Table of Contents

第18章 分析演算法-Regression 迴歸分析

18.1 資料準備

在介紹演算法之前，需要要先了解一般的數據分析的原理和需要準備的資料，要準備的

資料第一件事情要因果關係，在學術文件中的定義是依照自變數引起的依變數的關係，要留意的是因和果的變化一定要有彼此的連動關係。而大數據界所用的數據內容一定要以下兩種欄位的因果資料：

- 特徵 Features：因，在統計學稱之文自變數 Independent variable
- 標籤答案 Label ：果，在統計學稱為依變數 dependent vbariable

例如：天氣的濕度值（因，特徵）增加時，下雨的機率（果，標籤答案）就會提高。

而大數據需要有很多剛剛提到的因果關係的資料(Features 和 Label)並且大量就稱為數據集 Dataset，其資料量最少 100 個以上，能到數百或千萬的資料量，所求出的結果會更好，而這樣的數據集，要準備的二個的數據集，兩個數據集的欄位格式都是要一模一樣，並且需要用人工審查其內容的正確性，這兩大類的數據集分別為：

- 訓練用數據集 Training Dataset：交給電腦透過特定的演算法，來找出特癥 Features 和標籤答案 Label 其中之間的關係，有的時候比較複雜收據他的特徵值多個。
- 測試用數據集 Testing Dataset： 測試用，用來驗證演算法所求出的結果其正確率為多少。

其資料的筆數大小，通常是 80:20 的比率。接下來就是依照資料的內容分布形式，來挑選合適的演算法了。而大數據分析最重要的就是數據的蒐集，如果收集的數據是錯誤的，不管有什麼樣的演算法都找不到的答案喔！

舉的實際的案例：預測你家附近今天是否會下雨，就只要把過去同一個地點的濕度（特徵 Features）和結果是否下雨（標籤答案 Label），的資料記錄下來並用人工的方法一筆一筆的確認是否正確，收集到 100 筆，然後撥給訓練用數據集 Training Dataset 80 筆， 剩下 20 筆的給測試用數據集 Testing Dataset，這樣就完成大數據得資料收集，接下來就能透過演算法來分析依照同一個地點，依照手上的現在或未來的濕度，來預測是否會下雨的機率。

18.2 機器學習的資料準備

在機器學習之中，必須事先收集正確的資料，並提供答案，舉個例子來說：

案例：如何區檸檬和柳丁？

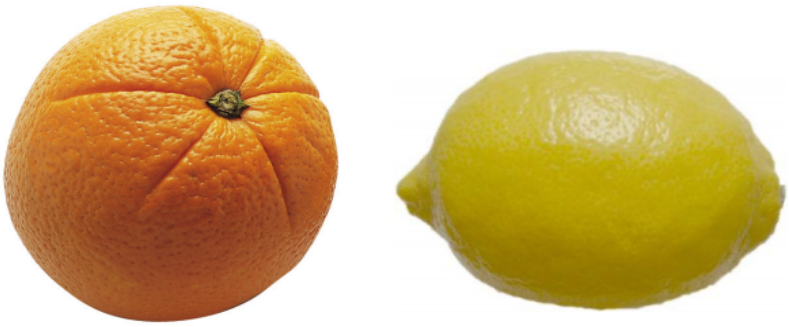

圖 1 如何區檸檬和柳丁

Feature 特癥值:

我們可以量測他的相關的資訊（專業用詞是 Feature 特癥值）：如：顏色、甜度、酸度、體積、重量、長度、寬度 等等……

1. 但會發現體積和重量因該不是好的特癥值，因為兩者太過相近。
2. 甜度和酸度雖然可以找出區別，但會破壞商品和成品的完整性。
3. 可以用長度、寬度 外型是否為趨近為圓型， 顏色來說檸檬偏黃色，而柳丁偏橘色。

所以特癥值的挑選，就會影響到結果。

Label 答案：

這個範例的來說，就是「檸檬」和「柳丁」，通常都會用一個數字來代表，如 1 為柳丁，2 為檸檬。

以實際的圖表的方法來看，就把把檸檬和柳丁的長度、寬度用圖表的方法化出來，會看到這特癥值的位置區分，而我們要做的目標就是畫出圖中的綠線的位置，用的方法叫做「演算法」，當然要盡可能的將所有的資料，可以透過這一條綠線做區分，而畫的好壞叫做「機器學習 Machine learning」的過程，而改善準確率的方法叫「數據挖掘 data mining」，而完成之後，就能用這一條綠線來當成判斷點，用來做為未來的新水果的判斷，也就是「人工智慧 Artificial intelligence」。

