Chapter 8

資料科學導論

/ 洪暉鈞

因應人工智慧的發展，Python 成為資料科學中的首選工具，在各應用領域都有大量方便實用的套件。本章介紹 Python 中兩大重要套件 numpy 與 pandas，本章共有兩小節，包括：運用 array 處理一維陣列與多維矩陣運算以及運用 DataFrame 處理結構化具列索引與欄標的二維資料集及檔案輸出入。學習完本章後，讀者將學會讀取檔案資料成資料表，處理遺漏值與重複值，以及對資料表的合併、分割與轉置等操作。

8-1 科學運算 NumPy & pandas

在 Python 在處理資料科學相關運用的時候，有兩個重要的套件，一個是 NumPy，一個是 pandas。本小節將依序介紹 NumPy 以及 pandas 的重要功能與使用情境。

NumPy

NumPy 是 Python 中有關數值計算最重要的套件，其他許多提供科學計算的套件都是基於 NumPy 的陣列作為基礎。NumPy 的資料結構可以處理一維或多維矩陣運算，稱為陣列 (array) 物件。陣列可以是一維或多維度的，我們可以針對這種陣列資料統一執行一些數學運算，所以陣列之中的每一個元素都必須是相同型態，也讓程式執行更有效率，更適合用於數學運算與資料較龐大時的運算。

程式 E8-1-1 將建立一維跟二維的陣列，並且示範陣列常見的屬性，像是陣列的 .ndim 會回傳該陣列的維度；.shape 與 .size 則會回傳陣列的列行數與資料個數。

E8-1-1.py

```
01  import numpy as np        # 使用 numpy 採用 np 簡寫
02  array1 = np.array([1,2,3])    # 一維的 array
03  array2 = np.array([[1,2,3], [4,5,6]]) # 二維的 array
04  print(array2)
05  print('維度:', array2.ndim)
06  print('列行:', array2.shape)
07  print('個數:', array2.size)
```

執行結果：

```
[[1 2 3]
 [4 5 6]]
維度: 2
行列: (2, 3)
個數: 6
```

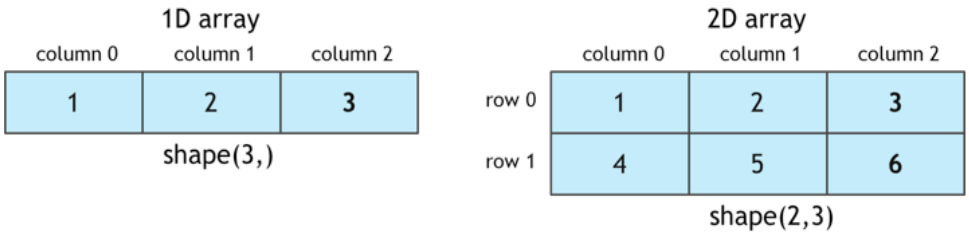


圖 8-1 numpy array

程式 E8-1-2 介紹建立一維陣列的不同方法，除了用 np.array 加上自定義的 list 以外，亦可以用 np.arange()、np.zeros()、np.ones() 等方式批次產生一維的 array。

E8-1-2.py

```
import numpy as np
a = np.array([3,6,9,12,15,18])
b = np.arange(6)   # 建立一個 size 為 6 的陣列，數據為 0~5
c = np.zeros(6)  # 數據全為 0，size 6
d = np.ones(6, dtype = np.int)# 數據為 1，size 6，型態整數
e = np.random.random(6)#size 為 6 的陣列，隨機亂數範圍為 [0.0,1.0]
print(a,b,c,d,e, sep='\n')
```

執行結果：

```
[ 3  6  9 12 15 18]
[0 1 2 3 4 5]
[ 0.  0.  0.  0.  0.  0.]
[1 1 1 1 1 1]
[ 0.9038974  0.19673665  0.8186001  0.13724558  0.05995684  0.9762709 ]
```

當有兩個陣列需要一起運算，如果這兩個陣列的個數相同，可以省略掉迴圈的使用而讓陣列每個元素進行相同的運算。例如程式 E8-1-3 中，我們希望能產生陣列 b 的內容是 0-5 共六個數字，然後陣列 b 的內容減掉陣列 a。此時 b-a 就會把這兩個陣列內六個對應的元素相減，產生新的陣列結果。不限定減法，加乘除或其他運算也可以，要特別注意的是這樣的運算必須是針對相同元素數量的陣列才可以。

E8-1-3.py

```
import numpy as np
a = np.array([3,6,9,12,15,18])
b = np.arange(6)
f = b-a
print(f)
print(f<-9)
print(f==-9)
```

執行結果：

```
[ -3  -5  -7  -9 -11 -13]
[False False False False  True  True]
[False False False  True False False]
```

