



機器學習

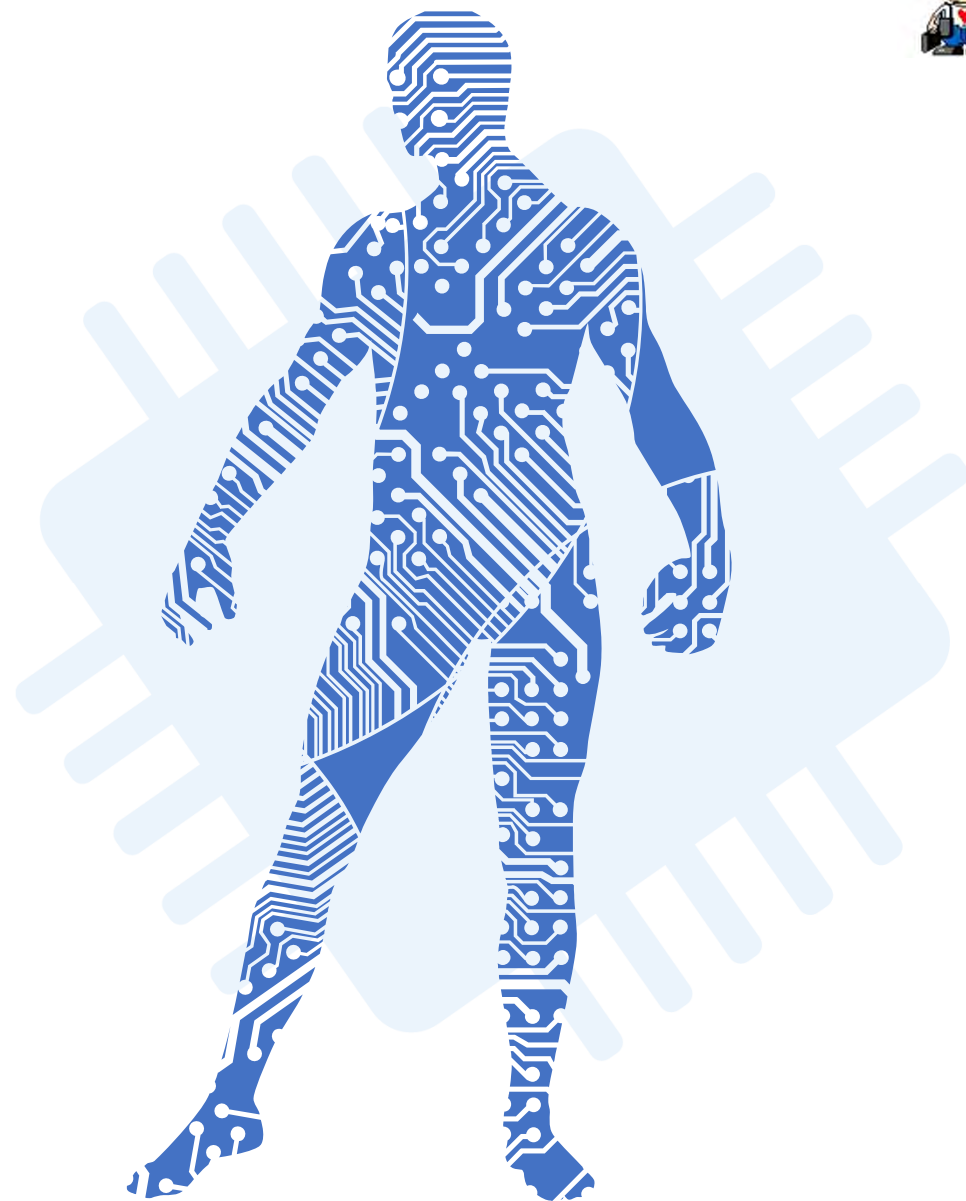
第 5 章 簡單線性迴歸 (Simple Regression)

