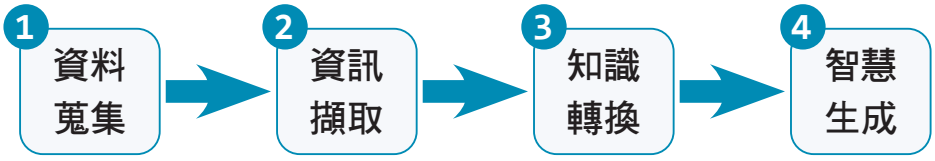
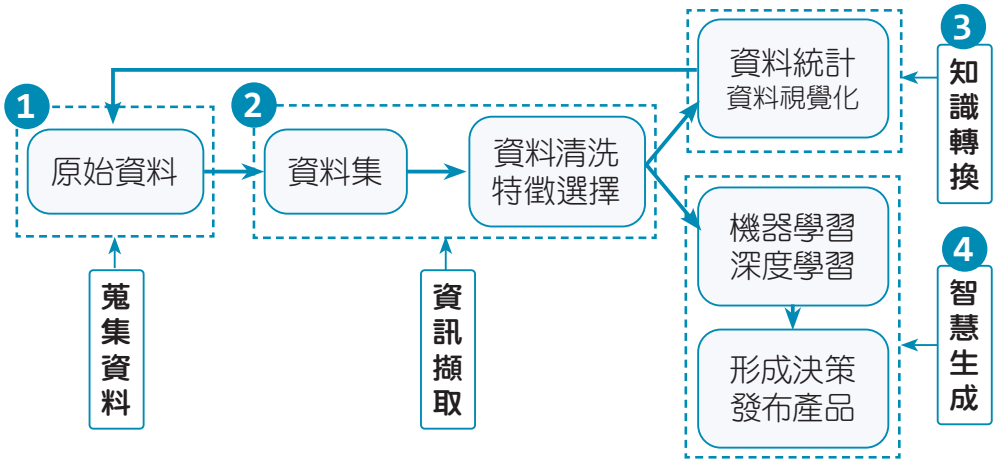
1.1.2 資料科學流程

資料 (Data) 經過處理後稱為 **資訊** (Information)，從這些資訊中分析出有用的訊息，就稱為 **知識** (Knowledge)，知識經過不斷的驗證與修正，最後形成 **智慧** (Wisdom)。資料科學流程就是由資料形成智慧的過程。



資料科學流程的實作過程為：



資料蒐集

資料的型態大致可以分為結構化資料及非結構化資料。結構化資料可以簡單定義為能夠透過表單系統如 Google 試算表、微軟 Excel 等呈現出來的資料，也就是資料可以透過行列式表格呈現出來。每一行 (column) 代表一種特殊的屬性 (特徵)，而每一列 (row) 為與該屬性相關的資料。行與列組成了表格，因而可以輕鬆引用。不同的表格可以互相連結，意即兩個表格之間同一列的資料可以互相關聯。

所有不是結構化資料的資料都屬於非結構化資料。非結構化資料通常是零亂的、沒有規律性的、持續產生的，因此這類資料具有不可預測性。一般使用者蒐集的「原始資料」多為非結構化資料。

資料蒐集的方式有企業或工廠運作產生的資料、由網路爬蟲爬取的網頁資料等。

4.11 Seaborn：更美觀的圖表工具

4.11.1 開始使用 Seaborn

認識 Seaborn

Seaborn 是一個基於 matplotlib 的 Python 資訊視覺化模組，它可以幫助使用者探索和理解資料的內容。Seaborn 支援多種不同的資料格式，無論是 pandas 或 numpy 產生的陣列物件，還是 Python 的串列和字典，它都能藉由簡單易懂的函式與指令進行相關的統計運算，不用執著於繪製的細節，即可將資訊進行視覺化的呈現。

安裝 Seaborn 與載入模組

可以使用下列指令在 Python 中安裝 Seaborn：

```
!pip install seaborn
```

注意：在 Colab 中預設已經安裝好 seaborn，不用再自行安裝。

使用 Seaborn 繪圖時首先要載入 Seaborn 模組，一般為了能在使用時更加方便，會設定別名 sns：

```
import seaborn as sns
```

設定圖表樣式

使用 Seaborn 所製作的圖表可套用樣式，即能擁有較美觀的呈現。設定的語法如下：

```
sns.set_theme([style= 樣式名稱 ])
```

- **style**：可以設定圖表的樣式字串，有 darkgrid(預設值)、whitegrid、dark、white 及 ticks。在使用時可以試試不同的樣式。

4.11.4 直方圖

直方圖是用 **sns.displot()** 的方法來繪製，語法為：

```
sns.displot(data= 資料來源 , x=x 軸欄位 , y=y 軸欄位 [, 其他參數 ])
```

常用的參數如下：

- **hue**： 資料欄位或是一維陣列資料，是根據設定的值為資料分類，並利用不同的顏色加以區分。
- **multiple**：設定 hue 分類的欄位後，可以設「stack」將每組資料化為堆疊圖，設「dodge」將每組資料化為分置圖。
- **col** 與 **row**：可以將 col 或 row 的值設為要分類的資料欄位。col 即會以指定欄位分成數個橫向圖表；row 即會以指定欄位分成數個直向圖表。

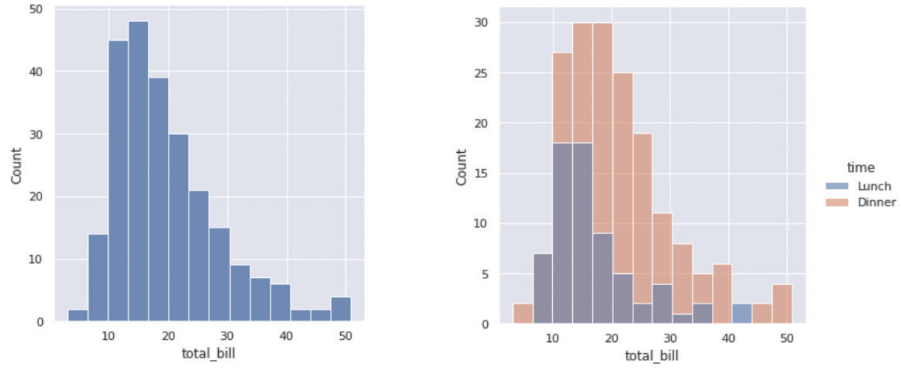
例如，這裡載入範例資料集「tips」，查詢消費總額的次數分佈狀況：

```
[ ]  1  sns.displot(data=tips, x='total_bill')
```

如果想要查詢不同的用餐時段消費總額的次數分佈狀況：

```
[ ]  1  sns.displot(data=tips, x='total_bill', hue='time')
```

顯示結果：



以用餐時段分類後，會將二個直方圖繪製在同一個圖表區中，一旁會自動顯示圖例。

4.11.5 散佈圖

散佈圖是用 **sns.scatterplot()** 的方法來繪製，語法為：

```
sns.scatterplot(data=資料來源, x=x軸欄位, y=y軸欄位[, 其他參數])
```

常用的參數如下：

- **hue**： 資料欄位或是一維陣列資料，是根據設定的值為資料分類，並利用不同的顏色加以區分。
- **style**：資料欄位或是一維陣列資料，是根據設定的值為資料分類，並利用不同的標記樣式加以區分。