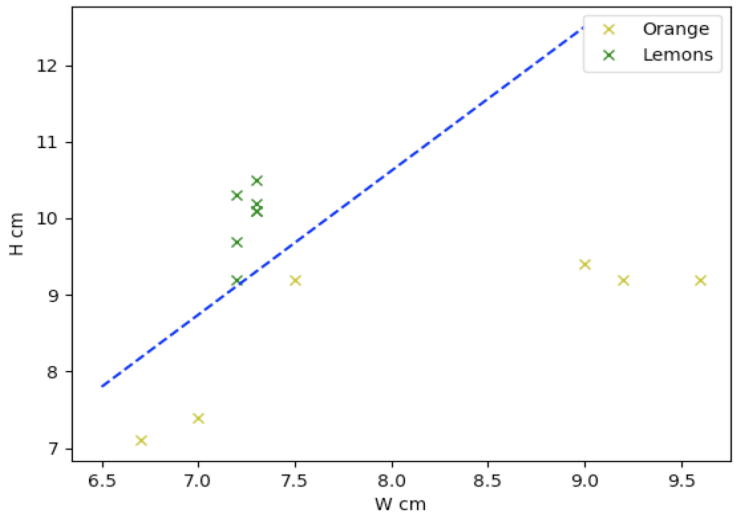

圖 2 檸檬和柳丁長度和寬度的關係圖和本程式執行結果

接下來還有兩個專有名詞:

Training 訓練和 Testing 測試，繼續以剛剛的檸檬和柳丁為例子，其實他們都是先用相同的資料，其內容都一模一樣，通常都是用同一個檔案，然後按照 7:3 的比例來做 Training 訓練和 Testing 測試的資料，如下圖所示：

- A 欄位是種類，1 是柳丁，2 是檸檬
- B 欄位是寬度 width
- C 欄位是長度 height

	A	B	C
1	1	9	9.4
2	1	9.2	9.2
3	1	9.6	9.2
4	1	7.5	9.2
5	1	6.7	7.1
6	1	7	7.4
7	1	7.1	7.5
8	1	7.8	8
9	1	7.2	7
10	1	7.5	8.1
11	1	7.6	7.8
12	1	7.1	7.9
13	1	7.1	7.6
14	1	7.3	7.3
15	1	7.2	7.8
16	1	6.8	7.4
17	1	7.1	7.5
18	1	7.6	8.2
19	1	7.2	7.2
20	2	7.2	10.3
21	2	7.3	10.5
22	2	7.2	9.2
23	2	7.3	10.2
24	2	7.3	9.7
25	2	7.3	10.1
26	2	5.8	8.7
27	2	6	8.2
28	2	6	7.5
29	2	5.9	8
30	2	6	8.4
31	2	6.1	8.5
32	2	6.3	7.7
33	2	5.9	8.1
34	2	6.5	8.5
35	2	6.1	8.1

圖 3 檸檬和柳丁長度和寬度的訓練和測試資料

範例程式： plot01.py

```
1.  import matplotlib.pyplot as plt# 繪圖函示庫
2.  # 繪製(黃)色 x 標記
3.  plt.plot([9,9.2,9.6,7.5,6.7,7], [9.4,9.2,9.2,9.2,7.1,7.4 ], 'yx' )
4.  # 繪製(綠)色 x 標記
5.  plt.plot([7.2,7.3,7.2,7.3,7.2,7.3,7.3 ], [10.3,10.5,9.2,10.2,9.7,10.1,10.1 ], 'gx' )
6.  plt.plot([6.5,9.0], [7.8,12.5], 'b--' )                    #繪製黑色虛線
7.  plt.ylabel('H cm')                                         # 設定顯示 Y 文字
8.  plt.xlabel('W cm')                                         # 設定顯示 X 文字
9.  plt.legend(('Orange','Lemons'),loc='upper right')          # 繪製圖表
10. plt.show()                                                 # 繪製圖表
```

18.3 迴歸分析數學介紹

迴歸分析（英語：Regression Analysis）是一種統計學上解析數據的方法，目的在於了解兩個或多個變數之間是否相關，並建立數學模型用來以便觀察特定變數來預測研究者感興趣的變數。更具體的來說，迴歸分析可以協助瞭解只有一個自變量，因變換的變化量。

迴歸分析根據自變數之多寡，可分為以下二 種：

1. 簡單迴歸分析(simple regression analysis)：用一個自變數 來解釋一個依變數之迴歸分析。
2. 複迴歸分析(multiple regression analysis)：用二個或二個 以上自變數來解釋一個依變數之迴歸分析。