講師：紀俊男



本章大綱

- 「迴歸」簡介
- 理論說明
- 資料前處理
- 使用「標準函式庫」實作
- 評估模型好壞
- 使用「快樂版」實作
- 將結果視覺化
- 重點整理





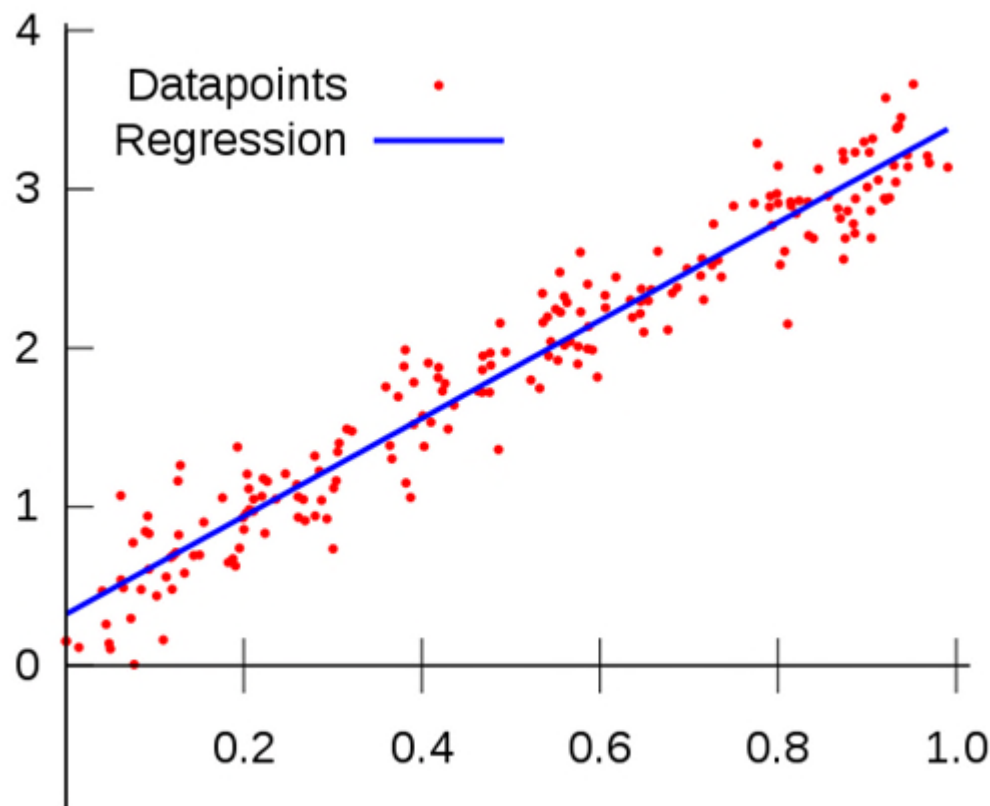
AI

「迴歸」簡介

何謂「迴歸」 (Regression)



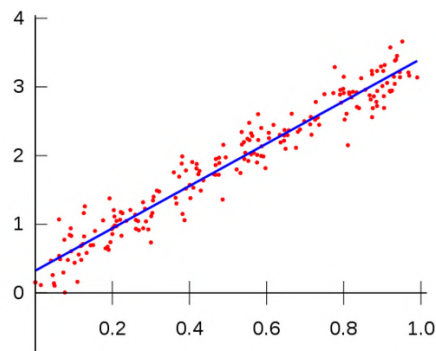
- 利用「**過往資料**」，計算「**近似線型**」，並用於「**預測未來**」的方法



機器學習裡的「迴歸種類」

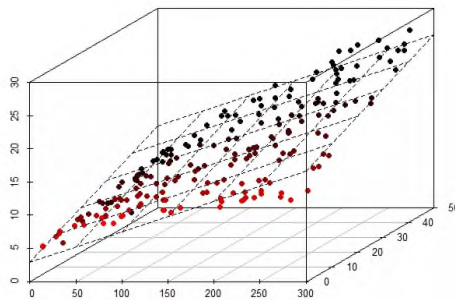


$$f(x) = c + mx$$



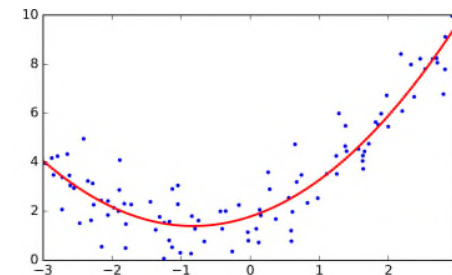
簡單線性迴歸
(Simple Linear Regression)

$$f(x, y, z, \dots) = c_0 + c_1X + \dots c_nZ$$



多元線性迴歸
(Multiple Linear Regression)

$$f(x) = c_0 + c_1X^1 + \dots c_nX^n$$



多項式迴歸
(Polynomial Regression)