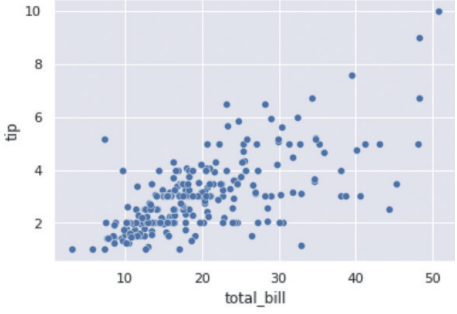
例如，這裡載入範例資料集「tips」，想要查詢消費總額與小費金額的相關情況：

```
[ ]  1 import seaborn as sns
     2 sns.set_theme()
     3 tips = sns.load_dataset("tips")
     4 tips.head()
```

```
[ ]  1 sns.scatterplot(data=tips, x="total_bill", y="tip")
```

顯示結果：

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4



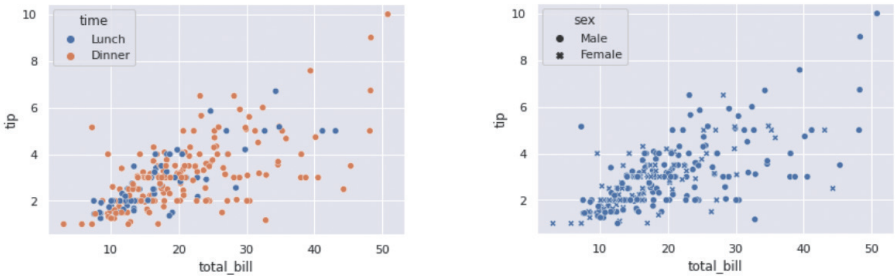
如果想要用不同顏色顯示不同用餐時段消費總額與小費金額的相關情況：

```
1 sns.scatterplot(data=tips, x="total_bill", y="tip", hue="time")
```

如果想要用不同標記樣式顯示不同用餐時段消費總額與小費金額的相關情況：

```
1  sns.scatterplot(data=tips, x="total_bill", y="tip", style="sex"
```

顯示結果：

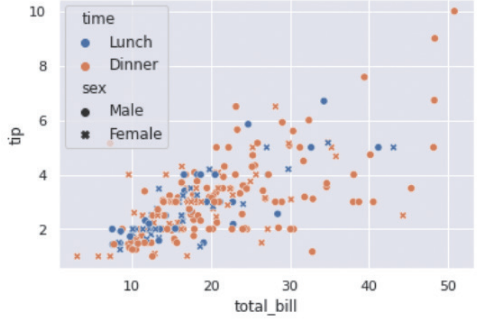


以 hue 將資料依用餐時段分類後，會用不同顏色顯示不同類別的資料；以 style 將資料依性別分類後，會用不同標記樣式顯示不同類別的資料。這二種方式都會在一旁自動顯示圖例。

如果想要同時顯示不同的用餐時段以及不同性別消費總額與小費金額的相關情況，可以將二個參數同時設定：

```
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="time", style="sex")
```

顯示結果：



用 hue 設定用餐時段分類後，會用不同顏色顯示分佈的資料，用 style 設定用性別分類後，會用不同標記樣式顯示分佈的資料，一旁顯示的圖例會標示二個欄位以及顏色與樣式的說明。

4.11.6 線箱圖

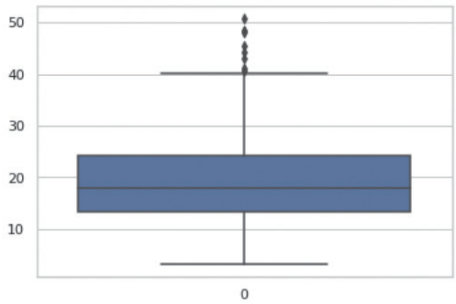
線箱圖是用 **sns.boxplot()** 的方法來繪製，語法為：

```
sns.boxplot(data= 資料來源 [, x=x 軸欄位 , y=y 軸欄位 ])
```

例如，這裡載入範例資料集「tips」，想要查詢消費總額的分佈狀況：

```
[ ]  1  import seaborn as sns
     2  sns.set_theme(style="whitegrid")
     3  tips = sns.load_dataset("tips")
     4  sns.boxplot(data=tips["total_bill"])
```

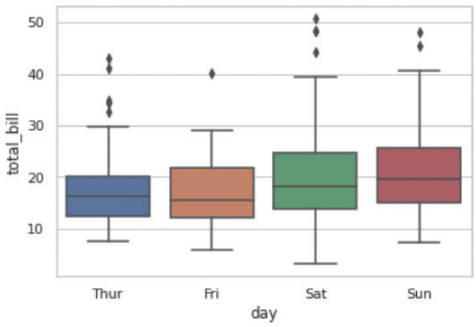
顯示結果：



例如，想要顯示不同星期日期消費總額的分佈狀況：

```
[ ]  1  sns.boxplot(data=tips, x="day", y="total_bill")
```

顯示結果：



5.5 圖片增量

在進行機器學習時，經常需要極大量的資料以確保模型的正確性。然而在如今數位時代，大部份有價值的資料都掌握在資金雄厚的公司手中，個人開發者很難蒐集完整的資料，尤其是圖片資料更加困難。

圖片增量技術是從既有的圖片中產生出更多的圖片讓系統去學習，意即是創造更多的「假」圖片，來彌補圖片不足的缺憾。雖然是假的圖片，但也是從原始圖片內容修改產生的，而且此技術的確可解決圖片不足的困境，提昇系統訓練的準確率。

一張圖片經過旋轉、調整大小、比例尺寸，或者改變亮度、色彩、翻轉等處理後，人眼仍能辨識出來是相同的相片，但是對機器來說則是完全不同的新圖像了，因此，將既有的圖片予以修改變形，就能創造出更多的圖片來讓機器學習，彌補資料量不足的困擾。

常用的圖片增量模組有 keras ImageDataGenerator 及 augmentor 模組。

5.5.1 keras ImageDataGenerator 模組

keras 是目前非常流行的機器學習框架，其中 ImageDataGenerator 類別具有圖片增量的功能，keras ImageDataGenerator 是最常用的圖片增量模組。

Colab 預設已安裝 keras 模組，要使用圖片增量功能需先載入 ImageDataGenerator 模組，語法為：

```
from keras.preprocessing.image import ImageDataGenerator
```

接著建立 ImageDataGenerator 物件，語法為：

```
增量變數 = ImageDataGenerator(參數 1= 值 1, 參數 2= 值 2, ……)
```

- **featurewise_center**：將輸入的特徵資料平均值設為 0。預設值為 False。
- **samplewise_center**：將每張圖片資料的平均值設為 0。預設值為 False。

圖片自動產生

ImageDataGenerator 物件參數頗多，可以僅設定一個變形參數來觀看其產生的圖片效果。以下程式設定「shear_range=50.0」讓使用者觀看圖片剪切效果。

```
1  from keras.preprocessing.image import ImageDataGenerator
2  import matplotlib.pyplot as plt
3  import numpy as np
4  import cv2
5
6  imgGen = ImageDataGenerator(shear_range=50.0, fill_mode='reflect')
7  n = 5
8  img = cv2.imread('cat.jpg')
9  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #BGR轉成RGB
10 img_sr = img.copy()
11 img = np.array(img, dtype=np.float32)
12 img = np.expand_dims(img, 0) #輸入圖片要是四維
13 flowGen = imgGen.flow(img, batch_size=10)
14
15 plt.figure(figsize=(20,10))
16 plt.subplot(1, n+1, 1)
17 plt.imshow(img_sr)
18 for i in range(n):
19   img = flowGen[0][0, :, :, :].astype(np.uint8)
20   plt.subplot(1, n+1, i+2)
21   plt.imshow(img)
22   plt.axis('off')
```