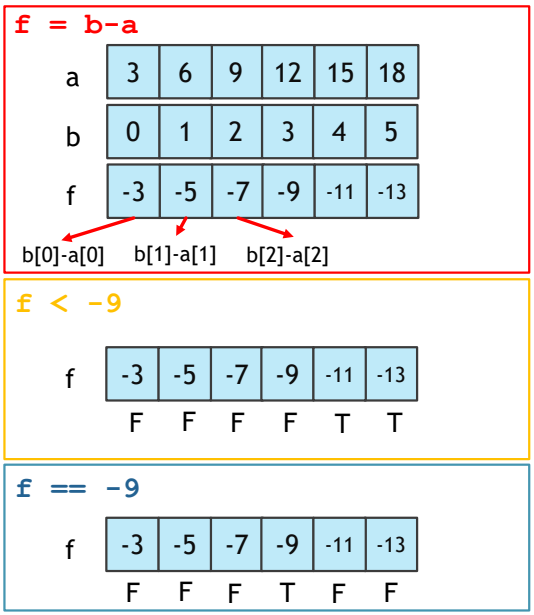


圖 8-2 一維 array 運算

如同一維陣列，我們也可以產生多維陣列。下面的例子就是我們產生不同的 2X3 的二維陣列，並且進行二維陣列中行、列的運算加總。

E8-1-4.py

```
import numpy as np
a = np.array([[3,6,9],[12,15,18]])
b = np.arange(6).reshape(2,3)
c = np.zeros((2,3))   # 數據全為 0
d = np.ones((2,3), dtype = np.int)   # 數據為 1，6 個
e = np.random.random((2,3))   # 亂數，6 個
print(a,b,c,d,e, sep='\n')
print(np.sum(a))
print(np.sum(a, axis=1))
```

執行結果：

```
[[ 3  6  9]
 [12 15 18]]
[[0 1 2]
 [3 4 5]]
[[ 0.  0.  0.]
 [ 0.  0.  0.]]
[[1 1 1]
 [1 1 1]]
[[ 0.59331307  0.94254093  0.06424767]
 [ 0.92768566  0.19110781  0.56701524]]
63
[18 45]
```

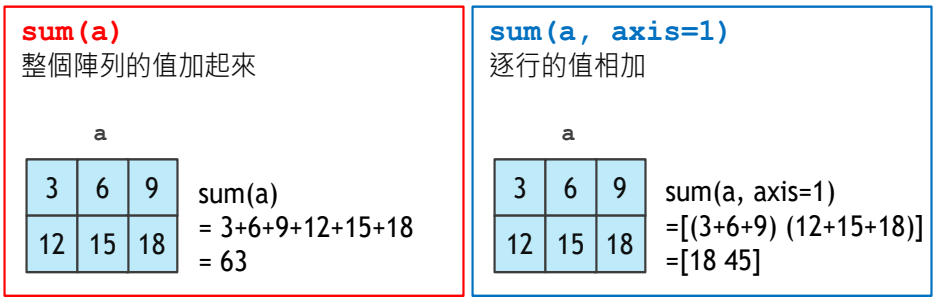


圖 8-3 二維 array 的 sum 運算

如同一維陣列，二維陣列也可以進行相對位置的運算 (加、減、乘、除)。

E8-1-5.py

```
import numpy as np
a=np.array([[2,4,6],[8,10,12]])
b=np.arange(6).reshape((2,3))
print(a)
print(b)
print(a-b)
print(a*b)
```

執行結果：

```
[[ 2  4  6]
 [ 8 10 12]]
[[0 1 2]
 [3 4 5]]
[[2 3 4]
 [5 6 7]]
[[ 0  4 12]
 [24 40 60]]
```

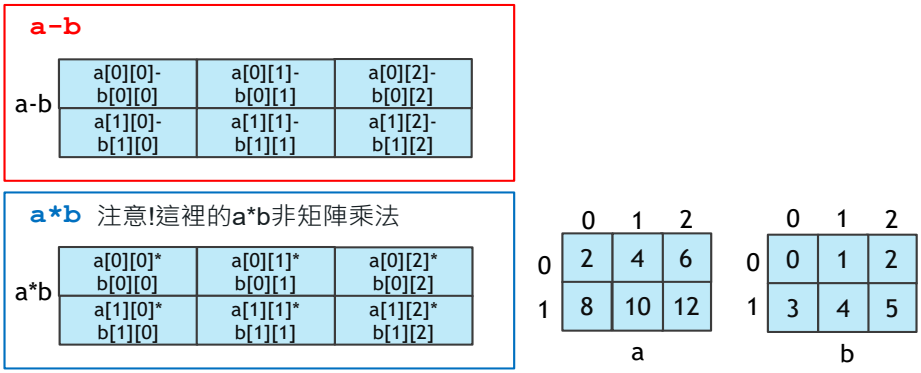


圖 8-4 二維矩陣減法與對應項乘法

而如果是要進行兩個矩陣之間的矩陣乘法，則須使用 np.dot() 來進行矩陣乘法。另外因為矩陣乘法的兩個矩陣行列數需要相反，所以下面的例子我們先將第二個矩陣進行轉置 (*transpose*)，將 2×3 的矩陣變成 3×2。

E8-1-6.py

```
import numpy as np
a=np.array([[2,4,6],[8,10,12]])
b=np.arange(6).reshape((2,3))
print(a)
print(b.T)  # 轉置
print(np.dot(a,b.T))
```