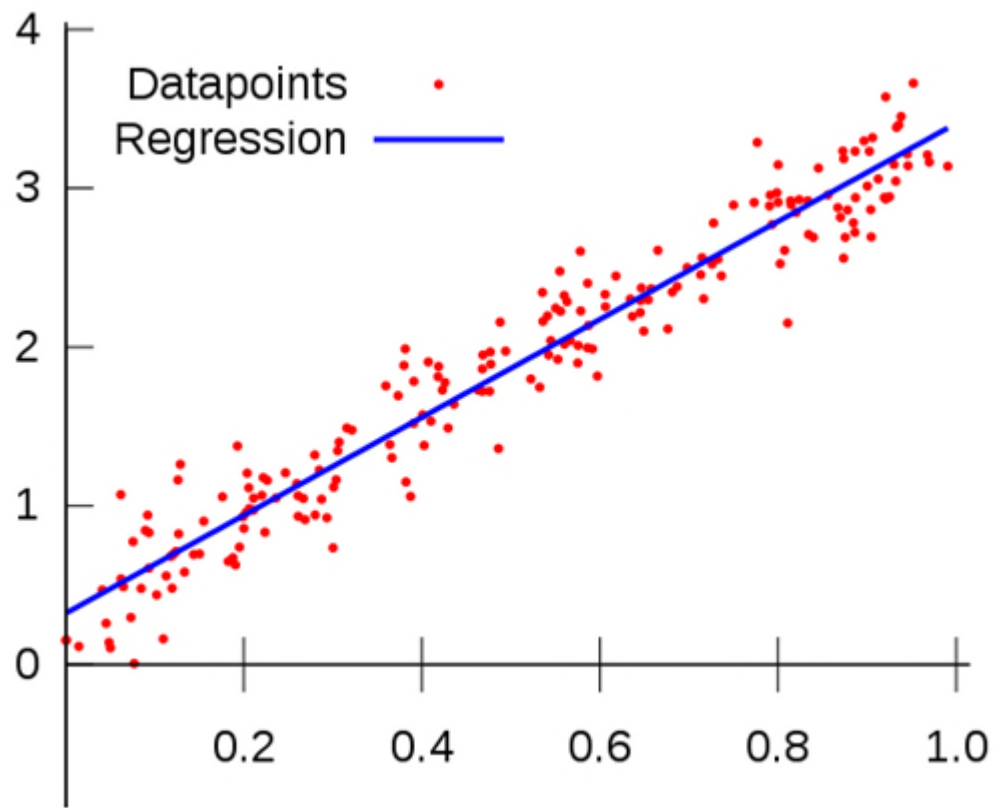


理論説明

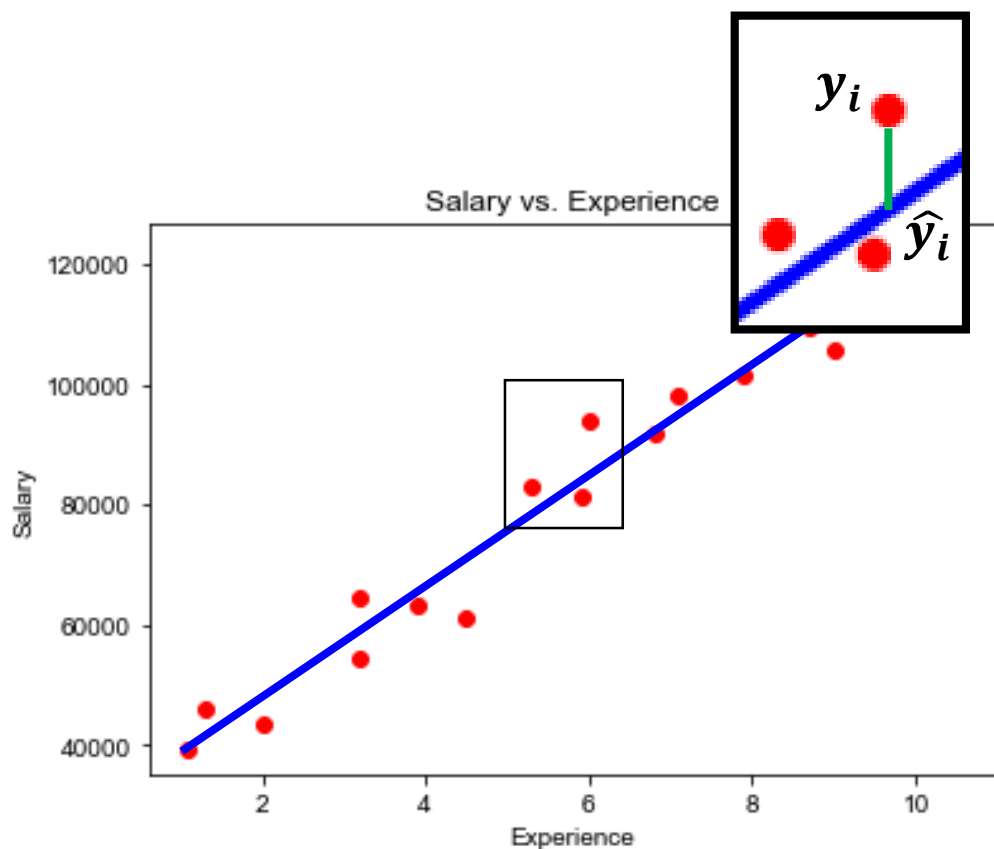
(Simple Linear Regression)