最左方圖片為原始圖片，右方 5 張為產生的圖片。

程式說明

- 6　建立剪切變形的 ImageDataGenerator 物件。
- 7　設定產生的圖片數量。
- 8　讀取原始圖片。
- 9　OpenCV 顏色使用 BGR，要轉換為 RGB 才能顯示正確顏色。

- 10　保留原始圖片複本，做為變形後的圖片對照。
- 11　將圖片轉為 numpy 陣列格式。
- 12　彩色圖片是 3 維，flow 方法的圖片必須是 4 維，此列程式將圖片增加 1 維成為 4 維。
- 13　使用 flow 方法建立產生變數。
- 15-16　顯示原始圖片做為對照。
- 18-22　利用迴圈產生 5 張圖片並顯示。

使用者可修改參數值觀看參數值大小的影響，要觀察其他效果可修改第 6 列程式參數設定，例如：

```
imgGen = ImageDataGenerator(rotation_range=60.0, fill_mode='reflect')
```

可觀看旋轉效果，執行結果為：

儲存自動產生的圖檔

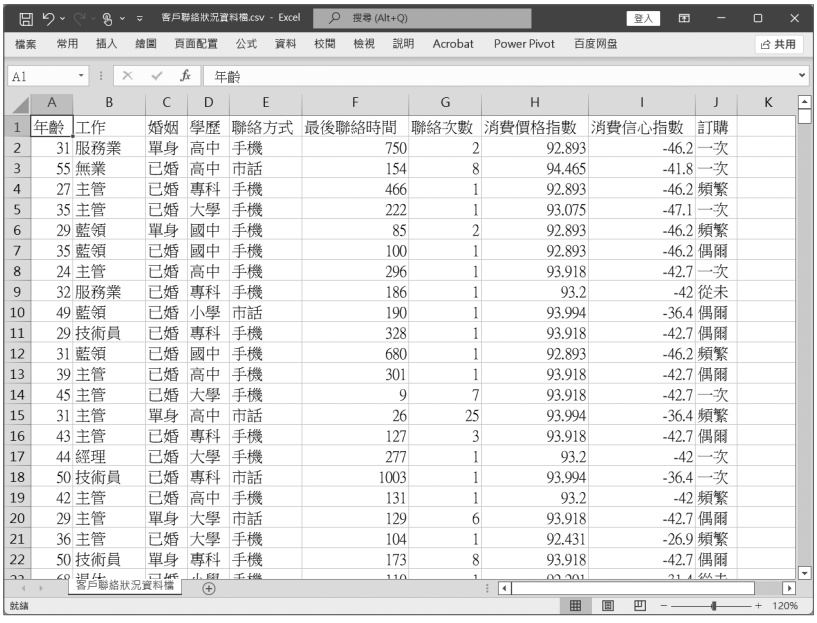
如果要將產生的圖片存檔，可在 flow() 方法設定 save_to_dir、save_prefix 及 save_format 參數即可。例如，在剛才的範例中產生了圖片後儲存的方式如下：

```
 6  imgGen = ImageDataGenerator(shear_range=50.0, fill_mode='reflect')
 7  n = 5
 8  img = cv2.imread('cat.jpg')
 9  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #BGR轉成RGB
10  img = np.array(img, dtype=np.float32)
11  img = np.expand_dims(img, 0) #輸入圖片要是四維
12  flowGen = imgGen.flow(img, save_to_dir='.',
13                        save_prefix='cat',
14                        save_format='jpg',
15                        batch_size=10)
16
17  for i in range(n):
18    img = flowGen[0][0, :, :, :].astype(np.uint8)
```

6.3 非數值資料轉換

機器學習演算法是一種數學模型，需在「數值」數據基礎上進行運算。然而，我們蒐集的資料集，大多會包含文字資料，例如性別 (男、女 ）、學歷 (大學、中學、小學) 等，這些非數值資料需經預先處理轉換為數值資料後，才能傳送給機器學習演算法進行處理。

這裡將使用 < 客戶聯絡狀況資料檔 .csv> 來進行非數值資料轉換，其中有 4100 筆銀行舉辦活動後聯絡客戶狀況的資料：



	A	B	C	D	E	F	G	H	I	J	K
1	年齡	工作	婚姻	學歷	聯絡方式	最後聯絡時間	聯絡次數	消費價格指數	消費信心指數	訂購	
2	31	服務業	單身	高中	手機	750	2	92.893	-46.2	一次	
3	55	無業	已婚	高中	市話	154	8	94.465	-41.8	一次	
4	27	主管	已婚	專科	手機	466	1	92.893	-46.2	頻繁	
5	35	主管	已婚	大學	手機	222	1	93.075	-47.1	一次	
6	29	藍領	單身	國中	手機	85	2	92.893	-46.2	頻繁	
7	35	藍領	已婚	國中	手機	100	1	92.893	-46.2	偶爾	
8	24	主管	已婚	高中	手機	296	1	93.918	-42.7	一次	
9	32	服務業	已婚	專科	手機	186	1	93.2	-42	從未	
10	49	藍領	已婚	小學	市話	190	1	93.994	-36.4	偶爾	
11	29	技術員	已婚	專科	手機	328	1	93.918	-42.7	偶爾	
12	31	藍領	已婚	國中	手機	680	1	92.893	-46.2	頻繁	
13	39	主管	已婚	高中	手機	301	1	93.918	-42.7	偶爾	
14	45	主管	已婚	大學	手機	9	7	93.918	-42.7	一次	
15	31	主管	單身	高中	市話	26	25	93.994	-36.4	頻繁	
16	43	主管	已婚	專科	手機	127	3	93.918	-42.7	偶爾	
17	44	經理	已婚	大學	手機	277	1	93.2	-42	一次	
18	50	技術員	已婚	專科	市話	1003	1	93.994	-36.4	一次	
19	42	主管	已婚	高中	手機	131	1	93.2	-42	頻繁	
20	29	主管	單身	大學	市話	129	6	93.918	-42.7	偶爾	
21	36	主管	已婚	大學	手機	104	1	92.431	-26.9	頻繁	
22	50	技術員	單身	專科	手機	173	8	93.918	-42.7	偶爾	

工作、婚姻、學歷、聯絡方式及訂購 5 個特徵是文字資料，若要做為機器學習資料集使用，需將這些特徵轉換為數值資料。

將非數值資料轉換為數值資料常用的方法有 **對應字典法**、**標籤編碼法** 及 **One-Hot 編碼法** 三種。

6.3.1 對應字典法

如果要轉換的特徵值不多，可用 **對應字典法** 直接將文字替換為數值。對應字典法是先將要替換的文字與對應的數值建立成字典，再以 Dataframe 的 df.replace() 或 df.map() 方法進行轉換。

資料取代：df.replace()