執行結果：

```
[[ 2  4  6]
 [ 8 10 12]]
[[0 3]
 [1 4]
 [2 5]]
[[ 16  52]
 [ 34 124]]
```

			0	3
			1	4
			2	5
2	4	6	16	52
8	10	12	34	124

圖 8-5 二維矩陣轉置與矩陣乘法

兩個 array 之間除了運算，還可以進行合併成一個 array，以下範例就是透過 np.vstack() 以及 np.hstack() 進行垂直與水平的合併。

E8-1-7.py

```
01  import numpy as np
02  A = np.array([2,4,6])
03  B = np.array([8,10,12])
04  print(np.vstack((A,B)))   #vertical
05  print(np.hstack((A,B)))   #horizontal
```

執行結果：

```
[[ 2  4  6]
 [ 8 10 12]]
[2 4 6 8 10 12]
```

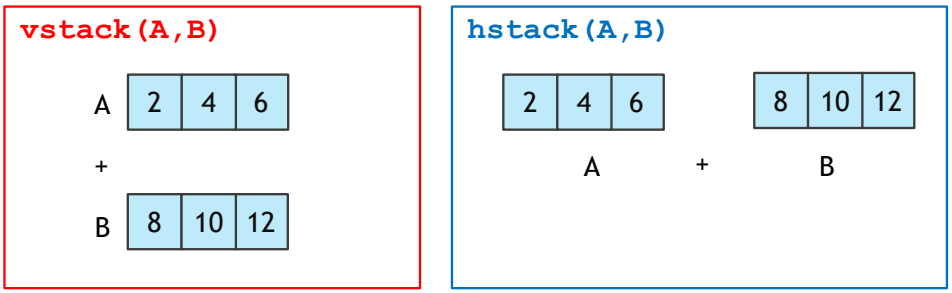


圖 8-6 array 垂直與水平的合併

同理，我們也可以將一個多維的 array，垂直或水平平均分隔成 2 個以上的 array，如下列程式所示：

E8-1-8.py

```
import numpy as np
a = np.arange(12).reshape((3,4))
print(a)
print(np.vsplit(a,3))
print(np.hsplit(a,2))
```

執行結果：

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[array([[0, 1, 2, 3]]), array([[4, 5, 6, 7]]), array([[ 8,  9, 10, 11]])]
[array([[0, 1],
       [4, 5],
       [8, 9]]), array([[ 2,  3],
       [ 6,  7],
       [10, 11]])]
```

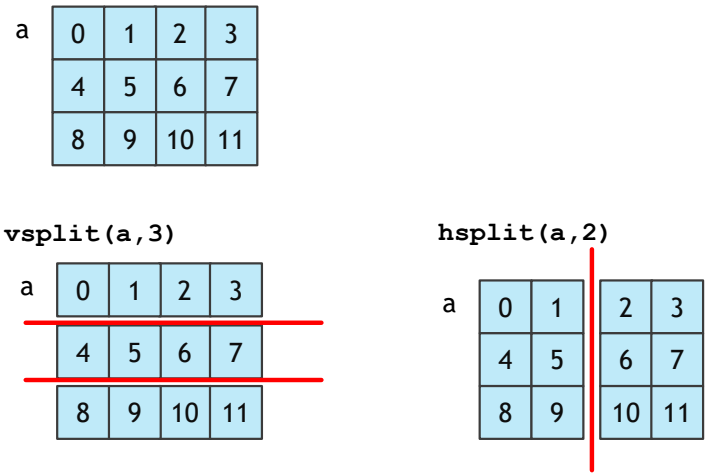


圖 8-7 array 切割

Pandas

pandas 是基於 numpy 的資料分析工具，能夠快速的處理結構化資料的大量資料結構和函數。接下來會跟大家介紹 pandas 之中常用的兩個資料結構：Series 跟 DataFrame。

Series 是由一組資料和一組索引組成，你可以視 Series 為有帶索引值的一維 array，以下示範建立一個 Series，可以看出他除了含有不同資料型態的內容，還帶有從 0 開始的索引值。

E8-1-9.py

```
01  import pandas as pd
02  import numpy as np
03  s = pd.Series([1,'abc','6',np.nan,44,1])
04  print(s)
```

執行結果：

```
0  1
1  abc
2  6
3  NaN
4  44
5  1
dtype: object
```

想一想：如果是 numpy 的 array 呢？

接下來我們來示範 pandas 的另外一個重要實用的資料結構 DataFrame。這是一個表格型的資料結構。有欄跟列的索引。DataFrame 可以用來處理結構化 (Table like) 的資料，有列索引與欄標籤的二維資料集，可以透過字典資料結構或是 array 來建立，但也可以利用外部的資料來讀取後來建立，例如： CSV 檔案、資料庫等等。

以下我們建立一個 DataFrame 的資料集，資料是隨機 (random)7×3 筆亂數。

E8-1-10.py

```
import pandas as pd
import numpy as np
# 生成方式一：用 array 的矩陣 (1)
df = pd.DataFrame(np.random.randn(7,3))
print(df)
```

執行結果：