- 1 個連續一次自變數 vs. 1 個連續應變數（一元一次方程式）

$f(x) = c + mx$



「簡單線性迴歸」理論說明



- 樣本點：年資 vs. 薪資

- 迴歸模型

$$y = c + mX$$

x ：年資 c ：截距（畢業生起薪）
 y ：薪資 m ：斜率（薪資增長幅度）

- 模型計算

- 使用最小平方方法（Ordinary Least Squares, OLS）

$$E_i = y_i - \hat{y}_i = y_i - c - mX_i$$

$$E = \sum E_i^2 = \sum (y_i - c - mX_i)^2 \quad \left\{ \begin{array}{l} \frac{\partial}{\partial c} E = 0 \\ \frac{\partial}{\partial m} E = 0 \end{array} \right. \quad \begin{array}{l} \text{求出} \\ c, m \\ \text{極值} \end{array}$$



AI

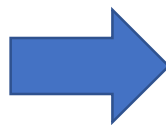
資料前處理



- 依照講師指示，**下載**並**瀏覽**資料集



Salary_Data.csv



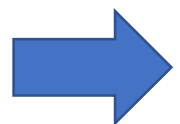
	A	B
1	YearsExperience	Salary
2	1.1	39343
3	1.3	46205
4	1.5	37731
5	2	43525
6	2.2	39891
7	2.9	56642
8	3	60150
9	3.2	54445
10	3.2	64445

目的：給定年資 --> 推算薪資

- 依照講師指示，下載並安裝自製函式庫



HappyML



Machine Learning

- Ch02
- Ch03
- Ch04
- Ch05
 - HappyML

設定工作路徑

C:\Users\俊男\OneDrive\工作\機器學習課程\DemoCodes\Ch05

撰寫程式碼

```
1 # In[] Pre-processing
2 from HappyML import preprocessor as pp
3
4 # Dataset Loading
5 dataset = pp.dataset("Salary_Data.csv")
6
7 # Independent/Dependent Variables Decomposition
8 X, Y = pp.decomposition(dataset, [0], [1])
9
10 # Split Training vs. Testing Set
11 X_train, X_test, Y_train, Y_test = pp.split_train_test(X, Y, train_size=2/3)
12
13 # Feature Scaling (optional)
14 # X_train, X_test = pp.feature_scaling(fit_ary=X_train, transform_arys=(X_train, X_test))
15 # Y_train, Y_test = pp.feature_scaling(fit_ary=Y_train, transform_arys=(Y_train, Y_test))
```

資料前處理流程：

- 1. 載入資料
- 2. 切分自變數、應變數
- 3. 處理缺失資料
(無缺失資料)
- 4. 類別資料數位化
(無類別資料)
- 5. 切分訓練集、測試集
- 6. 特徵縮放
(先註解掉，繪圖時會用到)



- 請依照講師指示，下載「**資料集**」、與「**自製函式庫**」
- 請設定工作路徑，至「**<範例資料夾>/Ch05**」
- 使用講師提供的「**自製函式庫**」，寫好下列程式碼：
- 執行、並用「**變數觀察面板**」，檢查結果是否正確

```
1 from HappyML import preprocessor as pp
2
3 # Dataset Loading
4 dataset = pp.dataset("Salary_Data.csv")
5
6 # Independent/Dependent Variables Decomposition
7 X, Y = pp.decomposition(dataset, [0], [1])
8
9 # Split Training vs. Testing Set
10 X_train, X_test, Y_train, Y_test = pp.split_train_test(X, Y, train_size=2/3)
11
12 # Feature Scaling (optional)
13 X_train, X_test = pp.feature_scaling(fit_ary=X_train, transform_arys=(X_train, X_test))
14 Y_train, Y_test = pp.feature_scaling(fit_ary=Y_train, transform_arys=(Y_train, Y_test))
```





使用「標準函式庫」
實作

撰寫程式碼

1

2

3

4

6

```
1 from sklearn.linear_model import LinearRegression
2
3 regressor = LinearRegression()
4 regressor.fit(X_train, Y_train)
5
6 Y_pred = regressor.predict(X_test)
```