迴歸模型亦可視其函數之型態區分為線性(linear) 與非線性(nonlinear)兩種。

- Y=a+bX 為 一線性模式
- Y=a+ X^b 則為非線性模式。

給一個隨機樣本 $(Y_i, X_{i1}, ..., X_{ip}), i=1,...,n$ ，一個線性回歸模型假設回歸子 Y_i 和回歸量 $X_{i1}, ..., X_{ip}$ 之間的關係是除了 X 的影響以外，還有其他的變數存在。我們加入一個誤差項 ε_i （也是一個隨機變量）來捕獲除了 $X_{i1}, ..., X_{ip}$ 之外任何對 Y_i 的影響。所以一個多變量線性回歸模型表示為以下的形式：

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta_p X_{ip} + \varepsilon_i$$

i=1,….,n

線性回歸作為條件預期模型的簡單線性回歸，可以表示為：

E(Y_i | X_i = x_i)=α+β x_i

18.4 迴歸分析繪圖

為了瞭解取得資料和數學公式的關係，將資料透過程式的方法繪製出來是最好的表現方法，為了達到這目的，首先請安裝繪圖函示庫，請透過以下指令安裝

pytho 2.x 請使用

```
$ pip install matplotlib
```

pytho 3.x 請使用

```
$ pip3 install matplotlib
```

matplotlib 是非常好用的會 Python 圖表繪製的函示庫，以下的程式，是將剛剛求出的答案繪製出來，方便讓各位知道彼此之間的關係。

假設手上的擁有的大數據，油價對民眾交通習慣的影響為例子，看到當油價得價價格，會影響到大眾交通工具的乘坐人數。例如油價為現行的價格 [1,2,3,4]倍數時，大眾交通工具的乘坐人數就會有[0,0.3,0.6,0.9]　的比例增加。首先第一步驟是先把手上的資料，先透過 matplotlib 圖片繪製圖的方式顯示出來。

範例程式 02_plot_dot.py

```
1. import matplotlib.pyplot as plt          # 繪圖函示庫
2. plt.plot([1,2,3,4], [0,0.3,0.6,0.9], 'gx')   # 在 x=1,y=0 四個位置繪製(g)色 x 標記
3. plt.plot([1,2,3,4], [0,0.3,0.6,0.9], 'r--')   #繪製(r)紅色- 標記也就是線（趨勢線）
4. plt.axis([0, 5, 0, 1])                   # 圖表大小範圍，寬度由 0 到 5 ,高度由 0 到 1
5. plt.ylabel('Y')                          # 設定顯示 Y 文字
6. plt.xlabel('X')                          # 設定顯示 X 文字
7. plt.show()                               # 繪製圖表
```

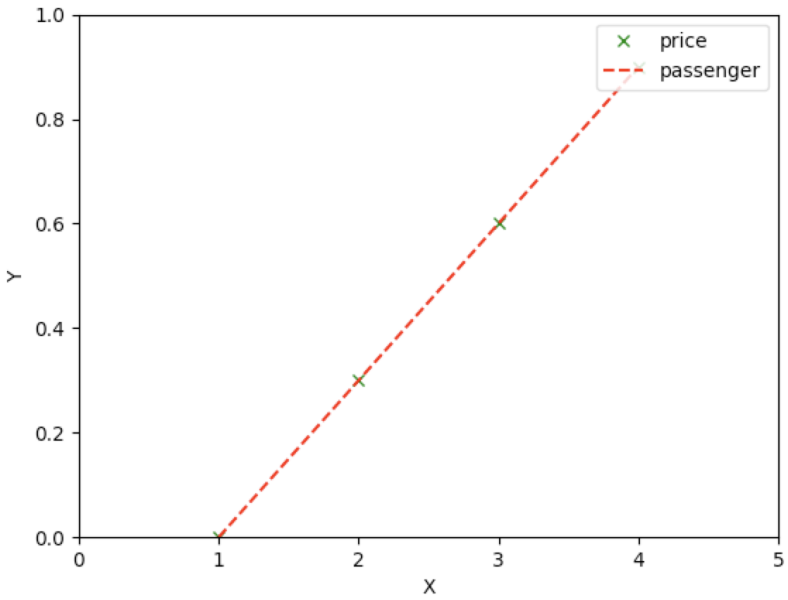

圖 4 執行結果

所以當然符合 Y=0.3*X- 0.3　計算式的情況。

而在剛剛的範例程式中，有畫出一條直線(趨勢線)，那要如何畫出？我們是用

```
plt.plot([1,2,3,4], [0,0.3,0.6,0.9], 'r--')   # 在 x=1,y=0 四個位置繪製(r)紅色- 標記也就是線
```

當然也可以改成

```
plt.plot([1,4], [0,0.9], 'r--')             # 在 x=1,y=0 二個位置繪製(r)紅色- 標記也就是線
```