以「婚姻」特徵為例，首先以 df.unique() 方法查看婚姻的所有特徵值，以便使用這些特徵值建立字典：

```
[28]  1  import pandas as pd
      2  df = pd.read_csv('客戶聯絡狀況資料檔.csv')
      3
      4  df['婚姻'].unique()
```

```
array(['單身', '已婚', '離婚', '未知'], dtype=object)
```

婚姻特徵有單身、已婚、離婚、未知四種值。

接著建立特徵值與對應數值的字典，語法為：

```
字典變數 = {'特徵值 1': 數值 1, '特徵值 2': 數值 2, ……}
```

最後即可使用 df.replace() 方法轉換為數值資料，語法為：

```
DataFrame 欄位.replace( 字典變數 , inplace=True)
```

「inplace=True」表示取代後的結果將直接更新原始的 DataFrame 中的資料。

例如，想將「婚姻」的值：單身、已婚、離婚、未知，分別對應更改為 1~4 的數值：

```
[29]  1  dict1 = {'單身':1, '已婚':2, '離婚':3, '未知':4}
      2  df['婚姻'].replace(dict1, inplace=True)
      3  df
```

	年齡	工作	婚姻	學歷	聯絡方式	最後聯絡時間	聯絡次數	消費價格指數	消費信心指數	訂購
0	31.0	服務業	1	高中	手機	750.0	2	92.893	-46.2	一次
1	55.0	無業	2	高中	市話	754.0	8	94.465	-41.8	一次
2	27.0	主管	2	專科	手機	466.0	1	92.893	-46.2	頻繁

已轉換為數值

在機器學習實作時，進行預測所得的類別結果是數值，使用者常需要將數值資料還原為對應的文字。要達成此目的，只要將字典改為數值對應文字型式即可還原：

```
字典變數 = {數值1:'特徵值1', 數值2:'特徵值2', ……}
```

例如，想將「婚姻」的數值再換回原來的文字資料：

```
[30]  1  dict2 = {1:'單身', 2:'已婚', 3:'離婚', 4:'未知'}
      2  df['婚姻'].replace(dict2, inplace=True)
      3  df
```

	年齡	工作	婚姻	學歷	聯絡方式	最後聯絡時間	聯絡次數	消費價格指數	消費信心指數	訂購
0	31.0	服務業	單身	高中	手機	750.0	2	92.893	-46.2	一次
1	55.0	無業	已婚	高中	市話	154.0	8	94.465	-41.8	一次
2	27.0	主管	已婚	專科	手機	466.0	1	92.893	-46.2	頻繁

已還原為文字

資料對應：df.map()

以「學歷」特徵為例，以 df.unique() 方法查看特徵值建立字典，然後使用 df.map() 方法將文字資料轉換為數值，語法為：

```
DataFrame 欄位 = DataFrame 欄位 .map( 字典變數 )
```

```
1  df['學歷'].unique()
```

```
array(['高中', '專科', '大學', '國中', '小學', '未知', '文盲'], dtype=object)
```

```
1  dict3 = {'高中':1, '專科':2, '大學':3, '國中':4, '小學':5, '未知':6, '文盲':7}
2  df['學歷'] = df['學歷'].map(dict3)
3  df
```

	年齡	工作	婚姻	學歷	聯絡方式	最後聯絡時間	聯絡次數	消費價格指數	消費信心指數	訂購
0	31.0	服務業	單身	1	手機	750.0	2	92.893	-46.2	一次
1	55.0	無業	已婚	1	市話	154.0	8	94.465	-41.8	一次
2	27.0	主管	已婚	2	手機	466.0	1	92.893	-46.2	頻繁
3	35.0	主管	已婚	3	手機	222.0	1	93.075	-47.1	一次
4	29.0	藍領	單身	4	手機	85.0	2	92.893	-46.2	頻繁
...	...	...	...	...	...	...	...	...	...	...

6.5 使用 Pandas 進行特徵選擇

Pandas 提供三種計算相關係數的方法：**皮爾森 (Pearson)**、**肯德爾 (Kendall)** 及 **斯皮爾曼 (Spearman)**，這三種方法都適用於**迴歸分析** (即目標為數值的分析)，例如房價、股價預測。

這裡將使用 < 地區房價資料檔 .csv> 房價特徵資料集為範例，內含 500 筆資料，前 12 個特徵為影響房價的因素，最後一個特徵為房價：

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	犯罪率	豪宅比	公設比	臨公園	NO濃度	房間數	屋齡	賣場距離	捷運距離	繳稅率	師生比	低收入比	房價
2	0.06211	40	2.5	0	0.429	6.49	44.4	8.7921	1	335	19.7	5.98	22.9
3	0.55778	0	43.78	0	0.624	6.335	98.2	2.1107	4	437	21.2	16.96	18.1
4	3.69311	0	36.2	0	0.713	6.376	88.4	2.5671	24	666	20.2	14.65	17.7
5	0.00906	90	5.94	0	0.4	7.088	20.8	7.3073	1	285	15.3	7.85	32.2
6	0.03537	34	12.18	0	0.433	6.59	40.4	5.4917	7	329	16.1	9.5	22
7	0.03445	82.5	4.06	0	0.415	6.162	38.4	6.27	2	348	14.7	7.43	24.1
8	0.09164	0	21.62	0	0.413	6.065	7.8	5.2873	4	305	19.2	5.52	22.8
9	1.38799	0	16.28	0	0.538	5.95	82	3.99	4	307	21	27.71	13.2
10	0.0578	0	4.92	0	0.488	6.98	58.4	2.829	3	193	17.8	5.04	37.2
11	0.18337	0	55.48	0	0.609	5.414	98.3	1.7554	4	711	20.1	23.97	7
12	0.01301	35	3.04	0	0.442	7.241	49.3	7.0379	1	284	15.5	5.49	32.7
13	0.06151	0	10.38	0	0.515	5.968	58.5	4.8122	5	224	20.2	9.29	18.7
14	0.10793	0	17.12	0	0.52	6.195	54.4	2.7778	5	384	20.9	13	21.7
15	0.03237	0	4.36	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	33.4
16	0.10084	0	20.02	0	0.547	6.715	81.6	2.6775	6	432	17.8	10.16	22.8
17	0.7842	0	16.28	0	0.538	5.99	81.7	4.2579	4	307	21	14.67	17.5
18	0.08265	0	27.84	0	0.437	6.127	18.4	5.5027	4	289	16	8.58	23.9
19	0.10659	80	3.82	0	0.413	5.936	19.5	10.5857	4	334	22	5.57	20.6
20	9.91655	0	36.2	0	0.693	5.852	77.8	1.5004	24	666	20.2	29.97	6.3
21	0.0315	95	2.94	0	0.403	6.975	15.3	7.6534	3	402	17	4.56	34.9
22	0.17899	0	19.38	0	0.585	5.67	28.8	2.7986	6	391	19.2	17.6	23.1
23	3.53501	0	39.16	1	0.871	6.152	82.6	1.7455	5	403	14.7	15.02	15.6

6.5.1 使用 Pandas 計算相關係數

Pandas 計算相關係數的語法為：

```
相關變數 = DataFrame 變數.corr(method= 計算方式 , min_periods= 數值 )
```

- **method**：設定使用的計算相關係數方法：pearson(皮爾森相關係數) 為預設值、kendall(肯德爾相關係數)、spearman(斯皮爾曼相關係數)。
- **min_periods**：設定最小資料數量。預設值為 1。