```
          0         1         2
0  1.238052 -1.034129  0.222476
1 -0.457043 -1.252934  0.955214
2 -0.476281 -1.509945  0.573486
3  1.046944  0.143649 -0.138385
4 -1.268997  0.033301 -0.989308
5 -0.572510 -0.950910 -0.086820
6  0.250895 -0.889284 -0.149622
```

在我們建立好二維的資料表後，我們進一步在這已建立好的 DataFrame 加上列索引與欄標籤。

E8-1-11.py

```
import pandas as pd
import numpy as np
# 生成方式一：用 array 的矩陣 (2)
eat = np.random.randint(10,size=(7,3))*5+50
dates = pd.date_range('20170812',periods=7)
df0 = pd.DataFrame(eat)
# 加上欄位
df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐'])
print(df1)
```

執行結果：

```
            早餐 午餐 晚餐
2017-08-12 65 75 65
2017-08-13 65 55 70
2017-08-14 70 55 80
2017-08-15 70 85 95
2017-08-16 65 85 60
2017-08-17 60 80 60
2017-08-18 95 95 50
```

Pandas 的 DataFrame 很方便的一個功能是可以透過索引值來選擇或搜尋數據，以下示範兩種方式：

- df1[' 欄位名稱 '] 取得該欄位底下的值
- df1[起始索引 : 結束索引] 取得起始索引列到結束索引列前的資料

E8-1-11.py

```
10  print(df1[' 午餐 '])
11  print(df1[0:3])
```

執行結果：

```
2017-08-12    75
2017-08-13    55
2017-08-14    55
2017-08-15    85
2017-08-16    85
2017-08-17    80
2017-08-18    95
Freq: D, Name: 午餐 , dtype: int32
             早餐  午餐 晚餐
2017-08-12  65    75   65
2017-08-13  65    55   70
2017-08-14  70    55   80
```

DataFrame 也可透過指定的位置來選擇數據，如 loc、iloc、ix。

E8-1-11.py

```
print(df1.loc['20170812'])
print(df1.loc[:,['早餐','晚餐']])

print(df1.iloc[3,1])
print(df1.iloc[3:5,1:3])

print(df1[df1.午餐>80])
```

執行結果：

```
早餐  65
午餐  75
晚餐  65
Name: 2017-08-12 00:00:00, dtype: int32
            早餐 晚餐
2017-08-12  65  65
2017-08-13  65  70
2017-08-14  70  80
2017-08-15  70  95
2017-08-16  65  60
2017-08-17  60  60
2017-08-18  95  50
85
            午餐 晚餐
2017-08-15  85  95
2017-08-16  85  60
            早餐 午餐 晚餐
2017-08-15  70  85  95
2017-08-16  65  85  60
2017-08-18  95  95  50
```

以下示範另外一種建立 DataFrame 的方式，是由字典開始：

E8-1-12.py

```
import pandas as pd
import numpy as np
# 生成方式二，字典方式
df2 = pd.DataFrame({
'小數':pd.Series (1,index=list(range(4)),dtype='float32'),
'整數':np.array([3] * 4,dtype='int32'),
'時間':pd.Timestamp('20170812'),
'類別資料':pd.Categorical(['test','train','test','train']),})

#dtype 指定資料格式
```

想一想：字典的 key 代表甚麼？

接下來介紹 DataFrame 常用的三個基本的屬性 dtypes、index、describe，可藉由這三個屬性了解 DataFrame 的基本資料與狀態。

1. dtypes 查看資料型態。
2. index 查看資料集的索引。
3. describe() 查看數字資料描述。

E8-1-12.py

```
#DataFrame 的屬性
print(df2)
print(df2.dtypes)
print(df2.index)
print()
print(df2.columns), print(df2.values)
print()
print(df2.describe())
```

執行結果：

```
    小數 整數        時間     類別資料
0  1.0    3 2017-08-12    test
1  1.0    3 2017-08-12   train
2  1.0    3 2017-08-12    test
3  1.0    3 2017-08-12   train
小數       float32
整數        int32
時間   datetime64[ns]
類別資料     category
dtype: object
Int64Index([0, 1, 2, 3], dtype='int64')

Index([' 小數', ' 整數','時間','類別資料 '], dtype='object')
[[1.0 3 Timestamp('2017-08-12 00:00:00') 'test']
 [1.0 3 Timestamp('2017-08-12 00:00:00') 'train']
 [1.0 3 Timestamp('2017-08-12 00:00:00') 'test']
 [1.0 3 Timestamp('2017-08-12 00:00:00') 'train']]

       小數  整數
count  4.0   4.0
mean   1.0   3.0
std    0.0   0.0
min    1.0   3.0
25%    1.0   3.0
50%    1.0   3.0
75%    1.0   3.0
max    1.0   3.0
```

在我們建立完 DataFrame 並且查詢了基本的資料型態後，我們亦可以針對 DataFrame 進行資料的轉置以及排序資料。

E8-1-12.py

```
19  print(df2.T)
20  print(df2.sort_index(axis=1, ascending=False))
21  print(df2.sort_values(by='類別資料'))
```

執行結果：