1. 引入 **LinearRegression** 類別

2. 建造 **LinearRegression** 物件

3. 訓練 **LinearRegression** 模型

4. 用 **LinearRegression** 來預測答案

Y_test

	0
0	83088
1	122391
2	113812
3	109431
4	56642
5	55794
6	63218
7	66029
8	57081
9	98273
10	101302

↔

Y_pred

	0
0	75425.4
1	120910
2	101806
3	106355
4	53592.7
5	63599.4
6	62689.7
7	73606
8	64509
9	91799.9
10	99077.4

A 隨堂練習：用 Python 做線性迴歸



- 請先引入下列套件
 - from sklearn.linear_model import LinearRegression
- 建立一個 LinearRegression 物件
 - regressor = LinearRegression()
- 訓練「簡單線性迴歸」模型
 - regressor.fit(X_train, Y_train)
- 利用模型預測答案
 - Y_pred = regressor.predict(X_test)
- 使用「變數觀察面板」，比較「實際值」與「預測結果」
- 參考程式碼如下所示：

```
1 from sklearn.linear_model import LinearRegression
2
3 regressor = LinearRegression()
4 regressor.fit(X_train, Y_train)
5
6 Y_pred = regressor.predict(X_test)
```





AI

評估模型好壞

```
1 from sklearn.linear_model import LinearRegression
2
3 regressor = LinearRegression()
4 regressor.fit(X_train, Y_train)
5
6 Y_pred = regressor.predict(X_test)
```



以肉眼
觀察效能

Y_test		Y_pred	
	0		0
0	83088	0	75425.4
1	122391	1	120910
2	113812	2	101806
3	109431	3	106355
4	56642	4	53592.7
5	55794	5	63599.4
6	63218	6	62689.7
7	66029	7	73606
8	57081	8	64509
9	98273	9	91799.9
10	101302	10	99077.4



不公正、也不客觀

比較公正客觀的方法



- 決定係數 (Coefficient of Determination) : R^2



$$R_{res} = \sum_i (y_i - \hat{y}_i)^2$$

殘差平方和
(Residual Sum of Squares)
(越小，越好)



$$R_{tot} = \sum_i (y_i - \bar{y})^2$$

總平方和
(Total Sum of Squares)
(最爛的那條迴歸線)

$$R^2 = 1 - \frac{R_{res}}{R_{tot}}$$

$R_{res} \rightarrow 0$
時有最佳效能

$$1 - 0 = 1 = 100\%$$

時有最佳效能

$$R^2 = 0\% \sim 100\%$$



```
1 from sklearn.linear_model import LinearRegression
2
3 regressor = LinearRegression()
4 regressor.fit(X_train, Y_train)
5 Y_pred = regressor.predict(X_test)
6
7 R_Score = regressor.score(X_test, Y_test)
```

- **LinearRegression** 內建 **score()** 函數可算 R^2
- 輸入 **X_test** --> 自動算 **Y_pred** --> 與 **Y_test** 比較



- 請用下列程式碼，計算你的迴歸模型之 R²
 - regressor.score(X_test, Y_test)
- 參考程式碼如下所示（紅框部分）：

```
1 from sklearn.linear_model import LinearRegression
2
3 regressor = LinearRegression()
4 regressor.fit(X_train, Y_train)
5 Y_pred = regressor.predict(X_test)
6
7 R_Score = regressor.score(X_test, Y_test)
```





使用「快樂版」
實作


```
1 from sklearn.linear_model import LinearRegression
2
3 class SimpleRegressor:
4     __regressor = None
5
6     def __init__(self):
7         self.__regressor = LinearRegression()
8
9     @property
10    def regressor(self):
11        return self.__regressor
12
13    def fit(self, x_train, y_train):
14        self.__regressor.fit(x_train, y_train)
15        return self
16
17    def predict(self, x_test):
18        return self.__regressor.predict(x_test)
19
20    def r_score(self, x_test, y_test):
21        return self.__regressor.score(x_test, y_test)
```

建構函數
讀取函數
一般函數

引入必要套件

存放「簡單線性迴歸」模型本身

負責建造「簡單線性迴歸」模型

可傳回「簡單線性迴歸」模型，供其它場合使用

訓練模型，最後將整個物件傳回

使用訓練好的模型預測結果，並將結果傳回

計算「決定係數 R^2 」，評估模型效能



```
1 from HappyML.regression import SimpleRegressor
2
3 regressor = SimpleRegressor()
4 Y_pred = regressor.fit(X_train, Y_train).predict(X_test)
5 print("R-Squared Score:", regressor.r_score(X_test, Y_test))
```

