之前，首要考量的通常是：『究竟眼前的問題是屬於迴歸還是分類問題？』以下，我們將帶領大家從這兩個名詞的分流渠道，逐步走進監督式學習的世界中，並一起透過實際操作來探究其中的奧妙。

3-1-1 迴歸（Regression）

如果我們期盼解決問題的方式是預測出一個實際值，譬如預測溫度、房價、PM2.5 含量等情況時，就將此類問題統稱為迴歸問題。舉例來說，讀者可以參考一些數據資料（如不動產實價登錄網站）來估算心目中某棟理想房屋的價格區間，它可能是 1,000 ~ 1,500 萬之間。而在監督式學習領域中，導入大量的房屋數據資料（如屋齡、坪數大小、建材等）就可利用迴歸分析的方法，去估算一個



相對精準的數值，它可能是 1,392 萬，而這樣的精確數值我們可透過實際成交的價錢去做比對，衡量當初預估值的準確性高低。最常見的迴歸演算法是**線性迴歸**（**Linear Regression**），或許讀者對於這個名詞並不陌生，它其實就是國中基礎數學中直線方程式（ $Y=aX+b$ ）的衍生應用。

3-1-2 分類 (Classification)

如果讀者遇到了想要判定某些事物的類別時，那麼在監督式學習中就統稱為分類問題。有別於上述的迴歸所產生的精確數值，分類所產生出來的結果是類別**標籤**（**Label**）。例如，想要從照片上判斷正在進食的究竟是何種動物時，透過分類演算法所得到的結果可能是貓（類別 A）或是狗（類別 B），這就是類別標籤。也因為分類問題所產生的結果並非精確數值，所以最終衡量預測結果時，只有答對或答錯這兩種情況。常見的分類演算法有 KNN、決策樹等，本章後面的小節也將陸續介紹。

表 3-1 是一個生活中常見的情境判定範例，讀者可以透過這些情境來加深對於迴歸或分類問題的觀念識別。

◎表 3-1 回歸與分類題目的比較

回歸問題 (估算一個預測值)	分類問題 (判別有限類別中的某一種類)
自駕車預估抵達目的地所需時間	自駕車判定是否該立即停車
教學系統預估學生期末考的數學成績	教學系統判定學生是否達到畢業標準
氣象系統預估明日 PM2.5 的數值	氣象系統判定明天是否下雨



想想看

下列哪些是迴歸問題？哪些是分類問題？

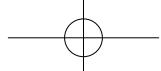
問題 1：明年小華會不會交女朋友？

問題 2：下個月白菜一斤多少錢？

問題 3：影片中的動物是熊還是老虎？

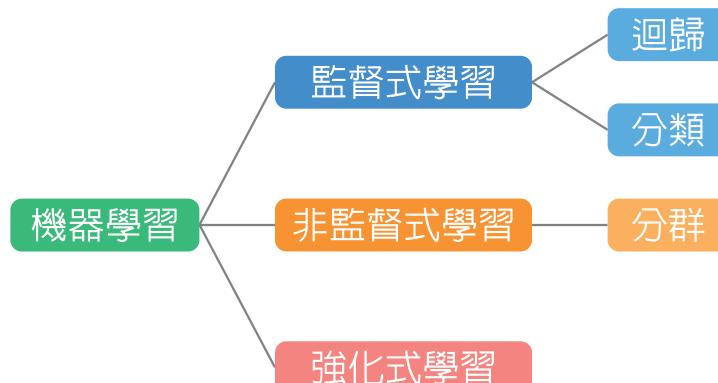
問題 4：明天下午三點的風向是東風、南風、西風，還是北風？

問題 5：今晚睡眠的時間長度？



經過上述的介紹與練習後，相信讀者對於迴歸與分類問題，都應具備一定程度的理解了。進一步，我們必須知道的是，無論迴歸或分類，想透過演算法讓機器進行學習，那麼就必須給予有正確答案的**標註資料（Labeled Data）**。

如同心理學家維高斯基（Vygotsky）的理論，若有一位導師可以從旁告知機器「對」或「錯」，讓機器從錯誤中不斷修正，進而加速學習的效率，我們就稱這種需要標準答案的學習法為**監督式學習（Supervised Learning）**。反之，另一種只需要訓練資料，而不需要導師給予正確答案或指導，透過類似反覆自行觀察資料的方式，產生調適與同化的學習法，則稱為**非監督式學習（Unsupervised Learning）**。

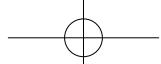


◆ 圖 3-1 機器學習領域分支概念圖

3-2 房價秒預測—線性迴歸介紹與應用

監督式學習中的迴歸問題，對應解決問題的演算法最常見的就是**線性迴歸（Linear Regression）**。接下來，我們將透過一個實際案例來實作**簡單線性迴歸（Simple Linear Regression）**模型。

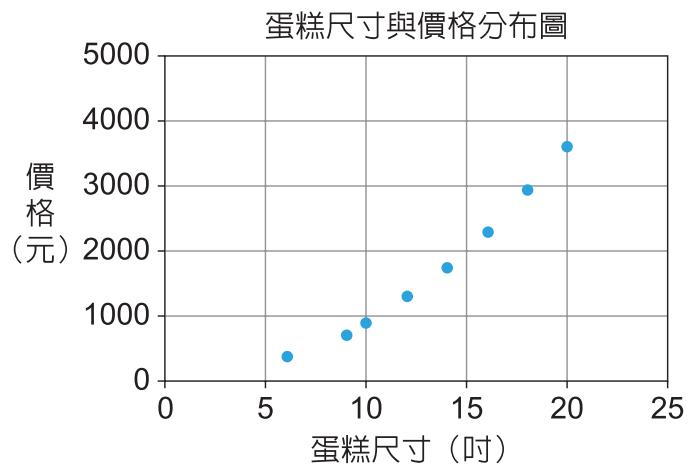
表 3-2 是台灣某知名蛋糕店的芋頭蛋糕尺寸價目表，其中尺寸的大小會影響價格高低，我們可以將表 3-2 透過數據轉換的方式，轉繪在直角坐標平面上，觀察兩個變數（尺寸、價格）之間的關係。



◎表 3-2 蛋糕尺寸與價格對照表

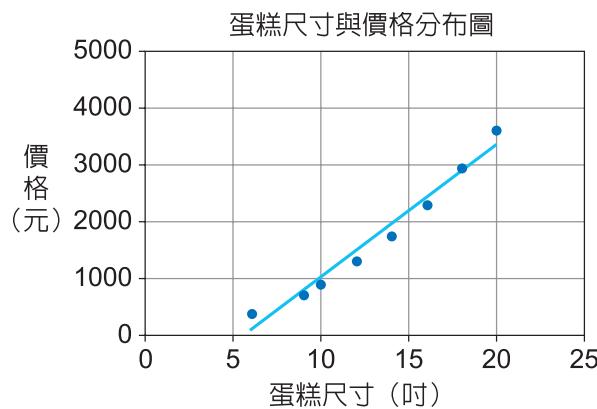
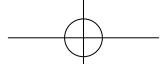
訓練資料	蛋糕尺寸 (吋)	價格 (元)
1	6	380
2	9	700
3	10	900
4	12	1300
5	14	1750
6	16	2300
7	18	2950
8	20	3600