```
0          ...                                3
小數                   1          ...                        1
整數          3          ...                           3
時間   2017-08-12 00:00:00       ...       2017-08-1200:00:00
類別資料      test          ...                       train

[4 rows x 4 columns]
  類別資料     時間 整數  小數
0   test 2017-08-12   3  1.0
1  train 2017-08-12   3  1.0
2   test 2017-08-12   3  1.0
3  train 2017-08-12   3  1.0
  小數 整數     時間  類別資料
0  1.0   3 2017-08-12   test
2  1.0   3 2017-08-12   test
1  1.0   3 2017-08-12  train
3  1.0   3 2017-08-12  train
```

而對於已建立好的 DataFrame，我們可以進一步新增輸入或是修改現有的值，這時候需要利用 loc、iloc 來進行修改，判斷的方式亦同。

E8-1-13.py

```
import numpy as np
import pandas as pd
eat = np.random.randint(10,size=(7,3))*5+50
dates = pd.date_range('20170812',periods=7)
df1 = pd.DataFrame(eat, index=dates, columns=['早餐','午餐','晚餐']
df1.iloc[2,2] = 95
df1.loc['20170818','晚餐'] = 60
df1.晚餐[df1.早餐>80] = 40
df1.loc['20170814','午餐'] = np.nan
print(df1)
```

執行結果：

```
            早餐  午餐 晚餐
2017-08-12  95  65.0  40
2017-08-13  70  60.0  95
2017-08-14  55   NaN  95
2017-08-15  85  75.0  40
2017-08-16  75  80.0  75
2017-08-17  80  85.0  75
2017-08-18  70  90.0  60
```

用 Pandas 來讀取或儲存檔案也是非常方便的，可以將檔案的內容存取成 DataFrame，亦可以非常方便的將處理結果儲存成新的 csv 檔。

E8-1-14.py

```
import pandas as pd
data = pd.read_csv('XXX.csv')
print(data)
data.to_csv('student.csv')
```

說明：針對不同的格式檔案的讀取與儲存，請參照 http://pandas.pydata.org/pandas-docs/stable/io.html

當有來自不同資料來源的時候，可以使用 pd.concat() 橫向或縱向合併兩個 DataFrame。

E8-1-15.py

```
import pandas as pd
import numpy as np

# 定義資料集
df1 = pd.DataFrame(np.ones((3,4))*0,
      columns=['a','b','c','d'])
df2 = pd.DataFrame(np.ones((3,4))*1,
      columns=['a','b','c','d'])
df3 = pd.DataFrame(np.ones((3,4))*2,
      columns=['a','b','c','d'])

#concat 縱向合併
res = pd.concat([df1, df2, df3], axis=0, ignore_index=True)
print(res)
```

	a	b	c	d
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0
3	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0
6	2.0	2.0	2.0	2.0
7	2.0	2.0	2.0	2.0
8	2.0	2.0	2.0	2.0

圖 8-8 使用 concat() 合併 DataFrame

使用 pd.concat() 合併資料時，可以選擇設定 axis=0 為直向合併，ignore_index=True 可以忽略合併時舊有的索引值欄位，而改採用自動產生的索引值。而 concat 的 join 屬性有兩種模式：inner 交集以及預設的 outer 聯集。

E8-1-16.py

```
import pandas as pd
import numpy as np

df1 = pd.DataFrame(np.ones((3,4))*0,
      columns=['a','b','c','d'], index=[1,2,3])
df2 = pd.DataFrame(np.ones((3,4))*1,
      columns=['b','c','d','e'], index=[2,3,4])
res = pd.concat([df1, df2], axis=0, join='outer',
      ignore_index=True )
print(res)
```

	a	b	c	d	e
0	0.0	0.0	0.0	0.0	NaN
1	0.0	0.0	0.0	0.0	NaN
2	0.0	0.0	0.0	0.0	NaN
3	NaN	1.0	1.0	1.0	1.0
4	NaN	1.0	1.0	1.0	1.0
5	NaN	1.0	1.0	1.0	1.0

圖 8-9 使用 concat() 合併 Dataframe

合併兩個 DataFrame 時，如果需要針對特定的索引合併，這時候就需要使用 merge() 進行合併，可使用 on= 指定要索引的欄位，並且使用 how= 指定 inner、outer、right 或 left 模式。

E8-1-17.py

```
import pandas as pd

left = pd.DataFrame(
{'key': ['K0', 'K1', 'K2', 'K3'],
 'A': ['A0', 'A1', 'A2', 'A3'],
 'B': ['B0', 'B1', 'B2', 'B3']})

right = pd.DataFrame(
{'key': ['K1', 'K2', 'K3', 'K4'],
'C': ['C0', 'C1', 'C2', 'C3'],
'D': ['D0', 'D1', 'D2', 'D3']})

res = pd.merge(left, right, on='key')
print (res)
```

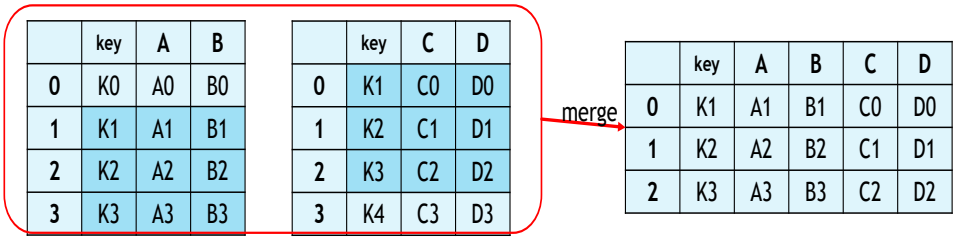


圖 8-10 使用 merge() 合併 Dataframe

E8-1-18.py：