1. 引入 **SimpleRegressor** 類別
2. 產生 **SimpleRegressor** 類別的物件
3. 訓練 + 預測
4. 印出 R^2 ，以評估模型好壞

- 「快樂版」的優點
 - 全程使用 **DataFrame**
 - 可將 **.fit()** 與 **.predict()** 串接使用



- 請用下列程式碼，引入講師製作好的套件：
 - `from HappyML.regression import SimpleRegressor`
- 用下列程式碼，建造 `SimpleRegressor` 的物件：
 - `regressor = SimpleRegressor()`
- 用下列程式碼，「訓練」+「預測」資料集：
 - `Y_pred = regressor.fit(X_train, Y_train).predict(X_test)`
- 用下列程式碼，計算 R2，以評估模型好壞：
 - `print("R-Squared Score:", regressor.r_score(X_test, Y_test))`
- 參考程式碼如下所示：

```
1 from HappyML.regression import SimpleRegressor
2
3 regressor = SimpleRegressor()
4 Y_pred = regressor.fit(X_train, Y_train).predict(X_test)
5 print("R-Squared Score:", regressor.r_score(X_test, Y_test))
```

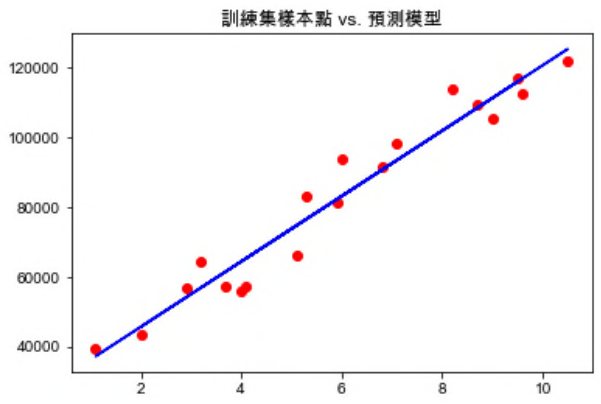




AI

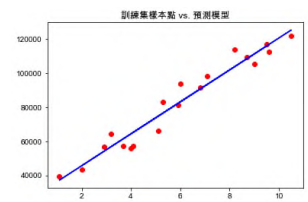
將結果視覺化

- 何謂結果視覺化？
- 為何要將結果視覺化？



將**樣本點**與**迴歸線**繪製出來，
好讓一般人了解訓練成果有多好！

95%



• 原始碼講解

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

def sample_model(sample_data=None, sample_color="red", model_data=None, model_color="blue", title="", xlabel="", ylabel="", font='Arial Unicode MS'):

for showing Chinese characters

plt.rcParams['font.sans-serif']=[font]

plt.rcParams['axes.unicode_minus'] = False

Draw for Sample Data with Scatter Chart

if sample_data != None:

plt.scatter(sample_data[0], sample_data[1], color=sample_color)

Draw for Model with line chart

if model_data != None:

plt.plot(model_data[0], model_data[1], color=model_color)

Draw for title, xlabel, ylabel

if sample_data!=None or model_data!=None:

plt.title(title)

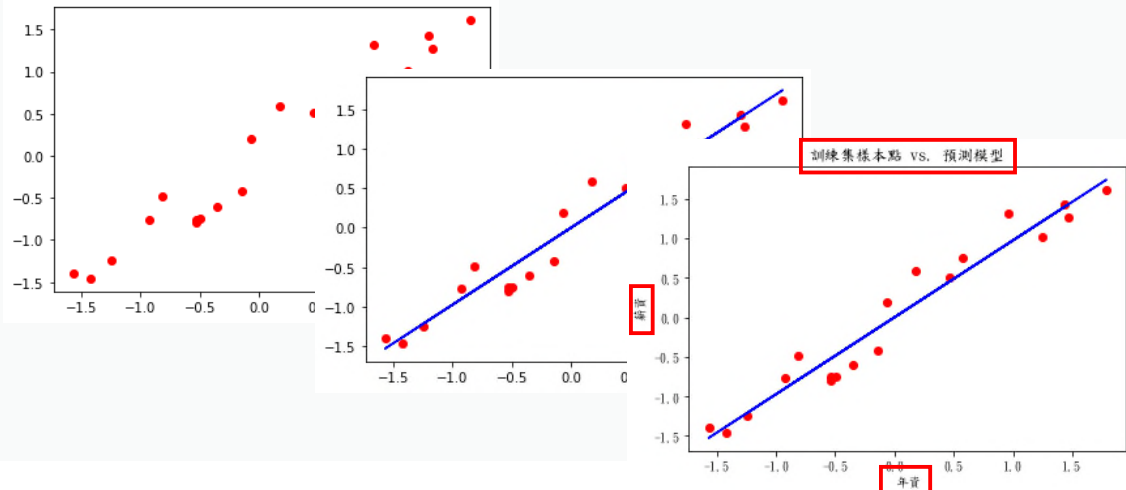
plt.xlabel(xlabel)

plt.ylabel(ylabel)

plt.show()