這組資料集，設定蛋糕尺寸為 x 值，價格為 y 值，以座標表示法 (x, y) 即可繪製直角坐標圖，圖 3-2 就是這八筆資料的分布圖，圖中的 X 軸是表示蛋糕尺寸 (Size)，Y 軸則是代表對應的價格 (Price)。

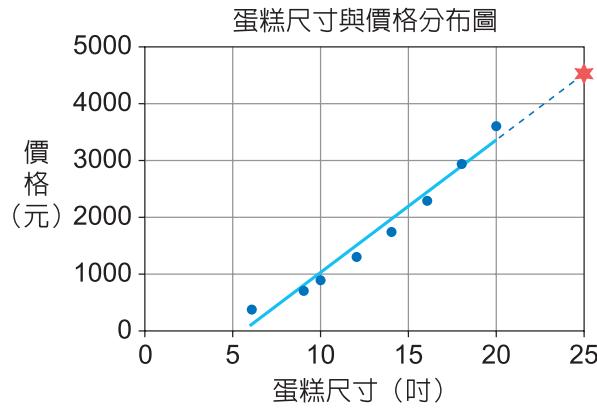


◆ 圖 3-2 蛋糕尺寸與價格分布圖

從圖 3-2 中可以清楚觀察到尺寸與價格之間是呈現正相關的關係，蛋糕尺寸越大，其價格就越高，這也與我們生活中的經驗相符合。圖 3-3 是透過數學方法所繪製出的迴歸直線（繪製方法會在實作課程中學習）。透過這條線的建立，讀者應該有發現，所有資料點的散佈位置幾乎都緊靠著這條直線，我們似乎可以透過這條直線來做後續相關數值的推估，譬如蛋糕店老闆接獲一個 25 吋的特製蛋糕訂單，那麼它的價格就可以透過這條迴歸直線來進行估價，如圖 3-4 中紅色星號所示，其預估價格可能為 4,500 元左右。



◆ 圖 3-3 蛋糕尺寸與價格分布圖



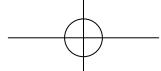
◆ 圖 3-4 蛋糕尺寸與價格分布圖

3-2-1 Scikit Learn (Sklearn) 函式庫的介紹

本書所有的程式都是在 Google 的 Colab 平台上進行，讀者可以透過在導讀頁的程式碼連結，直接點開 Colab 的 Python notebook，一行一行執行程式實作，每一行程式，都以清楚的註解加以說明，不用擔心 Python 程式碼看不懂。（Google Colab 相關使用教學可參考附錄 B）。

在上個小節，我們對監督式學習的迴歸問題有了初步認識後，為了更深入地熟練使用方法與時機，以下內容我們將開始 Python 程式實作，透過實際案例的處理，來加深相關知識與應用的連結能力。

目前在機器學習領域的程式套件大都已經相當成熟，我們不需要所有功能都自己從零開發，這些被撰寫整理好的程式集或稱函式庫大多也都是免費開源，



可以直接取用。這一節將介紹一個在機器學習、資料科學領域扮演重要角色的工具庫 scikit-learn（在程式中使用函式庫時，名稱是 sklearn），這是一個利用 Python 程式語法撰寫整理的函式庫，幾乎涵蓋了大部份的機器學習演算法與實作功能，廣受初學者歡迎。以下我們將針對 scikit-learn 做基本的介紹，以利讀者應用。

Scikit Learn 官網

scikit-learn 官網的網址為 <https://scikit-learn.org/stable/index.html>。進入官網後，讀者可以看到頁面分成六個區塊（如圖 3-5）：

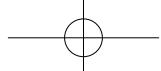


1. Classification：分類（監督式學習的分類模型套件）。
2. Regression：迴歸（監督式學習的迴歸模型套件）。
3. Clustering：分群（非監督式學習法，將在本書第四章介紹）。
4. Dimensionality reduction：降維（非監督式學習，減少維度去除不需要的特徵）。
5. Model selection：模型選擇（包含參數調整等進階能力，可協助優化模型）。
6. Preprocessing：資料預處理（特徵值的相關處理，在本書第五章介紹）。

The screenshot shows the official website for scikit-learn, which is a machine learning library for Python. The top navigation bar includes links for 'Getting Started', 'Release Highlights for 0.23', and 'GitHub'. The main content area is divided into six sections, each with a title, a brief description, and a corresponding image or plot:

- Classification**: Identifying which category an object belongs to. Applications include spam detection and image recognition. Algorithms listed are SVM, nearest neighbors, random forest, and more. An image shows a 2D scatter plot with three classes.
- Regression**: Predicting a continuous-valued attribute associated with an object. Applications include drug response and stock prices. Algorithms listed are SVR, nearest neighbors, random forest, and more. An image shows a line plot of a sinusoidal function with scattered data points.
- Clustering**: Automatic grouping of similar objects into sets. Applications include customer segmentation and grouping experiment outcomes. Algorithms listed are k-Means, spectral clustering, mean-shift, and more. An image shows a 2D scatter plot with data points colored by cluster assignment.
- Dimensionality reduction**: Reducing the number of random variables to consider. Applications include visualization and increased efficiency. Algorithms listed are k-Means, feature selection, non-negative matrix factorization, and more. An image shows a 3D scatter plot of the Iris dataset with labels 'Versicolor', 'Verginica', and 'Setosa'.
- Model selection**: Comparing, validating and choosing parameters and models. Applications include improved accuracy via parameter tuning. Algorithms listed are grid search, cross-validation, metrics, and more. An image shows a line plot of a metric like cross-validation score versus a parameter value.
- Preprocessing**: Feature extraction and normalization. Applications include transforming input data such as text for use with machine learning algorithms. Algorithms listed are preprocessing, feature extraction, and more. An image shows a 2D scatter plot with data points colored by feature values.

◆ 圖 3-5 scikit-learn 官網首頁的六大功能分頁



此外，sklearn 官網對於初學者在資料量與演算法的選擇上，有給予一張策略建議圖，讀者可以善加利用這樣的指引地圖來做模型建構，以減少自行摸索的時間。模型指引地圖網址為 https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html。



3-2-2 波士頓房價預測

在介紹過 scikit-learn 函式庫後，接下來我們將使用線性迴歸的方法來挑戰經典迴歸問題「波士頓房價預測」，讓讀者可以掌握先前所學的知識要如何與實務操作結合。本書使用 Google Colab 的平台環境進行所有操作（Google Colab 使用說明請參考附錄 B），利用 Python 程式語法做程式撰寫的動作（Python 操作可以參考附錄 A 的 Python 語法表），再參照本節的程式碼範例，或是參考線上平台的影音教學檔都可以輕鬆學習，慢慢熟練。

Step1 載入所需套件

首先，我們需要從 sklearn 函式庫載入波士頓房價問題的所有資料集，利用以下的程式碼下載資料，並開始一系列的實作歷程。

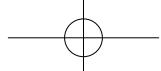
```
# 如果要使用一些 datasets，都會放在 sklearn 的 datasets 下面
# 從 sklearn 載入波士頓房價的資料集
from sklearn.datasets import load_boston

# 載入我們會用到的模型，線性迴歸模型：LinearRegression
from sklearn.linear_model import LinearRegression

# 載入切分資料集成訓練集及測試集的套件：train_test_split
from sklearn.model_selection import train_test_split

# 載入驗證模型的套件：mean_square_error
from sklearn.metrics import mean_squared_error
```