```
import pandas as pd

# 定義資料集並列印出
left = pd.DataFrame(
{'key1': ['K0', 'K1', 'K1', 'K2'],
 'key2': ['K0', 'K1', 'K0', 'K1'],
 'A': ['A0', 'A1', 'A2', 'A3'],
 'B': ['B0', 'B1', 'B2', 'B3']})

right = pd.DataFrame(
{'key1': ['K0', 'K1', 'K1', 'K2'],
  'key2': ['K0', 'K0', 'K0', 'K0'],
  'C': ['C0', 'C1', 'C2', 'C3'],
  'D': ['D0', 'D1', 'D2', 'D3']})

res = pd.merge(left, right, on=['key1', 'key2'], how='inner')
print (res)
```

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

圖 8-11 使用 merge() 合併 Dataframe

小結

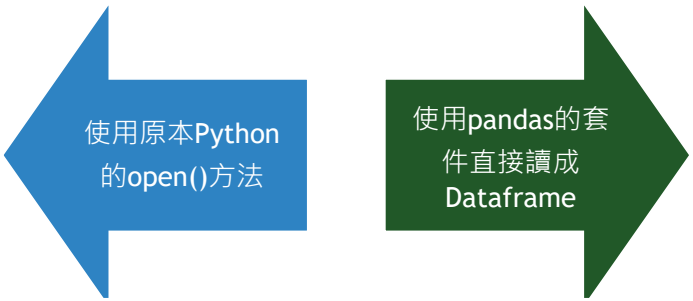
透過本小節的內容，我們已經可以熟悉 Python 中有關科學運算裡的兩個重要套件：NumPy 與 pandas。我們可以透過 NumPy 的 array 處理一維陣列以及多維矩陣運算；亦可利用 pandas 的 DataFrame 處理結構化具列索引與欄標籤的二維資料集，並且進一步針對資料表進行常見的新增、刪除、修改、合併等動作，以 Python 有效率地處理資料相關問題。

8-2 資料預處理實作

在處理大量資料時，首先的問題是大量資料怎麼進入程式的陣列、Series 或 DataFrame ？最常見的是從資料庫讀取資料，或者透過 CSV 等檔案匯入資料，也可以使用網路爬蟲到特定網站抓取需要的資料。抓取外部資料的來源很多樣，每一種方法都需要有相當篇幅的介紹，這邊我們先介紹最簡單的方法：檔案的匯入與匯出。在取得資料後，請先忍耐忍耐，還沒辦法立刻進行資料分析喔，因為仔細檢查外部資料時很容易發生各種問題，例如不同來源的資料格式不同、資料欄位有遺漏值或重複值，有時候我們還需要先將相關的資料合併，或者是進行交集與聯集的處理等。這一節我們就來看看資料在開始分析之前常見的各種前處理工作。

檔案匯入匯出

這裡會介紹兩種讀檔案的方式，可以用 open() 的方法讀取文字檔，也可以使用 pandas 的套件直接從檔案讀取成 DataFrame。



首先介紹第一種檔案開啟的方式，是使用 Python 的 open() 打開檔案。指令格式是 f = open(' 檔案 ', ' 模式 ')，其中模式有三種情形：

1. r 讀取 (檔案需存在)
2. w 新建檔案寫入 (檔案可不存在，若存在則清空)
3. a 資料附加到舊檔案後面 (游標指在 EOF)

在檔案開啟之後，可使用 read() 針對內容進行一次全部讀取。除此之外還有其他的讀取模式：

1. f.read(size)

   size 可以指定要讀進來的字串長度。

2. f.readline()

   讀取一行。

3. f.readlines()

   讀取成一個列表，每一行為列表中的一項。

在檔案寫入的部分，一樣是使用 open() 打開，並且使用 write() 寫入：

- f = open('檔案', 'a')
- f.write('要添加的內容….')
- f.close()

第二種常用的方式是使用 pandas 的套件直接讀成 DataFrame，如果你希望資料讀入程式後，資料以 DataFrame 的形式進行後續的分析處理，那麼建議你以這裡介紹的方式匯入與匯出資料，程式如下：

- import pandas as pd
- df = pd.read_csv('開啟的檔名.csv')
- df.to_csv('要存檔的檔名.csv')

中文要注意轉碼問題，有可能要設定編碼，如：