7.2 K-means 演算法

K-means 演算法 是一種非監督機器學習演算法，屬於「聚類」演算法，可以將資料分為指定數量的群組。

7.2.1 K-means 演算法原理

以學校開學時的新班級為例：開學時每個同學互不認識，所以每個同學都是獨立個體，每個同學相當於資料初始狀態。隨著時間流逝，同學間逐漸形成小團體，相當於資料分成不同的群組。

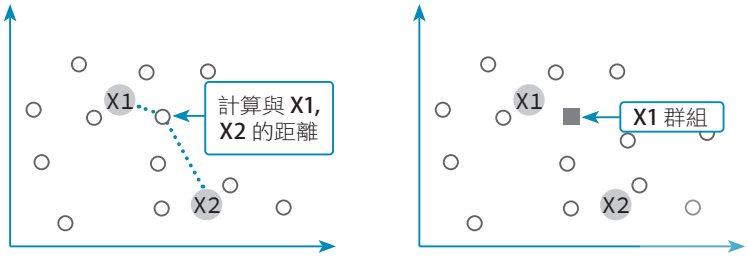
同學為什麼會形成小團體呢？可能是性別因素 (男、女)，可能是興趣因素 (喜歡打球、音樂等)，以及其他各種因素；相當於資料會因各種特徵值相近而聚集成群組。人類會因彼此相處而熟識，自然而然的形成各種小團體，但資料要如何因特徵值相近而聚集成群組呢？關鍵就是計算資料之間的特徵值歐式距離，歐式距離越小表示資料特徵越接近，然後將特徵相近的資料聚類成群組。

K-means 演算法的「K」表示群組數量，即要將資料分為 K 個群組，「means」表示每個群組的中心，稱為「群心」，K-means 演算法藉由不斷更新「群心」位置來達成將資料聚類成群組。

K-means 運作步驟

下面以將資料分為 2 個群組 (K=2) 做說明：圓圈為原始資料，正方形為第一個群組資料，三角形點為第二個群組資料，運作步驟如下：

1. 在資料範圍內任取兩個位置 X1 及 X2 做為兩個群組的群心。取一個資料計算與兩個群心的距離，將資料劃分為較近群組資料。下面圖形圓點距 X1 較近，故視為 X1 群組資料。



1. 使用 calinski_harabasz_score 之前需先含入模組：

```
from sklearn.metrics import calinski_harabasz_score
```

2. 使用 calinski_harabasz_score 模組的語法為：

```
評估變數 = calinski_harabasz_score(原始資料, KMeans變數.labels_)
```

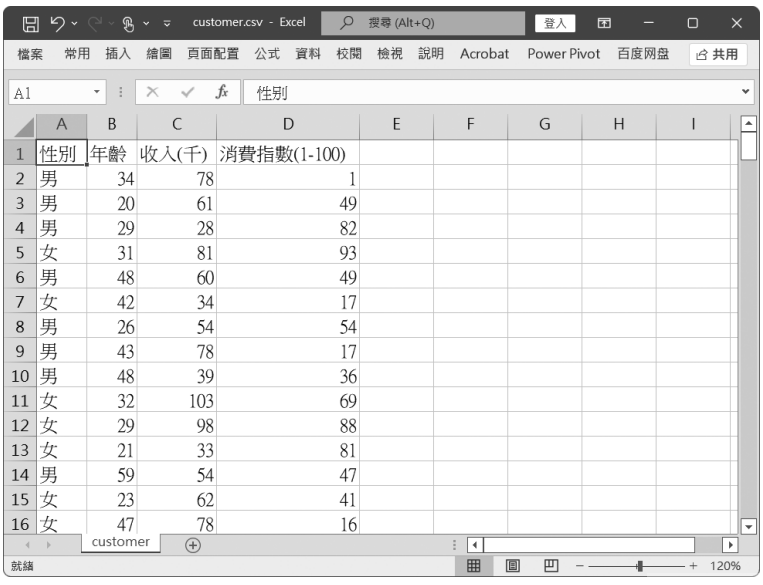
例如，評估變數為 metric，原始資料為 df，KMeans 變數為 km：

```
metric = calinski_harabasz_score(df, km.labels_)
```

7.2.3 K-means 應用：信用卡客戶分群

信用卡客戶資料來源

這裡將使用 <customer.csv> 以 K-means 演算法將信用卡客戶分群，其中包含 200 筆資料，所有欄位皆為持卡者個人資料：



	A	B	C	D
1	性別	年齡	收入(千)	消費指數(1-100)
2	男	34	78	1
3	男	20	61	49
4	男	29	28	82
5	女	31	81	93
6	男	48	60	49
7	女	42	34	17
8	男	26	54	54
9	男	43	78	17
10	男	48	39	36
11	女	32	103	69
12	女	29	98	88
13	女	21	33	81
14	男	59	54	47
15	女	23	62	41
16	女	47	78	16

使用 K-means 進行信用卡客戶分群

由於「性別」欄位是文字資料，無法進行機器學習的數學運算，所以使用字典對照法將文字資料轉換男為 1，女為 2 的數值資料：

```
[2]  1 import pandas as pd
     2 import numpy as np
     3 from sklearn.cluster import KMeans
     4 from sklearn.metrics import calinski_harabasz_score
     5
     6 df = pd.read_csv('customer.csv')
     7 dict1 = {'男':1, '女':2}
     8 df['性別'].replace(dict1, inplace=True)
     9 df
```

	性別	年齡	收入(千)	消費指數(1-100)
0	1	34	78	1
1	1	20	61	49
2	1	29	28	82
3	2	31	81	93
4	1	48	60	49
...	...	...	...	...

接著就可用 K-means 演算法進行分群：通常會根據需求決定 K 值，即分為多少群，此處以分為 3 群為例：

```
[3]  1 km = KMeans(n_clusters=3)
     2 km.fit(df)
     3 km.labels_
```

```
array([0, 2, 2, 1, 2, 2, 2, 0, 2, 1, 1, 2, 2, 2, 0, 2, 2, 2, 0, 2, 0, 1,
       0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 2, 1, 1, 2, 2, 0, 2, 1, 2,
       2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 0, 1, 2, 0, 2, 2, 2, 1, 1,
       2, 1, 2, 0, 0, 1, 2, 2, 2, 2, 1, 0, 2, 2, 0, 2, 2, 0, 0, 1, 2, 2,
       2, 2, 2, 2, 1, 1, 2, 2, 0, 1, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       2, 1, 1, 1, 2, 2, 2, 1, 1, 2, 0, 1, 2, 2, 2, 2, 2, 0, 1, 2, 2, 2,
       2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 1, 0, 0, 2, 2, 2, 2, 0, 2, 0, 1, 1,
       2, 2, 1, 2, 0, 2, 1, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1,
       2, 1, 2, 2, 2, 2, 2, 0, 2, 2, 0, 1, 2, 0, 0, 2, 1, 0, 2, 1, 2, 2,
       1, 1], dtype=int32)
```

每個人執行結果可能編號順序會與上圖不同，但群組分布是相同的。

可以將分組結果加入原始資料集，例如將新增的欄位命名為「類別」：

```
[4]  1 df['類別'] = km.labels_
     2 df.head()
```