此步驟中，我們已經將所有的工具（架構模型、資料切分、測試驗證）與素材（資料集）都準備好了，而這個動作就如同廚師在端出一盤好菜之前，必須先把刀具、爐具及食材都準備完畢，才能進行下一個動作是相同的道理。



Step2 載入並觀察資料集

在資料集匯入後，我們會先觀察內部的資料架構與內容，也如同前述廚師煮菜的例子，讀者可以試想當食材整備完畢後，應該做細部檢查，確認是否有腐爛或是清洗不乾淨的問題，進而排除並解決。在人工智慧領域，資料前處理的動作是相當重要且不可馬虎的，這樣才能夠在接下來訓練模型的步驟中，順利且有效率的進行。

```
# 載入資料集，放到 my_data 變數內
my_data = load_boston()

# 可以把 my_data 紿印出來，看一下裡面的內容
# 印出來後可知他有幾個 key name 有："data", "target", "target_
names", "DESCR", "filename" ... 等
print(my_data)

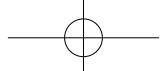
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
   4.9800e+00],
 [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
  9.1400e+00],
 [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
  4.0300e+00],
 ...,
 [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
  5.6400e+00],
 [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
  6.4800e+00],
 [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
  7.8800e+00]]), 'target': array([24., 21.6, 34.7, 33.4, 36.2, 28.7, 22.9,
 18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
 15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21., 12.7, 14.5, 13.2,
```

◆ 圖 3-6 房價資料集的矩陣數據

```
# 直接觀察有哪些 key 在資料集內
print(my_data.keys())
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

◆ 圖 3-7 房價資料集的 key



這些 key 分別代表的意涵是：

- data 資料
- taget 真實房價
- feature_name 欄位名稱
- DESCR 描述 (description 的縮寫)
- filename 檔案名稱

接下來我們更進一步來輸出資料的型態與內容

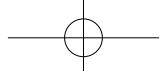
```
# 觀察資料筆數
print(" 資料筆數 :")
print(my_data.data.shape)
print("\n")

# 觀察我們資料的欄位名稱
print(" 資料的欄位名稱，分別是 :")
print(my_data.feature_names)
print("\n")

# 觀察我們第一筆的資料內容
print(" 第一筆的資料內容 :")
print(my_data.data[0])
print("\n")

# 觀察我們第一筆的預測目標
print(" 第一筆的預測目標 :")
print(my_data.target[0])
print("\n")

# 每個 sklearn 所附的資料集都有 DESCR 這個 key 可以選來看，代表這個資料集的描述 (description 的縮寫 )
```



資料筆數：

(506, 13)

資料的欄位名稱，分別是：

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'  
'B' 'LSTAT']
```

第一筆的資料內容：

```
[6.320e-03 1.800e+01 2.310e+00 0.000e+00 5.380e-01 6.575e+00 6.520e+01  
4.090e+00 1.000e+00 2.960e+02 1.530e+01 3.969e+02 4.980e+00]
```

第一筆的預測目標：

24.0

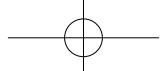
◆ 圖 3-8 房價資料集的資料型態與內容

Step3 切分資料集

上述步驟完成後，我們已初步理解資料集的組成架構與內容。接著，就要開始進入訓練模型的重要步驟：「資料切分」。讀者可以嘗試回想，在自己的學習經驗中，是否常透過寫考卷來熟練某些知識？同樣的，電腦科學家在機器學習領域也利用了類似概念。舉例來說，如果老師把手上的 100 張考卷全部一股腦地丟給學生書寫並告知正確答案，若學生真的很認份地把考卷全部寫完後，就會有一個問題產生：「究竟學生是把答案全部背起來了，還是真正透過這些考卷學習到相對應的知識呢？」尷尬的是，此刻電腦科學家的手上已沒有任何考卷可以來做上述問題的確認了。

聰明的讀者，您可能已經了解為什麼要將資料拆分成「訓練資料」與「測試資料」了。的確，如果我們一開始不把所有的資料都丟給機器去學習，而是拆分成一定的比例作為測試之用，那麼就能夠避免上述例子中，不確定機器是否真正做了學習，抑或是只將答案背起來的窘境。

在資料切分這個步驟，切分的比例並沒有一定的限制，但大多數的資料科學家都會先採用訓練資料集佔 80%，測試資料集佔 20% 這樣的比例開始做模型訓練，我們也從善如流依此比例做範例說明：



```
# 我們將資料切分成兩組，80% 當成訓練集，20% 當成測試集
# (train_x, train_y) 為 80% 的訓練集，用來訓練模型，x 是指資料，y 是指答案
# (也就是真實房價)
# (test_x , test_y ) 為 20% 的測試集，用來驗證模型的預測能力
train_x, test_x, train_y, test_y = train_test_split(my_data.data,
my_data.target, test_size=0.2, random_state=43, shuffle=True)

# 可以看一下這些資料集的維度，驗證沒有切錯
print(" 原始資料集的維度大小：" , my_data.data.shape)
print(" 訓練集的維度大小：" , train_x.shape)
print(" 測試集的維度大小：" , test_x.shape)
```

原始資料集的維度大小: (506, 13)
訓練集的維度大小: (404, 13)
測試集的維度大小: (102, 13)

◆ 圖 3-9 資料切割前後的維度，例如：506 是資料筆數、13 是欄位

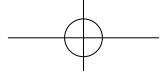
Step4 訓練模型及驗證模型

此步驟正式進入訓練模型的階段，我們將先前切分好的訓練資料集（佔全體 80%）載入預先設定的模型架構「線性迴歸：Linear Regression」模型中進行訓練，待模型訓練好後利用預先保留的測試資料集（佔全體 20%），驗證模型所預估的答案與正確答案是否相同，透過評估指標 MSE (mean_squared_error) 來判定其模型的優劣程度。

```
# 把預先載入好的套件「線性迴歸模型」拿出來使用，並且存到變數 my_model 中
my_model = LinearRegression()

# 訓練模型，放入要訓練的訓練集 (train_x, train_y)
my_model.fit(train_x, train_y)

# 訓練完模型後，我們就可以用 my_model 來預測測試集，產生出預測值，存到 pred
# 變數內
pred = my_model.predict(test_x)
# 接著就可以使用評估指標 MSE (mean_squared_error) 來評估模型的實際誤差
# 注意 MSE 分數是越接近 0 越好，代表預測的值與真實答案相差無幾
```



```
score = mean_squared_error(pred, test_y)
print("模型評估完測試集的誤差：", score)
```

模型評估完測試集的誤差： 22.01861344950045

◆ 圖 3-10 迴歸模型的 MSE 誤差值

整個模型訓練的步驟，其實只有兩行程式

```
my_model = LinearRegression()
my_model.fit(train_x, train_y)
```