```
df = pd.read_csv('檔名.csv', encoding='big5')
```

E8-2-1.py：

```
import pandas as pd
df1 = pd.read_('chap8.csv',encoding='big5')
print(df1[:5])
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-12 讀取 csv 檔

使用第二種方式直接讀成 DataFrame，如果要讀取的檔案是其他格式如 excel、sql、sas 等不同格式，可參考下表的方式與指令。

表 8-1: 讀取檔案格式

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

處理遺漏值

在我們讀取資料成 DataFrame 之後，我們可以先判斷是否有缺失數據，下列程式示範如何判斷是否有遺漏值，以及遺漏值的分布情形。

首先，我們可以用 .isnull() 去判斷 DataFrame 是否有遺漏值，有遺

漏值的資料會回傳 False。

E8-2-1.py：

```
# 判斷是否有缺失數據 NaN, 為 True 表示缺失數據：
print(df1.isnull())
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-13 csv 檔原內容

	ID	dept	gender	test01	test02	test03
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	True	False
3	False	False	False	True	False	False
4	False	False	False	False	False	False

圖 8-14 判斷是否有缺失數據

再者，我們可以使用 sum() 進行加總，這樣可以呈現不同欄位的遺漏值總數。

E8-2-1.py：

```
print(df1.isnull().sum())
```

	ID	dept	gender	test01	test02	test03
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	True	False
3	False	False	False	True	False	False
4	False	False	False	False	False	False

```
ID          0
dept        0
gender      0
test01      1
test02      1
test03      0
dtype: int64
```

圖 8-15 顯示遺漏值分布情形

當我們遇到遺漏值的時候，我們會有很多種處理的方法，例如針對遺漏值的欄位進行填值或刪除，以下我們針對不同處理方法一一介紹。

- 處理方法一：填入指定值

在 DataFrame 中使用 .fillna()，針對遺漏的欄位直接填補指定的值。

E8-2-1.py：

```
06  df2 = df1.fillna(value=0)
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

			gender	test01	test02	test03
			m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	0.0	85
3	4	牙醫學系(106)	m	0.0	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-16 填入指定值

- 處理方法二：刪除遺漏值

可使用 .dropna() 將遺漏欄位的相關筆數直接刪除。其中還可定義條件或針對特別的欄位刪除。

1. 將 NaN 值刪除

E8-2-1.py：

```
07  df3 = df1.dropna()
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

ID=3,4被刪除了

			gender	test01	test02	test03
			m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-17 將 NaN 值刪除

2. 參數

E8-2-1.py：

```
# 其他用法
df1.dropna(subset=['test02'])  # 針對特定欄位名稱
df1.dropna(axis=0)  #0: 刪除有 NaN 值的列；1: 刪除有 NaN 值的欄
df1.dropna(how='any')  #'any': 存在 NaN 就 drop 掉；
                       #'all': 全是 NaN 才 drop
df1.dropna(thresh=3)  # 至少要有 thresh 個非 NaN 值
# 其他用法
df3 = df1.dropna(subset=['test02'],axis=0,how='any')
# 針對 test02 列存在 NaN 就 drop 掉
```

	ID	dept	gender	test01	test02	test03
0	1	牙醫學系(104)	m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	3	牙醫學系(106)	m	95.0	NaN	85
3	4	牙醫學系(106)	m	NaN	95.0	90
4	5	牙體技術學系(104)	f	95.0	85.0	80

ID=3被刪除了

			gender	test01	test02	test03
			m	95.0	85.0	95
1	2	牙醫學系(104)	m	95.0	95.0	95
2	4	牙醫學系(106)	m	NaN	95.0	90
3	5	牙體技術學系(104)	f	95.0	85.0	80

圖 8-18 將 NaN 值刪除

- 處理方法三：使用其他值進行插補

可使用其他具有代表性的數值如平均值、中位數等針對遺漏欄位進行插補。下面範例則是使用 sci-kit learn 的套件進行插補，插捕時可選擇平均值 mean, 中位數 median 或眾數 most_frequent。

E8-2-1.py：

```
from sklearn.preprocessing import Imputer
# strategy='mean', 'median', 'most_frequent'
imr = Imputer(missing_values='NaN', strategy=mean, axis=0)
imr = imr.fit(df1)
imputed_data = imr.transform(df1.values)
print(imputed_data)
```

	ID	test01	test02	test03
0	1	95.0	85.0	95
1	2	95.0	95.0	95
2	3	95.0	NaN	85
3	4	NaN	95.0	90
4	5	95.0	85.0	80

	ID	test01	test02	test03
0	1	95.0	85.0	95
1	2	95.0	95.0	95
2	3	95.0	90.0	85
3	4	95.0	95.0	90
4	5	95.0	85.0	80

圖 8-19 使用其他值補值

移除重複值

在我們讀取資料成 DataFrame 之後，除了處理遺漏值的問題，常常我們也會遇到重複的資料這樣的問題。當我們有遇到重複值的問題，我們可以用 duplicated() 去判斷 DataFrame 是否有重複值，有重複值的資料會回傳 True。

而通常針對重複值的處理，一般我們會選擇把重複的值刪除，而刪除的方式也是非常簡單，還可以針對特定欄位進行重複值的比對進行刪除，亦可加上欄位名稱與保留順序。例如：df.drop_duplicates(['col1'], keep='last') 即是針對 col1 這個欄位進行重複值的判斷，當有重複值的時候，我們用 keep='last' 只保留最後一筆重複的資料。