所化出來的線，也會是一樣的。

18.5 亂數數據

在本章節之中，將會介紹的實際的資料表現型態，正常來說資料絕對不會乖乖的在坐落在線上，一定會小顆的誤差。然後好的回歸分析是要要想盡辦法讓畫出來的線，平均的若在每一個點的最短路徑之中。

透過以下的程式能就會產生出 30 個，連續的資料每一筆資料差為 0.1，也就是說到時候產生出來的資料範圍在[1.0,1.1,1.2,……..,3.7,3.8,3.9]之間。

```
import numpy as np
X = 1+np.arange(30)/10
```

接下來透過亂數，來產生出 30 個在這個簡單迴歸附近的的資料，也就是常態分位數。

```
delta = np.random.uniform(low=-0.1,high=0.1, size=(30,))        # 取亂數
Y=0.3*X- 0.3 + delta                                            #帶入公式
```

取亂數：

numpy.random.uniform(low=0.0, high=1.0, size=(30,))

- low：最小值
- high：最大值
- size: 數量

範例程式 04_plot_dots_not_onTheLine.py

```
1.  import matplotlib.pyplot as plt                              # 繪圖函示庫
2.  import numpy as np
3.  plt.plot([1,2,3,4], [0,0.3,0.6,0.9], 'gx')                   # 綠色  x 的點
4.  plt.plot([1,2,3,4], [0,0.3,0.6,0.9], 'r--')                  # 紅色色  --虛線
5.  X = 1+np.arange(30)/10
6.  delta = np.random.uniform(low=-0.1,high=0.1, size=(30,))     # 取亂數
7.  Y=0.3*X- 0.3  + delta                                        #帶入公式
8.  plt.plot(y1,y2,'bo')                                         # 藍色的圓點
9.  plt.ylabel('Y')                                              # 設定顯示 Y 文字
10. plt.xlabel('X')                                              # 設定顯示 X 文字
11. plt.show()                                                   # 繪製圖表
```

執行結果

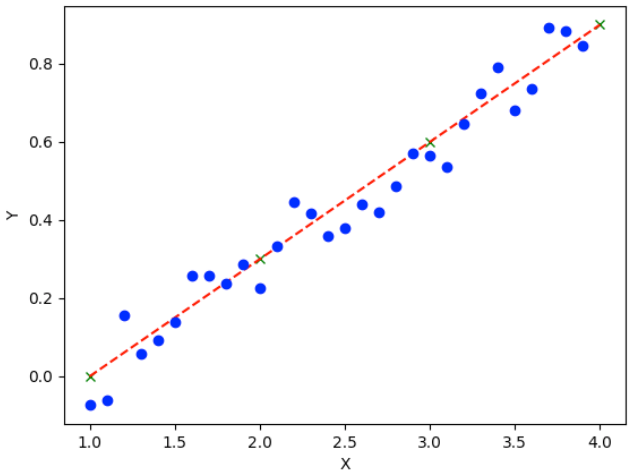

圖 5 執行結果

18.6 殘差 residual

好的迴歸分析結果，取決於利用線性回歸計算出趨勢線的位置和斜率，和實際統計資料之間的誤差，延續上一個程式，本章節將計算實際和誤差的差異，也就是預測的線段，和實際距離的差異，透過取絕對值的方法，把全部相減出來的距離值相加，求得出的總殘差平方和。

並且除與所有的數量，這就是殘差 residual，如果這個答案趨近於零，就代表這個迴歸分析發出來的答案是非常符合實際的資料。

範例程式 05_rsidual.py

```
1.  …                                                    #匯入函數，同上一個範例
2.  #產生亂數
3.  X = 1+np.arange(30)/10                               #30 個矩陣[1,1.1,1.2,1.3,….]
4.  delta = np.random.uniform(low=-0.1,high=0.1, size=(30,))  # 產生 30 個亂數在 0.1~-0.1
5.  Y=0.3*X- 0.3  + delta                                # 產生 30 個點
6.  ….                                                   # 繪圖，同上一個範例
7.  # 計算殘差
8.  sum1=0.0
9.  i=0
10. for X1 in X:                                         #取每一個實際值
```

```
    Y1 = 0.3 * X1 - 0.3
    Y2 = 0.3 * X1 - 0.3 + delta[i]
    sum1=sum1+abs(Y1-Y2)                    #計算相差和累計
    i=i+1
print("殘差",sum1/30)                      # 顯示列印殘差
```