也就是將線性迴歸模型指派到一個變數 my_model，然後用 my_model.fit() 就可以將訓練資料 train_x 與答案 train_y，進行運算，學習出的迴歸模型。

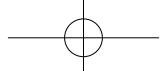
知識補充

MSE (mean_squared_error)

中文稱為均方誤差，是指預測值與真實值的誤差平方和取平均值的平方根（均方根誤差的平方），以房價預測這個應用案例來說，我們將訓練好的迴歸模型用測試資料集驗證，模型會對每一筆房屋資料預測出房價數值，再與該筆房屋的真實房價做誤差相減，把這些誤差值求平方，然後全部加總後取平均值，將平均值再開根號，就會是我們用來評估模型準確率的 MSE (mean_squared_error)。當 MSE 值越小，代表預測值與真實值誤差越小，說明預測模型具有更好的精確度。

讀者可以想想看，我們可以不使用全部的特徵（13 個）來訓練模型嗎？效果會有差異嗎？讀者可以嘗試只取資料集的其中 4 個特徵來訓練模型，並觀察其 MSE 來衡量新模型的預測能力。

```
# 挑選其中的 4 個特徵
train_x_f4=train_x[:,[4,5,6,7]]
test_x_f4=test_x[:,[4,5,6,7]]
# 這裡的 , 前面如果填上數字代表著要哪些列 ( 橫向 )，填上：就是從最前面到最後面，也就是全部都要
# 而後面代表著挑選哪些行 ( 直向 )，也就是特徵，所以直接填上數字代表你要挑選特徵欄位的位置
```



```
# 以下的程式碼與上一段的程式碼大同小異  
# 大家可以練習看看，但是注意 train_x,train_y,test_x,test_y 的位置要放對哦！  
model_f4=LinearRegression()  
model_f4.fit(train_x_f4,train_y)  
  
pred=model_f4.predict(test_x_f4)  
  
# 最後可以看看這次的預測分數如何，與上一段的分數做比較  
score=mean_squared_error(pred,test_y)  
print(" 模型評估完測試集的誤差 :", score)
```

模型評估完測試集的誤差: 39.04445392463858

◆ 圖 3-11 只用四個欄位特徵的迴歸模型 MSE 誤差值

在上述結果可清楚發現，僅有 4 個特徵值的模型其誤差率遠高於利用 13 個特徵值所訓練出來的模型，誤差率越高代表這個模型越無法正確預估真實情況。這也與我們真實生活的經驗相符，試想若我們要評估一個商品的價值，越多的參考資料如網路評價、第三方實測結果等，對於做決策會有很大的幫助。

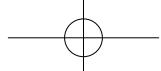


◆ 圖 3-12 房價預測迴歸模型實作流程圖



想想看

「資料的特徵越多，對於模型的建構就一定有幫助」這句話對嗎？我們鼓勵各位讀者自己動手去嘗試使用不同的特徵數來訓練不同模型，並透過評估指標來確認上面的論述是否正確。



3-3 乳癌機率有多高—KNN 分類器介紹與應用

在這個小節，我們要介紹監督式學習中常見的分類演算法：**K-最近鄰演算法（K Nearest Neighbor, KNN）**，接下來本書的內容中都會用 KNN 來統稱。

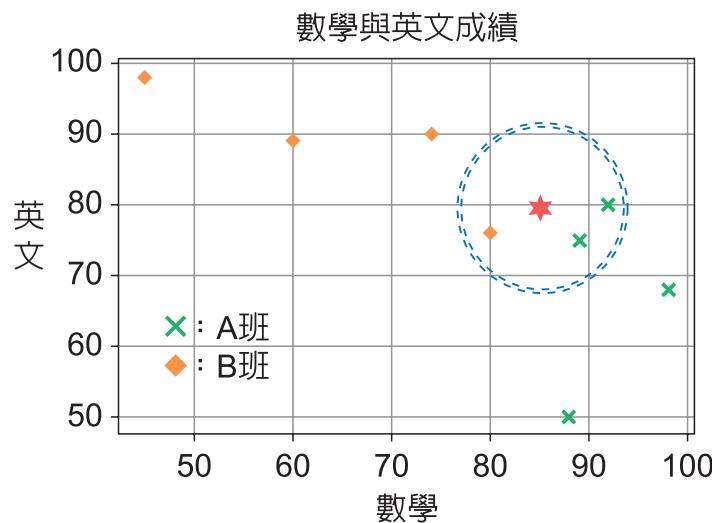
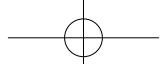
KNN 是一種威力強大的演算法，不僅可以處理分類的問題，同時也可以協助處理迴歸的問題，本節會利用 KNN 來協助判定罹患乳癌的機率高低。

讀者不難從 KNN 的中文名稱：「最近鄰演算法」來理解其中的原理與運作方式，它的判斷邏輯就是透過周圍的「鄰居」來幫忙決定。我們把 KNN 作為分類器（Classifier），當它分辨一筆新資料的類別，就是尋找與這筆資料最接近的 K 筆資料（K 值是使用者可以自行預先設定的一個整數），由這 K 筆資料的類別標籤協助判定。

表 3-3 是從某校高中一年級任意選取 8 名學生之數學、英文考試的相關資料表，若今天有一名學生考試成績是：數學 85 分、英文 70 分，透過 KNN 分類法，將 K 值設定為 3，那麼從圖 3-13 可清楚看出與其（星號位置）最近的三筆資料分別是：A、A、B，若按照 KNN 演算法的規則，3 個鄰居的最多數分類是 A，所以該位同學（圖中星號）就判定為來自於 A 班。

◎表 3-3 某校 8 名同學的數學、英文成績與其就讀班級資料表

訓練資料序號	數學成績	英文成績	就讀班級
1	92	80	A
2	89	75	A
3	45	98	B
4	60	89	B
5	74	90	B
6	88	50	A
7	98	68	A
8	80	76	B



◆ 圖 3-13 某校 8 名同學的數學、英文成績散佈圖

3-3-1 乳癌判斷的問題介紹與資料觀察

介紹完 KNN 演算法的觀念後，也同樣要帶領大家親自實作經典資料集「乳癌判斷（breast_cancer）」，跟著本書的內容，合併參考 Colab 程式碼與線上影音教學，一起實際動手操作，並熟練 KNN 演算法：

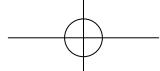
Step1 載入所需套件

從 sklearn 載入資料集 breast_cancer，此範例資料集是來自 UCI（加州大學爾灣分校）的「乳癌診斷數據集」（Breast Cancer Data Set），我們希望透過機器學習來協助醫師精準判定乳癌患者的腫瘤是良性或是惡性：

```
# 從 sklearn 載入 breast_cancer 資料集
from sklearn.datasets import load_breast_cancer

# 載入我們會用到的模型，KNN 分類模型
from sklearn.neighbors import KNeighborsClassifier

# 載入切分資料集成訓練集及測試集的套件，train_test_split
from sklearn.model_selection import train_test_split
```