(X) X, X00, X000000

(O) X, X平方, X三次方線圖



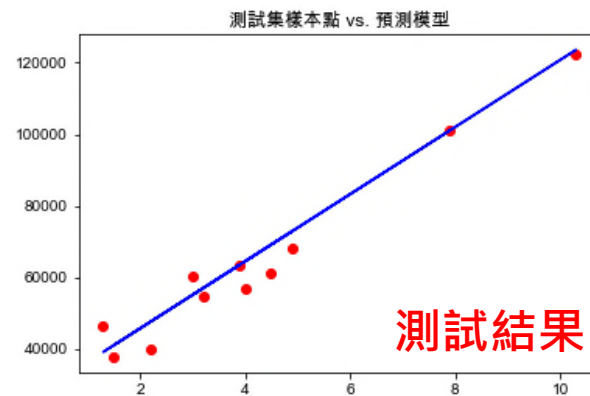
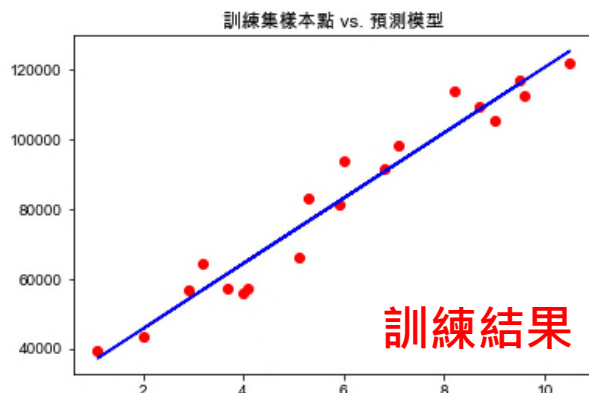
如何將結果「視覺化」



- 使用範例

```
1 from HappyML import model_drawer as md
2
3 sample_data=(X_train, Y_train)  樣本點
4 model_data=(X_train, regressor.predict(X_train))  模型
5 md.sample_model(sample_data=sample_data, model_data=model_data,  訓練結果
6                 title="訓練集樣本點 vs. 預測模型", font="DFKai-sb")
7 md.sample_model(sample_data=(X_test, Y_test), model_data=(X_test, Y_pred),  測試結果
8                 title="測試集樣本點 vs. 預測模型", font="DFKai-sb")
```

- 執行結果



A 隨堂練習：樣本點、模型「視覺化」



- 請先用下列程式碼，引入講師自製的套件：
 - from **HappyML** import **model_drawer** as md
- 將「**特徵縮放**」程式碼取消註解，使之生效。讓 X 與 Y 的比例尺差異不至於過大（若 X 與 Y 數量級差異不大，則非必要，不過做了會比較保險）
- 撰寫下列程式碼，以便繪製圖形：

```
1 from HappyML import model_drawer as md
2
3 sample_data=(X_train, Y_train)
4 model_data=(X_train, regressor.predict(X_train))
5 md.sample_model(sample_data=sample_data, model_data=model_data,
6                 title="訓練集樣本點 vs. 預測模型", font="DFKai-sb")
7 md.sample_model(sample_data=(X_test, Y_test), model_data=(X_test, Y_pred),
8                 title="測試集樣本點 vs. 預測模型", font="DFKai-sb")
```



課後作業：台灣學童身高、體重評估



- 要求
 - 請各位下載下面兩份資料集
 - 學生身高平均值(6歲-15歲)：<https://bit.ly/3XFxGVk> (資料來源：<https://data.gov.tw/dataset/6283>)
 - 學生體重平均值(6歲-15歲)：<https://bit.ly/3kJcjUC> (資料來源：<https://data.gov.tw/dataset/6229>)
 - 採用資料集內的自變數 vs. 應變數，製作四個簡單迴歸器
 - 年齡 vs. 男生身高、年齡 vs. 女生身高、年齡 vs. 男生體重、年齡 vs. 女生體重
 - 讓使用者輸入「性別、年齡、身高、體重」，並與全台灣同年齡、同性別者比較身高、體重。如下所示：