執行結果

殘差值 0.055321144869560496

這個程式的前半段的透過程式產生 30 個亂數點。後半段的部分透過回歸分析的線段，來計算每一個點跟這條線之間的距離，然後把全部之間的距離，用絕對值相加起來全部，就是殘差。

當然各位發現了這一個殘差並不是數字零，所以我們回歸分析之中，最重要的就是想辦法讓這個數字和線段趨近到零。這樣的意思是說找出的條線就是最符合所有的點，也能此線描述的趨勢和走向。

18.7 使用 SciKit 的 linear_model　函數求線性迴歸

SciKit 是一個相當好用的科學函式庫，裡面提供了非常多的演算法，這樣可以幫助非數學、物理、統計學相關科系的程式設計者，很快地進入大數據分析的一個電腦函式庫，程式設計者只要知道在哪個情況下要使用哪一個演算法，只要把數據資料帶入，就能求出結果。

安裝的方法如下：

```
pip install   sklearn
```

本章將把剛剛的資料，透過簡單線性迴歸函式庫，來求得出答案。首先顯把資料透過，收先透過以下的方法，新增線性迴歸計算式類別。

```
from sklearn import linear_model
body_reg = linear_model.LinearRegression()                    # 新增線性迴歸計算式類別
```

送入訓練資料。

```
body_reg.fit(x_values, y_values)
```

預測取得預測結果。

```
y_test_predict= body_reg.predict(x_text)
```

範例程式 05-Regression.py

```
1.  import pandas as pd                                  #匯入 pandas 模型
2.  from sklearn import linear_model                     #匯入線性模型
3.  import matplotlib.pyplot as plt                      #匯入繪圖
4.  # 準備訓練和測試的資料
5.  x_values =pd.DataFrame([0,1,2])
6.  y_values =pd.DataFrame([0,0.3,0.6])
7.  x_text =pd.DataFrame([-1,3,5])
8.
9.  body_reg = linear_model.LinearRegression()           #指定線性迴歸
10. body_reg.fit(x_values, y_values)                     #訓練
11.
12. y_test_predict= body_reg.predict(x_text)             # 預測
13. print(" body_reg.predict(x_text)",y_test_predict)    #列印
14.
15. #顯示圖形
16. plt.scatter(x_values, y_values)                        #畫出原本的資料
17. plt.scatter(x_text, y_test_predict, color='red')       #畫出預測的資料
18. plt.plot(x_text,y_test_predict, color='blue')          # 發出線性迴歸的線
19. plt.show()                                             # 顯示
```

執行結果：

```
body_reg.predict(x_text) [[-0.3]
[ 0.9]
[ 1.5]]
```

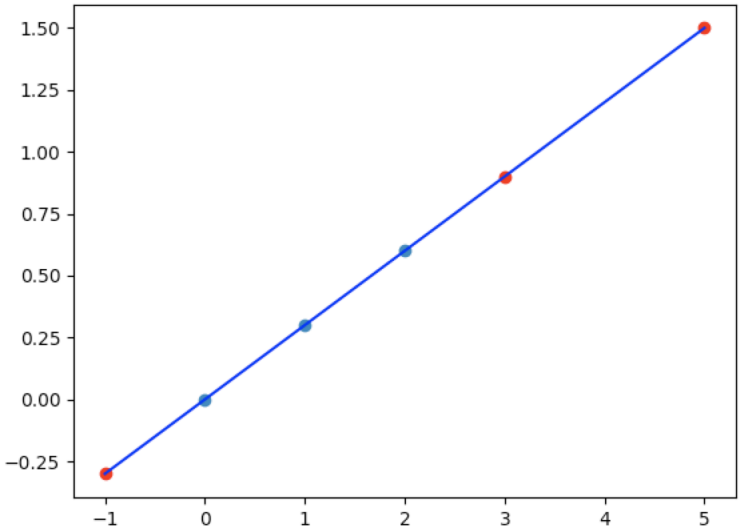

圖 6 執行結果

在這個程式之中，可以看到只要把要預測的數據資料，送入到 body_reg.fit(x_values, y_values)經過訓練之後，就能透過 body_reg.predict(x_text) 找出所預測出的答案，

所以在程式最後，透過圖形的方法把「訓練」「預測」「迴歸分析線段」顯示出來，會發現本程式完全的就是在這一條線上面，這也就是最棒的答案和情況。

18.8 實戰案例-動物大腦和身體的關係

在這個章節將使用真正的醫學資料，來做做迴歸統計。這次的數據中，包含記錄動物的體重和他大腦的重量，我們將透過以下的程式，實際的統計出自然界動物的體重和大腦之間的關係。

範例程式:06_BrainBody.py

```
1. import pandas as pd
2. from sklearn import linear_model                    # 繪圖函示庫
3. import matplotlib.pyplot as plt                     # 繪圖函示庫
4. #資料處理
```