	性別	年齡	收入(千)	消費指數(1-100)	類別
0	1	34	78	1	0
1	1	20	61	49	2
2	1	29	28	82	2
3	2	31	81	93	1
4	1	48	60	49	2

此資料集就可做為監督式機器學習的資料集：原始 4 個欄位為特徵值，新增的「類別」欄位為目標值。

「類別」欄位值 0、1、2 代表什麼意義呢？這需要觀察其資料歸納才能得知。以下程式會列出類別為 0 的前 30 筆資料：

```
[ ]  1 df2 = df[df['類別']==0]
     2 df3 = df2.iloc[0:30, :]
     3 df3
```

將第一列程式中「0」分別改為 1、2，就可分別列出類別為 1、2 的前 30 筆資料，仔細觀察這三組資料。

	性別	年齡	收入(千)	消費指數(1-100)	類別
3	2	31	81	93	0
9	2	32	103	69	0
10	2	29	98	88	0
21	1	32	126	74	0
36	1	28	77	97	0
37	1	32	73	73	0
42	1	39	78	88	0
45	1	30	137	83	0
51	1	28	87	75	0
58	1	40	71	95	0

▲ 類別為 0

	性別	年齡	收入(千)	消費指數(1-100)	類別
1	1	20	61	49	1
2	1	29	28	82	1
4	1	48	60	49	1
5	2	42	34	17	1
6	1	26	54	54	1
8	1	48	39	36	1
11	2	21	33	81	1
12	1	59	54	47	1
13	2	23	62	41	1
15	1	68	63	43	1

▲ 類別為 1

	性別	年齡	收入(千)	消費指數(1-100)	Class
0	1	34	78	1	2
7	1	43	78	17	2
14	2	47	78	16	2
18	1	42	86	20	2
20	2	45	126	28	2
22	2	47	120	16	2
24	2	34	103	23	2
32	1	19	74	10	2
33	1	20	73	5	2
34	1	37	78	1	2

▲ 類別為 2

結論：「類別」為 0 的客戶年收入較低，年齡及消費分布較廣，可視為「一般客戶」；「類別」為 1 的客戶年收入較高，年齡約在 20-40 歲，消費力相當高，可視為「優質客戶」；「類別」為 2 的客戶年收入較高，年齡約在 20-60 歲，但消費力相當低，可視為「待開發客戶」。

8.5.3 隨機森林演算法範例：葡萄酒種類判斷

在範例資料夾中的 <wine.csv> 是一個葡萄酒資料集，內含 160 筆資料，第一個欄位為「酒種類」，1 表示紅葡萄酒，2 表示粉紅葡萄酒，3 表示白葡萄酒；第 2 到 14 個欄位為葡萄酒各種持性資料。這裡使用 2 到 14 個個欄位為特徵值，第 1 個欄位「種類」為目標值進行隨機森林演算法訓練。

	酒種類	酒精	蘋果酸	灰分	灰分鹼性	鎂	總酚	黃酮類	非黃酮酚	原花青素	顏色強度	色相	透光度	脯氨酸
0	1	13.74	1.67	2.25	16.4	118	2.60	2.90	0.21	1.62	5.85	0.92	3.20	1060
1	1	13.87	1.90	2.80	19.4	107	2.95	2.97	0.37	1.76	4.50	1.25	3.40	915
2	3	12.96	3.45	2.35	18.5	106	1.39	0.70	0.40	0.94	5.28	0.68	1.75	675
3	3	12.25	3.88	2.20	18.5	112	1.38	0.78	0.29	1.14	8.21	0.65	2.00	855
4	2	11.65	1.67	2.62	26.0	88	1.92	1.61	0.40	1.34	2.60	1.36	3.21	562

下面程式以葡萄酒資料集進行隨機森林演算法判斷葡萄酒是哪一個種類。

```
[27]  1 from sklearn.model_selection import train_test_split
      2 from sklearn.ensemble import RandomForestClassifier
      3 import pandas as pd
      4
      5 df = pd.read_csv('wine.csv')
      6 x = df.iloc[:, 1:14]
      7 y = df.iloc[:, 0]
      8 x_train, x_test, y_train, y_test = train_test_split(
      9     x, y, test_size=0.2)
     10 rf = RandomForestClassifier(n_estimators=10,
     11                              min_samples_split=30)
     12 rf.fit(x_train, y_train)
     13 score = rf.score(x_test, y_test)
     14 print(score)
```

```
0.9375
```

程式說明

- 2　載入隨機森林模組。
- 5　讀入酒種類資料集。
- 6　以第 2 到 14 個欄位做為特徵值。
- 7　以第 1 個欄位「類別」做為目標值。
- 10-12　進行隨機森林模型訓練。
- 13-14　進行預測並顯示準確率。

由結果可見準確率在 0.91~0.97 之間，準確率相當高。

11.1 認識卷積神經網路 (CNN)

卷積神經網路 (Convolutional Neural Network) 簡稱 CNN，它在圖片辨別上甚至可以做到比人類還精準的程度。CNN 也是模仿人類大腦的認知方式，例如辨識一個圖像，會先注意到顏色鮮明的點、線、面，然後再將它們構成一個個不同的形狀，如眼睛、鼻子等，這種抽象化的過程就是 CNN 演算法建立模型的方式。卷積層就是由點的比對轉成局部的比對，透過一塊塊的特徵研判，逐步堆疊綜合比對結果，就可以得到比較好的辨識結果。

11.1.1 為什麼需要卷積神經網路？

深度神經網路處理圖片時，是將圖片中的每一個像素 (Pixel) 排成一列非常長的向量輸入到神經網路中處理，這樣的方式有兩大缺點：

- **權重數量龐大**：當圖片較大時，像素很多，造成需要調整的權重非常多。以一張長寬皆為 512 像素的彩色圖片，第一層隱藏層有 100 個神經元為例，該層隱藏層的權重數為 512 X 512 X 3 X 100 = 39321600，將近四千萬個權重，可怕吧！
- **破壞圖片結構**：深度神經網路輸入層是一維向量，所以需先將二維圖片轉換為一維才傳送給神經網路，此時二維圖片中許多像素就失去原先在圖片中的特性，導致部分圖片特徵遺失。