Step2 載入並觀察資料集

觀察資料集的內容與大致架構為何，一般來說若有缺失值是需要先做補缺失值處理的，但因為這些資料集都已被整理過，所以才省略了這個步驟。但讀者必須理解的是，現實生活中的資料蒐集，常有缺漏狀況，故在此步驟必須留意是否有類似狀況發生，若有，則必須優先處理，才能進行模型訓練。（本書後續章節會介紹）

```
# 載入資料集，放到 my_data 變數內
my_data = load_breast_cancer()

# 可以把 my_data 紿印出來，看一下裡面的內容
# 印出來後可知有幾個 key name : "data", "target", "target_
names", "DESCR", "filename" ... 等
print(my_data)

# 更多關於此資料集的敘述可以到官網瀏覽
#https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\_breast\_cancer.html

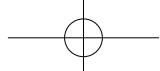
# 觀察有哪些 key 在資料集內
print(my_data.keys())

dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

◆ 圖 3-14 乳癌資料集的 key

```
# 觀察資料筆數
print("資料筆數 :")
print(my_data.data.shape)
print("\n")

# 觀察我們第一筆的資料內容
print("第一筆的資料內容 :")
```



```
print(my_data.data[0])
print("\n")

# 觀察我們第一筆的分類目標
print(" 第一筆的分類目標 :")
print(my_data.target[0])
print("\n")

# 觀察我們要預測目標的名稱種類
print(" 預測目標的名稱種類，分別是惡性腫瘤及良性腫瘤 :")
print(my_data.target_names)
print("\n")
```

資料筆數：

(569, 30)

第一筆的資料內容：

```
[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
 4.601e-01 1.189e-01]
```

第一筆的分類目標：

0

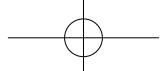
預測目標的名稱種類，分別是惡性腫瘤及良性腫瘤：

['malignant' 'benign']

◆ 圖 3-15 乳癌資料共 569 筆，30 個欄位資訊

Step3 切分資料集 - 訓練集 / 測試集

如同上個小節的迴歸問題，我們把所有資料切分成訓練資料（佔 80%），剩餘 20% 留作驗證資料，來確認機器（模型）是否真正達成學習的目的：



```
# 我們將資料切分成兩組，把 80% 當成訓練集，20% 當成測試集  
# (train_x, train_y) 為 80% 的訓練集，用來訓練模型  
# (test_x , test_y ) 為 20% 的測試集，用來驗證模型的預測能力  
  
train_x, test_x, train_y, test_y = train_test_split(  
    my_data.data, my_data.target, test_size= 0.2, random_state=18,  
    shuffle=True)  
  
# 可以看一下這些資料集的維度，驗證沒有切錯  
print(" 原始資料集的維度大小 :" ,my_data.data.shape)  
print(" 訓練集的維度大小 : " ,train_x.shape)  
print(" 測試集的維度大小 : " ,test_x.shape)
```

原始資料集的維度大小: (569, 30)
訓練集的維度大小: (455, 30)
測試集的維度大小: (114, 30)

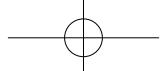
◆ 圖 3-16 乳癌資料切割後的結果

Step4 訓練模型及驗證模型

```
# 把預先載入好的套件 KNN 分類器拿出來使用，並且存到變數 my_model 中  
my_model = KNeighborsClassifier()  
  
# 訓練模型，放入要訓練的訓練集 (train_x, train_y)  
my_model.fit(train_x, train_y)  
  
# 訓練完模型後，我們就可以用 my_model.score 來評估模型的能力  
test_score = my_model.score(test_x, test_y)  
print(" 模型評估完測試集的準確度為 : ", test_score)
```

模型評估完測試集的準確度為: 0.9473684210526315

◆ 圖 3-17 模型的準確度有 94.7%



在這個步驟完成後，我們可以清楚發現整個模型的準確度達 94.7%，代表我們每 100 筆資料的預測（有或沒有乳癌），有 94.7 筆與真實的答案相同，是很不錯的預測準確值。但細心的讀者在此或許會提出疑問：整個實作過程中 KNN 的 K 值為什麼沒有指定，卻也可以跑出模型與預測？這是因為我們使用的 KNeighborsClassifier() 套件內部就已經預設 K 值為 5 了，所以在沒有特別設定下，模型就是以 K=5 的前提去做建模的。



◆ 圖 3-18 乳癌預測 KNN 模型實作流程圖

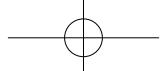


想想看

如果 K 值不同，會不會有更好或更壞的預測準確值呢？我們鼓勵讀者去做這樣的驗證，而方法也很簡單，只要在上述的範例程式碼中將 `my_model = KNeighborsClassifier()` 填入指定的 K 值即可，例如我們想看 K 值為 7 時，即可輸入下列程式碼：