Python

```
5.  dataframe = pd.read_fwf('brain_body.txt')        # 讀取 txt 的數據
6.  x_values = dataframe[['Brain']]                  # 大腦的數據
7.  y_values = dataframe[['Body']]                   # 身體的數據
8.  #模型和訓練
9.  body_reg = linear_model.LinearRegression()       #使用線性迴歸模型
10. body_reg.fit(x_values, y_values)                 #訓練
11. #圖形化
12. plt.scatter(x_values, y_values)
13. plt.plot(x_values, body_reg.predict(x_values))
14. plt.show()
```

執行結果

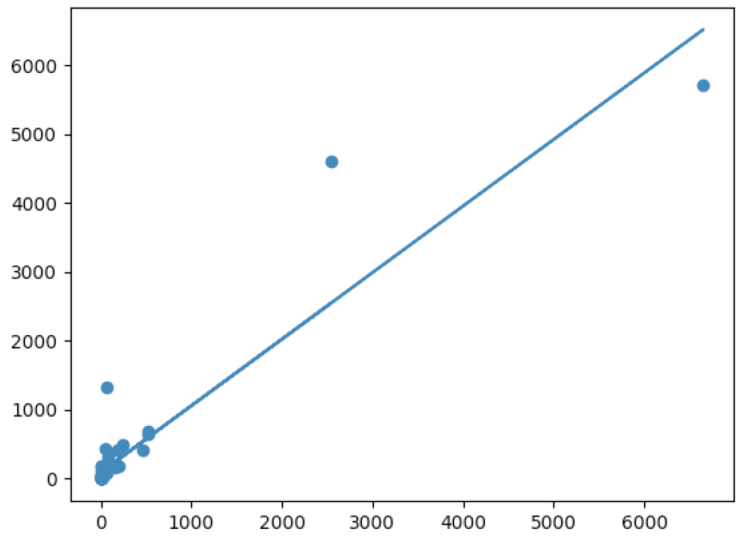

圖 7 執行結果

18.9 實戰案例-糖尿病

在這一個章節，將用另外一個醫學數據範例，來探討回歸分析的在糖尿病上的醫學研究，這個糖尿病數據資料來源是由，scikit-learn 函示庫中還附帶一些開發練習時的數據集，還有很多有趣的一些訓練資料，有機會會在後面的介紹和使用， 請看以下的列表。

- load_boston：波士頓房價數據集。
- load_iris：鳶尾花數據集。
- load_diabetes：糖尿病數據集。
- load_digits：手寫 OCR 數字圖片數據集。
- load_linnerud：linnerud 數據集。
- load_wine：葡萄酒數據集。
- load_breast_cancer：威斯康辛州乳腺癌據集。

本章節將會使用 scikit-learn 糖尿病數據集，這是一個糖尿病的數據集，主要包括 442 筆數據，10 個屬性值，分別是：

- Age(年齡)
- Sex(性別)
- Body mass index(體質指數 BMI)
- Average Blood Pressure(平均血壓)
- S1~S6 血液中各種疾病級數指標

而結果的部分為：

- Target 為一年後患疾病的指標數

18.9.1 繪製出資料

首先將透過以下的程式將資料下載取得，並透過圖形化可以清楚的瞭解這個糖尿病的數據的樣貌，

範例程式　07-diabets.py

```
1. import matplotlib.pyplot as plt                    # 繪圖函示庫
2. import numpy as np                                 #矩陣函示庫
3. from sklearn import datasets, linear_model         #線性迴歸函示庫
4. # 取得糖尿病的數據
5. diabetes = datasets.load_diabetes()                # 取得糖尿病的數據
6. # 取得
7. diabetes_X = diabetes.data[:, np.newaxis, 2]       #只取第三個特徵值 BMI
8. # 切分特徵值 BMI
9. diabetes_X_train = diabetes_X[:-20]                # 切割 0 到最後 20 筆特徵給訓練用
```

```
10. diabetes_X_test = diabetes_X[-20:]                    # 切割最後 20 筆特徵給訓練用
11. # 切分答案
12. diabetes_y_train = diabetes.target[:-20]              # 切割 0 到最後 20 筆答案給訓練用
13. diabetes_y_test = diabetes.target[-20:]               # 切割最後 20 筆答案給訓練用
14. # 繪圖
15. plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')       #畫出黑點
16. plt.show()                                                          #顯示繪圖
```