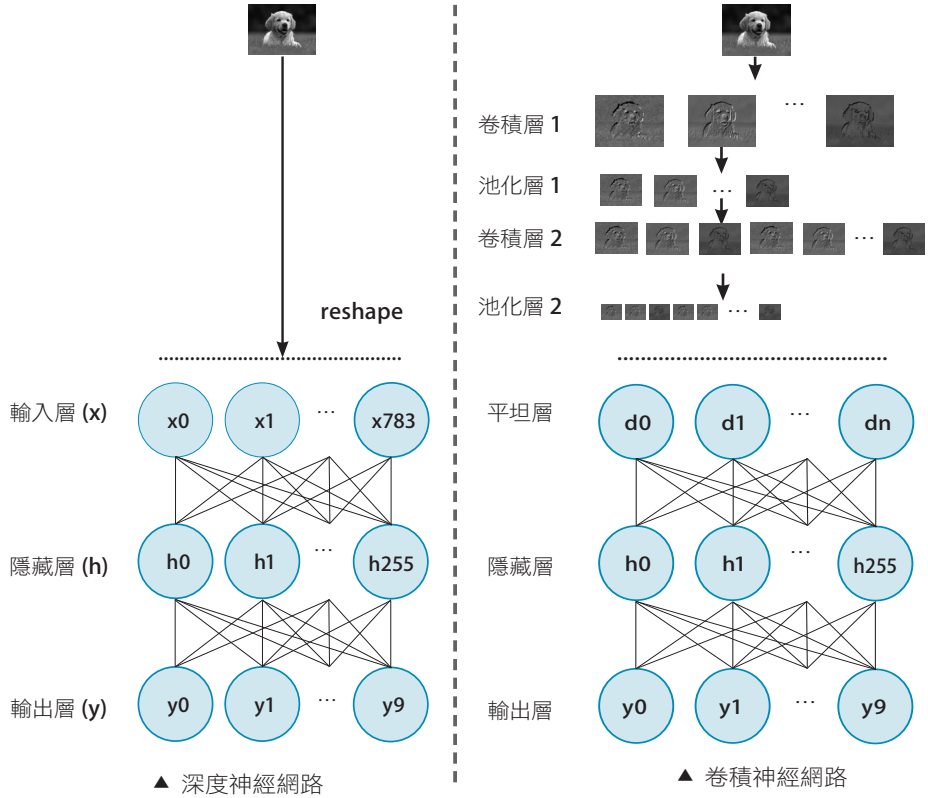
為了解決這些問題，於是卷積神經網路誕生了！卷積神經網路不是將整張圖片輸入神經網路，而是先對局部圖片提取特徵，再將特徵輸入神經網路，這樣就保持了圖片二維像素的特性，同時使用許多局部特徵來辨識圖片，比直接辨識整張圖片的效果更精確。

卷積神經網路在深度神經網路新增了兩種演算方法，來達成提取局部圖片特徵的目的：

- **卷積層**：提取局部圖片特徵。
- **池化層**：降低維度。

這兩種演算方法會交替使用，最後再連接深度神經網路，構成卷積神經網路。

深度神經網路與卷積神經網路結構比較：(以 2 層卷積層及池化層為例)



▲ 深度神經網路　▲ 卷積神經網路

11.1.2 卷積層

卷積層 (Convolution Layer) 是不斷對局部圖片提取特徵，實作時是以 **卷積核** (kernel) 對圖片局部區塊進行運算提取特徵，接著移動卷積核對另一個局部區塊提取特徵。

卷積核 (kernel) 與移動幅度 (stride)

卷積核是每次提取圖片特徵的區塊大小，通常會使用 3X3、5X5 或 7X7 區塊。移動幅度是指卷積核完成一次區塊特徵提取後，向右或向下移動指定像素繼續進行區塊特徵提取，通常是使用 1 或 2 個像素。

13.2 遷移學習

遷移學習 (Transfer Learning) 是將一個場景中學到的知識「遷移」到另一個場景中。以貓狗分類的模型為例，將其遷移到其他相似的任務上，例如用來分辨麻雀與燕子，因為它們都是以圖片來辨識物件，所以屬於相同領域，抽取特徵的方法是相同的。

13.2.1 認識遷移學習

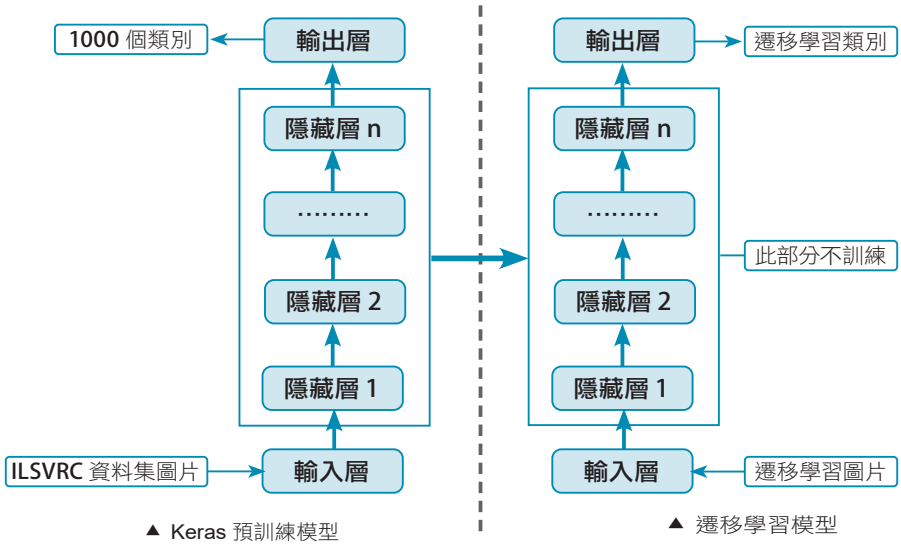
使用深度學習解決問題的過程中，最常見的障礙有兩個：

- **模型有海量的參數需要訓練**：以 Keras 內建的預訓練模型為例，參數數量少則數千萬個，VGG16 及 VGG19 則達到一億四千萬個，如此龐大數量的參數，必須效能相當良好的電腦設備才能執行，而訓練一次可能要花費數天時間，這樣的設備及時間是一般使用者無法負荷的。
- **需要海量資料進行訓練**：以訓練辨識圖片中物件的模型為例，通常一個物件的圖片需要超過 1000 張，才能有較佳訓練結果。以 Keras 內建的預訓練模型為例，ILSVRC 挑戰賽使用的圖片共分 1000 個類別，圖片超過 1000000 張，一般使用者應無能力蒐集如此大量的圖片資料。

人類可以將以前學到的知識應用於解決新的問題，這樣就可更快解決問題或取得更好效果。以同樣原理推論：是否有可能存在一個已經訓練好的模型，並將其學習得到的特徵應用到我們所需要預測的資料上呢？於是遷移學習誕生了！我們將已訓練模型所得到的參數鎖定，直接將這些訓練好的特徵「遷移」到需要進行下一步訓練的模型當中，這樣就能以少量資料在短時間輕易訓練出效果不錯的新模型。

了解遷移學習的原理後，要如何實作呢？首先需取得一個良好的模型，例如要製作辨識圖片物件的模型，Keras 預訓練模型就是非常適合的模型。進行遷移模型訓練時，移掉最頂層，例如 Keras 預訓練模型的頂層就是 1000 個輸出的全連結層，換上新的頂層，例如要訓練辨識 5 個物件的模型，需為 5 個輸出的全連結層。訓練模型時，只訓練新更換的全連結輸出層。這樣的操作就是將 Keras 預訓練模型的底層神經網路當做特徵提取器來使用。

遷移學習示意圖如下：



▲ Keras 預訓練模型　　▲ 遷移學習模型

13.2.2 蒐集資料：花朵資料集

這裡使用 Tensorflow 官網提供的花朵資料集進行遷移學習模型訓練。花朵資料集包含雛菊 (daisy)、蒲公英 (dandelion)、玫瑰 (rose)、向日葵 (sunflower)、鬱金香 (tulip) 5 種花朵圖片，每種花朵圖片數量在 600 張到 900 張之間，這裡利用這些花朵圖片做為遷移學習模型訓練資料。

下載花朵資料集

如果要在本機中使用，請由瀏覽器開啟下載網址：「http://download.tensorflow.org/example_images/flower_photos.tgz」，此時會自動下載圖片壓縮檔 <flower_photos.tgz>。解壓縮後會產生 <flower_photos> 資料夾，其中含有 <daisy>、<dandelion>、<roses>、<sunflowers>、<tulips> 5 個資料夾，分別存放所屬花朵圖片。