```
my_model = KNeighborsClassifier(n_neighbors=7)
```

請讀者嘗試多個相異 K 值產生新模型，並衡量其預測準確值是否有什麼有趣的發現呢？



3-4 傑克與蘿絲誰的生存機率高—決策樹分類器介紹與應用

你知道電影鐵達尼號的主角傑克和蘿絲嗎？

在鐵達尼號的悲劇故事中，儘管過程唯美而浪漫，結局卻只有蘿絲一人存活下來。你知道這個故事是由真實的事件改編的嗎？

本節會帶你去了解鐵達尼號上乘客的相關資料，讓我們去推敲蘿絲的存活究竟是因為傑克的偉大愛情，還是因為當時紳士們禮讓老弱婦孺先逃生的結果。

若為前者，則男性與女性和小孩的犧牲比率應該會大致相同；如果為後者，那麼女性和小孩的生存比率應該要顯著地大於成年男性。

3-4-1 鐵達尼資料集

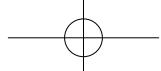
這次要使用的鐵達尼號資料集是國際機器學習競賽平台 Kaggle 上的數據（網址為 <https://www.kaggle.com/c/3136/download-all>）。我們可以註冊會員，然後下載資料集來練習。

The screenshot shows the Kaggle homepage with the navigation menu on the left. The main content area displays the 'Titanic: Machine Learning from Disaster' competition. It features a large image of the Titanic ship, the competition title, and a statistic that 16,308 teams are ongoing. Below the main image, there's a 'Join Competition' button. The 'Overview' tab is selected in the navigation bar. On the left, there's a sidebar with links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. The main content area has sections for Description, Evaluation, Tutorials, and Frequently Asked Questions. The 'Description' section includes a welcome message and a brief introduction to the competition.

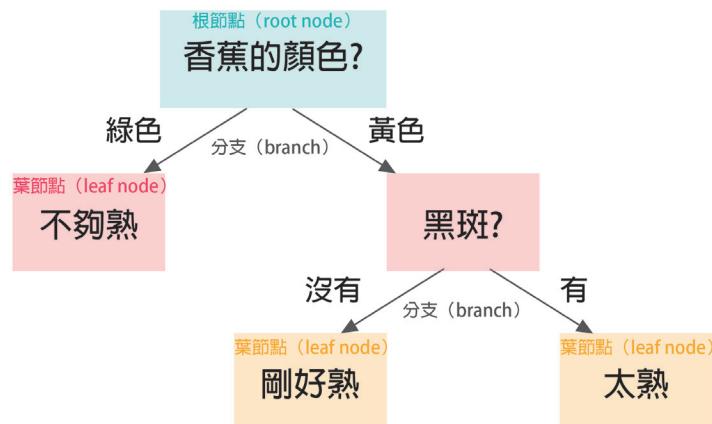
◆ 圖 3-19 Kaggle 平台頁面

決策樹介紹

決策樹是一個非常經典的人工智慧技術，歸類於機器學習領域。它是透過不斷進行二元分類來決定輸入最後會對應輸出。



從最源頭的問題，也就是**根節點（Root Node）**開始作答，依據條件延伸分枝，直到抵達最後的**葉節點（Leaf Node）**，葉節點顯示最後的機率。決策樹就像爬格子遊戲一樣，有多個終點，也有與之對應的起點。決策樹的訓練就是讓訓練資料從起點往下走，最後能夠走到正確終點的過程。



◆ 圖 3-20 決策樹示意圖

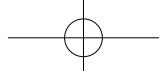
3-4-2 鐵達尼號資料集的介紹

首先，連結到這個網址 <https://www.kaggle.com/c/titanic> 找到經典中的經典題目「**鐵達尼生存預測（Titanic: Machine Learning from Disaster）**」。進入賽事網頁後，點選上方橫欄選項的 Data 欄，就會看到資料說明與下載區域。圖 3-21 的畫面說明資料的組成：

資料集：總共有兩個資料集

- 第一個資料集是**訓練資料（Train Dataset）**，總共有 891 筆，每一筆資料有 12 個欄位，也就是每一筆資料都記載著一個位乘客的 12 個欄位資料，分別是乘客的：

1. 編號
2. 出發地
3. 船票等級
4. 姓名
5. 性別
6. 年齡
7. 與兄弟姊妹及配偶一起上船的人數
8. 與直系親屬一起上船的人數
9. 船票號碼
10. 票價
11. 船艙編號
12. 在船難中存歿



- 第二個資料集是**測試資料集（Test Dataset）**，總共有 417 筆資料，代表 417 位乘客，但是只有 11 個欄位資料，比訓練資料少一個「乘客在船難中存歿」的欄位資料。

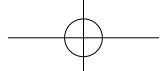
The screenshot shows the Kaggle competition page for 'Titanic: Machine Learning from Disaster'. The main title is 'Titanic: Machine Learning from Disaster' with the subtitle 'Start here! Predict survival on the Titanic and get familiar with ML basics'. Below the title, it says 'Kaggle - 16,405 teams · Ongoing'. The navigation menu includes Overview, Data (which is selected), Notebooks, Discussion, Leaderboard, Rules, Team, My Submissions, and Submit Predictions. The 'Data' section is titled 'Data Description' and contains an 'Overview' section. It explains that the data is split into two groups: training set (train.csv) and test set (test.csv). The training set is used to build machine learning models, providing the outcome (ground truth) for each passenger based on features like gender and class. The test set is used to see how well the model performs on unseen data. Below this, there's a table showing 'Data (34 KB)' with three rows: 'Data Sources' (listing 'gender_submission.csv' and 'test.csv'), 'About this file' (describing it as an example of what a submission file should look like), and 'Columns' (listing 'PassengerId' and 'Survived'). At the bottom, there are download buttons for 'API', 'kaggle competitions download -c titanic', and 'Download All'.

◆ 圖 3-21 鐵達尼生存預測競賽資料

測試資料集所缺少的「存歿」欄位，就是我們要藉由機器學習推測出來的資料，也就是去推估測試資料集中的人物在船難中是生是死。

我們把「訓練資料集」中的乘客存歿資料當作答案標籤（Label），利用其他欄位組成的訓練資料與此欄標籤資料，來訓練機器做好乘客的存歿預測。訓練好機器後，新增測試資料集到模型中去預估資料集裡乘客的存歿，最後將預測結果存檔上傳至 Kaggle 官網進行成果判定。

下載這些資料的方法是在圖 3-21 的畫面往下拉，一直到圖 3-22 的畫面後，點選右方的「Download All」將資料集儲存到自己的電腦，解壓縮後就得到訓練資料集（train.csv）、測試資料集（test.csv）以及上傳答案的檔案格式（gender_submission.csv）。



The screenshot shows the Kaggle interface for the 'titanic' competition. At the top, there's a navigation bar with 'API', a search bar containing 'kaggle competitions download -c titanic', a help icon, and a 'Download All' button which is highlighted with a red box. Below the navigation is a section titled 'Data (34 KB)' which includes 'Data Sources' (listing 'gender_submission.csv', 'test.csv', and 'train.csv'), 'About this file' (describing the submission file example), and 'Columns' (listing 'PassengerId' and '# Survived'). Below this is a preview of 'gender_submission.csv' showing two columns: 'PassengerId' and '# Survived'. The preview table has a single row with values: PassengerId 1, # Survived 0.

◆ 圖 3-22 下載鐵達尼生存預測的資料

3-4-3 決策樹模型實作

Step1 讀取資料集

接下來，我們使用下面的程式碼來讀取並觀察資料集：

因為我們已經預先將 Kaggle 的訓練資料集載到雲端，因此只要輸入以下程式，即可在 colab 讀取 csv 資料。

```
!wget --no-check-certificate "https://drive.google.com/uc?export=download&id=13bGRvk1Vq9tFRzMWsXZw0g7TzZLQj830" -O 'train.csv'
```

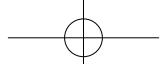
接著我們載入決策樹模型，與 NumPy、Pandas 兩個機器學習常用函式庫。

Pandas 是 Python 語言的資料分析函式庫，於 2009 年底開源出來，提供高效能、簡易使用的資料格式（Data Frame）讓使用者可以快速操作及分析資料。

NumPy 也是數學函式庫，支援高階大量的維度陣列與矩陣運算。

```
from sklearn import tree
import numpy as np
import pandas as pd

data = pd.read_csv('train.csv')
```



PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...
886	887	0	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W/C. 6607	23.4500	NaN	S
889	890	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

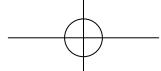
◆ 圖 3-23 執行上述程式可以將試算表中的內容儲存到變數 "data" 中

在資料集中的欄位分別為：

1. 'PassengerId' 乘客的編號
2. 'Embarked' 出發地
3. 'Pclass' 船票等級
4. 'Name' 姓名
5. 'Sex' 性別
6. 'Age' 年齡
7. 'SibSp' 與配偶或兄弟一起上船的人數
8. 'Parch' 與配偶和小孩一起上船的人數
9. 'Ticket' 船票號碼
10. 'Fare' 票價
11. 'Cabin' 船艙編號
12. 'Survived' 在船難中存活

雖然資料很多，不過我們先取其中的 4 個部分來分析就好。我們切割出 Sex、Age、Fare、Survived 來進行分析，如下所示：

```
data = data.loc[:, ['Sex', 'Age', 'Fare', 'Survived']]  
data
```



		Sex	Age	Fare	Survived
0	male	22.0	7.2500	0	
1	female	38.0	71.2833	1	
2	female	26.0	7.9250	1	
3	female	35.0	53.1000	1	
4	male	35.0	8.0500	0	
...
886	male	27.0	13.0000	0	
887	female	19.0	30.0000	1	
888	female	Nan	23.4500	0	
889	male	26.0	30.0000	1	
890	male	32.0	7.7500	0	
891 rows × 4 columns					

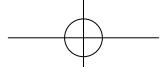
◆ 圖 3-24 程式執行後會顯現出設定好的四個欄位的變數內容

接下來，我們就可以從 Sex、Age、Fare 這三個屬性中，分析它們與 Survived 的關係。

Step2 資料轉換