執行結果

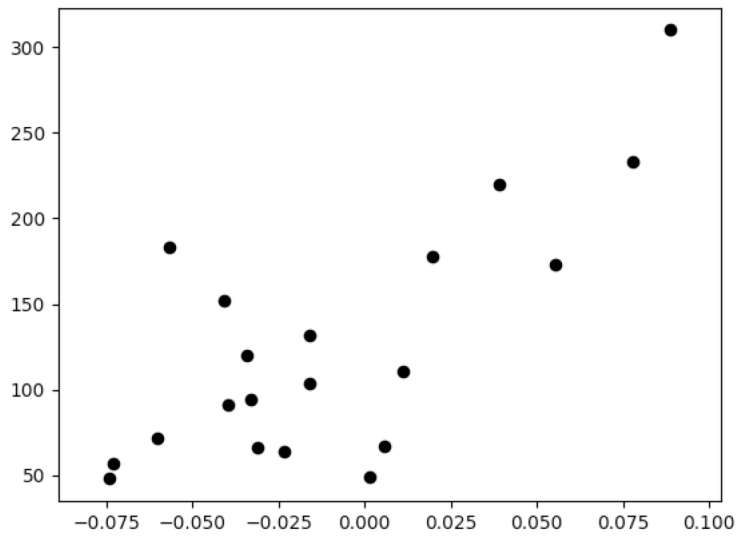

圖 8 執行結果

請留意這裡的 X 軸數據是只用 Body mass index(體質指數 BMI)。

18.9.2 將資料存到 Excel 檔案

這個糖尿病數據的特徵值只有 10 種之多，為了讓各位了解實際數字的樣子和範圍，在本章節將透過 pandas 函式庫，將所取得的數值儲存在 Excel 表之中，這樣的話方便觀看這一個醫學數據的內容。

範例程式 06-RegressionDiabetesLoad.py

```
1. import matplotlib.pyplot as plt                    # 繪圖函示庫
2. import numpy as np                                 # 矩陣函示庫
3. from sklearn import datasets, linear_model         # 線性迴歸函示庫
```

```
# 取得糖尿病的數據
diabetes = datasets.load_diabetes()
print("diabetes.data.shape=",diabetes.data.shape)              # 輸出 (442, 10)
print("dir(diabetes)",dir(diabetes))       # 輸出['DESCR', 'data', 'feature_names', 'target']
print("diabetes.target.shape=",diabetes.target.shape)          # 輸出 (442,)
try:
  print("diabetes.feature_names=",diabetes.feature_names)        # 顯示特徵值名稱
except:
  print("No diabetes.feature_names=")
import xlsxwriter                                              # Excel 函示庫
import pandas as pd                                            # pandas 函示庫
# 轉換資料型態
try:
  df = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
except:
   df = pd.DataFrame(diabetes.data, columns= ['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4',
's5', 's6'])
print(df.head())                                               # 顯示前五筆資料

df.to_csv("diabetes.csv", sep='\t')                            #儲存到 CSV

writer = pd.ExcelWriter('diabetes.xlsx', engine='xlsxwriter') # 儲存到 Excel
df.to_excel(writer, sheet_name='Sheet1')
writer.save()
```

執行結果

A1

	A	B	C	D	E	F	G	H	I	J	K
1		age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
2	0	0.038076	0.05068	0.061696	0.021872	-0.04422	-0.03482	-0.0434	-0.00259	0.019908	-0.01765
3	1	-0.00188	-0.04464	-0.05147	-0.02633	-0.00845	-0.01916	0.074412	-0.03949	-0.06833	-0.0922
4	2	0.085299	0.05068	0.044451	-0.00567	-0.0456	-0.03419	-0.03236	-0.00259	0.002864	-0.02593
5	3	-0.08906	-0.04464	-0.0116	-0.03666	0.012191	0.024991	-0.03604	0.034309	0.022692	-0.00936
6	4	0.005383	-0.04464	-0.03638	0.021872	0.003935	0.015596	0.008142	-0.00259	-0.03199	-0.04664
7	5	-0.0927	-0.04464	-0.0407	-0.01944	-0.06899	-0.07929	0.041277	-0.07639	-0.04118	-0.09635
8	6	-0.04547	0.05068	-0.04716	-0.016	-0.0401	-0.0248	0.000779	-0.03949	-0.06291	-0.03836
9	7	0.063504	0.05068	-0.00189	0.06663	0.09062	0.108914	0.022869	0.017703	-0.03582	0.003064
10	8	0.041708	0.05068	0.061696	-0.0401	-0.01395	0.006202	-0.02867	-0.00259	-0.01496	0.011349
11	9	-0.0709	-0.04464	0.039062	-0.03321	-0.01258	-0.03451	-0.02499	-0.00259	0.067736	-0.0135
12	10	-0.09633	-0.04464	-0.08381	0.008101	-0.10339	-0.09056	-0.01395	-0.07639	-0.06291	-0.03421
13	11	0.027178	0.05068	0.017506	-0.03321	-0.00707	0.045972	-0.06549	0.07121	-0.09643	-0.05907
14	12	0.016281	-0.04464	-0.02884	-0.00911	-0.00432	-0.00977	0.044958	-0.03949	-0.03075	-0.0425
15	13	0.005383	0.05068	-0.00189	0.008101	-0.00432	-0.01572	-0.0029	-0.00259	0.038393	-0.0135
16	14	0.045341	-0.04464	-0.02561	-0.01256	0.017694	-6.1E-05	0.081775	-0.03949	-0.03199	-0.07564
17	15	-0.05274	0.05068	-0.01806	0.080401	0.089244	0.107662	-0.03972	0.108111	0.036056	-0.0425
18	16	-0.00551	-0.04464	0.042296	0.049415	0.024574	-0.02386	0.074412	-0.03949	0.05228	0.027917
19	17	0.070769	0.05068	0.012117	0.056301	0.034206	0.049416	-0.03972	0.034309	0.027368	-0.00108
20	18	-0.03821	-0.04464	-0.01052	-0.03666	-0.03734	-0.01948	-0.02867	-0.00259	-0.01812	-0.01765