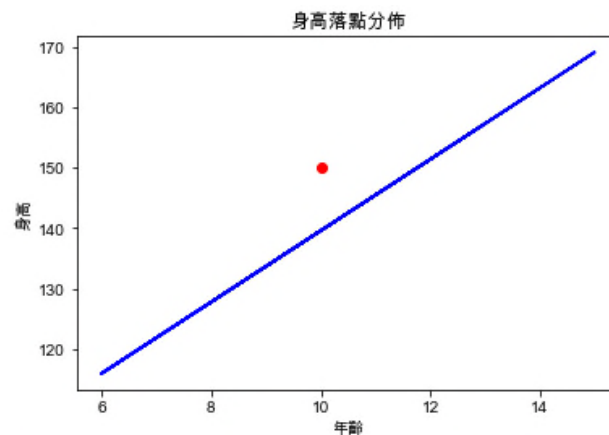
台灣 6~15 歲學童身高、體重評估系統

請輸入您的性別 (1.男 2.女) : 1

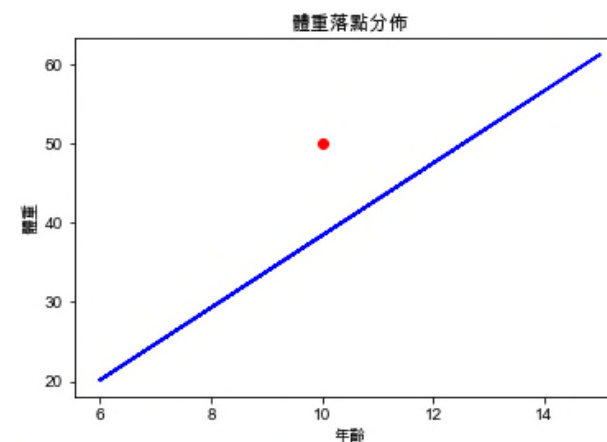
請輸入您的年齡 (6-15) : 10

請輸入您的身高 (cm) : 150

請輸入您的體重 (kg) : 50



10 歲男生平均身高為 139.60 公分，您的身高為 150 公分



10 歲男生平均體重為 38.42 公斤，您的體重為 50 公斤



- 提示
 - 取資料集中的下列欄位：

身高

學年度	年齡	總計	男	女
96	6	117.1	117.6	116.6
96	7	121.4	121.9	120.9
96	8	127.1	127.5	126.7
96	9	132.8	132.9	132.6
96	10	138.8	138.5	139.3

體重

學年度	年齡	總計	男	女
96	6	22.4	23	21.8
96	7	24.6	25.3	23.9
96	8	27.9	28.7	27
96	9	31.8	32.8	30.8
96	10	36.3	37.2	35.3

- 訓練出四個「簡單線性迴歸器」
 - regressor = [[SimpleRegressor(), SimpleRegressor()], [SimpleRegressor(), SimpleRegressor()]]
 - regressor[0][0] --> 年齡 vs. 男生身高
 - regressor[0][1] --> 年齡 vs. 女生身高
 - regressor[1][0] --> 年齡 vs. 男生體重
 - regressor[1][1] --> 年齡 vs. 女生體重

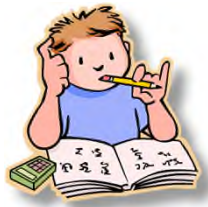




課後作業：台灣學童身高、體重評估



- 提示
 - 讓使用者輸入下列資料
 - `user_gender = eval(input("請輸入您的性別 (1.男 2.女) : ")) - 1`
 - `user_age = eval(input("請輸入您的年齡 (6-15) : "))`
 - `user_height = eval(input("請輸入您的身高 (cm) : "))`
 - `user_weight = eval(input("請輸入您的體重 (kg) : "))`
 - 可用下列方法，取得使用者同齡之平均身高、體重
 - `h_avg = regressor[0][user_gender].predict(x_test=pd.DataFrame([[user_age]]).iloc[0, 0])`
 - `w_avg = regressor[1][user_gender].predict(x_test=pd.DataFrame([[user_age]]).iloc[0, 0])`



- 定義
 - 可用 $f(x) = c + mx$ 擬合出模型的問題
 - 通常只有一個自變數
- 「簡單線性迴歸」的 Python 類別
 - `sklearn.linear_model.LinearRegression`
 - `.fit(X_train, Y_train)`：擬合（訓練）模型
 - `.predict(X_test)`：利用模型，預測答案
- 評判「簡單線性迴歸」模型的好壞
 - 決定係數 $R^2 = 1 - R_{\text{res}}/R_{\text{tot}}$
 - `LinearRegression` 內的 `.score(X_test, Y_test)` 可以幫你計算 R^2