```
# 把 male, female 轉換成 0, 1  
data.loc[data['Sex']=='male', 'Sex'] = 0  
data.loc[data['Sex']=='female', 'Sex'] = 1  
  
data
```

上面的敘述可以把 Sex 欄位的 male 和 female 兩個單字轉換成 0 與 1，讓電腦能看懂，結果如下所示。



	Sex	Age	Fare	Survived
0	0	22.0	7.2500	0
1	1	38.0	71.2833	1
2	1	26.0	7.9250	1
3	1	35.0	53.1000	1
4	0	35.0	8.0500	0

◆ 圖 3-25 性別資料 (Sex) 欄位資料轉成了數值 0 與 1

Step3 進行分析

首先，我們想要分析在存活的人之中，哪種性別的比例較高，透過下面的操作，我們可以略知概況。要記得，資料轉換的步驟已經把 male 轉換成 0、female 轉換成 1。分析有三個步驟：

1. 資料集中，取出最後有存活的資料集 data_sur

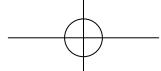
```
data_sur = data[data['Survived']==1]
```

2. 存活且為男性的資料集 male 和存活且為女性的資料集 female

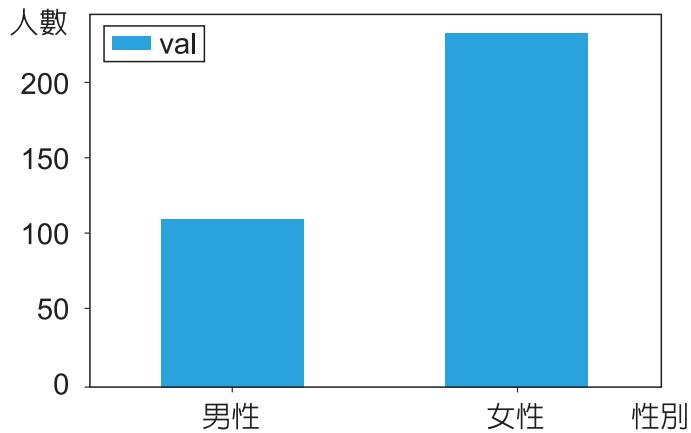
```
df_male = data_sur[data_sur['Sex']==0]
df_female = data_sur[data_sur['Sex']==1]
```

3. pandas 內建的 plot.bar() 畫出柱狀圖來分析

```
df = pd.DataFrame({'Sex':[ 'df_male', 'df_female'], 'val':[len(df_
male), len(df_female)]})
ax = df.plot.bar(x='Sex', y='val', rot=0)
```



結果如下所示：



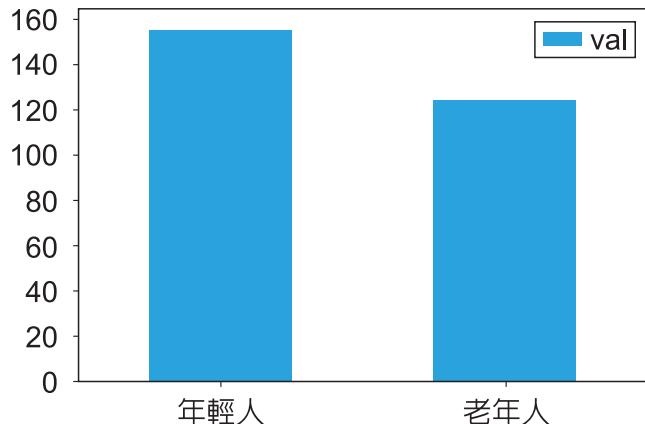
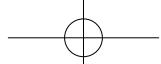
◆ 圖 3-26 程式顯示男女生的存活人數

可以看到男性存活的比率顯著地低於女性存活的比率，這可以說明當時羅絲的存活，不只是因為傑克的愛情，同時也是船上男性乘客展現紳士風度的關係。

透過下面的程序也可以看到年輕人（<30 歲）和老人（>30 歲）的存活比率，年輕人的存活比率也較老人為高。

```
# 先取出存活且小於 30 歲的資料集 df_young 和存活大於 30 歲的資料集 df_older
df_young = data_sur[data_sur['Age']<30]
df_older = data_sur[data_sur['Age']>30]

# 畫出柱狀圖觀察
df = pd.DataFrame({'Age':['young', 'older'], 'val':[len(df_young), len(df_older)]})
ax = df.plot.bar(x='Age', y='val', rot=0)
```



◆ 圖 3-27 程式顯示年輕人與年長者的存活人數

接下來，我們要根據訓練資料集的數據來訓練決策樹模型。由於我們想直接了解模型的準確率，所以使用有「存歿」欄位的訓練資料集。首先要從訓練資料集中分割出一個測試資料集。

我們的模型使用訓練資料集來訓練，測試資料集裡卻是模型完全陌生的一群人，但他們的確存在於鐵達尼號上。我們將用模型來預測這群陌生人是否會在鐵達尼號上存活。

Step4 資料清理

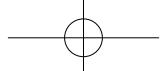
有些欄位與結果的關係並不大，例如：姓名、船票號碼、船艙編號，所以先把它移除（事實上，如果未來可以進行更深入的研究，也許這些被移除的欄位都相當有用）。

```
import pandas as pd

data = pd.read_csv('train.csv')
data = data.drop(columns=['Name', 'Ticket', 'Cabin'])
data
```

Step5 資料轉換

前面提到資料轉換就是為了把文字轉換成電腦能夠看得懂的語言，比如說把「S」這個英文單字轉換成「0」，因為相對於英文而言，數字對電腦來說更直覺。



需要轉換的欄位有兩個，Sex（性別欄位）和 Embarked（出發地欄位），轉換的方法如下：

```
# 轉換 Embarked 欄位
typeEmbarked = list(set(data['Embarked']))

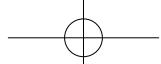
# 把 Embarked 欄位的文字按順序轉換成 0~n 的數字
for i in range(len(typeEmbarked)):
    print(typeEmbarked[i])
    row = data['Embarked'] == typeEmbarked[i]
    data.loc[row, 'Embarked'] = i