```
01  df.drop_duplicates()
```

E8-2-2.py

```
01  import pandas as pd
02  df = pd.DataFrame(
03       {'col1':['A','A','A','A','B','B','C','C'],
04       'col2':[1,2,2,3,3,4,5,5]})
05  df_d = df.drop_duplicates(['col1'], keep='last')
06  # 指定欄位 col1 重複就移除，保留最後一個
07  print(df_d)
```

	col1	col2
0	A	1
1	A	2
2	A	2
3	A	3
4	B	3
5	B	4
6	C	5
7	C	5

	col1	col2
3	A	3
5	B	4
7	C	5

圖 8-20 移除重複值

資料對應 \ 取代 \ 分箱

最後再補充一下，我們進行資料處理的時候，可能需要針對部分的欄位資料批次的處理，這時候可能就會需要用到資料的對應或取代等處理。

我們先說明資料的對應，在進行資料對應的時候，必須新建一個字典，然後用字典裡的 value 取代原本的 key (原本資料表的值)。在建立字典之後，可以在特定欄位用 map() 執行。例如：下面的例子就是將性別欄位 (女、男) 變成 (0、1)。

E8-2-3.py

```
import pandas as pd
df = pd.DataFrame({'gender':['male','male','female',
                             'male','female','female']})

gender_to_boolean = {'female':0,'male':1} # 字典
df['code'] = df['gender'].map(gender_to_boolean)
print(df)
```

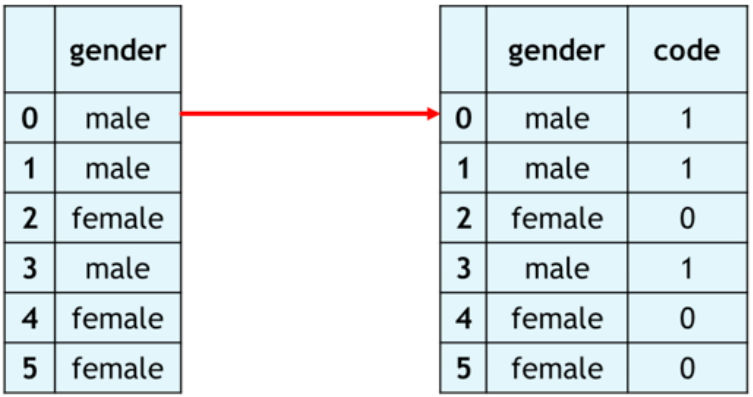


圖 8-21 資料對應

跟對應很相似的另外一種方法是取代。DataFrame 取代的方法是 df.replace(原本的值 , 新的值)。下面的例子就是針對 col2 欄位，用數字 0 來取代字串 '-' 。

```
E8-2-4.py
import pandas as pd
df = pd.DataFrame({
     'col1':['c01','c02','c03','c04','c05'],
     'col2':[65,'NULL','-','-',78],
     'col3':[321,34,'NULL','-',34]})
df['col2'] = df['col2'].replace('-',0)
print(df)
```

	col1	col2	col3
0	c01	65	321
1	c02	NULL	34
2	c03	-	NULL
3	c04	-	-
4	c05	78	34

	col1	col2	col3
0	c01	65	321
1	c02	NULL	34
2	c03	0	NULL
3	c04	0	-
4	c05	78	34

圖 8-22 資料取代

df.replace() 取代的方式也可以用字典，然後用字典裡的 value 取代原本的 key (原本資料表的值)。

```
df.replace({'NULL':0 , '-':-1})
```

另外一個在處理資料常會用到的是分箱，使用分箱的方法可以使連續的類別數值資料變成類別資料。接下來示範如何將 0 到 100 分的分數轉換成 A、B、C、D 的等級制。

一開始我們建立兩個列表，第一個 bins (分箱的間隔點列表) 跟第二個 labels (各區間對應的標籤)，然後使用 .cut() 方法進行分箱。

E8-2-5.py

```
import pandas as pd
df = pd.DataFrame({
     'id':['John','Mary','Tom','Nick','Alice'],
     'score':[90,59,68,77,60]})
bins = [0, 60, 70, 80, 90, 100]
labels = ['E','D','C','B','A']
df['label'] = pd.cut(df['score'],bins, right=False, labels=labels)
print(df)
```

	id	score
0	John	90
1	Mary	59
2	Tom	68
3	Nick	77
4	Alice	60

	id	score	label
0	John	90	A
1	Mary	59	E
2	Tom	68	D
3	Nick	77	C
4	Alice	60	D

圖 8-23 資料分箱

小結

本小節示範了從檔案匯入成 Python 可讀取的格式、遺漏值和重複值處理、資料表聯集等實際資料實作，這些都是建立統計學或機器學習模型時非常重要的前置作業。趕快拿起手邊的資料一起動手做做看吧！