如果要在 Colab 中使用，可利用「wget」命令下載資料集的壓縮檔，命令為：

```
!wget http://download.tensorflow.org/example_images/flower_photos.tgz
```

執行後在 Colab 根目錄會產生 <flower_photos.tgz> 檔。

13.2.4 花朵辨識卷積神經網路模型

圖形物件辨識需要極大量圖片做為訓練資料才能得到較佳的效果：手寫數字辨識 (Mnist) 模型使用六萬張圖片，貓狗圖片辨識使用近三萬張圖片，Keras 預訓練模型則使用超過百萬張圖片。花朵辨識圖片 5 個類別全部只有 3670 張圖片，是否能用來訓練模型？這裡將以卷積神經網路來訓練花朵圖片辨識模型，做為遷移學習模型的對照。

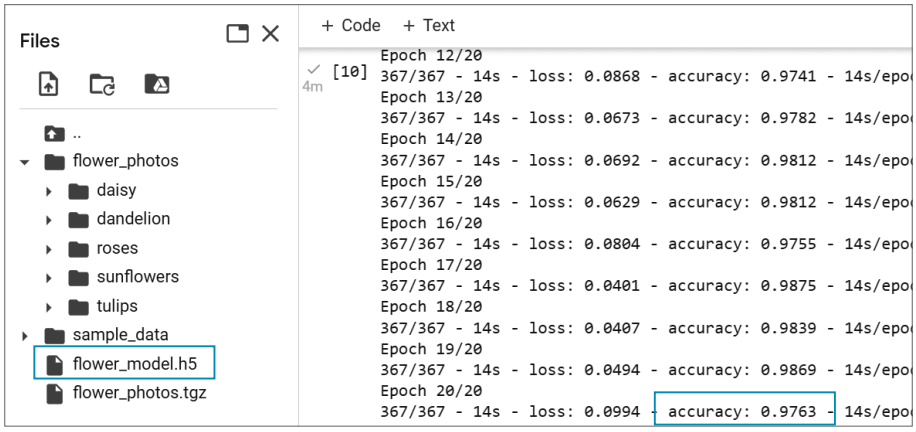
建立花朵辨識卷積神經網路模型

```
[ ]  1  from keras.models import Sequential
     2  from keras.layers import Conv2D, MaxPooling2D, \
     3                           Dropout, Flatten, Dense
     4
     5  model = Sequential()
     6  model.add(Conv2D(filters=8, kernel_size=(5,5), padding='same',
     7                   input_shape=(299, 299, 3), activation='relu'))
     8  model.add(MaxPooling2D(pool_size=(2, 2)))
     9  model.add(Dropout(0.2))
    10  model.add(Conv2D(filters=16, kernel_size=(5,5), padding='same',
    11                   activation='relu'))
    12  model.add(MaxPooling2D(pool_size=(2, 2)))
    13  model.add(Dropout(0.2))
    14  model.add(Conv2D(filters=32, kernel_size=(5,5), padding='same',
    15                   activation='relu'))
    16  model.add(MaxPooling2D(pool_size=(2, 2)))
    17  model.add(Dropout(0.2))
    18  model.add(Flatten())
    19  model.add(Dropout(0.2))
    20  model.add(Dense(units=128, activation='relu'))
    21  model.add(Dense(units=5, activation='softmax'))
    22  model.compile(loss='categorical_crossentropy', optimizer='adam',
    23                metrics=['accuracy'])
    24  model.fit(x=trainX, y=trainY, epochs=20, batch_size=10, verbose=2)
    25  model.save('flower_model.h5')
```

程式說明

- 6　輸入圖片尺寸為 299x299 的彩色圖片。
- 21　輸出層為 5 個類別。
- 24　訓練時沒有「validation_split」參數，表示沒有驗證資料，全部資料都做為訓練資料。
- 25　訓練完成後儲存的模型檔案為 <flower_model.h5>。

執行結果：



因為沒有驗證資料，顯示的訓練資訊沒有驗證損失函式值及驗證資料正確率。上圖資訊顯示訓練 20 次後，對訓練資料的正確率達到 0.9763。

預測未知花朵圖片：卷積模型

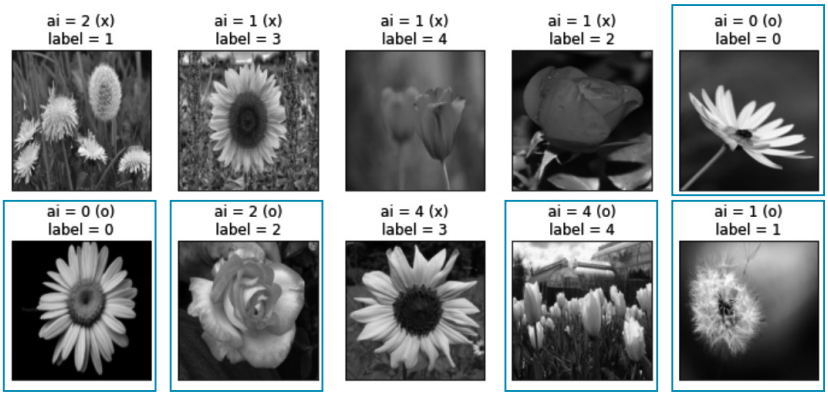
模型的優劣必須以預測未知資料是否準確做為依據，範例資料夾中的 10 張圖片，是分別在 5 個類別中各選取 2 張與訓練圖片不同的圖片組成，做為測試模型效果之用。

程式碼如下：

```
[ ]
import numpy as np
import matplotlib.pyplot as plt
from keras.models import load_model
import glob,cv2
from keras.applications.inception_v3 import preprocess_input

def show_images_labels_predictions(images, labels,
                                   predictions,start_id, num=10):
    plt.figure(figsize=(12, 14))
    if num>25: num=25
    for i in range(0, num):
        ax=plt.subplot(5,5, 1+i)
        ax.imshow(images[start_id])
        if( len(predictions) > 0 ) :
            title = 'ai = ' + str(predictions[start_id])
            title += (' (o)' if predictions[start_id]==
                      labels[start_id] else ' (x)')
            title += '\nlabel = ' + str(labels[start_id])
```

執行結果：



可見到只有 4 張圖片預測正確，準確率只有 40%，而在 5 個類別情況下，任意猜測的準確率也有 20% (即 1/5)，此模型準確率僅比亂猜高一點。訓練資料準確率達 100%，未知資料準率只有 30%，這是由於訓練資料數量太少所造成的過擬合所致。

13.2.5 花朵辨識遷移學習模型

這裡將用遷移學習以 InceptionV3 模型做為基礎模型，移除其最上層的全連結輸出層，再加入辨識 5 種花朵的全連結輸出層，並以原來的花朵圖片進行訓練。

1. 首先載入模型相關模組：

```
[14]  1  from keras.applications.inception_v3 import InceptionV3
      2  from keras.models import Sequential
      3  from keras.layers import Dense
```

2. 建立 InceptionV3 模型：

```
[15]  1  inception = InceptionV3(weights='imagenet',
      2                           include_top=False, pooling="avg")
```

「include_top = False」表示不載入最頂部全連結層，相當於移除模型的最頂部全連結層。「pooling = avg」是配合 InceptionV3 使用 AveragePooling。