圖 9 執行結果，儲存到 Excel

18.9.3 使用迴歸分析找出 BMI 與糖尿病的關係

在章節中將透過線性迴歸性的方法，將 BMI 與結果彼此的關係，找出其關聯性，並且預測出相關的結果。

範例程式 09-LinearRegression-diabetes.py

```
import matplotlib.pyplot as plt                          # 繪圖函示庫
import numpy as np                                       #矩陣函示庫
from sklearn import datasets, linear_model               #線性迴歸函示庫
# 取得糖尿病的數據
diabetes = datasets.load_diabetes()                      # 取得糖尿病的數據
# 取特徵值 BMI
diabetes_X = diabetes.data[:, np.newaxis, 2]             #只取第三個特徵值 BMI
# 切分特徵值 BMI
diabetes_X_train = diabetes_X[:-20]                      # 切割 0 到最後 20 筆特徵給訓練用
diabetes_X_test = diabetes_X[-20:]                       # 切割最後 20 筆特徵給訓練用
# 切分答案
diabetes_y_train = diabetes.target[:-20]                 # 切割 0 到最後 20 筆答案給訓練用
diabetes_y_test = diabetes.target[-20:]                  # 切割最後 20 筆答案給訓練用
# 研究計算
regr = linear_model.LinearRegression()                   # 建立線性迴歸
regr.fit(diabetes_X_train, diabetes_y_train)             #訓練
print('Coefficients: \n', regr.coef_)                    # 係數
# 均方誤差
print("Mean squared error: %.2f"
      % np.mean((regr.predict(diabetes_X_test) - diabetes_y_test) ** 2))
# 顯示方差分數：1 是完美預測
print('Variance score: %.2f' % regr.score(diabetes_X_test, diabetes_y_test))
```

Python

```
23. # 繪圖
24. plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')    #畫出測試黑點
25. plt.plot(diabetes_X_test, regr.predict(diabetes_X_test), color='blue',
26.         linewidth=3)                                              #畫出測試預測的線性迴歸
27. plt.xticks(())
28. plt.yticks(())
29. plt.show()                                                        #顯示
```

執行結果:

Coefficients: [938.23786125]

Mean squared error: 2548.07

Variance score: 0.47

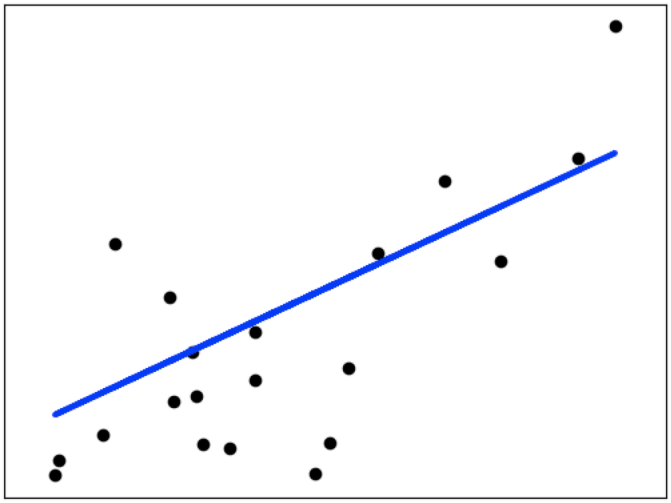

圖 10 執行結果