# 轉換 Sex 欄位
typeSex = list(set(data['Sex']))

# 把 Sex 欄位的文字按順序轉換成 0~n 的數字
for i in range(len(typeSex)):
    print(typeSex[i])
    rows = data['Sex'] == typeSex[i]
    data.loc[rows, 'Sex'] = i

data
```

轉換完成後會如圖 3-28 所示。每一個出發地都會對應到一個編號，但這是自動生成的，使用的方法叫做 Label Encoding。



PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	0	22.0	1	0	7.2500
1	2	1	1	1	38.0	1	0	71.2833
2	3	1	3	1	26.0	0	0	7.9250
3	4	1	1	1	35.0	1	0	53.1000
4	5	0	3	0	35.0	0	0	8.0500
...
886	887	0	2	0	27.0	0	0	13.0000
887	888	1	1	1	19.0	0	0	30.0000
888	889	0	3	1	NaN	1	2	23.4500
889	890	1	1	0	26.0	0	0	30.0000
890	891	0	3	0	32.0	0	0	7.7500

891 rows × 9 columns

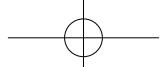
◆ 圖 3-28 把 Sex 欄位的資料轉成 0 與 1，把 Embarked（出發地欄位）轉換成 1 至 3

Step6 缺失值處理

對一個機器學習的模型來說，缺失值的存在會讓模型無法訓練，而我們因為資料很少，所以採用補值的方式，`fillna()` 函數會在整個資料集裡尋找缺失值並補值。

```
# fillna(999) 會讓所有缺失值都被補成 999，之所以選 999 是因為決策樹對離群值不敏感，所以使用 999 作為補值，但若使用神經網路或是 Regression 的各種演算法，則必須採取其他方法。
```

```
data = data.fillna(999)
```



Step7 成功補值

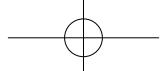
	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	0	22.0	1	0	7.2500	2
1	2	1	1	1	38.0	1	0	71.2833	1
2	3	1	3	1	26.0	0	0	7.9250	2
3	4	1	1	1	35.0	1	0	53.1000	2
4	5	0	3	0	35.0	0	0	8.0500	2
...
886	887	0	2	0	27.0	0	0	13.0000	2
887	888	1	1	1	19.0	0	0	30.0000	2
888	889	0	3	1	999.0	1	2	23.4500	2
889	890	1	1	0	26.0	0	0	30.0000	1
890	891	0	3	0	32.0	0	0	7.7500	3

◆ 圖 3-29 透過 `fillna` 函數將所有缺失值替換成 999 這個數值

Step8 分割資料並訓練模型

接下來，把編號前 750 筆資料當作訓練資料 `X_train`，編號 750 筆之後的則當作測試資料，並且把 `Survived` 欄位作為 `y_train` 獨立出來。

我們使用決策樹 `DecisionTreeClassifier()` 來進行訓練。



```
# 分割資料  
X_train = data[:750]  
X_test = data[750:]  
  
# 把`Survived`作為`y_train`從訓練資料中獨立出來。  
y_train = X_train.pop('Survived')  
  
# 建立並訓練(fit)決策樹  
clf = tree.DecisionTreeClassifier()  
clf = clf.fit(X_train, y_train)
```

Step9 評估模型正確率

最後，我們想要知道模型的準確率到底有多好，因此選擇了 Accuracy 和 Recall 兩個 metric（指標）來評估。

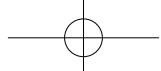
- 正確率（Accuracy）：模型每一次預測的準確度。
- 召回率（Recall）：代表模型是否能找出資料集中「存活」的樣本。

```
from sklearn.metrics import accuracy_score, recall_score  
  
# 從測試資料集取出y值作為真實答案  
y_test = X_test.pop('Survived')  
  
# 進行預測，取得預測答案  
y_pred = clf.predict(X_test)  
  
print('accuracy_score=',accuracy_score(y_test, y_pred))  
print('recall_score=',recall_score(y_test, y_pred))
```

Step10 準確率和召回率

```
accuracy_score= 0.7801418439716312  
recall_score= 0.6274509803921569
```

◆ 圖 3-30 準確率和召回率數值



訓練出來的鐵達尼模型匯入測試資料後，預測「存歿」標籤欄位的準確率有 81.5%。

以上就是很簡單的決策樹應用，決策樹對於類別型的資料有很強的分類能力，因為它的原理其實是很單純的二分法，不論數值大小或形態，都能夠用二分法來處理。

決策樹這一節的完整程式碼如下所示：

Step1: 取得資料並移除某些欄位

```
data = pd.read_csv('train.csv')
data = data.drop(columns=['Name', 'Ticket', 'Cabin'])
```

Step2: 資料轉換

```
# 轉換 Embarked 欄位
typeEmbarked = list(set(data['Embarked']))
```

把 Embarked 欄位的文字按順序轉換成 0~n 的數字

```
for i in range(len(typeEmbarked)):
    print(typeEmbarked[i])
    row = data['Embarked'] == typeEmbarked[i]
    data.loc[row, 'Embarked'] = i
```

轉換 Sex 欄位

```
typeSex = list(set(data['Sex']))
```

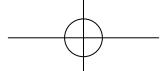
把 Sex 欄位的文字按順序轉換成 0~n 的數字

```
for i in range(len(typeSex)):
    print(typeSex[i])
    rows = data['Sex'] == typeSex[i]
    data.loc[rows, 'Sex'] = i
```

Step3: 缺失值處理

fillna(999) 會讓所有缺失值都被補成 999，之所以選 999 是因為不認為資料集的任何欄位中有 999 這個數字。

```
data = data.fillna(999)
```



```
# Step4: 分割資料並訓練模型
X_train = data[:750]
X_test = data[750:]

# 把 'Survived' 作為 'y_train' 從訓練資料中獨立出來。
y_train = X_train.pop('Survived')

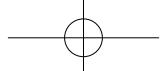
# 建立並訓練 (fit) 決策樹
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)

from sklearn.metrics import accuracy_score, recall_score

# Step5: 評估模型正確率
# 從測試資料集取出 y 值作為真實答案
y_test = X_test.pop('Survived')

# 進行預測，取得預測答案
y_pred = clf.predict(X_test)

print(accuracy_score(y_test, y_pred))
print(recall_score(y_test, y_pred))
```



3-5

總結

本章一開始，我們知道機器學習中若需要資料（觀察經驗）與標籤（有人告訴我們對錯與誤差），這種學習方式稱為「監督式學習」；另一類的機器學習沒有標籤資料，這類學習方法稱為「非監督式學習」。

了解監督式與非監督式學習後，3-1 節詳細說明監督式學習中代表性的問題種類：迴歸與分類。

如果要預測的是一個數值，例如溫度、股價，這類的問題稱為「迴歸」；若預測項目是有限的類別，就是「分類」題目，這些是目前人工智慧、機器學習中應用最多的。迴歸與分類題目的比較，請見 3-1-2 節的表 3-1 所示。

本章介紹數種演算法，包含預測房價的線性迴歸模型（K- 最近鄰演算法）、判斷是良性或惡性腫瘤的 KNN，以及推測鐵達尼船難中生存機率的決策樹等演算法。每道題目帶著讀者一步一步地執行機器學習的五大流程：定義問題、蒐集資料、處理資料、訓練模型、推論集預測。

請務必一步一步地實際操作，研讀程式中的註解去執行程式，了解結果的產生，思考為什麼要這麼設計機器學習的步驟，才算充分了解機器學習中監督式學習的理論與實作。

總結前面的小節內容：

- 監督式學習與非監督式學習的不同
- 監督式學習需要訓練資料與標籤資料
- 迴歸與分類的不同
- 三種演算法與模型：迴歸、KNN 與決策樹
- 程式與環境：基本的 Python、scikit-learn 套件與 Colab
- 專案的程序